$\mathcal{H}$-matrix Preconditioning for the Time-Harmonic Electromagnetic Discontinuous Galerkin Method

by

Hamidreza Bagherli

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg

Declaration of Authorship

I, Hamidreza Bagherli, declare that this thesis titled, '$\mathcal{H}$-matrix Preconditioning for the Time-Harmonic Electromagnetic Discontinuous Galerkin Method' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________________________

Date: ____________________________________

"The greatest education in the world is watching the masters at work."

Michael Jackson

UNIVERSITY OF MANITOBA

Abstract

Faculty of Engineering
Department of Electrical and Computer Engineering

Master of Science

by Hamidreza Bagherli

Hierarchical matrices, or $\mathcal{H}$-matrices are an error-controllable framework that permit efficient inversion and decomposition of matrices arising in time harmonic electromagnetics applications. This work evaluates the capabilities of $\mathcal{H}$-matrices for preconditioning iterative solutions to the time-harmonic discontinuous Galerkin method. Particular focus is given to exact radiating boundary conditions in the discontinuous Galerkin formulation. In order to ensure a deep understanding of $\mathcal{H}$-matrix theory and operations, an $\mathcal{H}$-matrix framework has been developed. Performance of $\mathcal{H}$-LU decompositions as error-controllable preconditioners is demonstrated, showing the desired time, memory, and accuracy scaling.

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

First of all, I would like to express my sincere gratitude to my advisor Dr. Ian Jeffrey for the endless support throughout my research, for his patience, motivation, and immense knowledge. It has been an honor, and I could not have imagined having a better advisor, mentor and friend over the years. I have been extremely lucky to have him as an advisor and I thank him for helping me to become the researcher and programmer I am today. I attribute the level of my Master's degree to his encouragement and effort. His careful editing contributed enormously to the production of this thesis.

I would like to thank Dr. Joe LoVetri for access to the Electromagnetic Imaging Lab.

My sincere thanks also goes to Dr. Vladimir Okhmatovski and Dr. Jason Morrison, for generously offering their time to review this work.

I offer my thanks to Dr. Shaahin Filizadeh and Amy Dario for their generous support and guidance.

I would like to give a special thanks to my director and colleague at DASCH, Dwight Woods and Shawn Schwark for their support, friendship, flexibility and encouragement throughout the last year.

I would like to thank Randy Holyk and Jim Hounslow from Extended Education for their endless support during my work at the University of Manitoba.

I would like to extend a special thanks to all my loyal friends in Winnipeg for their encouragement and emotional support throughout this study.

Lastly, and most importantly, I would like to thank my parents Mehri and Ghader, my brother Alireza, and sister Maryam, for providing me with unfailing support and continuous encouragement throughout my years of study and research, and my life in general. This accomplishment would not have been possible without them. Thank you

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**
- 1.1 Motivation . . . 3
- 1.2 Goals and Scope . . . 4
- 1.3 Outline . . . 5
- 1.4 Contributions . . . 6

2 Relevant Electromagnetic Theory **8**
- 2.1 Maxwell's Curl Equations . . . 8
- 2.2 Vector Wave Equation . . . 11
- 2.3 Transverse Magnetic Problems . . . 12
- 2.4 Boundary Conditions . . . 12
  - 2.4.1 Material Interfaces . . . 12
  - 2.4.2 Perfect Conductors . . . 13
  - 2.4.3 Radiation Boundary Conditions . . . 14
- 2.5 Huygens' Principle and Kirchhoff's Integral . . . 14
- 2.6 Chapter Summary . . . 15

3 The Time-Harmonic Discontinuous Galerkin Method **16**
- 3.1 Overview . . . 17
- 3.2 Discretization of The Problem Domain . . . 17
- 3.3 Nodal Basis Expansion . . . 19
- 3.4 Residual Testing . . . 20

3.5 Constructing the DGM System of Equations 22
3.6 Absorbing Boundary Conditions 26
3.6.1 Chapter Summary 28

4 Preconditioned GMRES 29
4.1 Direct Solutions Methods 30
4.2 LU Decomposition 30
4.3 GMRES 32
4.4 Preconditioned GMRES 34
4.4.1 Choosing a Preconditioner 35
4.5 Chapter Summary 36

5 $\mathcal{H}$-Matrices 37
5.1 Introduction 37
5.1.1 Why and How is Compression Possible? 38
5.1.2 Exploiting a Hierarchical Tree Structure 39
5.2 Building a Hierarchical Matrix 40
5.2.1 Constructing a Cluster Tree 41
5.2.2 Building a Block Cluster Tree 43
5.3 Rank Compression and Tree Pruning 44
5.3.1 $\mathcal{R}_k$-Matrices 45
5.3.2 $\mathcal{R}_k$-Matrix Approximations 46
5.3.3 Re-compression of an $\mathcal{R}_k$-matrix 48
5.3.4 Admissible Condition 49
5.3.5 Final construction of an $\mathcal{H}$-Matrix 49
5.4 Basic $\mathcal{H}$-matrix Operations 51
5.4.1 $\mathcal{R}_k$-matrix Addition 51
5.4.2 $\mathcal{H}$-matrix-vector Product 52
5.4.3 $\mathcal{H}$-matrix-$\mathcal{H}$-matrix Products 54
5.4.3.1 Multiplying $\mathcal{H}$-matrices 55
5.4.3.2 Adding $\mathcal{H}$-matrices 56
5.5 $\mathcal{H}$-Matrix Coarsening 57
5.6 Hierarchical Inversion 58
5.7 Hierarchical LU Decomposition 59
5.7.1 Obtaining Hierarchical L and U Factors 60
5.7.2 Comments on Pivoting Strategies 61
5.7.3 Hierarchical Triangular Solvers 62
5.8 Chapter Summary 63

6 Numerical Results 64
6.1 DGM Problem Specifications 64
6.2 Iterative Solution Specifications 66
6.3 $\mathcal{H}$-matrix Tolerance Testing 67
6.4 h-Refinement 70
6.5 p-Refinement 71

6.6 Frequency Scaling - The Effect of Problem Rank 74
6.7 Chapter Summary 76

7 Conclusions and Future Work **78**

Bibliography **80**

List of Tables

6.1 $\mathcal{H}$-matrix memory scaling as a function of tolerance. 68
6.2 $\mathcal{H}$-matrix accuracy as a function of tolerance. 68
6.3 $\mathcal{H}$-matrix computational time scaling as a function of tolerance. . 69
6.4 $\mathcal{H}$-matrix memory scaling as a function of element size. 70
6.5 $\mathcal{H}$-matrix accuracy as a function of element size. 70
6.6 $\mathcal{H}$-matrix computational time scaling as a function of element size. 71
6.7 $\mathcal{H}$-matrix memory scaling as a function of polynomial order. . . . 72
6.8 $\mathcal{H}$-matrix accuracy as a function of polynomial order. 72
6.9 $\mathcal{H}$-matrix computational time scaling as a function of polynomial order. 73
6.10 $\mathcal{H}$-matrix memory scaling as a function of frequency. 74
6.11 $\mathcal{H}$-matrix accuracy as a function of frequency. 75
6.12 $\mathcal{H}$-matrix computational time scaling as a function of frequency. . 76

List of Figures

1.1 Conceptual acceleration of field problems with $\mathcal{H}$-matrices. 4

2.1 Interdependence of electromagnetic fields 9
2.2 An example of a material boundary and the tangential fields. . . . 13

3.1 An example of mesh generation. 18
3.2 Geometries for imposing Silver-Müller boundary conditions and exact radiating boundary conditions. 27
3.3 Matrix entries introduced by exact radiating boundary conditions. 28

5.1 An illustration of source compression of light from distant sources. 39
5.2 An abstract representation of a tree structure 40
5.3 Construction of a 3-level cluster tree. 42
5.4 An example of an ORB constructed cluster tree's affect on element ordering. 43
5.5 An illustration of a block cluster tree based on two cluster trees. . 44
5.6 Illustration of an $\mathcal{H}$-matrix structure. 50
5.7 An illustration of $\mathcal{H}$-matrix agglomeration or coarsening. 57
5.8 $\mathcal{H}$-LU and the hierarchical factors. 60

6.1 The electromagnetic geometry for numerical testing. 65
6.2 DGM field solution simulation results for various frequencies . . 75

This thesis is dedicated to my beloved parents, who have been my source of inspirition and gave me strength when I thought of giving up.

To my brother, who continually provides his moral, spiritual, emotional, and financial support.

To my sister, whose love for me knew no bounds.

To my niece Nika, my nephews Kian, Sebastian, and Benjamin, my brother-in-law Shahram, and my sister-in-law Rebecca for all their love and support.

And lastly, I dedicate this work to my academic advisor and friend Dr. Ian Jeffrey for his extraordinary human qualities.

Chapter 1

Introduction

Computational eletromagnetics (CEM) is a field of applied mathematics, physics, and engineering that uses numerical techniques for determining the interactions between eletromagnetic fields and the physical medium in which the fields propagate. Pioneering efforts in numerical methods for electromagnetics problems can be dated back to the 1940s [1], and since that time CEM has grown into a vibrant research area. The ability to simulate electromagnetic interactions has many applications, including expedient and cost efficient optimized designs of electronic systems [2–6], and non-invasive electromagnetic imaging for biomedical and agricultural applications [7–9].

Today, the field of CEM provides many approaches and formulations, each with strengths and weaknesses in terms of the range of applications and computational performance. Field problems can be solved in either the time-domain, where transient effects are present and critical, or in the frequency-domain, where steady-state field behaviour is assumed. Formulations may also be based on either partial-differential-equation (PDE) formulations or integral equation (IE) formulations. Popular methods include finite difference methods [10], finite element methods [11], finite volume methods [12], integral equation formulations [1], pseudo-spectral methods [13] and discontinuous Galerkin methods [14].

Numerical solutions to time-harmonic problems generally lead to a system of linear equations that must be solved to determine approximations to the electromagnetic fields. Researchers and designers of electronic systems continually seek the ability to solve problems faster, or to solve more complicated (larger) problems that will help improve their products, servcies, and understanding of electromagnetic behaviour. While increased computational power and parallelism offer significant benefits [15], solution methods are ultimately hampered by the cost complexity associated with solving large systems of equations [16]. Consequently there is steady work on novel formulations and fast algorithms (acceleration methods) that aim to cut the cost of computations. Relevant fast algorithms include the multi-level fast multipole method (MLFMM) [2], pre-corrected fast Fourier transform (FFT) approaches [17], adaptive cross approximation (ACA) and its variants [4], domain decomposition techniques [3], multi-grid methods [18], and the relatively new approaches of hierarchical matrices (commonly denoted as $\mathcal{H}$-matrices) [19] [20] and hierarchically semi-separable matrices [21].

The diversity of formulations and acceleration schemes available in CEM is staggering, and developing skills in, and/or advancing, this area requires focus. This thesis focuses on the application of $\mathcal{H}$-matrix acceleration to discontinuous Galerkin method formulations for time-harmonic problems. Motivation for this choice, the goals and scope of study, an outline of the thesis, and a summary of contributions are provided in this chapter.

1.1 Motivation

The University of Manitoba's Electromagnetic Imaging Lab (EIL) designs electromagnetic imaging systems and algorithms, focusing on biomedical [9] and agricultural [7] applications. While a precise theory of electromagnetic imaging is beyond the scope of this thesis, we can briefly summarize the technology as follows: i) a target is interrogated with electromagnetic sources, ii) the resulting electromagnetic fields are sampled external to the target, and iii) an algorithm attempts to determine the electrical properties of the target from the field measurements. The imaging algorithms are optimization methods: estimates of the target parameters are iteratively updated based on comparisions of field simulations for a target estimate with the measured field data. The capabilities and computational cost of electromangetic imaging is tied to the choice of *forward solver*, that is to the method selected for performing the field simulations.

Recently, the EIL has expanded their imaging capabilities by implementing algorithms based on the time-harmonic discontinuous Galerkin method (DGM) [22]. This forward solver formulation enables imaging of electric and magnetic targets, simulation and reconstruction of high-order target properties, flexibile use of inhomogeneous backgrounds, and electric and/or magnetic field measurements. For modest problem sizes, and single-frequency imaging applications, it is possible, though expensive, to solve the DGM linear system of equations directly without any acceleration. However, to improve imaging speeds and capabilities, there is a need to accelerate the DGM forward solver solution. While many acceleration schemes are possible, this work focuses on acceleration of a DGM forward solver by means of hierarchical matrices ($\mathcal{H}$-matrices), a relatively new fast-algorithm framework for accelerating the solution to linear systems of equations (Figure 1.1) [19] [20]. The motivation for this choice of acceleration scheme is primarily driven by curiousity and flexibility; $\mathcal{H}$-matrix

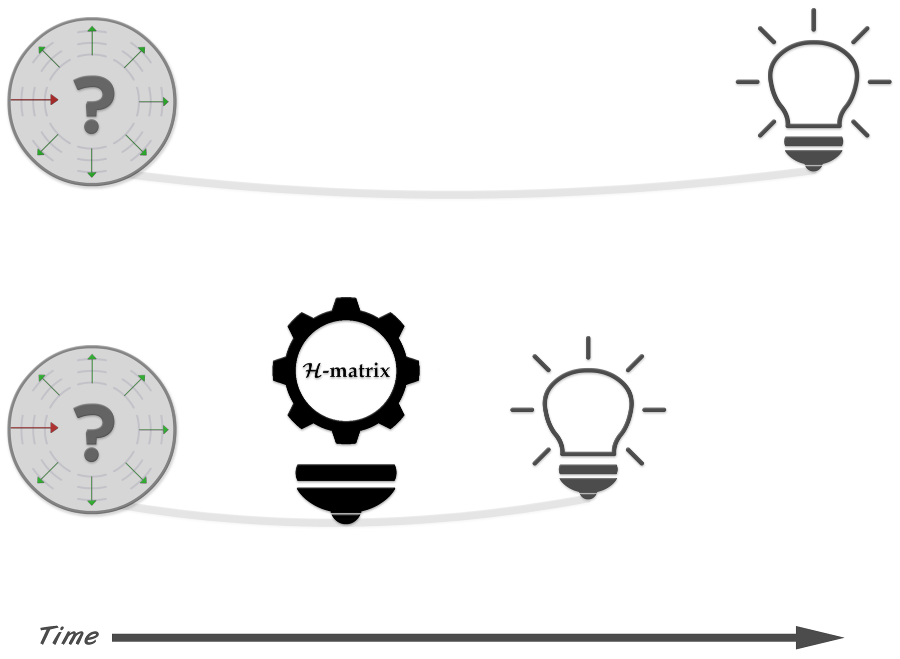


FIGURE 1.1: Conceptual acceleration of field problems with $\mathcal{H}$-matrices. The $\mathcal{H}$-matrix framework can improve the performance of storage requirements for solving systems of equations arising from field problems.

theory can be applied, without extensive modifications, to many different time-harmonic field solvers.

1.2 Goals and Scope

The goal of the work presented in this thesis was to understand, implement, and study the performance of $\mathcal{H}$-matrix acceleration for solving the system of equations arising from a time-harmonic DGM formulation of Maxwell's equations. As a result, this thesis serves as a survey of the overall $\mathcal{H}$-matrix framework, a guide towards implementation, and demonstrates performance for DGM acceleration.

The scope of this thesis is:

- $\mathcal{H}$-matrices are applied to the DGM solution of the two-dimensional (2D) transverse magnetic (TM) time-harmonic electric vector wave equation.
- Special attention is paid to the effects of *exact radiating boundary conditions* (ERBCs) in the DGM formulation as they relate to changes in the discrete forward operator.
- MATLAB [23] is used to implement an $\mathcal{H}$-matrix library of routines. The choice of implementation is consistent with the use of a pre-existing 2D TM DGM solver and the desire to produce a software solution that can be used as a teaching tool, i.e., MATLAB offers a path towards an easy-to-follow implementation that is not convoluted by dynamic memory management or complicated BLAS [24] routine calls.
- The developed $\mathcal{H}$-matrix software library is used to precondition iterative solutions to the resulting DGM system of equations.

Limiting the scope of this work to 2D problems does not greatly affect the overall goal of producing an $\mathcal{H}$-matrix code library as the $\mathcal{H}$-matrix approach is largely independent of the problem dimensions. It should be noted that commercial $\mathcal{H}$-matrix libraries exist [25], but the development of an in-house code is consistent with the goal of an in-depth understanding of $\mathcal{H}$-matrices and their implementation.

1.3 Outline

The remainder of this thesis is structured as follows:

- Chapter 2 summarizes the relevant electromagnetic theory, namely Maxwell's equations, the vector wave equation, and boundary conditions.

- Chapter 3 introduces the Discontinuous Galerkin Method (DGM) for the vector wave equation, with sufficient details provided to reach the system of linear equations whose solution provides approximations to the fields. Special consideration is given to ERBCs.
- Chapter 4 reviews some common methods for solving the linear system of equations, including direct solutions (specifically LU decomposition), and iterative techniques (specifically the Generalized Minimum Residual Method, GMRES). We end this chapter by emphasizing the need for a good error controllable preconditioner to improve iterative convergence.
- Chapter 5 presents the $\mathcal{H}$-matrix framework, structure, and arithmetic that enables approximate LU decompositions of the system matrix that are suitable to precoditioning GMRES.
- Chapter 6 consists of a variety of numerical results testing the performance of $\mathcal{H}$-matrices related to a number of solution parameters.
- Finally, Chapter 7 concludes the thesis and provides suggestions for future development.

1.4 Contributions

While $\mathcal{H}$-matrices are not a new theory, and have been used for solving many CEM problems in the literature [26], to the best of our knowledge this is the first time that $\mathcal{H}$-matrices have been applied to DGM based systems. Moreover, exploiting $\mathcal{H}$-matrices for ERBC-enabled DGM formulations appears to be a novel synthesis of numerical techniques that has promise for solving large-scale complicated electromagnetic interaction problems in an expedient way. From the perspective of the Electromagnetic Imaging Lab at the University of

Manitoba, this project serves a major contribution in terms of an understanding of $\mathcal{H}$-matrix theory and the development of a MATLAB code library that has immediate applications to the MATLAB imaging codes previously developed in the lab. Full-scale 3D codes used in the EIL are primarily C/C++ [27] and the MATLAB $\mathcal{H}$-matrix implementation will serve as a guide to implementation in other programming languages. This work on $\mathcal{H}$-matrix accleration of ERBCs for DGM problems has resulted in a conference publication "$\mathcal{H}$-Matrix Compression of Discontinuous Galerkin Method Exact Radiating Boundary Conditions" [28].

Chapter 2

Relevant Electromagnetic Theory

In this chapter, theory relevant to solving an electromagnetic scattering problem using the vector wave equation is presented. Maxwell's curl equations for isotropic linear media are presented and used to derive both a scattered-field formulation and the vector wave equation. Electromagnetic boundary conditions are also summarized. A time-harmonic problem, at a frequency f with associated radial frequency $\omega = 2\pi f$, and with an $e^{j\omega t}$ time-dependence for time t is assumed, where $j = \sqrt{-1}$ is the imaginary unit. The general presentation is for 3D problems, but the restriction to 2D transverse magnetic (TM) problems is discussed [29].

2.1 Maxwell's Curl Equations

We consider solving a time-harmonic field problem in a domain Ω that is governed by the interaction of impressed electric sources $\vec{J}(\vec{x})$, magnetic sources $\vec{M}(\vec{x})$ and a physical medium governed by isotropic electrical consitutive parameters, namely the complex dielectric $\varepsilon(\vec{x})$ and permeability $\mu(\vec{x})$. Here, $\vec{x} = x\hat{x} + y\hat{y} + z\hat{z}$ is the Cartesian position vector. For linear isotropic media, the total

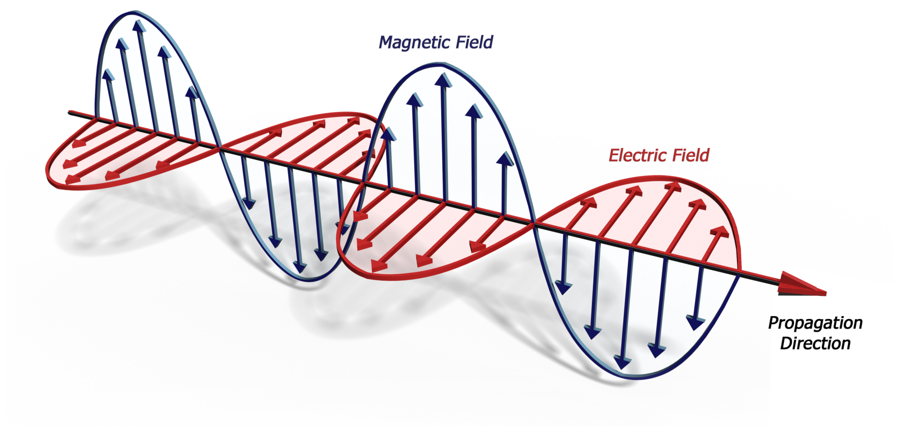


FIGURE 2.1: Interdependence of electromagnetic fields. An electromagnetic plane wave includes electric field ($\vec{E}$) and magnetic field ($\vec{H}$) that are coupled through Maxwell's equations.

fields resulting from this interaction of sources and medium satisfy Maxwell's curl equations[1]:

$$\begin{aligned} j\omega\varepsilon(\vec{x})\vec{E}^{tot}(\vec{x}) - \nabla \times \vec{H}^{tot}(\vec{x}) &= -\vec{J}(\vec{x}) \\ j\omega\mu(\vec{x})\vec{H}^{tot}(\vec{x}) + \nabla \times \vec{E}^{tot}(\vec{x}) &= -\vec{M}(\vec{x}) \end{aligned} \tag{2.1}$$

at any point $\vec{x} \in \Omega$ where $\vec{E}^{tot}$ is the total electric field and $\vec{H}^{tot}$ is the total magnetic field. Maxwell's curl equations are a system of coupled PDEs for the fields; at any point in space $\vec{E}^{tot}$ is related to $\vec{H}^{tot}$ and vice-versa, as illustrated in Figure 2.1. Solving these PDEs requires the specification of appropriate boundary conditions, discussed further in Section 2.4.

To isolate the effects of the medium $\varepsilon(\vec{x})$ and $\mu(\vec{x})$, scattered field formulations are beneficial. We assume without any loss of generality that *incident fields* $\vec{E}^{inc}(\vec{x})$ and $\vec{H}^{inc}(\vec{x})$ are supported by the same sources as the total fields but propagate in a *background medium* having a permittivity $\varepsilon_b(\vec{x})$ and permeability

[1]For linear media there is an assumed relationship between electric flux density $\vec{D}$, magnetic flux density $\vec{B}$, and the fields $\vec{E}$ and $\vec{H}$: $\vec{D} = \varepsilon\vec{E}$ and $\vec{B} = \mu\vec{H}$. Consequently we will not require the divergence equations.

$\mu_b(\vec{x})$:

$$\begin{aligned} j\omega\varepsilon_b(\vec{r})\vec{E}^{inc} - \nabla \times \vec{H}^{inc} &= -\vec{J} \\ j\omega\mu_b(\vec{r})\vec{H}^{inc} + \nabla \times \vec{E}^{inc} &= -\vec{M} \end{aligned} \tag{2.2}$$

One of the benefits of a scattered-field formulation is that the incident fields, for a variety of problems, are known analytically. For example, in homogeneous media, where $\varepsilon_b(\vec{x}) = \varepsilon_b$ and $\mu_b(\vec{x}) = \mu_b$ are independent of position, both plane-wave and point-source incident fields have closed forms. The *scattered fields*, i.e., field arising from differences between the medium $\varepsilon(\vec{x})$ and $\mu(\vec{x})$ and the assumed background $\varepsilon_b(\vec{x})$ and $\mu_b(\vec{x})$ are defined to satisfy:

$$\begin{aligned} \vec{E}^{tot} &= \vec{E}^{inc} + \vec{E}^{sct} \\ \vec{H}^{tot} &= \vec{H}^{inc} + \vec{H}^{sct}. \end{aligned} \tag{2.3}$$

From (2.1), and (2.2) it is straightforward to show that the scattered fields satisfy:

$$\begin{aligned} j\omega\varepsilon\vec{E}^{sct} - \nabla \times \vec{H}^{sct} &= -\vec{J}^{sct} \\ j\omega\mu\vec{H}^{sct} + \nabla \times \vec{E}^{sct} &= -\vec{M}^{sct} \end{aligned} \tag{2.4}$$

where $\vec{J}^{sct} = -j\omega(\varepsilon - \varepsilon_b)\vec{E}^{inc}$ and $\vec{M}^{sct} = -j\omega(\mu - \mu_b)\vec{H}^{inc}$.

Note that in the scattered-field equations above, explicit dependence on $\vec{x}$ has been dropped. This convention will be followed throughout this thesis unless additional clarity is required.

2.2 Vector Wave Equation

While Maxwell's scattered-field curl equations (2.4) can be numerically solved for approximations to the fields for a particular scattering problem, such solutions generally require solving for six field components, namely the three scalar components of $\vec{E}^{sct}$ and the three scalar components of $\vec{H}^{sct}$. To circumvent this additional computational cost it is beneficial to consider the vector wave equations.

Solving for the curl of $\vec{H}^{sct}$ in the second equation of (2.4) and substituting into the first equation of (2.4) leads to the *electric vector wave equation*:

$$(j\omega)^2\varepsilon\vec{E}^{sct} + \nabla \times (\frac{1}{\mu}\nabla \times \vec{E}^{sct}) = -j\omega\vec{J}^{sct} - \nabla \times (\frac{1}{\mu}\vec{M}^{sct}). \qquad (2.5)$$

The magnetic vector wave equation can be obtained by duality [2]:

$$(j\omega)^2\mu\vec{H}^{sct} + \nabla \times (\frac{1}{\varepsilon}\nabla \times \vec{H}^{sct}) = -j\omega\vec{M}^{sct} + \nabla \times (\frac{1}{\varepsilon}\vec{J}^{sct}). \qquad (2.6)$$

The benefits of the vector wave equations is that they operate on only a single vector field; the solution of a 3D vector wave equation requires discretizing three field components. However, the additional derivatives introduced by the vector wave equation must be handled appropriately. It should be noted that the magnetic fields are recoverable from the electric vector wave equation by means of the local operation

$$\vec{H}^{sct} = \frac{-\vec{M}^{sct}}{j\omega\mu} - \frac{\nabla \times \vec{E}^{sct}}{j\omega\mu}. \qquad (2.7)$$

[2]Here the duality substitution used is:

$$\vec{E} \to \vec{H}, \quad \vec{H} \to -\vec{E}, \quad \varepsilon(\vec{r}) \to \mu(\vec{r}), \quad \mu(\vec{r}) \to \varepsilon(\vec{r}), \quad \vec{J} \to \vec{M}, \quad \vec{M} \to -\vec{J}$$

2.3 Transverse Magnetic Problems

Simplifications to Maxwell's equations (or correspondingly the vector wave equation) occur when a problem geometry is invariant in one or more dimensions. A 2D problem arises when invariance exists in one dimension. If we assume that the problem is invariant in the z-direction, then any z-derivative of the fields or sources becomes zero. In this configuration, Maxwell's equations decouple into two independent problems, one involving E_z, H_x, and H_y, and the other involving E_x, E_y and H_z. The former case is referred to as a TM (transverse magnetic) problem and the latter as a TE (transverse electric) problem. The reduced dimensionality of 2D problems results in a reduction in the number of equations that must be solved. Throughout this thesis we consider a TM problem and the vector wave equation, in which case only the z-component of the electric field is involved in the partial differential operator.

2.4 Boundary Conditions

Regardless of the form that the PDEs representing Maxwell's equations take, boundary conditions are required to solve the mathematical problem. Briefly, the relevant boundary conditions for solving Maxwell's curl equations, or the vector wave equation, are summarized in the following subsections.

2.4.1 Material Interfaces

A material interface is denoted by a discontinuity in the constitutive parameters $\varepsilon(\vec{x})$ and/or $\mu(\vec{x})$. If the discontinuity is characterized as being between two regions, the first having parameters $\varepsilon_1(\vec{x})$ and $\mu_1(\vec{x})$, and the other having

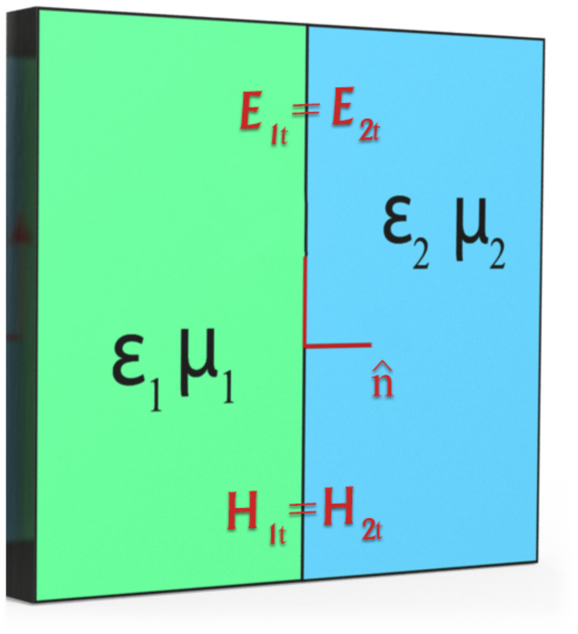


FIGURE 2.2: An example of a material boundary and its tangential field.

parameters $\varepsilon_2(\vec{x})$ and $\mu_2(\vec{x})$ then, the tangential fields are continuous across the interface. That is [30],

$$\hat{n} \times \vec{E}_1 = \hat{n} \times \vec{E}_2, \qquad \hat{n} \times \vec{H}_1 = \hat{n} \times \vec{H}_2, \tag{2.8}$$

where we note that for a unit normal $\hat{n}$ directed from medium 1 to medium 2, shown in Figure 2.2 the tangential electric field is actually given by $-\hat{n} \times \hat{n} \times \vec{E}$, but that enforcing continuous tangential fields is equivalent to enforcing (2.8).

2.4.2 Perfect Conductors

Perfect electric conductors (PECs), and/or perfect magnetic conductors (PMCs) are useful approximations to highly conductive materials [31, 32]. Both PECs and PMCs permit approximating a volumetric effect from a good conductor with an easily imposed boundary condition. It can be shown that at a PEC boundary:

$$\hat{n} \times \vec{E}^{tot} = 0, \qquad \rightarrow \qquad \hat{n} \times \vec{E}^{sct} = -\hat{n} \times \vec{E}^{inc} \tag{2.9}$$

while at a PMC boundary:

$$\hat{n} \times \vec{H}^{tot} = 0, \qquad \rightarrow \qquad \hat{n} \times \vec{H}^{sct} = -\hat{n} \times \vec{H}^{inc}. \tag{2.10}$$

Once again, $\hat{n}$ is the unit normal to the boundary.

2.4.3 Radiation Boundary Conditions

A final boundary condition of note, as it relates to numerical solutions to Maxwell's equations, is field behaviour at very large distances from sources and scatterers, conditions commonly referred to as *Sommerfeld radiation conditions* [33]. Numerical techniques that must approximate infinite unbounded domains with finite bounded regions require imposing a radiation condition in some form.

2.5 Huygens' Principle and Kirchhoff's Integral

In closing the theory presented in this chapter we briefly introduce the concept of a Huygens' surface. It can be shown that given a closed surface $\Gamma \in \Omega$ that contains all electromagnetic sources, which implies it contains all scatterers of a scattered-field problem, the fields on Γ can be used to determine the fields anywhere outside of Γ. Mathematically, this Huygens' surface results in an integral, also referred to as Kirchhoff's integral, that is given by [34][22]:

$$\begin{aligned}\vec{E}(\vec{x}) = \int\limits_{\Gamma} \Big(& - j\omega\mu_b[\vec{n}(\vec{x}') \times \vec{H}(\vec{x}')]G_b(\vec{x},\vec{x}') + [\vec{n}(\vec{x}') \cdot \vec{E}(\vec{x}')]\nabla' G(\vec{x},\vec{x}') \\ & + [\vec{n}(\vec{x}') \times \vec{E}(\vec{x}')] \times \nabla' G_b(\vec{x},\vec{x}') \Big) ds', \qquad \vec{x}' \in \Gamma,\end{aligned} \tag{2.11}$$

where $G_b(\vec{x}, \vec{x}\,')$ is the Green's function for the background medium characterized by $\varepsilon_b(\vec{x})$ and $\mu_b(\vec{x})$ external to the integration surface Γ. If the medium external to Γ is homogeneous, then this Green's function is known analytically.

Equation (2.11) provides a way of computing the electric fields anywhere outside of the surface Γ and will be exploited in developing radiation boundary conditions for DGM in the next chapter. Note that the presence of the magnetic fields in (2.11) can be replaced with electric fields by means of (2.7).

2.6 Chapter Summary

In this chapter the relevant electromagnetic theory for the electric vector wave equation has been presented. The goal of the next chapter is to formulate a numerical solution to scattered field problems.

Chapter 3

The Time-Harmonic Discontinuous Galerkin Method

This chapter presents the Discontinuous Galerkin Method (DGM) discretization of a generic electromagnetic scattering problem. The goal of the exposition is to provide a sense of the types of matrices that arise from DGM formulations, focused primarily on the structure of the matrix entries. As DGM is a high-order method, emphasis is placed both on the effects of order selection (p-refinement) and mesh density (h-refinement). Specifically, DGM is presented for discretizing the vector wave equation, but the concepts are the same as those that would be used to directly discretize Maxwell's curl equations, and are similar to the concepts used in more advanced formulations such as the Hybridizable Discontinuous Galerkin Method [35]. The presentation closely follows the work of Hesthaven and Warburton [14] and the discretization of the vector wave equation presented in [22]. The presentation in this chapter is for 3D problems, but as we ultimately end up solving 2D TM problems, restriction to the TM case is provided.

3.1 Overview

The Discontinuous Galerkin Method can be summarized succinctly as follows. We assume we solving an electromagnetic scattering problem in a domain Ω (which may be unbounded) and apply the following steps:

1. discretize the domain Ω into a discrete *mesh* consisting of *mesh elements* (or simply elements);
2. expand all unknown field components in each element as p-th order polynomials;
3. substitute these field expansions into the relevant electromagnetic equations;
4. weight (or test) the residuals of the relevant equations with the same polynomials used to expand the fields, while also applying appropriate boundary conditions; and
5. solve the resulting system of equations to determine the field coefficients, and hence recover an approximation to the fields.

The remainder of this chapter is focused on providing sufficient details of this procedure to clarify the structure of the resulting system of equations.

3.2 Discretization of The Problem Domain

In order to numerically solve the electric vector wave equation for the scattered field problem (2.5), repeated here for convenience

$$(j\omega)^2\varepsilon\vec{E}^{sct} + \nabla\times(\frac{1}{\mu}\nabla\times\vec{E}^{sct}) = -j\omega\vec{J}^{sct} - \nabla\times(\frac{1}{\mu}\vec{M}^{sct}) \tag{3.1}$$

we must specify the domain (space) of interest Ω, the sources $\vec{J}^{sct}$ and $\vec{M}^{sct}$, and the scattering targets via the profiles $\varepsilon(\vec{x})$ and $\mu(\vec{x})$. The geometry, namely the domain Ω and geometric extent of the scatterers $\varepsilon(\vec{x})$ and $\mu(\vec{x})$ can be represented by a geometric model in any number of CAD programs such as Gmsh [36]. Once specified, we break the problem domain (space) into a set of elements by *mesh generation* supported by the CAD program. This produces a model consisting of a number of *volumetric elements* and *surface elements*[1]. The set of elements produces an approximate *partition* Ω_h of the computational domain Ω, as illustrated in Figure 3.1. It should be emphasized that the domain Ω may have infinite extent, but that Ω_h must be finite. In this case, appropriate *absorbing boundary conditions* are required to effectively enforce the radiation conditions discussed in Section 3.6.

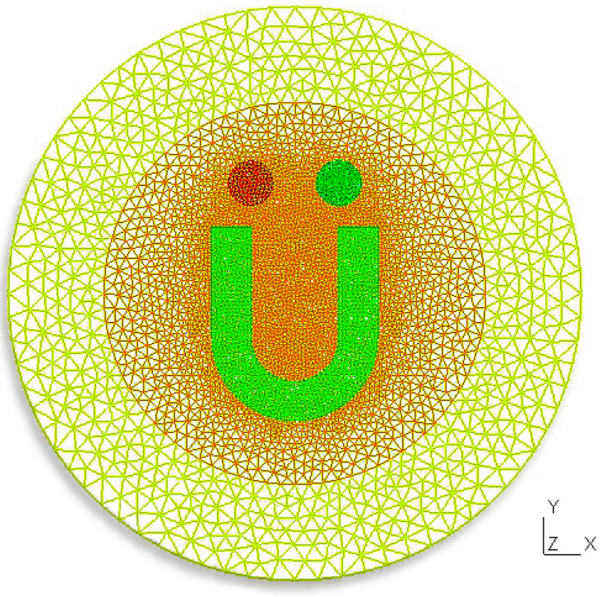


FIGURE 3.1: An example of mesh generation depicting the approximate partition Ω_h (mesh) for a 2D domain. Each triangle is referred to as a mesh element

[1]Here, a volumetric element has the same dimension as the problem space, i.e., for 2D problems we consider 2D volumes, while a surface element has a reduced dimension, i.e., for 2D problems we consider 1D surfaces.

3.3 Nodal Basis Expansion

The next step in the DGM numerical scheme is to expand all unknown field components in a basis[2]. For the 3D vector wave equation this implies expanding all three scalar components of $\vec{E}^{sct}$, while for a 2D TM problem only a single field component is required. A convenient and standard basis for this application is the nodal Lagrange basis, i.e., the basis is chosen to be pth order Lagrange interpolating polynomials on any given mesh element [37]. As the polynomial expansion varies from element to element, the expansion is *discontinuous* as the name DGM suggests. One advantage of this basis choice is that it leads to a numerical-integration-free implementation of the method for first-order geometric elements [14].

A pth order expansion uses the basis functions $\ell_0(\vec{x})$, $\ell_1(\vec{x})$, …, $\ell_{N_p-1}(\vec{x})$, where N_p is the total number of basis functions required to span the space of at most pth-order polynomials in either two or three dimensions. Corresponding to the set of basis functions $\{\ell_i\}$ is a set of nodal points $\{\vec{x}_i\}$ that define the polynomials such that $\ell_i(\vec{x}_j) = \delta_{ij}$ where δ_{ij} is the Kronecker delta.

Expanding a field component in this basis is straightforward: field values at the nodal points $\{\vec{x}_i\}$ serve as the basis coefficients. For example, the z-component of the electric field anywhere in the nth mesh element V_n is now assumed to

[2]A basis for any function space is a minimum set of linearly independent functions that spans the space.

take the form

$$\begin{aligned} E_z^{sct}(\vec{x}) &= \sum_{i=0}^{Np-1} E_z^{sct}(\vec{x}_i)\ell_i(\vec{x}) \qquad \vec{x} \in V_n \\ &= [\ell_0(\vec{x}), \ell_1(\vec{x}), ..., \ell_{N_p-1}(\vec{x})][E_z^{sct}(\vec{x}_0), E_z^{sct}(\vec{x}_1), ..., E_z^{sct}(\vec{x}_{N_p-1})]^T \\ &= \underline{\ell}(\vec{x})^T \underline{E}_z^{sct} \end{aligned} \tag{3.2}$$

where the nodal points $\{\vec{x}_i\}$ are confined to V_n and where we have introduced the vectors (arrays) $\underline{\ell}(\vec{x})$ and $\underline{E}_z^{sct}$ that respectively contain the basis functions and the field coefficients, and where superscript T denotes transposition.

With this basis we are now in the position to approximate the unknown fields everywhere in the computational domain as polynomials on each mesh element.

3.4 Residual Testing

Given that we have the nodal expansion (3.2) available to expand field components on every element, we can now consider the effect of substituting these expansions into the vector wave equation (3.1). It is not necessary to explicitly make this substitution to recognize the result: given N_V mesh elements, a pth order expansion in each element, and N_f field components, the total number of nodal basis unknowns resulting from basis expansion is $N_V \times N_p \times N_f$. As the vector wave equation itself provides $N_V \times N_f$ equations we immediately recognize that there is a deficiency in the number of equations provided by mere substitution. Therefore our goal is to generate enough equations to recover all of the coefficients. The DGM accomplishes this by appropriately weighting/testing the resulting expanded field equations on each mesh element with the same set of polynomials used to expand the fields. Testing is achieved by projecting the residual onto the basis functions via an inner product, with the net result being a system of equations for determining the polynomial basis coefficients that represent the fields. Using the expansion functions as testing functions is referred to as Galerkin testing, and contributes to the moniker DGM. As it turns out, the testing procedure is also critical to the DGM formulation in that it serves to couple field solutions in neighbouring elements[3].

To illustrate the importance of residual testing in the DGM formulation assume that a single testing function is represented by $\psi(\vec{x})$, $\vec{x} \in V_n$. Testing the vector wave equation and applying integration by parts twice, (i.e., twice applying the identity $\psi\nabla \times \vec{F} = \nabla \times (\psi\vec{F}) - \nabla\psi \times \vec{F}$) and using Stoke's theorem, gives:

$$\begin{aligned}(j\omega)^2 \int\limits_{V_n} \psi\varepsilon\vec{E}^{sct} dv + \int\limits_{V_n} \psi\nabla \times (\frac{1}{\mu}\nabla \times \vec{E}^{sct}) dv + \oiint\limits_{\partial V_n} \psi \Bigg((\vec{n} \times (\frac{1}{\mu}\nabla \times \vec{E}^{sct}))^{\wedge} \\ -(\vec{n} \times (\frac{1}{\mu}\nabla \times \vec{E}^{sct})) \Bigg) ds = -j\omega \int\limits_{V_n} \psi\vec{J}^{sct} dv - \int\limits_{V_n} \psi\nabla \times (\frac{1}{\mu}\vec{M}^{sct}) dv\end{aligned} \tag{3.3}$$

The magnetic vector wave equation equivalent form can be obtained by duality. The above equation is referred to as the *strong form* of the DGM testing procedure (the *weak form* is obtained by applying integration by parts once) [14]. Here, special attention should be paid to the resulting surface integral around the boundary ∂V_n of the element V_n. Close examination of the vector wave equation (3.1) suggests that this surface integral should evaluate to zero. A key concept in Discontinuous Galerkin formulations is that this surface integral must be used to couple information between neighbouring elements. It is important to recognize that the surface integral makes reference to a boundary value denoted by superscript $\wedge$ that can be reconstructed using information from both an element and its neighbours to ensure that the boundary conditions required for Maxwell's equations are enforced. Without this term, the DGM would be an entirely local formulation on each element and would not provide a method for solving Maxwell's equations.

[3]Here, elements are neighbours if they share a common face.

3.5 Constructing the DGM System of Equations

To illustrate the construction of the desired DGM system of equations we limit our formulation to consider i) constant constitutive parameters ε_n and μ_n on each mesh element V_n, and ii) that fields in each element are expanded to the same order p. These assumptions are not necessary and are only made to simplify the exposition. In fact, one of the strengths of DGM formulations is that they permit polynomial expansions of the constitutive parameters in each element and variable solution orders to be applied over the mesh elements [22][14].

Substitution of the nodal basis expansion and subsequent testing using each basis function in the set $\{\ell_i(\vec{x})\}$ leads to a local system of equations in V_n. Specifically, expanding and testing the z-component of the vector wave equation leads to:

$$\left((j\omega)^2\varepsilon_n\boldsymbol{\mathcal{M}}^n - \boldsymbol{\mathcal{S}}_y^n\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}_y^n - \boldsymbol{\mathcal{S}}_x^n\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}_x^n\right)\underline{E}_z^{sct,n} + \boldsymbol{\mathcal{S}}_x^n\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}_z^n\underline{E}_x^{sct,n} + \boldsymbol{\mathcal{S}}_y^n\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}_z^n\underline{E}_y^{sct,n}$$
$$+ \underline{F}_z^{E,n} = -j\omega\boldsymbol{\mathcal{M}}^n\underline{J}_z^{sct,n} - (-\boldsymbol{\mathcal{S}}_y^n\frac{1}{\mu_n}\underline{M}_x^{sct,n} + \boldsymbol{\mathcal{S}}_x^n\frac{1}{\mu_n}\underline{M}_y^{sct,n}) \tag{3.4}$$

where the x- and y-components of the vector wave equation lead to similar expressions that can be obtained from (3.4) through cyclic interchange of the coordinates $x \to y \to z \to x$. In the discrete equations the quantities $\underline{E}_{x,y,z}^{sct,n}$, $\underline{J}_{x,y,z}^{sct,n}$ and $\underline{M}_{x,y,z}^{sct,n}$ are nodal coefficients for the field and source components in V_n and the matrices $\boldsymbol{\mathcal{M}}^n$, $\boldsymbol{\mathcal{S}}_{x,y,z}^n$ and $\boldsymbol{\mathcal{D}}_{x,y,z}^n$ are respectively referred to as the local *mass*, *stiffness*, and *derivative* matrices for the nodal basis in V_n. These matrices arise from the weighted residual testing procedure and are given by:

$$\boldsymbol{\mathcal{M}}^n \triangleq \int\limits_{V_n} \underline{\ell}(\vec{x})\underline{\ell}(\vec{x})^T dv, \qquad \boldsymbol{\mathcal{S}}^n_\zeta \triangleq \int\limits_{V_n} \underline{\ell}(\vec{x})\frac{\partial}{\partial\zeta}\underline{\ell}(\vec{x})^T dv, \qquad \boldsymbol{\mathcal{D}}^n_\zeta = (\boldsymbol{\mathcal{M}}^n)^{-1}\boldsymbol{\mathcal{S}}^n_\zeta \tag{3.5}$$

Construction of these matrices is straightfoward in the nodal basis and, for first-order geometric elements, lead to analytic (numerical-integration-free) evaluations. The interested reader can refer to [14] for details. The remaining, and perhaps most important, quantity in (3.4) is the *flux term* $\underline{F}^{E,n}_z$ that arises from the evaluation of the surface integral in (3.3). For an element V_n this flux term involves contributions from the fields in V_n and each of its neighbours. For example, if $V_{n'}$ neighbours V_n, its contribution to $\underline{F}^{E,n}_z$, which we denote as $\underline{F}^{E,n,n'}_z$ can be shown to take the form

$$\underline{F}^{E,n,n'}_z = \boldsymbol{\mathcal{F}}^{n,-}_z \underline{E}^{sct,n} + \boldsymbol{\mathcal{F}}^{n,+}_z \underline{E}^{sct,n'} \tag{3.6}$$

where $\underline{E}^{sct,n}$ encompasses all nodal coefficients for all electric field values in V_n. In this way, both the unknowns in V_n and its immediate neighbours contribute the overall field solution, i.e., the fields become globally coupled. Through these flux terms, boundary conditions for Maxwell's equations such as dielectric interfaces, perfect electric conductors, and absorbing boundary conditions can be readily enforced. A complete description of the evaluation of these flux terms is beyond the scope of this thesis, but details can be found in [14] [22] [38]. The importance of these flux terms for enforcing absorbing boundary conditions is presented in Section 3.6.

We now examine the structure of the resulting global system of equations obtained by constructing the discrete local system (3.4) for each element. To simplify the notation we define

$$\begin{aligned}\boldsymbol{\alpha}^n_{\zeta\xi} &= (j\omega)^2\varepsilon_n\boldsymbol{\mathcal{M}}^n - \boldsymbol{\mathcal{S}}^n_y\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}^n_y - \boldsymbol{\mathcal{S}}^n_x\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}^n_x \\ \boldsymbol{\beta}^n_{\zeta\xi} &= \boldsymbol{\mathcal{S}}^n_\zeta\frac{1}{\mu_n}\boldsymbol{\mathcal{D}}^n_\xi\end{aligned} \tag{3.7}$$

and collect the equations for the x-, y-, and z-components of the vector wave equation discretization (3.4) leading to the following equations enforced on V_n (where the explicit dependence on n has been suppressed):

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{\alpha}_{yz} & \boldsymbol{\beta}_{yx} & \boldsymbol{\beta}_{zx}\\ \boldsymbol{\beta}_{xy} & \boldsymbol{\alpha}_{zx} & \boldsymbol{\beta}_{zy}\\ \boldsymbol{\beta}_{xz} & \boldsymbol{\beta}_{yz} & \boldsymbol{\alpha}_{yx}\end{bmatrix}\begin{bmatrix}\underline{E}^{sct}_x\\ \underline{E}^{sct}_y\\ \underline{E}^{sct}_z\end{bmatrix} + \begin{bmatrix}\underline{F}^{E,n}_x\\ \underline{F}^{E,n}_y\\ \underline{F}^{E,n}_z\end{bmatrix} = \\ &-j\omega\begin{bmatrix}\boldsymbol{\mathcal{M}} & & \\ & \boldsymbol{\mathcal{M}} & \\ & & \boldsymbol{\mathcal{M}}\end{bmatrix}\begin{bmatrix}\underline{J}^{sct}_x\\ \underline{J}^{sct}_y\\ \underline{J}^{sct}_z\end{bmatrix} - \mu^{-1}\begin{bmatrix} & -\boldsymbol{\mathcal{S}}_z & \boldsymbol{\mathcal{S}}_y\\ \boldsymbol{\mathcal{S}}_z & & -\boldsymbol{\mathcal{S}}_x\\ -\boldsymbol{\mathcal{S}}_y & \boldsymbol{\mathcal{S}}_x & \end{bmatrix}\begin{bmatrix}\underline{M}^{sct}_x\\ \underline{M}^{sct}_y\\ \underline{M}^{sct}_z\end{bmatrix}\end{aligned} \tag{3.8}$$

For 2D TM problems this system reduces to:

$$\boldsymbol{\alpha}_{yx}\underline{E}^{sct}_z + \underline{F}^{E,n}_z = -j\omega\boldsymbol{\mathcal{M}}\underline{J}^{sct}_z + \mu^{-1}\boldsymbol{\mathcal{S}}_y\underline{M}^{sct}_x - \mu^{-1}\boldsymbol{\mathcal{S}}_x\underline{M}^{sct}_y \tag{3.9}$$

Examining the systems arising on V_n provides the following insight:

- The dimensions of the system for a single element for the 3D vector wave equation are $3N_p \times 3N_p$. Enforcing an equivalent system on each element will result in a global system of equations of size $3N_VN_p \times 3N_VN_p$.
- For the 2D vector wave equation the local system dimension are $N_p \times N_p$. The global system of equations has dimensions $N_VN_p \times N_VN_p$.
- *Without the flux term,* the resulting global system in either 2D or 3D, obtained by enforcing the required equations on each element would be *block*

diagonal.

- Including the flux term introduces off-diagonal blocks in a global matrix that has a blockwise structure. The number of faces on each element dictates the number of off-diagonal blocks introduced by each element.
- Increasing the solution order p on each element, increases the size of the block structure in the global system.
- Increasing the number of elements N_V in the mesh, does not affect the size of the block structure but increases the total number of blocks.

Proper construction of the DGM equations for the vector wave equation, including evaluation of appropriate flux terms for imposed boundary conditions leads to the following global system of eqautions

$$\boldsymbol{\mathcal{K}}\underline{E}^{sct} = \boldsymbol{\mathcal{R}}_J\underline{J}^{sct} + \boldsymbol{\mathcal{R}}_M\underline{M}^{sct} \tag{3.10}$$

where the global system $\boldsymbol{\mathcal{K}}$ and has dimensions $N_fN_VN_p \times N_fN_VN_p$ where $N_f = 3$ for 3D problems, and $N_f = 1$ for 2D TM problems. The quantities $\underline{E}^{sct}$, $\underline{J}^{sct}$ and $\underline{M}^{sct}$ are $N_fN_VN_p \times 1$ column vectors respectively containing all field and source nodal coefficients over the entire discrete domain Ω_h. By construction, $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{R}}_{J,M}$ are sparse matrices, where $\boldsymbol{\mathcal{R}}_{J,M}$ accounts for appropriate testing of the sources.

In order to determine the global nodal coefficients for the scattered field, we must solve the sparse block system of equations given in (3.10). That is, solving the DGM problem now amounts to solving a system of equations $\boldsymbol{A}\underline{x} = \underline{b}$, where the matrix $\boldsymbol{A}$ and right-hand-side $\underline{b}$ are known.

3.6 Absorbing Boundary Conditions

For unbounded field problems it is necessary to truncate the domain Ω leading to a finite discrete domain Ω_h. The DGM formulation presented in the previous section is complete with the exception of how to handle the flux terms required for the scheme at the boundary of the computational domain. At this boundary, which we denote Γ^{ABC} for a surface where an *absorbing boundary condition* (ABC) is needed, elements V_n are without neighbours over one or more of their faces and special handling of the flux terms is required.

One common way of handling absorbing boundary conditions through the flux terms is to assume that the contribution from the neighbour is zero but otherwise leave the flux decomposition the same as any other dielectric interface [39]. This condition is known as a Silver-Müller absorbing boundary condition. The drawback of this approach is that it is only accurate if the boundary is sufficiently far from all scatterers that fields impinging on the boundary are well approximated by plane waves normally incident on the boundary [22].

Another method for handling absorbing boundary conditions more rigorously is to impose an *exact radiating boundary condition* (ERBC) by means of a Huygens' surface. As discussed in Section 2.5 the field values on a surface enclosing all sources can be used to determine the field values required from non-existing neighbours of an element adjacent to Γ^{ABC}. The idea is shown in Figure 3.2. An additional surface Γ^{ERBC} is built into the mesh (denoted in red in the figure) and elements connected to this surface (also shown in red) are identified as contributors to the boundary condition on Γ^{ABC}. Elements connected to Γ^{ABC} (denoted in blue) use the fields on Γ^{ERBC} in order to produce the values required to evaluate their fluxes. This is accomplished by simply specifying the surface integral and observation points in (2.11) as follows:

$$\vec{E}(\vec{x}) = \int_{\Gamma^{ERBC}} \Big(-j\omega\mu_b[\vec{n}(\vec{x}\,') \times \vec{H}(\vec{x}\,')]G_b(\vec{x},\vec{x}\,') + [\vec{n}(\vec{x}\,')\cdot\vec{E}(\vec{x}\,')]\nabla' G(\vec{x},\vec{x}\,')$$
$$+ [\vec{n}(\vec{x}\,') \times \vec{E}(\vec{x}\,')] \times \nabla' G_b(\vec{x},\vec{x}\,') \Big) ds', \qquad \vec{x}\,' \in \Gamma^{ERBC}, \qquad \vec{x} \in \Gamma^{ABC} \tag{3.11}$$

While enforcing ERBCs is far more accurate than Silver-Müller boundary conditions and permits mesh truncation boundaries Γ^{ABC} to be closer to scatterers than simpler absorbing conditions, ERBCs come with a significant price. Specifically if Γ^{ERBC} consists of N_{ERBC} mesh edges, and Γ^{ABC} consists of N_{ABC} mesh edges, then a block of dense matrix entries proportional to $N_{ERBC} \times N_{ABC}$ (depending on expansion order) is introduced into the DGM system matrix and

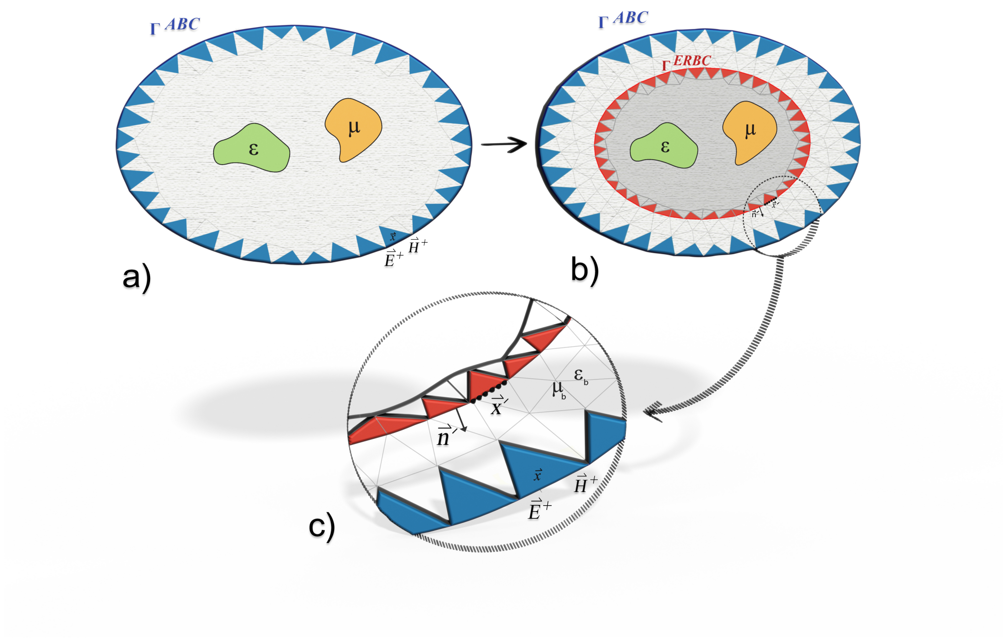


FIGURE 3.2: Geometric surfaces for imposing Silver-Müller boundary conditions and exact radiating boundary conditions: a) Silver-Müller boundary conditions; b) Exact radiating boundary conditions; c) The elements inside the ERBC surface (red) are used to determine fields on the problem boundary (blue).

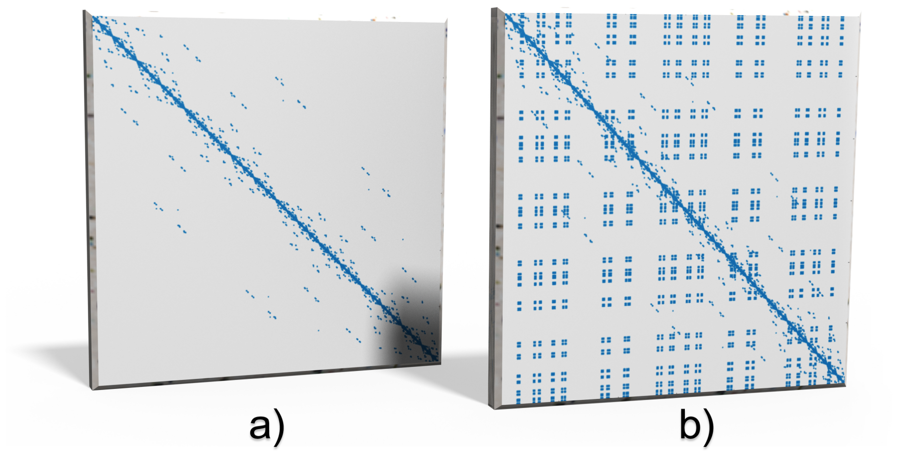


FIGURE 3.3: Matrix entries introduced by exact radiating boundary conditions: a) The sparse block matrix for Silver-Müller boundary conditions; and b) Additional matrix entries introduced as an effective dense sub-block as a result of ERBCs.

have a great affect on performance. An example of the matrix structures obtained for Silver-Müller ABCs and ERBCs is shown for emphasis in Figure 3.3.

3.6.1 Chapter Summary

In this chapter we have formally presented the important aspects of DGM formulations as they relate to $\mathcal{H}$-matrix preconditioning of the global DGM system (3.10). A complete implementation of DGM for the 2D TM vector wave equation, developed by Dr. Ian Jeffrey, Nicholas Geddert, Kevin Brown, and Dr. Joe LoVetri [22], was used throughout the remainder of this work. Although only briefly introduced, the effects of ERBC boundary conditions on the structure of the DGM system matrix have been empahsized. While ERBCs can improve the accuracy of the DGM solution, they add significant data to the global matrix that can largely affect the computational time and memory. In the upcoming chapters, we will formalize an $\mathcal{H}$-matrix framework that permits efficient and error-controllable solutions to DGM problems, with a focus on ERBC-enabled DGM.

Chapter 4

Preconditioned GMRES

Having constructed the discrete DGM system for the electric vector wave equation (3.10) we recognize that solving this system amounts to solving a square non-singular matrix equation

$$\boldsymbol{A}\underline{x} = \underline{b} \tag{4.1}$$

for system $\boldsymbol{A}^{n\times n}$ and right-hand-side vector $\underline{b}^{n\times 1}$ in order to determine the unknowns $\underline{x}^{n\times 1}$. Of note is the fact that the DGM system for time-harmonic problems is complex. The task at hand is to therefore produce the solution vector $\underline{x}$ in an efficient and scalable manner for a large problem size. This chapter surveys the choices that we've made to solve the DGM system of equations.

The literature on techniques of solving systems of equations is vast, and summarizing all available options is prohibitive. Instead, we simply note that adoption of the *generalized minimum residual method* (GMRES) [40] is a common choice for large complex systems. GMRES, an iterative method, benefits from accurate preconditioners to improve convergence [16]. $\mathcal{H}$-matrices, described in the next chapter, provide an error-controllable framework for producing an accurate preconditioner.

In this chapter we briefly discuss LU decomposition and review the preconditioned GMRES algorithm. It is assumed that the reader is familiar with Gaussian elimination.

4.1 Direct Solutions Methods

Direct solution methods may perhaps be best described as being *non-iterative*. By brute force these techniques aim to provide a guaranteed direct path to the solution to a system of equations. Common direct methods are Gaussian elimination [41], QR decomposition [16], and LU decomposition [16]. Decomposition (or factorization) methods such as QR and LU offer the benefit of increased performance when solving multiple right-hand-sides. The complexity of these methods for dense matrices is generally $\mathcal{O}(n^3)$ where n is the number of unknowns (although improvements are possible). For a fixed problem size LU decomposition is more efficient than QR. For sparse matrices, such as the DGM system (3.10), complexity depends largely on the ordering of the matrix, and *pivoting strategies* that attempt to reorder to matrix to improve performance are important [16]. For our applications, LU decomposition is an appropriate choice and we summarize the approach in the following section.

4.2 LU Decomposition

The Lower-Upper (LU) Decomposition of a matrix $\boldsymbol{A}$ seeks to represent the matrix as a product of lower- and upper-triangular matrices $\boldsymbol{L}$ and $\boldsymbol{U}$. This decomposition is theoretically possible for any non-singular matrix $\boldsymbol{A}$ (pivoting may be required) and can be obtained directly from Gaussian elimination [41, 42]. Formally the decomposition produces

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}, \qquad L_{ij} = 0 \quad \forall \quad i < j, \qquad U_{ij} = 0 \quad \forall \quad i > j \tag{4.2}$$

where A_{ij} indicates the entry at row i, column j of matrix $\boldsymbol{A}$.

Once the decomposition is available, solving the system of equations $\boldsymbol{A}\underline{x} = \underline{b}$ is equivalent to solving $\boldsymbol{L}\boldsymbol{U}\underline{x} = \underline{b}$ which can be solved as follows:

$$\begin{aligned} \text{Let} \quad & \boldsymbol{U}\underline{x} = \underline{d} \\ \text{Solve} \quad & \boldsymbol{L}\underline{d} = \underline{b} \text{ for } \underline{d} \\ \text{Solve} \quad & \boldsymbol{U}\underline{x} = \underline{d} \text{ for } \underline{x} \end{aligned} \tag{4.3}$$

thus, solving a system of equations using LU decomposition requires solving two triangular systems of equations once the decomposition is known. These solutions are readily computed using *forward* and *backward* substitution respectively [42]. From the above procedure it is clear that changing the right-hand-side $\underline{b}$ leaves the decomposition unchanged. Forward-substitution and backward-substitution require $\mathcal{O}(n^2)$ operations for dense systems [16], while the elimination steps required to produce the decomposition require $\frac{n^3}{3} + \mathcal{O}(n)$ operations. While this approach works very well for relatively small problems, the cubic complexity implies that this method does not scale appropriately for larger problems, especially considering that even when systems are sparse, there is no guarantee that their decompositions are sparse [16]. Again we stress the need for pivoting strategies [43] that are not considered herein due to the complications that they impose on a basic $\mathcal{H}$-matrix implementation discussed in the next chapter.

4.3 GMRES

For large systems of equations, the cost of direct methods such as LU decomposition is prohibitive. *Iterative* methods alleviate this cost by formulating solution procedures that iteratively update a solution estimate. Iterative update schemes typically incur the cost of performing matrix-vector-products at each iteration, a cost that is $\mathcal{O}(n^2)$ for dense matrices and generally $\mathcal{O}(n)$ for sparse matrices. Provided the total number of iterations required to *converge* to a given tolerance is small compared to n, iterative solutions provide expedient solutions methods.

One common iterative technique for solving general systems of equations (including complex systems) is the *generalized minimum residual method* (GMRES). MATLAB contains a built-in GMRES routine, that was used to produce the numerical results presented in Chapter 6. For completeness in this work, what follows outline the GMRES algorithm.

A general iterative method for solving a linear system of equations $\boldsymbol{A}\underline{x} = \underline{b}$ can be summarized as iteratively updating the solution vector $\underline{x}^{(i)}$ at iteration i, while minimizing the residual vector $\underline{r}^{(i)}$ at each iteration:

$$\begin{aligned} \underline{x}^{(i+1)} &= \underline{x}^{(i)} + \gamma^{(i)}\underline{d}^{(i)} \\ \underline{r}^{(i+1)} &= \boldsymbol{A}\underline{x}^{(i+1)} - \underline{b} \end{aligned} \tag{4.4}$$

where $\gamma^{(i)}$ is the step length taken in the search direction $\underline{d}^{(i)}$ at the ith iteration.

GMRES is one of a class of iterative solvers commonly referred to as Krylov subspace solvers [16, 44]. GMRES can be presented in a number of ways, but we use the *projection method* approach summarized in [16].

Projection methods use two affine subspaces $\mathcal{K}$ and $\mathcal{L}$ to seek appropriate spaces for representing the solution and residual. It is assumed that the solution vector $\underline{x}$ can be represented in $\mathcal{K}$, while constraints on the residual are enforced in $\mathcal{L}$. Given bases for the spaces, $\boldsymbol{V}$ and $\boldsymbol{W}$, it can be shown that the projection method update scheme is [16]:

$$\underline{x}^{(i+1)} = \underline{x}^{(i)} + \boldsymbol{V}(\boldsymbol{W}^H\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^H\underline{r}^{(i)} \tag{4.5}$$

So long as the dimensions of the subspaces $\mathcal{K}$ and $\mathcal{L}$ are relatively small compared to the system size, that is $\boldsymbol{V}$ and $\boldsymbol{W}$ have one relatively small dimension, the solution method is expedient.

GMRES is a projection method that specifically chooses the subspace $\mathcal{K}$ as the Krylov subspace $\mathcal{K}_m(\boldsymbol{A}, \underline{r}) = \text{span}\{\underline{r}, \boldsymbol{A}\underline{r}, \boldsymbol{A}^2\underline{r}, \ldots, \boldsymbol{A}^{m-1}\underline{r}\}$ and chooses the constraint space $\mathcal{L} = \boldsymbol{A}\mathcal{K}$. Note that for a non-zero initial guess these spaces are shifted from the origin and are thus affine spaces. The choice of $\mathcal{L} = \boldsymbol{A}\mathcal{K}$ imposes an *oblique* projection method, which guarantees that $\boldsymbol{W}^H\boldsymbol{A}\boldsymbol{V}$ is non-singular [16].

The following pseudocode summarizes the basic GMRES algorithm. It is based on the presentation in [16] and is an efficient implementation of the generation of the bases $\boldsymbol{V}$ and $\boldsymbol{W}$ based on Householder transformations. Note than when m increases, storage requirements and computational time increase; choosing very large m is not practical. There are a number of variations and improvements that can be made to GMRES in order to overcome this problem. One is to restart the algorithm (step 4), and the other is to use preconditioners to improve convergence.

Algorithm 1 GMRES without preconditioning

```
1- Start: Choose $\underline{x}^{(0)}$ and dimension $m$ of the Krylov subspaces
2- Efficient computation of subspaces using orthogonalization (Arnoldi process)
    Compute $\underline{r}^{(0)} = \underline{b} - \boldsymbol{A}\underline{x}^{(0)}$, $\beta := ||\underline{r}^{(0)}||_2$ and $\underline{v}_1 = \frac{r^{(0)}}{\beta}$
    For $j = 1, \dots, m$ Do
        Compute $\underline{\omega}_j := \boldsymbol{A}\underline{v}_j$
        For $i = 1, \dots, j$ , Do
            $h_{i,j} := (\underline{\omega}_j, \underline{v}_i)$ and $\underline{\omega}_j := \underline{\omega}_j - h_{i,j}\underline{v}_i$
        EndDo
        $h_{j+1,1} = ||\underline{\omega}_j||_2$   If $h_{j+1,j} = 0$ Stop   $\underline{v}_{j+1} = \frac{\underline{\omega}_j}{h_{j+1,1}}$
    EndDo
    Define $\boldsymbol{V}_m := [\underline{v}_1, \dots \underline{v}_m]$ and $\boldsymbol{H}_m = \{h_{i,j}\}$
3 Form the approximate solution: Compute $\underline{x}^{(m)} = \underline{x}^{(0)} + \boldsymbol{V}_m \underline{y}^{(m)}$
    where $\underline{y}^{(m)} = argmin_{\underline{y}} ||\beta \underline{e}_1 - \mathbf{H}_m \underline{y}||_2$ and $\underline{e}_1 = [1, 0, \dots, 0]^T$.
4 Restart:
    **if** satisfied **then** stop
    **else** set $\underline{x}^{(0)} \leftarrow \underline{x}^{(m)}$ and go to 2
```

4.4 Preconditioned GMRES

For iterative solvers, the spectrum (singular values) of operator $\boldsymbol{A}$ dictate the convergence rate [16]. Large systems of equations with poor spectral properties (often referred to as ill-conditioned systems), converges very slowly, if at all. As the spectrum of the system dictates convergence, it is natural to attempt to *condition* the system so as to have better spectral properties before solving the system, that is *preconditioning* the system. The general idea is that instead of solving $\boldsymbol{A}\underline{x} = \underline{b}$, we solve:

$$\boldsymbol{M}^{-1}\boldsymbol{A}\underline{x} = \boldsymbol{M}^{-1}\underline{b} \tag{4.6}$$

The problem is then the determination of a pre-conditioner $\boldsymbol{M}$ that is both easily an efficiently inverted (or more precisely systems involving the preconditioner can be easily solved) while providing better spectral properties of the product $\boldsymbol{M}^{-1}\boldsymbol{A}$. The ideal preconditioner is to choose $\boldsymbol{M} = \boldsymbol{A}$ which would lead to an identity matrix for the system of equations. Of course this is not feasible in practice, as solving systems with $\boldsymbol{M} = \boldsymbol{A}$ amounts to solving the original problem and we get nowhere. Augmenting GMRES to support a (left) preconditioner as in (4.6) is straightforward requiring only minor modifications [16]:

Algorithm 2 GMRES – with left Pre-conditioning

```
1- Start: Choose $\underline{x}^{(0)}$ and dimension $m$ of the Krylov subspaces
2- Efficient computation of subspaces using orthogonalization (Arnoldi process)
    Compute $\underline{r}^{(0)} = \boldsymbol{M}^{-1}(\underline{b} - \boldsymbol{A}\underline{x}^{(0)})$, $\beta := ||\underline{r}^{(0)}||_2$ and $\underline{v}_1 = \frac{\underline{r}^{(0)}}{\beta}$
    For $j = 1, \dots, m$ Do
        Compute $\underline{\omega} := \boldsymbol{M}^{-1}\boldsymbol{A}\underline{v}_j$
        For $i = 1, \dots, j$ , Do
            $h_{i,j} := (\underline{\omega}, \underline{v}_i)$ and $\underline{\omega} := \underline{\omega} - h_{i,j}\underline{v}_i$
        EndDo
        $h_{j+1,j} = ||\underline{\omega}||_2$ and $\underline{v}_{j+1} = \frac{\omega}{h_{j+1,j}}$
    EndDo
    Define $\boldsymbol{V}_m := [\underline{v}_1, \dots \underline{v}_m]$ and $\boldsymbol{H}_m = \{h_{i,j}\}$
3 From the approximate solution: Compute $\underline{x}^{(m)} = \underline{x}^{(0)} + \boldsymbol{V}_m\underline{y}^{(m)}$
    where $\underline{y}^{(m)} = argmin_{\underline{y}}||\beta\underline{e}_1 - \mathbf{H}_m\underline{y}||_2$ and $\underline{e}_1 = [1, 0, \dots, 0]^T$.
4 Restart:
    **if** satisfied **then** stop
    **else** set $\underline{x}^{(0)} \leftarrow \underline{x}^{(m)}$ and go to 2
```

4.4.1 Choosing a Preconditioner

As illustrated in the previous algorithm, preconditioners come with the associated cost of solving systems of equations. Preconditioners that are easily inverted rarely give good performance. One example of a common preconditioner is the incomplete LU decomposition (ILU) [45]. One drawback of ILU preconditioners is that they are not error-controllable, and preconditioners capable of providing error-controllable approximate solutions via either approximate inversion or approximate decompositions are desirable. An interesting recent approach that shows promise for providing this type of error-controllable framework for solving systems of equations is *hiearchical matrices*, a framework that permits inversion and or decomposition with controllable accuracy where we can sacrifice accuracy for performance. In the context of preconditioned iterative solvers, preconditioner accuracy requirements may be low enough that significant savings can be obtained by choosing just enough preconditioner accuracy for rapid convergence.

4.5 Chapter Summary

This chapter has briefly summarized both LU decomposition and preconditioned GMRES, approaches that are used for solving the DGM system of equations. LU decomposition was employed for solving the required preconditioner equations within the GMRES framework. The MATLAB implementation of GMRES is exploited in Chapter 6. In the next chapter the hierarchical matrix framework is developed so that we can represent, multiply, invert, and decompose, the DGM system matrix.

Chapter 5

$\mathcal{H}$-Matrices

A *hierarchical* matrix, or an $\mathcal{H}$-matrix, is a structured representation of a matrix that aims to permit efficient compression of matrix storage, while also attempting to enable acceleration of standard matrix operations, including decompositions, matrix-vector products and even matrix inversion. In this chapter the theory of $\mathcal{H}$-matrices is presented from the basic concept of approximating a matrix in a memory efficient way, to a complete description and implementation of $\mathcal{H}$-matrix LU decomposition. As we will see, $\mathcal{H}$-matrix representations and operations are error-controllable, where precision is gained by sacrificing time and memory gains. This suggests that $\mathcal{H}$-matrices are worth investigating as preconditioners for iterative solvers.

5.1 Introduction

The theory of $\mathcal{H}$-matrices presented in this chapter closely follows the work of both Bebendorf [19] and Hackbush [46]. As our overall goal of understanding and implementing $\mathcal{H}$-matrices is to accelerate DGM solutions to computational electromagnetics problems, cost complexity of $\mathcal{H}$-matrix arithmetic is provided but without proof; readers interested in formal arguments for the cost complexity can refer to [20].

5.1.1 Why and How is Compression Possible?

Before formally introducing $\mathcal{H}$-matrices and their theory, it is worth briefly discussing why it is generally possible to compress matrices and accelerate matrix operations arising from time-harmonic electromagnetics problems. A physical argument serves as motivation. Consider the light emitted by each individual star in a galaxy. An observer within the galaxy itself would be able to see the effect of many individual stars throughout the sky. A second observer located a very large distance from the galaxy may only be able to distinguish the galaxy as a single point in the night sky, that is, the complicated interaction between many sources and an observer at a large distance may be well approximated by a simplified source model. An illustration is provided in Figure 5.1. This observation is not new, and in fact is the theoretical basis for the multilevel fast multipole method (MLFMM) [2], a well-known acceleration scheme for time-harmonic electromagnetics problems. While MLFMM requires rigorous analytic mathematics related to the propagation of electromagnetic fields, an $\mathcal{H}$-matrix approximation to a given matrix attempts to obtain similar accleration and compression by simply discarding information stored in the matrix according to some desired accuracy.

In the case of a DGM solution to a scattering problem, we can tie the compression concept to the system matrix as follows: a sub-block of the matrix represents interactions between a set of sources within the mesh and a set of field points. Sub-blocks can be examined to determine if the sources and observation points represented by the sub-block are well-separated and warrant compression. We need to ensure that a matrix is represented in a way that it contains

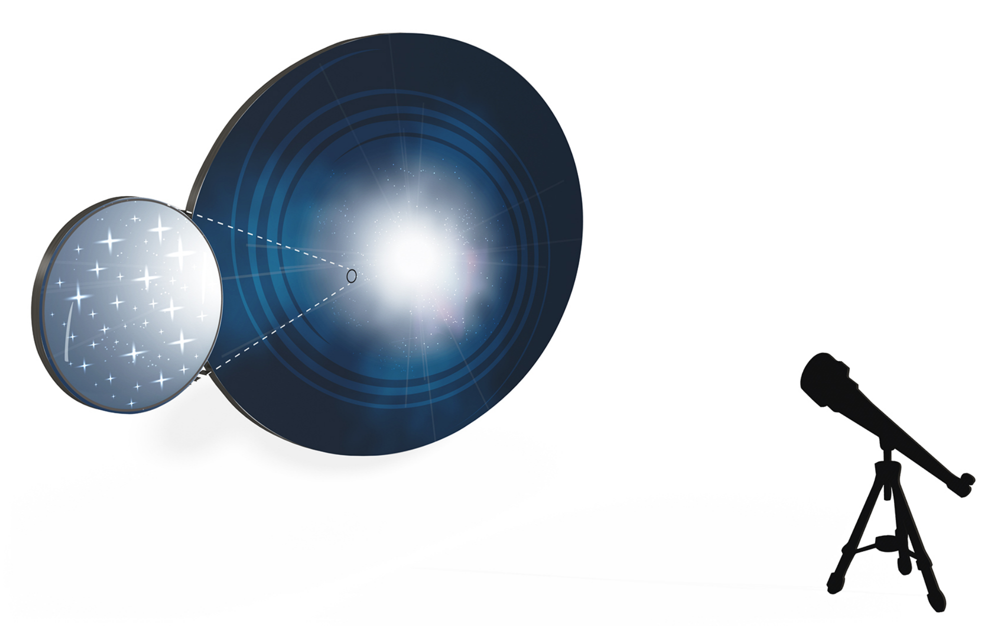


FIGURE 5.1: An illustration of source compression of light from distant sources. To an observer at a great distance, multiple sources may appear as a single source and can be represented with a simplified model.

compressible sub-blocks, i.e., sub-blocks represent interactions between sources and observation points that are well-separated. In this work, we order the rows and columns of a matrix based on *geometric partitioning*, as described in Section 5.2.1. Another approach is possible, *algebraic partitioning* which considers only the structure of a sparse matrix [47], and is especially valuable when no knowledge of the origins of the matrix (e.g., basis, mesh, etc.) are available. Note that in the context of (sparse) DGM systems it is actually the matrix inverse, or matrix factorization, that warrants compression. As is shown, the structure used to represent the DGM system matrix is the same structure used to represent either its inverse or its factorization. This ensures compression is possible.

5.1.2 Exploiting a Hierarchical Tree Structure

We may now begin to see an overall goal of identifying regions in a matrix that can be efficiently compressed. It turns out that a *tree structure* is a very

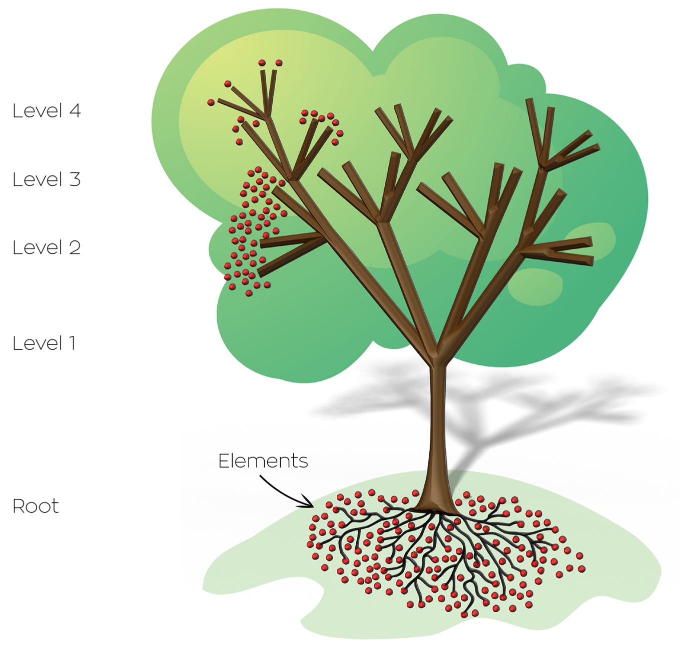


FIGURE 5.2: An abstract representation of a (quad) tree structure. Note that only elements on the left quadrant are shown. The level of the tree indicates the distance of a node to the root.

good tool for helping achieve this goal. Briefly, in the context of data structures (and mathematics), a hierarchical tree (rooted tree) is a structure in which each node of the tree may have children (other nodes), which may themselves have children, and so on. The maximum number of permitted children indicates the type of tree, e.g., a binary tree where each node has at most two children, or a quad tree where each node has at most four children. Tree structures are used extensively throughout $\mathcal{H}$-matrix theory and it is assumed that the reader is familiar with tree structures. An illustration is provided in Figure 5.2.

5.2 Building a Hierarchical Matrix

Constructing an $\mathcal{H}$-matrix can be summarized as a three step procedure:

1. partition the unknowns: this involves the technical step of building a *cluster tree*;

2. build a hierarchical data structure for representing the matrix: the technical step is building a *block cluster tree*; and
3. store the matrix entries in the hierarchical structure, compressing when possible: this requires the technical step of *rank compression*.

In order to understand $\mathcal{H}$-matrices, we next describe these three steps: constructing a cluster tree, building the block cluster tree, and a rank compression technique known as $\mathcal{R}_k$-matrices.

5.2.1 Constructing a Cluster Tree

The goal of a cluster tree is to provide a way of hierarchically partitioning a set. For simplicity we assume that we partition an *index set* whose entries $1, 2, \ldots N_V$ index a set of elements of interest. For $\mathcal{H}$-matrix acceleration of DGM problems, our aim is to partition the set of mesh elements. As will become clear, this partitioning aims to provide a way of localizing the DGM elements in space in such a way that the global DGM matrix can be hierarchically partitioned in a convenient way.

An informal definition of a (binary) cluster tree T_I for an index set $I = \{1, 2, \ldots n\}$ is [19]

- the root of the tree is associated with the index set I
- each node of the tree either has no children (a leaf node) or it has two children
- the index sets of the children of any non-leaf node form a partition of the node's index set

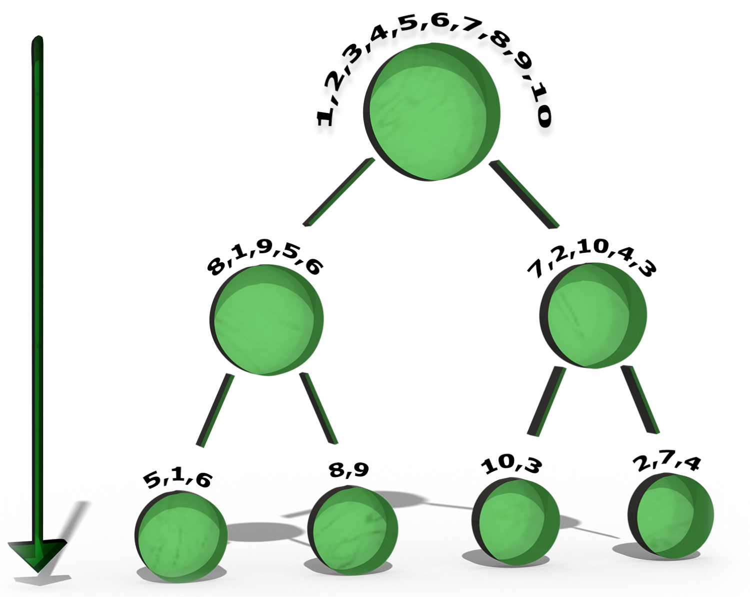


FIGURE 5.3: Construction of a 3-level cluster tree. The original index set is represented in a permuted order at the leaf nodes. The permuted order captures geometric locality of the elements referred to by the index set.

An illustration of a 3-level cluster tree can be seen in Figure 5.3.

In order to build a cluster tree we must specify a condition at which a node's index set is no longer subdivided to its two children. This is accomplished by specifying a quantity N_{max} such that only cluster tree nodes with associated index sets having more entries than N_{max} are partitioned. We also need to provide a way for determining how to subdivide the index set itself. A suitable partitioning scheme can be based on the geometry of the associated elements. One common way of doing this is Orthogonal Recursive Bisection (ORB) [48]. At each level of the tree a new coordinate x, y, or z is selected and the elements in a node's index set are sorted according to increasing coordinate and divided according to this coordinate into two sets of roughly equal size. The index set is correspondingly divided and the resulting two index sets are associated with the node's children. An example of the geometric partitioning that results from an ORB-generated cluster tree for a 2D problem is shown in Figure 5.4.

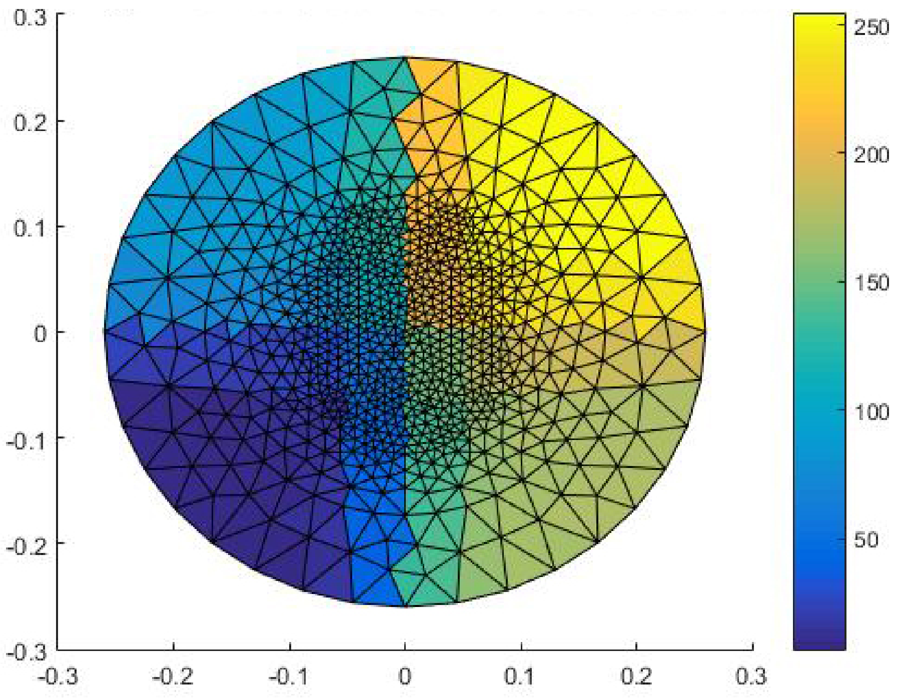


FIGURE 5.4: An example of an ORB constructed cluster tree's affect on element ordering. Elements are coloured according to their leaf node index in the tree. Elements sharing the same colour are geometrically close together.

Note that through the ORB construction, each leaf node of the cluster tree will contain elements that are spatially localized, and that the set of all cluster tree leaf nodes effectively re-orders the element index set into an ordering that has spatial localization. For example, in the Figure 5.3 the original ordering [1, 2, . . . , 10] may have no spatial structure, while the leaf-node ordering has spatial structure.

5.2.2 Building a Block Cluster Tree

While a cluster tree can be used to partition an index set, or a set of elements, our next goal is to partition the interactions between two sets of elements. That is, we want to hierarchically partition the Cartesian product $I \times J$ for index sets I and J. This is motivated by the fact that a DGM system matrix ultimately represents the interactions between all mesh elements (fields) with all mesh elements (sources). A hierarchical partition of $I \times J$ is referred to as a block cluster tree [19], which is obtained by recursively taking the Cartesian product of the index sets of the children of two cluster trees T_I and T_J. That is, any node $\tau \times \sigma$ in a block cluster tree (corresponding to two cluster trees τ and σ), is either a leaf node or has four children corresponding to $\tau_1 \times \sigma_1$, $\tau_1 \times \sigma_2$, $\tau_2 \times \sigma_1$ and $\tau_2 \times \sigma_2$, where τ_1 and τ_2 are the children of the cluster tree node τ and where σ_1 and σ_2 are the children of σ. If the block cluster tree node corrsponding to τ and σ is a leaf node then it represents the interactions of index sets τ and σ. An example is shown in Figure 5.5.

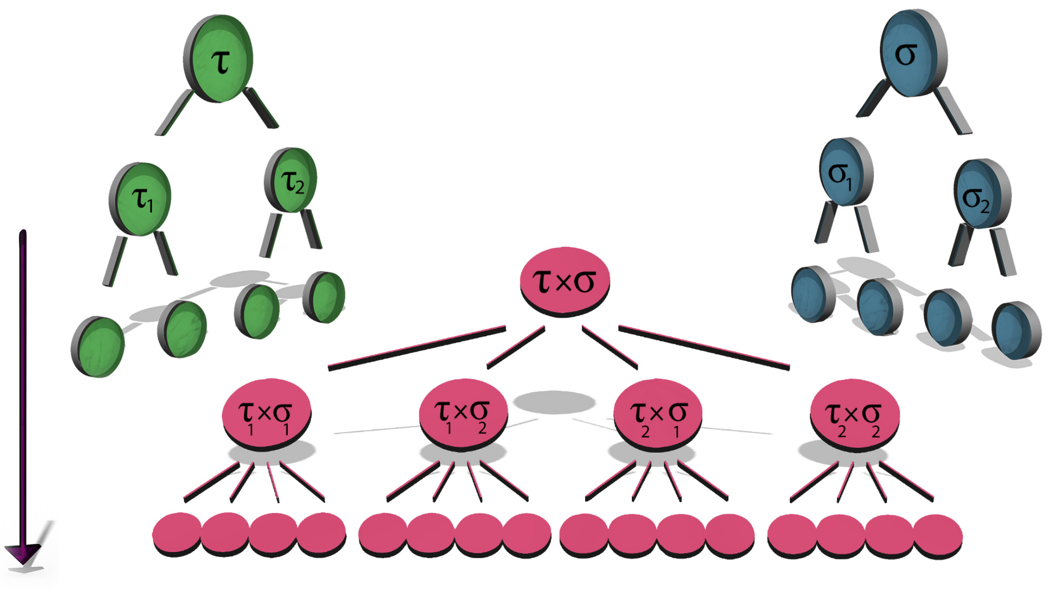


FIGURE 5.5: An illustration of a block cluster tree based on two cluster trees τ and σ. Nodes in the block cluster tree correspond to regions of element interactions.

5.3 Rank Compression and Tree Pruning

Based on the discussion in Section 5.1.1 and given that we have a structured representation of element interactions via a block cluster tree, it is natural to seek to compress interactions for given blocks in the tree, provided that the blocks represent well-separated interactions. Before formalizing the definition of an $\mathcal{H}$-matrix, we need to introduce the technique used to compress compressible blocks to a desired tolerance. This leads to the concept of $\mathcal{R}_k$-matrices, that provide the critical compression required for $\mathcal{H}$-matrices to offer acceleration and memory savings.

5.3.1 $\mathcal{R}_k$-Matrices

A general matrix $\boldsymbol{A}$ can be written in *outer product form*. The *rank* of a matrix is the number of linearly independent rows (or equivalently columns), and if the matrix $\boldsymbol{A}^{m\times n}$ has rank r then the outer product form can be written as:

$$\boldsymbol{A}^{m\times n} = \boldsymbol{U}^{m\times r}(\boldsymbol{V}^{n\times r})^H \tag{5.1}$$

where we are assuming complex matrices, and where H denotes the complex conjugate transpose (or Hermitian) operator. As an obvious example of this outer product form for a square matrix $\boldsymbol{A}^{n\times n}$ is $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^H$ where $\boldsymbol{U} = \boldsymbol{A}$ and $\boldsymbol{V} = \boldsymbol{I}$. For full-rank matrices, that is matrices whose rank is equal to the number of rows or columns, there is little benefit in representing a matrix in outer-product form. However, for low rank matrices, where r is small relative to the number of rows or columns, an outer-product representation can be very convenient.

A *rank-k* or $\mathcal{R}_k$ matrix is a matrix having rank at most k. From the proceeding discussion we know that a general $\mathcal{R}_k$-matrix $\boldsymbol{A}_k$ can be represented as

$$\boldsymbol{A}_k^{m\times n} = \boldsymbol{U}^{m\times k}(\boldsymbol{V}^{n\times k})^H \tag{5.2}$$

$\mathcal{R}_k$-matrices are also often called low-rank matrices, as there is little point in representing a (near) full-rank matrix in outer-product form. Throughout this work we use the terms $\mathcal{R}_k$-matrix and low-rank matrix interchangeably. At that core of $\mathcal{H}$-matrices is the concept of an $\mathcal{R}_k$-matrix approximation $\boldsymbol{A}_k$ to a given matrix $\boldsymbol{A}$.

5.3.2 $\mathcal{R}_k$-Matrix Approximations

If we seek an approximation to a general $m \times n$ matrix $\boldsymbol{A}$ among the set of $m \times n$ $\mathcal{R}_k$-matrices, the natural question is which approximation should be used? It can be shown that an error-controllable approximation is obtained through the singular value decomposition (SVD) of $\boldsymbol{A}$. Specifically we let:

$$\boldsymbol{A} = \tilde{\boldsymbol{U}} \Sigma \tilde{\boldsymbol{V}}^H \tag{5.3}$$

be the generalized singular value decomposition (SVD) of $\boldsymbol{A}^{n\times m}$ where Σ is a diagonal matrix where the diagonal entries are the singular values of $\boldsymbol{A}$, which may be assumed to be given in non-increasing order. The truncated singular value decomposition of $\boldsymbol{A}$ is then obtained by zeroing the singular values with indexes greater than k in the SVD of $\boldsymbol{A}$:

$$\boldsymbol{A} \approx \tilde{\boldsymbol{U}} \Sigma_k \tilde{\boldsymbol{V}}^H. \tag{5.4}$$

The presence of zero singular values implies that corresponding rows of $\tilde{U}$ and columns of $\tilde{V}$ do not contribute to the approximation. Therefore the truncated SVD enables determining the approximation:

$$\boldsymbol{A} \approx \boldsymbol{A}_k = \boldsymbol{U}\boldsymbol{V}^H \tag{5.5}$$

where $\boldsymbol{U}^{m\times k}$ and $\boldsymbol{V}^{n\times k}$ are computed from the truncated SVD. It can be shown that the error in this approximation is bounded by:

$$||\boldsymbol{A} - \boldsymbol{A}_k||_2 \leq \sigma_{k+1} \tag{5.6}$$

which implies that if a relative error tolerance ϵ is desired for approximating $\boldsymbol{A}$, then we can enforce that $\sigma_{k+1}/\sigma_1 \leq \epsilon$ [19].

The net result is an error-controllable $\mathcal{R}_k$-matrix approximation to $\boldsymbol{A}$ given by (5.5) where the error tolerance dictates the rank k of the resulting low-rank approximation. An $\mathcal{R}_k$-matrix approximation implies a significant reduction in storage and improved performance when computing operations like matrix-vector products provided k is sufficiently small [49].

Two brief points are worth discussing. First, we do not seek low-rank approximations to the global system. As we must invert the global system it should have full rank and compressing its overall global rank would be disastrous. Instead, only sub-blocks of the matrix are approximated, the choice of these subblocks being based on their position in the block cluster tree representing the matrix and an *admissible condition* discussed in Section 5.3.4. The second important point is that SVD is an expensive routine and attempts to minimize the size of the matrices that are being decomposed should be sought or other low-rank approximation yielding methods such as adaptive low-rank approximation [49], adaptive cross approximation (ACA) [50] [51], or a combination of SVD and ACA to compute the low rank approximation [52]. An investigation of these techniques is beyond the scope of this thesis, in which Matlab's SVD routine has been used, but future performance benefits could be obtained through other techniques.

5.3.3 Re-compression of an $\mathcal{R}_k$-matrix

Performance of $\mathcal{R}_k$-matrix generation (or matrix compression), is often efficient if the matrix in question is already low rank. Throughout $\mathcal{H}$-matrix algorithms, there are many requirements for re-compressing a matrix that is already represented in outer-product form. An efficent approach to re-compressing an $\mathcal{R}_k$-matrix to a tolerance ϵ, is to use a rank-revealing (economical) QR-factorization as follows. First, assume that the current rank of the approximation is k' and that the decomposition $\boldsymbol{A}_{k'} = \boldsymbol{U}'\boldsymbol{V}'^H$ is known.

- Calculate an economical QR-factorization $\boldsymbol{U}' = \boldsymbol{Q}_{U'}\boldsymbol{R}_{U'}$.
- Calculate an economical QR-factorization $\boldsymbol{V}' = \boldsymbol{Q}_{V'}\boldsymbol{R}_{V'}$.
- Note that the matrix $\boldsymbol{A} = \boldsymbol{U}'(\boldsymbol{V}')^H = \boldsymbol{Q}_{U'}(\boldsymbol{R}_{U'}\boldsymbol{R}_{V'}^H)\boldsymbol{Q}_{V'}^H$ where $\boldsymbol{Q}_{U'}$ and $\boldsymbol{Q}_{V'}$ are unitary matrices as a consequence of the QR decomposition [53].
- Calculate the singular value decomposition of $\boldsymbol{R}_{U'}\boldsymbol{R}_{V'}^H$.
- Truncate this decomposition to get the approximation $\boldsymbol{R}_U\boldsymbol{R}_V^H \approx \boldsymbol{R}_{U'}\boldsymbol{R}_{V'}^H$.
- Set $\boldsymbol{U} =: \boldsymbol{Q}_U\boldsymbol{R}_U$
- Set $\boldsymbol{V} =: \boldsymbol{Q}_V\boldsymbol{R}_V$.

This procedure will produce a (potentially new) approximation $\boldsymbol{A}_k = \boldsymbol{U}\boldsymbol{V}^H$ based on compression of $\boldsymbol{A}_{k'} = \boldsymbol{U}'\boldsymbol{V}'^H$. Critically, the rank-revealing nature of the QR decomposition ensures that the matrix $\boldsymbol{R}_{U'}\boldsymbol{R}_V'^H$ has dimensions at most $k' \times k'$, implying reduced effort to compute the required truncated SVD. The computational cost of the truncation can be shown to be $\mathcal{O}(k^2(n+m)+k^3)$ [49].

5.3.4 Admissible Condition

Now that a compression method in the form of $\mathcal{R}_k$-matrices is available, and a block cluster tree structure allows us to analyze blocks of interactions, we must next decide which blocks of a given matrix should be considered for compression. This decision is based on an *admissible condition*. A node $\tau \times \sigma$ in a block cluster tree, representing interactions between elements τ and σ is *admissible* if, for an admissibility parameter η,

$$\eta \times distance(\tau, \sigma) \geq min(diameter(\tau), diameter(\sigma)) \tag{5.7}$$

where $distance(\tau, \sigma)$ is a measure of the distance between the groups of elements represented by τ and σ. For the geometric partitioning considered in this work, this distance is calculated based on the minimum distance between bounding boxes containing the elements. This distance is compared to the size of the smaller bounding box[1]. A block cluster tree node representing $\tau \times \sigma$ is admissible if the distance between the boxes of appreciable size compared to the size of the boxes themselves. Admissible boxes are tagged for compression.

5.3.5 Final construction of an $\mathcal{H}$-Matrix

Having partitioned and labelled potentially compressible blocks, we can now formally introduce the structure of an $\mathcal{H}$-matrix. An $\mathcal{H}$-matrix is nothing more than a block cluster tree associated with a matrix, whose leaf nodes contain the corresponding entries in the matrix. All admissible nodes become leaf nodes, and any structure below an admissible node is pruned. Admissible leaf nodes

[1]We define the diameter of a bounding box as the distance between the the lower left and upper right corner of the smallest box containing all vertices of all elements indexed.s

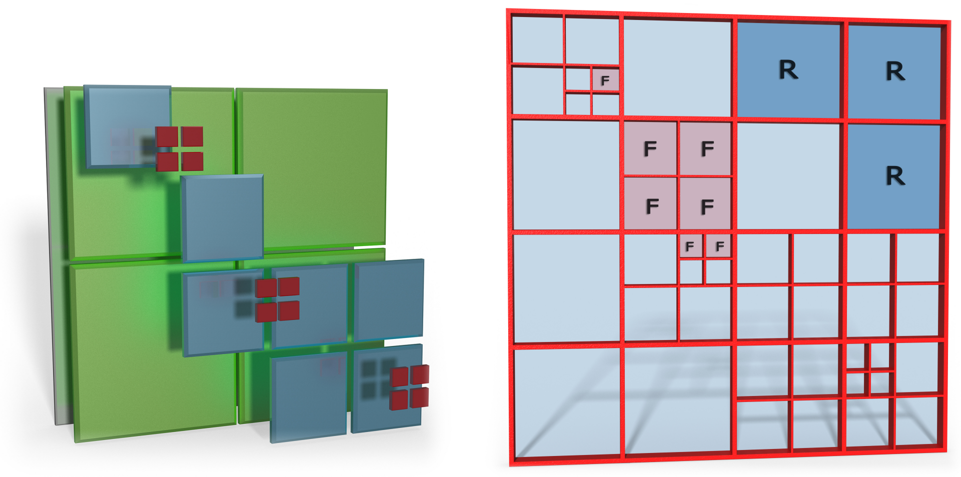


FIGURE 5.6: Illustration of an $\mathcal{H}$-matrix structure; (left) levels progressing from the root (back) to the leaf nodes (front), (right) a single level showing admissible leaf nodes $\mathcal{R}$, inadmissible leaf nodes $\mathcal{F}$ and inadmissible nodes that have children (no symbol).

are compressed, and store the associated matrix blocks as $\mathcal{R}_k$-matrices. Inadmissible leaf nodes (leaf nodes where the admissibility condition fails) store their portion of the matrix in uncompressed format. All other non-leaf nodes in the $\mathcal{H}$-matrix are inadmissible and simply provide hierarchical access to four children that represent four lower-level sub-blocks of the matrix structure. It is important to realize that any node in a $\mathcal{H}$-matrix can also be thought of as an $\mathcal{H}$-matrix due to the recursive nature of the block cluster tree. An example of the structure is shown in Figure 5.6.

As a node in an $\mathcal{H}$-matrix can itself be thought of as an $\mathcal{H}$-matrix it is beneficial to assign each node in the structure a *type*. The symbol $\mathcal{H}$ is used to refer to nodes that are hierarchical (have children), $\mathcal{R}$ is used to represent rank-compressed admissible leaf nodes, and $\mathcal{F}$ is used to represent inadmissible (full) leaf nodes. This labelling (with the exception of $\mathcal{H}$) is also shown in Figure 5.6.

Having introduced the concept of $\mathcal{H}$-matrices, we can now move on to a description of useful $\mathcal{H}$-matrix operations, and ultimately to a description of $\mathcal{H}$-

matrix LU decomposition.

5.4 Basic $\mathcal{H}$-matrix Operations

While the overall goal of the presentation on $\mathcal{H}$-matrices is to provide an algorithm for computing the approximate LU decompostion of a matrix, we first need to spend time on matrix operations that is required to implement the factorization. First, we should note that there are two ways that we can implement $\mathcal{H}$-matrix operations: we can assume a fixed-rank approach, in which the rank imposed in each subblock of the matrix is fixed, or we can implement an adaptive method, in which the blockwise rank of the matrix is permitted to vary, as necessary, to retain a certain accuracy. To ensure that operations are error-controllable, we chose adaptive $\mathcal{H}$-matrix operations.

5.4.1 $\mathcal{R}_\mathrm{k}$-matrix Addition

Assuming we have the two $\mathcal{R}_k$ matrices:

$$\boldsymbol{A} = \boldsymbol{U}_A \boldsymbol{V}_A^H, \qquad \boldsymbol{B} = \boldsymbol{U}_B \boldsymbol{V}_B^H, \tag{5.8}$$

with respective ranks k_A and k_B, the sum $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$ will have rank $k_C \leq k_A + k_B$. Construction of $\boldsymbol{C}$ can easily be implemented by understanding the outer-product as a sum of rank-1 matrices. That is:

$$\boldsymbol{A} = \boldsymbol{U}_A \boldsymbol{V}_A^H = \sum_{k=1}^{k_A} \underline{u}_{A,k} \underline{v}_{A,k}^H \tag{5.9}$$

where $\underline{u}_{A,k}$ is the kth column of $\boldsymbol{U}_A$. It follows that:

$$\begin{aligned} \boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} &= \boldsymbol{U}_A \boldsymbol{V}_A^H + \boldsymbol{U}_B \boldsymbol{V}_B^H \\ &= \sum_{k=1}^{k_A} \underline{u}_{A,k} \underline{v}_{A,k}^H + \sum_{k=1}^{k_B} \underline{u}_{B,k} \underline{v}_{B,k}^H \end{aligned} \tag{5.10}$$

Therefore, letting $[\boldsymbol{U}_A, \boldsymbol{U}_B]$ denote concatenation of the columns $\boldsymbol{U}_A$ and $\boldsymbol{U}_B$ we have:

$$\begin{aligned} \boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} &= [\boldsymbol{U}_A, \boldsymbol{U}_B][\boldsymbol{V}_A, \boldsymbol{V}_B]^H \\ &= \boldsymbol{U}_C \boldsymbol{V}_C^H \end{aligned} \tag{5.11}$$

Once the preliminary decomposition in terms of the defined $\boldsymbol{U}_C$ and $\boldsymbol{V}_C$ have been constructed, the resulting $\mathcal{R}_k$-matrix can be re-compressed to a given tolerance using the compression scheme presented in Section 5.3.3.

5.4.2 $\mathcal{H}$-matrix-vector Product

Computing the product of a general $\mathcal{H}$-matrix with a given vector, that is to compute

$$\underline{b} = \boldsymbol{A}_{\mathcal{H}} \underline{x} \tag{5.12}$$

where $\boldsymbol{A}_{\mathcal{H}}$ is an $\mathcal{H}$-matrix, and where $\underline{b}$ and $\underline{x}$ are standard vectors (arrays), is most easily done recursively. An easy understanding of the procedure is to consider a blockwise interpretation of the matrix-vector-product operation. If we assume that the four children of $\boldsymbol{A}_{\mathcal{H}}$ (should they exist) are denoted by $\boldsymbol{A}_{\mathcal{H},ij}$ where $i, j = 1, 2$ then the desired product is:

$$\boldsymbol{A}_{\mathcal{H}} \underline{x} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{H},11} & \boldsymbol{A}_{\mathcal{H},12} \\ \boldsymbol{A}_{\mathcal{H},21} & \boldsymbol{A}_{\mathcal{H},22} \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{H},11} \underline{x}_1 + \boldsymbol{A}_{\mathcal{H},12} \underline{x}_2 \\ \boldsymbol{A}_{\mathcal{H},21} \underline{x}_1 + \boldsymbol{A}_{\mathcal{H},22} \underline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix} \tag{5.13}$$

where $\underline{x}_1$ and $\underline{x}_2$ correspond to the rows of $\underline{x}$ associated with the column partitioning of $\boldsymbol{A}_{\mathcal{H}}$ and where $\underline{b}_1$ and $\underline{b}_2$ correspond to the row partitioning of $\boldsymbol{A}_{\mathcal{H}}$. This block-wise interpretation shows that the entries in $\underline{b}$ can be computed by recursive calls to smaller $\mathcal{H}$-matrix-vector products provided that results can be added together. For this reason, an $\mathcal{H}$-matrix implementation of a matrix vector product typically evaluates:

$$\underline{b} = \underline{b} + c\boldsymbol{A}_{\mathcal{H}}\underline{x} \tag{5.14}$$

where the factor c can be used to either add (c = 1) or subtract (c = -1) the results from $\underline{b}$ as desired.

The general procedure can now be described as:

- If $\boldsymbol{A}_{\mathcal{H}}$ is an $\mathcal{H}$-matrix, recursively call the matrix-vector-product for each child $\boldsymbol{A}_{\mathcal{H},ij}$, operating on the appropriate portions of $\underline{x}$ and adding the result (appropriately scaled by c) in the appropriate locations in $\underline{b}$.
- If $\boldsymbol{A}_{\mathcal{H}}$ is an $\mathcal{R}_k$-matrix, efficiently calculate $\boldsymbol{A}_{\mathcal{H}}\underline{x} = \boldsymbol{U}_A(\boldsymbol{U}_V^H\underline{x})$ and add it appropriately to the result. Note that based on the dimensions of $\boldsymbol{U}_A$ and $\boldsymbol{V}_A$, the order of operations (indicated by parenthesis here), is important to ensure an efficient implementation. The computational cost of this product is $\mathcal{O}(k(n+m))$ for an $\mathcal{R}_k$-matrix with dimensions $m \times n$ and rank k [49].
- If $\boldsymbol{A}_{\mathcal{H}}$ is a standard (full) matrix, calculate $\boldsymbol{A}_{\mathcal{H}}\underline{x}$ directly and add it appropriately to the result.

The base cases of this recursion are the $\mathcal{R}_k$-matrix-vector product and/or standard matrix-vector products incurred at leaf nodes of the $\mathcal{H}$-matrix. Note that vector-matrix products, i.e., computing $\underline{b}^T = \underline{x}^T\boldsymbol{A}_{\mathcal{H}}$ can be handled analogously. For square matrices, the general complexity of the $\mathcal{H}$-matrix-vector product is $\mathcal{O}(n \log n)$ compared to standard (dense) matrix-vector products with a cost that is $\mathcal{O}(n^2)$. More precisely the complexity of the $\mathcal{H}$-matrix-vector product is $\mathcal{O}(k(|\tau| + |\sigma|))$ where $(\tau, \sigma) \in T_I \times T_I$ are a pair of admissible blocks and k is the rank of the matrix [49].

5.4.3 $\mathcal{H}$-matrix-$\mathcal{H}$-matrix Products

Another operation required for $\mathcal{H}$-matrix factorizations and/or inversions is the computation of the product of two $\mathcal{H}$-matrices. We assume we are interested in computing the product

$$\boldsymbol{C}_{\mathcal{H}} = \boldsymbol{A}_{\mathcal{H}}\boldsymbol{B}_{\mathcal{H}} \tag{5.15}$$

and impose the constraint that the $\mathcal{H}$-matrix structure of $\boldsymbol{C}_{\mathcal{H}}$, $\boldsymbol{A}_{\mathcal{H}}$, and $\boldsymbol{B}_{\mathcal{H}}$ are the same[2]. Corresponding block structures are assumed because they are imposed for both $\mathcal{H}$-matrix LU decomposition and inversion.

The product of two $\mathcal{H}$-matrices is arguably the most complicated feature of $\mathcal{H}$-matrix arithmetic. Thankfully, an understanding and proper implementation of this operation makes both LU decomposition and inversion straightforward.

As in the case of a matrix-vector-product, we leverage the fact that $\mathcal{H}$-matrices are subdivided according to their block cluster tree structure. We have:

$$\begin{aligned}\boldsymbol{C}_{\mathcal{H}} = \begin{bmatrix} \boldsymbol{C}_{\mathcal{H},11} & \boldsymbol{C}_{\mathcal{H},12} \\ \boldsymbol{C}_{\mathcal{H},21} & \boldsymbol{C}_{\mathcal{H},22} \end{bmatrix} = \boldsymbol{A}_{\mathcal{H}}\boldsymbol{B}_{\mathcal{H}} &= \begin{bmatrix} \boldsymbol{A}_{\mathcal{H},11} & \boldsymbol{A}_{\mathcal{H},12} \\ \boldsymbol{A}_{\mathcal{H},21} & \boldsymbol{A}_{\mathcal{H},22} \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_{\mathcal{H},11} & \boldsymbol{B}_{\mathcal{H},12} \\ \boldsymbol{B}_{\mathcal{H},21} & \boldsymbol{B}_{\mathcal{H},22} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{A}_{\mathcal{H},11}\boldsymbol{B}_{\mathcal{H},11} + \boldsymbol{A}_{\mathcal{H},12}\boldsymbol{B}_{\mathcal{H},21} & \boldsymbol{A}_{\mathcal{H},11}\boldsymbol{B}_{\mathcal{H},12} + \boldsymbol{A}_{\mathcal{H},12}\boldsymbol{B}_{\mathcal{H},22} \\ \boldsymbol{A}_{\mathcal{H},21}\boldsymbol{B}_{\mathcal{H},11} + \boldsymbol{A}_{\mathcal{H},22}\boldsymbol{B}_{\mathcal{H},21} & \boldsymbol{A}_{\mathcal{H},21}\boldsymbol{B}_{\mathcal{H},12} + \boldsymbol{A}_{\mathcal{H},22}\boldsymbol{B}_{\mathcal{H},22} \end{bmatrix}\end{aligned} \tag{5.16}$$

[2]Having the same structure does not place constraints on the rank of the product, which may increase or decrease as necessary for adaptive $\mathcal{H}$-matrix operations

or simply,

$$\boldsymbol{C}_{\mathcal{H},ij} = \boldsymbol{A}_{\mathcal{H},i1}\boldsymbol{B}_{\mathcal{H},1j} + \boldsymbol{A}_{\mathcal{H},i2}\boldsymbol{B}_{\mathcal{H},2j} \tag{5.17}$$

As in the case of the matrix-vector-product, flexibility is built into this calculation by evaluating it as:

$$\begin{aligned}\boldsymbol{C}_{\mathcal{H},ij} &= \boldsymbol{C}_{\mathcal{H},ij} + \boldsymbol{A}_{\mathcal{H},i1}\boldsymbol{B}_{\mathcal{H},1j}\\ \boldsymbol{C}_{\mathcal{H},ij} &= \boldsymbol{C}_{\mathcal{H},ij} + \boldsymbol{A}_{\mathcal{H},i2}\boldsymbol{B}_{\mathcal{H},2j}\end{aligned} \tag{5.18}$$

and so we benefit by simply implementing a routine for computing

$$\boldsymbol{C}_{\mathcal{H}} = \boldsymbol{C}_{\mathcal{H}} + c\boldsymbol{A}_{\mathcal{H}}\boldsymbol{B}_{\mathcal{H}} \tag{5.19}$$

for general $\mathcal{H}$-matrices and a constant c.

To describe the general procedure we break the procedure into three steps:

1. We will first describe computing the product $\boldsymbol{A}_{\mathcal{H}}\boldsymbol{B}_{\mathcal{H}}$ based on the types $(\mathcal{H}, \mathcal{F}, \mathcal{R})$ of the matrices (scaling by a constant is trivial).
2. We will next address the evaluation of the sum of two $\mathcal{H}$-matrices.

5.4.3.1 Multiplying $\mathcal{H}$-matrices

The approach required to multiply two $\mathcal{H}$-matrices is dependent on the types of the matrices. There are 9 possibilities and they are handled as follows:

- $[\mathcal{H} \times \mathcal{H}]$: A recursive call according to the structure given in (5.16) is made.
- $[\mathcal{F} \times \mathcal{F}]$, $[\mathcal{R} \times \mathcal{F}]$, $[\mathcal{F} \times \mathcal{R}]$, $[\mathcal{R} \times \mathcal{R}]$: These case refers to any combination of $\mathcal{R}_k$-matrices of standard (full) matrices being present multiplied. In these cases standard matrix arithmetic can be used.

- $[\mathcal{H} \times \mathcal{F}]$, $[\mathcal{H} \times \mathcal{R}]$, $[\mathcal{R} \times \mathcal{H}]$, $[\mathcal{F} \times \mathcal{H}]$: As full matrices are small, and as efficient rank-compressed matrices have at least one small dimension k, each of these cases can be efficiently handled by multiple applications of $\mathcal{H}$-matrix-vector products (or vector-matrix products). For example if $\boldsymbol{A}_{\mathcal{H}}$ is an $\mathcal{H}$-matrix, and $\boldsymbol{B}_{\mathcal{H}}$ is an $\mathcal{R}_k$-matrix, then we seek to compute

$$\boldsymbol{A}_{\mathcal{H}} \boldsymbol{U}_B \boldsymbol{V}_B^H = \left(\sum_{k=1}^{k_B} \boldsymbol{A}_{\mathcal{H}} \underline{u}_{B,k} \right) \boldsymbol{V}_B^H \tag{5.20}$$

  where $\underline{u}_{B,k}$ is the kth column or $\boldsymbol{U}_B$ which has k_B total columns. The quantity in parentheses is simply multiple matrix-vector-products. The net result of any of these operations is either $\mathcal{R}$ or $\mathcal{F}$.

5.4.3.2 Adding $\mathcal{H}$-matrices

The approach required to add two $\mathcal{H}$-matrices is again dependent on the types of the matrices. We assume that when adding two matrices the structure of the left-operand is the target structure. For example $\mathcal{H} + \mathcal{R}$ should result in a $\mathcal{H}$ structure. This is consistent with the operations required for matrix-matrix-products according to (5.19). As the left operand dictates the output format, there remain 9 cases, despite the fact that matrix addition is commutative.

- $[\mathcal{H} + \mathcal{H}]$: A recursive call according to the blockwise structure is made.
- $[\mathcal{H}+\mathcal{R}]$, $[\mathcal{H}+\mathcal{F}]$ In these cases, the full or rank-compressed matrix formats must be distributed to the leaf nodes of the matrix in $\mathcal{H}$ format and added to the leaf-node data. This simple operation is referred to as deglommeration [19].
- $[\mathcal{R} + \mathcal{H}]$, $[\mathcal{F} + \mathcal{H}]$: In these cases, the matrix subblocks stored in the leaf nodes of $\mathcal{H}$ must be converted to either $\mathcal{R}$ or $\mathcal{F}$ matrices. This simple operation is referred to as agglomeration [19].

- $[\mathcal{R}+\mathcal{F}]$, $[\mathcal{R}+\mathcal{R}]$, $[\mathcal{F}+\mathcal{R}]$, $[\mathcal{F}+\mathcal{F}]$: These cases are treated directly.

The complexity of $\mathcal{H}$-matrix multiplication, for is $\mathcal{O}(k^2 n \log^2(n))$ where k denotes the (assumed block-wise constant) rank of the matrix. Proof and details of the complexity can be found in [19].

5.5 $\mathcal{H}$-Matrix Coarsening

Once an $\mathcal{H}$-matrix has been built, or calculated from products or factorizations etc., there may be an opportunity to reduce storage requirements and increase performance by simplifying the matrix structure. Effectively this is accomplished by agglomeration [19], the same operation that is required for $\mathcal{H}$-matrix addition described in 5.4.3.2. The procedure is simple: a depth-first search of the $\mathcal{H}$-matrix tree is performed. At each node, the memory requirements for storing all child information as an $\mathcal{R}_k$-matrix (agglomerating the children) is compared to the sum of the storage requirements for each child. If it is more efficient to agglomerate the children then they are agglomerated and removed from the structure. To ensure proper functionality of operations, the newly agglomerated block is marked as admissible. A simple illustrative example is provided in Figure 5.7.

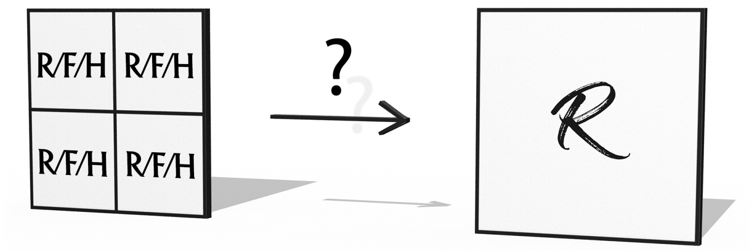


FIGURE 5.7: An illustration of $\mathcal{H}$-matrix agglomeration or coarsening; if memory is reduced, four children are combined to a single rank-compressed matrix.

5.6 Hierarchical Inversion

Although we do not consider approximate inversion of $\mathcal{H}$-matrices in the results present in Chapter 6, it is worth showing how to compute the inverse of an $\mathcal{H}$-matrix. Having a working knowledge of $\mathcal{H}$-matrix products, makes the presentation straightforward. Assume we have a general (not necessarily an $\mathcal{H}$-matrix) 2×2 block matrix $\boldsymbol{A}$ such that:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}$$

and assume that the 2×2 block matrix $\boldsymbol{B}$ is equal to the inverse of $\boldsymbol{A}$:

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}.$$

Then, by definition:

$$\begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \tag{5.21}$$

Using the Schur complement approach to solving a block system of equations [19] leads to the following sequence of operations:

1. $\boldsymbol{T}_{12} = \boldsymbol{T}_{12} - \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$

2. $\boldsymbol{T}_{21} = \boldsymbol{T}_{21} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$

3. $\boldsymbol{A}_{22} = \boldsymbol{A}_{22} + \boldsymbol{A}_{21}\boldsymbol{T}_{12}$

4. $\boldsymbol{B}_{22} = \boldsymbol{A}_{22}^{-1}$

5. $\boldsymbol{B}_{12} = \boldsymbol{B}_{12} + \boldsymbol{T}_{12}\boldsymbol{B}_{22}$

6. $\boldsymbol{B}_{21} = \boldsymbol{B}_{21} + \boldsymbol{B}_{22}^{-1}\boldsymbol{T}_{21}$

7. $\boldsymbol{B}_{11} = \boldsymbol{B}_{11} + \boldsymbol{T}_{12}\boldsymbol{B}_{21}$

where $\boldsymbol{T}$ is a temporary matrix with the same block-wise structure as $\boldsymbol{A}$ and $\boldsymbol{B}$. Note that in step 3 above, we destroy the contents of $\boldsymbol{A}$ to save storage requirements, and that after step 3, $\boldsymbol{A}_{22}$ contains the Schur complement of the 2×2 block matrix.

An implementation of $\mathcal{H}$-matrix inversion uses the preceding sequence of steps and proceeds recursively: if an operation is performed on an $\mathcal{H}$-matrix block, it is called recursively for the children should they exist. Operations at the leaf nodes are done using standard matrix inversion. $\mathcal{H}$-matrix products are called as required with the appropriate constant indicating addition or subtraction.

The complexity of the hierarchical inversion is bounded by the cost of $\mathcal{H}$-matrix multiplication. The cost complexity of $\mathcal{H}$-Inverse is bounded by $\mathcal{O}(k^2 n \log^2 n)$ for fixed-rank matrices having at most block-wise rank k. Proof and details of the complexity analysis can be found in [19].

5.7 Hierarchical LU Decomposition

Although the complexity of $\mathcal{H}$-matrix inversion is relativity low, the number of numerical computational steps to compute the overall inversion is still high. An alternative way of formulating the solution to an $\mathcal{H}$-matrix equation is to use LU decomposition [20] [19]. An $\mathcal{H}$-matrix LU factorization with high precision can be used as a fast direct solver [54] while lower precision can be used for preconditioning [55]. We refer to an $\mathcal{H}$-matrix LU decomposition as $\mathcal{H}$-LU. It can be shown that the complexity of an $\mathcal{H}$-LU factorization is $\mathcal{O}(k^2 n \log^2 n)$ [56].

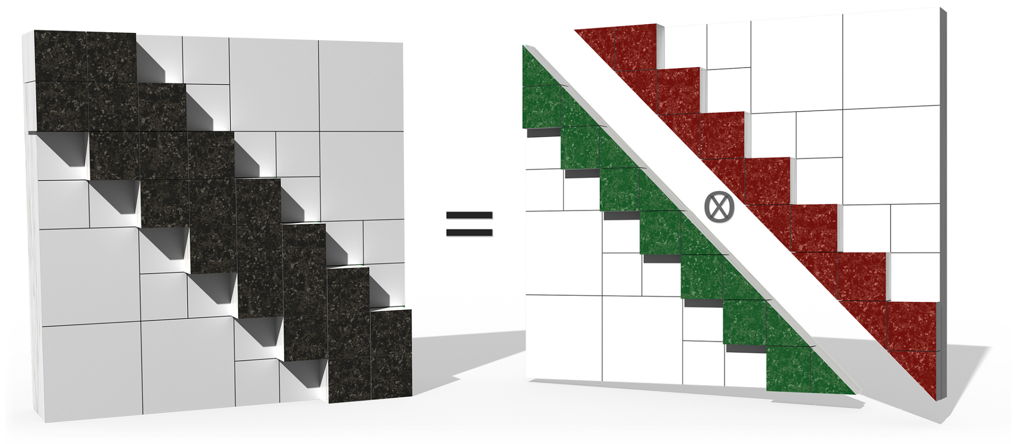


FIGURE 5.8: $\mathcal{H}$-LU and the hierarchical factors. An $\mathcal{H}$-LU factorization results in the decomposition of the $\mathcal{H}$-matrix (left) into the product of lower (green) and upper (red) triangular $\mathcal{H}$-matrices.

5.7.1 Obtaining Hierarchical L and U Factors

In Section 4.2 we described the general LU decomposition method. Like all other $\mathcal{H}$-matrix operations previously discussed, $\mathcal{H}$-LU exploits the block structure of the $\mathcal{H}$-matrix and recursion. For simplicity we limit consideration to the case where the LU factors have the same block structure as the original matrix. Assuming $\boldsymbol{L}_{\mathcal{H}}$ and $\boldsymbol{U}_{\mathcal{H}}$ are the factors for the matrix $\boldsymbol{A}_{\mathcal{H}}$, the steps required for hierarchical LU decomposition can be obtained directly from the block-wise factorization

$$\boldsymbol{A}_{\mathcal{H}} = \boldsymbol{L}_{\mathcal{H}}\boldsymbol{U}_{\mathcal{H}} \Longrightarrow \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{11} & \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_{11} & \boldsymbol{U}_{12} \\ & \boldsymbol{U}_{22} \end{bmatrix} \tag{5.22}$$

Note that $\boldsymbol{L}_{\mathcal{H}}$ and $\boldsymbol{U}_{\mathcal{H}}$ are lower and upper triangular $\mathcal{H}$-matrices respectively. This implies that $\boldsymbol{L}_{11}$ and $\boldsymbol{L}_{22}$ are lower triangular while $\boldsymbol{U}_{11}$ and $\boldsymbol{U}_{22}$ are upper triangular. Expanding the equations in (5.22) leads to the following steps for computing the LU decomposition:

1. Perform $\mathcal{H}$-LU factorization on $\boldsymbol{A}_{11}$ to get $\boldsymbol{L}_{11}\boldsymbol{U}_{11}$. If a recursive call is necessary, then apply it. The base case is that $\boldsymbol{A}_{11}$ has no children in which case the sub-block must be a full matrix because it lies on the diagonal and the admissible condition for diagonal blocks can never be satisfied. In this case, we perform a standard LU decomposition to determine $\boldsymbol{L}_{11}$ and $\boldsymbol{U}_{11}$.

2. Given $\boldsymbol{L}_{11}$ solve $\boldsymbol{A}_{12} = \boldsymbol{L}_{11}\boldsymbol{U}_{12}$ for $\boldsymbol{U}_{12}$. This requires solving a lower triangular system of equations, which for $\mathcal{H}$-matrices is described in Section 5.7.3.

3. Given $\boldsymbol{U}_{11}$ solve $\boldsymbol{A}_{21} = \boldsymbol{L}_{21}\boldsymbol{U}_{11}$ for $\boldsymbol{L}_{21}$. This requires solving a (right-sided) upper triangular system of equations and can be cast through a transposition as a left-sided lower-triangular solve.

4. Perform $\mathcal{H}$-LU factorization on $\boldsymbol{A}_{22} - \boldsymbol{L}_{21}\boldsymbol{U}_{12}$ to determine $\boldsymbol{L}_{22}$ and $\boldsymbol{U}_{22}$. If a recursive call is required, apply it.

Notice that this algorithm requires $\mathcal{H}$-matrix products, recursive calls to the $\mathcal{H}$-LU factorization. and a standard LU factorization routine in steps 1 and 4.

5.7.2 Comments on Pivoting Strategies

As we're dealing with $\mathcal{H}$-matrix representations of sparse matrices, pivoting strategies that aim to reduce the amount of fill-in during LU decomposition can be very beneficial [57]. However, pivoting strategies for $\mathcal{H}$-matrices add an additional layer of complication to the overall $\mathcal{H}$-matrix structure and routines, and are considered beyond the scope of this thesis.

5.7.3 Hierarchical Triangular Solvers

Triangular solvers are required both for computing the LU decomposition (which requires block solutions) as discussed in Section 5.7.1, as well as for applying the final LU decomposition to solving linear systems as described in Section 4.2. The solvers effectively perform block-wise forward or backwards substitution [58]. For brevity we present the block-wise Hierarchical forward substitution routine (lower triangular solver). The forward substitution routine operating on a vector is analogous [58] and the required upper triangular routines follow similar implementations [58].

Consider solving the lower-triangular equations $\boldsymbol{L}\boldsymbol{X} = \boldsymbol{B}$ where $\boldsymbol{L}$ is a lower triangular 2×2 block matrix, $\boldsymbol{B}$ is a known 2×2 block matrix, having the same structure as the unknown matrix $\boldsymbol{X}$. That is, the unknowns and the right-hand-side have the same structure, but are not required to have the same structure as the system $\boldsymbol{L}$. This setup is consistent with the hierarchical LU decomposition routine presented in Section 5.7.1. Then:

$$\begin{bmatrix} \boldsymbol{L}_{11} & \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_{11} & \boldsymbol{X}_{12} \\ \boldsymbol{X}_{21} & \boldsymbol{X}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}$$

and the solution can be obtained by (recursively) applying the following steps:

1. Solve $\boldsymbol{L}_{11}\boldsymbol{X}_{11} = \boldsymbol{B}_{11}$ to get $\boldsymbol{X}_{11}$
2. Solve $\boldsymbol{L}_{11}\boldsymbol{X}_{12} = \boldsymbol{B}_{12}$ to get $\boldsymbol{X}_{12}$
3. Solve $\boldsymbol{L}_{22}\boldsymbol{X}_{21} = \boldsymbol{B}_{21} - \boldsymbol{L}_{21}\boldsymbol{X}_{11}$ to get $\boldsymbol{X}_{21}$
4. Solve $\boldsymbol{L}_{22}\boldsymbol{X}_{22} = \boldsymbol{B}_{22} - \boldsymbol{L}_{21}\boldsymbol{X}_{12}$ to get $\boldsymbol{X}_{22}$

Note that the base case for each of the above recursive calls should be computed considering the sub-blocks type of $\boldsymbol{L}$, $\boldsymbol{X}$ and $\boldsymbol{B}$.

5.8 Chapter Summary

Based on the formulations provided in this chapter, and the GMRES formulation and DGM system presented in Chapters 3 and 4, we are now prepared to apply $\mathcal{H}$-matrices either for direct solutions or to preconditioning iterative solutions to the DGM system. Performance results and scaling are presented in the next chapter.

Chapter 6

Numerical Results

The goal of this thesis is to demonstrate a successful implementation of $\mathcal{H}$-matrices and to apply it to the system of equations arising from time-harmonic discontinuous Galerkin methods. In this chapter we provide the numerical simulation results validating the implementation and performance as compared to both built-in MATLAB solution methods (without $\mathcal{H}$-matrices) and expected complexity scaling.

6.1 DGM Problem Specifications

For a given 2D TM DGM problem, formulated on a mesh Ω_h, there are a number of physical and numerical parameters that should be considered. These are:

- the frequency f of the time-harmonic fields
- the mesh density h which implies a number of mesh elements N_V
- the solution order p which implies N_p nodes per element
- the geometry of the mesh, scatterers $\varepsilon(\vec{x})$ and $\mu(\vec{x})$, and sources

- the choice of absorbing boundary conditions

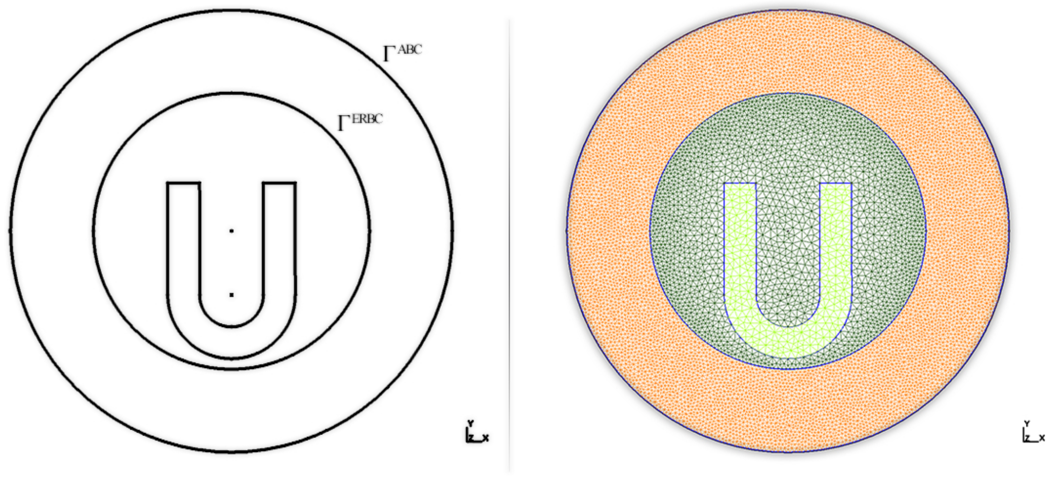


FIGURE 6.1: The electromagnetic geometry for numerical testing: (left) a U-shaped target represented in Gmsh [36] with ERBC surface and ABC surface, the relative permittivity of the target is $\varepsilon = 2.0$; (right) a sample mesh discretization for the problem geometry.

To limit the scope of producing numerical results, we only consider a single mesh geometry, shown Figure 6.1. This problem consists of a single U-shaped target having a relative complex permittivity of $\varepsilon = 2.0 - j0.0$. The background medium is assumed to be free space. A single line source located at $(x, y) = (0.18, 0)$m is the source for all problems. In all cases, ERBCs are applied as boundary conditions. While selecting a single, relatively simple problem is not all that exciting from an applications perspective, the selected problem is representative of an electromagnetic imaging configuration. Besides, we are only concerned with validating the $\mathcal{H}$-matrix performance and as is shown, the remaining parameters provide significant variation in testing. Specifically we test h-refinement (increasing the number of mesh elements), p-refinement (increasing the solution order), and the effects of increasing frequency (more complicated field patterns) on the block-wise rank of the $\mathcal{H}$-matrices.

6.2 Iterative Solution Specifications

All tests presented herein constitute applying $\mathcal{H}$-LU preconditioned GMRES to solving the 2D TM DGM electric vector wave equation (3.10) presented in Chapter 3. In order to analyze the performance of this approach we vary both the adaptive $\mathcal{H}$-matrix error tolerance ϵ and the admissible condition η. The number of elements in each leaf node of the $\mathcal{H}$-cluster tree N_{max} can also be varied and experiments have shown the benefits of small modifications of the parameter.

The $\mathcal{H}$-matrix framework was implemented in MATLAB. Limitations of this implementation include the inability to dynamically allocate memory, forcing potential memory thrashing when constructing systems. However, the benefits of optimized built-in MATLAB routines for SVD, GMRES, and so on greatly simplified the implementation. Another limitation in our implementation is the use of SVD for rank compression. Other methods such as ACA would likely speed up the implementation. Further, as discussed in Chapters 4 and 5 we do not employ a pivoting strategy. For fair comparisons with direct methods supplied by MATLAB we also enforce no pivoting strategy for MATLAB routines.

In MATLAB you may specify a number of a parameters for GMRES including the desired error tolerance, the total number of iterations, the restart m, and an LU-decomposed preconditioner [23]. As is shown, the performance of $\mathcal{H}$-LU as a preconditioner is quite good, providing convergence to 10^{-7} relative error in just a few iterations. Consequently we fix the GMRES error tolerance 10^{-7} for all examples.

All the simulations were performed on a Windows 10 machine with an Intel Core i7-5820K Haswell-E 6-Core Desktop Processor CPU @ Over-clocked 4.2Ghz, 32 GB of DDR4 RAM and M.2 storage.

6.3 $\mathcal{H}$-matrix Tolerance Testing

In this section we present the results for scaling the $\mathcal{H}$-matrix tolerance ϵ on the performance of the $\mathcal{H}$-LU preconditioner. All other solution parameters are fixed. Of note, the frequency f was chosen as 2 GHz, giving the entire domain Ω_h an electrical diameter of approximately 2 wavelengths. The mesh consisted of $30,250$ elements and a first-order solution, $p = 1$ and $N_p = 3$ was selected. The total degrees of freedom (matrix dimension size) is $n = 90,750$. The mesh contained 252 ERBC edges and 378 ABC edges. The admissible condition was selected as $\eta = 2$ and $N_{max} = 12$ was chosen. After construction of the $\mathcal{H}$-matrix, coarsening (agglomeration) as described in Section 5.5 is applied with the same tolerance as originally used to build the matrix to reduce storage and time costs [19]. Relevant evaluated metrics are broken into three categories: memory usage, accuracy, and speed/time and are summarized as:

- Memory: $\mathcal{H}$-matrix (memory to store the $\mathcal{H}$-matrix), Aggl$\mathcal{H}$-matrix (memory to store the coarsened matrix), $\mathcal{H}$-L (memory for $\boldsymbol{L}$) and $\mathcal{H}$-U (memory for $\boldsymbol{U}$)
- Accuracy: $\mathcal{H}k_{max}$ (maximum rank in the $\mathcal{H}$-matrix), Agglk_{max} (maximum rank in the coarsened matrix), $\mathcal{H}$-Lk_{max} (maximum rank in $\boldsymbol{L}$), $\mathcal{H}$-Uk_{max} (maximum rank in $\boldsymbol{U}$), $\mathcal{H}$-L Err (relative error in a lower triangular solve), and $\mathcal{H}$-U Err (relative error in an upper triangular solve)
- Time: $\mathcal{H}$ (time to build the $\mathcal{H}$-matrix, Aggl$\mathcal{H}$ (time to coarsen the matrix), LU (time to perform MATLAB LU), $\mathcal{H}$-LU (time for $\mathcal{H}$-matrix LU), GMRES (total time for all GMRES iterations), Iter (total number of GMRES iterations)

Note that the metrics $\mathcal{H}$-L/U Err are computed by solving a lower/upper triangular system of equations using the $\mathcal{H}$-LU decomposition for a random right-hand-side and comparing the result to a MATLAB computed direct LU decomposition solution on the original DGM matrix.

Tables 6.1, 6.2 and 6.3 show the effects of varying the tolerance ϵ from 10^{-1} to 10^{-8} on these performance metrics.

Tol ϵ	**$\mathcal{H}$-matrix (MB)**	**Aggl$\mathcal{H}$(MB)**	**$\mathcal{H}$-L (MB)**	**$\mathcal{H}$-U (MB)**
0.1	770.23	282.19	380.76	399.37
0.01	770.71	284.70	418.70	440.35
0.001	771.53	285.52	461.12	478.18
0.0001	772.18	286.16	501.04	517.50
1e-05	772.98	286.97	543.82	554.30
1e-06	773.66	287.64	583.32	593.09
1e-07	774.49	288.47	622.97	630.03
1e-08	775.18	289.16	660.71	665.32

TABLE 6.1: $\mathcal{H}$-matrix memory scaling as a function of tolerance.

Tol ϵ	**$\mathcal{H}$**k_{max}	**Aggl** k_{max}	**$\mathcal{H}$-L** k_{max}	**$\mathcal{H}$-U** k_{max}	**$\mathcal{H}$-L Err**	**$\mathcal{H}$-U Err**
0.1	3.00	3.00	3.00	3.00	0.0564	0.0223
0.01	4.00	4.00	7.00	6.00	0.0048	0.0023
0.001	6.00	6.00	10.00	9.00	5.6e-04	2.5e-04
0.0001	8.00	8.00	13.00	11.00	6.4e-05	2.7e-05
1e-05	10.00	10.00	15.00	14.00	9.4e-06	3.1e-06
1e-06	11.00	11.00	18.00	16.00	1.0e-06	3.3e-07
1e-07	13.00	13.00	19.00	19.00	1.0e-07	3.2e-08
1e-08	14.00	14.00	21.00	20.00	9.4e-09	3.2e-09

TABLE 6.2: $\mathcal{H}$-matrix accuracy as a function of tolerance.

Tol	$\mathcal{H}$(s)	Aggl (s)	LU (s)	$\mathcal{H}$-LU (s)	GMRES (s)	Iter
0.1	29.17	25.78	126.43	44.42	47.68	7.00
0.01	97.75	28.79	126.83	53.83	32.17	4.00
0.001	111.79	29.08	127.28	62.18	21.45	3.00
0.0001	132.55	30.36	126.56	65.16	21.42	2.00
1e-05	163.73	31.95	126.28	77.28	16.40	2.00
1e-06	164.83	31.25	126.64	79.62	16.43	2.00
1e-07	191.07	31.55	126.36	85.75	16.54	2.00
1e-08	201.83	32.10	127.19	93.57	16.40	1.00

TABLE 6.3: $\mathcal{H}$-matrix computational time scaling as a function of tolerance.

The first conclusion to draw from the results is that the implementation is error controllable. Indeed, the LU decomposition error shown in Table 6.2 shows accuracy below the requested tolerance for all cases. The next interesting conclusion to draw is that the convergence of GMRES in terms of total number of iterations varies only slightly as the tolerance is decreased as shown in the last column of Table 6.3. Certainly the $\mathcal{H}$-LU preconditioner for all selected tolerance levels is sufficient to guarantee convergence in a handful of iterations. Another interesting observation is that the time to construct the $\mathcal{H}$ and to perform $\mathcal{H}$-LU increases with decreased tolerance as expected, but that the memory required to store the $\mathcal{H}$-matrix does not as seen in Table 6.1. This can be attributed to the fact that the DGM system matrix is quite sparse, and that large portions of the ERBC compress quite well. Finally we note from Table 6.3 the total time to perform the $\mathcal{H}$-LU and GMRES iterations is comparable to the time required for MATLAB's LU decomposition (without pivoting).

At this point we have confidence that our implementation is functional, and can proceed to validating the effects of h- and p-refinement.

6.4 h-Refinement

To test the effects of h-refinement on the $\mathcal{H}$-LU preconditioned GMRES solution to the 2D TM DGM equations, we produce four different meshes for the problem studied in the previous section. For these tests the $\mathcal{H}$-matrix tolerance was chosen as $\epsilon = 10^{-3}$ based on the results in Table 6.3. In fact, this tolerance is fixed for the remainder of the results presented in this work. Note that by increasing the mesh density, we effectively increase the number of ERBC and ABC edges in the mesh. The results are summarized in Tables 6.4, 6.5 and 6.6. Note that in the second table we have added the total number of degrees of freedom (DOF) resulting from the number of elements N_V times the number of nodes for the assumed first-order $p = 1$ solution.

Elements	$\mathcal{H}$(MB)	**Aggl** (MB)	$\mathcal{H}$-**L** (MB)	$\mathcal{H}$-**U** (MB)
4970	79.52	37.74	62.22	65.80
19598	355.56	148.52	248.89	261.93
43658	750.95	298.77	534.03	567.71
78654	1499.28	591.58	1017.65	1069.95

TABLE 6.4: $\mathcal{H}$-matrix memory scaling as a function of element size.

Elements	**DOF**	$\mathcal{H} k_{max}$	**Aggl** k_{max}	$\mathcal{H}$-**L** k_{max}	$\mathcal{H}$-**U** k_{max}
4970	14910	4.00	4.00	7.00	7.00
19598	58794	4.00	4.00	8.00	7.00
43658	130974	4.00	4.00	7.00	7.00
78654	235962	4.00	4.00	7.00	7.00

TABLE 6.5: $\mathcal{H}$-matrix accuracy as a function of element size.

Elements	$\mathcal{H}$(s)	Aggl (s)	LU (s)	$\mathcal{H}$-LU (s)	GMRES (s)	Iter
4970	2.08	3.22	3.67	7.06	0.72	3.00
19598	17.77	13.90	55.63	26.48	3.70	3.00
43658	88.97	37.28	324.44	58.69	36.19	3.00
78654	273.75	93.13	932.32	135.71	128.76	3.00

TABLE 6.6: $\mathcal{H}$-matrix computational time scaling as a function of element size.

According to standard complexity analysis on $\mathcal{H}$-matrix operations, the complexity of $\mathcal{H}$-LU is $\mathcal{O}(k^2 n \log^2 n)$ [56] where n is the number of DOF. As the frequency for this problem is fixed, increasing the mesh density should not greatly increase the overall block-wise rank considering the fact that the leaf-node sizes are fixed at $N_{max} = 12$ elements. From Table 6.6 the $\mathcal{H}$-LU time scales approximately linearly with the number of elements (and hence the DOF) as expected. For example, going from $43,658$ elements to $78,654$ elements results in a time scaling of 2.3 times. A bothersome result is that the GMRES time does not scale almost linearly. At the time of writing this thesis, it is unclear why this is the case. It could be that our $\mathcal{H}$-LU triangular solvers are inefficient, or perhaps some way that MATLAB's GMRES is handling the preconditioning steps.

6.5 p-Refinement

As DGM is a higher order solver we have the capability of increasing the polynomial expansion order p. This section presents results for p-refinement. We note that we are not interested in the overall accuracy of the pth order DGM method itself, but rather only care about the comparison of $\mathcal{H}$-LU to direct methods for a fixed order. To explore the effects of p-refinement we consider a mesh having $19,598$ elements at different expansion orders. All other problem parameters are the same as previous examples, specifically N_{max} is still 12. This implies that the overall block sizes of leaf nodes in the $\mathcal{H}$-matrix approximation to the DGM system get larger as p increases. Results are summarized in Tables 6.7, 6.8 and 6.9. Of interest is that we have also added the memory requirements for MATLAB built-in LU decomposition (without pivoting) in Table 6.7.

Order	**$\mathcal{H}$**(GB)	**Aggl** (GB)	**L** (GB)	**U** (GB)	**$\mathcal{H}$-L** (GB)	**$\mathcal{H}$-U** (GB)
1	0.35	0.15	0.95	0.95	0.25	0.27
2	0.39	0.18	2.89	4.33	0.46	0.54
3	0.46	0.24	6.64	9.97	0.86	1.10
4	0.57	0.35	8.64	19.46	1.50	2.09
5	0.75	0.53	-	-	2.49	3.72
6	1.02	0.78	-	-	3.90	6.23
7	1.39	1.14	-	-	5.90	9.90

TABLE 6.7: $\mathcal{H}$-matrix memory scaling as a function of polynomial order.

Order	**DOF**	**$\mathcal{H}$**k_{max}	**Aggl** k_{max}	**$\mathcal{H}$-L** k_{max}	**$\mathcal{H}$-U** k_{max}
1	58794	5.00	5.00	10.00	8.00
2	117588	5.00	5.00	11.00	10.00
3	195980	5.00	5.00	13.00	11.00
4	293970	5.00	5.00	14.00	12.00
5	411558	5.00	5.00	15.00	13.00
6	548744	5.00	5.00	15.00	13.00
7	705528	5.00	5.00	17.00	13.00

TABLE 6.8: $\mathcal{H}$-matrix accuracy as a function of polynomial order.

Order	**$\mathcal{H}$(s)**	**Aggl (s)**	**LU (s)**	**$\mathcal{H}$-LU (s)**	**GMRES (s)**	**Iter**
1	10.86	13.04	56.25	28.63	3.95	3.00
2	39.49	23.43	326.33	42.48	13.96	3.00
3	62.28	45.04	1293.68	70.40	22.76	3.00
4	89.50	80.00	3821.58	122.20	77.40	3.00
5	123.54	117.64	-	209.76	77.45	4.00
6	169.92	247.50	-	356.91	126.14	5.00
7	230.51	452.64	-	634.86	186.49	6.00

TABLE 6.9: $\mathcal{H}$-matrix computational time scaling as a function of polynomial order.

Overall the tables summarize expected performance for $\mathcal{H}$-LU. and GMRES. As the degrees of freedom grow, the memory and time requirements grow almost linearly. Of particular interest is the fact that changing the solution order from 4 to 7 means roughly doubling the GMRES time and iterations. This result is consistent with expectations making it seem like the results of the previous section where GMRES did not perform as expected are suspect.

Most importantly, these tables illustrate the capabilities of $\mathcal{H}$-LU preconditioned GMRES to solve high-order ERBC-enabled DGM problems. MATLAB's built-in LU decomposition (without pivoting) fails to produce a decomposition in the available system memory after order 4 as shown in Tables 6.7 and 6.9. Furthermore, the $\mathcal{H}$-matrix computation time is at least an order of magnitude faster that the direct LU decomposition (without pivoting).

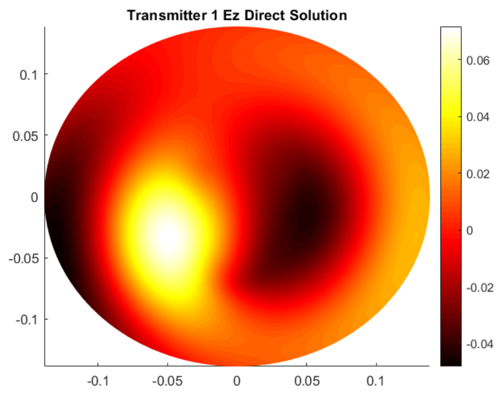
Note that in all cases the total number of GMRES iterations is very small, which indicates that additional savings may yet be available by increasing the $\mathcal{H}$-matrix tolerance. We have not pursued these tests.

6.6 Frequency Scaling - The Effect of Problem Rank

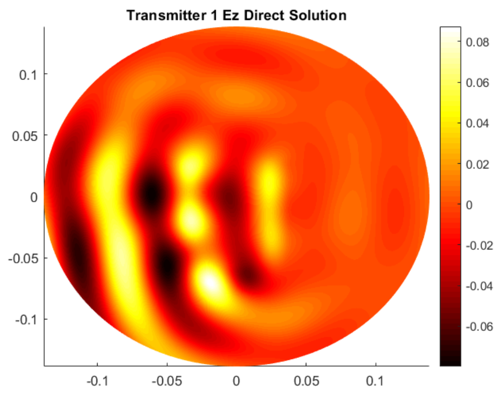
The results presented in this section correspond to changing the frequency f of the time-harmonic fields for a fixed problem geometry. Increasing the frequency reduces the wavelength and complicates the field pattern. Recovering more complicated fields ultimately implies that the block-wise rank of the DGM operator is increased. Thus, frequency scaling is roughly equivalent to increasing the overall block-wise rank. For this problem a mesh of 4970 elements and a $p = 6$th order expansion was used resulting in the total degrees of freedom being 139,160. For this problem, N_{max} was chosen as 10. Six frequencies, ranging from 500 MHz to 20 GHz and corresponding roughly to domain diameters of $1/2$ a wavelength to 20 wavelengths respectively. The resulting field patterns for select frequencies produced by a direct MATLAB solution (with pivoting) are shown for interests sake in Figure 6.2, while the $\mathcal{H}$-matrix solutions are not shown but are visibly indistinguishable. Tables 6.10, 6.11, and 6.12 summarize the results.

Frequency	**$\mathcal{H}$(MB)**	**Aggl (MB)**	**$\mathcal{H}$-L (MB)**	**$\mathcal{H}$-U (MB)**
500Mhz	253.77	204.44	1027.59	1682.31
1Ghz	254.14	204.81	1027.50	1690.20
2Ghz	254.97	205.63	1033.77	1703.15
5Ghz	255.87	206.53	1057.25	1728.09
10Ghz	256.79	207.45	1098.57	1769.62
20Ghz	258.52	209.19	1169.79	1836.94

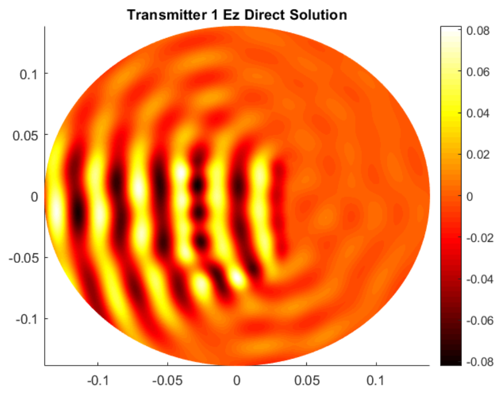
TABLE 6.10: $\mathcal{H}$-matrix memory scaling as a function of frequency.



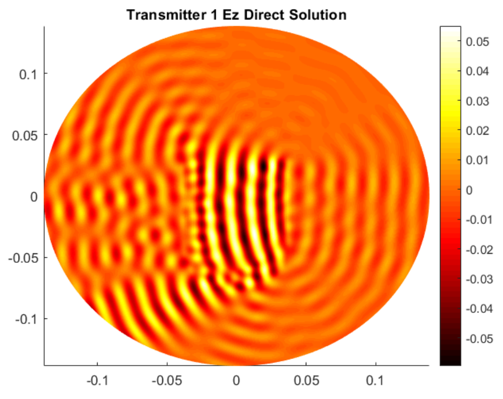
(a)



(b)



(c)



(d)

FIGURE 6.2: DGM field solution simulation results for various frequencies: a) 2 GHz, b) 5 GHz, c) 10 GHz and d) 20 GHz. Only the real part of the z-component of the electric field is shown.

Frequency	$\mathcal{H} k_{max}$	**Aggl** k_{max}	$\mathcal{H}$**-L** k_{max}	$\mathcal{H}$**-U** k_{max}
500Mhz	4.00	4.00	14.00	13.00
1Ghz	4.00	4.00	13.00	13.00
2Ghz	5.00	5.00	13.00	13.00
5Ghz	7.00	7.00	15.00	13.00
10Ghz	8.00	8.00	18.00	16.00
20Ghz	12.00	12.00	24.00	23.00

TABLE 6.11: $\mathcal{H}$-matrix accuracy as a function of frequency.

Frequency	$\mathcal{H}$ (s)	**Aggl** (s)	**LU (s)**	$\mathcal{H}$**-LU** (s)	**GMRES** (s)	**Iter**
500Mhz	19.00	59.08	1058.81	85.39	9.62	5.00
1Ghz	22.44	67.99	1058.89	87.57	6.88	3.00
2Ghz	22.45	68.67	1056.71	90.13	6.88	3.00
5Ghz	23.00	69.09	1073.45	91.88	5.60	3.00
10Ghz	23.95	75.82	1053.04	102.65	5.67	3.00
20Ghz	23.47	71.66	1055.21	104.08	5.73	3.00

TABLE 6.12: $\mathcal{H}$-matrix computational time scaling as a function of frequency.

The previous results can be summarized by noting that while the overall storage requirements for the $\mathcal{H}$-LU decomposition to not significantly increase, there is a small increase which can be correlated to the increased rank of the matrix and its factorization shown in Figure 6.11. The growth in the complexity is less than the growth expected if k_{max} was the same rank for all blocks. For example, moving from 2 GHz to 20 GHz increases the maximum rank from 5 to 12, and based on the complexity of $\mathcal{H}$-LU we would expect an increased time proportional to the square of this increase. However, the time to factor the system only slightly increases from 90 seconds to 104 seconds. The is largely due to the fact that average block-wise rank of the matrix is not the maximum rank of the matrix.

6.7 Chapter Summary

In this chapter we have verified the MATLAB implement of the $\mathcal{H}$-matrix developed in this work for solving 2D TM DGM systems. The results have demonstrated the error-controllable nature of the solution, the behaviour for both h-

and p-refinement, as well as looked at the effects of increasing problem frequency. The performance of the $\mathcal{H}$-matrix framework has been verified by comparing obtained results to the theoretical complexity. Most importantly the results have demonstrated that $\mathcal{H}$-matrix preconditioned GMRES can solve problems that direct LU factorization (without pivoting) cannot.

Chapter 7

Conclusions and Future Work

The focus of this work has been on understanding, implementing, and testing an $\mathcal{H}$-matrix software library programmed in MATLAB. The desire was to evaluate the performance of $\mathcal{H}$-matrices compared to other direct solution methods for solving systems of equations that arise from 2D TM ERBC-enabled DGM systems that might be found in electromagnetic imaging applications. Overall, the work has been educational, enjoyable, and difficult.

Previous chapters have aimed to show a concrete understanding of concepts related to electromagnetics[1], preconditioning iterative matrix solvers and the entire $\mathcal{H}$-matrix framework. Results demonstrate that the implementation works as expected.

A number of future improvements to the existing code should be pursued if the intention is to develop an in-house $\mathcal{H}$-matrix framework as opposed to leveraging the understanding of the framework that this thesis provides against existing commercial implementations [25]:

[1]Note that as a computer engineer this was initially completely foreign

- While MATLAB provides an excellent rapid prototyping environment, it is believed that control over dynamic memory provided by other languages would be beneficial.
- A major potential bottleneck in the developed code is the use of the singular value decomposition for rank compression. Alternative methods including ACA should be investigated.
- Next steps should also consider any number of high-performance computing options including shared-memory parallelism, co-processor acceleration, or distributed parallelism for large problems.
- Probably most importantly is that pivoting strategies for $\mathcal{H}$-matrices will likely greatly improve the overall performance should they be straightforward to implement. This concept has not been consider nor investigated herein.

It is hoped that the $\mathcal{H}$-matrix software developed in this work will have longevity as a prototyping and teaching tool. One of the most striking features of $\mathcal{H}$-matrices is that ultimately they can be used to attempt to accelerate any number of scientific computing applications, including both the electromagnetics and acoustics problems that are the current focus of the University of Manitoba's Electromagnetics Imaging Lab.

Bibliography

[1] G. C. Hsiao and R. E. Kleinman, "Mathematical foundations for error estimation in numerical solutions of integral equations in electromagnetics," *IEEE transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 316–328, 1997.

[2] P.-L. Rui, R.-S. Chen, Z. Liu, and Y.-N. Gan, "Schwarz-krylov subspace method for mlfmm analysis of electromagnetic wave scattering problems," *Progress In Electromagnetics Research*, vol. 82, pp. 51–63, 2008.

[3] B. Smith, P. Bjorstad, W. D. Gropp, and W. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.

[4] K. Zhao, M. N. Vouvakis, and J.-F. Lee, "The adaptive cross approximation algorithm for accelerated method of moments computations of emc problems," *IEEE transactions on electromagnetic compatibility*, vol. 47, no. 4, pp. 763–773, 2005.

[5] W. C. Gibson, *The method of moments in electromagnetics*. Chapman & Hall/CRC London, UK, 2008, vol. 1.

[6] W. A. Strauss, *Partial differential equations*. Wiley New York, 1992, vol. 92.

[7] M. Asefi, I. Jeffrey, J. LoVetri, C. Gilmore, P. Card, and J. Paliwal, "Grain bin monitoring via electromagnetic imaging," *Computers and Electronics in Agriculture*, vol. 119, pp. 133–141, 2015.

[8] P. Mojabi and J. LoVetri, "Ultrasound tomography for simultaneous reconstruction of acoustic density, attenuation, and compressibility profiles," *The Journal of the Acoustical Society of America*, vol. 137, no. 4, pp. 1813–1825, 2015.

[9] A. Baran, D. Kurrant, E. Fear, and J. LoVetri, "Immersion medium independent algorithm for breast microwave imaging," in *Radio Science Meeting (Joint with AP-S Symposium), 2015 USNC-URSI.* IEEE, 2015, pp. 303–303.

[10] G. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations, Applied Mathematical Series.* Wiley, New York, 1960.

[11] S. C. Brenner and C. Carstensen, "Finite element methods," *Encyclopedia of Computational Mechanics Second Edition*, pp. 1–47, 2017.

[12] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," *Handbook of numerical analysis*, vol. 7, pp. 713–1018, 2000.

[13] T. Y. Hou and R. Li, "Computing nearly singular solutions using pseudo-spectral methods," *Journal of Computational Physics*, vol. 226, no. 1, pp. 379–397, 2007.

[14] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications.* Springer Science & Business Media, 2007.

[15] F. Khalil, C. J. Barrios-Hernandez, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana, "Electromagnetic simulations via parallel computing: an application using scale changing technique for modeling of passive planar

reflectarrays in grid environment," in *Antennas and Propagation Society International Symposium, 2008. AP-S 2008. IEEE*. IEEE, 2008, pp. 1–4.

[16] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[17] J. Phillips and J. White, "Efficient capacitance extraction of 3d structures using generalized pre-corrected fft methods," in *Electrical Performance of Electronic packaging, 1994., IEEE 3rd Topical Meeting on*. IEEE, 1994, pp. 253–256.

[18] K. Stüben and U. Trottenberg, "Multigrid methods: Fundamental algorithms, model problem analysis and applications," in *Multigrid methods*. Springer, 1982, pp. 1–176.

[19] M. Bebendorf, *Hierarchical matrices*. Springer, 2008.

[20] W. Hackbusch, *Hierarchical matrices: algorithms and analysis*. Springer, 2015, vol. 49.

[21] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numerical Linear Algebra with Applications*, vol. 17, no. 6, pp. 953–976, 2010.

[22] I. Jeffrey, N. Geddert, K. Brown, and J. LoVetri, "The time-harmonic discontinuous galerkin method as a robust forward solver for microwave imaging applications," *Progress In Electromagnetics Research*, vol. 154, pp. 1–21, 2015.

[23] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[24] B. Kågström and C. F. Van Loan, *GEMM-based level-3 BLAS*. Cornell Theory Center, Cornell University, 1991.

[25] S. Börm and L. Grasedyck, "Hlib–a library for h-and h2-matrices, 1999," *URL http://www. hlib. org*.

[26] A. K. Saibaba, S. Ambikasaran, J. Yue Li, P. K. Kitanidis, and E. F. Darve, "Application of hierarchical matrices to linear inverse problems in geostatistics," *Oil and Gas Science and Technology-Revue de l'IFP-Institut Francais du Petrole*, vol. 67, no. 5, p. 857, 2012.

[27] B. Stroustrup, *The C++ programming language*. Pearson Education India, 2000.

[28] H. Bagherli and I. Jeffrey, "H-matrix compression of discontinous galerkin method exact radiating boundary conditions," in *Antenna Technology and Applied Electromagnetics (ANTEM), 2016 17th International Symposium on*. IEEE, 2016, pp. 1–2.

[29] R. F. Harrington, *Field computation by moment methods*. Wiley-IEEE Press, 1993.

[30] J. D. Jackson, *Classical electrodynamics*. John Wiley & Sons, 2012.

[31] M. M. Sadeghi, H. Nadgaran, and H. Chen, "Perfect field concentrator using zero index metamaterials and perfect electric conductors," *Frontiers of Physics*, vol. 9, no. 1, pp. 90–93, 2014.

[32] C. Brewitt-Taylor, "Limitation on the bandwidth of artificial perfect magnetic conductor surfaces," *IET microwaves, antennas & propagation*, vol. 1, no. 1, pp. 255–260, 2007.

[33] O. Zienkiewicz, D. Kelly, and P. Bettess, "The sommerfeld (radiation) condition on infinite domains and its modelling in numerical procedures," in *Computing Methods in Applied Sciences and Engineering, 1977, I*. Springer, 1979, pp. 169–203.

[34] D. K. Firsov and J. LoVetri, "Fvtd—integral equation hybrid for maxwell's equations," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 21, no. 1-2, pp. 29–42, 2008.

[35] N. C. Nguyen, J. Peraire, and B. Cockburn, "Hybridizable discontinuous galerkin methods," in *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 63–84.

[36] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities," *International journal for numerical methods in engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[37] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[38] B. Cockburn and C.-W. Shu, "The local discontinuous galerkin method for time-dependent convection-diffusion systems," *SIAM Journal on Numerical Analysis*, vol. 35, no. 6, pp. 2440–2463, 1998.

[39] H. Barucq, F. Delaurens, and B. Hanouzet, "Method of absorbing boundary conditions: phenomena of error stabilization," *SIAM journal on numerical analysis*, vol. 35, no. 3, pp. 1113–1129, 1998.

[40] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[41] J. D. Hoffman and S. Frankel, *Numerical methods for engineers and scientists*. CRC press, 2001.

[42] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*. McGraw-Hill New York, 1998, vol. 2.

[43] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[44] M. H. Gutknecht, "A brief introduction to krylov space methods for solving linear systems," in *Frontiers of Computational Science*. Springer, 2007, pp. 53–62.

[45] J. A. Meijerink and H. A. van der Vorst, "Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems," *Journal of computational physics*, vol. 44, no. 1, pp. 134–155, 1981.

[46] L. Grasedyck and W. Hackbusch, "Construction and arithmetics of h-matrices," *Computing*, vol. 70, no. 4, pp. 295–334, 2003.

[47] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM journal on matrix analysis and applications*, vol. 11, no. 3, pp. 430–452, 1990.

[48] R. D. Williams, "Performance of dynamic load balancing algorithms for unstructured mesh calculations," *Concurrency: Practice and experience*, vol. 3, no. 5, pp. 457–481, 1991.

[49] S. Börm, L. Grasedyck, and W. Hackbusch, "Introduction to hierarchical matrices with applications," *Engineering analysis with boundary elements*, vol. 27, no. 5, pp. 405–422, 2003.

[50] M. Bebendorf and S. Rjasanow, "Adaptive low-rank approximation of collocation matrices," *Computing*, vol. 70, no. 1, pp. 1–24, 2003.

[51] S. Börm and L. Grasedyck, "Hybrid cross approximation of integral operators," *Numerische Mathematik*, vol. 101, no. 2, pp. 221–249, 2005.

[52] M. Astner, H.-D. Bruns, G. Burger, and H. Singer, "Application of a hierarchical svd/aca compression technique to near-field calculations of monopole antennas," in *Electromagnetic Compatibility, 2007. EMC 2007. IEEE International Symposium on*. IEEE, 2007, pp. 1–6.

[53] A. Maltsev, V. Pestretsov, R. Maslennikov, and A. Khoryaev, “Triangular systolic array with reduced latency for qr-decomposition of complex matrices,” in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on.* IEEE, 2006, pp. 4–pp.

[54] H. Shao, J. Hu, H. Guo, F. Ye, W. Lu, and Z. Nie, “Fast simulation of array structures using t-epa with hierarchical lu decomposition,” *IEEE Antennas and Wireless Propagation Letters*, vol. 11, pp. 1556–1559, 2012.

[55] M. Bebendorf, “Hierarchical lu decomposition-based preconditioners for bem,” *Computing*, vol. 74, no. 3, pp. 225–247, 2005.

[56] T. Wan, X. Q. Hu, and R. S. Chen, “Hierarchical lu decomposition-based direct method with improved solution for 3d scattering problems in fem,” *Microwave and Optical Technology Letters*, vol. 53, no. 8, pp. 1687–1694, 2011.

[57] O. Schenk and K. Gärtner, “On fast factorization pivoting methods for sparse symmetric indefinite systems,” *Electronic Transactions on Numerical Analysis*, vol. 23, no. 1, pp. 158–179, 2006.

[58] M. R. Pino, L. Landesa, J. L. Rodriguez, F. Obelleiro, and R. J. Burkholder, “The generalized forward-backward method for analyzing the scattering from targets on ocean-like rough surfaces,” *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 6, pp. 961–969, 1999.