Algorithms for Electromagnetic Scattering Analysis of Electrically Large Structures

Borries, Oscar Peter

Publication date: 2015

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Algorithms for Electromagnetic Scattering Analysis of Electrically Large Structures

Oscar Borries

Kongens Lyngby 2014
PHD-2014-354
Abstract

Algorithms for Electromagnetic Scattering Analysis of Electrically Large Structures

Accurate analysis of electrically large antennas is often done using either Physical Optics (PO) or Method of Moments (MoM), where the former typically requires fewer computational resources but has a limited application regime. This study has focused on fast variants of these two methods, with the goal of reducing the computational complexity while maintaining accuracy.

Regarding MoM, the complexity is reduced by applying the Multi-Level Fast Multipole Method (MLFMM) in combination with an iterative solver. Using MLFMM with a MoM implementation based on Higher-Order (HO) basis functions has, by several authors, been dismissed as being too memory intensive. In the present work, we demonstrate for the first time that by including a range of both novel and previously presented modifications to the standard MLFMM implementation, HO MLFMM can achieve both memory reduction and significant speed increase compared to Lower-Order (e.g., RWG) based MLFMM. Further, issues surrounding an iterative solution, such as the iterative solver and preconditioning, are discussed. Numerical results demonstrate the performance and stability of the algorithm for very large problems, including full satellites at Ku band.

Accelerating PO is an entirely different matter. A few authors have discussed applying the Fast-PO technique to far fields, achieving relative errors of 0.1% – 1% for moderately sized scatterers. For near-fields, the state-of-the-art implementation of Fast-PO has several difficulties, in particular low accuracy and limited application regime. For the problems considered in this thesis, the error limit for PO is ≈ 0.01%, and the application limitations of the published Fast-PO are too prohibitive for our use. Therefore, results based on an improved Fast-PO implementation for far-fields, as well as a novel algorithm for near-fields, are presented. These results demonstrate that it is possible to achieve very accurate results, with relative errors around $10^{-5}$, at a much reduced time consumption. The method behind this part of the code is deemed confidential by TICRA.
Resumé (in Danish)

Algoritmer til Analyse af Elektromagnetisk Spredning fra Elektrisk Store Strukturer

Nøjagtige beregninger af elektrisk store antenner udføres ofte ved brug af Fysisk Optik (PO) eller Momentmetoden (MoM), hvor PO typisk kræver mindre beregningskraft men har et begrænset anvendelsesområde. Dette studie har fokuseret på accelererede varianter af PO og MoM, med det formål at reducere den beregningsmæssige kompleksitet uden at reducere nøjagtigheden.


Acceleration af PO er en helt anden sag. Enkelte forfattere har diskuteret anvendelsen af Fast-PO for fjernfeltsberegninger og har opnået relative fejl i størrelsesordenen 0.1% – 1% for strukturer af moderat størrelse. Anvendelsen af Fast-PO for nærfeltsberegninger har i litteraturen vist flere problemer, inklusive lav nøjagtighed. For de scenarier der studeres i denne afhandling er en fejl i størrelsesordenen 1% alt for høj for PO, og begrænsningerne i de publicerede Fast-PO algoritmer er for restriktive til vores brug. Derfor præsenteres resultater baseret på en forbedret implementation af Fast-PO til fjernfelter samt en helt ny algoritme til nærfelter. Disse resultater viser at det er muligt at opnå meget nøjagtige resultater, med relative fejl i størrelsesordenen 0.001%, samtidig med en væsentlig reduktion i beregningstiden. Detaljerne i denne del af koden er blevet dømt konfidentielle af TICRA.
Preface

The present work was carried out at DTU Compute and TICRA between December 2011 and December 2014 in partial fulfilment of the requirements for the Industrial Ph.D. Degree from the Technical University of Denmark.

The study was funded by TICRA with the support of the Danish Industrial Ph.D. program.

Lyngby, 15-December-2014

Oscar Borries
The general notation is as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Domain</th>
<th>Typeset</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar</td>
<td>$\mathbb{R}$ or $\mathbb{C}$</td>
<td>Minuscule</td>
<td>$e$</td>
</tr>
<tr>
<td>Physical vector</td>
<td>$\mathbb{R}^3$ or $\mathbb{C}^3$</td>
<td>Bold</td>
<td>$E$</td>
</tr>
<tr>
<td>Vector</td>
<td>$\mathbb{R}^m$ or $\mathbb{C}^m$</td>
<td>Overlined</td>
<td>$\vec{e}$</td>
</tr>
<tr>
<td>Matrix</td>
<td>$\mathbb{R}^{m \times n}$ or $\mathbb{C}^{m \times n}$</td>
<td>Double overlined</td>
<td>$\overline{E}$</td>
</tr>
<tr>
<td>Symbol</td>
<td>Name</td>
<td>Page</td>
<td>Equation</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>α</td>
<td>Relative weighting of EFIE in CFIE.</td>
<td>128</td>
<td>(C.19)</td>
</tr>
<tr>
<td>E</td>
<td>Electric field intensity.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ϵ</td>
<td>Material permittivity, in free-space</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_0 \approx 8.854187 \cdot 10^{-12} F/m$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>f</td>
<td>The frequency of the electromagnetic field.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>G</td>
<td>Green’s function.</td>
<td>127</td>
<td>(C.11)</td>
</tr>
<tr>
<td>$h^{(2)}_l(x)$</td>
<td>Spherical Hankel function of the second kind and order $l$ at $x$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T</td>
<td>Vector of excitation coefficients.</td>
<td>130</td>
<td>(C.27)</td>
</tr>
<tr>
<td>j</td>
<td>Imaginary unit $j^2 = -1$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>J, $J_S$</td>
<td>Surface current density.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>k</td>
<td>Wavenumber $k = 2\pi/\lambda$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K</td>
<td>Number of tabulated plane-wave directions.</td>
<td>146</td>
<td>(D.19)</td>
</tr>
<tr>
<td>$\hat{k}$</td>
<td>Unit wave vector.</td>
<td>142</td>
<td>(D.5)</td>
</tr>
<tr>
<td>k</td>
<td>Wave vector.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Scaling factor.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{K}$</td>
<td>MFIE operator.</td>
<td>128</td>
<td>(C.18)</td>
</tr>
<tr>
<td>L</td>
<td>Truncation parameter.</td>
<td>147</td>
<td>(D.22)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>The wavelength $\lambda = \frac{c}{f}$, $c$ is the speed of light.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>EFIE operator.</td>
<td>128</td>
<td>(C.16)</td>
</tr>
<tr>
<td>$M$</td>
<td>Magnetic field intensity.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\overline{M}$</td>
<td>Preconditioning matrix.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Material permeability, in free-space</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\mu_0 = 4\pi \cdot 10^{-7} H/m$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>N</td>
<td>Number of unknowns.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\hat{n}$</td>
<td>Normal vector.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Angular frequency $\omega = 2\pi f$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$p_{pq}^j$</td>
<td>SHE coefficients for the $j$’th EFIE transmit pattern.</td>
<td>20</td>
<td>(3.5)</td>
</tr>
<tr>
<td>$P_l(x)$</td>
<td>The Legendre polynomial of order $l$ at $x \in [-1, 1]$.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$q_{pq}^j$</td>
<td>SHE coefficients for the $j$’th MFIE transmit pattern.</td>
<td>20</td>
<td>(3.6)</td>
</tr>
<tr>
<td>$\mathcal{R}(k)$</td>
<td>Receive pattern, for EFIE given in:</td>
<td>145</td>
<td>(D.13)</td>
</tr>
<tr>
<td>$T_L$</td>
<td>Translation function with $L + 1$ terms.</td>
<td>143</td>
<td>(D.9)</td>
</tr>
<tr>
<td>$\mathcal{V}(k)$</td>
<td>Transmit pattern, for EFIE given in:</td>
<td>145</td>
<td>(D.12)</td>
</tr>
<tr>
<td>$\nabla$</td>
<td>Right-hand side vector.</td>
<td>130</td>
<td>(C.29)</td>
</tr>
<tr>
<td>$Y_{pq}^j(\theta, \phi)$</td>
<td>Spherical Harmonics of degree $p$ and order $q$ at $\theta, \phi$.</td>
<td>20</td>
<td>(3.2)</td>
</tr>
<tr>
<td>$\overline{Z}$</td>
<td>Impedance matrix.</td>
<td>130</td>
<td>(C.31)</td>
</tr>
<tr>
<td>Operator</td>
<td>Name</td>
<td>Page</td>
<td>Equation</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>*</td>
<td>Complex conjugation, ( v = a + jb ), ( v^* = a - jb ).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>*</td>
<td>Element-wise multiplication, ( \bar{c} = \bar{a} \star \bar{b} \rightarrow )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( \nabla a )</td>
<td>The gradient of the scalar function ( a ).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( \nabla \cdot A )</td>
<td>The divergence of the vector ( A ).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( \nabla \times A )</td>
<td>The curl of the vector ( A ).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( \nabla^2 a )</td>
<td>The Laplacian of the scalar function ( a ), ( \nabla^2 a = \nabla \cdot \nabla a ).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I )</td>
<td>The identity operator.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( \iint d^2 \hat{k} )</td>
<td>Integral over the surface of the sphere.</td>
<td>142 (D.6)</td>
<td></td>
</tr>
<tr>
<td>( K )</td>
<td>The MFIE operator.</td>
<td>128 (C.18)</td>
<td></td>
</tr>
<tr>
<td>( L )</td>
<td>The EFIE operator.</td>
<td>128 (C.16)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>Computer-Aided Design</td>
<td>-</td>
</tr>
<tr>
<td>CFIE</td>
<td>Combined Field Integral Equation</td>
<td>(C.19)</td>
</tr>
<tr>
<td>COO</td>
<td>Coordinate List</td>
<td>-</td>
</tr>
<tr>
<td>CSR</td>
<td>Compressed Sparse Row</td>
<td>-</td>
</tr>
<tr>
<td>DGIE</td>
<td>Discontinuous Galerkin Integral Equation</td>
<td>134</td>
</tr>
<tr>
<td>EBF</td>
<td>Excess Bandwidth Formula</td>
<td>D.22</td>
</tr>
<tr>
<td>EFIE</td>
<td>Electric Field Integral Equation</td>
<td>C.15</td>
</tr>
<tr>
<td>FGMRES</td>
<td>Flexible Generalized Minimum Residual Method</td>
<td>110</td>
</tr>
<tr>
<td>FMM</td>
<td>Fast Multipole Method</td>
<td>D.7</td>
</tr>
<tr>
<td>GMRES</td>
<td>Generalized Minimum Residual Method</td>
<td>105</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
<td>-</td>
</tr>
<tr>
<td>GRASP</td>
<td>General Reflector Antenna Software Package</td>
<td>-</td>
</tr>
<tr>
<td>GWP</td>
<td>Graglia-Wilton-Peterson</td>
<td>23</td>
</tr>
<tr>
<td>HO</td>
<td>Higher-Order</td>
<td>-</td>
</tr>
<tr>
<td>ILU</td>
<td>Incomplete LU</td>
<td>119</td>
</tr>
<tr>
<td>LO</td>
<td>Lower-Order</td>
<td>-</td>
</tr>
<tr>
<td>MFIE</td>
<td>Magnetic Field Integral Equation</td>
<td>C.17</td>
</tr>
<tr>
<td>MLFMM</td>
<td>Multi-Level Fast Multipole Method</td>
<td>11</td>
</tr>
<tr>
<td>MoM</td>
<td>Method of Moments</td>
<td>C.26</td>
</tr>
<tr>
<td>OOC</td>
<td>Out-of-Core</td>
<td>-</td>
</tr>
<tr>
<td>PEC</td>
<td>Perfect Electric Conductor</td>
<td>-</td>
</tr>
<tr>
<td>PMCHWT</td>
<td>Poggio-Miller-Chang-Harrington-Wu-Tsai</td>
<td>Appendix C.2</td>
</tr>
<tr>
<td>PO</td>
<td>Physical Optics</td>
<td>-</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
<td>4.1</td>
</tr>
<tr>
<td>RWG</td>
<td>Rao-Wilton-Glisson</td>
<td>21</td>
</tr>
<tr>
<td>SHE</td>
<td>Spherical Harmonics Expansion</td>
<td>Section 3.3</td>
</tr>
<tr>
<td>SPAI</td>
<td>Sparse Approximate Inverse</td>
<td>177</td>
</tr>
<tr>
<td>TPE</td>
<td>Trigonometric Polynomial Expansion</td>
<td>75</td>
</tr>
</tbody>
</table>
List of Publications

During the present study, the journal papers [J1–J4] and conference papers [C1–C5] were submitted. Further, participating as a co-author, the papers [O1–O5] were completed.

The journal papers [J1–J4] and conference papers [C1, C2, C5] are included in Appendix A on pages 55–104.

Journal Papers


MLFMM is presented and the challenges for HO MLFMM are discussed, including the consensus that going above 2nd order is not beneficial for HO MLFMM. Then, the sparse near-matrix format, adaptive grouping and the Spherical Harmonics Expansion are discussed. A few practical cases illustrate that with the modifications, HO MLFMM provides significant benefits over LO MLFMM.


The adaptive grouping approach is presented and discussed in detail. Further, the idea of a locally extended Octree is mentioned. Then, the performance is shown for both a regular mesh (a sphere) and an irregular mesh (a circular plate with small
The conclusion is that adaptive grouping is very beneficial for HO MLFMM, particularly for irregular meshes (which are widespread in practical applications).


A detailed discussion on how to implement the Gaussian Translation Operator for scattering problems. Since the potential numerical instabilities are devastating for scattering problems, we present some improved criteria for choosing the parameters involved. These criteria also allow automated and unsupervised selection of the parameters involved in the translation operator. Further, we demonstrate that the potential time savings are promising, if somewhat modest, and highlight scenarios where the operator would be more applicable.


A global interpolation procedure for MLFMM, based on low sample-rate trigonometric polynomials and first presented in [75] on a theoretical test case, is applied to compute the scattering from a sphere as well as a more general geometry. By examining the computational resources, as compared to an implementation using Lagrange interpolation, the approach is demonstrated to be superior to local interpolation for a shared-memory MLFMM implementation.

Conference Papers


Further elaborating on the results from [J1], we here compare our implementation directly to results from the literature. Using the same test case as [89], we demonstrate an order of magnitude better accuracy at less than half the memory.


We consider the implementation of Physical Optics (PO) on a Graphics Processing Unit (GPU). While the PO algorithm itself is easy to port to a GPU, we
discuss how to ensure sufficient accuracy while operating in IEEE single precision, which allows for much higher speeds than double precision on a GPU. We achieve speed-ups of roughly a factor of 20 compared to the GRASP PO code.


A discussion of our work on the Gaussian Translation Operator, preceeding [J3].


In this paper, we document some research that is in a sense orthogonal to the work in the rest of the thesis: The use of integral equations for the analysis of reverberation chambers. The conclusion is that by applying Higher-Order basis functions and the Magnetic Field Integral Equation, much larger problems can be solved than previously reported.


Documenting our work on the Fast Physical Optics algorithm, we consider a number of test cases, ranging from canonical to realistic. We demonstrate strong error control while significantly reducing the calculation time, and show that our implementation significantly improves the state-of-the-art in fast surface current integration from general surfaces.

**Co-author Papers**


A presentation of the work on designing and optimizing a 1.2 meter contoured beam reflectarray for $K_u$ band. With a direct optimization approach based on a spectral-domain periodic MoM (SDMoM), the optimized reflectarray design is able to fulfill all coverage requirements of a real Direct Broadcasting Mission. The HO MLFMM solution of a reflectarray involving 2 million unknowns on a very irregular mesh is used to validate the SDMoM, with very good agreement even 30 dB below peak.

A review of the research done at TICRA concerning methods for analysis of the supporting structures for reflector antennas. First, specialized methods for strut modelling are considered and validated by comparison with HO MLFMM. Then, reflector antennas mounted on satellite platforms, including a model of the Olympus satellite involving more than 4.3 million unknowns, are analyzed using HO MLFMM, demonstrating the importance of taking the surroundings of the antenna into account.


An overview of some of the new methods available in GRASP, this paper discusses the HO MLFMM solver and its advantages relative to LO MLFMM, as well as its combination with DGIE. Further, the Fast-PO algorithm is briefly discussed and both HO MLFMM and Fast-PO are used to solve an example involving a large reflector antenna ground station.


A milestone in reflectarray design and optimization, this paper analyzes a 1 meter curved contoured beam reflectarray at $K_u$ band. By applying advanced optimization methods, further specialized to curved reflectarrays, this paper shows that the performance of a reflectarray can match that of a shaped reflector. HO MLFMM is used to validate the specialized analysis method used during optimization.


A description of the pattern computations of the UHF antennas on the ExoMars spacecraft. The antenna is a quadrifilar helix antenna with a dielectric radome. HO MLFMM and DGIE is used to allow a full-wave solution with only modest computational effort.
This thesis would have never been successful without the tremendous amount of support I have gotten from my advisors. Per Christian Hansen is not only an extremely talented mathematician and engineer, but, equally important, he is also an amazing scientific communicator and teacher. It has been a privilege to be allowed to learn from him over all these years. From TICRA, Peter Meincke and Erik Jørgensen are true experts, and have taken the time to provide me with very careful and lengthy tutoring sessions—having daily access to their guidance has been fundamental to the results of this thesis.

Many other people have helped me during my thesis work. From TICRA, Stig B. Sørensen has become my idol, and Knud Pontoppidan, Hans-Henrik Viskum and Niels Vesterdal have helped me numerous times during the project. Outside of TICRA, I have enjoyed discussions with Thorkild Hansen and Seppo Järvenpää.

I would have never gotten to do a Ph.D. had it not been for the many gifted teachers I have had at the Technical University of Denmark. I would like to extend my gratitude towards Lars R. Knudsen, Ole Christensen and Hans B. Nielsen. Furthermore, my collaboration throughout my studies with Emil Sokoler has been extremely beneficial. Emil’s enthusiasm and relentless pursuit of the best solution to a given problem has been, and continues to be, a huge inspiration. Emil, I would have never gotten to where I am without you.

Finally, my family and friends have made everything more fun and have helped me remember that there is a world outside computational electromagnetics. My parents have provided constant support, and Siri has always been there for me. Last but not least, Lilje and little-sis: You are perfect.
CONTENTS

G Computers 185
G.1 Laptop ........................................ 185
G.2 Computing Server .............................. 186

Bibliography 187
The computational analysis of modern antennas, particularly for applications such as space or telecommunications, is a demanding task due to the high accuracy requirements and electrically large structures. In the present thesis, we focus on two of the most commonly used algorithms for the scattering analysis of electrically large structures: Physical Optics and Method of Moments. We note that in some scenarios, the Geometrical Optics (GO/UTD) approach can also be applicable.

Physical Optics (PO/PTD) locally approximates the scatterer by its tangent plane. This approach is computationally more intensive than GO/UTD but is in general also much more accurate, and has traditionally been the method of choice for reflectors with diameters above, say, 20λ. The computational time for PO/PTD scales as $O(f^4)$, with $f$ being the frequency, but the memory only scales as $O(f^2)$. The computational time is thus the main limiting factor for PO/PTD’s applicability towards reflector antenna problems.

The Method of Moments (MoM) is a full-wave method, which means that it can yield a highly accurate solution by solving the Helmholtz equation. Applying surface integral equations, the surface current density on the scatterer

---

1In this thesis, we employ the Big-O notation $O(g(x))$ to mean that the true scaling $h(x)$ of the algorithm in question as $x \to \infty$ satisfies $|h(x)| \leq M|g(x)|$ for a fixed positive $M \in \mathbb{R}$ for all $x > x_0, x_0 \in \mathbb{R}$. [4]
is obtained by solving a large linear system of equations discretized using the Galerkin approach [6]. The number of unknowns $N$ in this system scales as $O(f^2)$, and thus, the computational time for MoM scales as $O(f^4)$ or $O(f^6)$, depending on the solution method for the linear system. The memory scales as $O(f^4)$.

The topic of the present thesis is the acceleration of these two methods, Physical Optics and Method of Moments. The term acceleration, as used in this thesis, has a very specific meaning: Reduction of the scaling of the required computational resources against the frequency. Thus, by accelerating these methods, we can allow electrically much larger problems to be solved on the same hardware.

The emphasis is not on very large computing structures such as distributed computing, but on optimal use of the existing hardware, such that users can get the most benefit out of the systems already available to them.

Accelerating the Method of Moments is done through the use of the well-known Multi-Level Fast Multipole Method (MLFMM) [7–12], which achieves at least $O(N \log^2 N) = O(f^2 \log^2 f^2)$ scaling. This approach has been applied to a wide variety of scattering problems, and is by now considered a very mature technology, providing an extreme reduction in the required computational resources. It has been implemented on a variety of computing platforms [13–17], and used to solve problems in many other disciplines (e.g., [7,18,19] and [20, Section 5]).

Combining MLFMM with the benefits achieved by using Higher-Order (HO) basis functions [21–27] with orders higher than 2 has not been successfully done in spite of several attempts [28–32]. This is unfortunate, because the increasing the order of the basis functions allows for a significant reduction in the number of unknowns $N$. This means that implementations of surface integral equation solvers are forced to make an unappealing choice: Reduce the scaling against $N$ using MLFMM or reduce $N$ by using HO basis functions.

For Physical Optics, the research on reducing the scaling against the frequency is much less mature [33–36]. This is somewhat surprising, since the main bottleneck is the evaluation of the field from a current distribution, an ubiquitous operation in computational electromagnetics. For the accuracies required by TICRA, no accelerated results have been published. Furthermore, computing the reactive near-fields have not been accelerated significantly, with the published methods focusing on much greater distances between the source and observation points.

In summary, this means that none of the currently published methods can accelerate Physical Optics while satisfying the requirements from TICRA.

The thesis is distributed as follows. Chapter 2 presents an overview of the state-of-the-art surrounding MLFMM and, in particular, the challenges faced when using MLFMM in combination with HO basis functions, referred to as
HO MLFMM. Chapter 3 then presents the main contribution of the thesis, a summary of the modifications to the standard MLFMM algorithm that were performed in order to yield an effective HO MLFMM code. Chapter 4 then goes into detail with the performance of the code on a number of test cases, ranging from the purely canonical Mie scattering from a sphere to the radiation from a space telescope and the scattering from a realistic satellite model. Chapter 5 provides some conclusions.

In Appendix B we consider the development of a Fast-PO algorithm, relegated to an appendix since it the algorithms developed have been deemed confidential by TICRA. After a review of the fundamentals of Physical Optics in Section B.1 we perform a literature review in Section B.2 concerning Fast-PO, where the shortcomings of the existing approaches are discussed in detail. We briefly highlight the Fast-PO algorithm’s main capabilities and limitations in Section B.3. We finish with a detailed analysis of its performance on a variety of test cases in Section B.4.

The thesis contains a number of other appendices. While Appendix A contains the main papers written during the thesis, and Appendix B contains research, the remaining appendices contain secondary points relating to the thesis. This distribution of the thesis will probably seem inappropriate to some readers, since the material relating to conclusions in the main body of the thesis is relegated to the appendices, but it is done to allow the main part of the thesis to be short and concise. Appendix C briefly summarizes the electromagnetic theory regarding scattering, and then discusses discretization of integral equations in computational electromagnetics. Appendix D reviews the theory behind MLFMM — much of the main body of the thesis assumes that the reader is familiar with Appendix C and Appendix D. Finally, Appendix E discusses some aspects relating to the solution of the linear system of equations when applying MLFMM, and Appendix F contains a number of supporting arguments to conclusions made in the thesis. A list of symbols and abbreviations can be found at the start of the thesis, on page vii.

It is important to stress that the work done in this thesis stands on the shoulders of an enormous amount of work done by others at TICRA. This is fundamental to the results achieved in the thesis, since it has allowed our code to build upon the impressive capabilities already available in TICRAs flagship product GRASP. Thus, the following key components are used by the code developed in this thesis:

- Everything related to a Higher-Order Method of Moments solution. This includes the geometrical discretization of a wide range of scatterers and tabulated meshes as well as the creation of basis functions and EFIE
surface integration routines, including strong capabilities for handling dielectrics. The extension to allow CFIE solution of closed scatterers was done in cooperation with the advisers. Further, tools for iterative solution such as an iterative solver and a preconditioner were available, although their capabilities were extended by us as discussed in Section 3.6 and Section 3.7 respectively.

- Everything related to a Physical Optics solution. This includes specialized integration rules for canonical structures and an auto-convergence procedure to ensure the specified accuracy.

Thus, while the standard MoM code was somewhat extended as a consequence of our work, we stress that the contributions of this thesis are only related to the acceleration of MoM and PO. The most recent version of GRASP is GRASP 10.3, which is what we use as a reference in Appendix B.

An $e^{\jmath \omega t}$ time-dependence, with $\omega = 2\pi f$ the angular frequency, is assumed and suppressed throughout the thesis.
Multi-Level Fast Multipole Method (MLFMM)

This chapter is a discussion of the existing research on the Fast Multipole Method (FMM) and its multi-level variant, the Multi-Level Fast Multipole Method (MLFMM). In particular, we consider their application to a scattering problem discretized using Higher-Order (HO) basis functions. To keep the chapter concise, most of the discussion of the underlying theory is relegated to the appendices. This includes a discussion on the discretization of scattering problems in Appendix C as well as a review of the (ML)FMM method in Appendix D.

We note that when using the term *standard MLFMM implementation* in this thesis, we are referring to an implementation that utilizes Lagrange interpolation, $K = 2(L+1)^2$ samples of the basis function patterns, and Octree grouping. Such implementations are detailed in, e.g., Gibson [37] Chapter 8 and Chew *et al.* [38].
2.1 Literature Survey

The Fast Multipole Method (FMM) is a general purpose approach for solving problems involving operations with so-called *long-ranged forces*, i.e., rapidly decaying kernels in integral operators, also called *potentials*. Specifically, the FMM applies a series expansion of the kernel $K$ in an integral transform

$$
\int_a^b K(t, u)f(t)\,dt = g(u).
$$

(2.1)

For the series to converge, the kernel must decay rapidly as $\|t - u\|$ increases.

The mathematical foundations for the method were developed in Rokhlin's seminal paper [39] on the Laplace equation $\nabla^2 u = 0$, and were expanded with [40] concerning the 2-D Helmholtz equation $\nabla^2 u + k^2 u = 0$ and [41][42] on 3-D Helmholtz $\nabla^2 u + k^2 u = 0$. Since each equation, and thus the kernel used to solve it, typically has distinct properties, the research on FMM for electromagnetics is distinct from research on FMM as applied to other kernels. We note, however, that there is some work on *kernel-independent* FMM [43,44]. Important publications on applications other than electromagnetics include [7,18,19].

In the early stages of implementing Helmholtz FMM, which is considerably more involved than, e.g., the Laplace kernel [8, Section 1], several papers applied FMM to fairly simple structures. This includes Engheta *et al.* [9] (in 2-D) as well as the seminal paper by Coifman *et al.* [10]. Neither of these papers consider the topic of interpolation, and thus are applications of the traditional single-level FMM. Coifman *et al.* [10, p. 7] comment that the single-level FMM should be sufficient for problems smaller than approximately 50,000 (1st order) basis functions.

Applying the Multi-Level FMM (MLFMM) was first done by Lu & Chew [11], achieving $O(N \log^2 N)$ complexity, although only in 2-D where the interpolation and integration is simpler (done by Fourier series and the trapezoidal rule, respectively). Lu & Chew comment that $O(N \log N)$ complexity can be achieved with nearest neighbour interpolation. Song & Chew [12] considered 3-D with the CFIE and polynomial interpolation, achieving $O(N \log N)$ in a rather compact paper, which serves primarily as numerical illustration of the algorithm’s scaling. Other important early papers on applying (ML)FMM include:

- Song & Chew [45], considering scattering from arbitrarily shaped 3-D bodies using single-level FMM.
- Dembart & Yip [46], discussing error control.
• Song et al. [47], highlighting a number of important considerations in practical implementations, including the selection of the FMM truncation parameter $L$, the use of preconditioning and the use of an initial guess for the iterative solver.

• Gyure & Stalzer [48], a detailed discussion of how to implement MLFMM.

• Sheng et al. [49], discussing the important extension of MLFMM to scatterers containing dielectric material.

From the turn of the millenium, the focus naturally shifted away from development of the core of FMM, and towards the details in the implementation. Some papers provided general overviews [50], whilst others focused on specific improvements to each of the components involved in an MLFMM code.

Significant effort has also been spent on accelerated methods for pre-computing the translation function $T_L$ [10.9]. Simply calculating $T_L$ directly for all $K = 2(L+1)^2$ points in a standard MLFMM implementation requires $O(L^2)$ computational time and memory. However, it can be done in $O(L)$ time, using a variety of approaches. [51–53] consider simple interpolation of the translation function, using extreme oversampling to allow sufficient accuracy from low-order interpolation polynomials. Velamparambil & Chew [54] discuss how to efficiently evaluate the translation function on the fly, utilizing [55], and is notable because it is nearly storage-free, requiring only $O(L)$ memory. Song & Chew [51, p. 110] write that the method in [55] has a large crossover point, i.e., is too computationally expensive for most common problems. Velamparambil et al. [13] note that to use [54], a rewrite of parts of the MLFMM matrix-vector product algorithm is needed, a significant drawback. A better solution was proposed by Hänninen & Sarvas [56], utilizing Fourier series to oversample the translator and then polynomial interpolation to get the actual samples, yielding a speedup of an order of magnitude compared to the simple interpolation. We believe that [56] is the state-of-the-art on this subject. Michiels et al. [57] improved on the simple interpolation for distributed-memory applications.

Moving beyond the acceleration of the translation setup stage, some focus has also been devoted to alternative translation schemes, allowing for faster execution of the translation step in the matrix-vector product. These include:

• Wagner & Chew [58], using a windowing of the translation operator in 2D to only consider a small subset of the angular directions.

• Burkholder & Kwon [59], evaluating the translator asymptotically in the high-frequency limit in 2D, corresponding to a geometrical optics approach.
Multi-Level Fast Multipole Method (MLFMM)

- Michielssen & Chew \cite{60}, considerably improved by Hu et al. \cite{61}, applying a contour integral representation of the 2D translation function. We stress that this requires a significant recoding of the algorithm.

- Hansen \cite{62,64}, using the theory of Gaussian beams to derive an exact and directive translator.

We will return to the use of alternative translation schemes briefly in Section \ref{3.5} and discuss it in detail in \cite{J3}, and we note in passing that the Gaussian beam approach is the only applicable approach in 3-D, allowing a fully error-controllable scheme.

Error analysis, for the standard translator, has received detailed attention. Koc et al. \cite{65} and Darvé \cite{66} go into detail regarding the error analysis. However, their work on the truncation parameter has since been superseded by Song & Chew \cite{67} which is now the standard reference. We remark that \cite{67} contains an unfortunate misprint on p. 913, where it indicates that it uses a Gauss-Legendre integration rule in $\theta$ instead of the correct $\cos \theta$. Hastriter et al. \cite{68}, extending work done in 2-D by Ohnuki & Chew \cite{69}, focused on the truncation error of the translation function, and considered in detail how the translation error behaves and how to reduce it. We discuss some of these aspects further in Appendix \ref{F.4}.

The choices involved in finding the interpolation and anterpolation operators have also received some attention. The standard approaches, involving local polynomial interpolation, were discussed as part of the early general MLFMM papers \cite{12,48}, and later refined in \cite{70,72} and discussed in more detail, e.g., by \cite[Section 3.5]{38} and \cite[Section 8.5.3]{37}. Global interpolation was introduced by Sarvas \cite{73}, based on FFT, and by Chowdhury & Jandhyala \cite{74}, based on a fast spherical transform. Järvenpää & Ylä-Oijala \cite{75} significantly improved upon the FFT-based interpolation scheme. We will return to this in Section \ref{3.4} and in \cite{J4}.

A significant amount of attention has been devoted to parallelizing the MLFMM and achieving strong performance on distributed memory architectures. This is not trivial for MLFMM, primarily because of the tiering of the algorithm — for acceptable memory consumption, each stage of the algorithm has to be completed before the next one can begin. There are also other concerns, such as intersecting memory access patterns, leading to significant inter-process communication. The earliest attempts, using the FISC \cite{76,77} and ScaleME \cite{13,28} implementations, achieved significant speedups by applying an MPI scheme. Vešlamparambil & Chew \cite{14} achieved further improvements by a more elaborate load-balancing. Ergül & Gürel \cite{15} went into much more detail, and is one of the standard references on the topic, with Ergül & Gürel \cite{16} extending the approach further. A popular alternative is the FFT-(ML)FMM, as developed by Taboada
et al. [78–80], which yields a slightly higher complexity, between $O(N^{4/3} \log N)$ and $O(N \log N)$ as the number of workstations are increased — the actual scaling of the algorithm is dependent on the computing architecture. The advantage of FFT-(ML)FMM is its almost perfect parallelizability, which makes it suited for large, distributed-memory systems. Finally, we note that the use of Graphics Processing Units in connection with MLFMM is considered in [17,81]. We will not consider the use of distributed-memory architectures further in this thesis, restricting ourselves to considerations surrounding a shared-memory implementation.

The grouping of the basis functions, and the basis function patterns, have also been improved based on the standard MLFMM implementation. Eibert developed SE-MLFMM in [82], a version of MLFMM where the plane-wave expansions are stored as coefficients to a Spherical Harmonics Expansion, thereby achieving significant memory savings. Ismatullah & Eibert [83] discusses applying SE-MLFMM to higher-order basis functions. We will go into more detail on SE-MLFMM in Section 3.3 and we discuss the performance in [11, Section IIIb].

It is worth noting that applying MLFMM in the low-frequency regime, i.e., for sub-wavelength problems with a large number of unknowns, requires significant modifications to the algorithm that are not considered in this thesis. We refer to [84–86].

2.2 MLFMM Performance with HO Basis Functions

The use of Higher-Order (HO) basis functions in an MLFMM implementation (HO MLFMM) have received a modest amount of interest by a few research groups. Unfortunately, the conclusion was fairly grim, as will be discussed later. First, we will briefly summarize why HO MLFMM does not automatically yield a low-memory algorithm, contrary to the case for HO Method of Moments (HO MoM) where the reduction in the number of unknowns directly leads to a significant memory reduction [87]. More details on the combination of HO basis functions and MLFMM, including simple numerical illustrations, can be found in Appendix F.2. Further, we refer to the discussion on discretization in Appendix C.1.

For a HO discretization, the patches are larger than for a lower-order (LO) discretization. Since the number of levels in the Octree is determined by considering the largest patch sidelength, the number of levels for HO MLFMM will be
lower than for a corresponding LO MLFMM. Therefore, the level at which the basis function patterns are tabulated will have more samples for HO MLFMM than for LO MLFMM. Finally, particularly for the often encountered irregular meshes, the larger groups can also cause more interactions to be placed in the near-matrix. This significantly reduces the sparsity of the near-matrix, but only modestly increases memory consumption, since it is often counteracted by the reduced number of rows in the near-matrix.

There are several advantages in applying HO MLFMM. In addition to the drastically reduced number of unknowns, the lower number of groups and levels means less memory for group patterns and translators. Further, the reduced number of translations and in/an-terpolations results in a significantly reduced runtime. Finally, the reduced sparsity of the near-matrix allows for an improved preconditioner based on the near-matrix.

There is thus a trade-off. On one hand, we want a large patch side length to be able to reduce the number of levels and basis functions. On the other hand, we want a small sidelenst to keep a low sampling rate and have well-partitioned groups.

The consensus at the outset of the present thesis is clear: The sweet spot for that trade-off is the use of second-order basis functions. This has been verified by several different authors, as will be explained in the following.

The first to consider the use of Higher-Order basis functions with Helmholtz MLFMM were Donepudi et al., who in a series of papers [28–30,88] discussed various aspect related to the performance of HO MLFMM, using the Graglia-Wilton-Peterson (GWP) [23] basis functions, discussed as part of Appendix C.1.2.

Of particular interest are the two earliest papers from Donepudi et al. The first, [29], introduces a new approach to allow very small groups to be used with HO basis functions. The idea is to use the integration points from the numerical integration of (C.31) as the target of the FMM expansion (D.10), yielding a novel point-based approach. While this does indeed allow a smaller group size, it also means that increasing the integration accuracy in the near-matrix results in significantly increased FMM memory — in a traditional implementation, the memory is unaffected by the integration accuracy in the near-matrix. Further, the reduction in memory is reported as a factor of 4 [29, p. 1195], roughly what can be achieved by applying our adaptive grouping (see Section 3.1) for realistic scatterers. Note also that the result is an increased number of levels, and as argued above, this comes at a time cost. Finally, the implementation of the point-based approach requires significant rewriting of an existing implementation. Interestingly, we have not seen other authors adopt this approach.
The second paper by Donepudi et al., [28], considers a more traditional implementation. While the title of the paper refers to parallelization, the paper is a more general study of HO vs. LO MLFMM performance. The paper considers the use of preconditioner and solver, concluding that a block-diagonal preconditioner and the GMRES solver provides the strongest convergence of those considered, particularly for HO. The paper also includes, in Figures 3—4, a convergence plot of 1st – 3rd order basis functions in FMM. The main conclusion, given both in the discussion concerning Table II and the Conclusion, is that the increased patch size “limits the performance of higher order MLFMA”. Tables II & III in [28] show that 2nd order yields the lowest memory consumption.

This conclusion stood unchallenged for some years, until 2009 where two separate groups published some results. Ismatullah & Eibert [32] considered only 1st and 2nd order, i.e., either LO or HO basis functions, on flat triangular patches. The paper includes a large number of results, but does not perform a detailed comparison of accuracy versus memory use for LO and HO. Further, it is somewhat unclear what their memory counts include. They apply the SHE (see Section 3.3) to reduce the memory of the basis function patterns, but the conclusion is that only “carefully prepared meshes can have efficiency advantages with respect to LO solutions. Too fine HO meshes become quickly inefficient with respect to the LO case”. Remember, HO here refers to 2nd order.

Kolundžija et al. published two conference papers [31,89], which include the first implementation of hierarchical basis functions on quadrilateral meshes. The results from [31] have high RMS errors and include a threshold that needlessly increases the near-matrix significantly to yield acceptable error levels. In [89], their implementation is used to examine the optimal order for MLFMM, with the conclusion that “memory consumption increases with increasing the order of approximation for orders greater than 2”.

Thus, both Ismatullah & Eibert and Kolundžija et al. reach the same conclusion as that of Donepudi et al.: It is not beneficial, in terms of memory, to go above 2nd order.

Two other groups have published results on HO MLFMM. Wang et al. [90] implement the same basis functions as in the present thesis, but only consider single-level FMM for a very small problems (4λ cube, 5λ sphere) and have some odd results, including a significant increase in error for a nearly fixed number of unknowns as the basis function order is increased. Interestingly, they apply the approach from Donepudi et al. [29], and conclude that “second- or third-order polynomials should be preferred”. Sheng et al. [91] apply MLFMM with GWP functions, although they do not cite [23]. The paper discusses a few aspects concerning HO MLFMM, including a locally extended Octree and the use of a
Sparse Approximate Inverse preconditioner. However, they do not consider the relationship between accuracy and memory use for each order.
This chapter considers modifications applied to various components of an MLFMM implementation, in an effort to tailor MLFMM towards use with a Higher-Order discretization. We stress that unless otherwise noted, none of these modifications make MLFMM worse for LO discretizations—in fact, several of them are very applicable also for LO MLFMM.

Considering the components required for a typical MLFMM, that is,

- Grouping,
- Near-matrix,
- Basis Function Patterns,
- Translation Function,
- Interpolation,
- Iterative Solver,
- Preconditioner,
the following sections contain research on modifications of all of these components.

It is important to stress that not all modifications are novel. In fact, while a few modifications were developed specifically for the present thesis, it is the interplay between the modifications that create the strong performance to be detailed in Chapter 4.

3.1 Adaptive Grouping

In this section, we present the adaptive grouping approach for grouping basis functions. For numerical examples, we refer to [J1, Section IIIb] and [J2], which also have some further details on the implementation of the method.

Much of the literature on MLFMM, particularly early on, has been focused on regular scatterers such as spheres, cubes, and so on. However, for practical use, we have to consider that more arbitrary geometries are used as scatterers. This, in general, leads to irregular meshes with varying side lengths, particularly for HO discretizations with quadrilateral patches.

Such irregular meshes can lead to several problems:

- If the mesh is locally very fine, irregular meshes can cause problems for the convergence of the iterative solver. This is known as the dense discretization breakdown of the EFIE operator [92], which we will not discuss further, although we note that significant effort has been put into meshing the scatterer properly to avoid small patches. By using HO basis functions, we can increase the accuracy by increasing the order rather than reducing the patch size, which also helps avoid the dense discretization breakdown.

- When using basis function families that are interpolatory in nature (cf. p. 133), the same basis function order has to be used on all patches regardless of the patch size, significantly limiting the applicability of interpolatory basis functions in practice. Since we use hierarchical basis functions, this is not a problem for our implementation.

Regardless of the basis functions used, the primary problem with irregular meshes in connection with MLFMM is that they lead to oversized groups. Since the size of the finest level $q$ is determined by the largest patch size in the mesh, the finely meshed regions will still need to have their basis functions sampled...
3.1 Adaptive Grouping

at a fairly high sampling rate. Further, since the translator is more stable for larger groups (see Appendix F.5), it is unfortunate to have to let the group size be dictated by translator accuracy rather than the local patch size.

To alleviate this problem, we simply added an additional level to the Octree, hereafter termed the adaptive level. At the adaptive level, each patch is given a group — optionally, for clusters of very small patches, several patches can be put into the same group. The group diameter of each adaptive group is determined separately, such that the adaptive group encloses all patches inside it. Further, the center of the group is chosen to minimize the group diameter.

The basis function patterns are then tabulated on this adaptive level, which will often have a much lower sampling rate than in the finest level of the Octree. Considering basis function \( f_i \) situated in group \( m' \), we can express the adaptive grouping as an expansion of the basis function pattern \( V_{im'} \) from (D.13) as

\[
V_{im'} \simeq e^{-jk \cdot (r_{m'} - r_{\bar{m}'})} \mathcal{W} \{ V_{i\bar{m}'} \}.
\]  

(3.1)

where \( \bar{m}' \) is the adaptive group that lies inside group \( m' \) and \( \mathcal{W} \) is an interpolation operator as discussed in Section D.3.2. Except for the controllable interpolation error, (3.1) is exact.

When performing the matrix-vector product, the aggregation and disaggregation steps are then slightly modified. The multiplication of the excitation coefficients onto the transmit patterns (D.34) is done on the adaptive level, and is followed by the interpolation and phase-center shift in (3.1). When performing the disaggregation, a phase-center shift and anterpolation from the finest level to the adaptive level is inserted before (D.38).

To understand the benefits of this approach, it is necessary to review why an Octree is used to group basis functions. An Octree used as a grouping approach with MLFMM has three important properties:

1. It is fast.

2. It provides a hierarchical grouping, such that each level is completely contained within all previous levels, and has a fixed size.

3. It provides a structured grouping, such that the displacement of a child group from its parent is limited to 8 possible vectors.

These properties are very useful for MLFMM, because it allows precomputation and reuse of the translation and shifting operators. If each box had been
allowed a varying size, or if each child could be freely displaced from its parent, precomputation would have required a staggering amount of memory.

However, the properties 2 and 3 are not beneficial for the basis function patterns, and thus limit the performance. The hierarchical and structured grouping means that the Octree is not allowed to adapt to the contents of each group during grouping, so properties 2 and 3 leads to oversized groups. Furthermore, since we have to store a basis function pattern for each basis function, the concept of 'reuse' is not applicable.

Therefore, Octree grouping is great for the translators and shifting operators, but poor for the basis function patterns. Adaptive grouping, by allowing each group to be displaced and sized independently, results in memory savings for the basis function patterns with little effect on the runtime. In other words, the lack of structure in the adaptive level allows for it to yield a much tighter grouping of the patches and subsequently reduce the memory use compared to the Octree grouping.

It is important to note that translations are not done on the adaptive grouping level, precisely because of the lack of structure in the adaptive level. There would be no reuse of translation vectors and thus a translator would have to be calculated for each pair of interacting source and receiver adaptive groups, devastating the performance. As a consequence, since the near-matrix $\mathbf{Z}_{\text{near}}$ essentially contains the interactions for which the translator is not applied, the near-matrix is still based on the finest level of the Octree and is not affected.
by the adaptive grouping. Another reason for not using the adaptive level for translations is that the adaptive groups can become very small, resulting in a very unstable translation operator (see Appendix F.5).

For discretization using small patches, adaptive grouping has little effect on the memory. Remember that at each level, the group diameter is halved. Thus, when using a large number of levels, the diameters are fairly closely spaced, and thus the mismatch between group diameter and patch diameter is rarely that significant.

For Higher-order meshes, particularly for irregular meshes where the sidelength can vary between $0.1\lambda - 2.5\lambda$, the mismatch can be very significant. As mentioned previously, the finest level sidelength in the Octree is limited by the largest patch sidelength. Therefore, $0.1\lambda$ patches can be put into groups with sidelengths larger than $2\lambda$, yielding an extreme oversampling over the basis function pattern as a function of $\hat{k}$. Adaptive grouping alleviates this problem.

### 3.2 Near-matrix Format

A large part of the memory used when applying MLFMM is the sparse near-matrix $\mathbf{Z}_{\text{near}}$, containing the interactions between groups that are too closely spaced to apply the FMM operator.

The storage of this matrix, containing $\text{nnz}$ elements, takes up a significant proportion of the memory used for MLFMM. This is particularly true for HO MLFMM, because the larger groups, and high basis function density on each patch, means that a relatively large number of interactions are stored in the near-matrix compared to LO MLFMM.

There are a number of approaches for compressing the full MoM matrix by applying low-rank approximation, such as Adaptive Cross Approximation (ACA) \cite{93,94}, IE-QR \cite{95}, CBD \cite{96}, or $\mathcal{H}$-matrix approaches \cite{97,99}. However, these do not immediately appear applicable to the near-matrix, since the low-rank features are treated by MLFMM. We have not seen publications involving compression of the MLFMM near-matrix $\mathbf{Z}_{\text{near}}$, although some have considered using out-of-core storage (see Section 3.8).

While a reduction of the number of elements in $\mathbf{Z}_{\text{near}}$ is thus not immediately available, the auxiliary storage required for storing a sparse matrix can also take up a lot of memory and this can be reduced significantly, as we shall now see.
The most popular storage format for $Z_{\text{near}}$ appears to be a block format [80], which requires a modest amount of auxiliary storage. However, for HO MLFMM, basis functions can be defined across two patches, each patch belonging to separate groups. This results in a cumbersome implementation of block storage with extensive bookkeeping, and to duplicated elements, significantly increasing the auxiliary storage for HO MLFMM. However, we note that a block-based format would allow a straightforward implementation of out-of-core storage.

Another approach for storing $Z_{\text{near}}$ is using the more general formats such as Coordinate List (COO), used, e.g., by Guan et al. [17], and Compressed Sparse Row (CSR), used, e.g., by Gibson [37]. Unfortunately, these require a significant amount of storage for the indexing integers—for COO, the integers take up as much storage as the elements, while for CSR, the integers take up half as much as the elements.

We developed a format that would benefit from the best of both approaches—the low auxiliary storage of a block format with (almost) the general applicability of COO and CSR. The idea behind the format is to exploit the fact that elements in a row often appear in consecutive columns, since basis functions on the same patch will generally be numbered consecutively. Thus, when multiple elements appear in consecutive columns in a row, we do not need to store an integer to indicate the position of every element. Instead, we store the column index of the first element, and a negative number indicating the number of consecutive elements. Thus, storing the column indices from 1 to 100 is done by storing only the integers 1 and $-99$, a saving of a factor of 50 compared to CSR.

The format requires the storage of three arrays. The first is an array values of length nnz holding the elements of the matrix in row-major order to facilitate fast matrix-vector products, just as CSR. The second is the array of column indices cols indicating the column indices of the elements in values array, and the last array colinds indicates the elements in the cols array that begin a new row.

As an example of the format, the matrix

$$
\begin{bmatrix}
1 & 7 & 5 & 6 & 3 \\
8 & \times & \times & 2 & 9 \\
\times & 7 & 4 & 2 & \times
\end{bmatrix}
$$

is (with one-based indexing) stored as

values: $[1 \ 7 \ 5 \ 6 \ 3 \ 8 \ 2 \ 9 \ 7 \ 4 \ 2]$,

cols: $[1 \ -4 \ 1 \ 4 \ -1 \ 2 \ -2]$,

colinds: $[1 \ 3 \ 6 \ 8]$. 

### Table 3.1: The memory use for storing the sparse near-matrix in each of three different formats.

<table>
<thead>
<tr>
<th>Format</th>
<th>Memory [GB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>COO</td>
<td></td>
</tr>
<tr>
<td>CSR</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td></td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>21.12</td>
</tr>
<tr>
<td><strong>Row Indices</strong></td>
<td>10.56</td>
</tr>
<tr>
<td><strong>Col. Indices</strong></td>
<td>10.56</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>42.2</td>
</tr>
</tbody>
</table>

To illustrate the savings achieved from this format, we consider the Planck test case which we discuss in more detail in Section 4.2. Table 3.1 compares the performance of COO, CSR, and the new format. Clearly, the reduction in storage is significant.

### 3.3 Spherical Harmonics Expansion

The memory required for storing the basis function pattern (D.13) for each of the $N$ basis functions can constitute a large part of the memory required for MLFMM. Storing the $(\hat{\theta}, \hat{\phi})$ components for each of the $K$ directions, see (D.19), the total storage is $2NK$ elements. Unfortunately, since $K = 2(L + 1)^2$ and $L$ scales linearly with the group diameter $D$, as can be seen in (D.22), this translates to very large values of $K$ for large groups (cf. Appendix F.2).

In [82], Eibert developed a technique to reduce the memory required for the basis function patterns by avoiding storing the $2K$ samples for each basis function, instead storing coefficients to an expansion that represents the samples.

Before detailing Eibert’s work, we remind why it is not completely straightforward, and why a careful choice of expansion functions is necessary. First, the chosen expansion must allow fast operations, such that evaluating the expansion, or finding its coefficients, takes a modest amount of work. Second, the expansion must be well suited for representing functions on a sphere with high accuracy. These two requirements have ruled out other approaches that we have considered, such as using Wavelets or the Spherical Wave Expansion
(SWE) \cite{100}, although we note that de Zuytijd et al. \cite{101} have achieved strong performance using the SWE in MLFMM. Eibert uses a Spherical Harmonics Expansion (SHE), which we briefly review in the following.

We define the Spherical Harmonic of degree $p$ and order $q$ as

$$Y_{pq}(\theta, \phi) = \sqrt{(p - q)! (p + q)!} \frac{2p + 1}{4\pi} P_p^q(\cos \theta) e^{jq\phi}$$

(3.2)

where $P_p^q$ is the associated Legendre function of degree $p$ and order $q$. With the definition in (3.2), $Y_{pq}$ are orthonormal, such that

$$\int Y_{pq}(\theta, \phi) Y_{mn}^*(\theta, \phi) d^2\mathbf{k} = \begin{cases} 1, & \text{if } p = m \text{ and } q = n \\ 0, & \text{otherwise} \end{cases}$$

(3.3)

where $^*$ denotes the complex conjugate. We note that

$$Y_{pq}^*(\theta, \phi) = (-1)^q Y_{p(-q)}(\theta, \phi).$$

(3.4)

Using the Spherical Harmonics as a basis, we can express the transmit patterns for EFIE and MFIE, respectively, as

$$\int \mathbf{f}_m(\mathbf{r}) \cdot [\hat{\mathbf{t}} - \hat{\mathbf{k}} \hat{\mathbf{k}}] e^{-j\mathbf{k} \cdot (\mathbf{r}_m - \mathbf{r})} d^2\mathbf{r} = \sum_{p=0}^{W} \sum_{q=-p}^{p} p_{pq}^m Y_{pq}(\theta, \phi),$$

(3.5)

$$-\hat{\mathbf{k}} \times \int \mathbf{f}_m(\mathbf{r}) \times \hat{\mathbf{n}}(\mathbf{r}) e^{-j\mathbf{k} \cdot (\mathbf{r}_m - \mathbf{r})} d^2\mathbf{r} = \sum_{p=0}^{W} \sum_{q=-p}^{p} q_{pq}^m Y_{pq}(\theta, \phi),$$

(3.6)

where we note that the following symmetry relations are valid for the coefficients:

$$p_{p(-q)} = \begin{cases} p_{pq}^*, & \text{if } |p - q| \text{ is even} \\ (-p_{pq})^*, & \text{if } |p - q| \text{ is odd} \end{cases}$$

(3.7a)

$$q_{p(-q)} = \begin{cases} (-q_{pq})^*, & \text{if } |p - q| \text{ is even} \\ q_{pq}^*, & \text{if } |p - q| \text{ is odd} \end{cases}$$

(3.7b)

The difference between the EFIE and MFIE coefficient symmetry is due to the $\hat{\mathbf{k}} \times$ operation in (3.6).

Using the symmetry (3.7), storing a SHE of order $W$ requires $\frac{(W+1)(W+2)}{2}$ elements for each component. Interestingly, as discussed by Eibert \cite{82}, the singularity of $\hat{\mathbf{t}}$ at the poles means that $W$ would have to be unnecessarily high to achieve sufficient accuracy for the spherical components of the basis function.
3.3 Spherical Harmonics Expansion

Figure 3.2: A comparison of the number of elements to be stored for each basis function, when using the traditional approach of storing $2K$ elements and when using SHE (with $\beta = 2$).

patterns. Thus, it is more advantageous to switch to cartesian components, since it allows a much lower $W$ — our experiments confirmed this conclusion.

The value of $W$ was given by Eibert [82] as $W = L/2 - 1$ for RWG functions. Ismatullah & Eibert [32] suggested $W = \lceil L/2 \rceil$, but also noted that in their experience, $W > 5$ was not necessary even for $L > 10$, although some of their experimental values seem to disagree with this. In any case, we opted for a more thorough approach for determining the value of $W$. Since the SHE is an orthonormal basis on the sphere, we used Parseval’s theorem in [J1], comparing with the true integral of the basis function pattern. In practice this is cumbersome, so we merely used it as a guideline to determine a more simple rule, and found that $W = \lceil L/2 \rceil + \max(\beta - 2, 0)$ corresponds to the required $10^{-\beta}$ accuracy over a wide range of $L$ values.

With this value of $W$, the number of elements to be stored for the basis function patterns is $\frac{3N(\lceil L/2 \rceil + 1)(\lceil L/2 \rceil + 2)}{2} \approx 1.5N(L^2/4 + 3L/2 + 2)$ if $\beta \leq 2$. This is a massive improvement over $2NK = 4N(L + 1)^2$, particularly for large $L$, as Figure 3.2 shows. Implementing SHE storage in MLFMM is fairly straightforward, requiring only local changes. The implementation is summarized in Appendix F.1.
Results

In summary, the method trades a longer setup time and a slightly more complicated code with a significant memory decrease, particularly so for large \( L \). The change in matrix-vector product time is, in our experience \([\text{J1}]\), a modest 5% increase. Eibert \([\text{82}]\) reported a decrease in matrix-vector product time for the RWG functions used. He et al. \([\text{102}]\) discusses a method to further reduce the SHE basis pattern storage by a factor of two for RWG discretization of CFIE. The method is not applicable for HO discretizations.

Table 3.2 demonstrates the savings achieved in the test case described in \([\text{J1} \text{ Section IVb}]\), a PEC disk with a radius of 100\( \lambda \), solved with EFIE. The machine used is detailed in Appendix \( \text{G.1} \). The table uses a scenario where adaptive grouping has been applied, comparing the use of the usual \( K = 2(L + 1)^2 \) sampling to the use of SHE. Using the standard sampling rule, and taking into account the half-sphere symmetry of the basis function patterns, we see that the basis function memory increases as the order is increased, while using SHE results in a more or less constant memory use. This comes at a minor time cost. We note that the use of SHE also yields a slightly increased group pattern storage, as described in \([\text{J1}]\). However, since the publication of \([\text{J1}]\) we have significantly reduced this overhead, to the point where it is now almost irrelevant.

Table 3.2: Comparison of SHE and the standard \( K = 2(L + 1)^2 \) sampling, both with adaptive grouping.

<table>
<thead>
<tr>
<th>Order</th>
<th>Mesh Size ( \lambda )</th>
<th>Unknowns</th>
<th>Basis Patterns</th>
<th>Relative Time pr. Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Standard [GB]</td>
<td>SHE [GB]</td>
</tr>
<tr>
<td>1</td>
<td>0.20</td>
<td>2927300</td>
<td>4.50</td>
<td>1.42</td>
</tr>
<tr>
<td>2</td>
<td>0.49</td>
<td>1889496</td>
<td>4.99</td>
<td>1.45</td>
</tr>
<tr>
<td>3</td>
<td>0.97</td>
<td>1077948</td>
<td>5.17</td>
<td>1.33</td>
</tr>
<tr>
<td>4</td>
<td>1.52</td>
<td>784576</td>
<td>6.28</td>
<td>1.50</td>
</tr>
<tr>
<td>5</td>
<td>2.38</td>
<td>494340</td>
<td>6.95</td>
<td>1.55</td>
</tr>
</tbody>
</table>

3.4 Global Interpolators

The critical difference between FMM and MLFMM, i.e., between \( \mathcal{O}(N^{1.5}) \) and \( \mathcal{O}(N \log N) \) complexity, is the use of a multi-level partitioning of the scatterer. For the Helmholtz kernel, this requires interpolation to go between the levels in the hierarchy.
The defining characteristic of these interpolation methods is whether they are based on local or global interpolation, i.e., whether each interpolated value is the result of an operation performed on a subset or on the full set of source points. Typically, local interpolation is preferred due to its simple implementation and $O(K)$ scaling, yielding the familiar $O(N \log N)$ scaling of MLFMM. However, global interpolation offers much better error control and the ability to use the highly optimized FFT routines available on most modern architectures, although it results in $O(N \log^2 N)$ scaling of MLFMM. We stress that in practice, this difference in scaling is completely irrelevant for shared-memory implementations.

Recently, a variation (TPE-MLFMM) of the original FFT-based MLFMM by Sarvas [73] was presented by Järvenpää & Ylä-Oijala [75], offering an easy and fast implementation of the necessary routines required for a global interpolation MLFMM. What is particularly appealing in [75], apart from the strong error control and a clearly written paper, is that it results in a low sample-rate implementation. This results in very low memory consumption when compared to MLFMM based on Lagrange interpolation. Järvenpää & Ylä-Oijala discussed only theoretical performance, and we extended that work in [J4], documenting strong performance for practical cases and performing a comparison with Lagrange interpolation to demonstrate the favorable properties of TPE-MLFMM. The conclusion of our work in [J4] was that TPE-MLFMM results in a low-memory implementation that is easily error controlled and is not slower than Lagrange interpolation. Furthermore, combining TPE-MLFMM with the Spherical Harmonics Expansion from Section 3.3 and adaptive grouping from Section 3.1 results in even lower memory use, and thus these techniques should be the backbone of a modern HO MLFMM implementation. Besides [J4], we refer to the analysis in Section 4.2, particularly Table 4.1 on p. 43, for performance metrics.

### 3.5 Gaussian Translation Operator

The translation stage in MLFMM involves the interaction between a large number of well-separated groups on each level. For each such interaction, the translation operator $T_L$ is multiplied onto every direction of the incoming pattern, which is then added to the receiving pattern (D.36) of the group. This stage constitutes a significant portion of the matrix-vector product time, see Table 4.2 on p. 44.

Several attempts have been made at reducing the computation time by avoiding the need to consider every direction of the incoming pattern. Wagner & Chew
Figure 3.3: A demonstration of the directive Gaussian translator. With a 60 dB dynamic range, the Rokhlin translator (a) for this specific case (translation between two groups with $D = 20\lambda$ separated by $56\lambda$) does not allow us to discard any directions on the sphere. However, for the Gaussian translator (b), a significant proportion of the directions (slightly more than 50%) can be discarded without loss of accuracy at $10^{-4}$ relative error.

Instead, Hansen developed the Gaussian Translation Operator in a series of recent papers [62–64, 104]. The translation operator is based on an exact relation and thus offers complete error control. Furthermore, for large groups that are spaced far apart, it is extremely directive.

The previously published papers considered only purely theoretical cases, casting light on the theoretical performance of the operator but not how it would behave for practical problems. While implementing the operator, it became apparent that a range of modifications were needed in order to allow an efficient implementation without additional input from the user [J3].

The main conclusion from [J3], despite the improvements discussed in that paper, is that the Gaussian Translation Operator is not really worthwhile for general scattering problems in a shared memory implementation. For such purposes, the savings are too modest to be worth the added code complexity and careful
implementation required to avoid the numerical instabilities. Further, the directionality of the Gaussian Translation Operator does not carry over when using global interpolators. It would be much more beneficial for scenarios like those described in [J3, Section IV].

3.6 Iterative Solver

When using MLFMM to solve the linear system $Z\mathbf{I} = \mathbf{B}$, an iterative solver has to be applied because we do not have direct access to $Z$, but we can apply $Z$ to a vector. The preferred iterative solver at TICRA has been the Generalized Minimum RESidual method (GMRES) [105,106]. Since GMRES can take up a significant amount of memory, TICRA has set a fairly low fixed restart parameter. However, restarting seriously degrades performance [106, p. 46] and should be avoided if possible. Therefore other alternatives have been considered, particularly for the EFIE due to its poor conditioning. Appendix E.1 briefly reviews the literature concerning iterative solvers, focusing on their use in MLFMM.

An appealing alternative to GMRES is the family of Induced Dimension Reduction (IDR($s$)) solvers [107,108], where $s$ is the dimension of the subspace, somewhat comparable to the restart parameter in GMRES. They are short-recurrence solvers, like BiCG, but appear to achieve faster convergence [108] and crucially, they do not require access to the Hermitian of the operator which simplifies the matrix-vector product code. Based on the literature and the Matlab implementation from van Gijzen & Sonneveld [109], we implemented the IDR routine in our code and conducted a wide range of experiments.

The main conclusions were as follows:

- IDR converges slower than non-restarted GMRES. If GMRES is restarted, the comparison becomes more complicated since the convergence speed is very dependent on the restart parameter for GMRES and the subspace dimension in IDR.

- IDR often exhibits extremely unstable convergence, often jumping from 10% residual norm to 0.1% and back over the course of just a couple of iterations. This is not a problem per se, but since the EFIE sometimes yields problems that are so ill-conditioned that the iterative solver will never converge, the unsteady convergence using IDR makes it difficult to judge whether a solution will be achieved in reasonable time.
Both of these points, particularly the latter, meant that we discarded IDR. Also, since the vast majority of the computational time is spent performing MLFMM matrix-vector products, we could easily construct an out-of-core version of GMRES, storing the previous vectors on the disk without a significant effect on the overall runtime (see Section 3.8).

Figure 3.4 shows the typical behaviour of the two methods. The computer used is described in Appendix G.1. The test case is as described in Section 3.1, but we stress that the behaviour shown is very typical across the wide range of test cases that we have considered during this work. We see that IDR has very unstable convergence, and that contrasted with the GMRES, it takes considerably longer to converge. We remind that for MLFMM, the time for the matrix-vector products by far dominates the time per iteration, such that it is reasonable to compare IDR and GMRES by number of iterations even though they do not require the same amount of interior work.

We also considered the application of Flexible GMRES (FGMRES) [110], as discussed in [111–114]. FGMRES uses twice as much memory as GMRES per iteration. Here, we use as an inner preconditioner the GMRES solution of a problem using only $Z_{\text{near}}$, with coarse stopping criteria: 10% relative residual norm or $\max\{11 - i, 5\}$ iterations for the $i$’th outer FGMRES iteration. Some of the research focused on using the finer levels of the MLFMM matrix-vector product in the inner problem to further reduce the number of iterations, but a few numerical tests revealed that for a significant part of our problems, the
3.7 Preconditioner

Figure 3.5: Convergence of FGMRES, as compared to IDR(20) and GMRES.

use of the very fast $Z_{\text{near}}$ matrix-vector products provided a strong compromise between time per iteration and number of iterations.

Figure 3.5 shows the performance of FGMRES against IDR and GMRES, for the same case as used in Figure 3.4. Now, since each FGMRES outer iteration requires a (coarse) iterative solution involving $Z_{\text{near}}$, the comparison is based on wall-clock time rather than iterations. Still, we see a very strong performance for FGMRES, and thus, this has become our method of choice. We note that even for test cases where the scatterer’s behaviour is less dominated by $Z_{\text{near}}$, e.g., a corner reflector, FGMRES still performs very well.

### 3.7 Preconditioner

Preconditioning is an important aspect of iteratively solving the MLFMM problem, particularly due to the large condition number of the matrices resulting from the EFIE. In some of the early research on MLFMM [13,17], the large problems were often closed structures such that the well-conditioned CFIE could be applied. The preconditioner used was often of block-diagonal type.

The most popular preconditioners in modern applications appear to be Incomplete LU (ILU) and its variants [115,119] as well as the Sparse Approximate In-
verse (SPAI) (see [120–122] for more general problems, [123–125] for EM scattering problems). The Multi-Resolution preconditioner [126] is also popular [127]. Finally, analytical preconditioners such as the Calderon preconditioner [92] are increasing in popularity. See Appendix E.2-E.3 for a discussion of preconditioning, particularly Calderon, in connection with MLFMM.

The problem with ILU-variants is that they can be difficult to efficiently parallelize [110, p. 462], and some variants have problems with potential breakdowns due to numerical instability. In contrast, the SPAI does not appear to have any breakdown problems and can be very efficiently parallelized, as it is embarrassingly parallel. Therefore, in search of an efficient preconditioner, we implemented SPAI as discussed by Lee et al. [123], which introduces three thresholds ($\epsilon_1, \epsilon_2, \epsilon_3$) used to control the size and efficiency of the preconditioner, where larger values yield a smaller and faster, but less effective, preconditioner. We compared the performance to the overlapping preconditioner currently used at TICRA [128], based on inverting overlapping blocks of the $\nabla_{\text{near}}$ matrix, where block $i, i = 1, \ldots, N_g$ contains all basis functions defined on the patches in the finest-level group $i$.

The main conclusions were as follows, where we note that the conclusions are made based on reasonable choices of the parameters $\epsilon_1, \epsilon_2, \epsilon_3$ in SPAI (in the range $0.01 - 0.1$), as discussed in [123]:

- The SPAI preconditioner is better at reducing the number of iterations than the overlapping preconditioner. Using the overlapping preconditioner results in fewer iterations than SPAI only for very high settings of the parameters $\epsilon_1, \epsilon_2, \epsilon_3$.

- The SPAI preconditioner requires significantly more setup time than the overlapping preconditioner. The difference can easily be an order of magnitude or more. This is true even when applying a large number of cores, since the overlapping preconditioner is also embarrassingly parallel.

The final conclusion is that the overlapping preconditioner is better than the SPAI for all but the poorest conditioned problems. This is due to the fast HO MLFMM matrix-vector products, and the low setup time of the overlapping preconditioner, which means that the overall time is lower when applying the overlapping preconditioner instead of SPAI, despite the reduced number of iterations with SPAI. Further, some improvements of the existing preconditioner done during a separate project at TICRA resulted in even stronger performance, making the choice to stick with the existing preconditioner an easy one.
We further remark that due to our use of the FGMRES using GMRES as an inner solver, the choice of preconditioner only affects the inner solver, and thus is less critical than for the traditional (non-flexible) iterative solvers.

### 3.8 Out-of-core Storage

Using out-of-core (OOC) storage for large amounts of data is a well-known strategy for reducing the memory footprint of an application. Instead of using RAM storage, the harddisk is used. Since harddisk storage is much cheaper, and much slower, than RAM, the trade-off is straight-forward: Larger problems can be solved, but the solution process will be slower. Thus, the key is to only use OOC for quantities that are rarely accessed. We note that the overhead for file I/O depends as much on the number of read/writes as on the number of elements accessed in each read/write—thus, it is beneficial to arrange the file I/O such that larger portions of data are read/written with each disk access.

OOC storage specifically for MLFMM has not received that much dedicated attention. Van den Bosch et al. [129] discuss block-storage of the near-matrix on the disk and Hidayetoğlu & Gürel [130] briefly discuss OOC storage of the near-matrix and the basis function patterns.

The four parts of the code that are most suited for OOC are, in order of suitability:

**Near-matrix:** The most suitable OOC component in MLFMM is the near-matrix. The use of OOC storage of the near-matrix has been discussed in the literature, but is even more applicable to HO MLFMM, because a larger proportion of the memory is spent on the near-matrix than in LO MLFMM. The OOC overhead during the matrix-vector product is modest, because the memory is accessed row-wise.

**Preconditioner:** The preconditioner $\mathbf{M}$ is, similar to the near-matrix, only accessed block-wise during the matrix-vector products. It is read block-by-block, used and then discarded. This makes the file I/O very fast and straight-forward.

**(F)GMRES:** While (F)GMRES requires a rather large number of internal operations, these operations constitute a very small part of the total time consumption. The main data, one matrix for GMRES or two for FGMRES, are accessed column-wise, allowing for simple and fast file I/O.
Table 3.3: Performance metrics for out-of-core storage of various components.

<table>
<thead>
<tr>
<th>Out of core</th>
<th>Setup [s]</th>
<th>Runtime [s]</th>
<th>Memory [GB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \bar{Z}_{\text{near}} )</td>
<td>( M )</td>
<td>Iter. Total</td>
</tr>
<tr>
<td>Everything in-core</td>
<td>498</td>
<td>16.3</td>
<td>11.2</td>
</tr>
<tr>
<td>( \bar{Z}_{\text{near}} )</td>
<td>566</td>
<td>11.8</td>
<td>12.4</td>
</tr>
<tr>
<td>( \bar{Z}_{\text{near}} ) &amp; ( M )</td>
<td>—</td>
<td>16.5</td>
<td>13.5</td>
</tr>
<tr>
<td>( \bar{Z}_{\text{near}} ), ( M ), FGMRES</td>
<td>—</td>
<td>—</td>
<td>13.5</td>
</tr>
<tr>
<td>( \bar{Z}_{\text{near}} ), ( M ), FGMRES, Patterns</td>
<td>—</td>
<td>—</td>
<td>13.6</td>
</tr>
</tbody>
</table>

Basis function patterns: Each pattern for each basis function is accessed at least once during each matrix-vector product, meaning that this requires a significant amount of file I/O. However, the time overhead can be kept under control by reading and writing large blocks at once, e.g., all patterns for each adaptive group. Further, the reduced size of the basis function patterns due to the SHE storage also reduces the overhead.

The implementation of OOC for the near-matrix and preconditioner setup was quite involved, since the filling of the matrices has complicated memory access patterns for HO MLFMM, particularly when applying EFIE.

The (F)GMRES OOC implementation, on the other hand, was straight-forward. The basis function pattern OOC implementation required some blocking of the disk accesses, but using the adaptive grouping, this was fairly simple.

To illustrate the consequences of running with OOC storage of the various components, Table 3.3 shows runtimes obtained by using the OOC code. The machine used is described in Appendix G.1. The test case is the scattering from three 5 m by 5 m PEC plates, arranged to form a corner reflector, illuminated by a 5 GHz plane wave. This case is similar to the case discussed in [C1, Section IVb], and the discretization requires 517,734 unknowns.

The open structure requires EFIE, and the FGMRES solver is surprisingly effective, requiring only 15 FGMRES iterations, with an average of 6.3 inner iterations in each FGMRES iteration. The low number of iterations means that the total time is only modestly affected by the iteration time. The iteration time is slightly increased by OOC storage of the near-matrix, because the use of FGMRES results in multiple near-matrix-vector products in each iteration. Storing the preconditioner out-of-core adds a further small overhead to the iteration time, and OOC FGMRES has no impact at all—this is due to the low number of iterations, since the amount of work in (F)GMRES increases with
the iteration number. Storing the basis function patterns on the disk results in a very modest overhead. We note that the time for preconditioner generation when $Z_{\text{near}}$ is stored OOC is actually reduced compared to when $Z_{\text{near}}$ is stored in-core. This is due to a change in the parallelization approach of the preconditioner generation algorithm, and we are still considering whether this approach should also be used when the near-matrix is stored in-core.

Overall, the conclusion is that the cost for using OOC is fairly low for the above case. We stress that while the OOC performance metrics in Table 3.3 are compared with the scenario where everything is kept in-core, one would only use OOC if there is not enough memory available. Thus, a more realistic timing comparison would be for an in-core solution where the operating system begins to swap memory to the harddisk—this would yield extreme computation times.

We also remark that since all the MLFMM data for the present case can in fact be stored in memory, the operating system might be caching some of the data, which would reduce the overhead—for the OOC performance in a scenario where MLFMM requires more memory than is available on the system, see Table 4.3 on p. 47. Finally, we note that relative to the modest overhead, the OOC implementation does a quite impressive job at reducing the memory footprint: From 6.4 GB to around 480 MB.
Tailoring MLFMM to Higher-Order Discretizations
In this chapter, we consider several test cases that are meant to highlight the performance of the HO MLFMM algorithm as a whole, although we in some places isolate the performance metrics related to a specific modification. All experiments were performed on the machine detailed in Appendix G.2, aside from the out-of-core experiments which were done on the machine detailed in Appendix G.1.

We begin with a test case where an exact reference result exists, the PEC sphere. Besides basic validation of the code, the existence of an exact analytical solution allows us to compare computational performance relative to the achieved error level across various discretizations. These comparisons were also included in [J1] and [C1]. Further, analysis of the HO MLFMM algorithm scaling against the frequency, the number of unknowns and the number of processors is performed.

We then continue with a more realistic case, taken from a project regarding the Planck Space Telescope that TICRA was involved with. The structure is quite large, and a simplified version of the telescope geometry is analyzed with our HO
MLFMM algorithm. We discuss in detail various performance metrics, showing the performance of the algorithm for this realistic, albeit simplified, case.

Finally, we consider a model of a full-size telecommunication satellite. This model has all the features of a typical satellite, including the combination of small and large geometric details, large solar panels, etc.

We note that whenever we discuss MLFMM memory, we refer to the memory required to store the entire MLFMM structure but not to solve the scattering problem. Thus, we disregard the storage required for solvers, preconditioners and geometry, but include everything required to perform a matrix-vector product; from basis function patterns and near-matrix as well as minor temporary data such as interpolation matrices and various bookkeeping. Total memory refers to everything required to solve the scattering problem, i.e., MLFMM memory plus memory required for the solver and preconditioner.

We remind that when using the term standard MLFMM implementation in this thesis, we are referring to an implementation that utilizes Lagrange interpolation, pointwise storage of $K = 2(L + 1)^2$ plane-wave directions, and Octree grouping. Further, all memory requirements are based on using the near-matrix storage format from Section 3.2.

4.1 Sphere

We begin with a standard test case in electromagnetic scattering theory, a 1 m radius PEC sphere illuminated by a plane wave. In this scenario, an analytical solution exists, known as the Mie series [131]. This provides us with an exact reference solution, based on which we can compute the Relative RMS Error as

$$\text{Relative RMS Error} = \sqrt{\frac{\sum_{i=1}^{N_s} |E_{i,\text{ref}} - E_{i,\text{cal}}|^2}{\sum_{i=1}^{N_s} |E_{i,\text{ref}}|^2}},$$

(4.1)

where $E_{i,\text{ref}}$ and $E_{i,\text{cal}}$ denote the electric far field at the $i^{th}$ sample point from the reference and calculated scattered fields respectively, and $N_s$ is the number of samples. Since the sphere is a closed scatterer, we apply the CFIE to ensure rapid convergence. We note that [28, p. 1074] remarks that the error is larger in the E-plane than in the H-plane, when computing the error based on the dB radar cross section, as is common in the literature:

$$\text{dB Error} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} (\sigma_{i,\text{ref}} - \sigma_{i,\text{cal}})^2},$$

(4.2)
where $\sigma_{i,\text{ref}}$ and $\sigma_{i,\text{cal}}$ denote the reference and calculated RCS, measured in dB. Using the Relative RMS Error, we do not see a difference in error levels between the two planes and thus only consider the E-plane.

### Validation

We begin by considering the convergence to the true solution as we vary the discretization. This allows a strong, overall verification of the code, since the theoretical behaviour of the error was determined by Nédélec \[132\] to behave as $O(h^n)$, where $h < 1$ is the patch size and $n$ is the basis function order. Thus, beyond quantifying the error for a fixed discretization, this experiment allows us to verify the code for a wide range of discretizations.

Figure 4.1 shows the results for an incident plane wave at $f = 2$ GHz, demonstrating faster convergence (steeper lines in on a log-log plot) as the order $n$ is increased. For each order $n$, the Relative RMS is varied by changing the mesh size. Further, Figure 4.1(a) also includes the results from the MoM solution in black. The MoM results are barely visible in the figure, since they are covered by the MLFMM results for most of the points, demonstrating that the results are practically identical to those obtained with MoM, except from the solution for $n = 2$ when using around 110 unknowns/$\lambda^2$. The reason for the small difference between MLFMM and MoM for that solution is most likely that we have used $\beta = 3$ and so cannot expect better than $10^{-3}$ Relative RMS. Particularly the interactions between the closest spaced groups not in the near-matrix cannot be expected to achieve better than $10^{-3}$ Relative RMS. See the discussion about (D.23) on p.\[148\]. Note that while we are using an iterative solution method with a modest stopping criterion of $10^{-4}$ relative residual norm, this is sufficient as discussed in Appendix F.6.

### Increasing the basis function order

With the code now at least partially verified\[2\] we can begin considering the performance of our HO MLFMM algorithm relative to the standard MLFMM implementation. This is shown in Figure 4.2, where we consider the memory use

---

\[1\] Strictly speaking, the $10^{-\beta}$ limit for the relative error is on the Green’s function approximation, so it does not necessarily translate to a $10^{-\beta}$ error limit on Relative RMS Error of the scattered field.

\[2\] Obviously, we have conducted a wide range of experiments on different scatterers, comparing the results with the existing MoM code and with PO, to verify that the HO MLFMM code yields correct results.
Figure 4.1 provides some interesting conclusions. First, we see that based on the standard implementation (the red curves), the basis for the research consensus that \( n = 2 \) is the optimal order is clear — we would reach the same conclusion without our modifications. However, applying the modifications, i.e., adaptive grouping and Spherical Harmonics Expansion storage, results in a significant memory reduction, for all orders, although it is much more pronounced for higher orders than for \( n = 1 \). For the \( 4^{th} \) order in particular, where the meshes result in very poor Octree grouping of the basis function patterns for the standard implementation, our modifications (particularly the adaptive grouping) yields over a factor of 5 decrease in memory. We also see a better scaling of memory relative to the Relative RMS for \( n > 1 \). Without our modifications, increasing the accuracy is costly in terms of memory, but with the modifications, the increase in memory is very modest for \( n > 1 \).

Thus, based on the results in Figure 4.2(a), we can confidently say that applying the modifications result in important reductions in memory, and with a HO MLFMM implementation including the modifications, the memory for solutions...
4.1 Sphere

(a) The MLFMM memory for varying relative RMS error and polynomial order $n$ for accuracy $\beta = 3$ in [D.22], using CFIE, with and without the modifications: Adaptive grouping & SHE.

(b) The time per iteration, normalized to 1 for the fastest run.

Figure 4.2
for \( n \geq 2 \) is essentially independent of the order. Still, this on its own does not justify using orders \( n > 2 \). However, considering the time per matrix-vector product shown in Figure 4.2(b), we see a direct connection between increasing the order and reducing the time per iteration. This is due to the reduced number of levels compared to lower orders, meaning that less interpolations, translations and anterpolations are performed. We also see that the time penalty of adding the modifications, as compared to the standard MLFMM implementation, is noticeable, representing roughly a 20% increase, but this is justifiable based on the memory reduction.

In summary, the time per matrix-vector product decreases with increasing order, suggesting us to use as high an order as possible. While increasing the order would be very costly in terms of memory in the standard implementation, our modifications allows us to increase the order with essentially constant memory. We note that further experiments done in [J1, Section IVb] suggest that for EFIE scatterers with edges, the memory for fixed RMS will in fact decrease slightly for increasing orders, since higher-order basis functions are better at representing the surface current density at the edges, where the current exhibits a singularity.

**HO MLFMM Scaling**

We now turn to the question of the scaling of the algorithm. Theoretically, the scaling of single-level FMM against the number of unknowns is \( O(N^{1.5}) \) [10,58] or, equivalently, \( O(f^3) \). For MLFMM, the scaling is \( O(N \log N) \) [11,12] or \( O(f^2 \log f^2) \). While the modifications listed previously should in theory not change the scaling of the algorithm, it is always important to test such theoretical statements in practise.

Figure 4.3 shows the scaling against the frequency for a solution with up to \( 2\lambda \) patch sidelength, both for the solution time and for the MLFMM memory. We see that the scaling is very close to \( O(f^2) = O(N) \), in part because the \( N \log N \) term has a very small coefficient for our implementation and in part because the difference between \( N \) and \( N \log N \) scaling is very hard to distinguish in a log-log plot.

We then consider the parallel performance of the HO MLFMM algorithm for an incident plane wave at \( f = 3 \) GHz. The key performance metric when considering the parallel performance of our algorithm is the parallel efficiency, defined as

\[
\text{parallel efficiency} = \frac{T_1}{pT_p}, \quad (4.3)
\]
Figure 4.3: The scaling of the computational resources as a function of the frequency. The time is the complete time to achieve the solution, and the memory includes both the MLFMM memory and that required for the preconditioner and solver. Also drawn are the regressions, showing that the algorithm effectively scales as $O(f^2)$. 
where $T_1$ is the time required to complete the task for a single core, and $T_p$ the time required for $p$ cores. Ideally, the parallel efficiency should be 100% but in practice, this is not possible. The parallel performance has not been a key point in the code development, and thus no special attention has been given to ensure the parallelizability of the algorithm. Regardless, some performance metrics are shown in Figure 4.4 for a HO discretization with patch sidelengths between $1.5\lambda - 2\lambda$, with $n = 7$ the highest order of the basis functions used. The figure includes the near-matrix fill time, iteration time and total time to achieve the solution as a function of the number of cores (between 1 and 16). The performance is acceptable, at roughly 76% parallel efficiency for 16 cores.

Interestingly, we can compare this with the parallel performance for a discretization with smaller patches, sidelengths between $0.2\lambda - 0.3\lambda$, with $n = 2$ the highest basis function order in use. This discretization results in 3 additional levels in the Octree grouping, and since there is a barrier for each level in the matrix-vector product (both in the interpolation, translation and disaggregation stages), this in turn causes poorer parallel performance. This is reflected in Figure 4.5 where we see that the total time shows a noticeably poorer scaling, at 62% for 16 cores. This is primarily due to the poorer iteration time scaling, as well as the setup of the translators and basis function patterns—we stress,
4.2 Planck Satellite

An interesting application comes from a project in which TICRA has been heavily involved, the design and validation of the reflector antenna system on the Planck Space Telescope [133]. The system is a dual reflector setup, where the paraboloidal subreflector has an elliptical rim with axes 1.05 m, 1.1 m and the paraboloidal main reflector has an elliptical rim with axes 1.55 m, 1.1 m. The telescope has a focal plane array, utilizing frequencies from 30 GHz up to 857 GHz. To shield the very sensitive system from spurious radiation from the surroundings, particularly the Sun, the system is enclosed by a screen, called a baffie. An illustration of the telescope is shown in Figure 4.6(a).

During the design of the system, a combination of Geometrical Optics and Physical Optics was used to predict the performance, with runtimes often counted in days. However, applying such methods to the complicated configuration leaves plenty of room for errors, particularly when considering that the high sensitivity of the system results in very high accuracy requirements.

Figure 4.5: Parallel efficiency for a discretization with patch sidelengths between $0.2\lambda - 0.3\lambda$.

however, that the parallel efficiency of the near-matrix fill time is unaffected by the change in discretization.
As a test of the performance of our implementation for more realistic scatterers, we constructed a mockup of the critical parts of the system. Since we did not have a CAD file of the configuration, the mesh had to be made by hand, resulting in a very crude approximation to the true baffle. The mesh is shown in Figure 4.6(b). We note that the base and sunshields below the baffle are not included, since we are only concerned with the front hemisphere behaviour, as seen from the telescope, where these components have very little effect.

We began by performing some memory tests at 20 GHz, which is not an operating frequency for the system. However, we were forced to consider this frequency, since we ran out of memory at 30 GHz with the standard MLFMM implementation when using the machine detailed in Appendix G.2. The results were included in [J4], and are summarized in Table 4.1. The table clearly demonstrates the significant benefits of the adaptive grouping, SHE and TPE-MLFMM modifications for this fairly large structure.
4.2 Planck Satellite

<table>
<thead>
<tr>
<th>Discretization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total surface area</td>
</tr>
<tr>
<td>Unknowns</td>
</tr>
<tr>
<td>Order</td>
</tr>
<tr>
<td>Mesh sidelengths</td>
</tr>
<tr>
<td>Equiv. RWG unknowns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLFMM Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
</tr>
<tr>
<td>TPE-MLFMM</td>
</tr>
<tr>
<td>TPE-MLFMM with SHE</td>
</tr>
<tr>
<td>TPE-MLFMM with SHE and adaptive grouping</td>
</tr>
</tbody>
</table>

Table 4.1: Discretization and memory requirements for Planck at 20 GHz, considering the standard MLFMM implementation and then applying various modifications.

We then moved on to consider the 30 GHz frequency, which is the lowest operating frequency for the telescope. This case was also analyzed in [J1], although the results therein were based on using Lagrange interpolation and were done in a fairly early stage of the development. The memory use for our implementation is detailed in Table 4.2, which provides several interesting conclusions.

First, we note that with the significant effort we have put into reducing the storage of the basis function patterns (Sections 3.1 & 3.3) as well as the group patterns and the translation function (Section 3.4), the by far dominant memory contribution is from the near-matrix storage, constituting roughly two-thirds of the MLFMM memory. We stress that this is not a bad thing — rather, we prefer to use a large percentage of our MLFMM memory on the near-matrix, since this allows us to have a strong preconditioner.

Considering the setup time, the majority is used to compute the near-matrix. We note that for this case, we had slightly increased the integration accuracy for the near-matrix components, which leads to a relatively higher near-matrix setup time. Still, the distribution of the setup time shown in Table 4.2 is typical for many of the problems we have considered. There is on-going research on reducing the near-matrix setup time. We also see that the setup time for the translation operator even for this fairly large case is not significant, which is somewhat in contrast with the discussion in [51, p. 110], [56,57]. However, the setup time for the basis function patterns is much more significant than reported in the literature for LO MLFMM, primarily because our use of curved patches...
Applying Higher-Order MLFMM to Large-Scale Problems

Discretization

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total surface area</td>
<td>$170 \cdot 10^3 \lambda^2$</td>
</tr>
<tr>
<td>Unknowns</td>
<td>4,583,755</td>
</tr>
<tr>
<td>Order</td>
<td>up to 9&lt;sup&gt;th&lt;/sup&gt;</td>
</tr>
<tr>
<td>Mesh sidelengths</td>
<td>$0.34\lambda - 2\lambda$</td>
</tr>
<tr>
<td>Equiv. RWG unknowns</td>
<td>$\approx 20 \cdot 10^6$</td>
</tr>
</tbody>
</table>

Performance details - Memory

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of FMM levels $(q - 2)$</td>
<td>7</td>
</tr>
<tr>
<td>$L$ values</td>
<td>776, 396, 204, 107, 58, 32, 19</td>
</tr>
<tr>
<td>Near-matrix, indices</td>
<td>0.35 GB</td>
</tr>
<tr>
<td>Near-matrix, values</td>
<td>21.1 GB</td>
</tr>
<tr>
<td>Basis function patterns</td>
<td>8.7 GB</td>
</tr>
<tr>
<td>Group patterns</td>
<td>1.7 GB</td>
</tr>
<tr>
<td>Translation operators</td>
<td>1.9 GB</td>
</tr>
<tr>
<td><strong>MLFMM memory</strong></td>
<td><strong>35.4 GB</strong></td>
</tr>
<tr>
<td>Preconditioner</td>
<td>9 GB</td>
</tr>
<tr>
<td>FGMRES memory</td>
<td>4.6 GB</td>
</tr>
<tr>
<td><strong>Total memory</strong></td>
<td><strong>45 GB</strong></td>
</tr>
</tbody>
</table>

Performance details - Time

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup time, near-matrix</td>
<td>30 m</td>
</tr>
<tr>
<td>Setup time, preconditioner</td>
<td>14 s</td>
</tr>
<tr>
<td>Setup time, translation operators</td>
<td>7.5 s</td>
</tr>
<tr>
<td>Setup time, basis function patterns</td>
<td>7 m</td>
</tr>
<tr>
<td>FGMRES iterations</td>
<td>71</td>
</tr>
<tr>
<td>Near-matrix iterations</td>
<td>360</td>
</tr>
<tr>
<td>Time per FGMRES iteration</td>
<td>81 s</td>
</tr>
<tr>
<td>Iterative solution time</td>
<td>1:35 h</td>
</tr>
<tr>
<td><strong>Total time</strong></td>
<td><strong>2:18 h</strong></td>
</tr>
</tbody>
</table>

Matrix-vector details - Time

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for near-matrix vector product</td>
<td>1.4 s</td>
</tr>
<tr>
<td>Time for aggregation</td>
<td>6 s</td>
</tr>
<tr>
<td>Time for interpolation</td>
<td>4 s</td>
</tr>
<tr>
<td>Total time for translation</td>
<td>30 s</td>
</tr>
<tr>
<td>Time for translation at level $3, ..., q$</td>
<td>$4, 4.5, 4, \ldots$</td>
</tr>
<tr>
<td>Time for anterpolation</td>
<td>4 s</td>
</tr>
<tr>
<td>Time for disaggregation</td>
<td>9 s</td>
</tr>
</tbody>
</table>

Table 4.2: Data for Planck at 30 GHz. Note that the MLFMM memory also includes some auxiliary storage.
means that we cannot use simple analytical formulas for the basis function patterns, but also due to the use of SHE.

Further, we see that the use of the FGMRES is very effective. Using GMRES, with the overlapping preconditioner discussed in Section 3.7, 151 iterations are required, each costing roughly 64 seconds, so the time saving by applying the 71 FGMRES iterations, each costing 81 seconds, is over an hour. The cost is that we nearly double the solver memory, but this is acceptable since it can be made out-of-core (see Section 3.8). Further, the doubling of solver memory when moving from GMRES to FGMRES is only relevant for a fixed number of iterations—since the number of FGMRES iterations is less than half of that in GMRES, the net result is actually a memory saving. We also see the extremely low setup time for the preconditioner. Comparing this to the time required to compute the SPAI preconditioner (roughly 30 minutes) and the 66 FGMRES iterations required when using it, the conclusion is in line with that of Section 3.7: The setup time for the SPAI preconditioner is far too high to compensate for the reduced number of iterations. This is even more true when applying the FGMRES, since the preconditioner is then only used to precondition the inner solver.

Table 4.2 also lists the details of how the time was spent in each matrix vector product. We see that the near-matrix vector product is almost free—even for very large problems. This is a big part of the reasoning behind using FGMRES with an inner GMRES based on the near-matrix. Further, we also see why it is very useful to use as few levels as possible in the MLFMM implementation; the translation times increase as we go to the coarser groups, in spite of the reduction of the number of sampling points. This is due to the increasing number of groups that need to interact. The low time for the coarsest level is due to our use of a locally-extended Octree, yielding a level where there are only few groups that need to interact. Finally, we note that the times for aggregation and disaggregation are rather high—this is due to our use of SHE and adaptive grouping. This is a bit of a drawback, because it means that if we wanted to expand our FGMRES to use an inner GMRES based on, say, the near-matrix plus a few of the finest levels of MLFMM, there is a high, fixed cost associated with this.

Figure 4.7 shows the achieved surface current density, demonstrating some interesting refraction patterns on the baffle itself. The scattered field, as computed by our MLFMM implementation, is shown in Figure 4.8. Also shown is a result achieved by TICRA using a sophisticated GO/UTD+PO/PTD hybrid algorithm, where the two approaches are hybridized such that PO/PTD takes care of a small angular region around the main beam, as well as caustics, and GO/UTD represents the remainder.
Figure 4.7: The surface current density found on Planck, shown on a dB scale covering 60 dB. Although the reflectors are obviously the strongest radiators, we see that the baffle itself is also excited. In particular, refraction patterns on the parts of the baffle closest to the reflector edges are clearly shown.

Figure 4.8: Comparison of results from MLFMM applied to the mockup in Figure 4.6(b) with a hybrid of various asymptotic methods applied to the structure in Figure 4.6(a).
4.2 Planck Satellite

<table>
<thead>
<tr>
<th>Planck - Performance on a laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total memory</td>
</tr>
<tr>
<td>Setup time, near-matrix</td>
</tr>
<tr>
<td>Setup time, preconditioner</td>
</tr>
<tr>
<td>Setup time, translation operators</td>
</tr>
<tr>
<td>Setup time, basis function patterns</td>
</tr>
<tr>
<td>Time per FGMRES iteration</td>
</tr>
<tr>
<td>Iterative solution time</td>
</tr>
<tr>
<td><strong>Total time</strong></td>
</tr>
</tbody>
</table>

Table 4.3: Data for the analysis of the Planck space telescope at \( f = 30 \text{ GHz} \) on a laptop. The discretization is the same as that used in Table 4.2.

It is important to understand the full reason behind the discrepancies between the MLFMM and the GO/UTD+PO/PTD (hybrid) solution. First, the structures considered are simply different — the hybrid solution considers Figure 4.6(a) while MLFMM considers the mockup in Figure 4.6(b). Second, the output sampling is very coarse in the hybrid result to keep the runtime low. The coarse sampling gives the appearance of a much slower variation compared to MLFMM, which is not the case as we confirmed by closely-spaced evaluation of the hybrid solution in a small angular region. Third, the most significant discrepancy is around \( \theta \approx 50^\circ \), which is a region where the hybrid method has had some difficulties when being compared to actual measurements. Thus, we actually expect that the MLFMM result is the most accurate of the two in this region. We note that for the peak directivity, where the hybrid solution uses PO/PTD and is very accurate, the discrepancy between the two results is 0.003 dB, a relative deviation of less than \( 10^{-4} \).

In summary, the results are very acceptable and suggest that even with the fast solution and low memory use, the high accuracy that we have seen from other test cases is preserved.

As a further demonstration of the low memory use, we analyze the Planck space telescope at 30 GHz on the laptop from Appendix G.1. Since this does not have sufficient memory to run the problem in-core, we apply the out-of-core techniques outlined in Section 3.8 to store the near-matrix, preconditioner, FGMRES solver and basis function patterns on the harddisk. Table 4.3 shows some performance metrics from the run.
4.3 Emerald Satellite

As our final test case, we move to a model of a satellite based on a CAD file. The satellite, named Emerald, is used primarily as a test object for CAD processing in various computational electromagnetics software, and is therefore designed to mimic a realistic satellite design.

Before considering the MLFMM solution, it is important to stress that the discretization of such a structure, i.e., moving from a CAD file to a quadrilateral mesh, is a very difficult task. The topic is out of the scope of the present thesis, but we underscore that the discretization achieved in Figure 4.9(b) is the result of a very complicated process. In particular, the use of the Discontinuous Galerkin Integral Equation (DGIE) from Peng et al. [134] removes the otherwise ubiquitous requirement that neighbouring patches have to share vertices in order for the normal component of the current to be continuous across the patches. Instead, using DGIE, they merely have to share (parts of) the edge to allow a physical current to flow. This makes the meshing of complicated structures much easier.

We consider the lowest frequency in the Ku band, 10.7 GHz, and simulate a plane wave incident on the structure. In the mesh shown in Figure 4.9(b), the yellow coloring of the main body of the satellite indicates that the meshing procedure has automatically detected a closed region, allowing us to compute a hybrid CFIE-EFIE solution (see, e.g., Ergül & Gürel [135]), providing much
4.3 Emerald Satellite

(a) The computed surface current density on the Emerald satellite model. (b) Zoom on part of the structure, showing obvious shadowing effects on the structure due to the plane-wave illumination. Both the small cones on the side of the structure, and the hinges for the solar panels, shield part of the body of the reflector.

Figure 4.10

faster iterative convergence than a pure EFIE solution. The discretization yields 3,617,887 unknowns on patches with edge lengths between $0.02\lambda - 2.64\lambda$, using up to 10th order basis functions on the largest patches. About a third of those basis functions are on EFIE-only patches, i.e., the solar panels.

The MLFMM solution uses a 12-level Octree and requires around 80 GB of MLFMM memory, most of it used for the 55 GB near-matrix. The preconditioner takes up an additional 26 GB, while the FGMRES solver requires around 4.6 GB, for a total memory consumption of around 110 GB. Iterative convergence is achieved in 86 iterations, and the total run time is 3 hours, with the near-matrix fill time around half an hour and each iteration requiring 90 seconds for a total iteration time of 2:10 hours.

The surface current density is shown in Figure 4.10. We show both a view of the entire structure, at an angle roughly corresponding to that from Figure 4.9, as well as a zoom of part of the structure, where the plane-wave illumination leads to some interesting effects such as shadowing and refraction patterns on the body of the satellite. While the solution is obviously hard to verify, since no measurements exist, the currents are as expected.
In this thesis, we have considered the computational analysis of electromagnetic scattering from electrically large structures. The focus has been on “fast” methods, which improve the asymptotic scaling of the required computational resources against the frequency, in an attempt to allow users to solve larger problems on their existing hardware.

As a first step, we surveyed the existing literature on the Multi-Level Fast Multipole Method (MLFMM), with particular focus on combining MLFMM with a Higher-Order (HO) discretization (HO MLFMM). It was apparent that while a lot of work has been done on improving various components of an MLFMM implementation, little of this work had been applied to HO MLFMM. Further, while several research groups had considered HO MLFMM and had concluded that it did not allow a significant reduction in computational resources, this research seemed somewhat incomplete; a rigorous and careful HO MLFMM implementation with hierarchical HO basis functions had not been attempted.

After implementing the standard MLFMM, we considered a wide range of modifications that would potentially allow us to successfully adapt MLFMM to a HO discretization. Some modifications, such as adaptive grouping and a more suitable near-matrix format, were developed by us to tackle problems specific to HO MLFMM. Other modifications, such as the Spherical Harmonics Expansion storage and the use of global interpolators, were taken from the MLFMM
literature because they are particularly suitable for HO MLFMM. Further, we considered some approaches such as an improved iterative solver and out-of-core storage that, although not developed particularly for MLFMM, yields significant benefits.

We then considered a range of test cases, designed to illustrate the performance of the algorithm. First we considered scattering from a sphere, which allowed us to validate the code against an exact analytical solution. This scenario allowed us to obtain very good performance with our HO MLFMM algorithm. Hence, we disagree with the conclusion of the previous literature on the topic; we found that with a careful implementation of HO MLFMM, including modifications to the standard MLFMM code, we could achieve significant reductions in the required computational resources by increasing the basis function order and patch size. This is an important conclusion, because it allows, for the first time, the combination of the benefits in a HO discretization with the strong scaling of MLFMM against the frequency.

We also considered more realistic scatterers, specifically models of two satellites. The first model, of the Planck Space Telescope, consisted of a fairly coarse mockup of the true satellite geometry. However, its simple design allowed validations against another computational method, used during the design of the telescope. We found a strong agreement between the computed scattered fields, thereby further validating the code for more realistic problems. We also showed that with our HO MLFMM implementation, including the out-of-core components, we were able to solve this 4.5 million HO basis function problem on a laptop in 12 hours. To our knowledge, analysis of scatterers of this size has not previously been feasible on standard laptops.

The second satellite model was directly imported from a CAD file, yielding a very realistic scattering problem, including the mixing of both very small and very large geometric details, providing a strong challenge for an electromagnetic solver. We were able to solve this problem, the analysis of a full satellite at K\textsubscript{u} band, in 3 hours. This appears to be at least an order of magnitude faster, and requiring an order of magnitude less memory, than existing commercial MLFMM solvers.

However, further research showed that with our effective HO MLFMM solver, we had merely moved the bottleneck in the computational analysis of electromagnetic scattering. While finding the surface current density using MLFMM is the first step, scaling as $\mathcal{O}(f^2 \log f^2)$ against the frequency $f$, the second step is the evaluation of the scattered field, scaling as $\mathcal{O}(f^4)$. Thus, as we continued to solve larger problems, we ran into a second bottleneck. Further, in practice, many scattering problems are not solved by applying MLFMM but instead
by applying Physical Optics (PO). The bottleneck in the PO algorithm is the evaluation of radiated fields from surface current densities.

Thus, it was clear that to allow the solution of large scattering problems, we also had to accelerate the evaluation of radiated fields. Our research on this topic, discussed in Appendix B led us to conclude that the existing literature did not offer algorithms that were fully suited for our demands. For observation points in the far-field, we required a much higher accuracy, with full error control, than previously presented in the literature. For observation points in the near-field, we required higher accuracy as well as acceleration for scenarios where the observation points were located close to the scatterer.

To fulfil our demands, we thus had to significantly improve the existing algorithm for far-field observation points and develop a new algorithm for near-field observation points. While these developments were deemed confidential by TICRA, we were able to demonstrate the performance of the algorithms on a number of test cases. These results clearly showed that the algorithms achieved a scaling of at most $O(f^2 \log f^2)$ for field evaluation, successfully alleviating the bottleneck in the solution process, while maintaining the desired accuracies.

In conclusion, the techniques described in the present thesis allows the solution of much larger electromagnetic scattering problems than previously described in the literature, without forcing users to acquire new hardware, and without sacrificing the high accuracy required in many practical applications.
Appendix A

Papers
A.1 Paper I

MULTI-LEVEL FAST MULTipoLE METHOD FOR HIGHER-ORDER DISCRETIZATIONS

Oscar Borries, Peter Meincke, Erik Jørgensen, and Per Christian Hansen

Status

Published: September 2014

Bibliography

Multilevel Fast Multipole Method for Higher Order Discretizations

Oscar Borries, Student Member, IEEE, Peter Meincke, Member, IEEE, Erik Jørgensen, Member, IEEE, and Per Christian Hansen

Abstract—The multi-level fast multipole method (MLFMM) for a higher order (HO) discretization is demonstrated on high-frequency (HF) problems, illustrating for the first time how an efficient MLFMM for HO can be achieved even for very large groups. Applying several novel ideas, beneficial to both lower order and higher order discretizations, results from a low-memory, high-speed MLFMM implementation of a HO hierarchical discretization are shown. These results challenge the general view that the benefits of HO and HF-MLFMM cannot be combined.

Index Terms—Fast multipole method, higher order basis functions, integral equations.

I. INTRODUCTION

T HE electromagnetic scattering problem can be solved by setting up a surface integral equation for the unknown surface current density. The discretization of the integral equation is often done using the method of moments (MoM) with local-domain basis functions [1]. The resulting system of linear equations is subsequently solved using either a direct or an iterative approach. The latter is the only viable option for large-scale problems, though some research has been done in applying direct methods to large problems [2].

Unfortunately, the matrix itself quickly grows too large to store as the electrical size of the problem increases. With $N$ unknowns, the memory cost of storing the matrix increases as $O(N^2)$; and so does the computational cost of the matrix-vector products required during the iterative solution. Both of these factors make the iterative MoM approach prohibitive. The computational electromagnetics community has sought to avoid this by using two different approaches; either by reducing the number of unknowns required to achieve a given accuracy or by applying fast solution methods that have an asymptotic complexity less than $O(N^2)$.

The first of these approaches has primarily involved the study of basis functions and geometric discretizations with more desirable properties than the standard Rao–Wilton–Glisson (RWG) functions [3], which are first-order basis functions in the direction of the current, hereafter defined as lower order (LO). Improved basis functions can be obtained simply by defining basis function families that allow increasing the order of the RWG functions [4]. Other function families, which have the desirable property that they are hierarchical, have also been considered [5]–[7]. The hierarchical definition of the functions allows the expansion order to be chosen independently for each geometrical element (patch). A required supplement to higher order basis functions is the use of curved geometrical elements [4]–[6], that allows each element to represent a larger part of a curved scatterer and thus increases the support of the basis functions. The choice of basis functions and geometrical elements is critical for the accuracy of the solution and the number of iterations required for convergence [6].

The second approach has resulted in several new methods, some of which are still being refined. The most popular one seems to be the fast multipole method [8], [9] and its hierarchical variant, multilevel fast multipole method (MLFMM) [10]–[12].

Early research on combining the benefits of HO discretizations and MLFMM was done in [13], concluding that the larger patch sizes involved in HO discretizations severely limit the performance. To alleviate this problem, another approach was suggested in [14], where a Nyström-type approach is used to reduce the memory requirements. However, no comparisons between varying polynomial orders were done, and the savings achieved with this approach are significantly lower than those presented in the present paper. Later research focused on choosing basis functions suitable for MLFMM [15], thereby sacrificing the generality of the approach. Another line of research has focused on modifying some of the aspects of MLFMM [16], thereby alleviating some of the concerns regarding HO implementations. However, the results [17]–[19] only considered polynomials of first and second orders, as did [20], which describes an interpolatory HO implementation. Although [21] compared the number of iterations for a few implementations of HO function families, it did not consider the memory nor the accuracy of these implementations. Also, the effects of varying the basis function order were not considered.

More recently, research has switched to focusing on standard implementations of MLFMM with higher order, hierarchical discretizations [22]–[24]. However, these papers are rather brief and they suggest some suboptimal choices in the underlying implementations, causing concerns about whether the conclusions are viable and, more importantly, they fail to demonstrate higher order convergence. The explicit conclusion from [13],
[17], [22], [24] is that going beyond second-order is not worthwhile for MLFMM, either due to memory or accuracy concerns.

The main contribution of this paper is to challenge the previously established conclusion: HO convergence can be achieved while maintaining an efficient, low-memory, high-speed MLFMM implementation for high-frequency problems, and going beyond second order is indeed advantageous. The major contributions of this paper can be summarized as follows.

1) We demonstrate that the higher order convergence of the basis function set is maintained when using our MLFMM implementation.
2) We show that the memory cost of using HO discretizations with our proposed modifications (see below) to MLFMM is far lower than previously described in the literature.
3) We show that the computational time for a matrix-vector product is significantly lower using HO MLFMM compared to LO MLFMM.

To arrive at the accurate, low-memory, high-speed HO MLFMM scheme, several parts of the standard MLFMM implementation must be modified. The key modifications necessary for obtaining the new HO MLFMM implementation of the present paper are as follows:

1) We use a sparse matrix storage format suited for HO implementations.
2) We introduce a new grouping scheme, significantly reducing the memory required, especially for nonuniform discretizations.
3) We demonstrate the savings achieved by Eibert’s SE-MLFMM [16] for hierarchical HO discretizations of arbitrary order.

Some of the modifications are novel and of interest to both HO and LO MLFMM, but our main contribution is the revision of the paradigm that HO MLFMM is not worthwhile.

The paper is organized as follows. Section II describes the discretization employed, highlighting the key parameters to be used later. Section III then recaps the main points of an MLFMM implementation, focusing on the key differences imposed by the use of HO bases and introducing the novel contributions. Section IV analyzes three different test cases, discussing the relationship between computation time, memory usage and accuracy for various discretizations. Section V concludes on the results. The major notation is as follows.

- Scalar quantities are typeset using italics. Physical vectors, either with two or three components, are typeset using bold. Other vectors are overlined while matrices are double overlined.

II. DISCRETIZATION

The fundamental problem to solve is the integral equation (EFIE) formulation [25]. Denoting the surface $\mathcal{S}$, the EFIE is then

$$\mathcal{L}J_{\mathcal{S}} - \hat{n} \times E^i (1)$$

where $\hat{n}$ is a unit vector normal to $\mathcal{S}$, $E^i$ is the incident electric field, and $J_{\mathcal{S}}$ is the surface current density. $\mathcal{L}$ is the integral operator

$$\mathcal{L}J_{\mathcal{S}} = \hat{n} \times j\omega \mu \left[ \int_{\mathcal{S}} J_{\mathcal{S}}(r') G(r, r') d^2 r' \right.$$

$$+ \frac{1}{k^2} \nabla_{\mathcal{S}} \cdot J_{\mathcal{S}}(r') \nabla G(r, r') d^2 r' \left.] \right)$$

where $\mu$ is the free-space permeability and $k = 2\pi / \lambda, \lambda$ being the free-space wavelength. $G(r, r')$ is the free-space Green’s function $G(r, r') = e^{-jkr} / 4\pi r - r'$ and $r, r'$ denote observation and integration points, respectively.

To eliminate the interior resonances of the EFIE operator it can be combined with the magnetic field integral equation (MFIE) [25], which for a smooth, closed scatterer is

$$\left( \frac{1}{2} \mathcal{I} + \mathcal{K} \right) J_{\mathcal{S}} - \hat{n} \times H^i$$

in which $H^i$ is the incident magnetic field, $\hat{n}$ is an outward unit normal vector, $\mathcal{I}$ is the identity operator, and $\mathcal{K}$ is the operator

$$\mathcal{K}J_{\mathcal{S}} = \hat{n} \times \int_{\mathcal{S}} J_{\mathcal{S}}(r') \times \nabla G(r, r') d^2 r'$$

where $\mathcal{I}$ denotes the Cauchy principal value. This yields the combined field integral equation (CFIE) [25]

$$\left[ \alpha \mathcal{L} + \left( 1 - \alpha \right) \eta \frac{1}{2} \mathcal{I} + \mathcal{K} \right] J_{\mathcal{S}} - \alpha \hat{n} \times E^i + \left( 1 - \alpha \right) \eta \hat{n} \times H^i.$$}

Here, $\eta = \sqrt{\mu/\epsilon}$ is the free-space impedance, $\epsilon$ is the free-space permittivity and $\alpha \in [0, 1]$ is a weighting factor.

Through the Galerkin approach, a matrix equation $J = V$ is obtained, the solution to which will yield the required surface current densities—whose accuracy depends on the discretization. This motivates the discretization used in the present implementation, taken from [6], [26], where both the current $J_{\mathcal{S}}$ and the surface geometry $\mathcal{S}$ are discretized using a higher order approach. The geometry is discretized with curved quadrilaterals [4], such that a point $r(u, v)$ on a $p$th order patch can be expressed as

$$r(u, v) = \sum_{i=0}^{p} \sum_{j=0}^{p} r_{ij} \phi_i^p(u) \phi_j^p(v)$$

where $r_{ij}$ denotes an interpolation node and $\phi_i^p(u)$ is the Lagrange polynomial of $p$th order

$$\phi_i^p(u) = \prod_{k=0, k \neq i}^{p} \frac{u - u_k}{u_i - u_k}$$

where $u_i$ is the parametric coordinate of the interpolation node.
The currents are discretized using modified Legendre polynomials along the direction of the current, and Legendre polynomials in the transverse direction. Hence, $\mathbf{J}_S$ is expanded as

$$\mathbf{J}_S = J_S^x a_x + J_S^y a_y$$

where $a_x = \partial r/\partial x$ is the covariant unitary vector and similarly for $a_y$. Considering a $x$-directed current, $J_S^x$ is thus expanded as

$$J_S^x(u,v) = \frac{1}{J_S^x(u,v)} \sum_{m=0}^{M^x} \sum_{n=0}^{N^x} a_m r_m P_m(u) C_n P_n(v)$$

where the modified Legendre polynomials $\bar{P}_m$ are defined as

$$\bar{P}_m(u) = \begin{cases} 1 - u & m = 0 \\ 1 + u & m = 1 \\ P_m(u) - P_{m-1}(u) & m \geq 2 \end{cases}$$

and $P_m$ are the Legendre polynomials of order $m$. The expansion for a $y$-directed current is obtained by interchanging $u$ and $v$ in (9). Moreover, $C_m$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$. To satisfy the Nedelec constraint, $\mathbf{J}_S^x$ is a factor chosen to minimize the condition number of $J_S^x$.

III. MLFMM FOR HIGHER ORDER BASES

The MLFMM is a procedure for performing the operation $\overline{Z} T$ in $O(N \log N)$ time and memory, where $N$ is the number of basis functions. MLFMM relies on a hierarchical grouping of the patches, often done using an Octree [28], based on the center points of the patches. The interactions between closely positioned groups are computed as usual in MoM, resulting in a sparse near-matrix $\overline{Z}$ containing the elements from $Z$ that cannot be approximated by MLFMM.

Interactions that can be approximated with MLFMM are computed by performing an integral over the sphere, which allows for computing interactions between entire groups of basis functions simultaneously. This process involves gathering the radiation pattern of a group by summing up the excitations of the basis functions in the group, translating this using the Rokhlin transfer function (12), and then integrating against the receiving pattern of a testing group. Furthermore, by utilizing the hierarchical scheme, involving interpolations due to differing sample rates, interactions between larger and larger groups can be computed using this integral.

This yields a splitting of the operation $\overline{Z} T$ into

$$\overline{Z} T \simeq \overline{Z}_{\text{near}} T + \mathcal{F}(\overline{T})$$

It has been shown [11] that $\overline{Z}_{\text{near}}$ contains $O(N)$ elements, thus requiring $O(N \log N)$ memory and computational time, while the operation $\mathcal{F}(\overline{T})$ requires $O(N \log N)$ memory and computational time.

The essential part of MLFMM is Rokhlin’s translation function [8]

$$T_L(k, x) = \sum_{l=0}^{L} (-j)^l (2l + 1) h_l^{(2)}(k | x) J_l(k \cdot x)$$

where $k$ is the unit wave vector, $x$ is the vector between two group centers, $\overline{Z}_{\text{near}}$, and $h_l^{(2)}$ is the spherical Hankel function of second kind and order $l$. It is important to note that $T_L$ does not depend on the absolute position of the groups, but only on the vector $x$ between their centers. Thus, $T_L$ can be reused for pairs of groups with the same $x$; a key factor in keeping memory requirements low.

The truncation number $L$ from (12) is determined from the excess bandwidth formula [29]

$$L = k D + 1.8 \beta^2 (k D)^{1/2}$$

where $D = 3a$ is the diameter of the group, $a$ is the side-length, and $10^{-\beta}$ is the desired relative error. We apply the one buffer-box criterion $N_{\text{BUF}} = 1$, such that interactions between groups that are well-separated, i.e., $|x| > N_{\text{BUF}} D$, are calculated by MLFMM. We note that this can potentially cause problems for small group sizes—this problem is treated in detail in [30]. However, the approach from [30] yields extremely conservative estimates, which significantly increase the resource consumption, especially for small group sizes. Thus, to avoid being unfair towards LO basis functions, we do not use the methods from [30] but only warn that this can yield a pessimistic upper bound to the accuracy achievable, particularly for LO discretizations.

With $T_{kr}$, the mutual impedance between two basis functions $f_j$ and $f_l$, located in groups $m$ and $m'$, respectively, that are well separated, can be expressed as

$$Z_{j,l} \approx \kappa \int_{S_j} \overline{R}_{jm}(k) \cdot (T_L(k, r_{mm'}) V_{lm}(k)) d^2 \overline{k}$$

where $r_{xy}$ is defined as $r_{xy} = r_x - r_y$ and $r_{mm'}$ is the center of the group $m$. The basis function patterns $V_{jm}$ and $R_{jm}$ for EFIE are

$$V_{jm}(k) = \int_{S_j} f_j(r) \cdot \left[ \overline{\mathbf{I}} - k \mathbf{k} \right] e^{-jk(r_{mm'}-r)} d^2 \mathbf{r}$$

and for MFIE

$$V_{jm}(k) = -k \times \int_{S_j} e^{-jk(r_{mm'}-r)} f_j(r) \times \nabla(r) d^2 \mathbf{r}$$

$$R_{jm}(k) = \int_{S_j} e^{-jk(r_{mm'})} f_j(r) d^2 \mathbf{r}$$

where $\mathbf{k}$ covers the patch $S_j$ on which $f_j$ is defined. For EFIE, $\kappa = -j(\eta/4\pi)$, while for MFIE $\kappa = -(\eta/4\pi)$. 
We note that only the $\hat{\theta}$ and $\hat{\phi}$ components of the basis function patterns are nonzero. We also mention that there is a symmetry [31] that can be employed to store only half of the basis function patterns.

When (14) is evaluated numerically, the expression is rewritten through the substitution $t = \cos \theta$

$$Z_{j,i} \approx \kappa \int_0^{2\pi} R_{jm}(k) \cdot (T_L(k, r_{mm'}) V_{im'}(k)) \, dt$$  \hspace{1cm} (18)

which is then discretized to

$$Z_{j,i} \approx \kappa \sum_{p=1}^{K} w_p R_{jm}(k_p) \cdot (T_L(k_p, r_{mm'}) V_{im'}(k_p)) \cdot (20)$$

Here, $K = 2(L + 1)^2$ is the quadrature points; $L + 1$ Gauss–Legendre points in $t$ and $2(L + 1)$ equidistant points between 0 and $2\pi - 2\pi/(L + 1)$ in $\phi$. $w_p$ are the corresponding quadrature weights.

The above expressions are valid for a single-level scheme. With two levels, (19) becomes

$$Z_{j,i} \approx \kappa \sum_{p=1}^{K} \sum_{n=1}^{K} w_p w_n R_{jm}(k_p) \cdot (T_L(k_p, r_{mm'}) V_{im'}(k_p)) \cdot \phi$$

Here, the superscript $(q)$ denotes quantities at the $q$th level, with lower numbers indicating coarser levels, i.e., the diameter of the groups increases as $q$ decreases, such that $D^{(q)} = 2D^{(q+1)}$.

Further, $m_q$ is the group at the $q$th level containing group $m$, and levels 1 (the original bounding box) and 2 are not used in the MLFMM.

The above approach can be applied hierarchically for additional levels, i.e., the centers of the plane-wave expansions are moved from the center of groups at successive levels by the phase-center movement $e^{-j2\pi r_D}$. As (20) reveals, the key difference between the single- and multilevel scheme is that for multi-level we need an interpolation routine to get from one level to another in the Octree, since the sampling rates differ (13). In the present implementation, this is done by Lagrangian interpolation with anti-spherical and spherical boundary conditions on the $\hat{\theta}$ and $\hat{\phi}$ components, respectively.

$$f(\theta, \phi) = \begin{cases} f(2\pi - \theta, \phi + \pi) & \text{Spherical,} \\ -f(2\pi - \theta, \phi + \pi) & \text{Anti-spherical,} \end{cases} \quad \theta \in [\pi, 2\pi].$$

To avoid allocating both outgoing and incoming radiation patterns for each group, we reuse the storage such that memory is allocated for the outgoing patterns for each group plus a reusable space for the incoming patterns on each level.

Having set the stage and described the basic concepts of a standard MLFMM implementation, we now turn to the novel contributions of the paper, as detailed in the Introduction.

**A. Near-Matrix Elements**

An important consideration in HO-MLFMM is that each half of a basis function overlapping two patches, also referred to as an *edge basis function*, should be placed into separate groups. The scenario is illustrated in Fig. 1. Here, we see that half of the edge basis function $e$ is placed in group 1, while the other half is placed in group 2. As a consequence, when considering the interaction of function $e$ with function $b$ in group 6, the interaction will be split between the near-matrix and the MLFMM operation.

An important part of the standard MLFMM implementation is the storage format for the sparse near-field matrix. A common choice is a block storage format, where each block represents the near-interaction between two groups. Unfortunately, a block format is impractical when basis functions span two groups, since the contributions to element $Z_{j,i}$ will require a contribution from both $\bar{Z}_{near}$ and (20). In our implementation, we allow for this and thus let the group size equal the largest patch size, rather than expanding the groups such that $Z_{j,i}$ is entirely contained in $\bar{Z}_{near}$.

Another typical choice is the compressed sparse row (CSR) format [32], which is well suited for matrix-vector products. It does, however, have the unfortunate property of spending more than one-third of the storage of the near-matrix on storing integers. To avoid this, another approach is used here.

For most discretizations, where the basis functions belonging to a given group are placed consecutively in the impedance matrix, the elements in the near-matrix lie in contiguous blocks. This is particularly pronounced for HO discretizations, where there is a large number of basis functions on each patch.

We thus propose to modify the CSR format by not explicitly storing consecutive column indices. Instead, we store the first index in such a set, and store the number of consecutive indices in the set as a negative number. Thus, we only store roughly $2t$ integers for column indices, where $t$ is the number of sets in
This has the advantage of being completely adaptive to the discretization and requiring only minor modifications to existing code based on CSR, as well as being just as parallelizable and easy to modify for the symmetric case.

B. Well-Separated Elements

The limiting factor of performance in HO MLFMM is the size of the domains. The following presents two techniques, one novel and one which is adopted from [16], which together drastically reduce the memory required for storing the basis function patterns. Note that while these are relatively more advantageous for HO discretizations, they will still reduce the memory requirements, even for LO MLFMM. Whether they are worthwhile or not for LO MLFMM is not of concern here, although it is demonstrated in Section IV.

1) Adaptive Grouping: The major problem with the large patches in HO discretizations in connection with MLFMM is the very significant discrepancy between the size of the patches and the size of the finest group. Since for a given scatterer the group size at each level of an Octree is fixed, and since it is not worthwhile partitioning a patch into distinct groups, we easily risk a scenario where the sidewidth of the group is close to twice the patch sidewidth. This is extremely inefficient because the basis function patterns will have a far larger bandwidth than necessary, and a new approach has to be devised.

We propose an adaptive grouping, which essentially adds a separate, extra layer, called the adaptive level, at the finest level. This layer is distinct from the Octree grouping, such that the center and group size of each adaptive group can be chosen as desired. In principle, this works analogously to an extra layer in the Octree, but crucially, it is not restricted by the same rules as those that apply to the Octree grouping scheme. The advantages are:

- The center of each group can be chosen such that the region of validity is as small as possible, while still being conformal to the patch inside each group.
- For highly irregular meshes, the Octree grouping scheme results in groups that are not smaller than the largest patch, which might be a very poor choice in very finely meshed regions. The adaptive grouping allows for arbitrarily sized groups in different regions of the mesh.
- There is little need to carefully choose the patch size to fit with the Octree grouping, as [17] mentions is necessary if HO MLFMM is to be useful. With adaptive grouping, the bandwidth of the basis function patterns will fit the patch size.
- This approach allows us to efficiently solve the classic problem of patches sticking out of the groups. Instead of expanding all groups at the lowest level, one can instead expand just the adaptive group and the groups at higher levels, which is less costly since the groups now only hold aggregated patterns instead of basis function patterns. We stress that there is no translation done on the adaptive level, and therefore the near-interaction matrix is still based on the finest level in the Octree, and there is no loss of translation accuracy from the adaptive grouping since the categorization of well-separated elements is unaffected. The only change from the usual MLFMM scheme is thus that the basis function patterns are tabulated on the adaptive level. Fig. 2 illustrates the effects of the reduced size of the adaptive group.

The work by Pan et al. [20] discusses an approach to locally extend the Octree, providing extra levels that allow a lower sampling rate. The extra levels are obtained by further subdivision of the finest level of the Octree. Their approach differs from the adaptive grouping presented here in a few key areas:

- Being based on further subdivision, their approach does not allow for arbitrarily sized groups.
- Further only allowing for further halving of the group diameters. Adaptive grouping allows for any group size, adaptive to the specific patch, providing the lowest possible sampling rates.
- Their approach does not allow for an adaptive group center. This is a key feature of adaptive grouping, as it allows for significantly lower sampling rates, even for very irregular meshes where the finest level groups of the Octree has its centers positioned far from the patches.
- Both of these advantages of adaptive grouping over the technique presented in [20] are particularly important for very large patches. [20] did not consider patches larger than 0.2\(\lambda\).

When implementing the adaptive grouping in the matrix-vector product, (19) is thus expanded to

\[
Z_{j,m} \sim \kappa \sum_{p=1}^{\infty} w_p R_{j,m}(\mathbf{k}_p) \cdot \left( \overline{W} e^{-j\mathbf{k}_p \cdot \mathbf{r}_{m,m}} \right) \\
T_{E}(\mathbf{k}_p, \mathbf{r}_{m,m'}) e^{-j\mathbf{k}_p \cdot \mathbf{r}_{m,m'}} \overline{W} \mathbf{V}_{i,m'}(\mathbf{k}_p) 
\]

(21)

where \(\mathbf{r}_{m,m'}\) is the adaptive group in which \(f_{j,p}\) resides and equivalently for \(\mathbf{r}_{m'}\) and \(f_{j,p}\). Thus, we can express the adaptive grouping as an expansion of the basis function pattern \(R_{j,m}\) or \(\mathbf{V}_{i,m'}\) from (19) as

\[
\mathbf{V}_{m,m'}(\mathbf{k}) = e^{-j\mathbf{k} \cdot \mathbf{r}_{m,m'}} \overline{W} \mathbf{V}_{i,m'}(\mathbf{k}).
\]

(22)

Aside from the controllable interpolation error, (22) is exact.
Since this modification requires some additional auxiliary data, in particular an interpolation matrix $W$, a practical implementation would choose a number of possible group sizes, and then categorize the adaptive groups into these, instead of having auxiliary data for every adaptive group. For fairly uniform meshes, only 2 or 3 possible group sizes are required, while strongly nonuniform meshes might require a few more. In any case, the auxiliary data is negligible, but has been included in the memory counts in the numerical results presented later.

2) Spherical Harmonics Expansion: In [16], Eibert develops a method for storing the basis function patterns as coefficients to a spherical harmonics expansion (SHE), which drastically reduces the required memory. The key equations are repeated here for completeness.

We expand the basis function patterns (either (15) or (17)) can be used here as

$$\int_{S} \mathbf{f}_j(r) \left[ \mathbf{f} - k \mathbf{k} \right] = i^k \mathbf{f}[r] \mathbf{d}^2 \mathbf{r} = \sum_{\nu=0}^{W} \sum_{p} \mathbf{p}^*_{pq} Y_{pq}^{*}(\theta, \phi) \mathbf{d}^2 \mathbf{k}. \quad (23)$$

where $Y_{pq}$ are the orthonormalized spherical harmonics, $P_{pq}^*$ are the associated Legendre functions, and $W$ is the order of the SHE. The coefficients $P_{pq}^*$ are then stored instead of the sampled basis function patterns. The coefficients are computed numerically via the integral

$$\mathbf{p}^j_{pq} = \iint \mathbf{V}_{j}(\theta, \phi) Y_{pq}^{*}(\theta, \phi) \mathbf{d}^2 \mathbf{k}. \quad (24)$$

Although $V$ is stored most efficiently in $(\theta, \phi)$-components, $W$ can be made significantly lower by using $(\hat{x}, \hat{y}, \hat{z})$-components due to the singularity of $\mathbf{V}$ at the poles.

Further, converting the incoming fields as

$$T_{L}(\mathbf{k}, r_{mm'}) \mathbf{V}_{lm'}(\mathbf{k}) = \sum_{p} \sum_{q} \mathbf{p}^j_{pq} Y_{pq}^{*}(\theta, \phi) \quad (25)$$

we can convert (19) to

$$Z_{i,j} = \kappa \sum_{p} \sum_{q} (\mathbf{p}^*_{pq})^* : \mathbf{r}^j_{pq}. \quad (26)$$

The procedure for computing a matrix-vector product is only changed at the finest level. Here, we aggregate the SHE coefficients, evaluate the resulting radiation pattern for the group, convert to $(\theta, \phi)$ components, and then continue as in the standard MLFMM procedure. For disaggregation, we convert the incoming fields to Cartesian components, convert to a SHE for a given group using (25), and then evaluate (26). The integration weights required to evaluate (25) are multiplied onto the translation operators, so very little additional work is required to implement this method in an already working MLFMM scheme.

The choice of $W$ in (23) is proposed in [16] to be determined roughly as $W \approx L/2$ for LO discretizations. In [17], it is mentioned that the choice $W = \min(\text{ceil}(L/2), 5)$ is appropriate for HO discretizations. The latter estimate seems far too optimistic and a more theoretical approach is taken here.

Since the spherical harmonics form an orthonormal basis, we can calculate $\mathbf{V}_{j}$ for the largest of the adaptive groups and the highest order basis function $\mathbf{f}_{j}$ in that group. We can then adjust $W$ until $g(W) < 10^{-2\beta}$, where $g(W)$ is defined as

$$g(W) = \max \left\{ \frac{\| \mathbf{V}_{j}^{(c)}(\mathbf{k}) \|^2 d^2 \mathbf{k}}{\sum_{p} \sum_{q} |\mathbf{p}^j_{pq}|^2} \right\} \quad (27)$$

where $(c)$ runs through the components. The $W$ found here is then used in all subsequent calculations.

In practice, however, it is sufficient to investigate the general behavior of $g(W)$ for the scalar case of $V(k) = e^{ik\mathbf{r}}$, letting $|\mathbf{r}| = D/2$ vary. This allows for rough estimates of $W$ to be tabulated in advance using (27), as a function of $D$. It is worth mentioning that for $\beta \leq 2$, this approach reveals that the original $W \approx L/2$ estimate is reasonable, while for better accuracies it is slightly too optimistic. Thus, we use $W = L/2 + \max(\beta - 2, 0)$.

IV. NUMERICAL RESULTS

This section contains three test cases designed to demonstrate the savings achieved by the modified implementation. The first two examples have reference solutions, allowing us to compare the achieved accuracy and computational requirements across varying discretizations. The last example is a demonstration of the capabilities of the implementation on a realistic scatterer. We note that whenever we discuss MLFMM memory, we refer to the memory required to store the entire MLFMM structure but not to solve the scattering problem. Thus, we disregard the storage required for solvers, preconditioners and geometry, since this is not the focus of the present paper, but include everything required to perform a matrix-vector product; from basis function patterns and near-interaction matrix as well as minor temporary data such as interpolation matrices and various bookkeeping.

We apply GMRES and an overlapping near-field preconditioner [33], but stress that the results are independent of the iterative method and preconditioner. The iteration timings include the inner operations of GMRES and the cost to precondition, but these represent less than 0.5% of the times and are thus irrelevant to the conclusions.

As a measure of accuracy, we use the Relative RMS Error, defined as

$$\text{Relative RMS Error} = \sqrt{\frac{\sum_{i=1}^{N_s} |\mathbf{E}_{i,\text{ref}} - \mathbf{E}_{i,\text{calc}}|^2}{\sum_{i=1}^{N_s} |\mathbf{E}_{i,\text{ref}}|^2}} \quad (28)$$

where $\mathbf{E}_{i,\text{ref}}$ and $\mathbf{E}_{i,\text{calc}}$ denote the electric far field at the $i$th sample point from the reference and calculated scattered fields, respectively, and $N_s$ is the number of samples. The computation times shown are measured without parallelization.

A. Sphere

The first example concerns the scattering from a 1-m radius PEC sphere at 8 GHz, illuminated by an $\hat{x}$-polarized plane wave propagating along $+z$. The sphere is discretized using fourth-order curved quadrilaterals. The problem requires between 235 200 and 940 800 unknowns for the first-order basis functions and between 187 500 and 367 500 unknowns for the fifth-order functions. We apply MLFMM with $\beta = 3$ in (13)
to the CFIE with $\epsilon_r = 0.5$, calculate the scattered field in the E-plane, and compare it to the Mie series. The results are shown in Fig. 3 as a function of the mesh size for each order.

The key observation from Fig. 3 is the increasing slope of the curves as the order of the basis functions increases. This corresponds to the theoretical estimate [27] of the discretization error behaving as $O(h^n)$, where $h$ is the mesh spacing and $n$ is the polynomial order. We note that this is to our knowledge the first time such behavior has been demonstrated with MLFMM for hierarchical basis functions and for orders larger than 3. This demonstrates that there is no additional error when using MLFMM with HO rather than LO basis functions.

Fig. 4 demonstrates the required memory, as a function of relative RMS error, for varying polynomial order. While noting the nontrivial memory behavior as opposed to what we see with standard MoM, we also see that polynomial orders above 2 are perfectly competitive with lower orders in terms of memory. The lowest order $n = 1$ leads to a significantly higher memory consumption for a given relative RMS error, while the memory consumption for $n \geq 2$ is roughly independent of the polynomial order, except for high accuracies where $n = 2$ is not competitive. This behavior differs significantly from previously published conclusions [13], [22]–[24], where the conclusion was that $n > 2$ was not worthwhile.

The reduced memory consumption is due to the modifications described in Section III-B. This is clearly demonstrated by comparing the blue and the red set of curves that show the memory used with and without the modifications, respectively. The modifications result in a greater portion of the memory being spent in the group radiation patterns and the near-interaction matrix. The memory for storing group radiation patterns is far more significant for LO discretizations due to the vastly increased number of groups, while the near-interaction matrix is more significant for HO discretizations due to the larger group size at the lowest level. This is also evident in the plot, since the relative reduction in memory is much larger for the higher orders, although there is still a noticeable reduction for first order. We mention that the sidelong of the patches at the coarsest accuracy for order 5 is $1.6\lambda$. Considering the amount of memory used, these are far larger patches than we have seen used successfully in other higher order MLFMM implementations.

We note that the results from this test case, along with results from other experiments not shown here, indicate that the “best” order in terms of memory is very hard to quantify in general, because it depends on the geometry of the problem at hand as well as the desired error. Thus, we caution against recommending a specific order for all problems based on the results shown here, since these are based on a very theoretical case. Indeed, the almost uniformly meshed sphere, with no exterior edges and fixed polynomial order everywhere is not a common problem in practice.

Instead, we emphasize that, contrary to previously reported results, there is no significant penalty in terms of memory when increasing the order of the basis functions, provided that one applies the modifications described in the present paper. Indeed, increasing the order can be very beneficial in terms of memory, particularly when going for solutions with low relative RMS error.

Fig. 5 shows the most important feature of HO MLFMM—a significantly reduced time per iteration as the order is increased. This is the result of HO MLFMM having fewer levels and fewer unknowns, while also representing a larger portion of the interaction in the near-interaction matrix. Furthermore, the two sets of curves (in blue and red) allow us to gauge the time cost of the modifications. Overall, the increase in computational cost is roughly 20%, except for order 4 which fits very poorly with the mesh, resulting in a slightly larger cost for the interpolation involved in adaptive grouping. We note that the SHE could be used to render the interpolation step in adaptive grouping unnecessary, which would reduce the computational cost of adaptive grouping significantly.

The conclusion from Figs. 4 and 5, along with other experiments not shown here, is thus clear: Use as high an order as possible, since this significantly reduces the iteration time, while requiring roughly the same memory as lower orders.
We also bear in mind that we are using CFIE on a PEC scatterer, which doubles the number of basis function patterns and breaks the symmetry in the near-interaction matrix that would come from using EFIE. This is a greater disadvantage for HO than for LO, because a larger proportion of the memory is spent in the near-field matrix and on storing basis function patterns for HO, while LO uses a significant amount of memory on storing the group patterns, which is unaffected by switching between EFIE and CFIE. To illustrate the memory behavior in EFIE, Fig. 6 is the equivalent of Fig. 4 but with the memory used in EFIE. Again, while the “best order” will depend on the scatterer and desired accuracy, it is clear that going beyond second order is indeed beneficial, reducing both memory requirements and computation time.

B. Disk

The second example involves a PEC disk, located at \( z = 0 \) with a radius of 1 m, illuminated by a x-polarized, +z propagating plane wave at a frequency of 30 GHz and solved with MLFMM applied to the EFIE. This rotationally symmetric test case allows the use of the body of revolution method of moments (BoR-MoM) code [34] implemented in GRASP [35] as a very accurate reference solution.

Having chosen the mesh size such that the solution for each polynomial order yields roughly 0.3% relative RMS error, Table I shows the memory costs and time per iteration for each order. We see that it is clearly advantageous going above second order—all the way to order 5, both memory and computation time are reduced compared to both first and second order. We also see that the reduction in number of unknowns as the polynomial order is increased is now more significant than it was for the sphere. Thus, the higher order basis functions are relatively more advantageous than for the first case. This is due to the presence of edges, where the behavior of the true current is hard to model accurately using lower order polynomials due to its singularity.

To illustrate the savings achieved by the adaptive grouping and spherical harmonics techniques, Table II shows the memory for the basis function patterns and group patterns without these techniques and with only the adaptive grouping. The numbers without SHE take into account the symmetry of the basis function patterns. We see that both techniques are instrumental, particularly for HO implementations. The saving achieved with adaptive grouping depends significantly upon the initial grouping achieved by the Octree, but even for the lower orders, the saving is significant. For SHE, the savings are much larger; roughly a factor of 4–5. We note that both of these techniques require additional group storage at the lowest level, but we see that particularly for HO implementations, this is negligible.

Regarding the computation time, we see that the cost of the adaptive technique is negligible for HO, since it trades the cost of an extra level of interpolation/interpolation with a faster aggregation/disaggregation step. For the SHE, there is a roughly

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20</td>
<td>2927300</td>
<td>0.37</td>
<td>7.22</td>
<td>1.42</td>
<td>1.12</td>
<td>10.42</td>
<td>23.64</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>0.49</td>
<td>1889406</td>
<td>0.15</td>
<td>2.94</td>
<td>1.45</td>
<td>1.09</td>
<td>6.73</td>
<td>14.49</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0.97</td>
<td>1077948</td>
<td>0.09</td>
<td>3.62</td>
<td>1.33</td>
<td>1.08</td>
<td>4.76</td>
<td>12.97</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>1.52</td>
<td>784576</td>
<td>0.05</td>
<td>1.96</td>
<td>1.50</td>
<td>1.08</td>
<td>4.63</td>
<td>11.32</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>2.38</td>
<td>494340</td>
<td>0.04</td>
<td>2.99</td>
<td>1.55</td>
<td>1.08</td>
<td>3.49</td>
<td>11.21</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig. 5. Time per iteration, normalized to 1 for the fastest run, for the solutions shown in Fig. 4. We see a direct connection between increasing the order and reducing the time per iteration. We also see that the cost of adding the modifications, as compared to the standard MLFMM implementation, is noticeable, representing a roughly 20% increase. The legend is as in Fig. 4.

Fig. 6. MLFMM memory for varying relative RMS error and polynomial order for \( \beta = 3 \) for EFIE.

TABLE I
MEMORY USE FOR TEST CASE B. THE ACCURACY IS \( \beta = 3 \) IN (13), AND WE HAVE USED ONE BUFFER BOX EXCEPT FOR ORDER 1, WHERE WE NEEDED TO USE TWO BOXES TO GET SUFFICIENT ACCURACY. MLFMM MEMORY INCLUDES THE TEMPORARY DATA AND THUS EXCEEDS THE SUM OF THE MAIN CONTRIBUTORS LISTED HERE.
TABLE II
MEMORY USE FOR VARIOUS STRATEGIES—THE COST FOR ADAPTIVE, SHE STRATEGY CAN BE SEEN IN TABLE I. TIMES ARE NORMALIZED

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.56</td>
<td>7.69</td>
<td>3.56</td>
<td>4.50</td>
<td>9.94</td>
</tr>
<tr>
<td>2</td>
<td>13.68</td>
<td>5.29</td>
<td>1.65</td>
<td>4.99</td>
<td>6.12</td>
</tr>
<tr>
<td>3</td>
<td>16.78</td>
<td>3.88</td>
<td>1.31</td>
<td>5.17</td>
<td>4.31</td>
</tr>
<tr>
<td>4</td>
<td>11.45</td>
<td>3.87</td>
<td>1.22</td>
<td>6.28</td>
<td>4.18</td>
</tr>
<tr>
<td>5</td>
<td>20.35</td>
<td>2.89</td>
<td>1.01</td>
<td>6.95</td>
<td>3.12</td>
</tr>
</tbody>
</table>

5% time increase for HO, since the cost of (25) and conversion between components is traded for a much faster aggregation and disaggregation. All-in-all, we believe that these results demonstrate that any modern HO implementation of MLFMM should apply both adaptive grouping and SHE, since the massive memory reduction comes at a very minor computational cost.

C. Planck Mock-Up

As a final example, we compute the radiation pattern of a simple mock-up of the Planck space telescope [36]. The main part of the telescope features two large reflectors in a dual reflector configuration, inside a shielding structure. The structure is shown in Fig. 7. The ellipsoidal subreflector, at the bottom of the figure, is fed by a horn located at the bottom of the shielding structure.

The resulting mesh at 30 GHz is quite irregular, featuring 126,596 patches with sidelengths between 0.34λ and 2λ and a surface area of 170·10^3λ^2. This results in 4,583,755 unknowns, with up to ninth-order polynomials used on the largest patches. A typical RWG discretization would require roughly 20 million unknowns.

The required MLFMM memory for HO MLFMM is 39.5 GB, and the problem converges in 148 iterations using a preconditioner requiring an additional 9 GB of memory. The scattered field is shown in Fig. 8.

V. CONCLUSION

The results in this paper clearly show that the advantages of higher order basis functions and MLFMM can be combined, provided some additional modifications to the standard MLFMM setup are implemented. We stress that all test cases have employed Legendre basis functions—had we used RWG for the LO solutions, the comparison would have been even more beneficial to HO MLFMM, since RWG basis functions require a much greater basis function density [26]. We further note that while the exact memory requirements obviously depend on the scatterer, we have clearly demonstrated that HO MLFMM is advantageous both in terms of memory and particularly in terms of speed. Thus, although the optimal order cannot be generally determined since it depends strongly on the required accuracy and the geometry of the scatterer, we have demonstrated that using higher order MLFMM results in significantly better performance than LO MLFMM.

ACKNOWLEDGMENT

The authors would like to thank Dr. S. Järvenpää, Department of Radio Science and Engineering at Aalto University, Finland, and Prof. T. Eibert, Lehrstuhl für Hochfrequenztechnik, Technische Universität München, for helpful advice.

REFERENCES


lationary vector bases for computational electromagnetics,” IEEE Trans.


order hierarchical Legendre basis functions for electromagnetic

method for analysis of metallic antennas and scatterers,” in IEEE Prog. H Mic-

equation in three dimensions,” Yale Univ., New Haven, CT, USA, 1992,

method for the wave equation: A Laplacian prescription,” IEEE Anten-


solving combined field integral equations of electromagnetic

algorithm for electromagnetic scattering by large complex objects,”
1997.

“A higher order parallelized multilevel fast multipole algorithm for

novel implementation of multilevel fast multipole algorithm for higher

higher-order multilevel fast multipole algorithm for analysis of 3-D

with spherical harmonics expansion of currents’ expansion order in MLFMM
algorithm for electromagnetic scattering,” in Proc. IEEE Antennas Propag.

[17] Ismailullah and T. F. Ebert, “Surface integral equation solutions by
hierarchical vector basis functions and spherical harmonics based

[18] Ismailullah and T. F. Ebert, “Higher order surface integral equation
solutions employing spherical harmonics based multilevel fast multi-

[19] Ismailullah, “Analysis of space-borne antennas by higher-order method
of moments and inverse equivalent current methods,” Ph.D. disserta-
tion, Technical Univ. of Munich, Munich, Germany, Apr. 2010.

level fast multipole algorithm for electromagnetic scattering analysis,”

[21] X. Q. Sheng, M. L. Yang, X. M. Pan, and C. Q. Deng, “Parallel higher-
order FE-BI-MLFMA for 3D scattering,” in Proc. Int. Workshop Anten-

to higher-order hierarchical Legendre basis functions in electromagnetic
scattering,” IET Microw., Antennas Propag., vol. 2, no. 1, pp. 6–9,
2008.

[23] B. M. Kolundzija and D. Sumic, “Multilevel fast multipole method
for higher order basis functions implemented in WIPLD-Pro,” in Proc.

of currents’ expansion order in MLFMM algorithm for electromagnetic
scattering,” in Proc. IEEE Antennas Propag. Soc. Int. Symp. (AP-
SURSI), 2009, pp. 1–4.


it that big?,” IEEE Antennas Propag. Mag., vol. 45, no. 2, pp. 43–58,


[28] E. Jørgensen, J. Volakis, P. Meincke, and O. Breinbjerg, “Divergence-
conforming higher order hierarchical basis functions and precondition-
ting strategies for boundary integral operators,” in Proc. 6th Int.
Workshop Finite Elements in Microw. Eng., Chios, Greece, May 2002,
p. 63.

1381–1384.

conforming higher order hierarchical basis functions and precondition-
ting strategies for boundary integral operators,” in Proc. 6th Int.
Workshop Finite Elements in Microw. Eng., Chios, Greece, May 2002,
p. 63.

[31] J. M. Song, Y. Su, and X. Guan, “Fast and robust implementation of
multilevel fast multipole method for higher order Galerkin’s method,”
2006.

[32] P. Meincke (S’93–M’96) was born in Roskilde, Den-
mark, on November 4, 1986. He received the B.Sc.
and M.Sc. degrees in mathematical engineering
from the Technical University of Denmark (DTU),
Lyngby, Denmark, in 2009 and 2011, respectively,
where he is currently working towards the Ph.D.
degree.

[33] Peter Meincke’s research interests include fast methods and
integration equation methods, particularly higher order
methods.

[34] Oscar Borries (S’83) was born in Gentofte, Den-
mark, on November 4, 1986. He received the B.Sc.
and Ph.D. degrees in electrical engineering
from the Technical University of Denmark (DTU),
Lyngby, Denmark, in 1993 and 1996, respectively.

[35] In spring and summer 1995, he was a Visiting Re-
search Scientist at the Electromagnetics Directorate
of Rome Laboratory, Hanscom Air Force Base, MA.
In 1997, he was with a Danish cellular phone company,
working on theoretical aspects of radio-wave
propagation. In spring and summer of 1998, he was
visiting the Center for Electromagnetics Research, Northeastern University,
Boston, MA, USA, while holding a Postdoctoral position from DTU. In 1999,
he became a staff member of the Department of Electromagnetic Systems,
DTU. He was an Associate Professor with Ørsted-DTU, Electromagnetic
Systems, DTU, where his teaching and research interests included electromag-
netic theory, inverse problems, high-frequency and time-domain scattering,
target identification, and microwave imaging. Since 2008, he has been with TICRA,
Copenhagen, Denmark, where he is currently developing software for reflector
antenna analysis and antenna diagnostics.

[36] Dr. M. D. F. Harrison won the first prize award in the 1996 IEEE Antennas
and Propagation Society Student Paper Contest in Baltimore, MD, for his paper
on uniform physical theory of diffraction equivalent edge currents and received
the R.W. P. King Paper Award in 2000 for his paper “Time-domain version of
the physical theory of diffraction” published in the IEEE TRANSACTIONS ON
Erik Jørgensen (S’98–M’03) was born in 1974 in Denmark. He received the M.Sc. and Ph.D. degrees in electrical engineering from the Technical University of Denmark (DTU), Lyngby, Denmark, in 2000 and 2003, respectively. In fall 1998, he spent five months at the University of Siena, Italy. In fall 2001, he was a Visiting Scholar at the Radiation Laboratory, University of Michigan, Ann Arbor, MI, USA. In 2003, he joined the Danish company TICRA, Copenhagen, where he is now leading the development of commercial software for reflector antenna analysis and antenna diagnostics. His research interests include computational electromagnetics and high-frequency techniques.

Per Christian Hansen received the M.Sc. degree in electrical engineering in 1982, the Ph.D. degree in numerical analysis in 1985, and the Dr. Techn. degree in scientific computing in 1996, all from the Technical University of Denmark (DTU), Lyngby, Denmark. He was a Research Associate at Copenhagen University from 1985 to 1988, and senior consultant at UNIC from 1988 to 1996. Since 1996, he has been a Professor of scientific computing at DTU Compute, the Technical University of Denmark. He primarily works with numerical regularization algorithms for inverse problems, and he has published 80+ papers in leading journals as well as three books on numerical methods for inverse problems. He has also developed several related software packages.
A.2 Paper II

**Adaptive Grouping for the Higher-Order Multi-Level Fast Multipole Method**

Oscar Borries, Erik Jørgensen, Peter Meincke, and Per Christian Hansen

*Status*

Published: October 2014

**Bibliography**

Abstract—An alternative parameter-free adaptive approach for the grouping of the basis function patterns in the Multi-Level Fast Multipole Method is presented, yielding significant memory savings compared to the traditional Octree grouping for most discretizations, particularly when using Higher-Order basis functions. Results from both a uniformly and non-uniformly meshed scatterer are presented, showing how the technique is worthwhile even for regular meshes, and demonstrating that there is no loss of accuracy in spite of the large reduction in memory requirements and the relatively low computational cost.

Index Terms—MLFMM, Higher-Order Discretization, Irregular meshes

I. INTRODUCTION

When solving large-scale electromagnetic scattering problems, where the unknown is the surface current density induced by an incident electromagnetic field on a scatterer, the Multi-Level Fast Multipole Method (MLFMM) [1]–[4] is one of the most powerful methods for speeding up the necessary matrix-vector products involved in an iterative solution.

The MLFMM is a hierarchical algorithm which achieves an asymptotic complexity of $O(N \log N)$, $N$ being the number of unknowns, by computing interactions between groups of basis functions rather than individual basis functions. The multi-level aspect comes from using a hierarchical grouping to allow interactions over increasing distances to be done by considering increasingly larger groups.

The first step of the MLFMM is the application of a grouping algorithm. The grouping effectively determines the region of validity of the Green’s function expansion underlying the MLFMM. Furthermore, the grouping dictates the number of terms needed to represent the functions involved in MLFMM, often called the bandwidth, and therefore also the required number of samples of those functions on the unit sphere, called the sampling rate. In most implementations [2], [4], this grouping is done by using the Octree algorithm [5]. This is a fast, easily implemented and conceptually simple algorithm, designed in computer graphics to adapt very well to any geometrical shape. In the context of the MLFMM, it also has the extremely important feature of allowing reuse of some of the quantities involved [6].

However, as the discretizations become more irregular, which is often the case for Higher-Order discretizations applied to realistic problems, the Octree grouping at the finest level results in excessive sampling rates. This is due to the Octree only considering the center of the geometrical elements, known as patches, as well as the size of the largest patch, rather than taking into account the size and shape of the individual patch. To reduce the memory consumption significantly, particularly for very irregular meshes or for meshes with large patches, we suggest in this paper a grouping method that allows the grouping on the finest level to become completely adaptive to the shape of the patches. The method results in a modest increase in the computational cost of a matrix-vector product, which will be discussed in Section IV. While this method is conceptually simple, it has not to our knowledge been published previously.

The time factor $e^{j\omega t}$, where $\omega$ is the angular frequency, is assumed and suppressed throughout.

II. MULTI-LEVEL FAST MULTipoLE METHOD

In the present paper, the MLFMM is used when solving the Electric Field Integral Equation (EFIE) [7]

$$\hat{n} \times E^i = \mathcal{L} J_S,$$

(1)

using an iterative solver. In (1), $\hat{n}$ is a unit vector normal to the scatterer $S$, $E^i$ is the incident electric field, and $J_S$ is the unknown surface current density. Further, $\mathcal{L}$ is the integral operator

$$\mathcal{L} J_S = \hat{n} \times j \omega \mu \left[ \int_S J_S(r') G(r, r') d^2 r' + \frac{1}{k^2} \int_S \nabla S \cdot J_S(r') \nabla G(r, r') d^2 r' \right],$$

(2)

where $\mu$ is the free-space permeability, $k = 2\pi/\lambda$, with $\lambda$ being the free-space wavelength. $G(r, r') = e^{-jk|r-r'|}/4\pi|r-r'|$ is the free-space Green’s function and $r$, $r'$ denote observation and integration points, respectively. For some scenarios, it is more useful to apply the Combined Field Integral Equation (CFIE) [7],

$$\left[ \alpha \mathcal{L} + (1 - \alpha) \eta \left( \frac{1}{2} \mathcal{I} + \mathcal{K} \right) \right] J_S = \alpha \hat{n} \times E^i + (1 - \alpha) \eta \hat{n} \times H^i.$$

(3)
where \( \mathcal{I} \) is the identity operator, \( H \) is the incident magnetic field, \( \eta \) is the free-space impedance, \( \alpha \in [0,1] \) is a weighting factor, and \( \mathcal{K} \) is the operator

\[
\mathcal{K}J_S = \hat{n} \times \int_{S} J_S(r') \times \nabla G(r,r') \, d^2r',
\]

where \( \hat{n} \) denotes the Cauchy principal value.

The central part of the MLFMM is Rokhlin’s translator [1]

\[
T_L(k, \hat{k}, x) = \sum_{l=0}^{L} (-j)^{l} (2l+1) h_{l}^{(2)}(k \|x\|_{2}) P_l(\hat{k} \cdot \hat{x}),
\]

where \( \hat{k} \) is the unit wave vector, \( x \) is the vector between two group centers, \( \hat{x} = x/\|x\|_{2} \), \( h_{l}^{(2)} \) is the spherical Hankel function of second kind and order \( l \), and \( P_l \) is the Legendre polynomial of degree \( l \). It is important to note that the translator does not depend on the absolute position of the groups, but only on the vector \( x \) between their centers. Thus, the translator can be reused for groups with the same \( x \), a key factor in keeping memory consumption low. Typically, the number of terms \( L+1 \) in the translator is determined from the Excess Bandwidth Formula [8]

\[
L = kD + 1.8 \beta^{2/3}(kD)^{1/3},
\]

where \( D = \sqrt{3a} \) is the diameter of the group, \( a \) is the sidelength of the group, and \( 10^{-\beta} \) is the desired relative error.

To discretize the problem, we begin by representing the surface of the scatterer \( S \) by geometric elements known as patches. Then, (1) is discretized using a Galerkin method to yield a linear system \( \tilde{Z}I = B \). Here, \( I \) is a vector containing coefficients to the basis functions expressing the surface current density, \( \tilde{Z} \) is a matrix containing as its \((i,j)\)'th component the mutual impedance between basis functions \( f_i \) and \( f_j \), and \( B \) is a vector, representing the incident field as tested by the basis functions [9]. We can consider MLFMM as a method for splitting the matrix \( \tilde{Z} \) into two parts

\[
\tilde{Z} = \tilde{Z}_{near} + \tilde{Z}_{far},
\]

where the near-matrix \( \tilde{Z}_{near} \) is stored as a sparse matrix, while \( \tilde{Z}_{far} \) is not stored directly, but instead the elements required to multiply \( \tilde{Z}_{far} \) with an excitation vector \( f \) are stored.

Introducing the basis function patterns as

\[
V_{jm}(k, \hat{k}) = \int_{S} f_j(r) \cdot \left( \hat{l} - \hat{k} \right) e^{-j\hat{k} \cdot r} \, d^2r,
\]

where \( \hat{l} \) is the identity matrix, and utilizing (5), we can express the matrix elements resulting from the EFIE (1) as

\[
\tilde{Z}_{far(j,i)} = \iint V_{jm}^{*}(\hat{k}, \hat{k}') \cdot \left( T_L(k, \hat{k}', r_{mm'}) V_{im'}(k, \hat{k}') \right) \, d^2\hat{k},
\]

which is then discretized to

\[
\tilde{Z}_{far(j,i)} = \kappa \sum_{p=1}^{K} w_p V_{jm}^{*}(\hat{k}, \hat{k}_p) \cdot \left( T_L(k, \hat{k}_p, r_{mm'}) V_{im'}(k, \hat{k}_p) \right),
\]

where \( \kappa \) is a constant depending on the units of the impedance matrix \( \tilde{Z} \), \( K = 2(L + 1)^2 \) is the number of sample points on the unit sphere [10], and \( w_p \) are the integration weights.

We assume that \( f_i \) belongs to group \( nn' \) and \( f_j \) belongs to group \( mm' \), where \( r_{mm'} = r_m - r_{m'} \) where \( r_m \) denotes the center of group \( m \), and we further assume that \( |r_{mm'}| > D \). If \( |r_{mm'}| \leq D \), the element \( \tilde{Z}_{j,i} \) must be computed directly and stored in \( \tilde{Z}_{near} \). For the CFIE (3), the expression for the matrix elements corresponding to (10) are slightly more complicated [4].

The key issue from (8) and (10) is that the number of sample points for the basis function patterns \( V_{jm} \) is the same as that required for the translator, even though the bandwidth of \( V_{jm} \) is lower [10]. Note that the bandwidth of \( V_{jm} \) is directly related to the largest value of \( |r_{mm'}| \) attained on the domain of \( f_j \), due to the term \( e^{-j\hat{k} \cdot r_{mm'}} \) in (8).

III. GROUPING

An Octree [5] is used here as a hierarchical data structure, allowing a geometrical object to be spatially partitioned in a fast and simple manner, thereby providing a grouping of basis functions that are spatially near each other. The Octree grouping is done by creating an original bounding box for \( S \), hereafter termed level 1 or the coarsest level. Finer levels are then created by partitioning boxes such that the diameter of the boxes at level \( q \) is \( D_q = D_{q-1}/2 \). This results in eight potential boxes per partitioning, of which only those that contain the center of a patch are kept while the rest are pruned. This results in a very fast partitioning of the patches into clusters. The partitioning stops when

\[
D_q/2 < l,
\]

where \( l \) is the largest sidelength of any patch in the mesh surface, yielding \( q \) levels in the Octree.

However, as Figure 1 illustrates, this will occasionally result in unnecessarily large groups at the finest level, simply because the Octree scheme is not able to adapt to the patches. Effectively, the center point for \( V_{jm} \), \( r_{mm} \), is positioned such that the sample rate will be far too large. The memory cost for this can be very significant. Further, since the sampling rate for all boxes at each level is the same, and since the finest level box size is determined by the largest patch length in the mesh, scenarios with non-uniform patch sizes will result in far too large sampling rates for the groups with smaller patches.

Therefore, we propose to tabulate each \( V_{jm} \) based on a separate adaptive grouping. In this approach, each patch is associated with its own group, with the center point \( r_{mm} \) chosen to minimize the term \( |r_{nm} - r_{m} - r_{n} - r_{m'}| \) (8). \( D \) is then found by the maximum attained value of \( 2|r_{nm} - r_{m'}| \), and the sampling rate is determined from (6). In this way, the sampling rate is optimized for each patch, and the basis function patterns are stored at the coarsest possible sampling density.

With this adaptive grouping, (10) is changed to

\[
\tilde{Z}_{j,i} = \kappa \sum_{p=1}^{K} w_p V_{jm}^{*}(\hat{k}, \hat{k}_p) \cdot \left( W^T e^{-j\hat{k} \cdot r_{mm'}} \right) \left( T_L(k, \hat{k}_p, r_{mm'}) \right) \left( T_L(k, \hat{k}_p, r_{mm'}) \right) \left( W V_{im'}(k, \hat{k}_p) \right),
\]

where the notation \( \hat{m} \) refers to a group at the adaptive level, and group \( m \) is the group at the finest level of the Octree containing \( m \). \( W \) is an interpolation matrix, designed to
increase the sampling density of the basis function patterns to that of the translator, such that $W \in \mathbb{R}^{N \times N}$, where $N$ is the number of plane wave directions used in group $\bar{m}$. Comparing (10) and (12), we can express adaptive grouping as an expansion of the basis function pattern $V_{\bar{m}'}$ as

$$V_{\bar{m}'} = e^{-jkkr_{\bar{m}'}}W_{\bar{m}'},$$

which, aside from the controllable interpolation error, is exact.

We stress that there are no translations done on the adaptive grouping level, and therefore the near-matrix $Z_{\text{near}}$ is not based on the adaptive level. Basing $Z_{\text{near}}$ on the adaptive level would yield a smaller matrix, but this would imply that separate translators would have to be computed for each adaptive group interaction due to the adaptive group center. Without this possibility of reusing translators, which is perhaps the greatest strength of the Octree used with the MLFMM, the memory requirements for the translators, as well as the additional work in translation on the adaptive level, would impair the performance. Another advantage in using the larger Octree groups for the translation is that (5) is more numerically impairing for each group. Therefore, our focus was not on irregular meshes, so neither the approach nor the results can be compared with the present paper.

In a practical implementation, if each adaptive group has a unique diameter, and thus a unique sampling rate, this would require a significant amount of auxiliary data, in particular the interpolation matrix $W$ for each group. Therefore, our implementation uses a number of specifically allowed sizes, and categorizes each adaptive group into those. Note that only the sampling rate is affected by this categorization, not the center of the adaptive group. For fairly uniform scatterers, only 2 or 3 possible sizes are needed, while strongly non-uniform scatterers need a few more. As an estimate for the number of allowed group sizes $N_{\bar{m}}$, we use

$$N_{\bar{m}} = \left[ \frac{\max_{\bar{m}} D_{\bar{m}}}{\min_{\bar{m}} D_{\bar{m}}} \right],$$

where $\bar{m}$ runs through the adaptive groups, and $D_{\bar{m}}$ is the diameter of group $\bar{m}$. We stress that the setup time for the adaptive grouping is insignificant, countable in milliseconds.

The adaptive grouping does not affect the number of non-zeros in the near-matrix $Z_{\text{near}}$. To reduce the memory required for $Z_{\text{near}}$, a locally extended Octree grouping can be implemented. This means that $l$ in (11) is modified to equal the largest sidelength of any quad in the group under consideration only, rather than considering all quads in the mesh. Thus, if the mesh is locally very fine, the Octree will locally have additional levels compared to regions with a coarser mesh. Figure 2 provides a small 2-D illustration of a locally extended Octree grouping.

A locally extended Octree was discussed in [12], but was used as a way to reduce the size of the basis function patterns only, not to reduce the memory required for $Z_{\text{near}}$, since translations on the extended levels was not performed. Further, the focus was not on irregular meshes, so neither the approach nor the results can be compared with the present paper.

IV. Numerical Results

The first example involves a perfectly electrically conducting (PEC) sphere, designed to illustrate that even for uniformly meshed scatterers, it is beneficial to apply the adaptive grouping, particularly if the group size on the finest level of the Octree is much larger than the patch size. It further illustrates that there is no loss of accuracy from the adaptive grouping.

The second example concerns an irregularly meshed circular PEC plate with several small holes, designed to represent mounting holes. While we stress that these holes are so small that they should not be considered in the electromagnetic representation of the problem, it is fairly common in structures based on CAD to have such features.

The results are based on the implementation detailed in [13], but the implementation does not utilize the storage of basis functions using Spherical Harmonics Expansions (SHE) [14], since we want to isolate the effects of using adaptive grouping compared to standard Octree grouping at the finest level. However, these two techniques (adaptive grouping and SHE) can easily be combined, and their combination allows use of the SHE to reduce the computational cost of adaptive grouping. We use Lagrange interpolators to step between the sampling rates of the levels. When discussing total memory, we include the memory needed to store the entire MLFMM structure, including near-matrix, basis function patterns etc., as well as minor temporary data, including that needed for interpolation matrices in the adaptive grouping. Throughout, the accuracy setting $\beta = 3$ is used in (6). The error is
computed as the relative RMS

\[
\text{Relative RMS} = \sqrt{\frac{\sum_{i=1}^{N_s} (|E_{i,\text{ref}} - E_{i,\text{cal}}|^2)}{\sum_{i=1}^{N_s} |E_{i,\text{ref}}|^2}}, \tag{15}
\]

where \(E_{i,\text{ref}}\) and \(E_{i,\text{cal}}\) denote the reference and calculated electric fields at the \(i\)th sample point, respectively, and \(N_s\) is the number of samples.

**A. Sphere**

We consider an \(\hat{x}\)-polarized plane wave at 10 GHz, propagating in the \(+z\) direction, incident on a 1 m PEC sphere centered at the origin of the coordinate system. Using \(4^{th}\)-order basis functions, we vary the sidelength of the patches between \(0.9\lambda - 1.3\lambda\) for the fairly uniformly meshed structure, and use the CFIE (3) with \(\alpha = 0.5\) to solve the problem, requiring between 322752 and 668352 unknowns. Figure 3 shows the memory consumption as function of the RMS, illustrating the unfortunate property of the Octree grouping to have a very complicated dependence between memory and RMS. In particular, for the scenario \(D_q/2 < l \ll D_{\lambda 1}\), where \(l\) is the largest sidelength in the mesh and \(q\) is the finest level, the standard Octree grouping results in extreme memory consumption for the basis function patterns, visible as a peak Figure 3. The adaptive grouping does not have this problem for the basis function patterns, though we can still see the effects through the memory used for the near-matrix. We further see that including the locally extended Octree only has an effect at a single point, further smoothing out the "hump" when the discretization is getting close to allowing an additional level in the Octree.

Figure 4 illustrates the time spent per matrix-vector product. We see a modest increase from using the adaptive grouping due to the extra interpolation step. However, particularly for \(\text{Higher-Order discretizations}, where there are relatively few groups at the finest level, and thus fewer interpolation steps, this will be negligible compared to the significant reduction in memory. Thus, we can conclude that even for uniformly meshed scatterers, there is a very significant potential memory reduction to be achieved by using adaptive grouping at the cost of a modest increase in computation time.

**B. Circular plate with holes**

We now consider an \(\hat{x}\)-polarized plane wave at 300 MHz, propagating along \(-z\), incident on a 36 m diameter circular plate, centered at the origin and positioned in the \(xy\)-plane. The plate has 9 square mounting holes, each with a sidelength of 0.1\(\lambda\), placed in a cross around the center. The meshing of this surface with quadrilaterals using sidelengths between 0.1\(\lambda\) and 1.28\(\lambda\) is shown in Figure 5.

The discretization yields \(N = 51491\) unknowns, with polynomials up to \(7^{th}\) order being used on the largest patches. The default Octree grouping yields a fairly poor grouping, with a sidelength at the finest level of 2.25\(\lambda\). This results in 3.79 GB of total memory. Using the adaptive grouping and locally extended Octree, the memory is reduced to less than a sixth, 623 MB, while the time per matrix-vector product is increased by roughly 8\%. The memory for the basis function patterns alone is reduced from 3.1 GB to 426 MB, a factor of 7.5. We note that with adaptive grouping, but without the locally extended Octree, the total memory required would have been 1.09 GB, so for this strongly non-uniform mesh, the locally extended Octree is effective.

Figure 6 shows the scattered fields from each of the two techniques, further demonstrating that the proposed adaptive approach, with both adaptive grouping and a locally extended Octree, yields the same result as the standard Octree grouping.
Fig. 5. Circular plate with nine 0.1λ × 0.1λ mounting holes.

Fig. 6. The scattered far-field for a circular plate with holes, for φ = 0. Both the results from using Octree grouping and the adaptive grouping are shown, demonstrating that there is no loss of accuracy.

V. CONCLUSION

Our results demonstrate that the proposed adaptive grouping approach should be included in modern implementations of MLFMM, particularly when using a Higher-Order discretization with larger patch sidelengths. Further, for strongly non-uniform meshes, a locally extended Octree should also be implemented. For a modest increase in computation time, the reduction in memory obtained with these methods is significant and the implementation is simple.

REFERENCES


A.3 Paper III

**The Gaussian Translation Operator for Scattering Problems using MLFMM**

Oscar Borries, Erik Jørgensen, Peter Meincke, and Per Christian Hansen

Bibliography

The Gaussian Translation Operator for Scattering Problems using MLFMM

Oscar Borries, Student Member, IEEE, Erik Jørgensen, Member, IEEE, Peter Meincke, Member, IEEE, and Per Christian Hansen

Abstract—The use of a directive translation operator in the Multi-Level Fast Multipole Method (MLFMM), applied to general scattering problems, is presented. Applying a translation operator is a significant time-consumer in MLFMM, so a directive operator allows for a significant reduction in time per matrix-vector product. Recent research has described an operator based on Gaussian Beams and demonstrated that for theoretical cases, it is efficient and error controllable. In the present paper, several modifications are presented that allow for automatic control of the parameters involved, even for arbitrary scatterers. Computational results are also shown.

Index Terms—MLFMM, Gaussian Beams, Translation Operator, Computational Electromagnetics

I. INTRODUCTION

The Multi-Level Fast Multipole Method (MLFMM) [1]–[3] is a popular method for reducing the memory and computational complexity of electromagnetic scattering problems. When combined with a Galerkin discretization, MLFMM achieves reduced computational complexity by grouping the basis functions hierarchically using an Octree and letting larger groups interact over greater distances.

The MLFMM represents the matrix vector product $\mathbf{Z} \mathbf{T}$ as

$$\mathbf{Z} \mathbf{T} \simeq \mathbf{Z}_{\text{near}} \mathbf{T} + \mathbf{F}(\mathbf{T})$$

where $\mathbf{Z}_{\text{near}}$ is the near-matrix, containing the interactions between basis functions that are too closely spaced to apply MLFMM. The elements in $\mathbf{Z}_{\text{near}}$ are computed as in the normal Galerkin approach. $\mathbf{F}$ denotes the operation performed by applying MLFMM.

The interaction between two well-separated basis functions $f_j$, $f_i$, belonging to groups $m$ and $m'$ respectively, can be computed by

$$Z_{j,i} \simeq \kappa \iint R_{jm}(k) \cdot (T_L(k, r_{mm'})V_{mm'}(k)) \, d^2 \hat{k},$$

with $r_{mm'} = r_m - r_{m'}$, where $r_m$ is the center of group $m$, and $\kappa$ is a constant. For the Electric Field Integral Equation (EFIE) $\kappa = i \frac{\eta}{4\pi}$, where $\eta$ is the free-space impedance. Further, for EFIE, the basis function signature $R_{jm}(k) = V_{jm}^*(k)$, where * denotes complex conjugation, and

$$V_{jm}(k) = \int_{r^2} f_j(r) \cdot \hat{r} - \hat{k} e^{ik \cdot (r_m - r)} \, d^2 r,$$

while Rokhlin’s translation function $T_L$ [1] is defined as

$$T_L(k, r_{mm'}) = \sum_{l=0}^{L} l^2 (2l+1) h_l^L \bigl( k | r_{mm'} \bigr) P_l(k \cdot r_{mm'}).$$

Herein, $\hat{k}$ is the unit wave vector, $r_{mm'} = r_{m'} / |r_{mm'}|$, $h_l^L$ is the spherical Hankel function of the first kind and order $l$, and $P_l$ is the Legendre polynomial of order $l$. The number of distinct plane wave directions required to integrate (2) is $K_L = 2(L + 1)^2$ [4], and $L$ is typically chosen by applying the Excess Bandwidth Formula (EBF) [5]

$$L = kD + 1.8\beta^{2/3} (\kappa D)^{1/3}$$

where $D$ is the group diameter and the desired relative error is $10^{-\beta}$.

For each matrix vector product, the translation operator $T_L(k, r_{mm'})$ is performed on every pair of interacting groups. Thus, applying the translation operator is a significant contribution to the overall matrix vector multiplication time. Therefore, there is interest in deriving a directive translation operator that discards some of the plane-wave directions. One approach applies filtering [6], while another involves a contour integral formulation of the operator [7]. However, neither of these approaches offer any error control, and both are cumbersome to implement, requiring substantial changes to an existing code.

Recently, a directive translation operator was derived [8]–[10] by applying the theory of Gaussian beams. Compared to the directive operators previously mentioned, the Gaussian operator has several advantages:

- It is based on an exact identity, allowing for full error control.
- It is fairly easy to implement, requiring only modifications to the computation and application of the translator. However, in the previous works on the Gaussian translation operator, no procedure was given to automatically set the parameters involved. Furthermore, no analysis was done for scattering problems, which, due to the close proximity and smaller size of source and receiver groups, are much less amicable to directive translation operators than the cases discussed in the previous literature.

We discuss in detail how to implement the Gaussian translation operator for arbitrary scattering geometries, including approaches for automatically choosing the parameters involved.
To demonstrate, we consider the scattering from a sphere, which allows us to show that the increased directionality does not reduce the accuracy of the solution.

II. GAUSSIAN TRANSLATION OPERATOR

Adding an imaginary vector to the source point and subtracting this same imaginary vector from the origin of the source coordinate system, we see that the real displacement between source and receiver becomes

\[ \mathbf{r}_1 + \mathbf{r}_{mm'} + \mathbf{r}_2 = (\mathbf{r}_1 + \mathbf{r}_2 - i \Delta \hat{\mathbf{r}}_{mm'}) + \hat{\mathbf{r}}_{mm'} (|\mathbf{r}_{mm'}| + i \Delta), \]

where \( \mathbf{r}_1 = \mathbf{r}_m - \mathbf{r} \) and \( \mathbf{r}_2 = \mathbf{r}' - \mathbf{r}_{mm'} \). With this, we can express the Gaussian translation operator [9] for the MLFMM as

\[ T_N(\mathbf{k}, \mathbf{r}_{mm'}, \Delta) = e^{i \mathbf{k} \Delta (\mathbf{r}_{mm'} - \mathbf{1})} \sum_{n=0}^{N} i^n (2n + 1) \tilde{h}_n^{(1)}(k (|\mathbf{r}_{mm'}| + i \Delta)) P_n (\mathbf{k} \cdot \hat{\mathbf{r}}_{mm'}), \]

with \( \tilde{h}_n^{(1)}(x) = h_n^{(1)}(x) e^{i m \pi} \) being the normalized spherical Hankel function, and \( \Delta \geq 0 \) is a beam parameter. When \( \Delta = 0 \), \( T_N = T_L \). The translator \( T_N \) is based on a point source residing in the complex domain—such a point source has a far-field pattern which locally has Gaussian behaviour, hence the name.

In [9], the Gaussian Gegenbauer addition theorem is derived:

\[ G(\mathbf{r}_e, \mathbf{r}_{mm'}) = \sum_{n=0}^{N} U_n (\mathbf{k}, \mathbf{r}_{mm'}, \mathbf{r}_e, \Delta) \]

such that \( G(\mathbf{r}_e, \mathbf{r}_{mm'}) = e^{-i \mathbf{k} \cdot \mathbf{r}_{mm'}} \) and

\[ U_n (\mathbf{k}, \mathbf{r}_{mm'}, \mathbf{r}_e, \Delta) = i k (-1)^n (2n + 1) \tilde{h}_n^{(1)}(k (|\mathbf{r}_{mm'}| + i \Delta)) \cdot j_n (k \sqrt{\mathbf{b} \cdot \mathbf{b}}) P_n \left( \frac{\mathbf{r}_{mm'} \cdot \mathbf{b}}{\sqrt{\mathbf{b} \cdot \mathbf{b}}} \right) \]

where \( \mathbf{b} = \mathbf{r}_e - i \Delta \hat{\mathbf{r}}_{mm'} \). Further, \( j_n \) is the spherical Bessel function.

Inserting the Gaussian plane-wave Bessel identity [9, (16)]

\[ j_n (k \sqrt{\mathbf{b} \cdot \mathbf{b}}) P_n \left( \frac{\mathbf{r}_{mm'} \cdot \mathbf{b}}{\sqrt{\mathbf{b} \cdot \mathbf{b}}} \right) = \frac{i^{-n}}{4\pi} \iint e^{i k \mathbf{b} \cdot \mathbf{T}_N} (\mathbf{k}, \mathbf{r}_{mm'}) d^2 \mathbf{k} \]

into (8), we get

\[ G(\mathbf{r}_e, \mathbf{r}_{mm'}) = \frac{i k}{4\pi} \iint T_N (\mathbf{k}, \mathbf{r}_{mm'}, \Delta) e^{i \mathbf{k} \cdot \mathbf{r}_e} d^2 \mathbf{k} \]

which can be inserted into the EFIE to yield an analogous expression to (2), aside from the new translation operator \( T_N \).

The advantage of \( T_N \) is that by increasing \( \Delta \) the translation operator \( T_N \) becomes more directive. Consequently, only a small subset of the \( K_L \) tabulated directions are necessary, and the rest can be discarded during the translation stage. However, to allow for a large value of \( \Delta \) while maintaining the accuracy, \( N \) in (7) has to be increased slightly relative to the usual upper limit \( L \) chosen from (5) for the Rokhlin translator.

While [9] gave a theoretical upper limit for \( \Delta \), called \( \Delta_{max} \), practical applications need to choose \( \Delta \) significantly smaller than \( \Delta_{max} \) due to numerical cancellation [9, Section 5.3] when computing \( T_N \). Further, since we want to increase \( \Delta \) as much as possible to achieve a directive operator, it is important to note that \( \Delta \) can be chosen larger for groups that are far apart than for closely spaced groups.

This is particularly important when considering interactions where \( |\mathbf{r}_{mm'}| < N/k \), at which point the expression (5) is no longer valid. In general, the well-known numerical concerns [11] of applying \( T_L \) are worsened for \( T_N \) when \( \Delta > 0 \).

To apply the Gaussian translation operator according to the existing literature, one needs to choose allowed upper limits for \( N \) and \( \Delta/|\mathbf{x}| \). Based on this, a value of \( \Delta \) can be found by increasing \( \Delta \) until \( \frac{\Delta}{|\mathbf{x}|} \) is greater than the limit, or until

\[ \frac{\left| \overline{G(\mathbf{r}_e, \mathbf{r}_{mm'})} - \sum_{n=0}^{N} U_n (\mathbf{k}, \mathbf{r}_{mm'}, \mathbf{b}, \Delta) \right|}{\overline{|G(\mathbf{r}_e, \mathbf{r}_{mm'})|}} \approx 10^{-3} \]

where \( \mathbf{r}_e \) is a suitably chosen vector.

Having found \( \Delta \) for each unique translation vector \( \mathbf{r}_{mm'} \), the translator is evaluated and only the tabulated plane-wave directions \( \mathbf{k}_t \) for which

\[ |T_N (\mathbf{k}_t, \mathbf{r}_{mm'}, \Delta)| > \max_{\mathbf{k}} |T_N (\mathbf{k}, \mathbf{r}_{mm'}, \Delta)| \cdot p \]

are included in the translation stage, where \( p \) is some scalar threshold, e.g., \( p = 10^{-12} \) [9]. Since the sharpness of the translator is directly proportional to the value of \( \Delta \), this demonstrates why increasing \( \Delta \) as much as possible is desirable. Thus, there is a compromise between increasing \( \Delta \) (and thereby increasing \( N \) to maintain accuracy) as much as possible to increase computational speed, and keeping \( N \) and thus \( K_N = 2(N+1)^2 \) as small as possible, to reduce memory costs for the group patterns. Below, \( Q \) denotes the number of included plane-wave directions (e.g., as determined by (12)).

III. IMPLEMENTATION DETAILS

While the previous section presented the standard implementation, a number of parameters were not discussed. In this section, we discuss procedures to automatically choose them. Provided that the implementation is done as detailed in the following, applying the Gaussian translation operator will not require new parameters that must be chosen manually.

A. Choosing \( N \)

The truncation limit for the Gaussian translation operator (6) must be higher than \( L \), but not too high, as discussed previously. The optimal value of \( N \) is a tradeoff between computational speed and required memory, and will depend on the implementation. By increasing \( N \), the number of stored plane-wave directions \( K_N \) will increase, but the translator can be made more directive and therefore, the translation step will be much faster. We have used \( N = 1.1 L \) for all levels of the Octree, but we note that our implementation also uses adaptive grouping [12] of the basis function signatures, which means that adjusting \( N \) on the finest level of the Octree does not affect the sampling rate of the basis function signatures. For implementations without reduced sampling rates, we suggest to use the Rokhlin translator on the finest level.

Furthermore, for specific scenarios such as distributed memory MLFMM or scatterers with particularly simple geometries,
the optimal \( N \) will likely be significantly higher. For distributed memory MLFMM, a significant increase in \( K_N \) will be likely be acceptable, since the translation communication is often the bottleneck. For special geometries, not all \( K_N \) directions need to be stored, so the increase is irrelevant.

**B. Choosing \( \Delta \)**

We note that basing the choice of \( N \) on (11) will lead to inaccurate solutions when the Hankel function oscillates [11]. This is completely analogous to the scenario for the Rokhlin translator, where the EBF (5) is based on the Gegenbauer addition theorem. Thus, the EBF cannot be used when \( L > k|\mathbf{x}| \) and remedies have been discussed in the literature [11].

Instead, we propose to base the choice of \( \Delta \) on the Gaussian plane-wave Bessel identity (10). Using this identity, we have

\[
\frac{|G(r_e, r_{mm'})|}{|\tilde{G}(r_e, r_{mm'})|} = \frac{\int \int T_N(k, r_{mm'}, \Delta) e^{i k \cdot \mathbf{r}_e \cdot \mathbf{d}} d^2 \mathbf{k}}{\int \int T_N(k, r_{mm'}, \Delta) d^2 \mathbf{k}} < 10^{-\beta}.
\]

(13)

A practical algorithm would be as follows. Set \( r_e = D \mathbf{a} \), where \( \mathbf{a} \) is a vector orthogonal to \( r_{mm'} \), otherwise arbitrary. For each unique translation vector \( \mathbf{x} \) on each level of the Octree, where \( D \) is the diameter of the groups at the current level, perform the following:

1. If (13) is not fulfilled for \( N = 1.1L \) and \( \Delta = 0 \), set \( N = L \) and use the Rokhlin translator (4). In particular, this will happen to those translators where \( |\mathbf{x}| = 2D/\sqrt{3} \).
2. Otherwise, use bisection for the interval \( \Delta_{\text{min}} = 0 \), \( \Delta_{\text{max}} = |\mathbf{x}|^2 - D^2 \) [9] to find the value of \( \Delta \) that only just fulfills (13).

In practice, repeated calculation of the translation function in (13) is time consuming, so the above procedure might initially seem too costly. However, we stress that only \( \Delta \) is modified, so only the Hankel function in (6) needs to be recomputed during the bisection approach, while the other terms, in particular the Legendre polynomial, should be precomputed and stored. Using this approach, the time to set up the Gaussian translator is only negligibly increased compared to that of the Rokhlin translator. We also note that the application of interpolation procedures for computing the translation function [13], [14] is still possible, provided that the increased bandwidth of the Gaussian translator relative to the Rokhlin translator is taken into account.

**C. The number of plane-wave directions**

Having found the highest possible value of \( \Delta \) for each translator, we must choose the plane-wave directions to include in the translation procedure. The threshold (12) was used in previous work [9], with \( p = 10^{-12} \), not taking into account the desired relative error. The threshold was found to be far too pessimistic in [15], which suggested \( p = 10^{2.3} \). In the present paper, a more thorough criterion is considered.

We base the criterion on the integral over \( T_N \), such that

\[
\frac{\sum_{i=1}^{N_s} w_i T_N(k_i, r_{mm'}, \Delta) - \int \int T_N(k_i, r_{mm'}, \Delta) d^2 \mathbf{k}}{|\int \int T_N(k_i, r_{mm'}, \Delta) d^2 \mathbf{k}|} \approx 10^{-\beta}.
\]

(14)

In practice, this is done by sorting the values \( w_i T_N(k_i, r_{mm'}, \Delta) \) in ascending order, and then discarding the smallest values until (14) is fulfilled. We remark that in general, data corresponding to the directions to be included will rarely be consecutively placed in memory. Thus, we do not utilize the Gaussian translator if \( Q/K_N > 0.8 \), since there is some computational overhead involved in keeping track of the directions to be skipped by the Gaussian translator.

**IV. NUMERICAL RESULTS**

The implementation used to obtain the numerical results is described in [16], where we have used a maximum patch size of 1.5\( \lambda \). Implementation details other than those described previously have little impact on the results in this section.

We remark that the Gaussian translator cannot be efficiently combined with global interpolation algorithms [17], [18], as was noted by [10]. While it will yield correct results, provided that the increased bandwidth relative to the Rokhlin translator is taken into account, it will generally not be possible to disregard plane-wave directions since the directivity of the translator is “smeared” by the global interpolators.

As a test-case, we consider the scattered field from a 1 m radius sphere illuminated by a plane wave at 10 GHz, solved by applying the CFIE. This allows the use of the Mie series as an exact reference solution. We use \( \beta = 3 \).

As a measure of accuracy, we use the Relative RMS Error, defined as

\[
\text{Relative RMS Error} = \sqrt{\frac{\sum_{i=1}^{N_s} |E_{i, \text{ref}} - E_{i, \text{cal}}|^2}{\sum_{i=1}^{N_s} |E_{i, \text{ref}}|^2}},
\]

(15)

where \( E_{i, \text{ref}} \) and \( E_{i, \text{cal}} \) denote the electric far field at the \( i \)th sample point from the reference and calculated scattered fields respectively, and \( N_s \) is the number of samples.

The scattered field is shown in Figure 1. The relative error is 0.39% using the Gaussian translator, and this is
indistinguishable from the result using the Rokhlin translator, with a relative error of 0.38%. The computational resources are shown in Table I. \( t_N \) and \( t_L \) indicate the time to perform translation on the given level for the Gaussian and Rokhlin translator, respectively. Note that \( t_N = t_L \) on the finest level, because here we switched to the Rokhlin translator since \( Q/K_N > 0.8 \). The total time saving was roughly 40%. Table I shows that the benefits of using \( T_N \) is much larger as the group size increases. We also see that the memory for the translators is significantly reduced on the coarsest levels, but this comes at the cost of increasing the group memory. This is due to the fact that \( N > L \) and therefore \( K_N > K_L \), and consequently, more directions have to be tabulated for each group.

In summary, we stress that the primary advantages of the translator should not be found in MLFMM shared memory implementations for general scattering problems. For such scenarios, the relative gains might not be sufficient, and the directivity of the operator does not carry over to the group patterns, since groups will interact with groups from various directions, thereby requiring increased memory for the group patterns. We also refer to [15], which contains further numerical results.

However, the Gaussian translation operator will be extremely beneficial for more specialized implementations, i.e.

- Distributed memory MLFMM [19], where the translators have to be copied on each node. In this scenario, the vastly reduced memory for the translators on large levels would be very useful. Furthermore, the need for only a small angular region on the shared levels would require much less communication between the nodes.
- Directive scenarios, such as near-field to far-field transformations [10], [20] or imaging, where all the (very well-separated) far-field interactions occur along small angular regions. Using the Gaussian translation operator would not only yield a much faster translation step, but also allow us to only store a small subset of the aggregated group patterns.
- Quasi-planar scattering problems such as planar arrays, where most of the far-interactions are with elements that are spaced at a fairly large distance, allowing for significant advantages by only using a small angular region. Further, we note that for very specific configurations [9], which could be present in linear array problems, \( N < L \) and thus the Gaussian translator is even more advantageous.

### V. Conclusion

An implementation of a directive translation operator for MLFMM was presented. The Gaussian translator allows for any precision to be achieved, and can be implemented without modifying the other parts of an existing MLFMM setup. Methods for choosing all involved parameters automatically were discussed, and results for an actual scattering case were shown. For general scattering problems, the translator provides fairly modest gains, but for more specialized implementations such as array scattering and distributed memory MLFMM, the Gaussian translation operator would greatly outperform the standard translator.

### References

A.4 Paper IV

**Performance of a Low Sample Rate Interpolator for Multi-Level Fast Multipole Method**

Oscar Borries, Erik Jørgensen, Peter Meincke, Per Christian Hansen and Seppo Järvenpää

*Status*

Accepted: December 2014

**Bibliography**

Performance of a Low Sample Rate Interpolator for Multi-Level Fast Multipole Method

Oscar Borries, Erik Jørgensen, Peter Meincke, Per Christian Hansen, and Seppo Järvenpää

Abstract—Global interpolators with very low memory requirements, based on trigonometric polynomial expansions, are applied to the Multi-Level Fast Multipole Method and for the first time test cases demonstrate the significant memory savings for actual scattering problems.

Index Terms—MLFMM, Interpolation

I. INTRODUCTION

The Multi-Level Fast Multipole Method [1]–[3] (MLFMM) can be used to solve the electromagnetic scattering problem of finding the currents on a structure induced by an incident field. The scattering problem is solved by discretizing an integral equation using the Galerkin approach, resulting in a dense linear system. Solving the system by an iterative method, the MLFMM allows for fast matrix-vector products, allowing the computation and memory requirements to scale as \(O(N \log N)\), where \(N\) is the number of unknowns.

As part of an MLFMM implementation, interpolation is needed to go between the group patterns at various levels. This is done either with local or global methods for the interpolation — often using either Lagrange polynomials or trigonometric polynomials, respectively, with local interpolation appearing to be the method of choice for most applications. Local interpolation gives a simple implementation, while global methods provide better control of the error, remove the need for oversampling, and can utilize highly optimized fast methods such as FFT. As a compromise between local and global interpolators, quasi-bandlimited interpolators as discussed in [4] can be beneficial.

Recently, [5] improved the global interpolation method by Sarvas [6], presenting a global method with significantly lower sampling rates than Lagrange interpolation. The method is more accurate and requires a lower sampling rate than previously proposed interpolation methods, while offering a fairly straightforward implementation.

In [5], the authors described the method in a concise and thorough manner, including a detailed walkthrough of the interpolation process. However, they did not discuss scattering problems, leaving readers unable to directly gauge the performance and memory improvements achieved by the method.

This paper presents results from realistic test cases, using a scheme discussed in [7]. Our focus is not on the discretization or direct performance, but on the savings achieved by this interpolation method in comparison with the popular Lagrange interpolation for scattering problems. Also, the accuracy of an alternative method for finding the necessary sampling rates is briefly discussed. Moreover, we demonstrate further memory reductions through combining the interpolation method with the use of Spherical Harmonics Expansions [8]. The time factor \(e^{j\omega t}\), \(\omega\) being the angular frequency, is assumed and suppressed throughout.

II. MLFMM WITH LOCAL INTERPOLATORS

We consider the scattering problem from an incident electromagnetic wave on a perfectly electrically conducting (PEC) scatterer \(S\). Galerkin testing of the Electric Field Integral Equation (EFIE) [9] yields a linear system \(\mathbf{Z} \mathbf{T} = \mathbf{I}\), where \(Z_{j,i}\) is the mutual impedance between basis functions \(f_j\) and \(f_i\) [10]. MLFMM can be considered as a partitioning of the matrix \(\mathbf{Z}\) into two parts,

\[
\mathbf{Z} = \mathbf{Z}_{\text{near}} + \mathbf{Z}_{\text{far}},
\]

where the near-matrix \(\mathbf{Z}_{\text{near}}\) is stored as a sparse matrix, while \(\mathbf{Z}_{\text{far}}\) is not stored directly. Instead the information required to multiply \(\mathbf{Z}_{\text{far}}\) with an excitation vector \(\mathbf{T}\) are stored.

The MLFMM using local interpolators, applied to the interaction between basis function \(f_j\) in group \(m\) and basis function \(f_i\) in group \(m'\), can be computed by evaluating the integral

\[
Z_{j,i} \approx \kappa \iint R_{jm}(k) \cdot (T_L(k, r_{mm'}) V_{mn}(k)) \, dk,
\]

where the distance between the two group centers is \(|r_{mm'}| = |r_m - r_m'| > D\), \(D\) being the diameter of the groups. \(T_L\) is Rokhlin’s translation function [11], and the accuracy of the approximation to \(Z_{j,i}\) can be controlled by adjusting \(L\), which is chosen using the Excess Bandwidth Formula [12], requesting an error of \(10^{-\beta}\). The basis function patterns \(R_{jm}(k)\) and \(V_{jm}(k)\) are detailed in [1] — for the EFIE, \(R_{jm} = V_{jm}^*\), where * denotes complex conjugation. Further, \(\kappa\) is a constant, \(k\) is the wavenumber \(2\pi/\lambda\), \(\hat{k}\) is the unit wave-vector \(k = \sin \theta \cos \phi \hat{x} + \sin \theta \sin \phi \hat{y} + \cos \theta \hat{z}\) and \(k = \hat{k}\).

The integral (2) can be discretized and calculated in a multi-level approach by computing

\[
Z_{j,i} \approx \kappa \sum_{p=1}^{K^{(3)}} w_p^{(3)} R_{jm4}(k_p^{(3)}) \cdot \left( e^{-jk_p^{(3)} r_{m4}} V_{mn1}(k_p^{(3)}) \right).
\]

Here, \(K\) is the number of sample points on the unit sphere and \(w_p\) are integration weights. The superscripts \((q)\) define entities on the \(q\)’th level of the Octree [13], with lower numbers referring to coarser levels such that \(D^{(q)} = 2D^{(q+1)}\). Level 1 is the bounding cube for \(S\). Since \(V_{mn1}(k_p^{(3)})\) is only tabulated on level 4 and thus has a coarser sampling than required in (3), \(V_{mn1}(k_p^{(3)})\) has to be obtained through interpolation.

When using local interpolators to compute \(V_{mn1}(k_p^{(3)})\), matrix-vector products with a sparse interpolation matrix
\[ W \in \mathbb{R}^{K(3) \times K(4)} \] are performed. The elements of \( \mathbf{W} \) can be expressed as
\[
W_{j,i} = \prod_{n=1, \theta_n \neq \theta_i}^{H} \frac{\theta_j - \theta_n}{\theta_i - \theta_n}, \quad (16)
\]
such that the \( H \) points \((\theta_n, \phi_n)\) closest to \((\theta_j, \phi_j)\) are used in the interpolation.

The tabulation of \( \hat{k} \) is done by choosing \( L + 1 \) Gauss-Legendre points in \( \cos \theta \) and \( 2(L + 1) \) equidistant points in \( \phi \in [0, 2\pi - 2\pi/2(L + 1)]. \) An important consideration when using Lagrange interpolation is the use of oversampling, i.e., using \( \alpha(L + 1) \times M2(L + 1) \) samples, \( \alpha > 1, \) versus increasing the order of the interpolant. The relative error \( \delta \) of the local interpolator behaves as
\[
\delta \approx \sqrt{\frac{2}{H\pi}} \left( \frac{\pi}{\alpha} \right)^H, \quad (5)
\]
indicating that the interpolation error decreases exponentially when \( H \) is increased. This relation extends the result for \( \alpha = 1 \) shown in [14, (3.43)].

However, \( \mathbf{W} \) will contain \( K(3)H^2 \) elements, so in order to increase the speed of the matrix-vector product with \( \mathbf{W} \), one might favour a larger \( \alpha \) to decrease the error rather than increasing \( H \). It is noted that there is no need to oversample the basis function patterns \( V_{m\ell} \) and \( R_{j\ell} \), since these are oversampled by default [14, p. 918].

### III. Trigonometric Polynomial Expansions in MLFMM

The concept behind the approach of [5] is to represent all the functions involved in (2) as trigonometric polynomial expansions (TPEs) of two variables and order \((M, N)\), such that a vector function defined on the sphere \( \mathbf{F}(\theta, \phi) = F^{[\theta]}(\theta, \phi) + F^{[\phi]}(\theta, \phi) \) can be expressed as
\[
F^{[\theta]}(\theta, \phi) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} f_{m,n}^{[\theta]} e^{i(m \theta + n \phi)}, \quad (6a)
\]
\[
F^{[\phi]}(\theta, \phi) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} f_{m,n}^{[\phi]} e^{i(m \theta + n \phi)}. \quad (6b)
\]
The TPE coefficients \( f_{m,n}^{[\theta]}, f_{m,n}^{[\phi]} \) are computed on-the-fly when needed but are not stored. The storage of \( \mathbf{F}(\theta, \phi) \) consists instead of a set of samples of \( \mathbf{F} \) tabulated along latitudinal lines on the unit sphere. This provides memory efficient storage, since the behaviour of spherically smooth, band-limited functions can be represented using fewer samples near the poles of the sphere than at the equator, thereby reducing the number of samples needed compared to a grid in \((\theta, \phi)\). More precisely, the stored samples of \( \mathbf{F} \) are taken at the points
\[
\theta_i^M = \frac{2\pi}{2M + 1} i, \quad i = 0, \ldots, M \quad (7a)
\]
\[
\phi_j^N = \frac{2\pi}{2N + 1} j, \quad j = 0, \ldots, 2N \quad (7b)
\]
where the number of samples in \( \phi \) depends on \( i \) and thus on the value of \( \theta \). An important advantage is that the choice of \( M \) and \( N_i \) in (7) can be based on the bandwidth of the bandlimited functions and the desired relative accuracy, yielding a tight bound on the error. The order \( M \) or \( N_i \) is chosen to ensure that
\[
|e^{ijx cos(\phi)} - \sum_{n=-M}^{M} a_n e^{j(n+1)\phi}| \leq 10^{-\beta}, \quad \forall \phi, \quad (8)
\]
where \( a_n \) are coefficients from the Discrete Fourier Transform of \( e^{ijx cos(\phi)} \). This is treated in [5], which describes a function \exporder\ for determining \( M \) or \( N_i \), derived by numerically solving (8).

The problem (8) is equivalent to
\[
|e^{ijx cos(\phi)} - \sum_{n=-M}^{M} a_n j^n e^{j(n+1)\phi}| \leq 10^{-\beta}, \quad \forall \phi \quad (9)
\]
which is solved approximately, using analytical tools, in [15] as
\[
M = x + \gamma x^{1/3}, \quad \gamma = \frac{(-3 \ln 10^{-\beta})^{2/3}}{2}. \quad (10)
\]
However, our numerical experiments have demonstrated that (10) is a slightly less accurate solution to (8) than that provided by \exporder, and thus we use \exporder.

The advantage of the TPE representation is that it allows the use of the FFT to find the coefficients \( f_{m,n} \) and evaluate the series (6). Further, since it employs an orthogonal basis in (6), the integral of the product from (2) only has contributions corresponding to the lowest bandwidth, yielding a very efficient scheme. To evaluate the integral, (2) can be rewritten as
\[
Z_{j,i} \simeq \frac{1}{2} \int_0^{2\pi} \int_0^{2\pi} R_{j\ell}(k) \cdot (T_L(k, r_{m,n}) V_{m\ell}(k) | \sin \theta |) \, d\theta d\phi. \quad (11)
\]
Representing each of these factors as TPEs, by sampling at the points (7), such that
\[
R_{j\ell}(k) \simeq F(\theta, \phi), \quad (12)
\]
\[
\frac{1}{2} T_L(k, r_{m,n}) | \sin \theta | V_{m\ell}(k) \simeq G(\theta, \phi), \quad (13)
\]
allows (11) to be computed by performing the integration
\[
Z_{j,i} \simeq \int_0^{2\pi} \int_0^{2\pi} F(\theta, \phi) \cdot G(\theta, \phi) d\theta d\phi \quad (14)
\]
\[
= T(F^{[\theta]} \cdot G^{[\theta]} + T(F^{[\phi]} \cdot G^{[\phi]}), \quad (15)
\]
where the operator \( T \) integrates the product of the functions according to the trapezoidal rule
\[
T(F^{[\theta]} \cdot G^{[\theta]} = \frac{8\pi^2}{2M + 1} \left[ \frac{1}{2(2N_0 + 1)} \sum_{j=0}^{2N_0} F^{[\theta]}(0, \phi_{jN_0}) G^{[\theta]}(0, \phi_{jN_0}) + \sum_{i=1}^{M} \frac{1}{(2N_i + 1)} \sum_{j=0}^{2N_i} F^{[\theta]}(\theta_i, \phi_{jN_i}) G^{[\theta]}(\theta_i, \phi_{jN_i}) \right]. \quad (16)
\]
The complicated part of TPE-MLFMM is obtaining \( G \) from (13). This involves some elaborate procedures using (6),
detailed in [5], particularly because the bandwidth of the functions involved differs — indeed, |sin \theta| is not even band-limited.

An often cited drawback of global interpolators is that the asymptotic complexity of evaluating a TPE at W points is \( O(W \log W) \), while the evaluation of the local interpolation matrix is of complexity \( O(K) \). However, the numerical results in the next section demonstrate that if the same accuracy is to be reached, \( K \) will be significantly larger than \( W \). Further, while most programming languages have access to extremely optimized FFT procedures, local interpolation procedures often need to be customized to the application. This indicates that for most reasonable problem sizes, interpolation using TPE is likely to be more efficient than local interpolation.

The most important limit to the performance of TPE-MLFMM is the behavior when using a distributed memory implementation with group patterns that require too much memory to be stored on a single node. Due to the low memory required for storing a TPE, group patterns would need to be separated across nodes only for extreme problem sizes. In this case, one should switch to a local interpolation scheme on the shared levels, as discussed in [16].

A further reduction in memory, beyond that achieved in [5], is obtained by combining TPE-MLFMM with the Spherical Harmonics Expansion (SHE) approach for storing the basis function patterns [8]. This requires converting \( G \) in (13) to

\[
G(\theta, \phi) = \sum_{p=0}^{W} \sum_{q=-p}^{p} r_{pq} Y_{pq}(\theta, \phi),
\]

where \( Y_{pq}(\theta, \phi) \) are the orthonormalized spherical harmonics.

The integral required to obtain \( r_{pq} \) in (17) is

\[
r_{pq} = \int \int g_\psi(\theta, \phi) Y_{pq}^*(\theta, \phi) d^2 \mathbf{k},
\]

where \( \psi \) is a component of \( G \). For fast convergence of the SHE it is recommended to store the Cartesian rather than the spherical components, and do the conversions on-the-fly during the matrix-vector product, as discussed in [8]. The integral (18) can be done very effectively, by applying (16), setting \( F = Y_{pq}^* \). The conversion between spherical and Cartesian components can be avoided by the use of vector spherical harmonics [17] for MLFMM, as discussed in [18, Section 4.2.2].

IV. NUMERICAL RESULTS

Our numerical computations use the MLFMM implementation detailed in [7], based on a discretization using Higher-Order Legendre basis functions and curved quadrilateral patches [19], [20]. To isolate the effects of applying TPE instead of Lagrange interpolation, we do not apply adaptive grouping [21]. We merely note that the use of adaptive grouping is trivially combined with the present interpolators.

The numerical experiments are performed on a workstation with two Intel Xeon E5-2690 2.9 GHz processors using 16 cores, and 160 GB memory. Where applicable, MKL (particularly BLAS and MKL-FFT) routines have been applied to speed up the computations.

A. Sphere

The first test case involves a plane wave at 5 GHz incident on a 1 m radius PEC sphere. The problem is discretized with 2nd order basis functions on a fairly fine mesh with sidelength at most 0.3λ, yielding 371172 unknowns, and the Combined Field Integral Equation (CFIE) [9] is applied to a 4-level MLFMM structure, such that 4 interpolation and 4 antiprolation steps are performed.

We compute the error as a Relative RMS Error of the scattered field, i.e.

\[
\text{Relative RMS Error} = \sqrt{\frac{\sum_{i=1}^{N_s} |E_{i,\text{ref}} - E_{i,\text{cal}}|^2}{\sum_{i=1}^{N_s} |E_{i,\text{ref}}|^2}},
\]

where \( E_{i,\text{ref}} \) and \( E_{i,\text{cal}} \) denote the electric far field at the \( i \)-th sample point from the analytical reference and calculated scattered fields, respectively, and \( N_s \) is the number of samples.

The ideal error estimate would only take into account the interpolation accuracy, disregarding all other factors. However, such an estimate cannot be obtained directly, and instead the Relative RMS Error is used as an error estimate. While it is an indirect measure of the interpolation accuracy, since it is a combination of many factors, we can isolate the effect of the interpolation on the final result by only varying the interpolation method and keeping all other settings fixed. Further, we apply a fine discretization, yielding a lower discretization error than would be common in practice. This is done to ensure that the interpolation, and not the discretization, is the dominant error source.

Figure 1 shows the Relative RMS Error in the E-plane, obtained as a function of the interpolation settings for Lagrange, as well as that obtained using TPE-MLFMM. We clearly see that a fairly high interpolation order \( H \) is needed and that the lower orders require significant oversampling \( \alpha \), which in turn is costly in terms of memory. Table I shows the memory used for both Lagrange interpolation, and TPE-MLFMM. The order of the Lagrange interpolation used in Table I is \( H = 5 \), but the memory is almost independent of \( H \), aside from the storage for the interpolation matrices \( \mathbf{W} \).

While the conclusion from Table I would seem to suggest to use low oversampling, and thus a high order to achieve high accuracy, Figure 2 demonstrates that this results in a noticeable increase in computational time. Also shown in Table I is the memory required by combining TPE-MLFMM with SHE storage of the basis functions as discussed in Section III — while requiring a more involved algorithm, it significantly reduces the MLFMM memory. The additional time for the matrix-vector product using SHE is negligible, as was also observed in [7], [8].

B. Rooftop Mirror

As a second test case, a plane-wave at 3.75 GHz is incident on a rooftop mirror, consisting of two \( 5 \times 5 \) m square plates, joined at the edges, forming a right angle. The discretization with 1.5λ mesh length and 5-th order polynomials yields 175770 unknowns, grouped in a 3-level MLFMM. This problem is solved using the EFIE with both Lagrange interpolation,
TABLE I

<table>
<thead>
<tr>
<th>Sphere - 5 GHz (GB)</th>
<th>α = 1</th>
<th>α = 1.25</th>
<th>α = 1.5</th>
<th>α = 1.75</th>
<th>α = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lagrange</td>
<td>7.97</td>
<td>8.10</td>
<td>8.29</td>
<td>8.50</td>
<td>8.76</td>
</tr>
<tr>
<td>TPE-MLFMM</td>
<td>4.07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPE-SHE</td>
<td>2.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. The Relative RMS Error for Lagrange interpolation, with varying interpolation settings α and H, and for TPE-MLFMM which is independent of α. The discretization is the same for all interpolation settings.

Fig. 2. The time per iteration for Lagrange interpolation, with varying interpolation settings α and H, and for TPE-MLFMM which is independent of α.

C. Satellite Mock-up

As a final test case, we consider the scattered field from a rough mock-up of the Planck satellite [22]. The main part of the satellite features two large reflectors in a dual reflector configuration inside a shielding structure as shown in Figure 3. The ellipsoidal subreflector, at the bottom of the Figure, is illuminated by a horn radiating at 20 GHz near the bottom of the cavity. The total surface area is $75.4 \cdot 10^3 \lambda^2$.

Using patches with sidelengths between $0.56\lambda$ and $1.99\lambda$, this leads to a problem with 2135600 unknowns, grouped in a 4-level MLFMM, with up to 8th order polynomials used on the largest patches. An RWG [23] discretization would require roughly 9 million unknowns to achieve comparable accuracy.

The MLFMM memory using a Lagrange interpolation scheme with $H = 7$ and no oversampling is 164.3 GB. Using TPE-MLFMM reduces that to 52.1 GB, while using TPE-MLFMM and SHE requires 37.0 GB.

V. Conclusion

Our results, summarized in Table II, clearly show that using TPE-MLFMM significantly reduces the memory and time required to apply MLFMM. While the concern that global interpolators have a higher asymptotic cost is often cited as reason for using local interpolation methods, our results demonstrate that this is unlikely to affect the superiority of TPE-MLFMM until problems become so large that group patterns cannot be stored on a single node in a distributed
memory setup. Thus, for the vast majority of practical cases, TPE-MLFMM offers a fast, low-memory and fully error-controllable scheme for solving scattering problems.

ACKNOWLEDGEMENT

The authors thank Dr. Thorkild Hansen, Seknion Inc., for pointing out the connection between the problems (8) and (9).

REFERENCES


A.5 Conference Paper I

**IMPROVED MULTILEVEL FAST MULTIPOLe METHOD FOR HIGHER-ORDER DISCRETIZATIONS**

Oscar Borries, Peter Meincke, Erik Jørgensen, Stig Busk Sørensen and Per Christian Hansen

*Status*

Awarded the 2nd place in the Student Paper Competition.

**Bibliography**

Improved Multilevel Fast Multipole Method for Higher-Order Discretizations

Oscar Borries\textsuperscript{1,2}, Peter Meincke\textsuperscript{2}, Erik Jørgensen\textsuperscript{2}, Stig Busk Sørensen\textsuperscript{2}, Per Christian Hansen\textsuperscript{1}

\textsuperscript{1}Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark, \{opbo, pcha\}@dtu.dk
\textsuperscript{2}TICRA, Læderstræde 34, DK-1201 Copenhagen, Denmark, \{ob, pm, ej, sbs\}@ticra.com

Abstract—The Multilevel Fast Multipole Method (MLFMM) allows for a reduced computational complexity when solving electromagnetic scattering problems. Combining this with the reduced number of unknowns provided by Higher-Order discretizations has proven to be a difficult task, with the general conclusion being that going above 2nd order is not worthwhile. In this paper, we challenge this conclusion, providing results that demonstrate the potential performance gains with Higher-Order MLFMM and showing some modifications to the traditional MLFMM that can benefit both Higher-Order and standard discretizations.

Index Terms—Multilevel Fast Multipole Method, Computational Electromagnetics, Higher-Order Discretization, Electromagnetic Scattering

I. INTRODUCTION

Discretization of the surface current density occurring in either the Electric Field Integral Equation (EFIE) or the Combined Field Integral Equation (CFIE) can be done using a wide variety of methods, most popularly the RWG [1] first-order basis functions on a triangular mesh. When discretizing the scattering problem using a Method of Moments (MoM) approach, the resulting matrix equation has \(N\) unknowns and thus requires \(O(N^2)\) memory and computational time to solve iteratively. Thus, a key issue in an efficient solution is the reduction of \(N\) relative to the target accuracy of the obtained surface current density. It has been demonstrated [2] that Higher-Order (HO) basis functions significantly reduces the number of unknowns and thus requires \(O(N \log N)\) memory and computational time to solve.

Another line of research has focused on reducing the complexity of the matrix vector product \(ZI\). One of the most popular approaches is the Multilevel Fast Multipole Method (MLFMM) [3], [4], which achieves \(O(N \log N)\).

However, combining the advantages of those two approaches has proved elusive. While several groups have made the attempt [5], [6], [7], each has independently arrived at the conclusion that basis functions above 2nd order resulted in a memory increase and thus were not efficient for MLFMM use. We challenge that consensus with a carefully revised algorithm that is tailored towards the larger group sizes occurring in HO MLFMM, thereby allowing the reduced number of unknowns to result in a reduction in memory and, nearly as important, a significant reduction in computation time.

II. MULTILEVEL FAST MULTIPOLe METHOD

MLFMM achieves reduced complexity by grouping the basis functions hierarchically, using the Octree algorithm [8], and letting larger and larger groups interact over greater and greater distances. The grouping is based on the center of the geometric elements of the mesh, and the smallest allowed groups have sidelengths not smaller than the largest geometric element in the mesh.

This splitting allows performing the matrix-vector product as

\[
\bar{Z}I = \bar{Z}_{\text{near}}I + F(I) \tag{1}
\]

where \(\bar{Z}_{\text{near}}\) is the near-matrix, containing the interactions between basis functions that are too closely spaced to apply MLFMM — the elements in this matrix are computed as in the normal MoM approach. \(F\) denotes the operation performed by applying MLFMM.

The interaction between two well-separated basis functions \(f_j, f_i\), belonging to groups \(m\) and \(m'\) respectively, can be computed by

\[
\bar{Z}_{j,i} = \kappa \iint R_{jm}(k, \hat{k}) \cdot \left( T_L(k, \hat{k}, r_{mm'}) V_{im'}(k, \hat{k}) \right) d^2\hat{k}, \tag{2}
\]

with \(r_{mm'} = r_m - r_{m'}\), where \(r_m\) is the center of group \(m\), and for EFIE \(\kappa = -j\frac{\eta}{2\pi}\), while for the Magnetic Field Integral Equation (MFIE), \(\kappa = -\frac{\eta}{2\pi}\), where \(\eta\) is the free-space impedance. For EFIE, the basis function signature \(R_{jm}(k, \hat{k}) = V_{jm'}(k, \hat{k}, p)\) and Rokhlin's translation function \(T_L\) [9] is computed as

\[
T_L(k, \hat{k}, x) = \sum_{l=0}^{L} (-j)^l (2l+1) h_l^{(2)}(k|x|) P_l(\hat{k} \cdot \hat{x}), \tag{4}
\]

where \(\hat{k}\) is the unit wave vector, \(x\) is the vector between two group centers directed towards the receiving group, \(\hat{x} = x/|x|\), \(h_l^{(2)}\) is the spherical Hankel function of second kind and order \(l\), and \(P_l\) is the Legendre polynomial of order \(l\).

Discretizing (2), we get

\[
\bar{Z}_{j,i} = \kappa \sum_{p=1}^{K} w_p R_{jm}(k, \hat{k}_p) \cdot \left( T_L(k, \hat{k}_p, r_{mm'}) V_{im'}(k, \hat{k}_p) \right) \tag{5}
\]
Here, $w_p$ are the integration weights.

The number of terms $K$ in the sum (5) is related to the number of terms $L$ in the translation function (4) by the relation $K = 2(L + 1)^2$. Since $L$ is determined from the diameter $D$ of the groups and the required relative accuracy $10^{-3}$ as

$$L = kD + 1.8/2^{3/2}(kD)^{1/3},$$

the obvious question is whether the decrease in the number of unknowns can counteract the increase in $L$ and thus in $K$ that comes from the increased basis function support and thus increased $D$. For a standard MLFMM implementation with a HO discretization, the increase in $K$ (along with a larger $Z_{\text{near}}$) is far more devastating than the benefits of reducing $N$, which is essentially the conclusion reached by other research groups. However, the results in this paper show that with a few careful modifications to the standard algorithm, the benefits of Higher-Order discretizations can be combined with MLFMM to result in a low memory and computationally efficient algorithm.

### III. Improvements

To achieve high performance for Higher-Order MLFMM, some improvements are needed. First, to reduce the noticeable memory burden for storing index integers for the sparse near-field matrix, we apply a new format which uses only two integers to store an arbitrary number of consecutive column indices. With a good ordering of the basis functions, this yields a significant saving for a HO discretization, illustrated in Table I for a specific example.

Second, the use of an adaptive grouping scheme allows for much lower memory to be used for storing the basis function patterns (3) by adding an additional level beneath the lowest level in the Octree. With this, the center of the group $r_m$ is changed to the center of an adaptive group $r_m\prime$ in (3), such that (5) is changed to

$$\hat{Z}_{j,i} = \kappa \sum_{p=1}^{K} w_p V^*_{j,m}(k, \vec{k}_p) \cdot \left(W T e^{-jkk \cdot r_{m\prime}} T^*_L(k, \vec{k}_p) e^{-jkk \cdot r_{m\prime}} \right)^{ij},$$

where the overline refers to groups on the adaptive level. Noticeably, (7) requires an interpolation using the matrix $W$, which means that the adaptive grouping requires additional computational time. However, since this happens on the lowest level, where the number of samples is very low, the cost is low relative to the overall runtime. The concept of adaptive grouping is illustrated in Figure 1.

Finally, the application of the Spherical Harmonics Expansion (SHE), as introduced by Eibert [10], for storing the basis function patterns on the adaptive level is a vital component in any modern MLFMM implementation. The fundamental idea is the representation of the basis function patterns as a set of coefficients to the orthonormalized spherical harmonics, such that

$$\int_{S^2} f_j(r) |I - \hat{k}k| e^{-j\hat{k}k \cdot r} d^2r = \sum_{p=0}^{W} \sum_{q=-p}^{p} p_{pq} Y_{pq}(\theta, \phi),$$

The coefficients $p_{pq}$ are thus stored instead of $k$-space samples of (3). By representing the incoming patterns with another SHE, with coefficients $q_{pq}$, the integration step (5) can be replaced with

$$\hat{Z}_{j,i} = \kappa \sum_{p=0}^{W} \sum_{q=-p}^{p} (p_{pq})^i \cdot q_{pq}^i.$$

To ensure a fast convergence of the SHE in (8), the basis function patterns are expressed with cartesian rather than spherical components. An on-the-fly conversion is then done after aggregation at the adaptive level to allow the group patterns to be expressed using only the two spherical components $(\theta, \phi)$.

### IV. Results

This section deals with two testcases, a PEC sphere and a PEC corner scatterer. Throughout, we use GMRES and an overlapping near-field preconditioner [11], but stress that the results are independent of the solution method. We note that whenever we discuss total memory, we refer to the memory required to store the entire MLFMM structure but not to solve the scattering problem. Thus, we disregard the storage required.
for solvers, preconditioners and geometry, since this is not of focus in the present paper, but include everything required to perform a matrix-vector product; from basis function patterns and near-interaction matrix as well as minor temporary data such as interpolation matrices and various bookkeeping.

The discretization employed is the Higher-Order Legendre discretization [12]. The MLFMM implementation is described in detail in [13], but the discussion in the present paper focuses on practical considerations and includes a comparison with other results from the literature.

A. Sphere

The first testcase concerns the scattering from a 1 m radius PEC sphere at 8 GHz, illuminated by an $\hat{x}$-polarized plane wave propagating along $+z$. The scattered electric field in the E-plane is computed and compared to the Mie series solution. The surface of the sphere is discretized using $4^{th}$ order curved quadrilaterals. We apply MLFMM with $\beta = 3$ to the CFIE.

As a measure of accuracy, we use the relative RMS, defined as

$$\text{Relative RMS} = \sqrt{\frac{\sum_{i=1}^{N_s} (|E_{i,\text{ref}}| - |E_{i,\text{cal}}|)^2}{\sum_{i=1}^{N_s} |E_{i,\text{ref}}|^2}},$$

(10)

where $E_{i,\text{ref}}$ and $E_{i,\text{cal}}$ denotes the reference and calculated electric field at the $i^{th}$ sample point, respectively, and $N_s$ is the number of samples.

The memory usage is shown in Figure 2 as a function of the RMS for each order. The problem requires between 235200 and 940800 unknowns for the $1^{st}$ order (in the direction of the current) basis functions and between 187500 and 367500 unknowns for the $5^{th}$ order functions. We see that for higher accuracies, $3^{rd}$-$5^{th}$ order are more efficient than $2^{nd}$ order, and the first-order solution is substantially worse than the HO discretizations. However, we also clearly see that applying the modifications detailed in the previous section results in huge savings, even for a first-order discretization.

Another interesting aspect, the computational time per matrix-vector product, is shown in Figure 3, where we see the strong incentive for using HO MLFMM. Aside from the fact that the order and time per matvec are inversely proportional, we also see that for higher orders, since the number of levels are constant throughout the RMS interval, the time is independent of the requested accuracy. Further, it is evident that the modifications in the previous section only results in moderate increases in the computational time, primarily due to the adaptive grouping. From the figure, it is also evident that the RMS is unchanged by the modifications, showing that there is no loss of accuracy associated with the memory savings, aside from the effects of applying an iterative solver.

Finally, while the sphere is an important reference case due to the existence of an analytical solution, most practical cases will have to use the EFIE. Figure 4 shows the results corresponding to Figure 2 for the EFIE—note the different scale on the $y$-axis. We see that for orders higher than 1, the memory usage is essentially independent of the order, though for very accurate results, $2^{nd}$ order is somewhat less efficient.
Fig. 4. The total memory for varying RMS and polynomial order for $\beta = 3$ for EFIE.

B. Corner reflector

As a second testcase, we consider an $\hat{x}$-polarized plane wave travelling in the positive $z$-direction, illuminating a PEC corner reflector, as illustrated in Figure 5. This excellent testcase was considered by Kolundzija et al. [7] and provides a good testcase for an MLFMM implementation due to the strong interactions between well-separated parts of the scatterer. To compare our results with [7] we first consider the corner scatterer with a side length of $20\lambda$, yielding a total surface area of $600\lambda^2$, a small testcase for MLFMM but designed to allow validation against existing MoM code.

In [7], the error quantifier was the mean error, defined as

\[
\text{Mean error [dB]} = \frac{1}{N} \sum_{i=1}^{N} \Delta G_i \tag{11}
\]

where $\Delta G_i$ was defined effectively as

\[
\Delta G_i = \begin{cases} 
0, & G_i < G_{\max} - R \\
|H_i - G_i|, & G_i \geq G_{\max} - R
\end{cases} \tag{12}
\]

where $H_i$ and $G_i$ are the $i^{th}$ points of the calculated and reference bistatic radar cross sections in dB, respectively, while $G_{\max}$ is the peak value of $G_i$. The reference solution is obtained by very finely discretized MoM. $R$ is a threshold, set to 40 dB in [7]. However, we see no need to introduce a threshold and merely consider

\[
\text{Mean error [dB]} = \frac{1}{N} \sum_{i=1}^{N} |H_i - G_i| \tag{13}
\]

Having extracted the relevant results from that paper, Figure 6 is a comparison between the memory required for various orders in our implementation.

The figure shows the same overall conclusion as in the previous example — the modifications result in a very significant reduction in memory, and regardless of the choice of order, the memory required is roughly the same. In a direct comparison, we see that our implementation yields an order of magnitude better accuracy at roughly half the memory, except from order 2 from [7], where the accuracy is comparable but the memory used by our implementation reduced roughly by a factor of 5.

Finally, we consider a $60\lambda$ sidelength scatterer, a somewhat larger testcase. Here, since there is no straightforward way to achieve a reference solution, [7] just documents the required memory (5.6 GB) and number of unknowns (147987, up to 4th order). We apply a fairly fine discretization, yielding 432537 unknowns up to 4th order, using 1.15 GB memory, a factor of 5 reduction in memory. Although a direct accuracy comparison is of course unavailable, using nearly 3 times as many unknowns intuitively suggests that our accuracy is at least as good as in [7].

V. CONCLUSION

In spite of previous work suggesting otherwise, MLFMM with basis functions of orders higher than 2 can indeed be very efficient, both in terms of memory and speed, provided that extra care is taken in the implementation. We have demonstrated some additional techniques, both novel and known, and showed how their successful implementation leads to a very efficient Higher-Order MLFMM. Finally, we have compared our implementation to previously published
This paper demonstrates a significant reduction in memory at comparable and even improved accuracies.

ACKNOWLEDGMENT

This work was partially funded by ESA-ESTEC, Contract No. 4000107963/13/NL.

REFERENCES

A.6 Conference Paper II

**Reflector Antenna Analysis using Physical Optics on Graphics Processing Units**

Oscar Borries, Hans-Henrik Brandenburg Sørensen, Bernd Dammann, Erik Jørgensen, Peter Meincke, Stig Busk Sørensen and Per Christian Hansen

---

*Status*


*Bibliography*

Reflector Antenna Analysis using Physical Optics on Graphics Processing Units

Oscar Borries1,2, Hans Henrik Brandenborg Sørensen1, Bernd Dammann1, Erik Jørgensen2, Peter Meincke2, Stig Busk Sørensen2, Per Christian Hansen1
1Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark, {obor,hhbs,beda,pcha}@dtu.dk
2TICRA, Læderstræde 34, DK-1201 Copenhagen, Denmark, {ob,ej,pme,sbs}@ticra.com

Abstract—The Physical Optics approximation is a widely used asymptotic method for calculating the scattering from electrically large bodies. It requires significant computational work and little memory, and is thus well suited for application on a Graphics Processing Unit. Here, we investigate the performance of an implementation and demonstrate that while there are some implementational pitfalls, a careful implementation can result in impressive improvements.

Index Terms—Physical Optics, Computational Electromagnetics, Graphical Processing Units

I. INTRODUCTION

The Physical Optics (PO) approximation constitutes a high-frequency approximation to the induced surface current density $J_S$ on a perfectly electrically conducting scatterer $S$ given by

$$J_S = \begin{cases} 2\hat{n} \times H_i & \text{if illuminated} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $H_i$ is the incident magnetic field and $\hat{n}$ is the outward unit normal vector of $S$. PO is particularly applicable to reflector antenna systems since these are always electrically large. Finding $J_S$ from (1) requires very little work itself, but computing the far field from $J_S$ requires evaluation of a surface integral of the form

$$E_{far}(x, y, z) = \frac{e^{-jk|x - r'|}}{4\pi |r'|} \hat{r} \times \left( \int \int \left( J_S(r') e^{jkr'} dS' \right) \right), \quad (2)$$

where $k = 2\pi/\lambda$, $\lambda$ is the wavelength and $j$ is the imaginary unit, while $r = x\hat{x} + y\hat{y} + z\hat{z}$, $r' = x'\hat{x} + y'\hat{y} + z'\hat{z}$ denote observation and integration points, respectively. This integral takes up considerable computational resources for large reflectors.

In particular, for PO applied to a dual reflector setup, calculating the incident magnetic field on the main reflector due to the surface current distribution on the sub-reflector typically constitutes the vast majority of the computational load. The surface integral to be evaluated is of the form

$$H_i(x, y, z) = -\frac{1}{4\pi} \int \int \left( \hat{R} \times J_S(x', y', z') \right) \frac{1 + jk|\mathbf{R}|}{|\mathbf{R}|^2} e^{-jk|\mathbf{R}|} dS', \quad (3)$$

where $\mathbf{R} = r - r' = (x - x')\hat{x} + (y - y')\hat{y} + (z - z')\hat{z}$ and $J_S$ is the surface current density on the subreflector.

II. ALGORITHM

Defining a set of field points $\mathcal{F}$ containing $N$ points and a current distribution $\mathcal{J}$ discretized in $M$ points, the rough outline of the code is given below. The operator $\mathcal{V}(\mathcal{J})$ refers to the integrand in (3), and $w_j$ are the integration weights.

Algorithm 1 Pseudocode for implementing (3).

```
Zero \mathcal{F}
for i = 1, N do
    \mathcal{F}_i = \mathcal{F}_i + \mathcal{V}(\mathcal{J}_j)w_j
end for
```

The inner part of this algorithm is evaluated $MN$ times and requires an array of size $M + N$. Further, since the operator $\mathcal{V}(\mathcal{J})$ requires $O(MN)$ operations and $M$ and $N$ scale as $O(\lambda^{-2})$, the calculation of $\mathcal{F}$ requires $O(\lambda^{-4})$ operations. This suggests why the calculation of (3) is a significant computational load for electrically large $S$.

III. PERFORMANCE ON A GPU

To implement the code on a GPU (Graphics Processing Unit), we use the CUDA framework [1] developed by Nvidia. A first implementation is made by porting the existing code from the GRASP software package [2]. As a test case the scattered field from a $20\, \text{m}$ square plate, excited by a plane wave at $3\, \text{GHz}$, at normal incidence and polarized parallel to two of the sides, is computed.

The performance, relative to the highly optimized GRASP code, is given in columns 2 and 3 in Table I. The timings are done on a quad-core $2.9\, \text{GHz}$ Intel i5, and an Nvidia GTX 670 Mini-ITX. The retail prices of just the processor and the graphics card are roughly equivalent. The performance in double precision is somewhat disappointing, only a factor of $3$. Although the GPU is faster than the reference implementation, it might not seem enough to warrant the additional coding needed.
To discover why, we apply a High-Performance Computing benchmarking suite currently under development at the Technical University of Denmark [3] to investigate the behaviour of the GPU in detail. The key issue here, the number of floating point operations per second (flops), is shown in Figure 1. From this figure, we see an extreme discrepancy between the performance in single and double precision. Provided that there is sufficient work for the entire card, the single precision performance is roughly 2.2 teraflops while the double precision performance is only around 150 gigaflops.

Since the real performance advantage of low-end GPUs thus is in the single precision domain, we implement the code in single precision in CUDA to see the difference in performance. The fourth column in Table I demonstrates the results.

<table>
<thead>
<tr>
<th>$M = N = 200^2$</th>
<th>GRASP</th>
<th>GPU double</th>
<th>GPU single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time [s]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.7</td>
<td>3.2</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>$300^2$</td>
<td>44.4</td>
<td>14.0</td>
<td>2.4</td>
</tr>
<tr>
<td>$400^2$</td>
<td>142</td>
<td>42.6</td>
<td>7.1</td>
</tr>
<tr>
<td>$500^2$</td>
<td>355</td>
<td>101</td>
<td>16.4</td>
</tr>
<tr>
<td>$600^2$</td>
<td>707</td>
<td>210</td>
<td>33.8</td>
</tr>
</tbody>
</table>

**TABLE I**

The performance of the implementation, both in GRASP and on the low-end GPU, both in double and single precision.

The applicability of single precision in Physical Optics thus clearly depends on both the required accuracy and the case at hand. For 80 dB accuracy in a dual reflector setup, the distance between the sub- and main reflector thus cannot be more than 100 $\lambda$, a fairly low number in many cases. However, if 60 dB accuracy can be accepted, the distance cannot be more than 1000$\lambda$, a much less restrictive number.

Finally, we stress that these considerations are only applicable for the near-field scenario in (3). For the far-field computations (2), the exponential function has much smaller arguments, so there is no loss of accuracy involved in using single precision here.

V. REALISTIC CONFIGURATIONS

To demonstrate the performance on realistic scatterers, we consider here two setups. First, a Cassegrain dual reflector antenna, a typical benchmark for a Physical Optics code, designed from textbook formulas [2]. Second, we move to an actual application of a high-gain reflector antenna mounted on a satellite in low orbit, taken from an ongoing ESA project [5].

**A. Cassegrain dual reflector**

The setup is illustrated in Figure 2, with the relevant data shown in Table II.

The accuracy criterion is set to 60 dB, yielding a maximum distance of approximately 1000$\lambda$, sufficient for the present case to be analyzed in single precision. The electric far-field from the system is sampled on a $\theta \phi$-grid at a sample spacing of \( \frac{\lambda}{64} \) in $\theta$, where $D$ is the aperture diameter of the antenna, between $-10^\circ$ and $10^\circ$ in $\theta$. Further, the sampling in $\phi$ is done at $5^\circ$ increments between 0 and $90^\circ$. This yields a total of 3350 x 19 points.

The time spent in each task is shown in Table III, where we see that the GPU code is more than 7 times faster than the reference GRASP implementation. From the table, we

and thus the accuracy of $\cos$ and $\sin$ for a given architecture. For Nvidia CUDA, the accuracy of trigonometric single precision special functions is 2 ULP (Units in Last Place) throughout the range $[-2\pi;2\pi]$, meaning that the number of contaminated digits is roughly $\log_2(2) = 1$ in IEEE single precision [4]. This means that the relative error is approximately $10^{-6}$.

Therefore, to find the precision for large arguments, we consider

$$\log_{10} \left( \frac{k\lambda}{2\pi} \right) = \zeta$$

(5)

Thus, the relative error will be $10^{-(6-\zeta)}$.

In GRASP, the natural error criterion is a specification of the error level, i.e. an 80 dB criterion suggests that the power $|E|^2$ is accurate to 80 dB below peak. This criterion corresponds to $\log_{10} \left( \frac{80}{T} \right) = 4$ digits of accuracy, suggesting that $\zeta$ can be no more than 2. This means that we require

$$|R| < \frac{2\pi 10^6}{k} = 10^6\lambda$$

(6)

The accuracy criterion is set to 60 dB, yielding a maximum distance of approximately 1000$\lambda$, sufficient for the present case to be analyzed in single precision. The electric far-field from the system is sampled on a $\theta \phi$-grid at a sample spacing of $\frac{1}{64}$ in $\theta$, where $D$ is the aperture diameter of the antenna, between $-10^\circ$ and $10^\circ$ in $\theta$. Further, the sampling in $\phi$ is done at $5^\circ$ increments between 0 and $90^\circ$. This yields a total of 3350 x 19 points.

The time spent in each task is shown in Table III, where we see that the GPU code is more than 7 times faster than the reference GRASP implementation. From the table, we
also see that particularly the far-field computation is much faster on the GPU compared to the near-field calculations. The relatively slow computation of the near-field is primarily due to the use of auto-convergence, which uses repeated runs with a small number of field points to determine the number of current points on the reflectors. This in turn causes sub-optimal occupancy of the GPU, resulting in a smaller speed-up than otherwise achievable. We stress that for repeated runs, the auto-convergence is only done once, and thus the relative gain will be much larger. This also explains why the speed-up is poorer than expected from Table I, since in the latter case, the number of field points is large enough to ensure sufficient occupancy.

Another potential reason for the near-field calculations to yield a poorer speed-up is due to the implementation of trigonometric functions on the GPU. According to the documentation, costly argument reduction techniques are applied if the argument is larger than 48039. In this case, another code path is used, requiring significantly higher register use, which in turn causes poorer performance - the CPU code does not have such issues. While the argument in the present case is lower than 48039, it is still worth noting.

In conclusion, the speed-up is a respectable 6.2 for the large PO run, while the small task of performing PO on the subreflector from the feed is so fast that the overhead from transferring memory to and from the GPU results in a slowdown, resulting in a speed-up of 0.3.

The scattered field is shown in Figures 3–4 for the co- and cross-polar components, respectively. Although hard to note from the fast oscillations, the red and blue lines are essentially identical in the co-polar component. For the much lower cross-polar component, which is nearly 100 dB below peak, the effects of single precision results in a relative difference of 3%, acceptable for the low levels. Interestingly, while this cross-polar component should be numerically zero, due to a combination of integration error and numerical noise, a distinct pattern is computed. It is important to note that both the GPU and GRASP reference solution, though yielding slightly different values, still results in roughly the same pattern, including the artificial asymmetry at $\theta \approx 3^\circ$. This further confirms the applicability of the GPU solution, since even far below the requested accuracy, it still provides the same overall pattern as the reference solution.

B. Low-Orbit Reflector

To look into another case that is particularly suitable for acceleration from a GPU based implementation, we consider a high-gain antenna with a torus reflector mounted on a low-orbit satellite, designed for ocean surveillance [5]. The information for the system is described in Table IV and a figure illustrating its operating conditions and geometry is shown in Figures 5–6.

### Table II
**System data for the Cassegrain dual reflector setup.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelength</td>
<td>1 cm</td>
</tr>
<tr>
<td>Main reflector aperture</td>
<td>12 m</td>
</tr>
<tr>
<td>Main reflector f/D</td>
<td>0.4</td>
</tr>
<tr>
<td>Subreflector eccentricity</td>
<td>1.5</td>
</tr>
<tr>
<td>Subreflector diameter</td>
<td>104 cm</td>
</tr>
</tbody>
</table>

### Table III
**Timings from a complete Physical Optics analysis of the Cassegrain dual reflector setup.**

<table>
<thead>
<tr>
<th>Task</th>
<th>GRASP [s]</th>
<th>GPU single [s]</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO on sub from feed</td>
<td>0.9</td>
<td>2.8</td>
<td>0.3</td>
</tr>
<tr>
<td>PO on main from sub</td>
<td>293</td>
<td>47</td>
<td>6.2</td>
</tr>
<tr>
<td>Far-field from system</td>
<td>88</td>
<td>53</td>
<td>26.6</td>
</tr>
<tr>
<td>Total</td>
<td>382</td>
<td>53</td>
<td>7.2</td>
</tr>
</tbody>
</table>

### Figure 2
Illustration of the Cassegrain dual reflector setup.

### Figure 3
The total electric field from the Cassegrain dual reflector, calculated with both the reference GRASP implementation and the GPU single precision implementation discussed here. Co-polar component at $\phi = 0$. For clarity, only the interval $0-5^\circ$ is shown.

### Figure 4
Figure showing the co-polar component at $\phi = 0$. The red and blue lines are essentially identical, indicating negligible difference due to single precision.
Fig. 4. The total electric field from the Cassegrain dual reflector, calculated with both the reference GRASP implementation and the GPU single precision implementation discussed here. Cross-polar component at $\phi = 90^\circ$. Note the lack of symmetry around $\theta = 0$, indicating that both the GRASP and GPU implementations are affected by inaccuracies.

![Fig. 4](image_url)

**Table IV**

<table>
<thead>
<tr>
<th>Frequency</th>
<th>$10^3$ GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projected reflector aperture</td>
<td>5 m</td>
</tr>
<tr>
<td>Reflector f/D</td>
<td>1</td>
</tr>
<tr>
<td>Clearance</td>
<td>1 m</td>
</tr>
<tr>
<td>Operating altitude</td>
<td>817 km</td>
</tr>
</tbody>
</table>

**System data for the low-orbit reflector.**

Due to the low orbit, the far-field pattern needs to be tabulated for the complete angular region from 0 to $60^\circ$ relative to boresight, and the high-gain, large aperture of the antenna requires a closely spaced sampling. This results in a far-field grid of 1130913 field points when converted to a circular region in a $uv$-grid. The result is shown in Figure 7.

In this case, several runs have been made, so the number of current points on the reflector has been determined in advance. Thus, the only relevant task is the determination of the large and closely sampled far-fields, which are further complicated by the somewhat complicated geometry of the reflector, requiring a densely sampled PO grid, yielding 1122328 integration points. The closely sampled far-field required nearly 4 hours to evaluate in GRASP, while it took a little under 10 minutes on the GPU. The total speedup is thus roughly a factor of 24.

**VI. Conclusion**

The demanding task of computing the radiated fields from an electrically large current distribution has been converted to a low-end GPU and key performance issues have been identified. Taking into account those issues, several testcases have demonstrated very significant performance gains, even when considered relative to industry-standard code, for comparable accuracies.

![Fig. 5](image_url)

**Fig. 5.** Geometry for an antenna in low orbit. The reflected rays determine the boresight, while the thin line illustrates the ray at $\theta \approx 60^\circ$ relative to boresight.

![Fig. 6](image_url)

**Fig. 6.** Illustration of a torus reflector illuminated by three separate feeds, used as a testcase for the GPU implementation.

![Fig. 7](image_url)

**Fig. 7.** An example of the result achieved for the low-orbit antenna, co-polar component. We note the small high-gain region at $u \approx -0.8$, $v = 0$. The limits for the plot are approximately $\pm 0.89$ in both $u$ and $v$. 
REFERENCES

[1] NVIDIA Corp. CUDA 5.5.
A.7 Conference Paper III

Analysis of Electrically Large Antennas using Fast Physical Optics

Oscar Borries, Hans-Henrik Viskum, Peter Meincke, Erik Jørgensen, Per Christian Hansen and Carsten H. Schmidt

Status

Submitted: 9th European Conference on Antennas & Propagation (EuCAP), Lisbon, April 2015.

Bibliography

Analysis of Electrically Large Antennas using Fast Physical Optics

Oscar Borries1,2, Hans-Henrik Viskum2, Peter Meincke2, Erik Jørgensen2, Per Christian Hansen1, Carsten H. Schmidt3
1Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark, {opbo,pcha}@dtu.dk
2TICRA, Læderstræde 34, DK-1201 Copenhagen, Denmark, {ob,hhv,pme,ej}@ticra.com
3Airbus Defence & Space, Munich, Germany, carsten.schmidt@astrium.eads.net

Abstract—The design of electrically large antennas can be a significant challenge for computational electromagnetics (CEM) tools, particularly during the final stages of the design process where there are strict requirements for the accuracy. In the present paper, we consider the use of a newly developed accelerated Physical Optics (Fast-PO) and show that this approach allows for a timely and accurate solution of realistic designs. Several examples, ranging from canonical tests of the scaling of the method against the wavelength to real-life applications, illustrate the performance of the approach in practice.

Index Terms—Physical Optics, Reflector antennas, Fast methods.

I. INTRODUCTION

Computational electromagnetics (CEM) tools have long since established themselves as a vital part of the design approach for modern antennas. A range of approaches exist for modelling the behaviour of antennas of varying size and complexity, and choosing the right tool for the job is very critical in order to achieve an acceptable runtime while maintaining an adequate accuracy.

When considering electrically large reflector antennas, for applications such as space or telecommunications, achieving both low runtimes and low error levels typically involve the use of Physical Optics (PO). PO is an asymptotic method where the surface current density is approximated as if the surface at the point of observation is replaced by its tangent plane. This approximation becomes increasingly accurate with increasing frequency $f$, but unfortunately the computation time of PO behaves as $O(f^4)$. In other words, for electrically very large antennas, even PO can become prohibitively slow in practice.

The time-consuming part of PO is the evaluation of the radiated field from a surface current distribution, in which each observation point requires evaluation of a two-dimensional integral. Thus, speeding up PO requires speeding up the evaluation of fields from current distributions. The speedup of this subtask is also beneficial in other methods, such as Radar Cross Section (RCS) evaluation and Method of Moments (MoM), where it is used both to evaluate the right-hand side and to find the field radiated by the computed currents.

Several research groups have considered the acceleration of these tasks. An early version is the fast far-field approximation (FaFFA) [1] and its multi-level variants, which represent the behaviour of the field outside a sphere of radius $2D^2/\lambda$ as a sum of plane waves, one from each subgroup of current points, where $D$ is the diameter of the group. Since the Rayleigh distance $2D^2/\lambda$ is only an approximate value of the distance at which the far-field behaviour of the current distribution begins, the error level is not controllable when applying FaFFA.

An alternative was presented in [2] and further expanded in [3], [4], [5]. This algorithm, which we will refer to as Fast-PO, uses interpolation as the main tool to allow the computation of radiated fields on a reduced grid and then interpolate to the requested observation points. Since direct interpolation would not yield significant savings, the interpolation is performed on a phase-compensated field, which is much easier to interpolate, and the original phase is then restored. The accuracy is mainly controlled by increasing the order of the polynomial interpolant. However, choosing the parameters to achieve a specified accuracy is not straightforward for a general scatterer. Further, achieving the accuracy levels required for some reflector antenna applications is difficult using polynomial interpolants.

In the present paper, we briefly review the Fast-PO algorithm, highlighting its strengths and weaknesses. Then, we focus on the performance on an improved version when applied to a wide range of test cases.

II. FAST PHYSICAL OPTICS

PO constitutes a high-frequency approximation to the induced surface current density $J_S$ on a perfectly electrically conducting scatterer $S$ given by

$$J_S = \begin{cases} 2\hat{n} \times H_i & \text{if illuminated} \\ 0 & \text{otherwise} \end{cases}$$

where $H_i$ is the incident magnetic field and $\hat{n}$ is the outward unit normal vector of $S$. Finding $J_S$ from (1) requires very little work itself, but computing the field from $J_S$, or finding $H_i$ from a general surface current density, requires evaluation of an integral which may involve a significant amount of computational work. In the present context, the main applications are computing the electric far-field from $J_S$ or computing the magnetic near-field from another surface to form $H_i$ in (1). In the following, we consider each of these applications separately, since the involved integrands possess very different properties.
A. Far-field Fast-PO

Computing the electric far-field in the direction \((\theta, \phi)\) from a surface current distribution \(J_S\) is done by computing

\[
E_{\text{far}}(\theta, \phi) = \frac{e^{-jk|r|}}{|r|} \frac{jk}{4\pi} \hat{k} \times \left[ \hat{k} \times \int J_S(r') e^{jk \cdot r'} dS' \right], \tag{2}
\]

where \(k = 2\pi/\lambda\), \(\lambda\) is the wavelength and \(j\) is the imaginary unit, while the wave-vector \(\hat{k} = \sin \theta \cos \phi \hat{x} + \sin \theta \sin \phi \hat{y} + \cos \theta \hat{z}\) and the integration point is \(r' = x'\hat{x} + y'\hat{y} + z'\hat{z}\).

Since (2) is evaluated for \(N_{\text{obs}}\) observation points, each requiring a summation over all \(N_{\text{int}}\) integration points, the complexity is \(O(N_{\text{obs}}N_{\text{int}})\). Since \(N_{\text{obs}}\) and \(N_{\text{int}}\) typically scale as \(O(f^2)\), the total scaling is \(O(f^4)\). The value of \(N_{\text{obs}}\), as well as \(N_{\text{int}}\), is governed by the largest value of \(|r'|\). While PO can be sped up using Graphics Processing Units [6], this does not reduce the scaling against the frequency, and thus for very large problems, another approach is necessary.

The Fast Physical Optics (Fast-PO) algorithm allows a significant reduction in the scaling by using a multi-level approach. First, we apply a grouping (such as an Octree [7]) to the original phase using low-order polynomials, which, along with restoration of the radial component, although [10] to some extent avoids the radial component by interpolating on the surface of the structure) becomes more manageable. Since the Green’s function now contains the norm of a vector, rather than the dot product between two vectors as was the case for the far-field integral (2), the phase-compensation used for the far-field problem is no longer sufficient. Using a multi-level grouping of the source points, [9], [10] have considered the use of a phase-and-amplitude compensated field using

\[
V(r) = s(r, r') e^{jk \cdot r} H_i(r), \tag{6}
\]

where

\[
s(r, r') = \sqrt{|r - r'|^2 + (D_N/2)^2} \tag{7}
\]

in which \(r'\) is the center of the group in which the source point \(r'\) is located, and \(D_N\) is the diameter of that group at the level at which (6) is applied. With these compensations, the interpolation (which is performed both for the angular and the radial component, although [10] to some extent avoids the radial component by interpolating on the surface of the structure) becomes more manageable.

We stress that the accuracy of interpolation for near-field Fast-PO as presented in [9] will significantly depend upon the distance between the source and integration domains relative to the aperture of the source. In other words, the closer we move into the near-field, the more irregularly the field will behave (even with the phase-and-amplitude compensation), and thus the accuracy will suffer. This is treated in [9] by the introduction of a lower limit \(\Omega_R > 1\), such that observation points that are located within the distance

\[
\frac{D}{2} \Omega_R, \tag{8}
\]

the literature require an oversampling parameter to account for inaccuracies in the interpolation procedure, and no exact rule exists for selecting this parameter to achieve a specified accuracy, which means that one cannot be sure that a specified accuracy has been achieved.

To overcome these challenges, we have improved the existing Fast-PO implementation for far-field problems, resulting in a significantly faster routine. In particular, this allows the combination of Fast-PO with the highly efficient PO integration rules used in TICRA’s flagship product GRASP. Section III demonstrates the speed and accuracy relative to the direct PO in GRASP on a few examples.

B. Near-field PO

Calculating the magnetic field incident on a surface due to the surface current distribution on another surface constitutes a point on the unit sphere, and \(v'\) is the vector from the center of the group on the finest level to the integration point. Since \(|v'|\) is much smaller than \(|r'|\) in (2), we can settle for a much coarser sampling of the field, thereby reducing \(N_{\text{obs}}\) significantly. Having sampled \(V\) at a low number of output points, we can then apply interpolation to get the output at the original \(N_{\text{obs}}\) points, and subsequently restore the original plane. Thus, we can recover \(E_{\text{far}}\) using

\[
E_{\text{far}} = \int e^{jk \cdot (r' - v')} \mathcal{W}\{V\} dS'. \tag{4}
\]

Here, \(\mathcal{W}\) is an interpolation operator in \((\theta, \phi)\), usually based on low-order polynomials, which, along with restoration of the original phase using \(e^{jk \cdot (r' - v')}\), allows the required field to be computed much faster than by direct evaluation of (2). Since \(V\) varies much slower than \(E\) as a function of direction due to the fact that \(|v'| >> |r'|\), \(V\) is typically dubbed the phase-compensated field.

It is worth noting that in the work previously presented on far-field Fast-PO [5], [8], the accuracy has been fairly modest, with an error higher than 0.1% Relative RMS, which corresponds roughly to 1 dB deviation at 60 dB below peak. However, even 0.1% would generally be too large an error for use in reflector antenna problems, particularly in the final stages of a design. Further, the sampling rules presented in
from the center of the source points, with \( D \) being the diameter of the minimum sphere containing all source points, will be treated via direct application of (5). Again, choosing \( \Omega_2 \) along with the oversampling ratios to achieve a pre-specified upper bound on the error is not straightforward. Further, we note that the limit in (8) depends linearly on \( D \), while the region in which the near-field behaves irregularly depends quadratically on \( D \), meaning that the errors as a function of \( \Omega_2 \) discussed in [9] will not necessarily be applicable for larger structures.

Our numerical experiments have revealed that even with specialized radial interpolation routines based on the near-field behaviour of an electromagnetic field [11, App. A], rather than simple polynomial interpolation, the interpolation becomes incontrollable as we move far inside the near-field of the radiating antenna. This does not lead to errors in using the Fast-PO as presented in the current literature, since it is handled by switching to direct integration, but it means that Fast-PO will not accelerate the computation. While this limitation on the distance between source and observation regions might be acceptable in some scenarios, e.g. in dual reflector systems for sub-to-main interactions where the main reflector is sometimes located outside the limit specified by (8), it renders the approach useless for e.g. main-to-sub interactions (such as computing the blockage by the subreflector). For other applications such as a Compact Antenna Test Range (CATR), where the sub- and main reflectors are often of comparable size and located quite close, the Fast-PO as presented in the literature is simply not applicable at all.

To overcome these bottlenecks, we have developed a new algorithm that combines the speed of FaFWA with the thorough sampling rules and phase-compensation of Fast-PO, resulting in superior accuracy and much lower runtimes, particularly for large problems, with the limitation on the distance between source and observation points being on the order of \( 1\lambda \) before switching to direct PO. Section III will demonstrate the performance for some practical scenarios.

### III. EXAMPLES

In this section, we examine the accuracy and speed of the algorithms (near- and far-field) in comparison with regular PO.

The error is computed as the Relative RMS

\[
\text{Relative RMS} = \sqrt{\frac{\sum_{i=1}^{N} |E_{i,\text{direct}} - E_{i,\text{fast}}|^2}{\sum_{i=1}^{N} |E_{i,\text{direct}}|^2}}, \quad (9)
\]

where \( E_{i,\text{direct}} \) and \( E_{i,\text{fast}} \) denote the direct PO (from (2) or (5)) and the Fast-PO electric field at the \( i \)-th sample point, respectively, and \( N \) is the number of samples. We stress that the reference field is not the true physical field, but rather the field as computed by direct integration of (2) or (5). As such, the accuracy is independent of e.g. the choice of integration rule for the surface integrals, since this rule is the same for direct PO and Fast-PO. Thus, the Relative RMS is only affected by the error made by applying Fast-PO instead of direct PO and consequently, in theory, a Relative RMS of 0 is possible if perfect interpolation were possible and no rounding errors were present.

#### A. Scattering by a plate

As an illustration of the performance of the algorithms for near- and far-field radiation on a simple geometry, we consider the scattering from a \( 1 \times 1 \times 1 \) m plate, illuminated by a plane wave at 45° incidence at varying frequencies. The configuration is illustrated in Figure 1.

For the near-field setup, we evaluate the field on a half-sphere, with a radius of 0.8 m and origin at the center of the plate. The points are distributed equidistantly in a \( \theta \phi \) grid, with the total number of points \( N_{\text{obs}} \) determined based on the wavelength \( \lambda \) (in metres) of the incident field, such that

\[
N_{\text{obs}} = \frac{25}{\lambda^2}, \quad (10)
\]

while the number of integration points is determined based on an auto-convergence procedure, requiring \(-80\) dB accuracy [12]. The time required for auto-convergence is not included in the timings. We note that the observation sphere is located close to the plate, and with the frequency being varied between 12 GHz and 192 GHz, the observation points are located deep within the reactive near-field region, typically taken to be the region for which \( |R| < 0.62 \sqrt{D^2/\lambda} \) in (5).

By adjusting the frequency, we get the timings illustrated in Figure 2. The Relative RMS achieved varies slightly between the frequencies, but is not above the specified limit of \( 10^{-4} \). Fitting a linear curve to the timings, we get close to the expected \( O(f^4) \) behaviour from the direct PO routine, while the Fast-PO timings scale slightly better than \( O(f^2) \). If we increased the frequency further, we would expect the scaling to eventually reach the \( O(f^2 \log f^2) \) scaling suggested in the literature.

For the far-field algorithm, we choose a very similar setup. The field is evaluated in a \( \theta \phi \)-grid with \( \theta \in [-\pi/2, \pi/2], \phi \in [0, 2\pi[, \) and the number of points \( N_{\text{obs}} \) on the grid is chosen using (10) while the number of integration points are chosen using auto-convergence. We get the results illustrated in Figure 3. Once again, the Relative RMS achieved varies slightly between the frequencies, but is far below the limit \( 10^{-4} \).

#### B. Torus

Another example is the torus antenna shown in Figure 4, designed by TICRA for a specific remote-sensing application.
Fig. 2. Timings of our Fast-PO implementation for near fields relative to the direct PO implementation in GRASP 10.3. The Relative RMS is below $10^{-4}$ for all frequencies.

Fig. 3. Timings of our Fast-PO implementation for far fields relative to the direct PO implementation in GRASP 10.3. The Relative RMS is below $10^{-4}$ for all frequencies.

[13]. It is generated by rotating a parabola around a tilted axis, and has the characteristics that it minimizes the scan loss for beams generated by feed elements located on a curve intersecting the focal point. It is important to evaluate the pattern from the antenna over a very large angular region since the application is a radiometer on a low-orbit satellite from which the earth subtends an angle of more than 60°.

The observation points are distributed in a $3201 \times 3201$ $uv$ grid covering the region of earth visible from the satellite, yielding a total of $N_{\text{obs}} = 10246401$ far-field observation points.

We use the auto-convergence to determine the number of integration points on the surface, which yields $N_{\text{int}} = 607127$. We do not include PTD, since the auto-convergence finds that the PTD contribution is not relevant at the specified (-80 dB) accuracy. This is due to the low edge illumination from the array. With GRASP 10.3, we find the solution in 1:27 H, while the Fast PO algorithm requires 4:33 minutes, a speedup of almost a factor of 20. The Relative RMS in the co-polar component is $2 \cdot 10^{-5}$, while in the cross-polar component it is slightly higher at $5 \cdot 10^{-5}$, both of which are significantly below the requested $10^{-4}$ error.

C. CATR

As a final example, we consider the compensated compact range from Airbus Defence & Space, the CCR 75/60, which was also investigated in [14]. The main reflector is 7.5 m times 6.0 m, resulting in a quiet zone (QZ) with a diameter of 5 m, in which the field has the same characteristics as a plane wave. The setup is illustrated in Figure 5. The simulation is carried out by applying the direct path through the system; the feed illuminates the subreflector, which in turn illuminates the main reflector, and the electric field from the main reflector is computed in the QZ. The vast majority of the computing time is spent on computing the illumination of the main reflector, due to the large amount of current points on both the sub- and main reflectors, including the serrations. In particular, the effects of the serrations as shown in Figure 6 are quite challenging to accurately model using PO, requiring a large number of current points. We note that the number of current points is again found using an auto-convergence procedure. We do not use PTD since its contribution in the QZ is very low.

The PO algorithm in GRASP 10.3 requires 3:30 hours to compute the interaction between the sub and main reflector at 6 GHz. Using Fast-PO, we tighten our requirements to the accuracy due to the highly sensitive application, requesting a relative error of $10^{-5}$. The computation requires 1:20 minutes using Fast PO, resulting in a speed-up of a factor of 157 with a Relative RMS less than $4 \cdot 10^{-6}$. The performance in the QZ
**IV. CONCLUSION**

The examples show that our implementation of accelerated Physical Optics (Fast-PO), involving more efficient interpolation for the far field Fast-PO and a new approach for near field Fast-PO, provides extreme speedups while maintaining the very low error level required for the final stages of a design. Details of the algorithm, including the implementation of the Physical Theory of Diffraction (PTD) contribution into the acceleration scheme, are still under development. However, the performance of the current algorithm allows a reduction in time by a factor of 10 to 100, depending on the application. For a practical application involving the design of a Compact Antenna Test Range, Fast-PO offers a reduction in time from 3.5 hours to 1:20 minutes, thus facilitating a much more efficient design process and even opening up the possibility of allowing small-scale optimization.

**REFERENCES**


In spite of the low memory implementation of MLFMM detailed in the present thesis, some scatterers are still too electrically large to be handled by full-wave methods. For those scatterers, if they are relatively smooth, the method of Physical Optics (PO) might be an alternative. Physical Optics has existed for a long time, and as the name implies, it originated from the optics community, where it serves as a more accurate approximation than the ray-based methods to the behaviour of an electromagnetic field. However, PO is still an approximation, and rather than attempt to fully solve the Helmholtz equation as MoM does, it assumes that the scatterer can locally be approximated by an infinite tangent plane. For electrically large reflectors and other large and smooth structures, this turns out to be a sufficiently accurate approximation. Applying PO, the computational bottleneck is the evaluation of the scattered/incident field from/on the scatterer.

However, while PO only requires $O(f^2)$ memory, it requires $O(f^4)$ computational time and can therefore be very time consuming, although it is typically much faster than MoM. Several attempts have been made at developing a “fast” version by reducing the frequency scaling of PO, much like MLFMM reduces the frequency scaling of MoM.

In this appendix, we will provide a short introduction to the Physical Optics approach, and then turn our focus to Fast-PO, describing the current state-of-the-
art and providing results from our implementation of a vastly improved Fast-PO algorithm compared to the algorithms in the literature. As mentioned previously, Fast-PO corresponds to accelerating the computation of scattered/incident electromagnetic fields, and thus a more appropriate title for this appendix would have been Fast Field Computation. However, we have chosen to stay with the nomenclature of the community.

Unfortunately, the details on the improvements made to the Fast-PO algorithm as part of the present work have been made confidential by TICRA, and thus will not be presented here. This is primarily due to two reasons:

1. The improvements significantly extend the applicability of Fast-PO to much larger problems, and much better precision, than the current state-of-the-art.

2. The evaluation of scattered fields, primarily but not exclusively as part of a PO analysis, is an area where TICRA is already far ahead of the competition, and thus a key part of TICRA’s business.

Therefore, TICRA does not wish to aid their competitors by publishing details that will allow competitors to implement a much faster PO solver.

We mention that as part of the thesis work, an implementation of Physical Optics on Graphics Processing Units (GPUs) was completed and detailed in [C2]. This approach allows a strong speedup compared to typical CPU implementations, and provided some interesting challenges, but it does not reduce the scaling of the method against the frequency. Further, since the implementation is detailed in [C2], we do not discuss it further in the thesis.

### B.1 Physical Optics

The Physical Optics (PO) approximation constitutes a high-frequency approximation to the induced surface current density $J_S$ on a perfectly electrically conducting scatterer $S$ given by

$$J_S = \begin{cases} 2\hat{n} \times H_i & \text{if illuminated} \\ 0 & \text{otherwise} \end{cases} \quad (B.1)$$

where $H_i$ is the incident magnetic field and $\hat{n}$ is the outward unit normal vector on $S$. Finding $J_S$ from (B.1) requires very little work itself, but computing the
Figure B.1: An illustration of the stages involved in calculating the far-field behaviour of a 2D dual reflector system using Physical Optics. First (a), the illumination of the subreflector by the feed is calculated in a number of observation points (in blue). These points then become integration points for the illumination of the main reflector in a number of observation points (in red) using (B.4). Finally, the far-field behaviour of both sub- and main reflector is computed in the green directions using (B.3). With each line representing an interaction, it is evident that for multiple scatterers or multiple bounces between pairs of scatterers, the number of evaluations of the kernels in (B.4) or (B.3) can become very large.

field from $J_S$, or finding $H_i$ from a general surface current density, requires evaluation of an integral which may involve a significant amount of computational work. In the present context, the main applications are computing the electric far-field from $J_S$ or computing the magnetic near-field from another surface to form $H_i$ in (B.1). In the following, we consider each of these applications separately, since the involved integrands possess very different properties. Figure B.1 illustrates the process of applying PO to a dual reflector system.

B.1.1 Far-field PO

The electric far field can be defined as

$$E_{\text{far}} = \lim_{r \to \infty} (E(r) kr e^{jkr}),$$  \hspace{1cm} (B.2)
where \( r = \|r\| \) is the distance to the observation point. Computing the far field from \( \mathbf{J}_S \) requires evaluation of a surface integral of the form:

\[
\mathbf{E}_{\text{far}}(\hat{\mathbf{r}}) = \frac{j\eta_0}{4\pi} \hat{\mathbf{r}} \times \left[ \hat{\mathbf{r}} \times \int S(\mathbf{r'}) e^{jk\hat{\mathbf{r}} \cdot \mathbf{r'}} dS' \right],
\]

where \( k = \frac{2\pi}{\lambda} \), \( \lambda \) is the wavelength and \( j \) is the imaginary unit, while \( \hat{\mathbf{r}} \) and \( \mathbf{r}' \) denote the observation direction and integration point, respectively.

A coarse rule-of-thumb sample spacing of the far-field is \( \frac{1}{4} \lambda D \) radians, where \( D \) is the aperture diameter, i.e., the diameter of the smallest possible sphere that encloses all source points. Based on this sampling, one can perform cubic interpolation and expect to get a close approximation to the true pattern. Using the sampling spacing \( \frac{1}{8} \lambda D \) radians means that linear interpolation suffices, and therefore many antenna engineers will choose the sampling spacing somewhere in between \( \lambda/(8D) \) and \( \lambda/(4D) \). In many applications, a grid is needed in \( \theta \) and \( \phi \), and thus, the sampling density is between \( \left[ \frac{1}{64}, \frac{1}{16} \right] \lambda^2/D^2 \). For computing grids, the number of observation points \( N_{\text{obs}} \) therefore scales as \( O(f^2) \) since \( f \) scales as \( O(\lambda^{-1}) \).

The number of integration points, \( N_{\text{int}} \), is much more complicated to parametrize. A wide range of approaches exists for integrating the surface current density. The standard approach is to mesh the surface \( S \) using flat triangles, and then integrate independently on each triangle, using, e.g., a Gauss-Legendre rule. This yields a very general approach and can be applied to any meshable surface, but yields a large number of integration points and does not take the distribution of the observation points into account.

In TICRAs product GRASP, integration points are distributed over the entire surface using specialized rules for the common reflector rim shapes, i.e., paraboloids, ellipsoids and so on. Further, an auto-convergence procedure is applied, which finds the necessary integration point density to compute the field in the required observation points to a specified accuracy. If, e.g., only the main lobe of a properly designed reflector is evaluated, almost all contributions will be in phase and thus the kernel of (B.3) will not oscillate vehemently, leading to a low \( N_{\text{int}} \). For more details, see Pontoppidan [136]. However, regardless of the specific approach, the surface area scales as \( \lambda^{-2} \), and therefore \( N_{\text{int}} \) has an asymptotic complexity of \( O(f^2) \).

### B.1.2 Near-field PO

Calculating the magnetic field incident on a surface due to the surface current distribution on another surface constitutes the majority of the computational
load in many PO tasks. The surface integral to be evaluated is of the form

\[ H_i(r) = -\frac{1}{4\pi} \int \left( \hat{\mathbf{R}} \times J_S(r') \right) \frac{1 + jk|\mathbf{R}|}{|\mathbf{R}|^2} e^{-jk|\mathbf{R}|} dS', \]  

(B.4)

where \( \mathbf{R} = \mathbf{r} - \mathbf{r}' = (x - x')\hat{x} + (y - y')\hat{y} + (z - z')\hat{z} \) and \( J_S \) is the surface current density on the illuminating object.

For near-field PO, the sampling density of the observation points is much harder to estimate, since the characteristics of the near-field depend on the distance \( |\mathbf{R}| \) between the source and the observation points. In general, the behaviour of the near-field is categorized into three categories \[137, p. 34\], depending on \( |\mathbf{R}| \) relative to the aperture diameter \( D \) when \( D \) is large compared to the wavelength, e.g., \( D > 2.5\lambda \) \[138, (3.13)\]:

- **Reactive near-field:** \( |\mathbf{R}| < 0.62\sqrt{D^3/\lambda} \)
- **Radiating near-field:** \( 0.62\sqrt{D^3/\lambda} \leq |\mathbf{R}| < 2D^2/\lambda \)
- **Far-field:** \( |\mathbf{R}| \geq 2D^2/\lambda \)  

(B.5)

Thus, the field behaves like a far-field, i.e., its angular behaviour is independent of \( |\mathbf{R}| \), when \( |\mathbf{R}| > 2D^2\lambda \). In this region, the sampling densities discussed in the previous section can be used. For observation points closer to the source region, denser samplings are needed, but the asymptotic behaviour of \( N_{obs} \) is, just as in far-field PO, \( O(f^2) \). Furthermore, the sampling densities of the current points are computed using the same auto-convergence procedure as discussed above, and so, \( N_{int} \) also scales as \( O(f^2) \).

**B.2 Fast PO Literature**

In this section, we review the literature on accelerating the evaluation of the far-field (B.3) and near-field (B.4) surface integrals.

One of the most popular acceleration approaches is the Fast Far-Field Approximation (FaFFA), developed by Lu & Chew in \[139\] to estimate the RCS of scatterers, but applicable to field evaluation in general \[140, 142\]. It is, at its core, basically the application of MLFMM from the source points to the observation points, where only a single direction \( \hat{k} \) is used in the translator \[D.9\]. The chosen direction is the line connecting the source and observation groups.

This means that the interaction between source and observation groups is approximated by far-field behaviour, such that the radiated field from the source to the observation group can be approximated by a single plane wave, i.e., it
is applicable when the distance between source and observation groups is very large. Typically, the criterion that the distance should be greater $2D^2/\lambda$ is used to determine whether the separation is large enough, with $D$ the diameter of the groups on the specified level. In a multi-level approach, this means that if the criterion is not fulfilled on a specific level, one can go to a finer level (where $D$ is smaller), and expect the criterion to be fulfilled.

The very severe drawback of this approach, however, is that there is no error control—one cannot know a priori how large the error will be. Indeed, the criterion $2D^2/\lambda$ is only valid for large $D$, and is by no means an exact criterion for far-field behaviour. Therefore, we will not consider FaFFA-style approaches in the following.

### B.2.1 Far-field Fast-PO

Another approach originated in the papers by Boag [33,34] and Boag & Michielssen [143]. Here, the main tool is interpolation on compensated fields, where the compensation results in a field that is much easier to interpolate. In both [33] (for 2-D EM fields) and much later [144] (for 3-D), near-fields were considered, while [34] considered far-fields in 3-D and as a function of frequency. It is extremely important to distinguish between the near-field and far-field implementation of the Fast-PO approach.

For far-fields, the kernel is $e^{jk\hat{r}\cdot r'}$, which means that for large $r'$, the oscillations are very significant, requiring a large amount of observation points to accurately represent the field. By applying a multi-level grouping, we could rewrite (B.3) as

$$E_{\text{far}}(\hat{r}) = \frac{j\eta_0}{4\pi} e^{jk\hat{r}\cdot d} \hat{r} \times \left[ \hat{r} \times \int J_S(r') e^{jk\hat{r}\cdot(r'-d)} dS' \right],$$

where $d$ is the center of the source group in which (B.6) is computed. The expression inside the integrand is labeled the phase-compensated field, since with a properly chosen $d$, it has a much slower angular variation than (B.3), similar to what you would achieve by moving the phase center of a small, off-origin current distribution from the origin of the coordinate system to the center of the distribution.

This means that we could apply the following conceptual procedure for each source group:

1. Choose $d$ as the center of the source group.
2. Tabulate (B.6) for all $J_S$ inside the group, in a low number of observation points. A low number of points is sufficient because the angular variation as a function of $\hat{r}$ is slow, since $|d - r'| << |r'|$.

3. Interpolate, using, e.g., a low-order polynomial in $\theta$ and $\phi$, to the high number of observation points requested.

4. Restore the original phase variation by multiplying $e^{jkr \cdot d}$ onto each observation point.

5. Add the resulting field to a global field which represents the sum over all source groups.

It is important to stress that the compensated field, i.e., the integrand of (B.6), will indeed vary much slower than the real physical field, and that the angular variation is directly proportional by the maximum value of $|d - r'|$, i.e., the radius of the group. The outlined procedure can be applied in a multi-level fashion by interpolating and restoring the phase variation as we go up the tree.

B.2.2 Near-field Fast-PO

For near-field PO, the approach outlined in [144] is conceptually similar to the procedure for far-field PO outlined above, being based on interpolation of a compensated field in angular components as in the far-field case. Furthermore, the method requires interpolation for the radial component. Since the kernel is quite different, however, the method is significantly less effective, basically because it is much more difficult to derive a compensated field. The compensation used in [144] is

$$H_i(r) = -\frac{1}{4\pi} \int \int \left( \hat{R} \times J_S(r') \right) s(r, r') e^{jks(r, r')} \frac{1 + jk|\hat{R}|}{|\hat{R}|^2} e^{-jk|\hat{R}|} dS', \quad (B.7)$$

where

$$s(r, r') = \sqrt{|r - d|^2 + \frac{D^2}{4}}. \quad (B.8)$$

The integrand of (B.7) is termed the phase-and-amplitude compensated field. While the choice of compensation can initially appear somewhat arbitrary, it is based on the asymptotic behaviour of a field from a surface current distribution of diameter $D$ centered at $d$. Thus, the function $s(r, r')$ implicitly depends on $r'$, since in (B.8), $d$ is the center of the group containing $r'$, and $D$ is the diameter.
The radial dependence of an electromagnetic field can be represented as [100, App. A1]:

\[ E(r) = \sum_{n=0}^{\infty} Q_n h_n^{(2)}(kr) + \sum_{m=0}^{\infty} T_m \frac{1}{kr} \frac{d}{d(kr)} \left( k r h_m^{(2)}(kr) \right), \quad (B.9) \]

where \( r \) is the radial distance from the center of the source distribution to the observation point. Using the rewrite [100, (A1.10)]

\[ \frac{1}{kr} \frac{d}{d(kr)} \left( k r h_m^{(2)}(kr) \right) = \frac{1}{2m+1} \left( (m+1) h_{m-1}^{(2)}(kr) - m h_{m+1}^{(2)}(kr) \right), \quad (B.10) \]

we can write

\[ E(r) = \sum_{n=0}^{\infty} C_n h_n^{(2)}(kr). \quad (B.11) \]

Since the \( n \)'th spherical Hankel function can be expressed as

\[ h_n^{(2)}(kr) = \frac{1}{kr} e^{-jkr} \left[ p_{n-1} \left( \frac{1}{kr} \right) - j p_n \left( \frac{1}{kr} \right) \right], \quad (B.12) \]

where \( p_n \) denotes a polynomial of degree no more than \( n \), \( p_{-1} = 0 \). Then, we can write

\[ E(r) = \sum_{n=0}^{\infty} V_n \frac{1}{kr} e^{-jkr} \left[ p_{n-1} \left( \frac{1}{kr} \right) - j p_n \left( \frac{1}{kr} \right) \right]. \quad (B.13) \]

With these expressions, we see that the term \( s(r, r') e^{jkr} e^{jkr} \) is an approximate adjustment for the oscillations \( e^{-jkr}/kr \) from a source distribution of diameter \( D \). For high values of \( kr \), \( p_n \left( \frac{1}{kr} \right) \approx 0 \) for all but the lowest orders, and thus piecewise low-order polynomial interpolation in \( r \) can be expected to work fine for the compensated field. However, we also see that as \( kr \) is reduced, the radial oscillations will be the sum of high-order polynomials and interpolation using low-order polynomials cannot be expected to yield accurate results.

We attempted a number of approaches to correct for this in our first implementation of near-field Fast-PO, which closely resembled that in [144][145]. In particular, for the radial interpolation, we applied interpolation using

- spherical Hankel functions,
- piecewise interpolation in overlapping subdomains using the inverse monomial basis,
- Chebyshev nodes instead of equidistant nodes for polynomial interpolation.
Regardless of the details of the implementation, the approach ultimately requires a very high number of interpolation points in \( r \) to achieve even moderate (around 1\%) accuracy for observation points in the reactive or radiating near-field.

We note that while Boag [34] derives sampling rules for the angular components and the frequency spacing, and Brick & Boag [144] derives a sampling rule for the radial components, all of these are accompanied by oversampling parameters \( \Omega_{\theta}, \Omega_{\phi}, \Omega_f, \Omega_r \) to compensate for the fact that the interpolation is not exact. Further, [144] defines a boundary of radius \( \Omega R \), where \( R \) is the radius of the minimum sphere of the source distribution. All observation points inside a sphere of radius \( \Omega R \), centered at the center of the source distribution, have to be evaluated using direct application of (B.4). For many practical applications, this limitation will mean that all interactions are computed by direct PO.

There are surprisingly few results published using this method. Brick & Boag [144] consider a small spheroid of radii \( 5\lambda \) and \( \lambda \), but uses the approach to solve integral equations in acoustics and therefore the results are not directly comparable. Letrou & Boag [146] perform interpolation on grids defined on the surface of the target scatterer in a dual reflector system. This means that interpolation is performed in a local \((u,v)\)-grid on the main reflector, and thus only two components are needed. However, it also requires the generation of increasingly fine interpolation grids on a specified scatterer, which would be very hard to implement for arbitrary scatterers. Further, the accuracy of the interpolation near the edges of the scatterer cannot be expected to be sufficient for most of TICRAs applications. The method is applied to a dual reflector system with a parabolic main reflector with very low edge illumination. The achieved accuracy is not investigated.

Recently, Gendelman et al. [145] applied the 3-D interpolation approach with the addition of a parameter \( M \) that sets a distance beyond which the field is assumed to have far-field behaviour. They consider the back-scattering from a spheroid of radii \( 2\lambda \) and \( 10\lambda \) due to a point-source placed at \((\theta_0, \phi_0, r_0)\), with \( \theta_0 \) and \( \phi_0 \) covering the full sphere, and varying \( r_0 \). Figs. 5 and 6 in [145] investigate the dependence of the achieved Relative RMS Error on the oversampling and \( M \), respectively. The plots clearly demonstrate that significant oversampling is required to achieve even 1\% accuracy. Further, it is interesting to note that no results are presented for \( r_0 < 100\lambda \), and that no results are presented with better than 0.5\% accuracy for \( r_0 < 2000\lambda \), although the paper notes that the parameters can be calibrated to achieve better accuracies. Since \( D = 20\lambda \), the far-field of the structure begins at roughly \( 2 \cdot 20^2 \lambda = 800\lambda \), and thus it appears that accuracies much better than 1\% Relative RMS Error might be hard to achieve inside the near-field, which corresponds with our observations for our own initial implementation of the method. We stress that 1\% Relative RMS Error is likely sufficient for many applications, but not for the software released
by TICRA — particularly when it comes to PO, the Relative RMS Error is expected to be at most 0.01%.

**B.3 Our Implementation of Fast-PO**

In this section, we briefly describe the main attributes of our Fast-PO implementation, although we repeat that we cannot go into too much detail as discussed in the introduction to this appendix.

First, we remark that PO is often combined with PTD (Physical Theory of Diffraction) to account for the diffraction on the edges of the surface. The PTD contribution is not accelerated by our Fast-PO code, which means that for larger cases, the time used for PTD calculations can become the dominant term. This was discussed, and included in Fast-PO in Letrou & Boag [36], demonstrating that for the Fast-PO code there is no real trouble in including PTD. However, due to the structure of the PTD code used at TICRA, the implementation of PTD in our Fast-PO has been postponed for now.

The current Fast-PO code is primarily a proof-of-concept. It is not nearly as optimized as the MLFMM code, and in particular, no special care has been taken to ensure low memory use. The multi-level structure of both the near- and far-field Fast-PO means that it is fairly easy to partition to reduce the memory use, as Parrot et al. [147] do for a distributed-memory scheme and Manyas & Gürel [148] do for a non-standard Fast-PO implementation.

Our far-field Fast-PO is conceptually similar to that used by the group of Boag et al. as discussed in Section B.2.1 though with several important improvements. For the near-field Fast-PO, in contrast, the problems discussed in Section B.2.2 forced us to develop a novel approach that achieves much better performance.

In particular, for both near- and far-field, TICRA was adamant in their requirement that the specified accuracy should be kept, even for relative errors better than $10^{-4}$, which disqualified the existing implementations of both near- and far-field Fast-PO as discussed above. Further, many of the scenarios studied using TICRA's software involve observation points far inside the minimum sphere of the source distribution. Thus, the near-field Fast-PO code has to provide an acceleration even for this scenario while maintaining the accuracy, which further confirmed that we had to develop a new approach for near-field Fast-PO.

One type of scenario remains where our near-field Fast-PO cannot be used directly. In the case where a part of the source scatterer illuminates the front part
of the observation points, and another part illuminates the back part, an interesting problem appears. A sign change is needed for some of the contributions, due to the changing normal vectors. However, since the radiation of an entire group of source points is tabulated without taking the source points into account, it is not possible to include the sign change, which means that near-field Fast-PO cannot be used. This scenario is easily detected, and one can switch to direct PO, but allowing Fast-PO to accelerate the computations in this type of scenario is still an open problem.

B.4 Results

In this section, we present some results of our Fast-PO implementation. We begin by considering a theoretical scenario, which highlights the scaling of the approach against increasing frequency, in Section B.4.1. Both the near- and far-field algorithms are applied.

We then continue with a far-field setup in Section B.4.2 taken from the literature, demonstrating the performance for a case with a very high number of observation points. For a similar setup, though with a more realistic number of observation points, see [C5, Section IIIb] (for Fast-PO) and [C2, Section IIIb] (for PO on a Graphics Processing Unit).

Finally, we consider a reflector antenna case where our MLFMM and Fast-PO is combined. First, MLFMM is used to compute the surface current density due to an incident field, and subsequently the radiation in the aperture plane from the system is computed using Fast-PO.

B.4.1 Scattering by a plate

As an illustration of the performance of the algorithms for near- and far-field radiation on a simple geometry, we consider the scattering from a 1 m × 1 m plate, illuminated by a plane wave at 45° incidence at varying frequencies. The configuration is illustrated in Figure B.2. All computations are done on the machine detailed in Appendix G.2.

For the near-field setup, we evaluate the field on a half-sphere, with a radius of 0.8 m and origin at the center of the plate. The points are distributed equidistantly in a $\theta \phi$ grid, with the total number of points $N_{\text{obs}}$ determined based on
the wavelength $\lambda$ of the incident field, such that

$$N_{\text{obs}} = \frac{25}{\lambda^2}, \quad (B.14)$$

while the number of integration points is determined based on an auto-convergence procedure, requiring $-80$ dB accuracy [136]. The time required for auto-convergence is not included in the timings. We note that the observation sphere is located quite close to the plate, and with the frequency being varied between 12 GHz and 192 GHz, the observation points are located deep within the reactive near-field region as discussed in (B.5).

The results are shown in Figure B.3. The Relative RMS Error achieved can be seen to vary slightly between the frequencies, but is not above the specified limit of $10^{-4}$ and is actually kept below $10^{-5}$. Fitting a linear curve to the computation times, we get close to the expected $O(f^4)$ behaviour from the direct PO routine, while the Fast-PO timings scale slightly better than $O(f^2)$. If we further increased the frequency, we would expect the scaling to eventually reach the $O(f^2 \log f^2)$ scaling suggested in the literature.

For the far-field algorithm, we consider the same scenario as the radius $r$ of the observation sphere $r \to \infty$. The field is evaluated in a $\theta\phi$ grid with $\theta \in [-\pi/2, \pi/2]$, $\phi \in [0, 2\pi]$, and the number of points $N_{\text{obs}}$ on the grid is chosen using (B.14) while the integration points are chosen using auto-convergence. We get the results illustrated in Figure B.4. Once again, the Relative RMS error achieved varies slightly between the frequencies, but is below the limit $10^{-4}$. 
Figure B.3: Performance of our Fast-PO implementation for near fields compared to the direct PO implementation in GRASP 10.3. The Relative RMS Error between the fields obtained from direct PO and Fast-PO is shown in cyan.
Performance of our Fast-PO implementation for far fields compared to the direct PO implementation in GRASP 10.3. The Relative RMS Error between the fields obtained from direct PO and Fast-PO is shown in cyan.

Figure B.4: Performance of our Fast-PO implementation for far fields compared to the direct PO implementation in GRASP 10.3. The Relative RMS Error between the fields obtained from direct PO and Fast-PO is shown in cyan.
B.4 Results

B.4.2 Offset reflector

As a second test case, we consider the scenario used by Parrot et al. [147] and Letrou & Boag [36]. A single offset parabolic reflector with diameter $D = 40\lambda$, focal length $F/D = 0.8$ with a feed offset from the center by a vertical distance of $30\lambda$. A Gaussian feed model is used, with a $-12$ dB taper at the rim of the reflector. Initially, the frequency is set to $300$ MHz, and by multiplying the frequency by a factor of $q$, we can scale the electrical size of the problem. The observation points are distributed on a full far-field sphere, arranged in a $\theta\phi$-grid with 1473 $\theta$-cuts and 2945 $\phi$ cuts. Letrou & Boag [36] do not directly indicate whether they scale the number of observation points, but their computation times suggest that they do. The sampling corresponds to a sampling spacing of $\frac{1}{12} \frac{\lambda}{D}$, a significant oversampling as discussed in Section B.1.1. This oversampling makes the scenario more advantageous for Fast-PO. We also note that since the reflector is rotationally symmetric, only the principal cuts ($\phi = 0^\circ, 45^\circ, 90^\circ$) are really relevant to gauge the performance of the reflector.

Letrou & Boag [36] present several results, both for $q = 1$ and $q = 4$, and discuss the computational speed, both with and without the PTD contribution — since our code does not currently support PTD, we do not include PTD in the comparison, but note that they achieve strong performance with their acceleration of PTD. They use an Intel Xeon X5460 3.16 GHz processor and have applied a home-made direct PO code. To compare the relative speedup achieved by running Fast-PO for their code and ours, we run the scenario using the direct PO in GRASP 10.3, as well as our Fast-PO, and analyze our computation times in Table B.1. The machine used is detailed in Appendix G.1.

The table leads to a number of interesting conclusions:

• Their home-made direct PO is 17 and 24 times slower than GRASP 10.3, respectively, but it is unclear how much of this is due to the difference in processor use. We remark that from online benchmarks\(^1\) it appears that our processor is roughly twice as fast as theirs, which still leaves roughly a factor of 10 in performance between their direct PO and GRASP.

• They apply a simple 8-point Gauss quadrature rule on each patch of their meshed geometry, while GRASP applies an integration rule that is specialized to parabolic reflectors, yielding a much more sparse distribution of points on the reflector surface. This difference leads to an advantage for their Fast-PO speed-ups, since Fast-PO works relatively better (compared to the direct approach) as the density of the source points is increased.

\(^1\)See [http://www.cpubenchmark.net](http://www.cpubenchmark.net)
Fast Physical Optics

Table B.1: Comparison of the approaches used by Letrou & Boag \[36]\ and in our implementation. Note that the times cannot be compared between \[36]\ and our code, since we are not using the same computer. Further, we note that the speed-ups are not directly comparable, since their direct PO times are based on a home-made code while our times are based on GRASP 10.3.

<table>
<thead>
<tr>
<th></th>
<th>Our</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct PO [s]</td>
<td></td>
</tr>
<tr>
<td>( q = 1 )</td>
<td>125</td>
</tr>
<tr>
<td>( q = 4 )</td>
<td>125</td>
</tr>
<tr>
<td>Fast-PO [s]</td>
<td></td>
</tr>
<tr>
<td>( q = 1 )</td>
<td>7.1</td>
</tr>
<tr>
<td>( q = 4 )</td>
<td>115</td>
</tr>
<tr>
<td>Speed-up</td>
<td></td>
</tr>
<tr>
<td>( q = 1 )</td>
<td>17.6</td>
</tr>
<tr>
<td>( q = 4 )</td>
<td>209</td>
</tr>
<tr>
<td>Memory [MB]</td>
<td></td>
</tr>
<tr>
<td>( q = 1 )</td>
<td>10</td>
</tr>
<tr>
<td>( q = 4 )</td>
<td>150</td>
</tr>
</tbody>
</table>

- At \( q = 4 \), the number of output points is extreme, more than \( 69 \cdot 10^6 \). This also means that the memory use is dominated by the memory required to store the observation points. We do not include that in our memory use, which only includes the memory required for Fast-PO. Further, the Fast-PO time for \( q = 4 \) does not include the significant I/O overhead involved in handling the large amount of data by GRASP. The total time required to solve the problem in GRASP (i.e., including all data processing) is around 3 minutes.

- Letrou & Boag do not indicate the memory consumption in \[36]\, and while the same research group report memory consumption in \[147]\ for a very similar setup, \[147]\ does not indicate the number of observation points or whether the memory includes that required to store the observation points. Thus, we do not feel it is fair to compare the memory use.

We further note that the Relative RMS Error achieved by our Fast-PO solution is \( 8.5 \cdot 10^{-7} \) for \( q = 1 \), quite close to the machine precision for IEEE single precision. This even suggests that we could reduce the accuracy slightly, to further reduce the computational time.
B.4 Results

Figure B.5: Mesh of the Helios Command Station antenna system.

B.4.3 Evaluation of MLFMM currents

As a final example, we consider the use of Fast-PO in the more general case of evaluating the field from a surface current distribution. Further, we compare our MLFMM implementation, discussed in the previous chapter, to another implementation found in the literature. The case was also discussed in \[O3\].

The application is taken from Di Maria et al. \[149\] where the Helios Command Station was analyzed at 1.5 GHz. It is a Cassegrain dual reflector system, with a specialized feed system, akin to a beam waveguide—a 9.4 m long horn radiating onto a 3.2 m-diameter paraboloidal reflector. The subreflector is 4.1 m in diameter and the main reflector is 30 m in diameter. The mesh is shown in Figure B.5. The surface used in \[149\] was a least-squares fit to the points from a laser scanner model of the main reflector, which we obviously cannot replicate, and thus we are forced to use a nominal reflector surface. We stress, however, that for the purposes of analyzing and comparing the computational work between our algorithm and that in \[149\], the very minor difference in the geometry is irrelevant.

The comparison between the use of our MLFMM implementation and the MLFMM used by Di Maria et al. \[149\] is done in Table B.2 using the machine detailed in Appendix G.1. It allows for two interesting observations:

- The reduction in memory is very significant, a factor of 21. Remarkably, this reduction is so significant that it allows us to solve the problem on a
In Table B.2, we compare the computational resources used by our MLFMM implementation and that reported by Di Maria et al. \cite{149}. Thus, for this case, the strong focus in the MLFMM implementation on keeping the memory use to a minimum allows us to use the computer at hand, rather than requiring a dedicated computing server.

- In spite of us using a laptop rather than a server, and using an MLFMM implementation that focuses on low memory, the computing time is still reduced by a factor of 28. This is both due to the strong FGMRES (Section 3.6) and efficient preconditioner (Section 3.7), allowing a low number of iterations to reach the prescribed solution norm, as well as the efficient interpolation with TPE-MLFMM (Section 3.4).

Having completed the MLFMM solution, we achieve a surface current density tabulated in 1.2 million integration points on the surface. To see the effect of the subreflector on the aperture field, we evaluate the scattered field on a plane passing $\lambda/10$ behind the subreflector. The sampling density on this plane is chosen as $\lambda/5$ along each axis, resulting in half a million observation points. The direct calculation of the scattered near-field using (B.4) for this scenario requires roughly 86 minutes.

In other words, the different scalings of the algorithms in play are evident here. While the usual MoM scales as $O(f^4) - O(f^6)$ depending on the use of iterative or direct solvers, the MLFMM reduces this to $O(f^2 \log f^2)$. However, having found the currents, the integration of them in a sufficient number of observation points also scales as $O(f^4)$. Therefore, while our application of MLFMM to find the surface current density provides an extreme reduction in memory and time compared to direct MoM, it has simply moved the bottleneck in the solution process to the evaluation of the scattered field.

However, applying our Fast-PO algorithm, which is essentially an algorithm for the evaluation of the field from a surface current distribution, the computation time is reduced to 44 seconds, with a relative error of $5 \cdot 10^{-5}$. The speedup is a factor of 117.
The resulting co-polar and cross-polar field is shown in Figure B.6. We see a significant shadow effect behind the subreflector, as well as obvious diffraction patterns. The asymmetric nature of the field is due to the feeding arrangement.
This appendix contains a very brief summary of the most important result from classical time-harmonic electromagnetic field theory in relation to scattering problems from perfectly electrically conducting objects. There are numerous textbooks that go through these results in far more detail, see, e.g., [37, Chapter 2], [150]. Further, we consider the discretization of integral equations in general, and with special focus on the basis functions used in electromagnetics. Finally, we consider the scenario where homogenous dielectric materials are present on the scatterer.

Given a homogeneous, isotropic and linear medium, with constituent parameters $\epsilon$ and $\mu$, the electric and magnetic fields $E,H$ must satisfy Maxwells equations

\[
\begin{align*}
\nabla \times E &= -M - j\omega B \\
\nabla \times H &= J + j\omega D \\
\n\nabla \cdot D &= q_e \\
\n\nabla \cdot B &= q_m
\end{align*}
\]

where $B = \mu H$, $D = \epsilon E$ and the time dependence $e^{j\omega t}$ is assumed and suppressed.
The boundary conditions, expressing the behavior of the quantities at the interface between two connected regions, are

\[ -\hat{n} \times (E_2 - E_1) = M_S, \quad \text{(C.2a)} \]
\[ \hat{n} \times (H_2 - H_1) = J_S, \quad \text{(C.2b)} \]
\[ \hat{n} \cdot (D_2 - D_1) = q_e, \quad \text{(C.2c)} \]
\[ \hat{n} \cdot (B_2 - B_1) = q_m, \quad \text{(C.2d)} \]

where \( \hat{n} \) is the normal vector pointing from region 2 to region 1.

Solving Maxwell's equations for a specific scenario can be done by deriving and solving an integral equation. To do this we must decouple Maxwell's equations, at the cost of increasing the order of the derivatives. We start by taking the curl of (C.1a) and combining it with (C.1b):

\[ \nabla \times \nabla \times E = -\nabla \times M - j\omega \mu \nabla \times H \quad \text{(C.3)} \]
\[ = -\nabla \times M - j\omega \mu (J + j\omega \epsilon E) \quad \text{(C.4)} \]
\[ = -\nabla \times M - j\omega \mu J + \omega^2 \mu \epsilon E \quad \text{(C.5)} \]

Exploiting the identity \( \nabla \times \nabla \times E = \nabla (\nabla \cdot E) - \nabla^2 E \), introducing the wavenumber \( k = \omega \sqrt{\mu \epsilon} = \frac{2\pi}{\lambda} \), and realizing that \( \nabla \cdot E = \frac{q_e}{\epsilon} \), we rewrite (C.5) to

\[ \frac{\nabla q_e}{\epsilon} - \nabla^2 E = -\nabla \times M - j\omega \mu J + k^2 E. \quad \text{(C.6)} \]

Rearranging, we get

\[ \nabla^2 E + k^2 E = \nabla \times M + j\omega \mu J + \frac{\nabla q_e}{\epsilon}. \quad \text{(C.7)} \]

This is the so-called wave equation, which expresses the E-field at a given position as a function of the magnetic and electric currents as well as the gradient of the electric charge at that position. Exploiting the duality theorem, we can easily deduce the equation for the magnetic field.

\[ \nabla^2 H + k^2 H = -\nabla \times J + j\omega \epsilon M + \frac{\nabla q_m}{\mu}. \quad \text{(C.8)} \]

Considering the important special case of a source-free, lossless material (such that \( J = M = q_e = q_m = 0 \)), we reduce (C.7)-(C.8) to the Helmholtz equations

\[ \nabla^2 E + k^2 E = 0 \quad \text{(C.9a)} \]
\[ \nabla^2 H + k^2 H = 0 \quad \text{(C.9b)} \]
These are second-order PDEs, solvable through the use of a Green’s function. The Green’s function $G$ solves the scalar Helmholtz equation for a delta-input, i.e.,

$$\nabla^2 G(r, r') + k^2 G(r, r') = -\delta(r, r').$$  \hspace{1cm} (C.10)

Based on solving for spherical symmetry, we get that

$$G(r, r') = \frac{e^{-jk|r-r'|}}{4\pi|r-r'|}.$$  \hspace{1cm} (C.11)

With $G$, we can solve the wave-equation (C.7), when only electric currents are radiating, as

$$E(r) = -j\omega\mu \iint_V G(r, r') \left[ J(r') + \frac{1}{k^2} \nabla' \nabla' \cdot J(r') \right] dr',$$  \hspace{1cm} (C.12)

where $V$ is the support of $J$. In other words, (C.12) expresses the electric field vector at $r$ as a function of the current $J$ integrated over its support. We can simplify this for scattering problems by expressing the scattered electric field from the induced surface current as

$$E^s(r) = -j\omega\mu \int_S G(r, r') \left[ J_S(r') + \frac{1}{k^2} \nabla' \nabla' \cdot J_S(r') \right] dr',$$  \hspace{1cm} (C.13)

where $S$ is the surface on which $J_S$ is defined.

For scattering problems, however, we have neither the induced surface current nor the scattered field. Instead, we have the incident field $E^i$. For a perfect electric conductor (PEC), we can enforce the boundary condition

$$-\hat{n} \times E^i(r) = \hat{n} \times E^s(r)$$  \hspace{1cm} (C.14)

and use this to derive the scattered field $E^s$.

Inserting this in (C.13), we get the Electric Field Integral Equation (EFIE) for a perfect electric conductor:

$$-\frac{j}{\omega\mu} \hat{n} \times E^i(r) = \hat{n} \times \int_S G(r, r') \left[ J_S(r') + \frac{1}{k^2} \nabla' \nabla' \cdot J_S(r') \right] dr'.$$  \hspace{1cm} (C.15)

The EFIE can be represented in a variety of ways. In particular, the gradient and divergence operators can operate on either the observation or integration coordinate, with corresponding changes made elsewhere in the equation. We note in passing that the EFIE is a Fredholm integral equation of the first kind, since the unknown ($J$) appears inside the integral and the integration limits are constant. Further, the kernel of the EFIE is singular for $r = r'$ due to the
Green’s function. The EFIE can also be represented as an operator equation, i.e., as
\[- \frac{j}{\omega \mu} \hat{n} \times E^i = \mathcal{L}(J_S) \] (C.16)

We can similarly obtain the Magnetic Field Integral Equation (MFIE) for a perfect electric conductor by applying the boundary condition for magnetic fields, though we note that the process is significantly more involved due to the fact that the boundary condition yields an electric current instead of zero, since the right-hand side of (C.2a) is zero, while the right-hand side of (C.2b) is not. The MFIE is:
\[ \hat{n} \times H^i(r) = \frac{J_S(r)}{2} - \hat{n} \times \int_{S-\delta S} J_S(r') \times \nabla' G(r, r') dr', \] (C.17)
or, in operator notation:
\[ \hat{n} \times H^i = \left( \frac{I}{2} - \kappa \right) J_S, \] (C.18)
where $I$ is the identity operator.

The key difference between the EFIE and the MFIE is that the former is valid for both open and closed surfaces, while the latter is valid only for closed surfaces. The MFIE is a Fredholm integral equation of the second kind, and thus better conditioned than the EFIE. Unfortunately, the kernel of the MFIE is even more singular than for the EFIE, due to the gradient of the Green’s function, making numerical integration somewhat harder.

Both the EFIE and the MFIE suffer from the unfortunate property of yielding non-unique solutions for some frequencies for closed scatterers. This is typically called the interior resonance problem. For the EFIE, the singular frequencies correspond to the resonant frequencies of a cavity formed by filling the interior of the scatterer with the exterior medium [28, p. 1070]. There is a number of ways to combat this — the most common is the use of a Combined Field Integral Equation (CFIE), representing a linear combination between the EFIE and MFIE:
\[ \text{CFIE} = \alpha \cdot \text{EFIE} + P(1 - \alpha) \cdot \text{MFIE} \] (C.19)
Here, $\alpha$ is chosen such that spurious solutions are eliminated, typically $0.2 \leq \alpha \leq 0.3$. $P$ is a scaling parameter, designed to take into account the difference in magnitudes of the EFIE and MFIE. The strictly correct scaling would be $P = Z_0$, the free-space impedance defined as $|E| / |H|$ in the far-field — in this scenario, $\alpha$ is easily understood as the relative weighting, and is often chosen as $\alpha = 0.5$. However, in parts of the literature, particularly concerning MLFMM, the scaling used is $P = \frac{i}{k}$.
The CFIE results in an integral operator where the singular frequencies are
the resonant frequencies of a cavity with a resistive wall. Since those resonant
frequencies are complex, the CFIE operator is not singular for a real frequency
\[28, \text{p. 1070}\].

We mention that CFIE or MFIE can be combined with EFIE for a scenario with
both open and closed scatterers, such that CFIE or MFIE is used on the closed
scatterers, and EFIE is used on the open scatterer. This offers significantly
improved convergence compared to purely using EFIE, as discussed by Ergül &
Gürel \[135\].

\section*{C.1 Discretization}

This section discusses the discretization of the integral equations deduced above.
The section is based primarily on Jørgensen \[25\], since this is the foundation for
the MLFMM implementation of the thesis.

To discretize the EFIE, MFIE and CFIE, we employ the Method of Moments
(MoM) scheme, discretizing
\[Z \mathcal{F} = \mathcal{G}\] (C.20)
where \(Z\) is a linear integral operator, \(\mathcal{G}\) is the known response from the system,
and \(\mathcal{F}\) is the unknown function. The use of Method of Moments in electromagnetics
was popularized by Harrington in his book \[5\]. The first step is to
approximate the unknown function by a set of basis functions, such that
\[\mathcal{F} \approx \sum_{j=1}^{N} I_j B_j,\] (C.21)
where \(B_j\) are known vector basis functions, and \(I_j\) are the unknown coefficients.

Defining the residual of (C.20) as \(\mathcal{G} - Z \mathcal{F}\) yields
\[\mathbf{R} = \mathcal{G} - \sum_{j=1}^{N} I_j Z B_j.\] (C.22)

This residual is forced to be orthogonal to a set of testing functions \(T_i\), such that
\[\langle \mathbf{R}, T_i \rangle = 0,\] (C.23)
where \(\langle A, B \rangle\) is the inner product
\[\int_S A^* \cdot B \, dS.\] (C.24)
(C.23) is equivalent to
\[ \langle G - \sum_{j=1}^{N} I_j Z B_j, T_i \rangle = 0, \]  
(C.25)
which in turn is equivalent to
\[ (\sum_{j=1}^{N} I_j Z B_j, T_i) = \langle G, T_i \rangle. \]  
(C.26)
Equation (C.26) can be stated in matrix form as
\[ Z \mathbf{I} = V \]  
(C.27)
where
\[ Z_{ij} = \langle Z B_j, T_i \rangle \]  
(C.28)
\[ V_i = \langle G, T_i \rangle \]  
(C.29)
and the vector \( \mathbf{I} \) contains the unknown coefficients.

The choice of basis functions will be dealt with in a moment. However, we remark that the choice of testing functions is straightforward for scattering problems, since Wandzura [151] proved that the choice \( T_i = B_i \), also known as the Galerkin method\(^1\), yields the optimal scattered fields. To apply this to the EFIE, we first restate (C.15) by using the mixed potential formulation
\[ -\frac{j}{\omega \mu} \hat{n} \times E^i(r) = \hat{n} \times \left( \int_{S'} J_S(r') G(r, r') dr' + \frac{1}{k^2} \nabla \int_{S'} \nabla' \cdot J_S(r') G(r, r') dr' \right). \]  
(C.30)
Inserting this in the expression (C.28) yields the following expression for the matrix element \( Z_{ij} \):
\[ Z_{ij} = \int_{P^i} T_i(r) \cdot \int_{P^j} B_j(r') G(r, r') dr' dr \]  
(C.31)
\[ + \frac{1}{k^2} \int_{P^i} \nabla \cdot T_i(r) \int_{P^j} \nabla' \cdot B_j(r') G(r, r') dr' dr, \]
where \( P^i \) is the support of \( T_i \) and \( P^j \) is the support of \( B_j \). We note that, to remove the gradient operator on the Green's function in (C.15) resulting in a \( \frac{1}{r^2} \) rather than a \( \frac{1}{r} \) singularity, we used the following identity [152, Appendix A] together with the mixed potential formulation (C.30):
\[ \int_{P^i} T_i(r) \cdot \nabla G dr = - \int_{P^i} G \nabla \cdot T_i(r) dr, \]  
(C.32)

\(^1\)We note that in some communities, the “Method of Moments” is the “Galerkin” method for the special case of \( T_i = B_i \). However, we adopt the terms used by the electromagnetics community.
where $G$ is any scalar function. We note that the full version of this identity contains two other terms that are always 0 in our MoM implementation.

In going from (C.30) to (C.31), we implicitly used $\mathbf{J}_S = \sum_{j=1}^N I_j \mathbf{B}_j$, without discussing the properties of the basis functions $\mathbf{B}_j$. The following section will go into detail regarding the choice of basis functions.

### C.1.1 Basis Functions

The choice of basis functions is one of the fundamental choices to make when implementing an electromagnetic scattering code. In general, **subsectional** or **sub-domain** basis functions are used, where the scatterer $\mathcal{S}$ in (C.15) or (C.17) is defined on a set of small geometrical elements (patches) on which the basis functions are defined. The entire collection of patches is called a mesh of $\mathcal{S}$, and the accuracy of this geometrical discretization of $\mathcal{S}$ is a key part of the overall solution accuracy. Using sub-domain basis functions yields basis functions with compact support, which has several advantages over **entire-domain** basis functions, that have support over the entire structure $\mathcal{S}$.

### C.1.2 RWG Basis Functions

The standard sub-domain basis functions are the so-called Rao-Wilton-Glisson (RWG) functions, named after their inventors [21]. These are piece-wise linear functions, defined on small, planar, triangular patches. Each triangle has three basis functions, each of which is 0 at one corner and increase to 1 at the opposing side, and is constant along the other axis. To increase the solution accuracy, the sidelength of the triangles is reduced. Since the sidelength of the patches is typically termed $h$, this procedure of increasing the accuracy is called $h$-refinement.

A brief specification of the RWG basis function is provided in the following. Based on (C.21), we expand the current as

\[
\mathbf{J}_S(\mathbf{r}) = \sum_{j=1}^N I_j \mathbf{B}_j(\mathbf{r}),
\]

where $N$ is the total number of basis functions, $\mathbf{r}$ is a point on the discretized surface $\mathcal{S}$ of the scatterer, and for a fixed $\mathbf{r}$, $\mathbf{B}_j(\mathbf{r})$ will be zero for most $j$ since the basis functions have very compact support.
In RWG, the basis functions are defined across an edge joining two planar triangles, called an *interior edge*—no basis functions are defined on *exterior edges*, i.e., edges that are only connected to one triangle. Some work has been done with regards to so-called *singular basis functions* on exterior edges, see, e.g., Brown & Wilton [153].

For the interior edge $j$, joining triangles $T_1$ and $T_2$, the basis function $B_j$ is defined as

$$B_j(r) = \begin{cases} \frac{L_j}{2A_1} R_1(r), & \text{if } r \in T_1 \\ \frac{L_j}{2A_2} R_2(r), & \text{if } r \in T_2 \\ 0, & \text{otherwise}. \end{cases} \quad (C.34)$$

Here, $A_1$ and $A_2$ is the area of triangles $T_1$ and $T_2$, respectively. $L_j$ is the length of edge $j$. The vectors $R_1$ and $R_2$ are directed in the same direction relative to the edge, which means that $R_1$ points towards edge $j$, while $R_2$ points away from it (or vice-versa). More specifically, we label the vertices opposite edge $j$ as $v_1$ and $v_2$, depending on whether they are placed in $T_1$ and $T_2$. Based on this,

$$R_1(r) = r - v_1 \quad (C.35)$$

$$R_2(r) = v_2 - r \quad (C.36)$$

An illustration, taken from Gibson [37], is provided in Figure C.1.

However, as shown by Nédélec [132], the discretization error behaves as $O(h^n)$, where $n$ is the order of the basis functions in the direction of the current, and
\( h \) is assumed to be less than 1. Thus, a discretization based on RWG functions \((n = 1)\) will converge much slower to the true solution than higher-order basis function families using \( n > 1 \).

We stress that there is a slightly unclear terminology in the field, relating to the order of a basis function family. This is due to the Nédélec constraint from \([132]\), where it is shown that the order in the direction of the current has to be increased by 1 compared to the order in the orthogonal direction of the current. Therefore, basis functions with, e.g., order 3 in the direction of the current are labelled as order 2 by some authors, since the polynomial expansion is only complete to order 2, and others even use the half-order notation \( 0.5 \). In the present thesis, we refer to the order in the direction of the current.

We note that all the basis function families discussed in this section form so-called divergence-conforming basis function spaces. This means that the basis functions are members of a function space which

- Preserves normal continuity of the current across patch edges.
- Can represent any vector function completely up to \( n - 1 \)’st order.
- Can represent the divergence of a vector function completely up to \( n - 1 \)’st order.

The alternative in computational electromagnetics, curl-conforming basis functions, preserve tangential continuity and can represent the curl of a vector function. These are mainly used for Finite-Element or Finite-Volume approaches, but can also be used for a more accurate discretization of MFIE (see, e.g., \([154,155]\)).

A number of higher-order basis functions have been suggested. Graglia et al. \([23]\) introduced the Graglia-Wilton-Peterson (GWP) functions, which are interpolatory, meaning that the same order of basis functions has to be used on all patches, a significant drawback for practical meshes where the patch sidelength (and thus desired order) can vary greatly. GWP are defined on curvilinear triangular patches, typically using moderate patch lengths \( \lesssim 0.3 \lambda \).

In contrast to interpolatory basis functions, hierarchical basis functions allow varying \( n \) to be chosen throughout the mesh. Kolundžija and Popović \([87]\) defined hierarchical functions of arbitrary order on curved quadrilateral patches, allowing for high accuracy both in the geometrical discretization and the basis functions. However, their basis functions yields a poorly conditioned \( Z \), since the basis functions are not sufficiently orthogonal. This results in a very slow convergence when using an iterative solver.
C.1.3 Legendre Basis Functions

In summary, the discussion in the previous section showed that the optimal set of basis functions should

1. Yield a high accuracy (ruling out RWG).
2. Allow easy evaluation of the matrix elements.
3. Allow separate \( h \) and \( n \) refinement (ruling out interpolatory functions).
4. Result in a low number of unknowns (ruling out RWG).
5. Provide a matrix \( \mathbf{Z} \) that allows for fast convergence in an iterative solver (ruling out Kolundžija and Popović functions).

The basis functions used in this thesis, developed by Jørgensen \cite{25,26}, constitute a strong attempt at meeting all of these requirements. We will now go into a bit more detail about how these basis functions are defined and used.

The currents are discretized using modified Legendre polynomials along the direction of the current, and Legendre polynomials in the transverse direction. Hence, \( J_S \) from (C.15) or (C.17) is expanded as

\[
J_S(r) = J^u_S a_u(r) + J^v_S a_v(r),
\]

(C.37)

where \( a_u = \frac{\partial r}{\partial u} \) is the covariant unitary vector and similarly for \( a_v \). Considering a \( u \)-directed current, \( J^u_S \) is thus expanded as

\[
J^u_S(u,v) = \frac{1}{J_S(u,v)} \sum_{m=0}^{M^u} \sum_{n=0}^{N^v} a^u_{mn} C^u_m \hat{P}_m(u) C^v_n P_n(v)
\]

(C.38)

where the modified Legendre polynomials \( \hat{P}_m \) are defined as

\[
\hat{P}_m(u) = \begin{cases} 
1 - u & m = 0 \\
1 + u & m = 1 \\
P_m(u) - P_{m-2}(u) & m \geq 2
\end{cases}
\]

(C.39)

and \( P_m \) is the Legendre polynomial of order \( m \). \( J_S(u,v) \) is the surface Jacobian. The expansion for a \( v \)-directed current \( J^v_S \) is obtained by interchanging \( u \) and \( v \) in (C.38). Moreover, \( C_m \) is a factor chosen to minimize the condition number of \( \mathbf{Z} \).

To satisfy the Nédélec constraint \cite{132}, \( N^v = M^u - 1 \) for a \( u \)-directed current. The order of the expansion is said to be \( M^u \), but it is complete to order \( M^u - 1 \) only. The term \textit{order} in this thesis refers to the value of \( M^u \) and \( M^v \). Based on this definition, RWG basis functions are defined as first order basis functions.
C.2 Dielectrics

In the previous part of this appendix, we have discussed the formulation and discretization of surface integral equations for the common special case of scattering from a perfect electric conductor (PEC), i.e., a material with zero resistivity. However, materials in which there are ohmic losses or materials in which the permittivity or permeability is different from that of free-space, are also relevant for the applications in which TICRAs product are used.

Specifically, it is fairly common to include homogenous dielectric objects, such as supporting structures or other localized features, as part of a simulation. This means that we want to be able to combine PEC and dielectric materials, including a so-called "junction" where current can flow from a PEC onto a dielectric material. The integral equation used at TICRA is the PMCHWT equation, named after the authors Poggio-Miller-Chang-Harrington-Wu-Tsai \[156\] - \[158\], but we stress that there are several possible formulations, see, e.g., Ylä-Oijala et al. \[159\]. Just as for the case of PEC, where one can apply the boundary condition on the electric field to get the EFIE or on the magnetic field to get the MFIE, we now need to apply two boundary conditions on the interface between two regions, allowing us to choose two from the four conditions (C.2).

For the PMCHWT formulation\[2\], the equations for a scenario with \(N + 1\) homogenous regions with material parameters \(\epsilon_i, \mu_i\) for the \(i\)'th region are:

\[
\hat{n} \times \{E_i - E_j\} = \left\{ \sum_{p=0, p \neq i}^{N} \left[ L_i J_{ip} - K_i M_{ip} \right] - \sum_{p=0, p \neq j}^{N} \left[ L_j J_{jp} - K_j M_{jp} \right] \right\} \tag{C.40a}
\]

\[
\hat{n} \times \{H_i - H_j\} = \left\{ \sum_{p=0, p \neq i}^{N} \left[ K_i J_{ip} + \frac{\epsilon_i}{\mu_i} L_i M_{ip} \right] - \sum_{p=0, p \neq j}^{N} \left[ K_j J_{jp} + \frac{\epsilon_j}{\mu_j} L M_{jp} \right] \right\} \tag{C.40b}
\]

Here, the quantities are

- \(J_{ik}, M_{ik}\): The electric or magnetic currents at the interface between the \(i\)'th and the \(k\)'th region. If these two regions do not touch, then the currents at the interface are obviously non-existent.

\[2\] We remark that some authors \[159\] - \[160\] distinguish between the PMCHWT and CTF (Combined Tangential Formulation) by the use of the \(\frac{\epsilon_i}{\mu_i}\) scaling. In that nomenclature, (C.40) is the CTF formulation.
• $E_i, H_i$: The electric or magnetic field incident in the $i$'th region. Zero if there are no sources within this region.

• $\hat{n}_{ik}$: The normal vector pointing into the $i$'th region from the $k$'th region. Since $\hat{n}_{ik} = -\hat{n}_{ki}$, there is a minus between the fields.

and the integral operators are

$$L_i X_{ik} = j\omega \mu_i \left[ \iint_{S_{ik}} X_{ik}(r') G_i(r, r') dS_{ik} + \frac{1}{k_i^2} \iint_{S_{ik}} \nabla' \cdot X_{ik}(r') \nabla G_i(r, r') dS_{ik} \right],$$

$$K_i X_{ik} = \iint_{S_{ik}} X_{ik}(r') \times \nabla G_i(r, r') dS_{ik},$$

where $k_i$ and $G_i(r, r')$ are the wavenumber and the Green’s function of the $i$'th region, respectively. At first glance, $L$ and $K$ as defined here appear to be the same as those used for the PEC case in (C.15) and (C.17). However, the $J/2$ term is not present in the $K$ operator, since there is now a $J$ term on both sides of the interface with opposite signs, yielding zero. This allows the normal vector to be moved out of the operator, which is not the case for the usual MFIE and is one of the main advantages of the PMCHWT formulation, since it allows us to disregard the normal vector.

Another advantage of PMCHWT is that, since $M_{ik}$ is zero on a PEC, (C.40a) reduces to the EFIE, which in turn allows a simple junction between a dielectric region and an open PEC surface by enforcing both (C.40a) and (C.40b) on the dielectric region and only (C.40a) on the PEC.

The discretization of the components in PMCHWT yields

$$\mathcal{L}(J) : \langle T_m, L_i B_n \rangle = \iint_{p_m} T_m(r) \cdot \iint_{p_n} B_n(r') G_i(r, r') dr' dr \quad (C.41)$$

$$+ \frac{1}{k_i^2} \iint_{p_m} \nabla T_m(r) \iint_{p_n} \nabla' B_n(r') G_i(r, r') dr' dr,$$

$$\mathcal{K}(M) : \langle T_m, K_i B_n \rangle = \iint_{p_m} T_m(r) \cdot \iint_{p_n} B_n(r') \times \nabla G_i(r, r') dr' dr, \quad (C.42)$$

$$\mathcal{L}(M) : \langle \tilde{T}_m, L_i \tilde{B}_n \rangle = \iint_{p_m} \tilde{T}_m(r) \cdot \iint_{p_n} \tilde{B}_n(r') G_i(r, r') dr' dr$$

$$+ \frac{1}{k_i^2} \iint_{p_m} \nabla \tilde{T}_m(r) \iint_{p_n} \nabla' \tilde{B}_n(r') G_i(r, r') dr' dr, \quad (C.43)$$

$$\mathcal{K}(J) : \langle \tilde{T}_m, K_i B_n \rangle = \iint_{p_m} \tilde{T}_m(r) \cdot \iint_{p_n} B_n(r') \times \nabla G_i(r, r') dr' dr. \quad (C.44)$$

Here, a function with a tilde denotes a magnetic basis function.
C.2 Dielectrics

C.2.1 FMM on dielectrics

For the fundamental description of FMM on PEC structures, see Appendix D. For details on our research on FMM, see Chapter 3. The following discussion assumes that the reader has an understanding of both of those sections. Further, we remark that a number of authors have discussed the application of FMM to dielectrics, see, e.g., [30, 49, 159, 160].

Disregarding the Multi-Level part of MLFMM, we here restate the fundamental FMM expression (D.15):

\[ \kappa \iint \mathbf{R}_m(k) \cdot (T_L(k, x)\mathbf{V}_n(k)) \, d^2\hat{k} \]  

(D.15 revisited)

We immediately note a couple of interesting features in relation to dielectrics:

1. All three operators, \( \mathbf{R}_m, \mathbf{V}_n \) and \( T_L \) depend on \( k_i \). Therefore, we need to apply (D.15) on each region separately, taking care to include the negative sign between the two regions in (C.40). In other words, we need a matrix-vector product for each region to account for the varying \( k_i \).

2. Since \( \mathbf{R}_m \) tests the entire group radiation \( \mathbf{V}_n \) in (D.15), and is the same for \( \mathcal{K} \) and \( \mathcal{L} \), there is no way to provide the different scalings of the operators in (C.40) without performing separate matrix-vector products for electric and magnetic basis functions. See Sheng et al. [49] for more details on this.

These two features together mean that we need two matrix-vector products, one where the electric basis functions are excited and one where the magnetic functions are, for each region. Further, we need to allocate separate structures in our MLFMM code for each region to account for the fact that the truncation number \( L \) (D.22) differs between the regions due to the varying \( k_i \). In summary, there is significant overhead involved in applying the PMCHWT formulation for MLFMM.

Moving on to the basis function patterns themselves, the receive pattern \( \mathbf{R}_m \) comes in two variants:

\[ \mathcal{L}(\mathbf{J}), \mathcal{K}(\mathbf{M}) : \mathbf{R}_m(k) = \iint_S \mathbf{f}_m(r) e^{-jk \cdot \hat{k} \cdot (r - r_m)} \, dS, \]  

(C.45a)

\[ \mathcal{L}(\mathbf{M}), \mathcal{K}(\mathbf{J}) : \mathbf{R}_m(k) = \iint_S \mathbf{f}_m(r) e^{-jk \cdot \hat{k} \cdot (r - r_m)} \, dS. \]  

(C.45b)
However, the transmit pattern is different for each term:

\[ L(J) : V_m(k) = \int_S f_m(r) \cdot \left[ \mathbf{I} - \hat{k} \mathbf{k} \right] e^{-jk \cdot (r_m - r)} dS, \tag{C.46a} \]

\[ K(M) : V_m(k) = \hat{k} \times \int_S f_m(r) e^{-jk \cdot (r_m - r)} dS, \tag{C.46b} \]

\[ L(M) : V_m(k) = \hat{k} \times \int_S \tilde{f}_m(r) e^{-jk \cdot (r_m - r)} dS, \tag{C.46c} \]

\[ K(J) : V_m(k) = \hat{k} \times \int_S f_m(r) e^{-jk \cdot (r_m - r)} dS. \tag{C.46d} \]

We again note that the tilde above \( f \) denotes that it is a magnetic basis function.

At first glance, it could appear that for each basis function, we would have to store three patterns for each region, e.g., for a magnetic function we would have to store (C.45b), (C.46b) and (C.46c). However, since

\[ V_L(J)_m(k) = -\hat{k} \times V_K(J)_m(k) = \text{conj}(R_L(J)_m(k), K(M)_m(k)), \tag{C.47a} \]

\[ V_L(M)_m(k) = -\hat{k} \times V_K(M)_m(k) = \text{conj}(R_L(M)_m(k), K(J)_m(k)), \tag{C.47b} \]

where \( \text{conj}() \) is the complex conjugate, we only need to store one pattern per basis function, and then we can apply the \(-\hat{k} \times\) operation on the fly, which in \( \hat{\theta}, \hat{\phi} \) components is simple: \( \hat{k} \times \left[ \hat{\theta}, \hat{\phi} \right] = \left[ \hat{\phi}, -\hat{\theta} \right] \).

We remark that when using the Spherical Harmonics Expansion (SHE) storage of the patterns, as detailed in Section 3.3, the patterns are stored in \( \hat{x}, \hat{y}, \hat{z} \) components. This makes the \( \hat{k} \times \) operation complicated to perform on the patterns, and thus it should instead be performed on the incoming group pattern (which is in spherical components) during the disaggregation stage (see Section D.4). Note that the symmetry relations for the SHE expansion coefficients (3.7) depend on whether the \( \hat{k} \times \) operation has been performed on the basis function pattern (3.7b) or has been moved to the incoming group pattern (3.7a).

### C.2.2 MFIE/CFIE with Dielectrics

When combined with CFIE on a closed PEC, FMM becomes more involved. Since the \( \hat{n} \times \) operation can be ignored in (C.40b), it is not present in the receive and transmit functions. However, when combining dielectrics with a closed PEC structure on which we are using the MFIE or CFIE, (C.17) includes the \( \hat{n} \times \) operation which cannot be ignored. This is also evident in the usual MFIE transmit function, (D.16).
This, in turn, means that the computation of interactions between the closed PEC structure and the dielectric becomes slightly more complicated.

With a test function residing on the closed PEC volume, and a basis function on the dielectric region, the receive patterns necessary are

\[
\mathcal{L}(J), \mathcal{K}(M) : R_m(k) = \int_S f_m(r) e^{-jk_0 k \cdot (r-r_m)} dS,
\]

(C.48a)

\[
\mathcal{L}(M), \mathcal{K}(J) : R_m(k) = \int_S [\hat{n} \times f_m(r)] e^{-jk_0 k \cdot (r-r_m)} dS,
\]

(C.48b)

where \(k_0\) is the free-space wavenumber. The transmit patterns are as indicated in (C.46).

With a test function residing on the dielectric region, and a basis function on the closed PEC structure, the receive patterns necessary are

\[
\mathcal{L}(J), \mathcal{K}(M) : R_m(k) = \int_S f_m(r) e^{-jk \cdot k \cdot (r-r_m)} dS,
\]

(C.49a)

\[
\mathcal{L}(M), \mathcal{K}(J) : R_m(k) = \int_S \tilde{f}_m(r) e^{-jk \cdot k \cdot (r-r_m)} dS,
\]

(C.49b)

while the transmit patterns are as indicated in (D.12) and (D.16).

In summary, this means that for basis functions on the PEC, two patterns are stored (C.48), while for the electric and magnetic basis functions on the dielectric, only one pattern (C.49) in each region is stored.
In this appendix, we introduce the key concepts and notions used when considering the Fast Multipole Method. Presenting the FMM in a structured manner is inherently difficult because of the many different variables involved. We have chosen to follow the approach of the seminal paper by Coifman et al. [10], where the quantities are first described in an informal way, postponing the discussion of how they are found to later sections. We ask for the reader’s patience in this regard.

As discussed in Appendix C, when solving a scattering problem using the Method of Moments, we arrive at a discretized linear system of the form

\[ \overline{Z} \overline{I} = B, \]

and solve to find the coefficients \( \overline{I} = \overline{Z}^{-1} B \), which are coefficients to the basis functions expressing the induced surface current density. Discretizing (C.15), an element of the matrix \( \overline{Z} \) can be expressed as

\[ Z_{m,n} = j \frac{\omega \mu}{4\pi} \int_S \int_{S'} \left( \mathcal{I} - \frac{1}{k^2} \nabla \nabla' \right) \frac{e^{-jk|\mathbf{r} - \mathbf{r}'|}}{|\mathbf{r} - \mathbf{r}'|} \cdot f_m(r') dS' dS. \]

In words, we are testing the \( m \)'th basis function against the \( n \)'th basis function by integrating them over their domains (\( S' \) and \( S \), respectively) and letting their interaction be represented by the dyadic Green’s function.
FMM for time-harmonic electromagnetic fields rests on two main identities, required to express the Green’s function outside some region as a rapidly converging series. The key identity is that the scalar Green’s function can be expressed as a series by manipulating the Gegenbauer addition theorem (see App. F.3 for details) to yield

$$e^{-jk\|x+d\|}/\|x+d\| = jk \sum_{l=0}^{\infty} (-1)^l (2l + 1) j_l(k \|d\|) h_l^{(2)}(k \|x\|) P_l(\hat{d} \cdot \hat{x}).$$

This expansion is valid provided that \(\|x\| > \|d\|\). Thus, in a sphere of radius \(\|d\|\) centered at \(x\), we can express the Green’s function as the above series, noting that the smaller \(\|d\|\) is relative to \(\|x\|\), the faster the convergence.

Further, part of the terms in the sum can be expressed as propagating plane waves through the identity

$$4\pi (-j)^l j_l(k \|d\|) P_l(\hat{d} \cdot \hat{x}) = \iint e^{-jk \cdot d} P_l(\hat{k} \cdot \hat{x}) d^2 \hat{k}.$$  

Inserting this into (D.3), we avoid the terms that depend on \(d\) aside from the phase-center movement \(e^{-jk \cdot d}\), at the price of having to do an integral over the surface of the sphere. An integral over the surface of a sphere is expressed throughout the thesis as \(\iint d^2 \hat{k}\), where

$$\hat{k} = \sin \theta \cos \phi \hat{x} + \sin \theta \sin \phi \hat{y} + \cos \theta \hat{z},$$

such that

$$\iint d^2 \hat{k} = \int_0^{2\pi} \int_0^\pi \sin \theta \ d\theta d\phi.$$  

We also note that \(k = k \hat{k}\).

Combining (D.4) into (D.3) yields

$$e^{-jk\|x+d\|}/\|x+d\| = \frac{k}{4\pi} \sum_{l=0}^{\infty} (-j)^l (2l + 1) h_l^{(2)}(k \|x\|) \iint e^{-jk \cdot d} P_l(\hat{k} \cdot \hat{x}) d^2 \hat{k}.$$  

By truncating the sum, we can exchange the order of integration and summation. Note that by truncating the sum, we limit the accuracy of the expansion, but as we will see later, this can be done in a controlled manner.

Simplifying (D.7), we arrive at the expression

$$e^{-jk\|x+d\|}/\|x+d\| \simeq -jk \iint e^{-jk \cdot d} T_L(k, x) d^2 \hat{k},$$  

where \(T_L(k, x)\) is the propagator of the scattered field.
Figure D.1: Illustration of the quantities in the expansion (D.8) of the Green’s function. Shown in dashed is the region of validity of the translation function $T_L$.

where

$$T_L(k, x) = \sum_{l=0}^{L} (-j)^l (2l + 1) h_l^{(2)}(k \|x\|) P_l(\hat{k} \cdot \hat{x})$$

is the so-called translation function, also called Rohklin’s translation operator. The quantities involved are illustrated in Figure D.1. A key feature of $T_L$ is that it is independent of $d$. This allows us to precompute and then reuse it between any pair of groups, with radius at most $\|d\|$, separated by $x$. We note that the choice of $L$ is governed by the desired accuracy and the size of the groups, and that it would have to be extremely large\(^1\) to ensure convergence for $\|d\| \approx \|x\|$. Another key feature is that $T_L$ is diagonal, in the sense that for a specific direction $k$, $T_L$ yields a scalar.

Continuing in the scalar domain, we can thus approximate the field strength at $r_m$ from a point source located at $r_n$ as

$$e^{-jk\|r_m - r_n\|/\|r_m - r_n\|} \approx -\frac{jk}{4\pi} \int \int e^{-jk \cdot (r_m - r_m')} T_L(k, x) e^{-jk \cdot (r_n' - r_n)} d^2\hat{k},$$

where $x = r_{m'} - r_{n'}$.

The process is illustrated in Figure D.2. From the illustration, it is clear that the FMM approximation is simply just a decomposition of the vector $r_m - r_n$ between source and observation point. Obeying certain rules of this decomposition, namely that $\|r_m - r_{m'}\| < \|r_{m'} - r_{n'}\|$ and correspondingly for $\|r_{n'} - r_n\|$, the series converges.

---

\(^1\)We remark that while the Gegenbauer addition theorem (D.3) is uniformly convergent, the series (D.9) in (D.8) diverges for $L \to \infty$, and so, one cannot always expect that an increase of $L$ leads to a more accurate approximation in (D.8).
D.1 Application to Integral Equations

We now consider the use of FMM for the Electric Field Integral Equation (EFIE) and the Magnetic Field Integral Equation (MFIE). In the previous section, an expression \( (D.8) \) for the Green’s function was found based on Rokhlin’s translation operator. Inserting this into the expression for the matrix elements in the EFIE \( (D.2) \), we get

\[
Z_{m,n} \simeq \frac{\omega \mu}{4\pi} \iint_{S'} \left[ f_m(r') \cdot \left[ \mathbf{I} - \mathbf{k}\mathbf{k} \right] e^{-j\mathbf{k}\cdot(r'-r_m)} dS' \right] \cdot \mathbf{T}_L(k, r_m - r_n) \cdot \left[ f_n(r) \cdot \left[ \mathbf{I} - \mathbf{k}\mathbf{k} \right] e^{j\mathbf{k}\cdot(r_n - r)} dS \right] d^2k.
\]

(D.11)

Comparing with the expression \( (D.8) \), we see that \( x = r_m - r_n \) and \( d = r' - r_m + r_n - r \). Thus, to keep the region of validity small and thus keep \( L \) small, \( r_m \) should be chosen to minimize the maximum value of \( ||r' - r_m|| \) attained on \( S' \) (and similarly for \( r, r_n \) and \( S \)).

We note that the formula \( (D.11) \) is only applicable provided that \( ||r_m - r_n|| \) is sufficiently large, with \textit{sufficiently large} to be defined later. If \( ||r_m - r_n|| \) is too small, the interaction cannot be approximated by FMM, and the interaction is instead computed as usual in Method of Moments \( (C.31) \) and stored in a sparse near-matrix \( \mathbf{Z}_{\text{near}} \).
The integrals

\[ V_m(k) = \int_S f_m(r) \cdot \left[ \mathbf{T} - \hat{k}\hat{k} \right] e^{-jk(r_m-r)} dS, \]  
\[ R_m(k) = \int_S f_m(r)e^{-jk(r-r_m)} dS, \]

are typically termed the \textit{basis function patterns} or, respectively, the \textit{transmit} and \textit{receive} pattern of the basis function \( f_m \). The term

\[ \mathbf{T} - \hat{k}\hat{k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \hat{\theta} + \hat{\phi} \]

since \( \hat{k} = \hat{r} \). Therefore, if the transmit pattern is expressed in spherical components, only the two components \( (\hat{\theta}, \hat{\phi}) \) are needed of both receive and transmit patterns.

Using (D.13), we can restate (D.11) as the more generic

\[ Z_{m,n} \simeq \kappa \int \int \mathbf{R}_m(k) \cdot (T_L(k, x) \mathbf{V}_n(k)) d^2\hat{k}, \]

where the transmit pattern \( \mathbf{V}_m = \mathbf{R}_m^* \) due to the symmetry of the EFIE operator. \( \kappa \) is a scaling factor, \( \kappa = -\frac{\eta}{4\pi} \) for EFIE.

Using (D.15), we can test the electric field generated from basis functions on a patch that is far enough away against all the testing functions in a testing cluster. The strength of FMM, and source of its savings, is thus that interactions between entire groups of basis functions can be evaluated using the same translation operator.

For the MFIE, the transmit and receive patterns are, respectively,

\[ V_m(k) = -\hat{k} \times \int_S [f_m(r) \times \hat{n}(r)] e^{-jk(r_m-r)} dS, \]
\[ R_m(k) = \int_S f_m(r)e^{-jk(r-r_m)} dS. \]

Since

\[ \hat{k} \times \begin{bmatrix} \hat{r} & \hat{\theta} & \hat{\phi} \end{bmatrix} = \begin{bmatrix} 0 & \hat{\phi} & -\hat{\theta} \end{bmatrix}, \]

only the \( (\hat{\theta}, \hat{\phi}) \) components of \( \mathbf{V}_m \) are needed, and due to the dot product with \( \mathbf{V}_m \), this also applies to \( \mathbf{R}_m \). Note that \( \kappa = -\frac{\eta}{4\pi} \) for MFIE, which differs by a factor of \(-j/k\) from the factor from EFIE.
As is apparent from the expressions (D.11), we need a grouping scheme that dictates the group centers \( r_m \) for each basis function. The most common choice is the Octree scheme, a data structure conceptually similar to a tree. It works by forming an initial square bounding box for the scatterer and then creates additional levels to the tree by successively halving the bounding box until some lower threshold. The patches are then positioned into boxes on each level according to the center points of the patches, and empty boxes are pruned during creation. We will return to this topic in Section D.3.

D.2 Error Sources

There are two main error sources in the expression (D.11), which is the fundamental equation for the FMM. The first is the quadrature error, due to the numerical integration over the sphere. The second is the truncation error, owing to the finite number of terms \( L + 1 \) in (D.9).

We note that, strictly speaking, there is also a third error source, namely the integration error when computing the basis function patterns (D.13). This rarely discussed in the literature, since it can be done analytically for flat patches. For curved quadrilaterals, the integral has to be computed numerically. However, it is well-behaved, and thus it is easy to control the error and it only requires a modest amount of computing time.

D.2.1 Integration Error

The outer integral (over the sphere) in (D.15) is calculated using a quadrature rule, such that the discretized version of (D.15) is

\[
Z_{m,n} \simeq \kappa \sum_{p=1}^{K} w_p \mathbf{R}(k_p) \cdot (T_L(k_p)\mathbf{V}(k_p)),
\]

(D.19)

where \( K \) is the number of tabulated directions, and \( k_p \) is the wave vector corresponding to the \( p \)th direction. Determining the ideal integration weights \( w_p \), i.e., the quadrature rule yielding the lowest \( K \) for a given accuracy, requires some analysis.

Rewriting (D.15) to perform numerical quadrature, we get

\[
Z_{m,n} \simeq \kappa \int_0^{2\pi} \int_0^{\pi} \mathbf{R}_m(k) \cdot T_L(k, x)\mathbf{V}_n(k) \sin \theta \, d\theta d\phi,
\]

(D.20)
where we note that the jacobian element of the transformation to polar coordinates is $d^2k = r^2 \sin \theta d\theta d\phi$, and since we are integrating on the unit sphere, $r^2 = 1$. To avoid the $\sin \theta$ term, which is not band-limited, the substitution $t = \cos \theta$ is introduced, with $\int_0^\pi \sin \theta d\theta = -\int_1^{-1} dt$, yielding

$$Z_{m,n} \simeq \kappa \int_0^{2\pi} \int_{-1}^1 R_m(k) \cdot T_L(k, x) V_n(k) \, dtd\phi. \tag{D.21}$$

We thus see that the integrand is a product of three separate functions, each with their own distinct bandwidth. As is well-known the bandwidth of such a product is the sum of the bandwidths of the individual functions. The bandwidth of the basis function patterns is roughly $L/2$, while the bandwidth of $T_L$ is $L$ [38, pp. 92–98]. We note that the term bandwidth here refers to the order of the spherical harmonics expansion required to represent it, see Koc et al. [65].

The integration error of (D.21) is discussed in Koc et al. [65, pp.913–914], where it is noted that it can be integrated exactly by choosing $L + 1$ Gauss-Legendre (GL) points in $\theta$. This is an unfortunate typo, it should be $L + 1$ GL points in $t$. For the $\phi$ integration, we need $2L + 2$ equidistant points in $[0, 2\pi - 2\pi/(2L + 2)]$, i.e., the trapezoidal rule. This yields a total of $K = 2(L + 1)^2$ directions. We remark that more effective approaches for integrating band-limited functions on a sphere have been suggested by McLaren [162] and Bažant & Oh [163], but these do not appear to have been used in MLFMM, likely because they are difficult to implement for arbitrary orders and because oversampling is needed anyway to ensure accurate interpolation as discussed in Section D.3.2 and [J4].

### D.2.2 Truncation Error

The upper limit $L$ in the translation function to achieve a desired accuracy is significantly harder to determine than the integration scheme. The addition theorem (D.3) is used as the foundation for deriving $L$. Quite some work has been done in this field, as discussed in Appendix F.4. Here, we recap the formula used in modern implementations, known as the Excess Bandwidth Function (EBF):

$$L = kD + 1.8\beta^{2/3}(kD)^{1/3}, \tag{D.22}$$

where $D$ is the diameter of the groups at the given level.

We note that the EBF is derived based on the assumption that $L < k \|x\|$, where $x$ is the translation vector. This means that we can now specify exactly what interactions have to be included in the near-matrix $Z_{\text{near}}$: If the distance $\|x\|$ between the group centers is less than or equal to $L/k$, then they have to be included in the near-matrix. We call groups that have their interactions
computed via FMM well-separated, while those that are too closely positioned are called non separated.

However, in practice, the one buffer-box criterion is used instead to determine which interactions are well-separated. The one buffer-box relaxes the criterion $\|x\| \leq \frac{L}{k}$ to

$$\|x\| \leq D.$$  \hspace{1cm} (D.23)

In practice, this significantly reduces the size of the near-matrix while yielding acceptable accuracy. It is, however, important to note that for those interactions where $\frac{L}{k} > \|x\| > D$, we are actually not guaranteed the desired accuracy. As Appendix F.5 details, it turns out that this is particularly a problem for small groups, i.e., small $D$.

### D.3 The Step to Multilevel

The algorithm considered until now, FMM, has a single, significant drawback. MoM requires the interaction between every basis function with every other basis function; FMM has reduced this to interactions between entire groups, provided that they are far enough apart. However, as the problems increase in size, even the interaction between every pair of groups becomes a bottleneck.

Remembering that the concept of the FMM is to gather contributions from well-separated groups into coming from the same point (the center), we can continue this argument in a hierarchical manner, thereby yielding the Multi-Level Fast Multipole Method (MLFMM). In other words, instead of letting a large number of small groups interact with each other, we let larger groups interact with larger groups, thereby reducing the number of interactions.

#### D.3.1 Grouping

For this, we apply a hierarchical Octree grouping of the patch center points. This yields $q$ levels, with a number of boxes at each level. Level 1 constitutes the square bounding box for the entire point cloud while level 2 is the first partitioning, yielding a maximum of 8 boxes on level 2. Empty boxes are not considered. The grouping of a scatterer is illustrated in Figure D.3.

Beginning at level 3, for each box, we check the distance $\|x\|$ to groups which were "near" on the previous level against the criterion (D.23), where $D$ equals the group diameter on the current level.
Figure D.3: Illustration of a Octree with $q = 4$ levels in 2 dimensions. Note that, for illustrative purposes, the bounding box is slightly larger than the scatterer.
D.3.2 Interpolation & Anterpolation

To perform interactions between groups that are well-separated, we apply (D.15). However, this immediately yields the question of how to find \( V_n(k) \) for larger groups. Consider the interaction between basis functions \( f_1 \) and \( f_2 \), residing in groups \( m \) and \( n \) on the finest level \( q \), as illustrated in Figure D.4. In a single-level scheme, the corresponding EFIE matrix element would be calculated as

\[
Z_{2,1} \simeq \kappa \iint R_2(k) \cdot T_L(k, r_n - r_m) V_i(k) d^2 \hat{k}, \quad (D.24)
\]
with

\[ V_1(k) = \int_S f_1(r) \cdot \left[ \mathbf{1} - \hat{k} \hat{k} \right] e^{-jk \cdot (r_m - r)} dS, \quad (D.25) \]

\[ R_2(k) = \int_S f_2(r) e^{-jk \cdot (r - r_n)} dS. \quad (D.26) \]

However, since their parent groups on level \( b = q - 1 \) are also well-separated by the one buffer-box criterion \((D.23)\) (their boxes do not touch), we can instead compute their interaction as

\[ Z_{2,1} \simeq \kappa \int \int R^b_2(k) \cdot T_L(k, r^b_m - r^b_n) V^b_1(k) \, d^2 \hat{k}, \quad (D.27) \]

with

\[ V^b_1(k) = \int_S f_1(r) \cdot \left[ \mathbf{1} - \hat{k} \hat{k} \right] e^{-jk \cdot (r_m^b - r)} dS, \quad (D.28) \]

\[ R^b_2(k) = \int_S f_2(r) e^{-jk \cdot (r - r_n^b)} dS. \quad (D.29) \]

The vector \( r^b_m - r^b_n \) has been marked as \( T_L \) in the figure. The superscript \( b \) denotes quantities on level \( b \).

An immediate question is then how to obtain \( V^b_1 \). If we have to store \( V_1 \) on all levels for every basis function, it would take up too much storage. Specifically, since \( D_b > D_a \) then \( L_b > L_a \) and therefore \( K_b > K_a \). Not only will we thus need more tabulated directions on level \( b \) compared to level \( q \), there isn’t even a guarantee that a subset of the directions on level \( b \) are available at level \( q \) due to the irregular distribution of points.

Further, on level \( b \), we need the center of the expansion to be \( r^b_{b,a} \).

This problem is solved by applying interpolation and phase-center movement. We can find the necessary basis function patterns on the \( b \)'th level as

\[ V^b_1(k) = e^{-jk \cdot (r^b_m - r_n)} W(V^b_1(k)). \quad (D.30) \]

\( W \) is an interpolation operator, mapping from \( \mathbb{C}^{K_b} \to \mathbb{C}^{K_b} \) in such a way that \( V^b_1(k) \) is equal to that obtained by setting \( r_m = r^b_m \) in \((D.13)\). We note that the order of interpolation and phase-center shift (multiplication with \( e^{jk \cdot r} \)) is important, as the phase-center shift above increases the bandwidth of the function, making interpolation less accurate. There are several possible interpolation methods in MLFMM, see Gibson \[37\], Section 8.5.3 or Chew \textit{et al.} \[38\]. Section \textit{3.4} discusses our research on this topic.
Similarly, after the translation, we need to integrate $T_L(k, r_n^b - r_m^b) V_1^b(k)$ against $R_3^b(k)$. Since we do not have $R_3^b$, but only $R_1^b$, we need to downsample $T_L(k, r_n^b - r_m^b) V_1^b(k)$ in a way that preserves the integrand. More precisely, we need an operator $\tilde{W} \in \mathbb{C}^{K_b} \rightarrow \mathbb{C}^{K_q}$ such that

$$\iint T_L(k, r_n^b - r_m^b) V_1^b(k) d^2 k \simeq \iint \tilde{W} \{ T_L(k, r_n^b - r_m^b) V_1^b(k) \} d^2 k. \quad (D.31)$$

In practice, this is achieved by letting $\tilde{W}$ integrate the spectral components that cannot be represented at the sample rate $\mathbb{C}^{K_q}$, and then interpolate the rest. This process is typically called anterpolation or adjoint interpolation [164] (3.3)].

Having found a suitable operator, we perform

$$H(k) = \tilde{W} \left\{ e^{-jk \cdot (r_m^b - r_m^b)} T_L(k, r_n^b - r_m^b) V_1^b(k) \right\}. \quad (D.32)$$

Note that the phase-center shift is now performed before the operator is applied, contrary to (D.30), to keep the bandwidth of the function being in/an-terpolated as low as possible. Combining (D.30) and (D.32), we can now to perform (D.27) as

$$Z_{2,1} \simeq \kappa \iint R_2(k) \cdot H(k) \ d^2 k. \quad (D.33)$$

In summary, moving from single-level FMM to multi-level FMM, we trade the application of in/an-terpolation operators for a vast reduction of the number of translations and separate integrations. The process is illustrated in Figure D.5, where we note that the single-level procedure would require 4 translations for each of the 3 source groups, for a total of 12 translations and integrations against $R_2$. The multi-level approach requires 3 interpolations, 1 translation and 4 an-terpolations and integrations against $R_2$.

### D.4 Algorithm

We are now ready to outline the FMM algorithm. We begin by considering the setup phase, done prior to beginning the iterative solution, and in the next section we consider the procedure for computing the matrix-vector products. We note that references that describe the FMM algorithm in detail include Coifman et al. [10], Darvé [50] and Gibson [37] Chapter 8]

First, an Octree is generated with $q$ levels. We note that diameter of the box on the first level $D_1$ will equal the diameter of the minimum enclosing sphere of the structure in consideration. Then, $D_i = 2^{-i+1} D_1$, and from this we can compute the value of the truncation parameter $L_i$ on each level using (D.22).
Figure D.5: Illustration of interaction between groups in a multi-level scheme. The blue arrows constitute both interpolation and phase-shift, i.e., both processes involved in (D.30), while the magenta arrows similarly illustrate (D.32). The orange arrow shows a translation.
The finest level $q$ of the Octree will have a sidelength $\alpha_q = D_q/\sqrt{3}$ at least as large as the largest sidelength of any patch in the mesh.

Based on the Octree grouping, we move from the coarsest (level 1) to the finest level. On levels 1 and 2, all groups are near-groups of each other. On levels $3, \ldots, q$, we run through each group $g$ and check the distance to groups that are children of $g$’s parent’s near-groups. If these are now well-separated according to (D.23), we add the translation vector $\mathbf{x}$ between the group centers to a global list.

When we reach the finest level $q$, interactions between groups that are too closely spaced are computed through direct integration (cf. Appendix C.1, (C.31)) and placed in the sparse near-interaction matrix $\mathbf{Z}_{\text{near}}$.

For each group on the finest level, we now run through each basis function in that group, computing and storing the necessary basis function patterns (D.13). Then, we consider the global list of translation vectors. For each unique vector, we compute and store the translator $T_L$ at the required sample rate $K_i$, multiplying the integration weights $w_k$ (D.19) onto the translators if the anterpolation operator $\tilde{W}$ requires this.

We also run through each group on each level and preallocate storage for the total pattern coming from that group, i.e., the sum of the radiated patterns from all child groups (at the finest level, the total pattern is the sum of the radiated patterns from the basis functions). We further allocate storage for received patterns [13, Section 7].

Finally, for each level, we compute and store the maximum 8 possible phase-center shift terms $e^{-jk \cdot r}$ involved in (D.30) and (D.32) at the required sampling rates. We also compute and store necessary information for the interpolation and anterpolation operators, $W$ and $\tilde{W}$, respectively.

This completes the setup of the MLFMM, and we are now ready to compute matrix-vector products.

**D.5 Performing Matrix-vector Products**

To perform a matrix-vector product $\mathbf{B} = \mathbf{Z} \mathbf{I}$, we are given a vector of excitation coefficients $\mathbf{I} \in \mathbb{C}^N$. These coefficients are then multiplied onto the basis function patterns (D.13) and added together to yield the total radiated pattern from each
D.5 Performing Matrix-vector Products

This yields a total pattern from group $g$ of

$$
G^c = \sum_{i=1}^{N_g} V^c_i I_{1N}
$$

(D.34)

where the number of basis functions in group $g$ is $N_g$ and the global basis function index of function $i$ is $i_N$. $V^c_i \in \mathbb{C}^{K_q}$ is the $c$'th component, across all $K_q$ directions, of the basis function pattern (D.13) of function $i$. This step is called the aggregation step. As noted, we only use $\hat{\theta}, \hat{\phi}$ components, so $c = 1, 2$.

With the group patterns for all groups at the finest level computed, we move to the interpolation step. Traversing from the finest to the coarsest level in the Octree, starting at level $q - 1$, we gather all radiated patterns from each group’s children using (D.30). Thus, when this step is completed, we have computed and stored the aggregated patterns from all groups at all levels. Thus, group $g$’s pattern storage (for levels $q - 1, q - 2, ..., 3$) now contains

$$
\mathcal{G}^c = \sum_{i=1}^{G_g} P \ast \mathcal{W}\{G^c_i\}
$$

(D.35)

where $G_g$ is the number of child groups (at most 8) of group $g$. The vector $P$ of length $K_b$, where $b$ is the specific level, contains the relevant phase-center shift $\epsilon^{j k \cdot r}$ for each direction $k$. $\ast$ denotes element-wise multiplication.

For the translation step, we then perform the translations. At each level, we begin by moving the aggregated patterns of all groups to a global array, giving each group $g$ an index to that array to indicate where $g$’s pattern is located, and zeroing each groups pattern storage. Then the translations are performed, by multiplying the relevant $T_L$ onto every well separated groups pattern and aggregating the received patterns, such that each groups pattern storage now becomes the total received pattern. Thus, each groups pattern storage now contains

$$
\mathcal{G}^c = \sum_{i=1}^{F_g} T_L \ast G^c_i
$$

(D.36)

where $F_g$ is the number of well separated groups, $T_L$ is a vector of length $K_b$, $b$ being the level number. $G^c_i \in \mathbb{C}^{K_q}$ is $c$'th component of the aggregated pattern from the well separated group $i$.

On every level, the stored patterns $\mathcal{G} = [G^\theta \ G^\phi]$ now contain the received patterns from other groups. The anterpolation step then begins, where the Octree is traversed from level 4 to $q$. The patterns from the parent groups are anterpolated to the child groups by applying (D.32), being added to the patterns.
already present in the groups. Thus, group $g$ at level $b$ will have a pattern $\overline{G}$ that can be represented as

$$\overline{G}^c = \overline{G}^c + \mathcal{W} \left\{ \overline{P} \ast \overline{G}^c_{b-1} \right\}$$  \hspace{1cm} (D.37)

where $\overline{G}^c_{b-1}$ is the pattern from the parent group of $g$, and $\overline{P} \in \mathbb{C}^{K_{b-1}}$ is the relevant phase-center shift.

After the completion of the anterpolation step, the pattern storage $\overline{G}$ for groups on the finest level will contain the total received pattern from all other groups. The integration is then performed at the final \textit{disaggregation} step, finding the $i$'th component of the resulting vector $\overline{B} \in \mathbb{C}^N$, corresponding to basis function $f_i$ in group $g$, as

$$B_i = B_i + \sum_{c=1}^{2} \overline{G}^c \cdot \overline{R}^c_i$$  \hspace{1cm} (D.38)

where $c$ runs through the components, and $\overline{R}_i \in \mathbb{C}^{K_q}$ is the receiving pattern for basis function $i$. 
This appendix considers methods for efficiently solving a linear system
\[
\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{C}^{N \times N}, \quad \mathbf{x} \in \mathbb{C}^N, \quad \mathbf{b} \in \mathbb{C}^N. \tag{E.1}
\]

When solving a linear system of equations, there are two classes of methods, namely direct or iterative methods.

The direct methods aim to solve the system \( \mathbf{A}\mathbf{x} = \mathbf{I} \), or at least approximate its solution, by implicitly computing and applying the inverse \( \mathbf{A}^{-1} \). Typically, this is done via factorizations, the efficiency of which depend on the structure of \( \mathbf{A} \). The most general approach is the LU factorization. General direct methods for dense matrices scale as \( \mathcal{O}(N^3) \), which for large dense systems make them all but useless—their primary strength is that once the approximation to \( \mathbf{A}^{-1} \) is found, many right-hand sides can be solved with little effort. This is key to, e.g., RCS computations and therefore a significant amount of effort has been put into exploiting modern computer architecture to efficiently compute such factorizations even for relatively large problems \([165,167]\). However, the \( \mathcal{O}(N^3) \) scaling for the dense systems encountered in MoM means that such methods will never be able to compete with iterative methods in terms of the tractable size of problems. Some effort has gone into developing reduced scaling direct methods for MoM as discussed in Section 3.2.
Iterative methods solve the system by using approaches derived from optimization theory, reformulating the problem to

$$\min_{\bar{x}} \| \bar{A} \bar{x} - \bar{b} \|_2$$  \hfill (E.2)

The methods have the significant advantage of having the same asymptotic scaling with $N$ as applying the operator $\bar{A}$ to $\bar{x}$ — in other words, for MLFMM, these methods scale as $O(N_{\text{iter}}N \log N)$, where $N_{\text{iter}}$ scales significantly slower than $O(N)$ for most problems. One of the disadvantages, however, is that iterative methods generally only solve for one right-hand side at a time. Further, iterative methods can suffer from poor convergence, yielding a very long solution time.

This chapter will focus on iterative methods, primarily in the context of MLFMM. The lack of efficiency when dealing with multiple right-hand sides, is only considered briefly, while the convergence issues are the focal point of the material. First, however, we will briefly mention the most popular iterative solvers.

### E.1 Iterative Solvers

When considering iterative solvers, there are essentially two classes of methods. First, the stationary iterative methods, such as Jacobi and Gauss-Seidel, which are typically simple to implement but can have poor performance for general systems and will not be considered further. Second, Krylov subspace methods, which choose a basis of successive operator applications to some initial vector, and then minimize the residual over that basis. Krylov methods include GMRES \cite{105}, BiCG-STAB \cite{168}, (TF)QMR \cite{169}, CG \cite{170} and LSQR \cite{171}. A brief summary of the properties of these solvers when applied to some electromagnetic scattering problems is given in \cite{172}, and some performance analysis for HO basis functions is done in \cite[Chapter 4]{25}.

Ergül & Gürel \cite{135} go into more detail, discussing a number of iterative methods and comparing them for some scattering problems. Algorithms considered are GMRES, TFQMR, BiCG and its stabilized version BiCG-STAB, CGS and LSQR, with the latter two applied to the normal equations. The main conclusion is that GMRES is best for both CFIE and EFIE, but that LSQR is a good alternative to GMRES, since it uses less memory and converges acceptably. The use of LSQR, BiCG and BiCG-STAB requires the Hermitian of the operator $\bar{Z}$. This is achievable in MLFMM by interchanging the transmit and receive functions \cite[Section 3.4]{20}, but adds code complexity. Carpentieri et al. \cite{173} consider a new family of solvers for MoM. \cite{106,174,175} gives a good overview of iterative solvers and related topics for general linear systems.
E.2 Preconditioning

The key factor when considering the convergence of iterative methods is the distribution of the eigenvalues of the operator. Note that this concept is related to, but not the same as, the condition number of the operator. The latter expresses the ratio between the largest and smallest singular value and thus, informally, is only related to the end points of the spectrum. Thus, it is possible to have a fairly ill-conditioned operator in an iterative method, and still achieve reasonable convergence if the eigenvalues are fairly well-clustered.

The fundamental problem with the EFIE is that the integral operator is not well-behaved, due to one of the following three problems [176]:

1. The spectrum may accumulate at zero.
2. The operator may have an unbounded spectrum.
3. The operator may have small eigenvalues associated with resonances.

To improve the distribution of the eigenvalues, a preconditioner is used.

A preconditioner is an operator $\overline{M}$ applied to (E.1) from the left, yielding

$$\overline{M} A \overline{x} = \overline{M} \overline{b},$$

(E.3)

or, from the right,

$$A \overline{M} \overline{u} = \overline{b},$$

(E.4)

where $\overline{x} = \overline{M} \overline{u}$. In the following, we will only consider (E.3), but note that the FGMRES discussed in Section 3.6 is based on (E.4).

The basic concept is that $\overline{M} A$ should have significantly better convergence properties than $\overline{A}$, i.e., $\overline{M}$ should be a fairly good approximation to $\overline{A}^{-1}$. This naturally leads to most preconditioners being a tradeoff between the setup time required to compute $\overline{M}$ and the benefit gained from solving (E.3) instead of (E.1).

First, we will discuss algebraic preconditioners, where $\overline{M}$ is based on (parts of) the matrix $\overline{A}$. The mathematical literature is abound with thorough analysis of algebraic preconditioners in a wide range of scenarios. For MLFMM, however, these preconditioners are not as applicable as one might initially assume, since...
we only have the near-matrix available algebraically. Therefore, such preconditioners will only accelerate the solution of the “near-part” of the problem, and will not help with the parts of the spectrum of $\bar{A}$ that are due to the interactions between well-separated regions. This is sufficient for some problems, such as reflector antennas, but can cause slow convergence for more complicated scatterers. Therefore, this appendix concludes with a slightly longer discussion on the state-of-the-art analytical preconditioner for EFIE, the Calderon preconditioner, which is very promising, but appears to have some open problems remaining before it is usable for HO discretizations of arbitrary structures.

E.2.1 Diagonal

The simplest preconditioner is the diagonal preconditioner, where the matrix $\bar{M}$ is diagonal, with diagonal elements

$$M_{i,i} = \frac{1}{Z_{\text{near},(i,i)}}$$  \hspace{1cm} (E.5)

This is generally an ineffective preconditioner, except for extremely diagonally dominant systems. However, it has the advantage of being extremely simple to implement, very fast both to compute and to apply, and can be implemented without requiring any memory at all.

In extension of this, a popular preconditioner is the block-diagonal preconditioner. Here, the blocks on the diagonal, from self-patch interactions, are LU factored separately, and their inverse applied. While costing more time and memory to set up than the diagonal preconditioner, the block-diagonal preconditioner is much more effective and still fast to apply. Song et al. [47], an early application paper on MLFMM, used a block-diagonal preconditioner.

E.2.2 ILU

The incomplete LU (ILU) and its variants such as ILU($k$) and ILUTP constitute a much more thorough attempt at achieving $\bar{M} = \bar{A}^{-1}$ by considering an incomplete factorization of $\bar{A}$. The term *incomplete* refers to a sparsity-preserving factorization. See Saad [119, Section 10.3] for a thorough discussion. The variant called ILU(0) yields a sparse $\bar{M}$ with the same sparsity pattern (and thus even lower memory requirements, since the memory for the indices can be reused) as $\bar{Z}_{\text{near}}$, and it is a significantly better preconditioner than the diagonally-based ones.
Unfortunately, ILU has a number of drawbacks. Aside from being very costly to compute and store, it can potentially break down. Furthermore, ILU is not straight-forward to parallelize, a significant drawback in modern computer systems. Saad \cite[Section 12.6]{119} demonstrates a fairly efficient parallelization, based on the results in Ma & Saad \cite{116}, but it is still not well-suited for parallelization.

The costly computation and storage can be alleviated by applying the ILUTP variant, which drops elements smaller than a threshold during computation, and also pivots the factorization to attempt to avoid the potential breakdown, although it cannot be guaranteed to do so. And ILUTP is still not efficient in parallel.

### E.2.3 Sparse Approximate Inverse

Benzi & Tůma \cite{177} is a thorough discussion of Sparse Approximate Inverse (SPAI) preconditioning. In SPAI, the preconditioner is computed by solving the optimization problem

\[
\min \| \bar{M} \bar{Z}_{\text{near}} - \bar{F} \|_F \tag{E.6}
\]

where we stress the use of the Frobenius norm. The use of the Frobenius norm allows the problem to be solved by considering each row separately:

\[
\forall i, \min \| \bar{M}_i \bar{Z}_{\text{near}} - \bar{e}_i \|_2 \tag{E.7}
\]

where \( \bar{e}_i \) is the \( i \)th cartesian unit vector. This means that the SPAI preconditioner is embarrassingly parallel. Further \cite{123}, it is possible to use thresholding on both the elements in \( \bar{M} \) and \( \bar{Z}_{\text{near}} \) to yield acceptable memory use of \( \bar{M} \), i.e., significantly lower than \( \bar{Z}_{\text{near}} \).

### E.3 Calderon Preconditioning

Calderon preconditioning for the EFIE was introduced by Christiansen & Nédélec \cite{178}, and is currently being explored by several research groups as an effective cure for some of the convergence problems in EFIE, particularly when used with MLFMM, where the lack of information regarding the spectrum ”outside” the near-matrix is the fundamental limit for the algebraic preconditioners.

Calderon preconditioning is an analytical preconditioner since it acts on the analytical representation of the operators involved. It is based on the so-called
Calderon identity, derived in Hsiao & Kleinman [179, Section 4], concerning the EFIE operator [C.16] $L$ when applied twice:

$$L(L(J)) = \frac{-1}{4} J + \mathcal{V}(J),$$

(E.8)

where $\mathcal{V}(J)$ is the integral operator

$$\mathcal{V}(J) = \hat{n} \times \nabla \times \int_S G(r, r')J(r')dr'$$

(E.9)

This equation thus illustrates that when the operator $L$ is applied twice, the eigenvalues will lie around $-0.25$, with the spectrum governed by the operator $\mathcal{V}$, which is compact on smooth surfaces [92]. We note that while this expression is only valid for closed surfaces, numerical experiments show that the beneficial properties are also present in the open surface case [92].

Thus, Calderon preconditioning means that we apply the EFIE operator to the EFIE once again, solving the following integral equation instead of the EFIE (C.15):

$$L(L(J)) = -\frac{j}{\omega \mu} \mathcal{E}^i$$

(E.10)

For convenience when discussing Calderon preconditioning, one normally splits up the EFIE operator into two separate components, such that

$$L(J) = L_s(J) + L_h(J),$$

(E.11)

where

$$L_s(J) = \frac{-jk}{4\pi} \hat{n} \times \int_S e^{-jk|r-r'|} |r-r'| J(r')dS'$$

(E.12)

$$L_h(J) = \frac{j}{4\pi k} \hat{n} \times \int_S \nabla' e^{-jk|r-r'|} |r-r'| \nabla' S \cdot J(r')dS'$$

(E.13)

Where the subscripts $s$ and $h$ refer to the singular and hypersingular kernels that they possess. We note that this notation is adopted from [92] and [180]. In the typical notation of integral equations, a kernel $\int_{\mathbb{R}^n} \frac{1}{r^a}$ is called weakly singular if $a < n$, strongly singular if $a = n$ and hypersingular if $a > n$. By this notation, $L_s$ would be weakly singular and $L_h$ would be strongly singular, not hypersingular. However, we will continue with the standard notation in the field.

With this notion, we can describe the double EFIE operator as

$$L^2(J) = L_s^2(J) + L_h^2(J) + L_s L_h(J) + L_h L_s(J)$$

(E.14)
and it turns out that part of the explanation behind the well-behaved spectrum of the $L^2$ operator is that the double hypersingular operator, $L^2_h$, vanishes.

However, this property ($L^2_h(J) = 0$) is quite hard to satisfy in the discretized expression for $L^2$. One has to be very careful in choosing a basis that ensures that the double hypersingular operator is zero.

The main complication arises from the fact that the outer $L$ operator discretizes $\hat{n} \times \text{EFIE}$. The cross-product with the normal vector thus means that to discretize $L^2$, we need curl-conforming basis functions instead of the normal divergence-conforming functions (see [92] and the discussion on p. 133). To keep using the same basis functions that we already use for the interior part, a typical approach is thus to express the problem as

$$Z^{\text{CMP}} I = V^{\text{CMP}} \quad \text{(E.15)}$$

where

$$Z^{\text{CMP}} = Z_C G_{D,C}^{-1} Z_D \quad \text{(E.16)}$$
$$V^{\text{CMP}} = (Z_C G_{D,C}^{-1}) V_D \quad \text{(E.17)}$$

and

$$[G_{D,C}]_{i,j} = \langle \hat{n} \times f_i, \tilde{f}_j \rangle \quad \text{(E.18)}$$

Thus we have two function sets: $D$, which are divergence-conforming (see the discussion in Section C.1.2) basis functions $f$ as used in the discretization of the EFIE (C.15), and $C$, which are curl-conforming basis functions $\tilde{f}$. The intuitive choice would be $\hat{n} \times f = \tilde{f}$, however this is not possible because it renders the Gram-matrix $G_{D,C}$ singular.

The above-mentioned approach is called Calderon Multiplicative Preconditioning (CMP). Here, the key word *multiplicative* refers to the multiplication by one or more additional matrix or matrices onto the left and right-hand sides. This makes it significantly easier to implement than approaches that are not multiplicative and thus require specific discretizations, since users can keep on using their existing basis functions and indeed most of their code. Unfortunately, CMP requires basis function families that are curl-conforming and where the Gram-matrix between the curl-conforming and divergence-conforming basis function families is singular.

[181] lists three criteria which must be fulfilled in order for (E.15) to be well-conditioned, i.e., inherit the properties of the continuous (E.8):

1. The functions $f$ and $\tilde{f}$ have to be divergence-conforming
2. The Gram matrix $\bar{G}_{D,C}$ has to be well-conditioned.

3. The function spaces to which $f$ and $\tilde{f}$ belong have to ensure the cancelation of $L_h^2$.

Since in practice the functions $f$ are already fixed by whatever basis functions are used in the present implementation of MoM or MLFMM, the question thus becomes how to find $\tilde{f}$. As mentioned, requirement 2 is not fulfilled by choosing $f = \tilde{f}$, as proven in detail in \cite[Section 3.1]{178}.

It is important to note that several of the earlier papers on the subject avoided the tricky discretization by explicitly leaving out the double hypersingular operator, instead discretizing each of the other operator products by separate techniques. However, this is problematic for three key reasons, as detailed in \cite{92}:

1. It increases the computational costs by requiring several matrix-vector products
2. It introduces errors because the discretization employed would yield a non-zero result from $L_h^2$
3. It means that one cannot use the original MoM matrix from a present implementation, but instead have to create an entirely new discretization strategy.

### E.3.1 Literature survey

The first focused paper on Calderon preconditioning was the short paper by Adams & Brown \cite{182}, based on Adams’ work in his Ph.D. thesis \cite{183}. The Calderon identities were described in more detail in the earlier paper by Hsiao & Kleinman \cite{179}, but this paper focused on the theory and on proving (E.8), while \cite{182} seems to be the first to actually include numerical examples.

Unfortunately, while Adams’ work on the subject, including his later papers \cite{184} and \cite{180}, were very educational and highlighted several interesting properties and key implementation issues, they did not lead to widespread adoption of the method. A key reason for this was that they were not multiplicative, but instead required an entirely new discretization strategy due to assuming, rather than ensuring, that $L_h^2 = 0$. None-the-less, their numerical examples highlighted the potentially impressive gains to be achieved by using Calderon preconditioning.
It was, however, clear that further theoretical work remained before a solid implementation could be made. Contopanagos et al. [176] provided some of this, based on earlier theoretical work by the authors in various technical reports, but the work by Christiansen & Nédélec [178] seems to be the most important work on the continuous formulation of the identity. The thorough work, summarizing the work done in several earlier manuscripts that are only available in French, yields several interesting insights. This includes showing how the spectrum does not depend on the discretization density, and why choosing the intuitive $\tilde{f} = \hat{n} \times f$ is a bad idea.

It wasn’t until 2007 that a scheme suitable for an actual implementation of a Calderon Multiplicative Preconditioner, without the potential ailments discussed in [178], was presented by Buffa & Christiansen [185]. Here, the set of basis functions known as Buffa-Christensen (BC) are shown to fulfil the previously mentioned requirements in conjunction with the RWG functions. While somewhat fiddly due to requiring a slightly different mesh, it enabled an actual implementation of a CMP and can thus only be considered as a major breakthrough.

An actual implementation of a CMP including numerical examples and thorough description of the manipulations required to enable the CMP in conjunction with RWG functions was presented by Andriulli et al. [92]—this was a very detailed paper, including several test-cases and descriptions of all the adjustments both for closed and open scatterers as well as non-uniform meshes. [92] can thus be seen as a true enabler in the field, opening the door to a significant amount of research, a process that is still ongoing.

The natural extensions of the CMP to the CFIE and to an implementation in MLFMM was done by Bagci et al. [186] and Peeters et al. [187], respectively, while the implementation for transient fields was done by Cool et al. [188]. A CMP for penetrable bodies was presented in [189]. All of these papers are by the same group of people, and surprisingly few papers have been published by people outside this group. [190] and [191] are two conference papers, concerned with CMP for MLFMM, while [192] discusses CMP for curvilinear triangular patches, as does [193], [194] discusses CMP in a few scenarios.

As previously mentioned, an unfortunate requirement of implementing a CMP for a given basis function set is that one has to come up with another set while adhering to some very specific rules. Thus, there is little research into CMP for higher-order basis functions, since it would require devising an entirely new set of basis functions. [181] describes some theoretical work, by the previously mentioned group, but it would still require significant work to adapt to the Higher-Order Legendre basis functions used in this thesis. Further, very few
results have been presented for high-frequency problems, since the focus has been on the low-frequency and dense-discretization breakdowns of the EFIE.

In summary, while Calderon preconditioning appears promising, it is not yet mature enough to implement in our solver.
This appendix contains a number of miscellaneous details and secondary points to the discussion in the main body of the thesis.

F.1 Implementation of SHE

In this appendix, we consider the implementation of the Spherical Harmonics Expansion (SHE) storage of the basis function patterns, as discussed in Section 3.3.

During setup, after having computed the basis function pattern \((D.12)\) for a given function \(f_j\), the corresponding SHE has to be computed and stored. This is fast due to the orthogonality \((3.3)\) of the spherical harmonics, such that each of the SHE coefficients \(p_{pq}^j\) based on the pattern \(V(\hat{k})\) can be found by performing

\[
p_{pq}^j = \iint V(\hat{k})Y_{pq}^*(\hat{k})d^2\hat{k},
\]

where we note that here, \(V\) is represented by its cartesian components.

During the matrix-vector product, a few changes have to be made to the aggregation and disaggregation steps.
For the aggregation step, (D.34) is split up into two steps. The first step yields a set of SHE coefficients $p_{pq}^g$ for each of the $N_g$ basis function in group $g$, such that

$$ p_{pq}^g = \sum_{i=1}^{N_g} p_{pq}^i I_{1N_i}, \quad (F.2) $$

where we note that $p_{pq}^g$ does not conform to the symmetry relations (3.7).

Based on the $p_{pq}^g$ coefficients, the total group pattern $G^c$ for each $c$ component is computed as

$$ G^c = \sum_{p=0}^{W} \sum_{q=-p}^{p} p_{pq}^g Y_{pq} \quad (F.3) $$

where $Y_{pq}$ is a row of length $K$ containing the values of the spherical harmonic of degree $p$ and order $q$ at each of the $K$ directions. $I_{\theta\phi}$ is an operator that maps from $(\hat{x}, \hat{y}, \hat{z})$ to $(\hat{\theta}, \hat{\phi})$ components.

After continuing with the interpolation, translation and anterpolation steps as described in Section D.5, the disaggregation is modified since we can no longer integrate as in (D.38) since we do not have the receive pattern $R$ stored in the $K$ points.

Instead, we exploit the orthogonality of the Spherical Harmonics. We represent the incoming group patterns $\overline{G} = [\overline{G}^\theta \overline{G}^\phi]$ from (D.37) as

$$ \sum_{p=0}^{W} \sum_{q=-p}^{p} q_{pq}^g Y_{pq}(\theta, \phi) = I_{\theta\phi} \overline{G}. \quad (F.4) $$

Then, we perform

$$ B_i = B_i + \sum_{p=0}^{W} \sum_{q=-p}^{p} p_{pq}^i \cdot q_{pq}^{o*} \quad (F.5) $$

to find the component of the resulting vector $\overline{B}$ corresponding to basis function $f_i$ in group $g$, equivalent to (D.38).

### F.2 Theoretical Considerations on Combining FMM with HO

This section offers a brief look into the amount of memory used in an MLFMM implementation, comparing the memory required for LO MLFMM with that
required for HO MLFMM. In the following three sections, theoretical scenarios are considered, using quantities established from other places in the thesis or in the literature.

F.2.1 Values of $L$ and $K$

Here, we consider the MLFMM performance for a HO discretization, based purely on the values of the truncation parameter $L$ and the corresponding sample rate $K$.

First, a few early notions are set. The number of unknowns in RWG, $N_{\text{RWG}}$, is approximately four times greater than that used when applying HO Legendre functions, such that $N_{\text{RWG}} \approx 4N_{\text{HO}}$. Further, a typical group diameter in RWG-FMM is approximately $D_{\text{RWG}} = 0.3\lambda$. For illustrative purposes, we let the group-diameter in HO be flexible, such that $sD_{\text{RWG}} = D_{\text{HO}}$. We note that a typical value of $s$ could be 8.

From here, we can look at the ratio between the $L$’s for each of the two scenarios, based on (D.22):

$$\frac{L_{\text{HO}}}{L_{\text{RWG}}} = \frac{kD_{\text{HO}} + 1.8\beta^{2/3}(kD_{\text{HO}})^{1/3}}{kD_{\text{RWG}} + 1.8\beta^{2/3}(kD_{\text{RWG}})^{1/3}}$$  \hspace{1cm} (F.6)

$$= \frac{ksD_{\text{RWG}} + 1.8\beta^{2/3}(ksD_{\text{RWG}})^{1/3}}{kD_{\text{RWG}} + 1.8\beta^{2/3}(kD_{\text{RWG}})^{1/3}}$$  \hspace{1cm} (F.7)

Now, varying $s$ and keeping $\beta = 3$, Figure F.1 illustrates values of $L_{\text{HO}}$ and $K_{\text{HO}} = 2(L_{\text{HO}} + 1)^2$ relative to $L_{\text{RWG}}$ and $K_{\text{RWG}}$, respectively.

Obviously, as Figure F.1 reflects, increasing the group size significantly increases the value of $K$. The main consideration is whether this increase in $K$ leads to an increase in memory that outweighs the reduction in number of unknowns. Noting that the total number of elements stored for all basis function patterns is $2NK$, we unfortunately see that for $s \gtrsim 6$, where $N_{\text{RWG}} \gtrsim 4$, $K_{\text{HO}}/K_{\text{RWG}}$ is much larger than 4 and thus the increase in $K$ from switching to HO basis functions is dominates the decrease in $N$ in a standard MLFMM implementation.

While the basis function patterns is only a part of the total memory consumption, it is an important part, and the purely theoretical conclusion here helps to explain why the overall research consensus has been that increasing the patch size to allow HO discretizations is a bad idea in MLFMM.
We continue with our theoretical considerations on the behaviour of a standard HO MLFMM implementation by considering a test case where the memory consumption is fairly easy to analyze.

Consider a square plate, of dimension $\alpha \lambda \times \alpha \lambda$, where we limit $\log_2 \alpha$ to be an integer. Consider further the scenario where all patches on the square plate have sidelength $s \lambda$, where we for simplicity limit $\log_2 s$ to be an integer. Further, we label the number of basis functions on each patch as $N_p$. We also choose the order $W = 5$ of the Lagrange interpolation.

Now, we can find some quantities relating to the problem, as shown in Table F.1. Further, defining the 1st level to be the bounding box, centered around the plate, we get the quantities for each level shown in Table F.2.

Further, we define the spherical harmonics order $Q = L_q/2 + \max\{\beta - 2, 0\}$.

Based on these numbers, we can begin to quantify the number of elements needed to be stored in several categories. Most easily, we can find the memory

**Figure F.1:** The effect of using larger patches on $L$ and $K$, normalized against the scenario for RWG functions.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Found as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patches per side</td>
<td>$P$</td>
<td>$\frac{\alpha}{\pi}$</td>
</tr>
<tr>
<td>Number of levels</td>
<td>$q$</td>
<td>$\lfloor \log_2 P \rfloor + 1$</td>
</tr>
<tr>
<td>Basis functions</td>
<td>$N$</td>
<td>$N_p P^2$</td>
</tr>
</tbody>
</table>

**Table F.1:** Basic quantities for the test case.

**F.2.2 Theoretical Test Case**
Figure F.2: Illustration of the test case, including the levels determined by the Octree. The blue lines indicate groups on level 2, the green lines indicate groups on level 3 and finally the red lines correspond to groups on level 4.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Found as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sidelength at $i$’th level</td>
<td>$a_i$</td>
<td>$\frac{\alpha}{2^{i-1}}$</td>
</tr>
<tr>
<td>Diameter at $i$’th level</td>
<td>$D_i$</td>
<td>$\sqrt{3}a_i$</td>
</tr>
<tr>
<td>Truncation at $i$’th level</td>
<td>$L_i$</td>
<td>$2\pi D_i + 1.8\beta^{2/3}(2\pi D_i)^{1/3}$</td>
</tr>
<tr>
<td>Number of directions at $i$’th level</td>
<td>$K_i$</td>
<td>$2(L_i + 1)^2$</td>
</tr>
<tr>
<td>Number of groups at $i$’th level</td>
<td>$G_i$</td>
<td>$2^{2(i-1)}$</td>
</tr>
</tbody>
</table>

Table F.2: Quantities for each level, derived from the basic quantities in Table F.1.
for storing the group patterns:

\[
\text{Group Memory: } \sum_{i=3}^{q} G_i K_i \frac{2}{\theta, \phi} \quad (F.8)
\]

Further, the basis function memory is also easily found

\[
\text{Basis function Memory: } N \frac{2}{\theta, \phi} K_q \quad (F.9)
\]

Then, consider the memory for the near-matrix. Here, using the one-buffer box criterion, we have to separately consider the patches along the edges of the plate. Most easily, the patch on each of the 4 corners has 3 neighbours plus itself in the near-interaction zone. So the memory is:

\[
\text{NF memory, corners: } 4 \frac{16}{\text{corners}} N_p^2 \quad (F.10)
\]

Then comes the patches along each side, aside from the corner elements. There are \(P - 2\) patches along each side, each of which has 5 neighbours plus itself.

\[
\text{NF memory, side: } 4 \frac{6}{\text{sides}} (P-2) N_p^2 \quad (F.11)
\]

Finally, the interior patches all have 8 neighbours plus themselves.

\[
\text{NF memory, side: } 9 \frac{(P-2)^2}{\text{patches}} N_p^2 \quad (F.12)
\]

Gathering these, we get

\[
\text{NF memory, total: } N_p^2 \left\{ 16 + (P-2) [24 + 9(P-2)] \right\} \quad (F.13)
\]

But then we have to remember that the matrix can be stored symmetrically, which means that we can divide the above by 2 (and then add \(N/2\) since we cannot store the diagonal symmetrically).

\[
\text{NF memory, total: } N_p^2 \left\{ \frac{16 + (P-2) [24 + 9(P-2)]}{2} \right\} + \frac{N}{2} \quad (F.14)
\]

With that done, we need to consider the translator memory. Remember that only the unique translators need to be stored. There are at most \(4 \cdot 9\) unique
translators per level (except for level 3, where there are fewer). We disregard the memory for translators on level 3, since this will be the same for all cases, so it does not affect the conclusions.

\[
\text{Translator memory: } \sum_{i=4}^{q} 4 \cdot 9K_i \quad (F.15)
\]

Finally, the memory for the interpolation matrices can be found as:

\[
\text{Interpolator memory: } \sum_{i=3}^{q-1} K_i W^2 \quad (F.16)
\]

We then consider a case with \( \alpha = 256\lambda, \beta = 2 \). We know that for order \( C \) along the direction of the current, the number of basis functions on a patch is \( N_p = 2(C + 1)C \).

We then try some different combinations of orders and sidelengths:

<p>| Memory use for varying discretizations using standard HO MLFMM |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th>s [( \lambda )]</th>
<th>Order</th>
<th>Basis func</th>
<th>Near-mat</th>
<th>Group</th>
<th>Translator</th>
<th>Interp</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>1</td>
<td>18.0</td>
<td>2.3</td>
<td>10.3</td>
<td>0.4</td>
<td>0.3</td>
<td>31.3</td>
</tr>
<tr>
<td>0.25</td>
<td>2</td>
<td>24.0</td>
<td>5.1</td>
<td>5.8</td>
<td>0.4</td>
<td>0.3</td>
<td>35.5</td>
</tr>
<tr>
<td>0.5</td>
<td>3</td>
<td>27.0</td>
<td>5.1</td>
<td>3.8</td>
<td>0.4</td>
<td>0.3</td>
<td>36.5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>28.2</td>
<td>3.5</td>
<td>2.6</td>
<td>0.4</td>
<td>0.3</td>
<td>35.0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>39.4</td>
<td>3.8</td>
<td>1.9</td>
<td>0.4</td>
<td>0.3</td>
<td>45.8</td>
</tr>
</tbody>
</table>

Obviously, choosing patch sidelengths which for a specific order yields a given accuracy is difficult. However, the numbers above are taken to be roughly the same as the results from [J1, Case B], and thus should be acceptable for this rough theoretical consideration.

In summary, the table shows clearly why there has been a consensus that increasing the basis function order is detrimental to performance. The reduction in group memory from increasing the order comes at a cost of vastly increased basis function memory.
We note that had we used the SHE storage (Section 3.3), the memory for the basis functions would have been

\[
\text{Basis function Memory, SHE: } N \sum_{x, y, z} \frac{(Q + 1)(Q + 2)}{2} \text{ SHE coefficients}
\]  

(F.17)

which, when we put these numbers into the table above, yield:

<table>
<thead>
<tr>
<th>$s ,[\lambda]$</th>
<th>Order</th>
<th>Basis func</th>
<th>Near-mat</th>
<th>Group</th>
<th>Translator</th>
<th>Interp</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>1</td>
<td>3.8</td>
<td>2.3</td>
<td>10.3</td>
<td>0.4</td>
<td>0.3</td>
<td>17.0</td>
</tr>
<tr>
<td>0.25</td>
<td>2</td>
<td>4.2</td>
<td>5.1</td>
<td>5.8</td>
<td>0.4</td>
<td>0.3</td>
<td>15.7</td>
</tr>
<tr>
<td>0.5</td>
<td>3</td>
<td>3.9</td>
<td>5.1</td>
<td>3.8</td>
<td>0.4</td>
<td>0.3</td>
<td>13.4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3.2</td>
<td>3.5</td>
<td>2.6</td>
<td>0.4</td>
<td>0.3</td>
<td>10.0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4.2</td>
<td>3.8</td>
<td>1.9</td>
<td>0.4</td>
<td>0.3</td>
<td>10.6</td>
</tr>
</tbody>
</table>

Thus, with the use of the SHE storage for basis function patterns, along with perfect grouping (automatically achieved for this test case, but otherwise achieved through the adaptive grouping from Section 3.1), increasing the basis function order is indeed worthwhile. The group memory decreases far more than the increase for the near-matrix, and the BF memory does not increase significantly. Further, as discussed in Chapter 4, the reduced number of levels (from $q = 12$ to $q = 8$ for the above numbers) remarkably increases the speed.

\section*{F.3 Deducing (D.3)}

The traditional form of the Gegenbauer addition theorem is \[195\] (10.23.8)]

\[
\frac{C_{\nu}(w)}{w^{\nu}} = 2^{\nu}\Gamma(\nu)\sum_{l=0}^{\infty}(\nu + l)\frac{C_{\nu+l}(u)}{u^{\nu}}J_{\nu+l}(v)C_{l}^{(\nu)}(\cos \alpha),
\]  

(F.18)

where $w = \sqrt{u^2 + v^2 - 2uv \cos \alpha}$.

This is valid for $\nu \neq 0, -1, ..., $ and for $|ve^{\pm j\alpha}| < |u|$. Here, $J$ is the Bessel function of the first kind, $C_{l}^{(\nu)}$ is the $l$th Gegenbauer polynomial of order $\nu$, and $C_{\nu}(w)$ is any linear combination of Bessel, Neumann and Hankel functions,
such that \( a, b, c, d \) can be chosen to be any value in the relation
\[
C_\nu(w) = aJ_\nu(w) + bY_\nu(w) + cH^{(1)}_\nu(w) + dH^{(2)}_\nu(w) \tag{F.19}
\]
The Gegenbauer addition theorem can be considered geometrically, as shown in Figure F.3. Inserting \( \nu = \frac{1}{2} \) allows us to employ the spherical Bessel and Hankel functions, while choosing \( C = H^{(2)} \) will help us in a way that becomes obvious in a moment:
\[
H^{(2)}_{\frac{1}{2}}(w) = \sqrt{2} \sqrt{\pi} \sum_{l=0}^{\infty} (2l + 1) \frac{H^{(2)}_{l+\frac{1}{2}}(u) J_{l+\frac{1}{2}}(v)}{\sqrt{u} \sqrt{v}} C_l^{(1/2)}(\cos \alpha). \tag{F.20}
\]
Note that the factor \( 2l + 1 \) is due to there being \( 2l + 1 \) linearly independent spherical harmonics of order \( l \).

The following relations hold between the usual Bessel and Hankel functions, i.e., \( J \) and \( H \), and the spherical functions \( j \) and \( h \) \([195, (10.47.3-6)]\):
\[
j_l(z) = \sqrt{\frac{\pi}{2z}} J_{l+1}(z), \tag{F.21}
\]
\[
h^{(2)}_l(z) = \sqrt{\frac{\pi}{2z}} H^{(2)}_{l+1}(z). \tag{F.22}
\]
Further, since \([196, p. 74]\) \( C_l^{(1/2)} = P_l \), with \( P_l \) the \( l \)'th Legendre polynomial, we find that:
\[
\sqrt{\frac{\pi}{2}} h^{(2)}_0(w) = \sqrt{2} \sqrt{\pi} \sum_{l=0}^{\infty} (2l + 1) \sqrt{\frac{2}{\pi}} h^{(2)}_l(u) \sqrt{\frac{2}{\pi}} j_l(v) P_l(\cos \alpha). \tag{F.23}
\]
This can further be reduced to:
\[
h^{(2)}_0(w) = \sum_{l=0}^{\infty} (2l + 1) h^{(2)}_l(u) j_l(v) P_l(\cos \alpha). \tag{F.24}
\]
By exploiting that \([195, (10.49.7)]\):
\[
h^{(2)}_n(z) = e^{-jz} \sum_{l=0}^{n} (-j)^{l-n-1} a_l(n+\frac{1}{2})/z^{l+1}, \tag{F.25}
\]
\[
a_l(n+\frac{1}{2}) = \begin{cases} 
\frac{(n+l)!}{2^n l!(n-l)!} & \text{for } l = 0, 1, \ldots, n, \\
0 & \text{for } l = n+1, n+2, \ldots \tag{F.26}
\end{cases}
\]
we get that
\[
h^{(2)}_0(w) = e^{-jw}/(jw), \text{ and thus:} \tag{F.27}
\]
\[
e^{-jw}/(-jw) = \sum_{l=0}^{\infty} (2l + 1) h^{(2)}_l(u) j_l(v) P_l(\cos \alpha)
\]
Letting $u = k\|x\|$ and $v = k\|d\|$, then $w = k\|x + d\|$, and further $\cos \alpha = -\hat{x} \cdot \hat{d}$. This yields

$$e^{-jk\|x+d\|}/\|x+d\| = -jk \sum_{l=0}^{\infty} (2l+1)h_l^{(2)}(k\|x\|)j_l(k\|d\|)P_l(-\hat{x} \cdot \hat{d})$$  \hspace{1cm} (F.28)

Finally, realizing that $P_l(-x) = (-1)^l P_l(x)$, we end up with (D.3):

$$e^{-jk\|x+d\|}/\|x+d\| = -jk \sum_{l=0}^{\infty} (-1)^l(2l+1)h_l^{(2)}(k\|x\|)j_l(k\|d\|)P_l(\hat{x} \cdot \hat{d})$$  \hspace{1cm} (F.29)

**F.4 Selecting the Truncation Parameter**

In this section, we briefly review the theory behind the choice of the truncation parameter $L$ in (D.9). We consider the error contribution from the $L+1$st term (which we know is the largest error contribution since the addition theorem (D.3) has uniform convergence if $\|x\| > \|d\|$ [8, Lemma 2.4]). If we further assume that $L > kd$ and $L < kX$, where $D = \|d\|$ and $X = \|x\|$, we have that the leading error term is

$$\frac{2(L+1)+1}{kX} j_{L+1}(kD)$$  \hspace{1cm} (F.30)

where we have applied the large argument approximation for the spherical Hankel function, valid only because we have assumed $L < kX$. The literature to be described in the following has generally focused on ensuring that (F.30) is below the specified threshold.

**Figure F.3:** Geometrical illustration to visualize the relation described by the Gegenbauer addition theorem [F.18].
Coifman et al. [10] used the "semi-empirical" formula
\[ L = kD + 5 \ln(kD + \pi) \] (F.31)
This formula has a number of drawbacks, most obvious the fact that it cannot easily be adjusted to various accuracies—it yields a relative error of approximately $10^{-6}$ according to the authors. Work by Bucci [197] and Rokhlin [41] made this error-adjustable by the approximate formula
\[ L = kD + \beta(kD)^{1/3} \] (F.32)
While this formula is better, as it is error adjustable, it is not straight-forward how $\beta$ relates to the error. Based on (F.32), more thorough work by [67] yielded the Excess Bandwidth Formula (EBF) (D.22), which is now the standard formula in the field.

The accuracy of the EBF depends on the Bessel function being the leading error term in (D.3). This in turn assumes that $L < k \|x\|$, since the Hankel function oscillates vehemently when the order surpasses the argument in magnitude. The EBF does not take the error contribution into account if $L > kX$.

Therefore, when $L > kX$, the EBF is not applicable. Thus, the threshold for including interactions in $\mathbf{Z}_{\text{near}}$ would have to be $X < L/k$. In practice, this results in a large number of non-zeros in $\mathbf{Z}_{\text{near}}$, and the one buffer-box criterion is used instead, such that interactions where
\[ X \leq N_{\text{BUF}}D, \] (F.33)
for $N_{\text{BUF}} = 1$, are placed in $\mathbf{Z}_{\text{near}}$.

An immediate concern is then how to choose $L$ for those interactions where $X > N_{\text{BUF}}D$, but $X < L/k$ such that sufficient accuracy is maintained. This was discussed in detail by Hastriter et al. [68], adding on work done by Ohnuki & Chew [69] for 2-D FMM. In particular, [68] derived a formula for the lost number of digits of accuracy $d_1$:
\[ d_1 = \left[ \frac{L - (N_{\text{BUF}} + 1)ka}{1.8((N_{\text{BUF}} + 1)ka)^{1/3}} \right]^{3/2}, \] (F.34)
where $a = D/\sqrt{3}$ is the sidelength of a group. Thus, the achieved relative error will be $10^{-(\beta-d_1)}$. We note that (F.34) is only applicable when $L > (N_{\text{BUF}}+1)ka$, since otherwise the assumptions for using EBF are fulfilled (and $d_1$ is very close to 0). [68] then gives a procedure to minimize $d_1$ (it might not be possible to achieve $d_1 = 0$, i.e., it might not be possible to achieve the desired accuracy, as discussed in Appendix F.5).
However, the approach has one significant drawback: It results in very large values of $L$. And since all groups have to use the same value of $L$, it further means that all groups have to store a very large number of tabulated points. Furthermore, the resulting values for $L$ are extremely pessimistic, ensuring the required accuracy even at the perimeter of the groups.

Therefore, the approach has received very little use in practice. In fact, the main contribution from [68] seems to be that it is often not possible to achieve $d_1 = 0$, and that the achievable accuracy is increased as the group size increases. This is discussed further in the next section.

### F.5 Achievable Translation Accuracy

In this section, we consider the achievable accuracy of the Rokhlin translator (D.9), repeated here for completeness:

$$T_L(k, x) = \sum_{l=0}^{L} (-j)^l (2l + 1) h_l^{(2)}(k \|x\|) P_l(\hat{k} \cdot \hat{x}),$$  \hspace{1cm} (D.9 revisited)

where $x$ is the vector from the center of the source group to the center of the observation group. The translator is tabulated in $K = 2(L + 1)^2$ directions on the unit sphere, where the truncation parameter $L$ depends on the diameter $D$ of the source and observation groups as defined by (D.22). An integral is then performed over the surface of the unit sphere, to yield an error-controllable approximation to the Green’s function as shown in (D.8), repeated below

$$e^{-jk\|x+d\|} \|x+d\| \approx -\frac{jk}{4\pi} \iint e^{-jk \cdot d} T_L(k, x) \, d^2 \hat{k},$$  \hspace{1cm} (D.8 revisited)

where we note that $D = \|d\|$.

The choice of $L$ for a given $D$ is, as discussed in Appendix F.4, based on the assumption that $L < k \|x\|$, which might not be valid in practice. However, even if we had a magical way of choosing the perfect $L$, we could still run into problems: It turns out that the Rokhlin translator cannot always achieve the required accuracy, since (D.8) actually diverges for $L \to \infty$. Thus, there is some optimal value of $L$ that provides the maximum accuracy for given $x$ and $d$, and increasing $L$ beyond this value reduces the accuracy.

As the groups grow smaller, the so-called sub-wavelength breakdown occurs, resulting in a very limited accuracy for small groups as discussed in detail by
Figure F.4: A 2D illustration of the geometry for the test case behind Figure F.5.

Hastriter et al. [68]. In practice, the sidelength of the groups cannot be smaller than $0.25\lambda - 2\lambda$ depending on the required accuracy [75, p. 1298].

To illustrate this concept, consider a point source located at $(a/2, a/2, a/2)$, with a corresponding source group box of diameter $D$ and sidelength $a = D/\sqrt{3}$ centered at $(0,0,0)$. We then compute the Green's function due to this point source at the observation point $(2a - a/2, -a/2, -a/2)$ in the observation group centered at $(2a,0,0)$. The scenario is illustrated Figure F.4. Since the distance between the two points is $D$ and the length of the translation vector is $||\mathbf{x}|| = 2a$ and the two local vectors in the groups are co-linear [65], this is the worst-case scenario in FMM if we disregard situations where patches "stick-out" of groups.

We now consider the best achievable relative error achieved by applying (D.8). We thus consider a wide range of possible values of the truncation parameter $L$ as we vary the electrical size of the groups, and only keep the best achieved accuracy for each group size.

The results in Figure F.5. From the figure, three things are evident. First, the best achievable error is surprisingly high for small sidelengths. Remembering that the group size is between $0.2\lambda - 0.4\lambda$ for LO MLFMM, Figure F.5 shows that LO MLFMM cannot achieve much better than around $10^{-3}$ relative error, regardless of the setting of the truncation parameter.

Second, the overall tendency is that as the size of the groups increase, the error achieved decreases. This happens because the Rokhlin translator is better behaved for larger groups, a sideeffect of the behaviour of the Hankel function when the order surpasses the argument in magnitude.

Third, the error behaves rather erratically. This is due to the integration error in the spherical integral, since the usual integration rules are based on the assumption that the Hankel function is well-behaved. Increasing the number of sample points beyond $2(L + 1)^2$ reduces the erratic behaviour but does not change the
Figure F.5: The lowest achievable relative error in (D.8) as a function of the sidelength $a = D/\sqrt{3}$.

other conclusions: The achievable error for small groups is still around $10^{-3}$ and the larger the group size, the better the achievable accuracy.

F.6 Connection Between Residual Norm and RMS

In this appendix, inspired by the discussion in Engsig-Karup [198, p. 762], we consider the relationship between the residual norm and the Relative RMS error of the solution. When applying an iterative solver, a succession of iteration vectors $\overline{T}^{[k]}$ are achieved, where $k > 0$ is the iteration number. Ideally, these vectors should approach the true solution as $k$ is increased, in such a way that the relative residual norm
\[
\frac{\|\overline{T}^{[k]}\|}{\|\overline{V}\|}
\] (F.35)
has been reduced below a desired threshold. The residual norm at the $k$’th iteration, $\|\overline{T}^{[k]}\|$, is
\[
\|\overline{T}^{[k]}\| = \|\overline{V} - \overline{Z}\overline{T}^{[k]}\|.
\] (F.36)
We note that when using a preconditioner $\overline{M}$, the residual norm is
\[
\|\overline{T}^{[k]}\| = \|\overline{M} \left(\overline{V} - \overline{Z}\overline{T}^{[k]}\right)\|
\] (F.37)
However, the relative residual norm (F.35) can only provide information about the discretized solution error, and thus does not take into account the discretization error itself, i.e., the error associated with representing the true solution using polynomials of limited order. Further, the Euclidean norm of the residual is typically not the quantity of interest. In the cases discussed in the present thesis, the key quantity is the error of the scattered field, quantified using the Relative RMS error (4.1).

To investigate the relationship between the residual norm and the Relative RMS error, we consider the scattering from a PEC sphere illuminated by a $f = 2$ GHz plane wave, as we did in Section 4.1. We use the GMRES solver, and for each iteration vector, we compute the Relative RMS error. For a standard discretization, the results are shown in Figure F.6, while Figure F.7 shows the results for a finer discretization.

We see that the Relative RMS error is, for both discretizations, dominated by the residual error until a “plateau” is reached, signifying the discretization error. In other words, if we set our residual norm threshold on to $\epsilon$, we will get a Relative RMS error at most $\epsilon$ or the discretization error, whichever is higher. This is an interesting result, since it allows us to set $\epsilon$ to be our target RMS, and then actually achieve that RMS or the best possible with the current discretization.

However, we stress that scattering from a sphere is a special case in this regard. We are using the CFIE, which is better conditioned than the EFIE, and we are considering scattering from the sphere, which is by far dominated by the backscattering. Thus, the conclusions here might be less valid for other scatterers such as the corner reflector, or problems such as computing the input impedance for a generator.

To further test the connection between the residual norm and the Relative RMS Error, we thus consider scattering from a corner reflector, using an overdiscretized MoM solution as our reference solution. The comparison, analogous to Figure F.6, is shown in Figure F.8. While the residual norm is much closer to the Relative RMS Error than was the case for the sphere, the residual norm remains a very usable estimate of the Relative RMS Error achieved.

We remind that the Relative RMS error is not always the correct error norm—for some applications, very minor effects at low levels can be important and, in that case, the Relative RMS error might be a poor quantification of the perceived “quality” of the solution.
Figure F.6: Comparison between residual norm and Relative RMS Error for scattering from a sphere.

Figure F.7: Comparison between residual norm and Relative RMS Error for a finer discretization. Scattering from a sphere.
Figure F.8: Comparison between residual norm and Relative RMS Error for scattering from a corner reflector.
This appendix lists the computer architectures used to compute the various results in the paper.

G.1 Laptop

The laptop used is a Late 2013 15-inch Retina MacBook Pro. The Intel i7 4960HQ has 4 physical cores, capable of running 2 threads each using Intel’s hyperthreading technology.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Intel i7 4960HQ @ 2.6 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>16 GB 1600 MHz DDR3</td>
</tr>
<tr>
<td>Disk</td>
<td>512 GB SSD disk</td>
</tr>
</tbody>
</table>

**Table G.1**: Specifications for our laptop.
The computing server used for larger cases. Each Intel Xeon E5-2690 has 8 physical cores, capable of running 2 threads each using Intel’s hyperthreading technology, for a total of 16 cores or 32 threads in the machine.

<table>
<thead>
<tr>
<th>Processor</th>
<th>2 × Intel Xeon E5-2690 @ 2.9 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>160 GB PC3-12800R</td>
</tr>
<tr>
<td>Disk</td>
<td>7200 RPM HDD</td>
</tr>
</tbody>
</table>

*Table G.2: Specifications for our computing server.*
Bibliography


