UNIVERSITY OF CALIFORNIA

Santa Barbara

Multipole for Scattering Computations: Spectral Discretization, Stabilization, Fast Solvers

A Dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Electrical and Computer Engineering

by

Timothy Paul Pals

Committee in charge:
Professor Shivkumar Chandrasekaran, Co-chair
Professor Hua Lee, Co-chair
Professor Linda Petzold
Doctor Dean Mensa

December 2004

The dissertation of Timothy Paul Pals is approved.

Professor Linda Petzold

Doctor Dean Mensa

Professor Shivkumar Chandrasekaran, Committee Co-chair

Professor Hua Lee, Committee Co-chair

December 2004

Multipole for Scattering Computations:

Spectral Discretization, Stabilization, Fast Solvers

Timothy Paul Pals

ACKNOWLEDGMENTS

With love to my family and friends

To my research advisors Hua Lee and Shiv Chandrasekaran
Whose generous efforts have guided this body of work

To my mother Mary and my father Paul
To my sister Kristin and my brother Steve
Who have supported me as I work through
All the nontechnical problems presented by life

With gratitude to my grandparents
Richard and Eleanor
Ambrose and Stella
Who have long been sage advocates of education
For all their grandchildren

CURRICULUM VITAE

Timothy Pals
Department of Electrical and Computer Engineering
University of California, Santa Barbara
tpals@engr.ucsb.edu

Education

Ph.D. University of California, Santa Barbara
Electrical and Computer Engineering
December 2004

M.S. University of California, Santa Barbara
Electrical and Computer Engineering
December 1995

B.S. University of Illinois, Urbana–Champaign
Engineering Physics
January 1994

Research

Graduate Student Researcher
Department of Electrical and Computer Engineering
University of California, Santa Barbara

Numerical solution of elliptic PDEs for wave scattering
Under the supervision of Shivkumar Chandrasekaran
1999–2004

Constructed new high-order discretizations of boundary integral equations. Eliminated numerical instability in the fast multipole method for the Helmholtz equation. Introduced *noniterative* fast multipole solvers. Prototyped this numerical software in Matlab.

Modeling of radio wave propagation
Under the supervision of Hua Lee
2000–2004

Developed ray tracing software to simulate multipath communication channels in indoor and urban environments. Proposed new algorithms to detect communication links lacking a line-of-sight path in ad hoc radiolocation networks.

Fast solution of Toeplitz linear systems
Under the supervision of Shivkumar Chandrasekaran
1998–1999

Investigated direct and iterative solvers to compute mutual element impedance and antenna pattern of microstrip patch antenna arrays, for which low-order discretizations give nested block Toeplitz matrix structure.

Microwave imaging systems
Under the supervision of Hua Lee
1996–1998

Studied mathematics and algorithms for inverse scattering. Designed and constructed 10 GHz microwave transceiver with printed antenna front-end. Gained experience in microwave measurement techniques. Wrote C++ code for instrument control and data collection.

Teaching

Graduate Teaching Assistant
Department of Electrical and Computer Engineering
University of California, Santa Barbara

ECE 130A: Continuous-time signal and system analysis
Fall 1994

ECE 130B: Discrete-time signal and system analysis
Winter 1995, Winter 1998, Winter 2000

ECE 140: Probability and statistics
Fall 1999

ECE 130C: Linear algebra
Spring 2000

Industry Experience

Consultant
Akela Inc., Santa Barbara
January 2001

Retargeted research ray tracing code to be a software component for through-wall radar imaging.

Member of Technical Staff
Hughes Aircraft Company, Los Angeles
January–September 1994

Worked in airborne radar system integration and test laboratories for F-14, F-15, and F/A-18 programs. Designed, documented, and conducted tests to verify system requirements. Improved implementation of problem tracking database. Wrote

Fortran code to model flight dynamics. Supported analysis and development of single-target tracking modes.

Received an educational leave of absence to pursue graduate study at UCSB.

Co-op Intern
Hughes Aircraft Company, Los Angeles
Four rotations from January 1991 through August 1993

Internship in cooperation with University of Illinois, alternating semesters of full-time study in Urbana with tours of full-time employment in Los Angeles.

Worked in special test equipment design laboratory, primarily on rack-mount digital/RF/microwave systems for the physical simulation of airborne radar targets. Designed, documented, and tested digital circuit boards which used a VME interface and comprised TTL and ECL ICs at small through large levels of integration. Wrote firmware for onboard programmable logic.

Selected Honors

National Science Foundation Graduate Research Fellowship
Fall 1994–Fall 1997

University of California Regents Special Fellowship
Fall 1997–Fall 1999

University Honors (Bronze Tablet)
May 1994
University of Illinois award to top 3% of each college's graduating class.

Parsons Graduate Fellowship
Fall 1999–Fall 2001
UCSB College of Engineering award for research in computational engineering.

Journal Articles

Shivkumar Chandrasekaran, Ming Gu, and Timothy Pals, *Fast and Stable Algorithms for Hierarchically Semi-Separable Representations.* Submitted to SIAM Journal on Matrix Analysis and Applications.

Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, Timothy Pals, X. Sun, A.-J. van der Veen, and Daniel White, *Some Fast Algorithms for Sequentially Semi-Separable Representations.* Accepted by SIAM Journal on Matrix Analysis and Applications.

Sunwoo Kim, Andrew P. Brown, Timothy Pals, Ronald A. Iltis, and Hua Lee, *Geolocation in Ad hoc Networks using DS-CDMA and Generalized Successive Interference Cancellation.* Submitted to IEEE Journal of Selected Areas in Communications.

Conference Papers

Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, Timothy Pals, and A.-J. van der Veen, *Fast Stable Solver for Sequentially Semi-Separable Linear Systems of Equations.* Proceedings of the 9th International Conference on High Performance Computing, Bangalore, India, December 2002. [Lecture Notes in Computer Science, 2552:545–554, 2002.]

Sunwoo Kim, Timothy Pals, Ronald A. Iltis, and Hua Lee, *CDMA Sparse Channel Estimation using a GSIC/AM Algorithm for Radiolocation.* Conference Record of the 36th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California, November 2002.

Sunwoo Kim, Timothy Pals, Ronald A. Iltis, and Hua Lee, *CDMA Multipath Channel Estimation using Generalized Successive Interference Cancellation Algorithm for Radiolocation.* Proceedings of the 36th Annual Conference on Information Sciences and Systems, Princeton, New Jersey, April 2002.

Selected Presentations

Direct Solution of Certain Dense Linear Systems using the Fast Multipole Method, invited talk, Matrix Computations and Scientific Computing seminar organized by Beresford Parlett, University of California, Berkeley, February 2004.

Numerical Solution of Integral Equations for Wave Scattering, invited talk, seminar organized by Beresford Parlett, University of California, Berkeley, April 2002.

Fast High-Order Solution of Electromagnetic and Acoustic Scattering Problems, contributed talk, SIAM Annual Meeting, San Diego, California, July 2001.

ABSTRACT

Multipole for Scattering Computations:
Spectral Discretization, Stabilization, Fast Solvers

by

Timothy Paul Pals

In 1985, Vladimir Rokhlin introduced the fast multipole method (FMM) for the solution of the Laplace equation. The method has since been named, by *Computing in Science & Engineering* magazine, one of the 20th century's ten most influential algorithms. In 1990, Rokhlin introduced a version of FMM for the Helmholtz equation that has greatly influenced algorithm development in the computational electromagnetics community.

Unfortunately, scattering FMM is numerically unstable. Its instability can be traced to the asymptotic behavior of the multipole basis functions. After demonstrating the instability, I introduce measures that eliminate it without sacrificing the method's efficiency. Relative accuracies approaching 10^{-16} are possible, even for scattering obstacles with diameters smaller than 10^{-200} wavelengths.

When applied to the solution of PDEs, multipole methods are embedded into iterative solvers such as GMRES. In many circumstances, the discretization produces a poorly conditioned algebraic system, and iterative solvers are slow to converge. I introduce a fast *direct* solver which avoids that difficulty. The multipole structure is embedded into a large sparse system, to which a standard sparse solver is applied.

I demonstrate scattering problems for which the new solver is clearly superior to either dense Gaussian elimination or iterative FMM.

To apply FMM, the PDE is first reformulated as a weakly singular integral equation. Multipole methods constrain the integral discretization in a way that makes a high-order numerical solution difficult. The existing high-order discretizations are numerically unstable. I present a stable discretization of arbitrarily high order that satisfies the multipole constraints. I routinely solve scattering problems using rules with order 32, and I have constructed stable discretizations with orders as high as 288.

To illustrate these techniques, I exhibit numerical solutions in two space dimensions. All computations have been carried out in MATLAB. Scattering obstacles with diameters greater than 1000 wavelengths can be comfortably treated with a personal computer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: The Helmholtz Equation

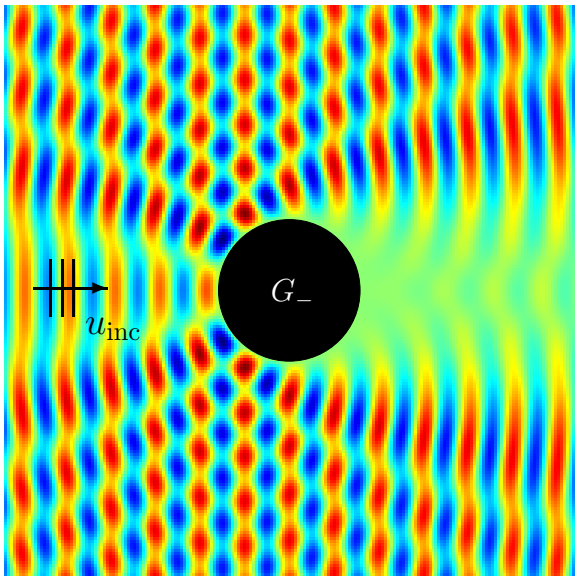


Figure 1.1: Snapshot of the total field $u + u_{\text{inc}}$ scattered by a circular cylinder G_- under illumination by a plane wave u_{inc} incident from the left. The axis of the cylinder is perpendicular to the plane of the paper. The total field vanishes at the cylinder surface.

This dissertation covers the efficient numerical solution of the Helmholtz equation, an elliptic partial differential equation (PDE) that governs linear wave scattering processes. Figure 1.1 illustrates a simple problem setting.

Since this work has been carried out in the Department of Electrical and Computer Engineering at UC Santa Barbara, I will emphasize *electromagnetic* scattering processes. The Helmholtz equation does, however, play an important role in a host of other natural phenomena:

- The field of acoustics is concerned largely with the scattering of *sound* waves.
- The structure of materials at submicroscopic scales can be revealed by the scattering of *matter* waves. Particle scattering is a standard problem in quantum mechanics.
- General relativity predicts the existence of *gravity* waves that are scattered by concentrations of matter and energy in the universe.

The numerical techniques contained herein apply to these domains as well.

1.1 The Helmholtz Boundary Value Problem

Although the Helmholtz equation typically materializes from more fundamental physical laws (for examples, see Section 1.3), it is mathematically expedient to distill those various starting points into a single unifying point of departure. Here, then, is a nearly precise description of the problem we shall consider.

Problem 1.1 (Exterior Helmholtz) *Given: (1) a real constant $k > 0$, (2) a simple, bounded, piecewise very smooth curve Γ in the Euclidean plane $\mathbb{E}^2$, and (3)*

a complex-valued function $f : \Gamma \to \mathbb{C}$ on the curve. On the unbounded domain G_+ with boundary Γ, construct a function $u : G_+ \to \mathbb{C}$ such that

$$\Delta u(\boldsymbol{x}) + k^2 u(\boldsymbol{x}) = 0 \quad \textit{for} \quad \boldsymbol{x} \in G_+, \tag{1.1a}$$

$$u(\boldsymbol{x}) = -f(\boldsymbol{x}) \quad \textit{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.1b}$$

$$\lim_{r\to\infty} \sqrt{r}\left(\frac{\partial u}{\partial r}(\boldsymbol{x}) - iku(\boldsymbol{x})\right) = 0 \quad \textit{where} \quad r := \|\boldsymbol{x}\|, \tag{1.1c}$$

and the latter condition is to hold uniformly with respect to the direction of the vector $\boldsymbol{x} \in G_+$.

The following remarks may clarify the hypotheses and the equations (1.1).

Problem 1.1 is the Dirichlet boundary value problem (BVP) for the Helmholtz equation (1.1a). The differential operator Δ is the Laplacian, also commonly denoted by the symbol ∇^2. Its definition in rectangular coordinates $\boldsymbol{x} = (x, y)$ is $\Delta u(x, y) := \partial^2 u/\partial x^2 + \partial^2 u/\partial y^2$.

The constant k is the wavenumber, the angular spatial frequency of a plane wave solution to (1.1a). It is proportional to the temporal frequency ν through $k = 2\pi\nu/c$, where c is the phase velocity of the plane wave. High frequency problems are those for which $k \operatorname{diam} \Gamma$ is large.

In (1.1b) the data f is the restriction of a prescribed incident field u_{inc} to the boundary Γ. The solution u is the scattered field produced in response to that excitation.

Other BVPs for the Helmholtz equation commonly occur. Introducing a linear differential operator $\mathcal{B} : (G_+ \to \mathbb{C}) \to (\Gamma \to \mathbb{C})$, the condition (1.1b) might be replaced with

$$\mathcal{B}u(\boldsymbol{x}) = -f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma. \tag{1.2}$$

In Section 1.3.2, I comment on other boundary conditions that are typically encountered, but otherwise our attention will be confined to the Dirichlet condition.

The condition (1.1c) is the Sommerfeld radiation condition. Another way of writing it,

$$\frac{\partial u}{\partial r} - iku = o\Big(\frac{1}{\sqrt{r}}\Big) \quad \text{as} \quad r \to \infty, \tag{1.3}$$

makes use of the "little oh" notation for asymptotic behavior. This is a boundary condition applied on a circle that encloses Γ and expands to infinity. The linear combination of field values u and normal derivatives $\partial u/\partial r$ rejects functions with the asymptotic character of incoming circular waves $r^{-1/2}e^{-ikr}$, but it accepts outbound waves $r^{-1/2}e^{ikr}$. The equivalent condition in the time domain (Section 1.3.1) is that the scattering response must depend causally on the illuminating field.

Solutions of the Helmholtz equation that also satisfy (1.1c) are called *radiating* solutions. The combination of (1.1a) and (1.1c) also ensures that $u = O(r^{-1/2})$ and $\partial u/\partial r = O(r^{-1/2})$.

Regarding the restrictions on the curve Γ, a *simple* curve is one that does not intersect itself. The qualifier *piecewise very smooth* demands that Γ may be partitioned into a finite number of arcs, each of which may be parametrized by a vector function $\boldsymbol{x} = \boldsymbol{\gamma}(t)$ with *two* continuous derivatives on the arc interior. Furthermore, a continuously turning tangent line must exist on each arc, so $\|\boldsymbol{\gamma}'(t)\| \neq 0$ at all interior points. Every bounded piecewise smooth curve has a finite arc length, so fractal curves are not allowed here. We shall consider both open and closed simple curves. A closed simple curve, or *Jordan curve,* is the common boundary of a bounded domain G_- and its unbounded exterior G_+.

These restrictions on the curve have been adopted primarily because they are the requirements of the MATLAB code that I wrote to solve Problem 1.1. The user

provides a symbolic piecewise parametrization of the curve, and the Maple kernel underlying MATLAB's Symbolic Math Toolbox is used to compute two symbolic derivatives of the parametrization. It is typical in existence proofs to see more restrictive qualifications of Γ, but my code may actually be extended to allow still more general boundaries. In particular, the problem might be generalized to allow a finite collection of disjoint curves, but I will give no such examples of multiple scattering obstacles in this work.

1.2 Survey of Contents and Contributions

Details of the new ideas mentioned in the abstract are contained in Chapters 3–5. I have also devoted considerable energy to the expositions in this chapter and the next, and they capture a lot of the hard-won insights that I have accumulated during my studies. For the reader new to multipole, or to computational electromagnetics, these chapters hopefully convey those insights without an unnecessary struggle.

After a section that gives the derivation of the Helmholtz equation from three starting points, the remainder of this chapter is devoted to the construction and numerical solution of scattering integral equations. The PDE can instead be solved directly, with finite difference or finite element methods, but the integral equation has two advantages:

- The computational grid occupies only the 1-D boundary Γ, rather than the unbounded 2-D domain G_+.
- Discretizations with a high order of accuracy are easier to construct.

On the other hand, integral equations generate dense algebraic systems, and the real work lies in decreasing solution times.

In Chapter 2, I present Rokhlin's fast multipole method, from scratch. The treatment is different in places from Rokhlin's, and a few new ideas appear. In particular, the multipole algorithm is represented with a dag, and that sets the table for the direct solver introduced in Chapter 4.

The standard approach to Problem 1.1 is to embed the fast multipole method into an iterative solver for linear systems. Not all discretizations, however, are compatible with multipole. In Chapter 3, I introduce a stable, high-order discretization that I expressly designed to work seamlessly with multipole methods. Then I explore the application of various Krylov solvers and various preconditioners to various curves Γ.

In Chapter 4, I abandon the iterative approach. I introduce a direct solver utilizing multipole structure. As far as I know, it is the first direct multipole solver to be applied to 2-D scattering problems like Problem 1.1. It is then rather straightforward, by abusing the direct solver, to introduce new preconditioners for scattering integral equations, and we are led back again to an iterative solver.

In Chapter 5, I face the numerical instability of scattering multipole, conveniently ignored in Chapters 2–4. Anyone who wishes to develop a multipole solver must confront this issue. Here, after analyzing the instability, I give my solution, which is to selectively scale the multipole basis and to selectively replace fast translations with slow translations. Functions that compute $\log J_n(x)$ without underflow, and $\log H_n(x)$ without overflow, are required.

I have implemented all numerical methods in MATLAB version 6.5. For many computations, MATLAB is not as fast as optimized machine code emitted by a good compiler. MATLAB, however, is a superior vehicle for experimentation. The MATLAB implementations do exhibit the predicted computational complexities. All

computations have been carried out on an inexpensive desktop computer[1] running GNU/Linux.

1.3 Origins of the 2-D Helmholtz Equation

The Helmholtz equation originates from the wave equation, which is itself often derived from more fundamental physical equations. Here, following a general discussion of the wave equation, I give specific connections to Maxwell's equations and to Schrödinger's equation.

1.3.1 The Wave Equation

The wave equation for a function $w : \mathbb{E}^2 \times \mathbb{R} \to \mathbb{R}$ is

$$\Delta w(\boldsymbol{x}, t) - \frac{1}{c^2} \frac{\partial^2 w}{\partial t^2}(\boldsymbol{x}, t) = 0, \tag{1.4}$$

where c is the wave propagation speed. A Fourier transform with respect to the variable t converts this hyperbolic PDE into an elliptic PDE.

Define the Fourier transform of w by[2]

$$\widetilde{w}(\boldsymbol{x}, k) := \int_{-\infty}^{\infty} w(\boldsymbol{x}, t) e^{ikct}\, dt. \tag{1.5}$$

The inverse transform of $\widetilde{w} : \mathbb{E}^2 \times \mathbb{R} \to \mathbb{C}$ is then

$$w(\boldsymbol{x}, t) = \frac{c}{2\pi} \int_{-\infty}^{\infty} \widetilde{w}(\boldsymbol{x}, k) e^{-ikct}\, dk. \tag{1.6}$$

[1] 1.4 GHz AMD Athlon model 4 CPU with 512 MB RAM

[2] This may look more familiar after the replacements $i \to -j$ and $kc \to \omega$, and if that is your preference then those substitutions may be propagated painlessly throughout this dissertation.

If (1.6) is substituted into (1.4), and if the order of integration and differentiation is exchanged, we have

$$\begin{aligned} 0 &= \frac{c}{2\pi}\int_{-\infty}^{\infty}\Big(\Delta\widetilde{w}(\boldsymbol{x},k)e^{-ikct} - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\widetilde{w}(\boldsymbol{x},k)e^{-ikct}\Big)dk \\ &= \frac{c}{2\pi}\int_{-\infty}^{\infty}\Big(\Delta\widetilde{w}(\boldsymbol{x},k) + k^2\widetilde{w}(\boldsymbol{x},k)\Big)e^{-ikct}\,dk. \end{aligned} \tag{1.7}$$

A Fourier transform of each side of the latter equation gives the desired result

$$\Delta\widetilde{w}(\boldsymbol{x},k) + k^2\widetilde{w}(\boldsymbol{x},k) = 0. \tag{1.8}$$

The transformed wave function $\widetilde{w}$ must satisfy the Helmholtz equation.

I have declined to select a particular function space so that operations such as the exchange of differentiation and integration may be rigorously justified. I have also chosen not to completely identify function spaces in the specification of Problem 1.1. That avoids the trouble of switching function space settings from time to time, since there is not a single choice that is preferred in all circumstances. The choice depends on the details of the supplied data (Γ, f).

Since Fourier analysis of the wave equation gives the Helmholtz equation, we can construct solutions of the wave equation through Fourier synthesis of solutions of the Helmholtz equation at multiple values of k.

In scattering problems, attention is commonly restricted to *time-harmonic* wave functions, which are functions that display a purely sinusoidal variation in time. A time-harmonic solution has only two Fourier components, with wavenumbers k and $-k$. The two components are complex conjugates, so that their sum is real-valued.

Thus a time-harmonic solution assumes the form

$$\begin{aligned} w(\boldsymbol{x},t) &= \widetilde{w}(\boldsymbol{x},k)e^{-ikct} + \overline{\widetilde{w}(\boldsymbol{x},k)}e^{ikct} \\ &= 2\,\mathrm{Re}\,\widetilde{w}(\boldsymbol{x},k)e^{-ikct} \\ &= 2|\widetilde{w}(\boldsymbol{x},k)|\cos(\arg\widetilde{w}(\boldsymbol{x},k) - kct). \end{aligned} \tag{1.9}$$

Since throughout a time-harmonic scattering problem k is fixed, it is usually dropped from the argument list of $\widetilde{w}$.

Another way to derive (1.8) for time-harmonic waves is to substitute (1.9) into (1.4).

1.3.2 The Maxwell Equations

In SI units, Maxwell's equations for the electric field $\boldsymbol{e} : \mathbb{E}^3 \times \mathbb{R} \to \mathbb{E}^3$ and the magnetic field $\boldsymbol{h} : \mathbb{E}^3 \times \mathbb{R} \to \mathbb{E}^3$ in vacuum are the first-order system of PDEs,

$$\nabla \times \boldsymbol{e} = -\mu \frac{\partial \boldsymbol{h}}{\partial t}, \tag{1.10a}$$

$$\nabla \times \boldsymbol{h} = \boldsymbol{j} + \varepsilon \frac{\partial \boldsymbol{e}}{\partial t}, \tag{1.10b}$$

$$\nabla \cdot \boldsymbol{e} = \rho/\varepsilon, \tag{1.10c}$$

$$\nabla \cdot \boldsymbol{h} = 0, \tag{1.10d}$$

where ε and μ are universal physical constants associated with the vacuum continuum, and the electric current density $\boldsymbol{j}$ and charge density ρ are the sources that generate the electromagnetic field $(\boldsymbol{e}, \boldsymbol{h})$. The sources $(\boldsymbol{j}, \rho)$ may *not* be independently specified, since all electric current is simply charge in motion. Their mathematical connection is the conservation law obtained by substituting (1.10c)

into the divergence of (1.10b),

$$\nabla \cdot \boldsymbol{j} + \frac{\partial \rho}{\partial t} = 0. \tag{1.11}$$

The most important mathematical property of the system (1.10) is that the fields $(\boldsymbol{e}, \boldsymbol{h})$ are a linear mapping of the sources $(\boldsymbol{j}, \rho)$. The most important physical property is that they support wave solutions that propagate at the speed of light.

Note in (1.10) that there are eight equations in only six unknowns, the three elements each of $\boldsymbol{e}$ and $\boldsymbol{h}$. In fact, taking the divergence of (1.10a–b) and using (1.11) gives

$$\frac{\partial}{\partial t} \nabla \cdot \boldsymbol{e} = \frac{\partial}{\partial t} \rho/\varepsilon \tag{1.12a}$$

$$\frac{\partial}{\partial t} \nabla \cdot \boldsymbol{h} = 0, \tag{1.12b}$$

so that (1.10c–d) are simply constraints on the data supplied to the Cauchy problem, in which the fields in $\mathbb{E}^3$ for $t > 0$ are evolved from the initial values $\boldsymbol{e}(\boldsymbol{x}, 0)$ and $\boldsymbol{h}(\boldsymbol{x}, 0)$. Thus, (1.10a–b) comprise the expected six equations in six unknowns, and the remaining equations are attached to the initial condition.

Time-Harmonic Restriction

The time derivatives may be eliminated if we look only for time-harmonic solutions $\boldsymbol{e}(\boldsymbol{x}, t) = \operatorname{Re} \boldsymbol{E}(\boldsymbol{x}) e^{-ikct}$ and $\boldsymbol{h}(\boldsymbol{x}, t) = \operatorname{Re} \boldsymbol{H}(\boldsymbol{x}) e^{-ikct}$ that develop in response to time-harmonic currents $\boldsymbol{j}(\boldsymbol{x}, t) = \operatorname{Re} \boldsymbol{J}(\boldsymbol{x}) e^{-ikct}$, where the *speed of light* c is given by $c := (\mu\varepsilon)^{-1/2}$. Then, assuming $k \neq 0$, the system (1.10) is equivalent to

$$\nabla \times \boldsymbol{E} = ik\eta \boldsymbol{H}, \tag{1.13a}$$

$$\nabla \times \boldsymbol{H} = \boldsymbol{J} - ik\eta^{-1} \boldsymbol{E}, \tag{1.13b}$$

where $\eta := \sqrt{\mu/\varepsilon}$ is yet another universal physical constant, the *vacuum impedance*. Unless $k = 0$, the Fourier transforms of (1.10c–d) are recovered by taking the divergence of each equation in (1.13).

By taking the curl of (1.13a) and inserting (1.13b), it is demonstrated that the first-order system (1.13) is equivalent to the second-order system

$$\nabla \times \nabla \times \boldsymbol{E} - k^2 \boldsymbol{E} = ik\eta \boldsymbol{J}, \tag{1.14a}$$

$$\boldsymbol{H} = \frac{1}{ik\eta} \nabla \times \boldsymbol{E}, \tag{1.14b}$$

in which the fields $\boldsymbol{E}$ and $\boldsymbol{H}$ are decoupled. After a solution $\boldsymbol{E}$ to (1.14a) has been found, the magnetic field may be determined by differentiating $\boldsymbol{E}$ as in (1.14b). In contrast with the time-dependent equations (1.10), auxiliary potentials are not required to decouple the time-independent equations (1.13).

By taking the curl of (1.13b), an alternative equivalent system is

$$\nabla \times \nabla \times \boldsymbol{H} - k^2 \boldsymbol{H} = \nabla \times \boldsymbol{J}, \tag{1.15a}$$

$$\boldsymbol{E} = \frac{1}{ik\eta^{-1}} \left(\boldsymbol{J} - \nabla \times \boldsymbol{H} \right). \tag{1.15b}$$

Once (1.15a) is solved for $\boldsymbol{H}$, the electric field is easily determined with (1.15b).

Boundary Conditions

Maxwell's equations do not apply at material interfaces, where the fields are generally not differentiable. Physical models of source distributions on lower-dimensional manifolds supply the boundary conditions needed to connect fields across an interface.

We shall concern ourselves with solutions only for interfaces that separate vacuum from an ideal metal that conducts electric current without dissipation. Inside such

a metal, the time-dependent electric field $\boldsymbol{e}(\boldsymbol{x}, t)$ vanishes, while the magnetic field $\boldsymbol{h}(\boldsymbol{x}, t)$ may be nonzero but must be *static,* exhibiting no variation with time. For time-harmonic fields with $k \neq 0$, these interior conditions reduce to $\boldsymbol{E} = \boldsymbol{H} = \boldsymbol{0}$.

The appropriate condition at the boundary is the vanishing of the components of $\boldsymbol{E}$ tangent to the conductor surface. A perfect conductor cannot support an electromotive force. The boundary condition is

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.16}$$

where $\hat{\boldsymbol{n}}$ is a unit vector normal to the surface Γ and directed into the vacuum.

The boundary condition (1.16) is sufficient to determine a unique radiating solution of Maxwell's equations exterior to a smooth conductor [80, §3.3].

While the tangential electric field is continuous on crossing Γ, the tangential magnetic field has a jump discontinuity there. If $\boldsymbol{H}_+$ is the one-sided limit at Γ of the magnetic field in G_+, then the jump $\hat{\boldsymbol{n}} \times \boldsymbol{H}_+$ equals the magnitude and direction of a physical current sheet $\boldsymbol{J}_\Gamma$ of zero thickness bound to the interface,

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{H}_+(\boldsymbol{x}) = \boldsymbol{J}_\Gamma \quad \text{for} \quad \boldsymbol{x} \in \Gamma. \tag{1.17}$$

Since $\boldsymbol{J}_\Gamma$ is unlikely to be known a priori, the utility of (1.17) is somewhat less than that of (1.16).

The normal field components may not be specified independently of the tangential components. Together with Maxwell's equations, (1.16) implies a boundary condition for the normal component of the magnetic field. Taking the surface divergence of (1.16) and substituting (1.13a) gives

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{H}(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in \Gamma. \tag{1.18}$$

(The surface curl of $\hat{\boldsymbol{n}}$ vanishes at smooth points of the boundary.) Similarly, taking the surface divergence of (1.17) and substituting (1.13b) gives

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{E}_+(\boldsymbol{x}) = \frac{1}{ik\eta^{-1}} \nabla \cdot \boldsymbol{J}_\Gamma(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma. \tag{1.19}$$

Using (1.11), the right-hand side of (1.19) can be rewritten as the product of ε^{-1} and the surface charge density.

While the normal component of $\boldsymbol{H}$ is continuous at the interface, the normal component of $\boldsymbol{E}$ has a jump discontinuity.

Neither boundary condition (1.18) nor (1.19) is sufficient to determine a unique solution of Maxwell's equations.

At singular points of the surface, such as at edges and corners, a continuously turning tangent plane does not exist, and boundary conditions (1.16)–(1.19) cannot be applied. An edge condition [107] [87, Ch. 9] may be necessary to select the appropriate solution in the presence of such singularities. A typical condition is that $\|\boldsymbol{E}\|^2$ and $\|\boldsymbol{H}\|^2$ be integrable on any region $\bar{N} \setminus G_-$, where $\bar{N}$ is a closed neighborhood of a point singularity and G_- is the interior of the conductor. This condition guarantees that bounded regions of space contain a finite amount of electromagnetic energy.

For reference, I provide boundary conditions for other material junctions. An *impedance* boundary condition models the behavior of an imperfect electric conductor. The tangential components of $\boldsymbol{E}$ and $\boldsymbol{H}$ are related according to

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \big(\boldsymbol{E}_+(\boldsymbol{x}) - Z\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{H}_+(\boldsymbol{x})\big) = \boldsymbol{0} \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.20}$$

where $Z \in \mathbb{C}$ is the surface impedance.

A *transmission* boundary condition connects the fields at the junction of simple dielectric materials. A *simple* material is one that is linear, homogeneous, and

isotropic in its response to an electromagnetic field. Upon substitution of values of ε and μ specific to the material, the vacuum equations (1.13) apply to simple materials in a macroscopic sense [83, §6.7]. The macroscopic fields are not the actual fields that exist between atoms of the material, but are smoothed versions of the rapidly fluctuating microscopic fields. The transmission boundary condition is the continuity of the tangential components of the macroscopic fields $(\boldsymbol{E}, \boldsymbol{H})$. Maxwell's equations then imply that the normal components of $(\varepsilon\boldsymbol{E}, \mu\boldsymbol{H})$ are also continuous.

A Scattering Problem

Referring to Figure 1.2, consider a solution to Maxwell's equations in the exterior G_+ of a perfectly conducting obstacle with boundary Γ. The electromagnetic field $(\boldsymbol{E}, \boldsymbol{H})$ is generated by impressed current sources $\boldsymbol{J}_{\text{inc}}$ that lie in G_+, and by a current sheet $\boldsymbol{J}_\Gamma$ on the scattering obstacle that is induced in order to satisfy the boundary condition (1.16). The impressed currents may be, for instance, the currents developed on an antenna that illuminates the obstacle. It is assumed that the antenna is far away from the scattering obstacle, so that the presence of the induced currents $\boldsymbol{J}_\Gamma$ causes a negligible perturbation of the impressed currents $\boldsymbol{J}_{\text{inc}}$.

The total electric field satisfies (1.14a), supplemented by boundary conditions at Γ and at infinity:

$$\nabla \times \nabla \times \boldsymbol{E}(\boldsymbol{x}) - k^2 \boldsymbol{E}(\boldsymbol{x}) = ik\eta \boldsymbol{J}_{\text{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.21a}$$

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.21b}$$

$$\lim_{r\to\infty} \boldsymbol{x} \times \nabla \times \boldsymbol{E}(\boldsymbol{x}) + ikr\boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{where} \quad r := \|\boldsymbol{x}\|. \tag{1.21c}$$

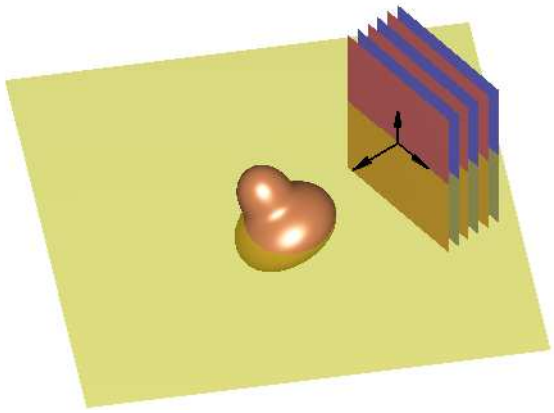


Figure 1.2: A plane electromagnetic wave irradiates a perfectly conducting obstacle in three-dimensional space.

Once these equations are solved for $\boldsymbol{E}$, the magnetic field is given by

$$\boldsymbol{H} = \frac{1}{ik\eta}\nabla \times \boldsymbol{E}. \tag{1.21d}$$

The condition (1.21c) is the Silver–Müller radiation condition, the vector field counterpart of the Sommerfeld radiation condition. The magnetic field, too, will satisfy the Silver–Müller condition. It is the appropriate boundary condition under the assumption that the obstacle Γ and the impressed sources $\boldsymbol{J}_{\text{inc}}$ are contained in a sphere with finite diameter.

In the absence of the obstacle, the impressed currents generate an incident electromagnetic field $(\boldsymbol{E}_{\text{inc}}, \boldsymbol{H}_{\text{inc}})$. Typically in a scattering problem, the incident field is specified in lieu of the impressed currents. In any case, when the impressed sources have bounded support, the incident field satisfies the equations

$$\nabla \times \nabla \times \boldsymbol{E}_{\text{inc}}(\boldsymbol{x}) - k^2 \boldsymbol{E}_{\text{inc}}(\boldsymbol{x}) = ik\eta \boldsymbol{J}_{\text{inc}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{E}^3, \tag{1.22a}$$

$$\lim_{r\to\infty} \boldsymbol{x} \times \nabla \times \boldsymbol{E}_{\text{inc}}(\boldsymbol{x}) + ikr\boldsymbol{E}_{\text{inc}}(\boldsymbol{x}) = \boldsymbol{0}, \quad r := \|\boldsymbol{x}\|, \tag{1.22b}$$

$$\boldsymbol{H}_{\text{inc}} = \frac{1}{ik\eta}\nabla \times \boldsymbol{E}_{\text{inc}}, \tag{1.22c}$$

and these equations may be solved for $(\boldsymbol{E}_{\text{inc}}, \boldsymbol{H}_{\text{inc}})$ if the current $\boldsymbol{J}_{\text{inc}}$ is provided. Often the incident field is specified to be a plane wave, and such a field can only be approximately generated by some $\boldsymbol{J}_{\text{inc}}$ with bounded support. However, the departure of the actual incident field from a plane wave may be made arbitrarily small in any bounded region of space enclosing the obstacle, so there is little need to provide a separate formulation for plane wave illumination.

Now, using the linearity of the equations (1.21), the total fields are taken to be the superposition of the incident fields and the scattered fields generated by the surface current $\boldsymbol{J}_\Gamma$. The scattered fields are

$$\boldsymbol{E}_{\text{s}} := \boldsymbol{E} - \boldsymbol{E}_{\text{inc}}, \tag{1.23a}$$

$$\boldsymbol{H}_{\text{s}} := \boldsymbol{H} - \boldsymbol{H}_{\text{inc}}, \tag{1.23b}$$

and by subtracting the equations in the system (1.22) from the corresponding equations in (1.21), the scattered electric field must solve the following problem, the 3-D vector field analog of Problem 1.1,

Problem 1.2 (Exterior Vector Helmholtz) *Given: (1) a real constant $k > 0$, (2) a simple, bounded, orientable, very smooth surface Γ in Euclidean space $\mathbb{E}^3$, with a continuous unit normal $\hat{\boldsymbol{n}} : \Gamma \to \mathbb{E}^3$, and (3) a function $\boldsymbol{E}_{\text{inc}} : \Gamma \to \mathbb{C}$ on the surface. Construct a function $\boldsymbol{E}_{\text{s}} : G_+ \to \mathbb{C}$ such that*

$$\nabla \times \nabla \times \boldsymbol{E}_{\text{s}}(\boldsymbol{x}) - k^2 \boldsymbol{E}_{\text{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \textit{for} \quad \boldsymbol{x} \in G_+, \tag{1.24a}$$

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}_{\text{s}}(\boldsymbol{x}) = -\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}_{\text{inc}}(\boldsymbol{x}) \quad \textit{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.24b}$$

$$\lim_{r\to\infty} \boldsymbol{x} \times \nabla \times \boldsymbol{E}_{\text{s}}(\boldsymbol{x}) + ikr\boldsymbol{E}_{\text{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \textit{where} \quad r := \|\boldsymbol{x}\|, \tag{1.24c}$$

where the latter condition is to hold uniformly with respect to the direction of the

vector $\boldsymbol{x}$.

With the addition of a boundary condition at singular points, the problem may be extended to cover piecewise smooth surfaces. The scattered magnetic field is given by

$$\boldsymbol{H}_{\mathrm{s}} = \frac{1}{ik\eta}\nabla \times \boldsymbol{E}_{\mathrm{s}} \tag{1.25}$$

after the equations (1.24) have been solved for $\boldsymbol{E}_{\mathrm{s}}$.

The double curl equation (1.24a) is equivalent to the pair of equations

$$\Delta \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) + k^2 \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) = \boldsymbol{0}, \tag{1.26a}$$

$$\nabla \cdot \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) = 0. \tag{1.26b}$$

Each rectangular component of $\boldsymbol{E}_{\mathrm{s}}$ satisfies the Helmholtz equation, but those components are coupled through (1.26b).

This elliptic system is peculiar because on G_+ there are four equations in three unknowns, while on Γ there are only two boundary conditions. (The component of $\boldsymbol{E}_{\mathrm{s}}$ normal to the boundary is not constrained.) An equivalent system is [30, Ch. 4]

$$\Delta \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) + k^2 \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.27a}$$

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) = -\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{E}_{\mathrm{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.27b}$$

$$\frac{\partial}{\partial n}\big(\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x})\big) = \nabla \cdot \big(\boldsymbol{E}_{\mathrm{inc}}(\boldsymbol{x}) - (\hat{\boldsymbol{n}} \cdot \boldsymbol{E}_{\mathrm{inc}})\hat{\boldsymbol{n}}(\boldsymbol{x})\big) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.27c}$$

$$\lim_{r\to\infty} \boldsymbol{x} \times \nabla \times \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) - \boldsymbol{x}\nabla \cdot \boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) + ikr\boldsymbol{E}_{\mathrm{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{where} \quad r := \|\boldsymbol{x}\|, \tag{1.27d}$$

Here the equation (1.26b) on G_+ has been replaced with an extra boundary condition (1.27c) on Γ. The radiation boundary condition is also modified. This set of equations is a system of second-order PDEs with one condition per unknown at each boundary point.

The boundary condition (1.27c) is a Neumann condition on the normal component of the scattered field. The right-hand side is the surface divergence of the tangential components of the incident field. Using vector Green identities, it can be shown that any solution of (1.27) must also satisfy (1.26b).

Another formulation of Problem 1.2 emphasizes the magnetic field. Starting from (1.15) instead of (1.14), the equations read

$$\nabla \times \nabla \times \boldsymbol{H}_{\mathrm{s}}(\boldsymbol{x}) - k^2 \boldsymbol{H}_{\mathrm{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.28a}$$

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \nabla \times \boldsymbol{H}_{\mathrm{s}}(\boldsymbol{x}) = -\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \nabla \times \boldsymbol{H}_{\mathrm{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.28b}$$

$$\lim_{r \to \infty} \boldsymbol{x} \times \nabla \times \boldsymbol{H}_{\mathrm{s}}(\boldsymbol{x}) + ikr \boldsymbol{H}_{\mathrm{s}}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{where} \quad r := \|\boldsymbol{x}\|. \tag{1.28c}$$

Once these equations are solved for $\boldsymbol{H}_{\mathrm{s}}$, the electric field is given by

$$\boldsymbol{E}_{\mathrm{s}} = -\frac{1}{ik\eta^{-1}} \nabla \times \boldsymbol{H}_{\mathrm{s}}. \tag{1.29}$$

The boundary condition $\hat{\boldsymbol{n}} \cdot \boldsymbol{H}_{\mathrm{s}} = -\hat{\boldsymbol{n}} \cdot \boldsymbol{H}_{\mathrm{inc}}$ is insufficient to determine a unique solution, and so (1.28b) is the same as (1.24b). The boundary condition (1.28c) is obtained by substituting (1.29) into (1.24c).

I have nothing further to say about the vector equations in their full generality, but proceed now to distill them into the Helmholtz equation.

Reduction of 3-D Vector Fields to 2-D Scalar Fields

It is well-known [80, §3.12] that if all dependent variables $(\boldsymbol{J}, \boldsymbol{E}, \boldsymbol{H})$ are invariant in the z-direction then the vector scattering problem splits into two separate scalar problems.

In particular, split the electric field as $\boldsymbol{E} = \boldsymbol{E}_{\mathrm{TE}} + \boldsymbol{E}_{\mathrm{TM}}$, where the rectangular

components of $\boldsymbol{E}_{\mathrm{TE}}$ and $\boldsymbol{E}_{\mathrm{TM}}$ are

$$\boldsymbol{E}_{\mathrm{TE}} := (E_x(x,y), E_y(x,y), 0), \tag{1.30a}$$

$$\boldsymbol{E}_{\mathrm{TM}} := (0, 0, E_z(x,y)). \tag{1.30b}$$

The subscript TE stands for "transverse electric," and refers to a solution of the Maxwell equations in which the electric field is polarized perpendicular to the z-axis of translational symmetry, as is evident in (1.30a). The subscript TM stands for "transverse magnetic," and it indicates that the magnetic field is polarized perpendicular to $\hat{\boldsymbol{z}}$. By (1.13a), the rectangular components of the magnetic fields paired with these electric fields are

$$\boldsymbol{H}_{\mathrm{TE}} = \frac{1}{ik\eta}\left(0, 0, \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right), \tag{1.31a}$$

$$\boldsymbol{H}_{\mathrm{TM}} = \frac{1}{ik\eta}\left(\frac{\partial E_z}{\partial y}, -\frac{\partial E_z}{\partial x}, 0\right), \tag{1.31b}$$

and the latter justifies the name "transverse magnetic." To be consistent with (1.13b), the current density must also be split as $\boldsymbol{J} = \boldsymbol{J}_{\mathrm{TE}} + \boldsymbol{J}_{\mathrm{TM}}$, with components

$$\boldsymbol{J}_{\mathrm{TE}} := (J_x(x,y), J_y(x,y), 0), \tag{1.32a}$$

$$\boldsymbol{J}_{\mathrm{TM}} := (0, 0, J_z(x,y)). \tag{1.32b}$$

With these definitions, any data–solution triple $(\boldsymbol{J}, \boldsymbol{E}, \boldsymbol{H})$ may be split as

$$(\boldsymbol{J}, \boldsymbol{E}, \boldsymbol{H}) = (\boldsymbol{J}_{\mathrm{TE}}, \boldsymbol{E}_{\mathrm{TE}}, \boldsymbol{H}_{\mathrm{TE}}) + (\boldsymbol{J}_{\mathrm{TM}}, \boldsymbol{E}_{\mathrm{TM}}, \boldsymbol{H}_{\mathrm{TM}}), \tag{1.33}$$

where each triple on the right-hand side itself satisfies Maxwell's equations (1.13).

Substituting the TM expressions into (1.14a), and observing that (1.30b) implies $\nabla \cdot \boldsymbol{E}_{\mathrm{TM}} = 0$, we obtain the Helmholtz equation

$$\Delta E_z(x,y) + k^2 E_z(x,y) = -ik\eta J_z(x,y) \tag{1.34}$$

for the scalar function E_z. From a solution to (1.34), the magnetic field can be recovered with (1.31b).

Substituting the TE expressions into (1.15a), and using the fact that $\nabla \cdot \boldsymbol{H}_{\mathrm{TE}} = 0$, we obtain the Helmholtz equation

$$\Delta H_z(x,y) + k^2 H_z(x,y) = -\frac{\partial J_y}{\partial x}(x,y) + \frac{\partial J_x}{\partial y}(x,y) \tag{1.35}$$

for the scalar function H_z. The electric field can be recovered with (1.15b),

$$\boldsymbol{E}_{\mathrm{TE}} = \frac{1}{ik\eta^{-1}} \left(J_x - \frac{\partial H_z}{\partial y}, J_y + \frac{\partial H_z}{\partial x}, 0 \right), \tag{1.36}$$

after a solution to (1.35) has been found.

We may specialize these results to the scattering configuration of Figure 1.3, in which the symmetry of the cylindrical object and the incident field guarantees that the induced surface currents and the scattered fields are invariant in the z-direction.

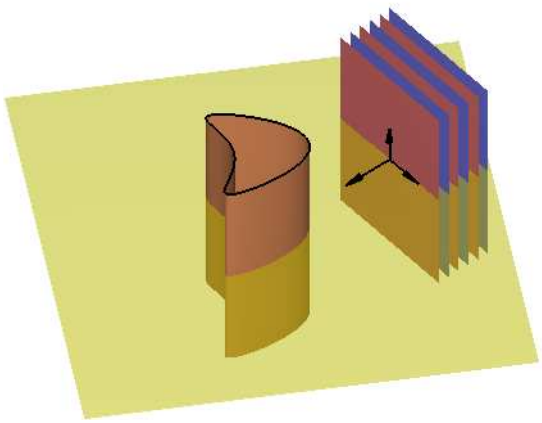


Figure 1.3: A plane wave with wavefronts orthogonal to the x-y plane illuminates a perfectly conducting cylinder with noncircular cross section. The axis of the cylinder is parallel to the z-axis. If either the electric field or the magnetic field is polarized in the z-direction, then the scattering is described by the 2-D Helmholtz equation.

In the TM scattering problem, the incident field $\boldsymbol{E}_{\text{inc}}$ is polarized in the z-direction, and by (1.24b) the scattered field $\boldsymbol{E}_{\text{s}}$ will also be polarized in the z-direction. Collapsing the z-dimension, let $\boldsymbol{x}$ now be a point in the x-y plane and let Γ be the boundary of the open cross section G_- of the cylinder in the x-y plane. In Figure 1.1, the cross section is a circle, and the terminology of the problem was given in two dimensions. But as Figure 1.3 indicates, every 2-D scattering problem may be reinterpreted as the TM scattering from a 3-D cylinder.

After making the replacements $\boldsymbol{E}_{\text{inc}} = f(\boldsymbol{x})\hat{\boldsymbol{z}}$ and $\boldsymbol{E}_{\text{s}} = u(\boldsymbol{x})\hat{\boldsymbol{z}}$, the vector equations (1.24) simplify to

$$\Delta u(\boldsymbol{x}) + k^2 u(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.37a}$$

$$u(\boldsymbol{x}) = -f(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.37b}$$

$$\lim_{r\to\infty} \boldsymbol{x} \cdot \nabla u(\boldsymbol{x}) - ikru(\boldsymbol{x}) = 0 \quad \text{where} \quad r := \|\boldsymbol{x}\|. \tag{1.37c}$$

Once these equations are solved for u, the scattered magnetic field is given by

$$\boldsymbol{H}_{\text{s}} = \frac{1}{ik\eta}\left(\frac{\partial u}{\partial y}\hat{\boldsymbol{x}} - \frac{\partial u}{\partial x}\hat{\boldsymbol{y}}\right). \tag{1.38}$$

The equations (1.37a–b) are identical to equations (1.1a–b). The condition (1.37c) is equivalent to

$$\frac{\partial u}{\partial r} - iku = o\Big(\frac{1}{r}\Big) \quad \text{as} \quad r \to \infty, \tag{1.39}$$

which is stronger than the Sommerfeld condition (1.1c).

In the TE scattering problem, the incident field $\boldsymbol{H}_{\text{inc}}$ is polarized in the z-direction, and by (1.28b) the scattered field $\boldsymbol{H}_{\text{s}}$ will also be polarized in the z-direction. Making the replacements $\boldsymbol{H}_{\text{inc}} = g(\boldsymbol{x})\hat{\boldsymbol{z}}$ and $\boldsymbol{H}_{\text{s}} = u(\boldsymbol{x})\hat{\boldsymbol{z}}$, the vector equations (1.28) simplify to

$$\Delta u(\boldsymbol{x}) + k^2 u(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.40a}$$

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) = -\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \nabla g(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.40b}$$

$$\lim_{r\to\infty} \boldsymbol{x} \cdot \nabla u(\boldsymbol{x}) - ikru(\boldsymbol{x}) = 0 \quad \text{where} \quad r := \|\boldsymbol{x}\|. \tag{1.40c}$$

Once these equations are solved for u, the electric field is given by

$$\boldsymbol{E}_{\mathrm{s}} = -\frac{1}{ik\eta^{-1}} \left(\frac{\partial u}{\partial y}\hat{\boldsymbol{x}} - \frac{\partial u}{\partial x}\hat{\boldsymbol{y}} \right). \tag{1.41}$$

In view of (1.40b), when the obstacle is a perfect conductor, TE illumination produces the Neumann BVP for the Helmholtz equation. In the general boundary condition (1.2), the differential operator is $\mathcal{B} = \hat{\boldsymbol{n}} \cdot \nabla$ and the boundary data f is the normal derivative of the incident magnetic field.

If the incident field $\boldsymbol{E}_{\mathrm{inc}}$ in Figure 1.3 has an arbitrary polarization, then the solution to Problem 1.2 is the sum of solutions to TE and TM scalar problems. The excitation for the TM subproblem is given by (1.30b) as the orthogonal projection of $\boldsymbol{E}_{\mathrm{inc}}$ onto the z-axis, and the remainder (1.30a) is the excitation for the TE subproblem.

In general the computation of scattering from a metal cylinder requires the solution of both Dirichlet and Neumann problems for the Helmholtz equation. In its present state, however, my code treats only the Dirichlet problem. It is capable of computing the solution to the cylindrical scattering problem when the incident electric field is polarized parallel to the cylinder axis.

Expanding the code to include the Neumann problem would enable solutions not only for arbitrary polarizations but also for obstacle materials that give transmission or impedance boundary conditions.

1.3.3 The Schrödinger Equation

In SI units, Schrödinger's equation for the motion of a nonrelativistic particle with mass m through a conservative force field with potential energy $V(\boldsymbol{x})$ is the linear second-order PDE

$$-\frac{\hbar^2}{2m}\Delta\varphi(\boldsymbol{x},t)+V(\boldsymbol{x})\varphi(\boldsymbol{x},t)=i\hbar\frac{\partial\varphi}{\partial t}(\boldsymbol{x},t), \tag{1.42}$$

where $\hbar$ is Planck's constant and $\varphi:\mathbb{E}^3\times\mathbb{R}\to\mathbb{C}$ is a field associated with the particle. That field may be normalized so that $|\varphi|^2$ is a probability density function. Under this normalization, the probability that the particle lies in any region $R\subset\mathbb{E}^3$ at some time t is $\int_R|\varphi(\boldsymbol{x},t)|^2\,d\boldsymbol{x}$.

The time-independent Schrödinger equation results from a Fourier transform of (1.42) with respect to time t. Often interest is restricted to time-harmonic functions, $\varphi(\boldsymbol{x},t)=\psi(\boldsymbol{x})e^{-i\omega t}$, which have a single Fourier component. Substitution into (1.42) gives

$$-\frac{\hbar^2}{2m}\Delta\psi(\boldsymbol{x})+V(\boldsymbol{x})\psi(\boldsymbol{x})=\hbar\omega\psi(\boldsymbol{x}), \tag{1.43}$$

which shows that ψ is an eigenfunction of the Hamiltonian operator $-\frac{\hbar^2}{2m}\Delta+V(\boldsymbol{x})$. The eigenvalue $\hbar\omega$ is the energy E of the particle, which does not change with time.

Since φ is complex-valued, I have written $\varphi=\psi e^{-i\omega t}$ instead of $\varphi=\operatorname{Re}\psi e^{-i\omega t}$. It is unnatural to impose on (1.43) a restriction $\omega\geq 0$. The physical meaning of any negative-energy solutions ($\omega<0$) should be investigated.

Consider the particle scattering from an infinite energy barrier with boundary Γ separating a (possibly empty) bounded domain G_- from an unbounded domain G_+,

$$V(\boldsymbol{x})=\begin{cases}\infty & \text{if } \boldsymbol{x}\in\Gamma\cup G_-,\\ 0 & \text{if } \boldsymbol{x}\in G_+.\end{cases} \tag{1.44}$$

Then (1.43) reduces to the Helmholtz equation,

$$\Delta\psi(\boldsymbol{x}) + k^2\psi(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.45}$$

where $\hbar k = \sqrt{2mE}$ is the particle momentum.

The particle can never be found inside the energy barrier, so $\psi(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in G_-$. The probability field is continuous on $\mathbb{E}^3$, so the appropriate boundary condition to apply to a solution of (1.45) is $\psi(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \Gamma$.

It is natural to impose the Sommerfeld radiation condition at infinity, but then, lacking a source term on the right-hand side of (1.45), the solution will be $\psi \equiv 0$. Rather than adding a source term as in (1.21a), which would require the introduction of additional physics, I will simply assume that there is some external source that produces a nonzero incident field $\psi_{\text{inc}}(\boldsymbol{x})$ that satisfies the Helmholtz equation for $\boldsymbol{x} \in \mathbb{E}^3$.

As in Section 1.3.2, the linearity of Schrödinger's equation may be utilized to decompose the wave function as $\psi = \psi_{\text{inc}} + \psi_{\text{s}}$, where ψ_{s} is a scattered field which is the solution of a Dirichlet BVP,

$$\Delta\psi_{\text{s}}(\boldsymbol{x}) + k^2\psi_{\text{s}}(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.46a}$$

$$\psi_{\text{s}}(\boldsymbol{x}) = -\psi_{\text{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.46b}$$

$$\lim_{r\to\infty} r\left(\frac{\partial\psi_{\text{s}}}{\partial r}(\boldsymbol{x}) - ik\psi_{\text{s}}(\boldsymbol{x})\right) = 0 \quad \text{where} \quad r := \|\boldsymbol{x}\|, \tag{1.46c}$$

where (1.46c) is the 3-D Sommerfeld radiation condition.

For the case $\omega < 0$, the wavenumber k is imaginary. As indicated in Table 6.1, the negative-energy solutions are evanescent, decaying exponentially fast away from the boundary Γ.

In the subsequent discussion, I use terminology specific to the scattering of a clas-

sical electromagnetic wave, but in every instance the solution may be reinterpreted as the unnormalized probability field for the scattering of a matter wave from an impenetrable 2-D obstacle. One reason I stick to classical electromagnetics is that the potential energy (1.44) may have limited application. The 3-D Coulomb potential energy $V(\boldsymbol{x}) = \|\boldsymbol{x}\|^{-2}$ is representative of more realistic forces, but putting that into (1.43) gives a variable-coefficient exterior problem, to which quite different numerical techniques must be applied.

1.4 From PDE to Integral Equation

The standard way to transform the boundary value problem (1.1) into an integral equation begins by mirroring Green's development of an integral representation for a solution to the Laplace equation [92]. The principal tool in that development is the divergence theorem of vector calculus.

Rather than pursue that angle, I will take a more expedient approach that starts by assuming that the scattered field can be expressed as a linear combination of selected elementary solutions.

1.4.1 Elementary Solutions

We can compile a catalog of elementary radiating solutions to the Helmholtz equation. Elementary solutions are loosely defined as the fields produced by canonical source distributions—point, surface, volume. Since the actual scattered field will be represented as a superposition of those elementary fields, we restrict our interest to elementary solutions that are free of singularities in the domain G_+. An elementary

solution v should satisfy

$$\Delta v(\boldsymbol{x}) + k^2 v(\boldsymbol{x}) = 0 \quad \text{for} \quad \boldsymbol{x} \in G_+, \tag{1.47a}$$

$$\frac{\partial v}{\partial r}(\boldsymbol{x}) - ikv(\boldsymbol{x}) = o\Big(\frac{1}{\sqrt{r}}\Big) \quad \text{where} \quad r := \|\boldsymbol{x}\|. \tag{1.47b}$$

Since no regard is given to the values assumed by v on the boundary of G_+, this function is not generally a solution of Problem 1.1. The less restrictive problem (1.47) has infinitely many solutions.

Consider, for example, the field generated by a point source. If (1.47a) is to be satisfied, the point source must be located at some position $\boldsymbol{y}$ outside of G_+. The corresponding field $v_{\boldsymbol{y}}$ is the solution of

$$\Delta v_{\boldsymbol{y}}(\boldsymbol{x}) + k^2 v_{\boldsymbol{y}}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{y}) \quad \text{for} \quad \boldsymbol{x} \in \mathbb{E}^2, \tag{1.48a}$$

$$\frac{\partial v_{\boldsymbol{y}}}{\partial r}(\boldsymbol{x}) - ikv_{\boldsymbol{y}}(\boldsymbol{x}) = o\Big(\frac{1}{\sqrt{r}}\Big) \quad \text{where} \quad r := \|\boldsymbol{x}\|, \tag{1.48b}$$

which is known to be

$$v_{\boldsymbol{y}}(\boldsymbol{x}) = -\tfrac{i}{4} H_0(k\|\boldsymbol{x} - \boldsymbol{y}\|), \tag{1.49}$$

where H_0 is the Hankel function [1, Ch. 9] of first kind and order 0. The Hankel function is a solution of Bessel's differential equation, which is the radial equation that materializes when separation of variables in polar coordinates is applied to (1.48a).

The elementary solution (1.49) is the radiating *fundamental solution*[3] $\Phi(\boldsymbol{x}, \boldsymbol{y})$ of the Helmholtz equation. It is also known as the free-space Green function of the Helmholtz equation.

The fundamental solution may be multiplied by an arbitrary complex number

[3]Most authors define the fundamental solution to have the opposite sign, i.e., $\frac{i}{4} H_0(k\|\boldsymbol{x} - \boldsymbol{y}\|)$, so that $(\Delta_{\boldsymbol{x}} + k^2)\Phi(\boldsymbol{x}, \boldsymbol{y}) = -\delta(\boldsymbol{x} - \boldsymbol{y})$. That choice changes the sign of the step discontinuities in the jump relations (1.62) and (1.72) at surface distributions.

without violating (1.47). A particular choice of multiplier gives the electric field produced by an oscillating point current in a TM configuration. (The point current in 2-D space becomes a line current in 3-D space.) Maxwell's equations for the current density $\boldsymbol{J}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{y})\hat{\boldsymbol{z}}$ are

$$\nabla \times \boldsymbol{E}(\boldsymbol{x}) = ik\eta \boldsymbol{H}(\boldsymbol{x}), \tag{1.50a}$$

$$\nabla \times \boldsymbol{H}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{y})\hat{\boldsymbol{z}} - ik\eta^{-1}\boldsymbol{E}(\boldsymbol{x}), \tag{1.50b}$$

and upon substitution of $v(\boldsymbol{x})\hat{\boldsymbol{z}}$ for $\boldsymbol{E}(\boldsymbol{x})$ we have $\Delta v(\boldsymbol{x}) + k^2 v(\boldsymbol{x}) = -ik\eta\delta(\boldsymbol{x} - \boldsymbol{y})$. The field generated by the point current is therefore $-ik\eta\Phi(\boldsymbol{x}, \boldsymbol{y})$.

The derivatives of $\Phi(\boldsymbol{x}, \boldsymbol{y})$ with respect to either $\boldsymbol{x}$ or $\boldsymbol{y}$ are also elementary solutions. A directional derivative in the variable $\boldsymbol{y}$ is the field generated by a current dipole. Second-order derivatives in $\boldsymbol{y}$ give the fields generated by current quadrupoles. In general, a derivative of order p gives the field of a multipole with 2^p poles.

Another class of elementary solutions are the fields generated by continuous current distributions. If the current $J(\boldsymbol{x})$ is supported on some region R contained in G_-, the corresponding elementary solution is

$$v(\boldsymbol{x}) = -ik\eta \int_R \Phi(\boldsymbol{x}, \boldsymbol{y}) J(\boldsymbol{y})\, d\boldsymbol{y}. \tag{1.51}$$

The currents may instead be supported only on a lower-dimensional set such as the boundary Γ. The latter choice is interesting because it has a clear physical connection to the scattering problem: The scattered field is generated by a continuous sheet of electric current on the conductor surface. Denoting this surface current density by σ, the corresponding elementary solution is

$$v(\boldsymbol{x}) = -ik\eta \int_\Gamma \Phi(\boldsymbol{x}, \boldsymbol{y})\sigma(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}), \tag{1.52}$$

in which $d\Gamma(\boldsymbol{y})$ is the differential element of arc length at the point $\boldsymbol{y}$. The monopole distribution σ is also known as a single layer. A double layer is a surface distribution of dipoles, which generates the elementary solution

$$v(\boldsymbol{x}) = -ik\eta \int_\Gamma \frac{\partial \Phi(\boldsymbol{x}, \boldsymbol{y})}{\partial n(\boldsymbol{y})} \chi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}), \tag{1.53}$$

where χ is the source density and the dipoles are everywhere oriented in the direction of the surface normal. The directional derivative of $\Phi(\boldsymbol{x}, \boldsymbol{y})$ is taken with respect to the variable $\boldsymbol{y}$.

Table 1.1 lists these elementary solutions. We may choose to represent the solution of Problem 1.1 using only a physically correct source distribution—the single layer—or we may choose nonphysical sources. The use of nonphysical sources is just an application of the equivalence principle [80, §3.5]. It offers greater flexibility than the restriction to physical sources, and that flexibility can be utilized to construct an integral equation with desirable mathematical properties.

1.4.2 Scattering Integral Equations

All the elementary solutions already satisfy (1.1a) and (1.1c). Any linear combination of those elementary solutions also satisfies those equations. The only equation left to satisfy is the Dirichlet boundary condition (1.1b). If the solution is represented by the field of a continuous source distribution, such as v_c, v_d, or v_e in Table 1.1, then forcing the field to satisfy (1.1b) will generate an integral equation.

All integral equations developed in this section are specific to TM illumination of a metal cylinder. The incident field does not, however, need to be a plane wave.

Table 1.1: Some Elementary 2-D Solutions

(a) Point monopole of amplitude I at position $\boldsymbol{y}$

$$v_a(\boldsymbol{x}) = -ik\eta I\Phi(\boldsymbol{x},\boldsymbol{y})$$

(b) Point dipole of vector amplitude $I\hat{\boldsymbol{d}}$ at position $\boldsymbol{y}$

$$v_b(\boldsymbol{x}) = -ik\eta I\hat{\boldsymbol{d}}\cdot\nabla_{\boldsymbol{y}}\Phi(\boldsymbol{x},\boldsymbol{y})$$

(c) Single layer of density σ on curve Γ

$$v_c(\boldsymbol{x}) = -ik\eta\int_\Gamma \Phi(\boldsymbol{x},\boldsymbol{y})\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y})$$

(d) Double layer of density χ on curve Γ

$$v_d(\boldsymbol{x}) = -ik\eta\int_\Gamma \frac{\partial\Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{y})}\chi(\boldsymbol{y})\,d\Gamma(\boldsymbol{y})$$

(e) Continuous monopole distribution of density J in the region R

$$v_e(\boldsymbol{x}) = -ik\eta\int_R \Phi(\boldsymbol{x},\boldsymbol{y})J(\boldsymbol{y})\,d\boldsymbol{y}$$

Electric Field Integral Equation

The electric field integral equation (EFIE) derives from a physical source representation. Since the scattered field is generated by a current sheet on the metallic obstacle, it is given by an expression of type (1.52). The unknown currents σ are determined by forcing the single-layer field to satisfy the Dirichlet condition. In the limit as the point $\boldsymbol{x}$ approaches the boundary Γ from the exterior G_+, the scattered field must take on the values $-f(\boldsymbol{x}) = -u_{\text{inc}}(\boldsymbol{x})$, so

$$-ik\eta\int_\Gamma \Phi(\boldsymbol{x},\boldsymbol{y})\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) = -u_{\text{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x}\in\Gamma \tag{1.54}$$

is the desired integral equation [115]. The unknown function σ appears in the integrand, where it is multiplied by the *kernel* $\Phi(\boldsymbol{x}, \boldsymbol{y})$.

Once σ is found by solving (1.54), the scattered field can be computed by evaluating the integral

$$u(\boldsymbol{x}) = -ik\eta \int_{\Gamma} \Phi(\boldsymbol{x}, \boldsymbol{y})\sigma(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}) \tag{1.55}$$

at points $\boldsymbol{x} \in G_+$.

Magnetic Field Integral Equation

The EFIE is an integral equation of the *first kind.* In an integral equation of the *second kind,* the unknown function also appears as a free term outside the integral operator. The magnetic field integral equation (MFIE) is a well-known example of a second-kind equation [104] [115].

The MFIE starts with the same physical representation of the scattered field that was the starting point for the EFIE. But while the EFIE was derived from the boundary condition (1.16) for the electric field, the MFIE is constructed from the boundary condition (1.17) for the magnetic field. Substituting the TM magnetic field expression (1.38), the required cross product at the boundary is

$$\hat{\boldsymbol{n}}(\boldsymbol{x}) \times \boldsymbol{H}_+(\boldsymbol{x}) = -\hat{\boldsymbol{z}} \frac{1}{ik\eta} \hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \nabla \big(u(\boldsymbol{x}) + u_{\text{inc}}(\boldsymbol{x})\big). \tag{1.56}$$

Since this cross product must equal the surface current density $\sigma\hat{\boldsymbol{z}}$, the boundary condition is

$$\frac{\partial u}{\partial n}(\boldsymbol{x}) = -\frac{\partial u_{\text{inc}}}{\partial n}(\boldsymbol{x}) - ik\eta\sigma(\boldsymbol{x}), \tag{1.57}$$

where u has the same expression (1.55) as it did in the development of the EFIE.

Application of (1.57) to that expression seems to generate the integral equation

$$\int_\Gamma \frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{x})}\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) = \frac{1}{ik\eta}\frac{\partial u_{\text{inc}}}{\partial n}(\boldsymbol{x}) + \sigma(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x}\in\Gamma, \tag{1.58}$$

where the normal derivative has been exchanged with the integration.

We have made a mistake, however, and (1.58) is incorrect. The problem is that the elementary solution

$$v(\boldsymbol{x}) = \int_\Gamma \frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{x})}\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) \tag{1.59}$$

is discontinuous at boundary points $\boldsymbol{x}\in\Gamma$. In either G_- or G_+ it is an analytic function, but it suffers a step discontinuity in crossing the curve Γ. Since the solution (1.55) is evaluated only at points $\boldsymbol{x}\in G_+$, the correction to (1.58) is

$$\lim_{\boldsymbol{z}\to\boldsymbol{x}} \int_\Gamma \frac{\partial \Phi(\boldsymbol{z},\boldsymbol{y})}{\partial n(\boldsymbol{x})}\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) = \frac{1}{ik\eta}\frac{\partial u_{\text{inc}}}{\partial n}(\boldsymbol{x}) + \sigma(\boldsymbol{x}), \tag{1.60}$$

where $\boldsymbol{x}\in\Gamma$ and $\boldsymbol{z}\in G_+$.

The discontinuity has a simple characterization, given by a *jump relation* [30, §2.4–2.5]. Let $v_+(\boldsymbol{x})$ be the one-sided limit of (1.59) as the boundary point $\boldsymbol{x}\in\Gamma$ is approached from the exterior G_+, and let $v_-(\boldsymbol{x})$ be the interior limit. More precisely,

$$v_\pm(\boldsymbol{x}) := \lim_{\delta\to 0+} v(\boldsymbol{x} \pm \delta\hat{\boldsymbol{n}}(\boldsymbol{x})), \tag{1.61}$$

so the boundary point is approached along the normal direction. Then the jump relation for (1.59) is

$$v_\pm(\boldsymbol{x}) = v(\boldsymbol{x}) \pm \tfrac{1}{2}\sigma(\boldsymbol{x}) \tag{1.62}$$

at any nonsingular point $\boldsymbol{x}$ of the boundary. (The factor $\frac{1}{2}$ is modified at corners.) The magnitude of the jump is $|\sigma|$.

The jump relation can be used to evaluate the limit in (1.60). The result is

$$\int_\Gamma \frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{x})}\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) = \frac{1}{ik\eta}\frac{\partial u_{\rm inc}}{\partial n}(\boldsymbol{x}) + \frac{1}{2}\sigma(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x}\in\Gamma, \tag{1.63}$$

and this is the MFIE. Note that its construction actually requires more information than given in Problem 1.1, because the forcing term cannot be determined from the Dirichlet data f alone.

Note also that the integral operator in the MFIE is similar to, but not the same as, the double-layer operator. Here the kernel is the normal derivative of $\Phi(\boldsymbol{x},\boldsymbol{y})$ at the evaluation point $\boldsymbol{x}$ instead of at the source point $\boldsymbol{y}$. The elementary solution (1.59) is the directional derivative of the field produced by a monopole layer.

Despite its apparent added complexity, the MFIE is superior to the EFIE in some important respects. The advantages are conferred by a broad class of operators of the form $\lambda\mathcal{I}-\mathcal{A}$, where $\mathcal{I}$ is the identity and $\mathcal{A}$ is *compact.* A thorough treatment of compact operators is beyond the scope of this discussion, but I will make a brief characterization. Much of what I have learned about functional analysis has been gleaned from books by Naylor and Sell [112] and Stakgold [129], and there are many other fine references.

Let the linear operator $\mathcal{A}$ act on a normed linear space $\mathbb{V}$ of infinite dimension. If $\mathcal{A}$ is compact, then its range $\mathcal{A}(\mathbb{V})\subset\mathbb{V}$ can be approximated to arbitrary accuracy by a finite-dimensional subspace of $\mathbb{V}$. Thus $\mathcal{A}$ is in a sense a smoothing transformation: Its range has essentially fewer degrees of freedom than its domain. The smoothing property is easy to grasp when $\mathcal{A}$ is an integral operator with a smooth kernel.

If the compact operator $\mathcal{A}$ is also one-to-one, then it has a left inverse $\mathcal{A}^{-1}:\mathcal{A}(\mathbb{V})\to\mathbb{V}$. However, unless $\mathcal{A}(\mathbb{V})$ is exactly finite dimensional, the inverse is necessarily unbounded. The first-kind equation $\mathcal{A}\varphi=f$ is then *ill posed.* If $f\in$

$\mathcal{A}(\mathbb{V})$, then a unique solution $\varphi \in \mathbb{V}$ exists, but arbitrarily small perturbations in the data f can produce arbitrarily large fluctuations in the solution φ. Clearly that portends trouble for any solution computed with machine arithmetic.

The singularity in the EFIE kernel is not strong enough to keep the integral operator from being compact on $\mathbb{V} = L^2(\Gamma)$. If the boundary is not everywhere sufficiently smooth, then compactness is lost, but we proceed under the assumption that Γ is smooth. Let

$$\big(\mathcal{S}\sigma\big)(\boldsymbol{x}) := \int_\Gamma \Phi(\boldsymbol{x},\boldsymbol{y})\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) \quad \text{where} \quad \boldsymbol{x} \in \Gamma \tag{1.64}$$

be the single-layer integral operator, so that the EFIE is

$$\mathcal{S}\sigma = (ik\eta)^{-1} f, \tag{1.65}$$

where $f(\boldsymbol{x}) = u_{\text{inc}}(\boldsymbol{x})$ is the Dirichlet data of the incident field. The operator $\mathcal{S}$ is compact, and the condition number $\|\mathcal{S}\|\|\mathcal{S}^{-1}\|$ of the integral equation is infinite.

A discretization of the integral operator may be exact for only an N-dimensional subspace of $L^2(\Gamma)$. Unless we happen to generate a singular $N \times N$ coefficient matrix, the condition number of the discrete system is finite. But as $N \to \infty$, that finite condition number grows without bound.

To decrease the discretization error, we would like to increase N. Unfortunately, that also increases the sensitivity of the computed solution to data uncertainties and rounding errors. If the condition number increases rapidly as $N \to \infty$, the latter numerical errors will quickly dominate the discretization error.

That sensitivity is not exhibited by computed solutions of the MFIE. Let

$$\big(\mathcal{S}'\sigma\big)(\boldsymbol{x}) := \int_\Gamma \frac{\partial\Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{x})}\sigma(\boldsymbol{y})\,d\Gamma(\boldsymbol{y}) \quad \text{where} \quad \boldsymbol{x} \in \Gamma \tag{1.66}$$

be the normal derivative of the single-layer integral operator, so that the MFIE is

$$\big(\tfrac{1}{2}\mathcal{I} - \mathcal{S}'\big)\sigma = -(ik\eta)^{-1} g, \tag{1.67}$$

where $g(\boldsymbol{x}) = \hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \nabla u_{\text{inc}}(\boldsymbol{x})$ is the Neumann data of the incident field. The operator $\mathcal{S}'$ is compact, but the operator $\frac{1}{2}\mathcal{I} - \mathcal{S}'$ is not. The Riesz–Fredholm theory [98] [109] shows that if $\frac{1}{2}$ is not an eigenvalue of $\mathcal{S}'$, then $\frac{1}{2}\mathcal{I} - \mathcal{S}'$ has a bounded inverse. The condition number $\|\frac{1}{2}\mathcal{I} - \mathcal{S}'\| \|(\frac{1}{2}\mathcal{I} - \mathcal{S}')^{-1}\|$ is bounded as well.

That property ought to be preserved by the discretization. As $N \to \infty$, the condition number of the finite algebraic system will *not* grow without bound. A bounded condition number has two important consequences:

- Discretization error can be decreased without fear of instability.
- Iterative solvers such as CGN [62] are more efficient.

Those are the virtues of the MFIE. It does, however, suffer from two limitations:

- It cannot be applied to open curves Γ.
- $\mathcal{S}'$ can have an eigenvalue of $\frac{1}{2}$, and in that event $\frac{1}{2}\mathcal{I} - \mathcal{S}'$ is not invertible.

The latter defect can be fixed by a number of established techniques. The former is more serious. Only recently [85] [86] has an integral equation of the second kind been constructed for the Dirichlet BVP exterior to an open curve.

When Γ is an open curve, there is no interior G_- where the magnetic field vanishes, and the boundary condition (1.17) for the magnetic field is replaced by the jump condition

$$\hat{\boldsymbol{n}} \times (\boldsymbol{H}_+ - \boldsymbol{H}_-) = \sigma \hat{\boldsymbol{z}}, \tag{1.68}$$

where $\boldsymbol{H}_\pm$ are the one-sided limits $\lim_{\delta \to 0+} \boldsymbol{H}\big(\boldsymbol{x} \pm \delta \hat{\boldsymbol{n}}(\boldsymbol{x})\big)$. Upon substitution of

the integral representation of the magnetic field, (1.68) simplifies to the tautology $\sigma = \sigma$. The MFIE construction fails to generate a useful equation.

In general, an elementary solution with a discontinuity at Γ is needed to reformulate Problem 1.1 as an integral equation of the second kind. For the elementary solutions (1.53) and (1.59), the discontinuity at boundary point $\boldsymbol{x}$ is a nonzero jump proportional to the unknown source density at $\boldsymbol{x}$. But if G_- is empty, then the Dirichlet condition requires that the field values be the same on either side of the curve Γ, so a nonzero jump cannot be tolerated. And if the jump is zero at $\boldsymbol{x} \in \Gamma$, then the source density of the elementary solution must also be zero there.

Other types of BVPs exterior to open curves are more easily treated. If the Dirichlet condition is replaced with a jump condition, then the solution can be expressed as a boundary integral in which the surface densities are known. No integral equation needs to be solved. Following Kirchhoff, jump boundary conditions are often used to model perfect absorbers [10, §II.4].

The limitation of the MFIE for closed curves is that at a countable set of wavenumbers $\{k_i\}$ the integral operator has a nontrivial nullspace. Those *irregular* wavenumbers depend on the shape of the boundary. The temporal frequencies $\{ck_i/2\pi\}$ are the resonant frequencies of the cavity G_- with a boundary Γ that is not a perfect metal, but rather a perfect *magnetic* conductor [80, §1.14]. The tangential magnetic field vanishes at the surface of such a material.

Combined Field Integral Equation

The combined field integral equation (CFIE) [105] [114] is a scattering integral equation for closed curves Γ that does not suffer from the MFIE's interior resonance problem. The CFIE is simply a linear combination of the EFIE (1.65) and the

MFIE (1.67). In operator notation, it is

$$\left(\tfrac{1}{2}\mathcal{I} - \mathcal{S}' + i\lambda\mathcal{S}\right)\sigma = -(ik\eta)^{-1}(g - i\lambda f). \tag{1.69}$$

If $\operatorname{Re}\lambda \neq 0$, then the CFIE operator is guaranteed to have no irregular wavenumbers. It is bounded and one-to-one for any $k > 0$, and it has a bounded inverse.

Combined Source Integral Equation

A relative of the CFIE is the combined source integral equation (CSIE) [106] [114], another equation that cleans up the MFIE for closed curves. Unlike the previous three integral equations, the CSIE represents the scattered field with a nonphysical source distribution. The nonphysical sources generate a scattered field identical to the one produced by the actual currents induced on the metal surface.

Assume that two continuous source distributions lie on the boundary Γ. Let one be a layer of monopoles, and the other a layer of normally directed dipoles. Moreover, we assume that the monopole density is everywhere proportional to the dipole density. If $\chi = -(ik\eta)^{-1}\varphi$ is the unknown dipole density, and the proportionality constant coupling the sources is $i\lambda$, then the z-component of the scattered electric field is

$$u(\boldsymbol{x}) = \int_\Gamma \left(\frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{y})} + i\lambda\Phi(\boldsymbol{x},\boldsymbol{y}) \right) \varphi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}). \tag{1.70}$$

The function φ must be selected so that u satisfies the Dirichlet boundary condition $u(\boldsymbol{x}) = -f(\boldsymbol{x})$ for $\boldsymbol{x} \in \Gamma$.

The elementary solution of the single layer is continuous on $\boldsymbol{x} \in \mathbb{E}^2$. There is no jump at the boundary, so simple replacement of $\boldsymbol{x} \in G_+$ with $\boldsymbol{x} \in \Gamma$ is acceptable, just like in the development of the EFIE.

On the other hand, the elementary solution of the double layer is discontinuous

at the boundary. If

$$v(\boldsymbol{x}) = \int_\Gamma \frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{y})} \varphi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}) \tag{1.71}$$

is the elementary solution, then the jump relation is

$$v_\pm(\boldsymbol{x}) = v(\boldsymbol{x}) \mp \tfrac{1}{2}\varphi(\boldsymbol{x}) \tag{1.72}$$

at any nonsingular point $\boldsymbol{x}$ of the boundary. This differs from (1.62) only in the sign of the jump.

Using the jump relation, in the limit as the point $\boldsymbol{x} \in G_+$ approaches a boundary point, the field is

$$u_+(\boldsymbol{x}) = -\tfrac{1}{2}\varphi(\boldsymbol{x}) + \int_\Gamma \left(\frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{y})} + i\lambda \Phi(\boldsymbol{x},\boldsymbol{y}) \right) \varphi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}), \quad \boldsymbol{x} \in \Gamma. \tag{1.73}$$

This must equal $-f(\boldsymbol{x})$, and so the CSIE is

$$\left(\tfrac{1}{2}\mathcal{I} - \mathcal{D} - i\lambda\mathcal{S}\right)\varphi = f \tag{1.74}$$

in operator form. The double-layer integral operator is

$$\big(\mathcal{D}\varphi\big)(\boldsymbol{x}) := \int_\Gamma \frac{\partial \Phi(\boldsymbol{x},\boldsymbol{y})}{\partial n(\boldsymbol{y})} \varphi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}) \quad \text{where} \quad \boldsymbol{x} \in \Gamma, \tag{1.75}$$

which is the complex conjugate of the adjoint of $\mathcal{S}'$: $\overline{\mathcal{D}}^* = \mathcal{S}'$.

If $\frac{1}{2}$ is an eigenvalue of $\mathcal{S}'$, then $\frac{1}{2}$ is also an eigenvalue of $\mathcal{D}$, so a double layer alone cannot avoid interior resonances. If $\operatorname{Re}\lambda \neq 0$, then no resonances bother the CSIE. Some work [95] has been done on choosing λ to minimize the condition number. I use $\lambda = k + 1$, which works well at both high and low frequencies.

Following Colton and Kress [31], I apply the CSIE to closed curves Γ. Its right-hand side is simpler than the right-hand side of the CFIE. If, however, the physical currents σ are wanted, then it is much easier to find them by solving the CFIE,

rather than finding them from the solution φ of the CSIE.

1.5 Discretization of Integral Equations

Starting from an analytic problem description, such as a PDE or an integral equation, the unified approach of numerical analysis is to systematically replace integrals and derivatives with sums and differences. An elliptic PDE or a Fredholm integral equation is transformed by this discretization into a system of N algebraic equations. The discretization scheme should be *convergent,* so that as $N \to \infty$ the discretization error—the difference between the solutions of the discrete and continuous problems—vanishes.

The discrete system is a better candidate for a solution algorithm that uses simple arithmetic in $\mathbb{R}$. If it is solved on a computer, inexact arithmetic generally prevents an exact solution. The algorithm must be *stable,* so that rounding errors do not accumulate catastrophically. Furthermore, all intermediate quantities must be neither too large nor too small to be represented in machine arithmetic.

In the field of computational electromagnetics, the influential work of Harrington [81] has established the method of moments as the discretization scheme of choice for linear integral equations. Briefly, the approximate solution is drawn from a selected linear manifold of dimension N. A member of this space is singled out by requiring the residual to be orthogonal to a second linear manifold of dimension N. In other fields of study, this solution framework is known as the Petrov–Galerkin method. The great flexibility afforded by the choice of the two linear spaces is the method's hallmark. But that flexibility is also an invitation to inefficiency. Oftentimes the work expended in preparing the algebraic system dominates the computa-

tion. In my work, I have consistently used an efficient variant of *collocation*, which is a subclass of moment methods.

Assume that we have a Fredholm equation of the second kind—perhaps the CSIE—which has the general form

$$\varphi(s) = f(s) + \int_a^b K(s,t)\varphi(t)\,dt, \quad s \in [a,b], \tag{1.76}$$

where the kernel $K(s,t)$ and the driving function $f(s)$ are known. Like the general Petrov–Galerkin method, collocation begins by choosing an N-dimensional linear space to which the computed solution will belong. If a basis for that space is $\{\varphi_n\}$, then the approximate solution is uniquely represented as a linear combination

$$\widehat{\varphi}(s) = \sum_{n=1}^{N} x_n \varphi_n(s). \tag{1.77}$$

To determine the coefficients $\{x_n\}$, substitute (1.77) into (1.76) to give

$$\sum_{n=1}^{N} x_n \varphi_n(s) \approx f(s) + \sum_{n=1}^{N} x_n \int_a^b K(s,t)\varphi_n(t)\,dt, \quad s \in [a,b]. \tag{1.78}$$

The $=$ sign has been replaced with a $\approx$ sign because equality cannot be expected to hold for all $s \in [a,b]$. Under the approximation (1.77), we anticipate a nonzero residual $f(s)+\int_a^b K(s,t)\widehat{\varphi}(t)\,dt-\widehat{\varphi}(s)$. The *residual* measures the pointwise violation of equality in the integral equation. In the method of collocation, the residual is forced to vanish at a set of N points $\{t_m\}$ selected from $[a,b]$. This prescription generates the algebraic system

$$\sum_{n=1}^{N} x_n \varphi_n(t_m) = f(t_m) + \sum_{n=1}^{N} x_n \int_a^b K(t_m,t)\varphi_n(t)\,dt, \quad 1 \le m \le N, \tag{1.79}$$

which can be written in the vector form

$$M\boldsymbol{x} = \boldsymbol{f} + Q\boldsymbol{x}. \tag{1.80}$$

The matrices M and Q have elements

$$M_{mn} = \varphi_n(t_m), \tag{1.81a}$$

$$Q_{mn} = \int_a^b K(t_m, t)\varphi_n(t)\, dt. \tag{1.81b}$$

If $M - Q$ is nonsingular, the system (1.80) has a unique solution.

Other ways to generate a linear system, starting from the basis expansion (1.77), include minimization of the residual 2-norm (the method of least squares) and the Galerkin testing procedure that is the backbone of most finite element methods. These discretization schemes produce a system of the form (1.80), but the matrix elements are more complicated.

A particular constraint on the choice of basis will impart greater meaning to the term "collocation." If the basis functions satisfy $\varphi_n(t_m) = \delta_{mn}$, then the sum on the left-hand side of (1.79) collapses, and M reduces to the identity matrix. At the points $s \in \{t_m\}$, the sum in (1.77) collapses as well, giving

$$\widehat{\varphi}(t_m) = x_m. \tag{1.82}$$

The coefficients $\{x_m\}$ are samples of the approximate solution $\widehat{\varphi}(s)$ at the points $\{t_m\}$, the same points used to establish the linear system (1.79). After solving (1.80) for $\boldsymbol{x}$, the formula (1.77) interpolates those samples to produce an approximate solution for $s \in [a, b]$.

Collocation is still not particularly attractive if we must compute N^2 integrals to fill the matrix Q. Each matrix element might be computed with a quadrature rule—well-known examples are the trapezoidal rule and Simpson's rule—which ap-

proximates the integral of an arbitrary smooth function g as

$$\int_a^b g(t)\,dt \approx \sum_{p=1}^{P} w_p g(t_p), \tag{1.83}$$

where $\{w_p\}$ are the quadrature weights and $\{t_p\}$ are the quadrature nodes. If a P-point quadrature rule is applied to the integral in (1.81b), filling the matrix Q requires $O(N^2P)$ flops.

To slash the time required for the matrix fill, I always use a quadrature rule with nodes that coincide with the collocation points $\{t_m\}$. Then, since $\varphi_n(t_p) = \delta_{np}$, the quadrature sum collapses:

$$\begin{aligned} Q_{mn} &\approx \sum_{p=1}^{N} w_p K(t_m, t_p)\delta_{np} \\ &= w_n K(t_m, t_n). \end{aligned} \tag{1.84}$$

If $K := \big[K(t_m, t_n)\big]$ is a matrix of kernel samples and $\boldsymbol{w} := \big[w_n\big]$ is the weight vector, then $Q = K\,\mathrm{diag}(\boldsymbol{w})$. Given the weights, each matrix element requires only a single kernel function evaluation and a single multiplication.

The efficient version of collocation described here produces the same algebraic system as Nyström's method. That method dispenses with the expansion (1.77), but instead applies a quadrature rule directly to the integral equation (1.76). A square system of equations is constructed by forcing the residual to vanish at the quadrature nodes. The main distinction between the collocation and Nyström methods is that Nyström's approximate solution is not the interpolation (1.77) of the computed vector $\boldsymbol{x}$ but rather the following interpolation:

$$\widehat{\varphi}(s) = f(s) + \sum_{n=1}^{N} w_n K(s, t_n)\widehat{\varphi}(t_n), \quad s \in [a, b]. \tag{1.85}$$

(Note that no such interpolation exists for equations of the first kind.) This simply

revisits the approximation of the integral equation before the residual constraint is applied to find $\{\widehat{\varphi}(t_n)\}$. In view of (1.85), the Nyström method is a collocation method in which the expansion basis is automatically selected by the choice of quadrature rule.

This discussion has so far assumed that the integrand in (1.76) is smooth. In scattering integral equations, however, $K(s,t)$ or one of its derivatives will be unbounded on the diagonal $s = t$. (If the boundary Γ is not smooth, singularities also occur on lines parallel to the s- and t-axes.) Then the integral in (1.81b) must be treated more carefully to achieve high performance. In (1.84) the same quadrature rule has been used for all rows of Q. This cannot be maintained for the scattering integral equation, because the diagonal kernel singularity has a different location in each row. A singular quadrature rule has weights that of course depend on the location of the singularity. Each row of Q must use the same nodes but a different set of weights, and (1.84) is changed to

$$\begin{aligned} Q_{mn} &\approx \sum_{p=1}^{N} w_p(t_m) K(t_m, t_p) \delta_{np} \\ &= w_n(t_m) K(t_m, t_n). \end{aligned} \tag{1.86}$$

Using the componentwise Schur–Hadamard matrix product $\odot$, this is compactly expressed as $Q = W \odot K$, where the matrix $W := \big[w_n(t_m)\big]$ stores the weights of N singular quadrature rules.

I use collocation instead of a Nyström method because the Nyström interpolation formula becomes a burden when the kernel is singular. The only modification to (1.85) is to replace w_n with $w_n(s)$, showing the dependence of the weights on the singularity location. But that slight change dictates that a distinct set of weights be used for each evaluation point s, and those weights are usually too expensive to

generate on the fly.

1.6 Spectral Product Rule for the EFIE

Colton and Kress [31] apply a spectral quadrature rule to the CSIE in order to efficiently solve Problem 1.1 on the exterior of analytic closed curves that are not optically large. A *spectral* quadrature rule has a discretization error that vanishes exponentially fast as the number of quadrature points increases. A positive constant $\zeta < 1$ may be found such that $\epsilon = O(\zeta^N)$ as $N \to \infty$.

In this section, I describe the construction of this singular quadrature rule for the EFIE. The CSIE includes a double-layer kernel in addition to the single-layer kernel of the EFIE, but the double-layer kernel is treated the same way.

Once again, the EFIE is

$$\int_{\Gamma} \Phi(\boldsymbol{x}, \boldsymbol{y}) \sigma(\boldsymbol{y}) \, d\Gamma(\boldsymbol{y}) = (ik\eta)^{-1} u_{\text{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{1.87}$$

where the function σ is the unknown current density on the boundary. Let the boundary curve Γ be parametrized by the function $\boldsymbol{\gamma} : [0, 1] \to \Gamma$. In this section, the curve is assumed to be closed, so $\boldsymbol{\gamma}$ may be extended to a function on $\mathbb{R}$ with unit period. At the point $\boldsymbol{y} = \boldsymbol{\gamma}(t)$ on the curve, the differential element of arc length is $d\Gamma(\boldsymbol{y}) = \|\boldsymbol{\gamma}'(t)\| dt$. After substituting the parametrization into (1.87), the line integral becomes a standard 1-D integral. The equation assumes the standard Fredholm form

$$\int_0^1 K(s, t) \varphi(t) \, dt = f(s), \quad s \in [0, 1], \tag{1.88}$$

after defining the new functions

$$\varphi(t) := \sigma\big(\boldsymbol{\gamma}(t)\big), \tag{1.89a}$$

$$f(t) := (ik\eta)^{-1} u_{\text{inc}}\big(\boldsymbol{\gamma}(t)\big), \tag{1.89b}$$

$$\begin{aligned} K(s,t) &:= \Phi\big(\boldsymbol{\gamma}(s), \boldsymbol{\gamma}(t)\big) \|\boldsymbol{\gamma}'(t)\| \\ &= -\tfrac{i}{4} H_0\big(k\|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|\big) \|\boldsymbol{\gamma}'(t)\|. \end{aligned} \tag{1.89c}$$

The kernel $K(s,t)$ is unbounded at $s = t$.

Were we to construct a singular quadrature rule of the type

$$\int_0^1 K(s,t)\varphi(t)\,dt \approx \sum_{n=1}^{N} w_n(s) K(s,t_n)\varphi(t_n), \tag{1.90}$$

then collocation would produce a matrix with main diagonal entries $w_n(t_n)K(t_n,t_n)$. But since $K(t_n,t_n) = \infty$, this fails. One alternative is to design a quadrature rule that does not include the node at the singularity. For such a rule, the main diagonal entries would be zero, and collocation would give a system

$$(W \odot K)\hat{\boldsymbol{\varphi}} = \boldsymbol{f}, \tag{1.91}$$

where the matrix K samples the kernel at all quadrature node pairs,

$$K_{mn} := \begin{cases} K(t_m,t_n) & \text{if } m \neq n, \\ 0 & \text{if } m = n. \end{cases} \tag{1.92}$$

Here, however, we follow Colton and Kress [31] in a different approach.

A product rule is also capable of eliminating the infinite diagonal. A *product* rule is designed to integrate a class of functions g that can be factored as $g(t) = s(t)h(t)$, where s is a known function and h is an arbitrary smooth function. In particular, s

may be nonsmooth. The product rule has the form

$$\int_a^b s(t)h(t)\,dt \approx \sum_{n=1}^{N} w_n h(t_n). \tag{1.93}$$

Note that only the factor h is sampled at the quadrature nodes. The singular factor s influences the weights $\{w_n\}$, which are usually determined by making the rule (1.93) exact for an N-dimensional linear space of smooth functions h.

In the application of this idea to the EFIE, we do not, however, set out to construct a product rule of the type

$$\int_0^1 K(s,t)\varphi(t)\,dt \approx \sum_{n=1}^{N} w_n(s)\varphi(t_n) \tag{1.94}$$

because the kernel depends on the boundary Γ. If every conceivable boundary required its own product rule, then the quadrature rule construction would be the dominant cost of the numerical solution to Problem 1.1. Moreover, (1.89c) shows that the kernel is a complicated function, and our product rule only needs to treat the singularity. It does not, for instance, need to treat the oscillatory behavior of the Hankel function.

The singularity of the single-layer kernel is logarithmic, a fact that can be exposed by writing the Hankel function of order 0 as

$$H_0(z) = (\log z)A_0(z^2) + B_0(z^2), \tag{1.95}$$

where A_0 and B_0 are entire functions. For an analytic boundary, the parametrization $\boldsymbol{\gamma}$ is an analytic function, and $\|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|^2$ is also analytic. Assuming the parametrization is nonsingular, the metric coefficient $\|\boldsymbol{\gamma}'(t)\|$ is analytic as well, because $\boldsymbol{\gamma}'(t)$ is nowhere zero. Consequently, we only need a singular quadrature rule

for the integral

$$(\mathcal{L}h)(s) := \int_0^1 \log(k\|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|)h(t)\,dt, \tag{1.96}$$

for an arbitrary analytic function h with unit period.

In (1.96) the singularity still shows, through the presence of the function $\boldsymbol{\gamma}$, a dependence on the boundary. Let us proceed nevertheless to develop the quadrature rule for the simplest case, in which Γ is a circle of radius α. Let the origin of coordinates lie at the center of the circle. The obvious parametrization of this curve in rectangular coordinates is $\boldsymbol{\gamma}(t) = (\alpha\cos 2\pi t,\ \alpha\sin 2\pi t)$, so the Euclidean distance between any two points on the curve is

$$\begin{aligned} R_\circ(s,t) &:= \|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\| \\ &= \alpha\sqrt{(\cos 2\pi s - \cos 2\pi t)^2 + (\sin 2\pi s - \sin 2\pi t)^2} \\ &= 2\alpha\sin\pi|s-t|. \end{aligned} \tag{1.97}$$

The product rule should take the form

$$\int_0^1 \log(2k\alpha\sin\pi|s-t|)h(t)\,dt \approx \sum_{n=1}^{N} w_n(s)h(t_n). \tag{1.98}$$

Furthermore, since we intend to solve the integral equation by collocation, we do not need the weights $\{w_n(s)\}$ for all values of s, but only at the quadrature nodes $s \in \{t_n\}$. The weights for $s = t_m$ form a row vector $\boldsymbol{w}_m^T$, and the N row vectors for each singularity location can be stacked together to give a weight matrix W with elements $W_{mn} = w_n(t_m)$.

Since the boundary is invariant under rotations, the quadrature rule should also possess circular symmetry. Let the nodes $\{t_n\}$ be equally spaced in the interval $[0,1)$, so $t_n = (n-1)/N$. Then the matrix W is *circulant:* Row $m+1$ is the right

circular shift of row m. If we take $s = 0$ to find the first row of W, (1.98) becomes

$$\sum_{n=1}^{N} W_{1n} h(t_n) \approx \int_0^1 \log(2k\alpha \sin \pi t) h(t)\, dt. \tag{1.99}$$

All other rows of W are determined from $\boldsymbol{w}_1^T$ by the circulant property. Circulant matrices will be discussed in greater detail in Section 2.3.2.

Since the Fourier coefficients of h decay at an exponential rate, the quadrature rule will have spectral accuracy if it is designed to be exact for trigonometric polynomials. Assuming that $N = 2p + 1$ is odd, we force the rule to be exact for $h \in \text{span}\{e^{-i2\pi pt}, \ldots, e^{-i2\pi t}, 1, e^{i2\pi t}, \ldots, e^{i2\pi pt}\}$. With the substitutions $h_m(t) = e^{i2\pi(m-1-p)t}$ for $1 \le m \le N$, and replacing the $\approx$ sign with an $=$ sign, (1.99) generates the linear system

$$\overline{F} D^p \boldsymbol{w}_1 = \boldsymbol{b}, \tag{1.100}$$

where, if $\omega := e^{-i2\pi/N}$, then $D := \text{diag}(1, \omega, \ldots, \omega^{N-1})$ and $F := \left[\omega^{(m-1)(n-1)}\right]$ is the discrete Fourier transform (DFT) matrix of order N.

The elements of $\boldsymbol{b}$ are

$$b_m := \int_0^1 \log(2k\alpha \sin \pi t) e^{i2\pi(m-1-p)t}\, dt, \tag{1.101}$$

and these integrals may be computed numerically with an adaptive integration routine. Because the integrand is unbounded, the computation can be time-consuming if many digits of accuracy are required. Fortunately, the integrals submit to further analysis. It is an interesting exercise in contour integration in the complex plane to

show that [63, Eq. 4.384-1 and Eq. 4.384-3]

$$\int_0^1 \log(2k\alpha \sin \pi t) e^{\pm i2\pi qt}\, dt = \begin{cases} \log k\alpha & \text{if } q = 0, \\ -\dfrac{1}{2|q|} & \text{if } q \in \mathbb{Z} \setminus 0. \end{cases} \tag{1.102}$$

Therefore $\boldsymbol{b}^T = \big[-(2p)^{-1}, \ldots, -\frac{1}{4}, -\frac{1}{2}, \log k\alpha, -\frac{1}{2}, -\frac{1}{4}, \ldots, -(2p)^{-1}\big]$, and numerical integration is unnecessary.

Solving (1.100), the weights for $s = 0$ are $\boldsymbol{w}_1 = (1/N)\overline{D}^p F\boldsymbol{b}$. That expression can be further simplified using a shift property of the DFT.

Let $Z := \big[\boldsymbol{e}_2, \ldots, \boldsymbol{e}_N, \boldsymbol{e}_1\big]$ be a circular permutation matrix constructed by reordering the columns $\{\boldsymbol{e}_n\}$ of the identity matrix. If Z premultiplies a column vector, it effects an downward circular shift of the vector elements. The relevant shift property is $\overline{D}F = FZ^T$. Applying the shift p times to $\boldsymbol{b}$,

$$\boldsymbol{w}_1 = \frac{1}{N} F\big[\log k\alpha, -\tfrac{1}{2}, -\tfrac{1}{4}, \ldots, -(2p)^{-1}, -(2p)^{-1}, \ldots, -\tfrac{1}{4}, -\tfrac{1}{2}\big]^T, \tag{1.103}$$

and this can be computed in $O(N \log N)$ flops with a fast Fourier transform (FFT).

This discussion assumed that N is odd, but the case of even N is similar. If $N = 2p$, then the entry $-(2p)^{-1}$ occurs only once on the right-hand side of (1.103).

Now a product rule for a circle is by itself uninteresting. Fortunately, with a little more work it can be applied to all smooth Jordan curves, which is a set of boundaries large enough to be interesting. In (1.95), the kernel was split apart to expose the elementary behavior of the singularity. The simplified kernel in (1.96) is $\log kR$, where $R(s,t) := \|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|$ gives the Euclidean distance between any two points on Γ. The kernel splitting can be further extended to utilize the quadrature rule for

the circle. Rewrite the kernel as

$$\begin{aligned} \log kR &= \log\Big(kR_\circ \times \frac{R}{R_\circ}\Big) \\ &= \log kR_\circ + \log \frac{R}{R_\circ}, \end{aligned} \tag{1.104}$$

where $R_\circ$, defined in (1.97), is the distance function for points on the circle.

This splitting works only because the second term in (1.104) is smooth. To avoid the logarithmic singularity in that term, there must exist constants a and b such that $0 < a \le R/R_\circ \le b$. This requirement is satisfied because

1. R and $R_\circ$ vanish at the same points, namely $s = t$ modulo the interval $[0, 1)$
2. $\lim_{s\to t} R/R_\circ$ exists and is bounded away from zero

Figure 1.4 illustrates the smoothness of this term for an analytic Jordan curve.

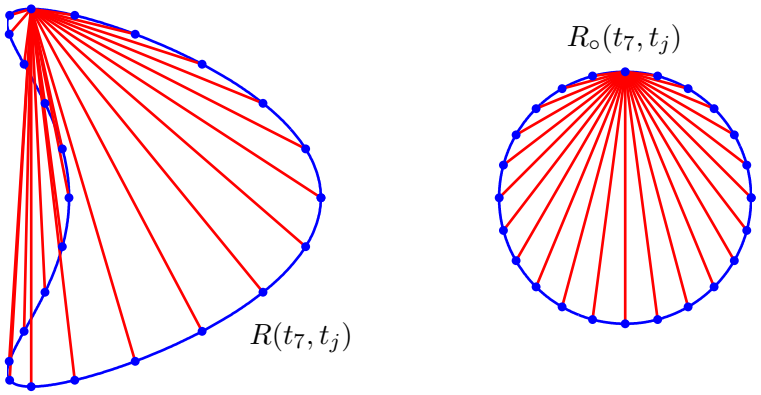
We still have the freedom of choosing the circle radius α to make $R/R_\circ$ as smooth as possible, but that choice only weakly affects the accuracy of the quadrature rule.

Note that we have achieved a remarkably clean separation of the singular behavior from the details of the boundary geometry:

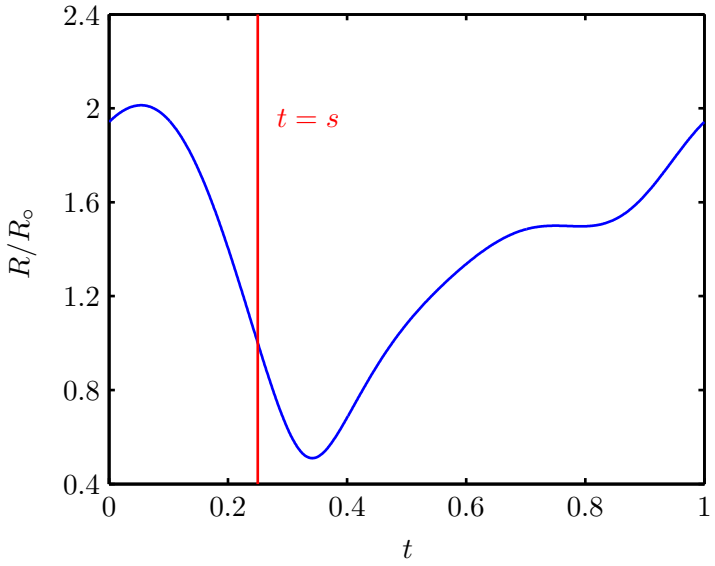
$$\begin{aligned} \text{single-layer kernel} = {} & \text{singular function independent of } \Gamma \\ & + \text{nonsingular function dependent on } \Gamma. \end{aligned}$$

It is unreasonable to assume that such a clean separation may be found for all kernels.

The quadrature rule for smooth pieces of the kernel must use the same nodes as the singular product rule. We already selected those nodes to be equally spaced points in $[0, 1)$. The $(N + 1)$-point trapezoidal rule uses these points, and it is a spectral discretization for analytic periodic integrands [35].



(a)



(b)

Figure 1.4: (a) A uniform grid of 24 points is applied to both a circle and a the smooth boundary of a nonconvex domain. The 24 chords with lengths $\|\boldsymbol{\gamma}(t_i) - \boldsymbol{\gamma}(t_j)\|$ are shown at $i = 7$ for both curves. (b) Graph of the chord ratio $R(s,t)/R_\circ(s,t)$ at $s = t_7$. The chord ratio is a smooth function at any value of s.

In summary, the discretization covered here produces the algebraic system

$$\left[W \odot A + \frac{1}{N}(L \odot A + B)\right]\hat{\boldsymbol{\varphi}} = \boldsymbol{f}. \tag{1.105}$$

The elements of $\boldsymbol{f}$ are the samples $\{f(t_m)\}$ of the driving function (1.89b), and the elements of $\hat{\boldsymbol{\varphi}}$ approximate the samples $\{\varphi(t_m)\}$ of the actual solution (1.89a). The circulant matrix W comprises the product-rule weights specified in (1.103). A, B, and L are smooth matrices, meaning that their elements are the values of smooth functions on a rectangular grid. In particular, those elements are

$$A_{mn} = -\tfrac{i}{4}A_0(k^2\|\boldsymbol{\gamma}(t_m) - \boldsymbol{\gamma}(t_n)\|^2)\|\boldsymbol{\gamma}'(t_n)\|, \tag{1.106}$$

$$B_{mn} = -\tfrac{i}{4}B_0(k^2\|\boldsymbol{\gamma}(t_m) - \boldsymbol{\gamma}(t_n)\|^2)\|\boldsymbol{\gamma}'(t_n)\|, \tag{1.107}$$

$$L_{mn} = \begin{cases} \log \dfrac{\|\boldsymbol{\gamma}(t_m) - \boldsymbol{\gamma}(t_n)\|}{2\alpha \sin \pi|s-t|} & \text{for } m \neq n, \\ \log \dfrac{\|\boldsymbol{\gamma}'(t_m)\|}{2\pi\alpha} & \text{for } m = n. \end{cases} \tag{1.108}$$

These three matrices originate from the smooth function leftovers of the kernel splitting.

1.7 Why Multipole?

Solving (1.105) requires $O(N^2)$ space and $O(N^3)$ time, and these costs prohibit solving the equations for many obstacles of interest to engineers.

For optically large boundaries, N is too large for (1.105) to be useful. Since the product quadrature rule does not address the oscillations in the EFIE integrand, a large number of quadrature nodes are necessary to sample the integrand. Aliasing errors prevent the spectral accuracy of the discretization from engaging until the oscillations are resolved.

In lieu of solving a large dense linear system, asymptotic techniques such as ray tracing are often brought to bear on large scattering problems. Unfortunately, those methods are semiconvergent only as $k \to \infty$. For a fixed obstacle and a fixed illumination frequency, the methods are nonconvergent, and are consequently held in slight regard by numerical analysts.

The fast multipole method can reduce the cost of solving the integral equation to $O(N \log^{\varsigma} N)$ space and time for a small integer ς. It is an algorithm for multiplying a vector by the matrix K of (1.92). Since K does not appear in (1.105), FMM does not help there. In Chapter 3, I modify the construction of the previous section to generate an algebraic system compatible with multipole.

Chapter 2

Three Fast Multipole Methods

In the original paper [116] on the fast multipole method, Rokhlin introduced a new solution of the boundary value problem for the 2-D Laplace equation $\Delta u = 0$. Components of Rokhlin's approach include integral equations, quadrature rules, and iterative solvers for linear systems. These mathematical objects are, however, peripheral to the new ideas contained in the multipole algorithm. To understand FMM, it is more expedient to start not with the Laplace equation but rather with a discrete particle problem.

Problem 2.1 (Particle Interaction) *Let a collection of N particles be located at points $\{\boldsymbol{x}_n\}$. Particle n generates a field $u_n(\boldsymbol{x})$, and individual particle fields are added together to give the net field. At the location of each particle m, evaluate the total field generated by all other particles $n \neq m$,*

$$u(\boldsymbol{x}_m) = \sum_{\substack{1 \le n \le N \\ n \neq m}} u_n(\boldsymbol{x}_m) \quad \textit{for} \quad 1 \le m \le N. \tag{2.1}$$

It is trivial to solve this problem in $O(N^2)$ operations by evaluating all partial interactions $\{u_n(\boldsymbol{x}_m)\}$, but for certain fields it is possible to construct an algorithm that reduces this cost to $O(N)$.

The fast solution of Problem 2.1 was addressed by Greengard and Rokhlin [64] for the case of Newton–Coulomb potential interactions in two dimensions, and subsequently by Greengard [66] for 3-D Newton–Coulomb interactions. These fields—apart from a positive physical constant dependent on the chosen unit system—are given by

$$u_n(\boldsymbol{x}) = \mp q_n \log\|\boldsymbol{x} - \boldsymbol{x}_n\|, \quad \boldsymbol{x} \in \mathbb{E}^2, \tag{2.2a}$$

$$u_n(\boldsymbol{x}) = \pm \frac{q_n}{\|\boldsymbol{x} - \boldsymbol{x}_n\|}, \quad \boldsymbol{x} \in \mathbb{E}^3, \tag{2.2b}$$

where in Coulomb's law of electricity, the upper sign is taken and q_n stands for particle charge. In Newton's law of gravity, the lower sign is taken and q_n is the particle mass. In both cases the field u is a scalar potential, and the force on a test particle with a tiny mass/charge δ at position $\boldsymbol{x}$ is $-\delta\nabla u(\boldsymbol{x})$.

Problem 2.1 is important in many applications, several of which are reviewed by Greengard [67]. One example is the simulation of galactic dynamics [11], in which each particle might represent a star. Actual galaxies contain as many as $N = 10^8$ stars, so reducing the $O(N^2)$ complexity is critical. Another example is the study of fluid dynamics using vortex methods [120], in which each particle might represent a point vortex or a vortex blob.

The Coulomb potential is important to many problems in electrical engineering as well. Examples include the modeling of electrostatic forces in low-speed solid-state devices and the calculation of pairwise capacitances between interconnect wires in a high-speed integrated circuit [111].

The focus of this dissertation, however, lies with an oscillatory interaction that is fundamentally more challenging than those in (2.2). This interaction was reviewed in Chapter 1, and it is discussed again in the following section in connection with Problem 2.1. Rokhlin has applied the multipole framework to this interaction as well, in both two [117] and three [118] space dimensions.

A generalization of Problem 2.1 is important in some contexts. If we allow field evaluation at points other than the particle locations, then we have the following problem.

Problem 2.2 (Field Summation) *Let a collection of N particles be located at points $\{\boldsymbol{y}_n\}$. Particle n generates a field $u_n(\boldsymbol{x})$. The total field u is the sum of the individual particle fields, $u(\boldsymbol{x}) = \sum_n u_n(\boldsymbol{x})$. Evaluate the total field at the M points $\{\boldsymbol{x}_m\}$,*

$$u(\boldsymbol{x}_m) = \sum_{n=1}^{N} u_n(\boldsymbol{x}_m) \quad \textit{for} \quad 1 \leq m \leq M, \tag{2.3}$$

under the assumption that the sets $\{\boldsymbol{x}_m\}$ and $\{\boldsymbol{y}_n\}$ are disjoint.

A multipole treatment of a PDE such as the Laplace equation represents the solution as a boundary integral. Problem 2.2 then finds application in the evaluation of that integral at various points of interest. To conveniently graph the solution, the selected points $\{\boldsymbol{x}_m\}$ may form a rectangular grid.

2.1 The 2-D Helmholtz Interaction

Like the Coulomb interaction, the Helmholtz interaction assumes the form

$$u_m = \sum_{\substack{1 \leq n \leq N \\ n \neq m}} q_n \Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) \quad \text{for} \quad 1 \leq m \leq N, \tag{2.4}$$

which gives the fields $\{u_m\}$ produced by N point sources with amplitudes $\{q_n\}$ at locations $\{\boldsymbol{x}_n\}$. This interaction is linear in the amplitude vector $\boldsymbol{q} := \big[q_n\big]$, and it can be rewritten as a matrix-vector product,

$$\boldsymbol{u} = \Phi\boldsymbol{q}, \tag{2.5}$$

in which $\boldsymbol{u} := \big[u_m\big]$ is the vector of field values. The *interaction matrix* Φ has zeros on its main diagonal, and its other elements are $\Phi(\boldsymbol{x}_m, \boldsymbol{x}_n)$.

Multipole methods provide an efficient way to both store Φ and compute the product $\Phi\boldsymbol{q}$. Rather than working directly with the matrix elements, they exploit certain properties of the underlying function $\Phi(\boldsymbol{x}, \boldsymbol{y})$ that defines the matrix elements.

For the Helmholtz interaction, the amplitudes $\{q_n\}$ are allowed to be complex numbers, and the function $\Phi(\boldsymbol{x}, \boldsymbol{y})$ is the radiating fundamental solution of the Helmholtz equation,

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) := -\tfrac{i}{4} H_0(k\|\boldsymbol{x} - \boldsymbol{y}\|), \tag{2.6}$$

where H_0 is the first-kind Hankel function of order 0.

Hankel functions are solutions of Bessel's ordinary differential equation [1, Ch. 9]. The Hankel function of the first kind is defined as $H_\nu(z) := J_\nu(z) + iY_\nu(z)$, a linear combination of Bessel's analytic solution J_ν and Neumann's singular solution Y_ν. The function (2.6) has a logarithmic singularity at $\boldsymbol{x} = \boldsymbol{y}$. Away from the singularity, it oscillates with spatial angular frequency k. The envelope of the oscillations decays slowly, like $\|\boldsymbol{x} - \boldsymbol{y}\|^{-1/2}$, as the separation $\|\boldsymbol{x} - \boldsymbol{y}\|$ increases. These properties are illustrated in Figure 2.1.

The factor of $-\frac{i}{4}$ in (2.6) only clutters the multipole formulas. In Sections 2.2 through 2.5.3, we absorb it into the charge vector $\boldsymbol{q}$.

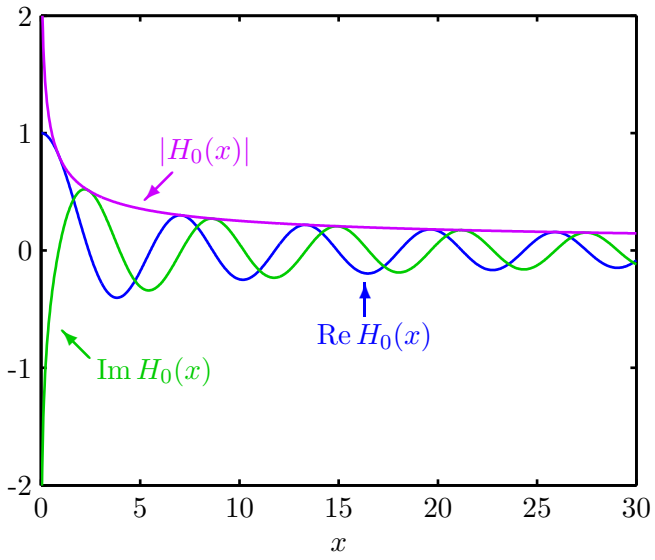


Figure 2.1: Hankel function $H_0(x)$ of the first kind and order zero.

2.2 Spectral Expansions

The function Φ is a solution of the Helmholtz equation, and this bestows on it certain properties that are exploited by Rokhlin's algorithm. This section begins a detailed account of the fast multipole method.

Section 2.2.1 covers the ingredient that gives the method its name: a *multipole* series representation of the field on a simple unbounded domain. On equal footing are series representations on simple bounded domains, the subject of Section 2.2.2. Both of these series result from a separation of variables solution of the Helmholtz equation. Typically, the "simple" domains are the exteriors and interiors of disks in $\mathbb{E}^2$ and spheres in $\mathbb{E}^3$, but other choices are possible. Recently, Greengard and Rokhlin [70] have discovered advantages to expansions on half-spaces: unbounded domains with a boundary of a line in $\mathbb{E}^2$ or a plane in $\mathbb{E}^3$.

2.2.1 On the Exterior of a Disk

Consider, as in Figure 2.2, the field generated by a collection of N oscillating point sources contained in the disk $G := \{\boldsymbol{x} : \|\boldsymbol{x}\| < \alpha\}$ centered at the origin. If the sources have amplitudes $\{q_n\}$ and locations $\{\boldsymbol{x}_n\}$, the net field is

$$v(\boldsymbol{x}) = \sum_{n=1}^{N} q_n \Phi(\boldsymbol{x}, \boldsymbol{x}_n), \quad \boldsymbol{x} \in \mathbb{E}^2. \tag{2.7}$$

The objective is to avoid this field representation, since for large N it is too costly.

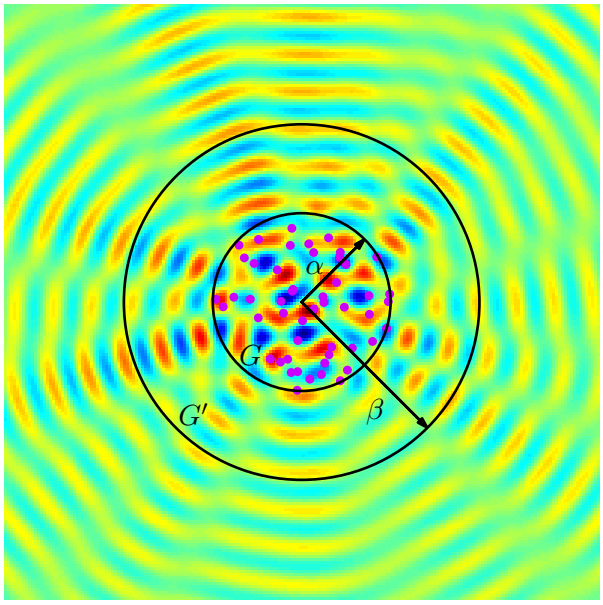


Figure 2.2: Real part of the field generated by a random collection of 50 point oscillators located inside the disk G with radius α. The series (2.10) represents the field outside the disk G' with radius β.

A different series representation can be derived from the fact that the field (2.7) is a radiating solution of the Helmholtz equation exterior to the disk $G' := \{\boldsymbol{x} : \|\boldsymbol{x}\| < \beta\}$ for any $\beta \geq \alpha$,

$$\Delta v(\boldsymbol{x}) + k^2 v(\boldsymbol{x}) = 0 \quad \text{for} \quad \|\boldsymbol{x}\| > \beta \geq \alpha, \tag{2.8a}$$

$$\frac{\partial v}{\partial r}(\boldsymbol{x}) - ikv(\boldsymbol{x}) = o(r^{-1/2}) \quad \text{where} \quad r := \|\boldsymbol{x}\| \to \infty. \tag{2.8b}$$

A separation of variables solution in polar coordinates gives a generalized Fourier series expansion of v in the basis functions

$$\varphi_m(\boldsymbol{x}) := H_m(k\|\boldsymbol{x}\|)e^{im\theta(\boldsymbol{x})}, \quad m \in \mathbb{Z}, \tag{2.9}$$

where the function $\theta : \mathbb{E}^2 \to (-\pi, \pi]$ returns the polar angle of its vector argument. The generalized Fourier series is

$$v(\boldsymbol{x}) = \sum_{m=-\infty}^{\infty} a_m \varphi_m(\boldsymbol{x}), \tag{2.10}$$

where $\{a_m\}$ are the Fourier coefficients.

This series is called a multipole expansion because an oscillating current dipole located at the origin of coordinates generates a field $a_{-1}\varphi_{-1}(\boldsymbol{x}) + a_1\varphi_1(\boldsymbol{x})$ with $|a_{-1}| = |a_1|$. In general, the two terms of order p correspond to a point source at the origin with an internal structure of 2^p oscillating poles.

The natural representation (2.7) is a finite series that taxes computer resources when its length is too large. The Fourier representation is an infinite series, so at a glance there appears to be no advantage to (2.10). Given an acceptable error level, however, the infinite series may always be truncated to a finite number of terms. Indeed, it may happen that many fewer than N terms are required, so that the sequence $\{a_m\}$ is a more efficient description of the field than the pair of sequences $(\{q_n\}, \{\boldsymbol{x}_n\})$.

The Fourier coefficients can be computed in several ways. The standard approach requires the values of the field v on the circle $\|\boldsymbol{x}\| = \gamma \geq \alpha$ concentric with the disk G. Let $\boldsymbol{x} = (r, \phi)$ in polar coordinates, so that $v(\boldsymbol{x}) \equiv v(r, \phi)$. The Dirichlet data on the circle is $f(\phi) := v(\gamma, \phi)$. Inserting the basis (2.9) into the expansion (2.10),

the function f has the Fourier series

$$f(\phi) = \sum_{m=-\infty}^{\infty} a_m H_m(k\gamma) e^{im\phi}. \tag{2.11}$$

Since the Hankel function H_m has no real zeros, the coefficients $\{a_m\}$ are easily determined from the Fourier coefficients of f,

$$a_m = \frac{1}{2\pi H_m(k\gamma)} \int_0^{2\pi} f(\phi) e^{-im\phi}\, d\phi, \quad m \in \mathbb{Z}. \tag{2.12}$$

Since f is analytic and 2π-periodic, this integral should be approximated by a trapezoidal rule, giving a sum in the form of a discrete Fourier transform. An FFT may then be used to efficiently compute $\{a_m\}$.

Another way to compute the coefficients—the one we shall employ—instead uses the natural expansion (2.7) together with an *addition theorem* for H_0. In its simplest form, the "addition" is a binary sum appearing in the function argument, and the formula is

$$H_0(z+w) = \sum_{m=-\infty}^{\infty} J_{-m}(z) H_m(w) \quad \text{for} \quad |z| < |w|. \tag{2.13}$$

More generally, the addition is a vector sum, as in

$$H_0(\|\boldsymbol{z}+\boldsymbol{w}\|) = \sum_{m=-\infty}^{\infty} J_{-m}(\|\boldsymbol{z}\|) e^{-im\theta(\boldsymbol{z})} H_m(\|\boldsymbol{w}\|) e^{im\theta(\boldsymbol{w})}, \quad \|z\| < \|w\|, \tag{2.14}$$

and this particular addition theorem appears commonly in texts on electromagnetism [80, §5.8].

Addition theorems play a pivotal role in the fast multipole method, and we shall soon need one not just for H_0, but also H_ν for any integer ν. Of critical importance is the separation of $\boldsymbol{w}$ and $\boldsymbol{z}$ in the summand of (2.14): Each term is the product of a function of $\boldsymbol{w}$ only and a function of $\boldsymbol{z}$ only.

The addition theorem (2.14) is applied to (2.7) by making the substitutions $\boldsymbol{z} \longrightarrow$

$-k\boldsymbol{x}_n$ and $\boldsymbol{w} \longrightarrow k\boldsymbol{x}$, giving

$$\begin{aligned} v(\boldsymbol{x}) &= \sum_{n=1}^{N} q_n H_0(k\|\boldsymbol{x}-\boldsymbol{x}_n\|) \\ &= \sum_{n=1}^{N} q_n \sum_{m=-\infty}^{\infty} J_{-m}(k\|\boldsymbol{x}_n\|)e^{-im\theta(-\boldsymbol{x}_n)} H_m(k\|\boldsymbol{x}\|)e^{im\theta(\boldsymbol{x})}. \end{aligned} \tag{2.15}$$

Exchanging the summation order, and also using the facts $J_{-m}(z) = (-1)^m J_m(z)$ and $\theta(-\boldsymbol{z}) = \theta(\boldsymbol{z}) \pm \pi$ (pick the sign that puts the result in the range $(-\pi, \pi]$), we arrive at the desired expression for the exterior coefficients,

$$a_m = \sum_{n=1}^{N} q_n J_m(k\|\boldsymbol{x}_n\|)e^{-im\theta(\boldsymbol{x}_n)}, \quad m \in \mathbb{Z}. \tag{2.16}$$

The coefficients $\{a_m\}$ are clearly a linear transformation of the charges[1] $\{q_n\}$, and (2.16) can be recast in vector form. Let the charge amplitudes be the elements of a vector $\boldsymbol{q}$, as in (2.5). Then computation of $\{a_m\}$ for $|m| \leq p$ requires the matrix-vector multiplication

$$\boldsymbol{a} = V\boldsymbol{q}, \tag{2.17}$$

where $\boldsymbol{a}^T := \big[a_{-p}, \ldots, a_p\big]$ and the matrix elements of $V \in \mathbb{C}^{(2p+1)\times N}$ are given by

$$v_{mn} = J_m(k\|\boldsymbol{x}_n\|)e^{-im\theta(\boldsymbol{x}_n)}. \tag{2.18}$$

Here the row index m takes on the values $-p, \ldots, p$.

For the multipole solution of the Helmholtz equation, Problem 2.1 must be solved repeatedly for different charge vectors $\boldsymbol{q}$. The charge locations $\{\boldsymbol{x}_n\}$ do not change from one iteration to the next, so it is advantageous to compute V once and store it, saving the expense of recomputing the Bessel functions $J_m(k\|\boldsymbol{x}_n\|)$. Using the symmetry $v_{-m,n} = (-1)^m\overline{v_{mn}}$, only half of V needs to be stored, and the other half

[1] I use the terms "charge" and "field" broadly to describe the vectors $\boldsymbol{q}$ and $\boldsymbol{u}$ in (2.5). If particle n is a point oscillator with complex-valued current I_n, then the "charge" $q_n = -\frac{1}{4}k\eta I_n$ has the units V/m of an electric field.

can be quickly generated by applying a pattern of sign changes.

Truncation of the Infinite Series

The expression (2.16) can be used to determine a suitable truncation of the infinite series (2.10). The terms of the series are bounded as

$$\begin{aligned} |a_m \varphi_m(\boldsymbol{x})| &= \left| \sum_{n=1}^{N} q_n J_m(k\|\boldsymbol{x}_n\|) H_m(k\|\boldsymbol{x}\|) e^{im(\theta(\boldsymbol{x})-\theta(\boldsymbol{x}_n))} \right| \\ &\leq \sum_{n=1}^{N} |q_n| |J_m(k\|\boldsymbol{x}_n\|) H_m(k\|\boldsymbol{x}\|)|. \end{aligned} \tag{2.19}$$

To further simplify this, it helps to know something about the qualitative behavior of the functions $J_m(z)$ and $H_m(z)$ as both order $m \in \mathbb{Z}$ and argument $z \in \mathbb{R}$ are varied [5].

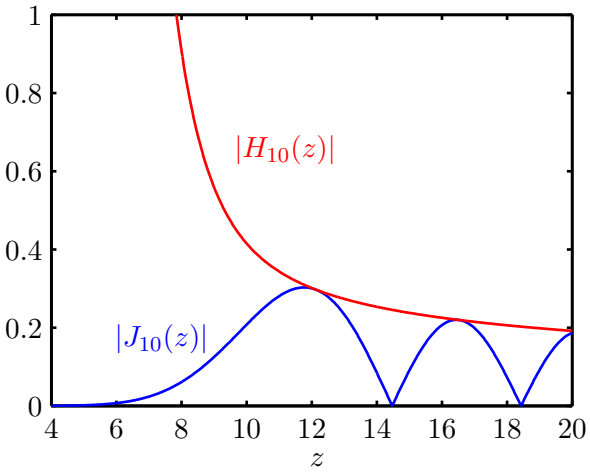
While $|H_m(z)|$ is monotone in either parameter, $|J_m(z)|$ oscillates if $|m| < |z|$. For fixed m, as in Figure 2.3(a), $|J_m(z)|$ increases and $|H_m(z)|$ decreases as $|z|$ increases from zero, until oscillations in $|J_m(z)|$ appear for $|z| > |m|$. Thus, since $\|\boldsymbol{x}_n\| < \alpha$ while $\|\boldsymbol{x}\| > \beta$, the terms in the tails of the bilateral series can be bounded more loosely as

$$|a_m \varphi_m(\boldsymbol{x})| < \|\boldsymbol{q}\|_1 |J_m(k\alpha) H_m(k\beta)| \quad \text{for} \quad |m| > k\alpha. \tag{2.20}$$

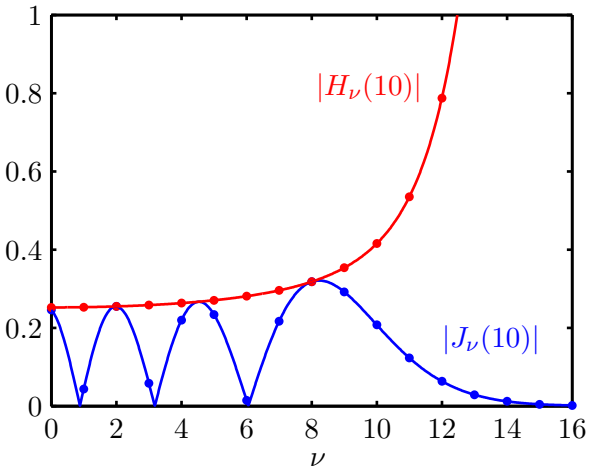
The product $|J_m(k\alpha)H_m(k\beta)|$ controls the rate of convergence. As $|m| \to \infty$, $|J_m(z)|$ decreases to zero, while $|H_m(z)|$ increases without bound [see Figure 2.3(b)]. Using the fixed-argument asymptotic expansions [1, §9.3.1], we have

$$|J_m(k\alpha)H_m(k\beta)| = O\left(\frac{1}{|m|} \left(\frac{\alpha}{\beta} \right)^{|m|} \right) \quad \text{as} \quad |m| \to \infty. \tag{2.21}$$

Thus the tails of the multipole series (2.10) decay with exponential speed as long as $\beta > \alpha$. This is the same behavior exhibited by the Fourier series spectrum of a



(a)



(b)

Figure 2.3: (a) Behavior of $|J_{10}(z)|$ and $|H_{10}(z)|$ as argument z changes. (b) Behavior of $|J_\nu(10)|$ and $|H_\nu(10)|$ as order ν changes. Dots indicate integer values of ν.

periodic analytic function, and the adjective *spectral* is used to draw this connection. For $\beta > \alpha$, the multipole series converges with spectral accuracy.

Fast convergence is critical, since we want to truncate the series at a small number of terms. For this purpose, the choice $\beta > \alpha$ is not strict enough, because it does not rule out a slow exponential decay like $(1-\delta)^{|m|}$ for some tiny $\delta > 0$. The suitable amount of separation is rather arbitrary, but I typically take $\beta = 2\alpha$, so that the series converges at the quite respectable rate of $o(2^{-|m|})$. With this choice, an evaluation point $\boldsymbol{x}$ is *well separated* from the disk G if $\|\boldsymbol{x}\| > \beta$.

Note that, while the terms rapidly decay as $|m| \to \infty$, the upper bound in (2.20) does not engage until $|m| > k\alpha$. If $k\alpha$ is large, many terms may still have to be kept.

In summary, given a permissible truncation error ϵ relative to $\|\boldsymbol{q}\|_1$, we can choose a cutoff p so that

$$v(\boldsymbol{x}) = \sum_{m=-p}^{p} a_m \varphi_m(\boldsymbol{x}) + O(\epsilon), \quad \|\boldsymbol{x}\| > \beta. \tag{2.22}$$

The number of terms required, $2p+1$, may be estimated as $O(k\alpha + \log \epsilon^{-1})$ as $\epsilon \to 0$ and $k \to \infty$.

In my own multipole code, I pick p to satisfy the requirement

$$\sum_{|m|>p} |J_m(k\alpha) H_m(k\beta)| < \epsilon. \tag{2.23}$$

A simpler criterion is to find the smallest p so that

$$|J_{p+1}(k\alpha) H_{p+1}(k\beta)| < \epsilon. \tag{2.24}$$

Since the convergence is spectral, this works as well as (2.23) if ϵ is not too large. To reduce the number of Bessel function evaluations, a production code might estimate

p from a least-squares fit to the surface $p(\epsilon, k\alpha)$ generated by the criterion (2.23). I have not been particularly impressed by the accuracy of published formulas[2] along these lines, at least not when ϵ and $k\alpha$ are allowed to range over many orders of magnitude. Furthermore, I have found that neither (2.23) nor (2.24) significantly add to the total computation time.

If the number of particles N is much greater than $k\alpha + \log \epsilon^{-1}$, then the truncated spectral expansion (2.22) is clearly more efficient than the exact finite expansion (2.7). When applying multipole to an integral equation on a boundary Γ, this behavior is observed for low frequency problems in which $k \operatorname{diam} \Gamma$ is small. It can happen too for high frequency problems if the boundary Γ exhibits strong variations on scales smaller than a wavelength, but that is not typical. Nevertheless, although its length may not be smaller than N, the spectral expansion has other properties that facilitate an efficient solution to Problem 2.1 at high frequencies. Those properties will be investigated in Section 2.3.

2.2.2 On the Interior of a Disk

In addition to a spectral expansion on a disk's exterior, FMM also utilizes a spectral expansion on the interior. Consider the same disk G as in Section 2.2.1. The field $v(\boldsymbol{x})$ was generated by charges inside G, but consider now a different field $u(\boldsymbol{x})$ produced by sources outside G, as in Figure 2.4. Let M oscillating point sources with amplitudes $\{r_m\}$ be located at the points $\{\boldsymbol{z}_m\}$ in the disk exterior. Let $\|\boldsymbol{z}_m\| > \beta$

[2]Coifman et al. [29] choose $p = 2k\alpha + d \log(\pi + 2k\alpha)$ in three dimensions, where d is a constant that depends on ϵ. This formula is apparently used as well for 2-D problems. More recently developed cutoff formulas [127] [113] have proved more accurate.

for some constant $\beta \geq \alpha$. The natural representation of the field is

$$u(\boldsymbol{x}) = \sum_{m=1}^{M} r_m \Phi(\boldsymbol{x}, \boldsymbol{z}_m), \quad \boldsymbol{x} \in \mathbb{E}^2, \tag{2.25}$$

but for large M the spectral representation may be more efficient.

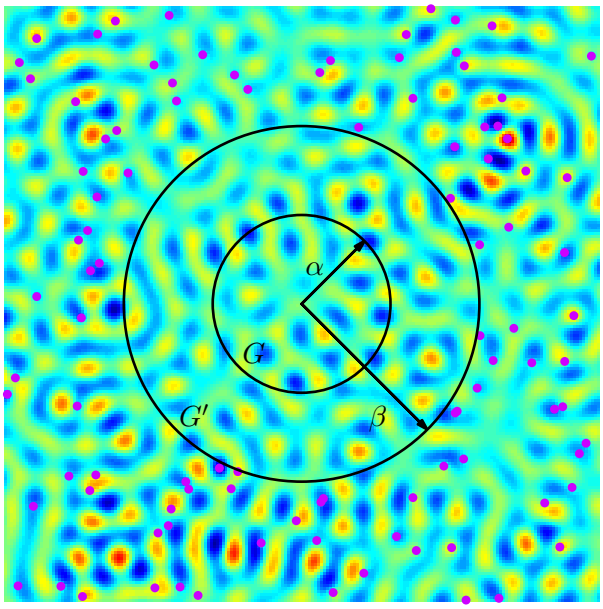


Figure 2.4: Real part of the field generated by a random collection of 100 point oscillators located outside the disk G' with radius β. The series (2.28) represents the field inside the disk G with radius α.

Inside the disk, the field is bounded and satisfies the Helmholtz equation

$$\Delta u(\boldsymbol{x}) + k^2 u(\boldsymbol{x}) = 0 \quad \text{for} \quad \|\boldsymbol{x}\| < \alpha. \tag{2.26}$$

A separation of variables solution in polar coordinates gives a generalized Fourier expansion in the basis functions

$$\psi_n(\boldsymbol{x}) := J_n(k\|\boldsymbol{x}\|)e^{in\theta(\boldsymbol{x})}. \tag{2.27}$$

The representation of the field in this basis is

$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} b_n \psi_n(\boldsymbol{x}), \tag{2.28}$$

with Fourier coefficients $\{b_n\}$.

As with the coefficients of the exterior expansion, Fourier analysis easily gives an integral representation of the interior coefficients. Since J_n, unlike H_m, does have real zeros, the formula is slightly more complicated. The integral (2.12) required only the Dirichlet data on a circle, but for the interior coefficients we generally need both Dirichlet and Neumann data on a circle $\|\boldsymbol{x}\| = \gamma \leq \beta$. In polar coordinates, $u(\boldsymbol{x}) \equiv u(r, \phi)$, so that the Dirichlet values are $f(\phi) := u(\gamma, \phi)$ and the Neumann data are the normal derivatives $g(\phi) := (\partial u/\partial r)(\gamma, \phi)$. One possible formula for the coefficients of the interior expansion is

$$b_n = \frac{1}{2\pi\big(J_n(k\gamma) + ikJ_n'(k\gamma)\big)} \int_0^{2\pi} \big(f(\phi) + ig(\phi)\big) e^{-in\phi}\, d\phi, \quad n \in \mathbb{Z}. \tag{2.29}$$

Since the real zeros of J_n and J_n' interleave, the combination $J_n + ikJ_n'$ cannot be zero if $k > 0$. Given equally spaced samples of f and g, this integral may be computed efficiently with the FFT.

The formula (2.29) is attractive because it is independent of the type of source distribution that generates the field. Making use of an appropriate addition theorem, however, we can obtain a more useful formula for our known distribution of point sources. The addition theorem (2.14) for H_0 works just as well for the interior expansion as for the exterior expansion. A slight alteration of that formula, replacing m with $-n$, gives

$$H_0(\|\boldsymbol{z} + \boldsymbol{w}\|) = \sum_{n=-\infty}^{\infty} H_{-n}(\|\boldsymbol{w}\|) e^{-in\theta(\boldsymbol{w})} J_n(\|\boldsymbol{z}\|) e^{in\theta(\boldsymbol{z})}, \quad \|z\| < \|w\|. \tag{2.30}$$

Making the substitutions $\boldsymbol{z} \longrightarrow k\boldsymbol{x}$ and $\boldsymbol{w} \longrightarrow -k\boldsymbol{z}_m$ and applying this to (2.25), we

have

$$\begin{aligned} u(\boldsymbol{x}) &= \sum_{m=1}^{M} r_m H_0(k\|\boldsymbol{x} - \boldsymbol{z}_m\|) \\ &= \sum_{m=1}^{M} r_m \sum_{n=-\infty}^{\infty} H_{-n}(k\|\boldsymbol{z}_m\|) e^{-in\theta(-\boldsymbol{z}_m)} J_n(k\|\boldsymbol{x}\|) e^{in\theta(\boldsymbol{x})}. \end{aligned} \tag{2.31}$$

Exchanging the summation order, and using the facts $H_{-n}(z) = (-1)^n H_m(z)$ and $\theta(-\boldsymbol{z}) = \theta(\boldsymbol{z}) \pm \pi$, we obtain

$$b_n = \sum_{m=1}^{M} r_m H_n(k\|\boldsymbol{z}_m\|) e^{-in\theta(\boldsymbol{z}_m)}, \quad n \in \mathbb{Z}. \tag{2.32}$$

The coefficients $\{b_n\}$ are a linear transformation of the charges $\{r_m\}$, and (2.32) can be written as a matrix-vector product. These transformations are used in the adaptive FMM of Carrier, Greengard, and Rokhlin [19], but we will have no occasion to use them. For our purposes, this development is important because it enables the determination of the effective length of the interior expansion.

An analysis of the terms $|b_n \psi_n(\boldsymbol{x})|$ of the absolute series, patterned after the analysis of the exterior expansion, shows that the same bound (2.20) applies,

$$|b_n \psi_n(\boldsymbol{x})| < \|\boldsymbol{r}\|_1 |J_n(k\alpha) H_n(k\beta)| \quad \text{for} \quad |n| > k\alpha. \tag{2.33}$$

Consequently, if we again separate $\boldsymbol{x}$ and $\{\boldsymbol{z}_m\}$ with the choice $\beta = 2\alpha$, the number of terms required for the expansion interior to the disk G is the same as the number of terms required for the expansion exterior to G', and we have a computable approximation

$$u(\boldsymbol{x}) = \sum_{n=-p}^{p} b_n \psi_n(\boldsymbol{x}) + O(\epsilon), \quad \|\boldsymbol{x}\| < \alpha. \tag{2.34}$$

The expansion length scales asymptotically as $2p + 1 = O(k\alpha + \log \epsilon^{-1})$.

Now consider the evaluation of the truncated interior expansion at a set of M

points $\{\boldsymbol{y}_m\}$ inside G. We have, within a truncation error $O(\epsilon)$,

$$u(\boldsymbol{y}_m) = \sum_{n=-p}^{p} b_n J_n(k\|\boldsymbol{y}_m\|) e^{in\theta(\boldsymbol{y}_m)}, \quad 1 \le m \le M, \tag{2.35}$$

or, in vector notation,

$$\boldsymbol{u} = U\boldsymbol{b}, \tag{2.36}$$

with $\boldsymbol{b}^T := \big[b_{-p}, \ldots, b_p\big]$. The matrix elements of $U \in \mathbb{C}^{M\times(2p+1)}$ are

$$u_{mn} = J_n(k\|\boldsymbol{y}_m\|) e^{in\theta(\boldsymbol{y}_m)}, \tag{2.37}$$

where column index n takes on the values $-p, \ldots, p$. If the evaluation points coincide with the charge locations of Section 2.2.1, then U in (2.36) and V in (2.17) are an adjoint pair: $V = U^H$.

2.3 Flat Multipole

Once I describe the connection between interior and exterior expansions, we will have all the elements in place for a rudimentary FMM implementation. That connection is provided by an addition theorem for Hankel functions H_ν of integer order,

$$H_n(\|\boldsymbol{z}+\boldsymbol{w}\|) e^{in\theta(\boldsymbol{z}+\boldsymbol{w})} = \sum_{m=-\infty}^{\infty} J_m(\|\boldsymbol{z}\|) e^{im\theta(\boldsymbol{z})} H_{n-m}(\|\boldsymbol{w}\|) e^{i(n-m)\theta(\boldsymbol{w})}, \quad \|\boldsymbol{z}\| < \|\boldsymbol{w}\|. \tag{2.38}$$

This is a specialization of Graf's addition theorem [1, §9.1.79].

2.3.1 Translation of Spectral Expansions

Consider the field, illustrated in Figure 2.5, generated by N oscillating charges in the disk G and evaluated at a collection of M points in the disk H. Since each evaluation

point is well separated from G, the exterior spectral expansion of Section 2.2.1 exhibits rapid exponential convergence. Choose an integer q so that discarding terms with indices $|n| > q$ produces an approximation $\hat{u}$ of the field u,

$$\hat{u}(\boldsymbol{x}) = \sum_{n=-q}^{q} a_n \varphi_n(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in H, \tag{2.39}$$

such that $|u - \hat{u}| = O(\epsilon)$ in disk H.

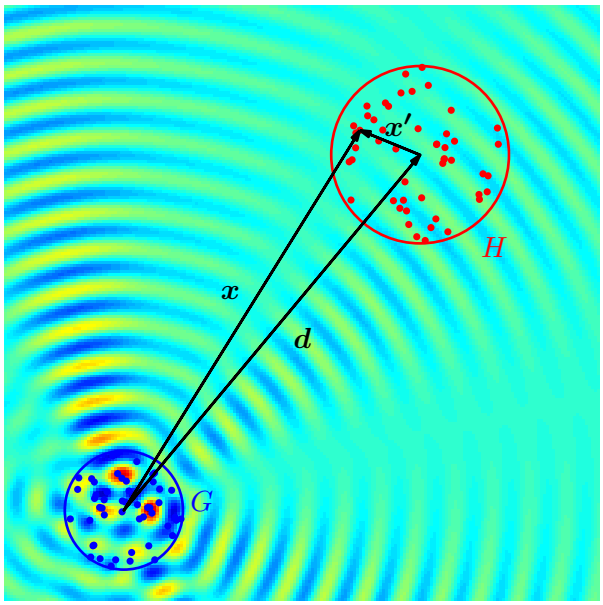


Figure 2.5: Real part of the field $u(\boldsymbol{x}) \equiv v(\boldsymbol{x'})$ generated by 50 point oscillators distributed randomly in disk G. The field is evaluated at another random collection of 50 points inside disk H.

The tail of the vector $\boldsymbol{x}$ is attached to the origin of coordinates, which lies at the center of disk G. Let the center of disk H lie at $\boldsymbol{d}$ in this coordinate system, and introduce a new coordinate system in which the tail of a position vector $\boldsymbol{x'}$ is attached to the center of disk H. The coordinate systems are related by the simple translation $\boldsymbol{x} = \boldsymbol{x'} + \boldsymbol{d}$. Let $v(\boldsymbol{x'}) := u(\boldsymbol{x'} + \boldsymbol{d})$ describe the field in the new coordinates.

Since each source point is well separated from the disk H, the interior spectral expansion of Section 2.2.2 exhibits rapid exponential convergence. Choose an integer p so that discarding terms with indices $|m| > p$ produces an approximation,

$$\hat{v}(\boldsymbol{x'}) = \sum_{m=-p}^{p} b_m \psi_m(\boldsymbol{x'}) \quad \text{for} \quad \boldsymbol{x'} \in H, \tag{2.40}$$

such that $|v - \hat{v}| = O(\epsilon)$ in disk H.

Clearly, since $\hat{u}(\boldsymbol{x})$ and $\hat{v}(\boldsymbol{x'})$ describe the same field, the coefficients of expansions (2.39) and (2.40) are related. In fact, $\{b_m\}$ can be obtained from $\{a_n\}$ through a linear transformation called a *translation operator.* Assembling the respective coefficients into the vectors $\boldsymbol{b} \in \mathbb{C}^{2p+1}$ and $\boldsymbol{a} \in \mathbb{C}^{2q+1}$, the translation operator has a matrix representation $T \in \mathbb{C}^{(2p+1)\times(2q+1)}$, such that

$$\boldsymbol{b} = T\boldsymbol{a}. \tag{2.41}$$

Proof of this fact, together with a formula for the elements of T, can be obtained from the addition theorem (2.38). Substituting that into (2.39), after making the replacements $\boldsymbol{z} \longrightarrow k\boldsymbol{x'}$ and $\boldsymbol{w} \longrightarrow k\boldsymbol{d}$, gives

$$\begin{aligned}
\hat{u}(\boldsymbol{x'} + \boldsymbol{d}) &= \sum_{n=-q}^{q} a_n H_n(k\|\boldsymbol{x'} + \boldsymbol{d}\|) e^{in\theta(\boldsymbol{x'}+\boldsymbol{d})} \\
&= \sum_{n=-q}^{q} a_n \sum_{m=-\infty}^{\infty} J_m(k\|\boldsymbol{x'}\|) e^{im\theta(\boldsymbol{x'})} H_{n-m}(k\|\boldsymbol{d}\|) e^{i(n-m)\theta(\boldsymbol{d})} \\
&= \sum_{m=-\infty}^{\infty} \left(\sum_{n=-q}^{q} a_n H_{n-m}(k\|\boldsymbol{d}\|) e^{i(n-m)\theta(\boldsymbol{d})} \right) J_m(k\|\boldsymbol{x'}\|) e^{im\theta(\boldsymbol{x'})}.
\end{aligned} \tag{2.42}$$

The finite exterior expansion generates an infinite expansion in the interior of H, with coefficients

$$b_m = \sum_{n=-q}^{q} a_n H_{n-m}(k\|\boldsymbol{d}\|) e^{i(n-m)\theta(\boldsymbol{d})}. \tag{2.43}$$

Truncating the infinite series at $m = \pm p$ introduces an error of $O(\epsilon)$. The field $\hat{v}$

computed by substituting (2.43) into (2.40) has two sources of error: truncation of $\{a_n\}$ and truncation of $\{b_m\}$. We shall forgo a detailed analysis of this error combination.

Note that each interior coefficient b_m depends on all the exterior coefficients $\{a_n\}$. Since the sequence of exterior coefficients has been truncated, the values of b_m computed in (2.43) are not the same as those given by the exact formula (2.32). Nevertheless, when the series cutoffs p and q are selected according to the considerations of Section 2.2.1, these two sets of interior coefficients produce fields $\hat{v}$ that differ by only $O(\epsilon)$.

The elements of the translation matrix T are evidently

$$\begin{aligned} t_{mn} &= H_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} \\ &= H_{m-n}(k\|\boldsymbol{d}\|)e^{-i(m-n)\theta(-\boldsymbol{d})}, \end{aligned} \tag{2.44}$$

where the row index m takes on values $-p, \ldots, p$ while the column index n takes values $-q, \ldots, q$. Note that these elements depend only on the optical displacement $k\boldsymbol{d}$ between the disk centers.

The formula (2.44) reveals the crucial structure that enables multipole to be fast for a highly oscillatory interaction, when the set of particles of Problem 2.1 is many wavelengths in diameter. Since the matrix elements are a function of the index difference $m - n$, those elements are constant on any diagonal of the matrix: T is a *Toeplitz* matrix.

That Toeplitz structure can be utilized to rapidly compute the matrix-vector product (2.41). Instead of the $O(pq)$ flops required for a general matrix T, the Toeplitz product can be performed in $O\big((p+q)\log(p+q)\big)$ flops, which is a tremendous improvement when p and q are large.

For low frequency problems, the spectral expansions are short, and it is unnecessary to utilize the Toeplitz structure of the translation matrix. A standard matrix-vector multiplication algorithm then suffices for the computation of (2.41).

2.3.2 Fast Toeplitz Matrix-Vector Products

The fast algorithm for computing $\boldsymbol{y} = T\boldsymbol{x}$ for an arbitrary vector $\boldsymbol{x}$ starts by embedding the Toeplitz matrix T into a *circulant* matrix C [34]. Circulant matrices are a subclass of square Toeplitz matrices with columns that are successive circular downshifts of the first column.

The embedding $T \mapsto C$ is not unique. One choice puts T in the top left corner of C. A simple example that embeds a 3×2 Toeplitz matrix into a 4×4 circulant matrix is

$$\begin{bmatrix} a & d \\ b & a \\ c & b \end{bmatrix} \mapsto \left[\begin{array}{cc|cc} a & d & c & b \\ b & a & d & c \\ c & b & a & d \\ \hline d & c & b & a \end{array}\right]. \tag{2.45}$$

When the embedding is done in this manner, the vector $\boldsymbol{y}$ can be recovered from the product of C with a vector $\boldsymbol{z}$ obtained by appending zeros to $\boldsymbol{x}$,

$$\begin{bmatrix} \boldsymbol{y} \\ \times \end{bmatrix} = \begin{bmatrix} T & \times \\ \times & \times \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \mathbf{0} \end{bmatrix} = C\boldsymbol{z}. \tag{2.46}$$

The $\times$'s in the matrix stand for the entries added to complete the embedding. The left-hand side entries symbolized by $\times$ are discarded after the product is computed. Other embeddings are simple variations on this theme.

Another way of expressing the product using the embedding of (2.45) is

$$\begin{bmatrix} a & d \\ b & a \\ c & b \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{cc|cc} a & d & c & b \\ b & a & d & c \\ c & b & a & d \\ \hline d & c & b & a \end{array}\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}. \tag{2.47}$$

In general, an $m \times n$ Toeplitz matrix may be uniquely represented as $T = E_m^T C E_n$, where C is a circulant matrix of order $N := m + n - 1$ and $E_j := \big[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_j\big]$ is the rectangular matrix comprising the first j columns of the order-N identity matrix. The rightmost factor E_n in the matrix product implements the zero padding, and the leftmost factor E_m^T implements the truncation.

Now we use the fact that the eigenvalue decomposition of any circulant matrix $C \in \mathbb{C}^{N \times N}$ is $C = F^{-1} \Lambda F$, where F is an N-point DFT matrix. The diagonal matrix Λ holds the eigenvalues of C, which are the Fourier transform of C's first column. If this first column is $\boldsymbol{c} = C\boldsymbol{e}_1$, then $\Lambda = \operatorname{diag}(F\boldsymbol{c})$ and the matrix-vector product is computed as

$$\begin{aligned} \boldsymbol{y} &= E_m^T C E_n \boldsymbol{x} \\ &= E_m^T F^{-1} \operatorname{diag}(F\boldsymbol{c}) F E_n \boldsymbol{x} \\ &= E_m^T F^{-1} (F\boldsymbol{c} \odot F E_n \boldsymbol{x}), \end{aligned} \tag{2.48}$$

where $\odot$ is the componentwise product operator. FFTs should be used to implement the discrete Fourier transforms. If an FFT longer than N points would be faster, then the Toeplitz matrix may be embedded into an arbitrarily larger circulant matrix.

In summary, $\boldsymbol{y}$ can be computed with three FFTs and a single componentwise vector product, and when T is large this is much faster than the standard matrix-vector product $T\boldsymbol{x}$. The fast multiply requires $O\big((m+n)\log(m+n)\big)$ flops, versus $O(mn)$ flops for the standard algorithm.

The proof of the above eigenvalue decomposition is not difficult. The well-known shift property of the FFT states that a circular shift of a vector $\boldsymbol{x}$ is equivalent to a *modulation*—a componentwise product with a trigonometric function—of the Fourier spectrum of $\boldsymbol{x}$. Using the circular permutation matrix $Z := \left[\boldsymbol{e}_2, \ldots, \boldsymbol{e}_N, \boldsymbol{e}_1\right]$, this property is expressed as $FZ\boldsymbol{x} = DF\boldsymbol{x}$, where $D = \operatorname{diag}(1, \omega, \ldots, \omega^{N-1})$ and $\omega := e^{-i2\pi/N}$. Since $\boldsymbol{x}$ is arbitrary, $FZ = DF$. The main diagonal of D is evidently the second column of F, so $D = \operatorname{diag}(F\boldsymbol{e}_2)$. Generally, the powers of D are $D^j = \operatorname{diag}(F\boldsymbol{e}_{j+1})$.

Let $C = [\boldsymbol{c}, Z\boldsymbol{c}, \ldots, Z^{N-1}\boldsymbol{c}]$ be an $N \times N$ circulant matrix. Premultiplying C by F, we have

$$\begin{aligned} FC &= \left[F\boldsymbol{c}, FZ\boldsymbol{c}, \ldots, FZ^{N-1}\boldsymbol{c}\right] \\ &= \left[F\boldsymbol{c}, DF\boldsymbol{c}, \ldots, D^{N-1}F\boldsymbol{c}\right] \\ &= \left[F\boldsymbol{c} \odot F\boldsymbol{e}_1, F\boldsymbol{c} \odot F\boldsymbol{e}_2, \ldots, F\boldsymbol{c} \odot F\boldsymbol{e}_N\right] \\ &= \left[\operatorname{diag}(F\boldsymbol{c})F\boldsymbol{e}_1, \operatorname{diag}(F\boldsymbol{c})F\boldsymbol{e}_2, \ldots, \operatorname{diag}(F\boldsymbol{c})F\boldsymbol{e}_N\right] \\ &= \operatorname{diag}(F\boldsymbol{c})F, \end{aligned} \tag{2.49}$$

so that $C = F^{-1}\operatorname{diag}(F\boldsymbol{c})F$ is the desired diagonalization. Rewriting this as $CF^{-1} = F^{-1}\operatorname{diag}(F\boldsymbol{c})$ shows that the eigenvectors of C are the columns of F^{-1}.

A standard topic in the undergraduate electrical engineering curriculum is the application of the FFT to efficiently compute discrete convolutions. This is a special case of the algorithm sketched here: In a convolution the Toeplitz matrix is also banded. The circulant matrix embedding corresponds to a replacement of the linear convolution with a circular convolution.

For example, the convolution of the length-2 sequences (a, b) and (w, z) may be

written as the matrix-vector product

$$\begin{bmatrix} a & 0 \\ b & a \\ 0 & b \end{bmatrix} \begin{bmatrix} w \\ z \end{bmatrix} = \left[\begin{array}{cc|c} a & 0 & b \\ b & a & 0 \\ 0 & b & a \end{array}\right] \left[\begin{array}{c} w \\ z \\ \hline 0 \end{array}\right]. \tag{2.50}$$

Because of the zeros in the upper-right and lower-left corners of the Toeplitz matrix on the left-hand side, the embedding is more compact than in example (2.45). The equivalent circular convolution is computed as the inverse FFT of the componentwise product of the FFTs of the zero-padded sequences $(a, b, 0)$ and $(w, z, 0)$. Naturally, the fast convolution is unnecessary for these short sequences, but the same principles apply to the general case.

2.3.3 Flat Multipole Algorithm

We are now prepared to describe an elementary multipole method for the solution of Problem 2.1 for the Helmholtz potential (2.6). In its fullest expression, multipole is a hierarchical method, with an organization that has much in common with multigrid [8]. Section 2.5 contains a description of hierarchical multipole. Here, we begin progress toward this goal under the simplifying assumption of a single-level hierarchy. I call the resulting algorithm a *flat* multipole method.

Figure 2.6 illustrates a simple example in which N point oscillators are distributed on the interval $[0, L]$ of the x-axis. The location x_n of each particle is a uniformly distributed random variable, and the locations are independent of one another.

The interaction matrix Φ is uniquely defined only after an order has been assigned to the particles. This order gives the row and column indices of each particle. Shuffling the order with the permutation matrix P changes the interaction matrix to $P\Phi P^T$. (Problem 2.1 can be slightly redefined to allow a particle to have different row

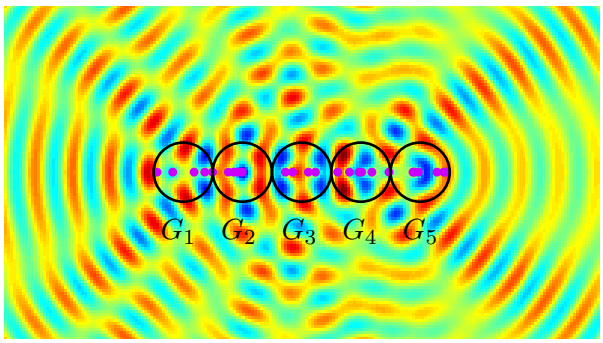


Figure 2.6: Real part of the field generated by 25 point oscillators uniformly distributed on the interval $[0, L]$ of the x-axis. The particles are grouped into 5 clusters.

and column indices, generalizing the equivalence to $P\Phi Q^T$ for distinct permutation matrices P and Q.) For our simple problem, we choose the natural order of the real numbers $\{x_n\}$: The particles are numbered from left to right.

If the particles were equally spaced, occupying the sites $x_n/L = (n-1)/(N-1)$ for $1 \leq n \leq N$, then the interaction matrix Φ would be Toeplitz, and the technique of Section 2.3.2 would give a solution in $O(N \log N)$ flops. Multipole is capable of handling more general distributions such as the one in Figure 2.6.

Partition the interval into s subintervals of equal length, each circumscribed by a circle as shown in Figure 2.6. Since the particles are uniformly distributed, the expected number of particles in each disk is N/s. Any particle that happens to lie on the boundary of two disks may be arbitrarily assigned to one or the other.

The particles inside a single disk constitute a particle cluster. Each particle belongs to one of s distinct clusters. This terminology is somewhat forced, since two particles in adjacent clusters may be arbitrarily close to one another. The characteristic of this particle classification important to a multipole algorithm is that the particles within a cluster may not be separated by large distances. An ideal

automatic clustering algorithm maximizes the number of well-separated cluster pairs, while allocating the same number of particles to each cluster.

The concept of well-separated clusters is introduced by generalizing the definition on page 64 of a point that is well separated from a disk. Given two disjoint disks G and H of radius α and β, respectively, we may call H well separated from G if every point in H is well separated from G, so that

$$\|\boldsymbol{x} - \boldsymbol{y}\| > \alpha \quad \text{for all} \quad \boldsymbol{x} \in G,\ \boldsymbol{y} \in H. \tag{2.51}$$

This definition is unsymmetric: If H is well separated from G, it is not necessarily true that G is well separated from H. We symmetrize the definition and call the pair of disks (G, H) well separated if

$$\|\boldsymbol{x} - \boldsymbol{y}\| > \max(\alpha, \beta) \quad \text{for all} \quad \boldsymbol{x} \in G,\ \boldsymbol{y} \in H. \tag{2.52}$$

The definition may be further extended to two general planar sets G and H by considering the smallest disks that enclose G and H. Thus, two point clusters are well separated if their minimum enclosing disks are well separated.

Referring again to Figure 2.6, any two disks G_i and G_j are well separated if and only if $|i - j| > 1$, so there are altogether 6 well-separated pairs. The interactions between particles in these disk pairs shall be approximated with the multipole technique described in Section 2.3.1, translating a spectral expansion exterior to G_i to a spectral expansion interior to G_j, and translating a spectral expansion exterior to G_j to a spectral expansion interior to G_i. The remaining interactions, those between particles in the same disk or in adjacent disks, are to be computed directly from the corresponding entries of the interaction matrix Φ. Multipole algorithms do not change the way in which short-range interactions are computed.

In our example, the particles in each cluster are indexed with a range of consec-

utive integers. An ordering with this property is not necessary, but it does improve vectorization of the FMM code. It also induces a block partitioning of the interaction matrix Φ that facilitates a description of the algorithm.

With $s = 5$ clusters, partition the charge vector as $\boldsymbol{q}^T = \left[\boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_5^T\right]$, where $\boldsymbol{q}_i$ are the charge amplitudes of the particles in disk G_i. If the field vector is partitioned in the same way, $\boldsymbol{u}^T = \left[\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_5^T\right]$, then the interaction matrix Φ has the 5×5 block structure

$$\Phi = \begin{bmatrix} D_{11} & D_{12} & U_1 T_{13} V_3 & U_1 T_{14} V_4 & U_1 T_{15} V_5 \\ D_{21} & D_{22} & D_{23} & U_2 T_{24} V_4 & U_2 T_{25} V_5 \\ U_3 T_{31} V_1 & D_{32} & D_{33} & D_{34} & U_3 T_{35} V_5 \\ U_4 T_{41} V_1 & U_4 T_{42} V_2 & D_{43} & D_{44} & D_{45} \\ U_5 T_{51} V_1 & U_5 T_{52} V_2 & U_5 T_{53} V_3 & D_{54} & D_{55} \end{bmatrix} + O(\epsilon). \tag{2.53}$$

A submatrix D_{ij} on the block tridiagonal gives the field at the particle locations in disk G_i due *only* to the charges in nearby disk G_j. Nominally, D_{ij} is an $(N/s) \times (N/s)$ matrix, but since we do not require the disks to contain exactly N/s particles, the off-diagonal blocks may be rectangular. The blocks $\{D_{ij}\}$ are treated by multipole as dense unstructured matrices. Using the spectral approximations of the interaction law, the remaining blocks have the structure $U_i T_{ij} V_j$ indicated.

The matrix U_i was introduced in Section 2.2.2. It maps the coefficients of a spectral expansion on the interior of disk G_i to field values at the particle positions in that disk. Each of these matrices has exactly $2p + 1$ columns, and approximately N/s rows. The matrix $V_j = U_j^H$ maps charge amplitudes of the particles in disk G_j to the coefficients of a spectral expansion on the exterior of a concentric disk G_j' with twice the radius. That operation was covered in Section 2.2.1.

The Toeplitz matrix $T_{ij} \in \mathbb{C}^{(2p+1)\times(2p+1)}$ maps the exterior coefficients of disk G_j to interior coefficients of disk G_i, the operation treated in Section 2.3.1. These interior coefficients describe the field in disk G_i due *only* to those charges in disk

G_j. In this example, the expansion lengths $2p+1$ are the same for each disk, since the disks are all the same size. In general the disks will have different sizes, the expansions will have different lengths, and the matrices $\{T_{ij}\}$ will be rectangular.

A block factorization of (2.53) is suggested by the observation that, excluding the block tridiagonal, all blocks in row i are premultiplied by U_i while all blocks in column j are postmultiplied by V_j. The factorization is

$$\Phi = \text{tridiag}(D_{ij}) + \text{diag}(U_i) \begin{bmatrix} 0 & 0 & T_{13} & T_{14} & T_{15} \\ 0 & 0 & 0 & T_{24} & T_{25} \\ T_{31} & 0 & 0 & 0 & T_{35} \\ T_{41} & T_{42} & 0 & 0 & 0 \\ T_{51} & T_{52} & T_{53} & 0 & 0 \end{bmatrix} \text{diag}(V_j), \tag{2.54}$$

and it exposes another main idea of the algorithm: The interior expansions for disk G_i that are generated by a collection $\{G_j\}$ of well-separated disks should be added together *before* applying the evaluation operator U_i. Note that the truncation error term $O(\epsilon)$ of (2.53) has been dropped from the equation, and it will be suppressed in all subsequent equations.

If the translation matrices $\{T_{ij}\}$ are large, then their Toeplitz structure must be utilized. Following Section 2.3.2, each of these matrices can be embedded into a circulant matrix with order $P \geq 4p+1$, and each circulant matrix is diagonalized by a discrete Fourier transform. The factorization becomes

$$\Phi = \text{tridiag}(D_{ij}) + \text{diag}(U_i E^T F^{-1}) \begin{bmatrix} 0 & 0 & \Lambda_{13} & \Lambda_{14} & \Lambda_{15} \\ 0 & 0 & 0 & \Lambda_{24} & \Lambda_{25} \\ \Lambda_{31} & 0 & 0 & 0 & \Lambda_{35} \\ \Lambda_{41} & \Lambda_{42} & 0 & 0 & 0 \\ \Lambda_{51} & \Lambda_{52} & \Lambda_{53} & 0 & 0 \end{bmatrix} \text{diag}(FEV_j), \tag{2.55}$$

where F is the DFT matrix of order P, and the matrices $\Lambda_{ij} = \text{diag}(\boldsymbol{\lambda}_{ij})$ are all diagonal. If the embedding is done following the example (2.45), then the rectangular matrix E comprises the leading $2p+1$ columns of the order-P identity matrix, and

the eigenvalue vector $\boldsymbol{\lambda}_{ij}$ is

$$\boldsymbol{\lambda}_{ij} = F\boldsymbol{h}_{ij}, \tag{2.56}$$

where

$$\begin{aligned}\boldsymbol{h}_{mn}^T := \big[&H_0(k\|\boldsymbol{d}_{mn}\|), H_1(k\|\boldsymbol{d}_{mn}\|)e^{-i\theta(-\boldsymbol{d}_{mn})}, \ldots, H_{2p}(k\|\boldsymbol{d}_{mn}\|)e^{-i2p\theta(-\boldsymbol{d}_{mn})},\\ &\underbrace{0, \ldots, 0,}_{P-4p-1} H_{-2p}(k\|\boldsymbol{d}_{mn}\|)e^{i2p\theta(-\boldsymbol{d}_{mn})}, \ldots, H_{-1}(k\|\boldsymbol{d}_{mn}\|)e^{i\theta(-\boldsymbol{d}_{mn})}\big].\end{aligned} \tag{2.57}$$

The vector $\boldsymbol{d}_{mn}$ in this definition is the displacement vector from the center of disk G_n to the center of disk G_m.

A step-by-step description of the algorithm for $\Phi\boldsymbol{q}$ implied by this matrix factorization is:

1. Compute exterior expansion coefficients for the charge in each disk: $\boldsymbol{a}_j = V_j\boldsymbol{q}_j$.
2. Zero-pad each coefficient vector and take its FFT: $\widetilde{\boldsymbol{a}}_j = FE\boldsymbol{a}_j$.
3. Translate exterior coefficient spectra to interior coefficient spectra for all well-separated pairs of disks: $\widetilde{\boldsymbol{b}}_{ij} = \boldsymbol{\lambda}_{ij} \odot \widetilde{\boldsymbol{a}}_j$.
4. Add interior coefficient spectra for each disk: $\widetilde{\boldsymbol{b}}_i = \sum_j \widetilde{\boldsymbol{b}}_{ij}$.
5. Truncate the inverse FFT of the interior coefficient spectrum for each disk: $\boldsymbol{b}_i = E^T F^{-1}\widetilde{\boldsymbol{b}}_i$.
6. Evaluate the interior expansion for each disk at the locations of the particles contained in the disk: $\boldsymbol{u}_i = U_i\boldsymbol{b}_i$.
7. Add the near interactions: $\boldsymbol{u} \leftarrow \boldsymbol{u} + \text{tridiag}(D_{ij})\boldsymbol{q}$.

Each step in this high-level description loops over all disks. The loops of steps 1–4 can be fused into a single loop, as can the loops of steps 5–7, giving an implementation that requires two passes over the collection of disks.

Some pseudocode suggesting this program structure is presented in Algorithm 2.1. The function calls **fft** and **ifft** implement the forward and inverse FFT, respectively. For the details of the zero pad in line 4 and the truncation in line 8, refer to Section 2.3.2. The elements of the input matrices $\{U_i\}$ are given by (2.37), and the vectors $\{\boldsymbol{\lambda}_{ij}\}$ are computed from (2.56). The critical parameters are p, the cutoff for the spectral expansions, and the number of clusters s. The latter affects the algorithm efficiency, while the former affects accuracy and numerical stability.

Algorithm 2.1: Flat Multipole

```
    Input: Charge amplitudes {q_i},
      Representation ({D_ij}, {U_i}, {λ_ij}) of Φ,
      Set of disks {G_i} containing particle clusters
    Output: Field values {u_i}

1   b := 0
2   for each disk G_i
3       a_i := U_i^H q_i
4       a_i ← fft(a_i with appended zeros)
5       for all disks G_j well separated from G_i
6           b_j ← b_j + λ_jℓ ⊙ a_ℓ
7   for each disk G_i
8       b_i ← leading elements of ifft(b_i)
9       u_i := U_i b_i
10      for all disks G_j not well separated from G_i
11          u_i ← u_i + D_ij q_j
```

The principal message that should be conveyed by this pseudocode segment is that a multipole algorithm is not necessarily difficult to implement. A simple code can capture the essential ideas, exhibit the expected performance, and invite experimentation and further development. The code does become more complicated if we must allow more general particle distributions, but the same sort of code inflation attends other algorithms for the numerical solution of PDEs when the problem

domain does not have a simple shape.

Unfortunately, this advertisement for multipole is flawed, since

Algorithm 2.1 is numerically unstable.

In fact, any algorithm that relies on the basis $\{\varphi_m\}$ in (2.9) or $\{\psi_m\}$ in (2.27) is bound to run into trouble. This behavior will be explored further—*and corrected*—in Chapter 5. I have decided against inserting that discussion here, since it would obscure the basic multipole algorithm. But numerical stability is certainly not optional, and you should retain a healthy skepticism until the instability is addressed.

Implementation difficulty aside, the merits of the algorithm should be judged from its time and space complexities. A tabulation of the computational cost of each step in the above list is:

1. N particles $\times \dfrac{O(p) \text{ flops}}{\text{particle}} = O(pN)$ flops
2. s disks $\times \dfrac{O(p \log p) \text{ flops}}{\text{disk}} = O(sp \log p)$ flops
3. $O(s^2)$ well-separated disk pairs $\times \dfrac{O(p) \text{ flops}}{\text{pair}} = O(s^2 p)$ flops
4. s disks $\times \dfrac{O(sp) \text{ flops}}{\text{disk}} = O(s^2 p)$ flops
5. s disks $\times \dfrac{O(p \log p) \text{ flops}}{\text{disk}} = O(sp \log p)$ flops
6. N particles $\times \dfrac{O(p) \text{ flops}}{\text{particle}} = O(pN)$ flops
7. N particles $\times \dfrac{O(N/s) \text{ flops}}{\text{particle}} = O(N^2/s)$ flops

The total is $O(s^2 p + sp \log p + pN + N^2/s)$ flops.

The parameter p is the cutoff of the spectral expansions. Given a relative accuracy goal ϵ, we previously estimated it to be $p = O(k\alpha + \log \epsilon^{-1})$, where α is the disk radius. The particles here are distributed on a line segment of length L, so $\alpha =$

$L/(2s)$ and $p = O(kL/s + \log \epsilon^{-1})$.

If the number of particles N, optical length kL, and accuracy ϵ are prescribed, then the only remaining parameter is the number of disks s, and it should be selected to minimize the total cost. Steps 3 and 4 favor a small number of disks, $s = O(1)$, while the other steps become faster as the number of disks increases to $s = O(N)$. The best choice lies between these extremes, but that choice depends on the relationship between N, kL, and ϵ. Two of the most common situations are:

- Scaling for high accuracy: kL is frozen while $N \to \infty$ and $\epsilon \to 0$
- Scaling for high frequency: N and kL grow at equal rates, so $kL/N = O(1)$ while $N \to \infty$

These scaling laws will be called *accuracy scaling* and *frequency scaling,* respectively.

The growth of N under accuracy scaling reflects the connection of the particle problem to an underlying differential equation. In addition to the multipole truncation error, governed by ϵ, a convergent approximation of the continuous problem introduces a discretization error that decays to zero as $N \to \infty$. If these two errors are forced to be the same size, then a relationship is established between N and ϵ. For instance, if the discretization scheme has order ν, then $\epsilon = O(N^{-\nu})$.

If the number of disks is an increasing function of N, then under accuracy scaling the optical size of the disks approaches zero. The spectral cutoff is $p = O(\log \epsilon^{-1})$. For a fixed ϵ, the total cost is $O(s^2 + N + N^2/s)$ flops. This expression achieves its minimum of $O(N^{4/3})$ flops when $s = O(N^{2/3})$. With this choice of s, and freeing the parameter ϵ, the complexity is $O(N^{4/3} \log \epsilon^{-1})$ operations. (By allowing s to depend on ϵ as well as N, this complexity can be further reduced to $O(N^{4/3} \log^{1/3} \epsilon^{-1})$. But this freedom also causes the time and space complexities to be optimal at different values of s. To simplify matters, we restrict s to be a function of N only.)

The frequency scaling law reflects the Nyquist criterion. To avoid aliasing errors, a continuous signal must be sampled at a rate exceeding twice the frequency of the Fourier component that oscillates most rapidly. The restriction of a plane wave with wavenumber k to a line is a sinusoidal function with an angular frequency of at most k, and the corresponding Nyquist requirement is k/π samples per unit length. On a segment of the line with length L, there must be $N > kL/\pi$ samples.

Scaling the parameters according to the Nyquist criterion gives a more useful measure of an algorithm's capability to solve *large* problems. The growth in N accommodates a greater optical diameter, rather than decreasing the approximation error. With ϵ fixed, the expansion cutoff is $p = O(N/s)$, giving a total cost of $O(sN + N\log(N/s) + N^2/s)$ flops. The asymptotic minimum of $O(N^{3/2})$ flops is achieved for $s = O(\sqrt{N})$ disks.

In Algorithm 2.1, the interaction matrix Φ is represented with the triple ($\{D_{ij}\}$, $\{U_i\}$, $\{\boldsymbol{\lambda}_{ij}\}$), and we should estimate the cost in both time and space of computing this representation. The storage required by each of these components is:

1. Space for $\{D_{ij}\} = O(s)$ matrices$\times \dfrac{O\big((N/s)^2\big) \text{ words}}{\text{matrix}} = O(N^2/s)$ words
2. Space for $\{U_i\} = s$ matrices $\times \dfrac{O(pN/s) \text{ words}}{\text{matrix}} = O(pN)$ words
3. Space for $\{\boldsymbol{\lambda}_{ij}\} = O(s^2)$ vectors $\times \dfrac{O(p) \text{ words}}{\text{vector}} = O(s^2 p)$ words

The total memory requirement is $O(s^2 p + pN + N^2/s)$ words, and this complexity is minimized by the same choices of s that minimize the execution time.

The work required to generate the representation is at least $O(1)$ flops per word. In addition, each vector $\boldsymbol{\lambda}_{ij}$ requires an FFT for its construction, yielding a total computational requirement of $O(s^2 p \log p + pN + N^2/s)$ flops. The term $s^2 p \log p$ is new, and it contributes a factor $\log\log \epsilon^{-1}$ of very slow growth to the cost of accuracy

scaling. It also raises the total cost under frequency scaling to $O(N^{3/2} \log N)$ flops. (The optimum choice of s actually changes slightly, and as few as $O(N^{3/2} \log^{1/2} N)$ flops are possible, but we will ignore the small difference between time and space minimization.)

Table 2.1: Flat Multipole Complexity

Scaling	s	p	Flops	Memory
Accuracy	$O(N^{2/3})$	$O(\log \epsilon^{-1})$	$O(N^{4/3} \log \epsilon^{-1} \log \log \epsilon^{-1})$	$O(N^{4/3} \log \epsilon^{-1})$
Frequency	$O(N^{1/2})$	$O(N^{1/2})$	$O(N^{3/2} \log N)$	$O(N^{3/2})$

Table 2.1 summarizes the time and space complexities of Algorithm 2.1. When N is large, these are substantial improvements on the standard matrix-vector multiply $\boldsymbol{\Phi q}$, which costs $O(N^2)$ flops and $O(N^2)$ storage. Even greater cost reduction can be realized by extensions of the flat multipole algorithm. Section 2.4 concerns an improvement quite specific to the geometric arrangement of particles in Figure 2.6. Greater flexibility is possible with a multilevel generalization of Algorithm 2.1, to be carried out in Section 2.5.

2.4 Multipole–Grid

The simple pattern of cluster disks in our example produces additional structure in the factorization (2.54). Since they are collinear and separated by equal distances, the centers of disks $\{G_i\}$ compose a uniform 1-D grid. This grid imposes more Toeplitz structure on the factorization.

As shown in (2.44), the elements of the translation matrix T_{ij} depend only on $k\boldsymbol{d}_{ij}$, where $\boldsymbol{d}_{ij}$ is the displacement vector from the center of disk G_j to the center

of disk G_i. Since $\boldsymbol{d}_{13} = \boldsymbol{d}_{24} = \boldsymbol{d}_{35}$, it follows that $T_{13} = T_{24} = T_{35}$. In general, $\boldsymbol{d}_{ij} = \boldsymbol{d}_{i+1,j+1}$ implies $T_{ij} = T_{i+1,j+1}$. So the blocks T_{ij} repeat along the diagonals of the block translation matrix

$$\begin{aligned} \mathcal{T} &:= \begin{bmatrix} 0 & 0 & T_{13} & T_{14} & T_{15} \\ 0 & 0 & 0 & T_{24} & T_{25} \\ T_{31} & 0 & 0 & 0 & T_{35} \\ T_{41} & T_{42} & 0 & 0 & 0 \\ T_{51} & T_{52} & T_{53} & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & T_{13} & T_{14} & T_{15} \\ 0 & 0 & 0 & T_{13} & T_{14} \\ T_{31} & 0 & 0 & 0 & T_{13} \\ T_{41} & T_{31} & 0 & 0 & 0 \\ T_{51} & T_{41} & T_{31} & 0 & 0 \end{bmatrix}. \end{aligned} \tag{2.58}$$

$\mathcal{T}$ has two layers of Toeplitz structure: It is block Toeplitz, and each block is itself a Toeplitz matrix.

This double Toeplitz structure occurs throughout the field of image processing. It is a property shared by all matrix representations of linear shift-invariant filters that operate on 2-D images. More generally, a matrix with d levels of Toeplitz structure represents a linear shift-invariant transformation of a uniformly sampled d-dimensional signal.

Algorithm 2.1 made use of the fine-grained Toeplitz structure in $\mathcal{T}$, but not of the coarse-grained Toeplitz structure at the block level. This extra structure can, however, be utilized by extending the procedure for a fast Toeplitz matrix-vector product.

Just as any scalar Toeplitz matrix T_1 can be embedded into a scalar circulant matrix C_1, any two-level Toeplitz matrix T_2 can be embedded into a two-level circulant matrix C_2. For example, let A, B, C, and D be $m \times n$ Toeplitz matrices, and let $\widetilde{A}$, $\widetilde{B}$, $\widetilde{C}$, and $\widetilde{D}$ be their respective $M \times M$ circulant matrix embeddings.

Then

$$T_2 := \begin{bmatrix} A & D \\ B & A \\ C & B \end{bmatrix} \mapsto \begin{bmatrix} \widetilde{A} & \widetilde{D} \\ \widetilde{B} & \widetilde{A} \\ \widetilde{C} & \widetilde{B} \end{bmatrix} \mapsto \left[\begin{array}{cc|cc} \widetilde{A} & \widetilde{D} & \widetilde{C} & \widetilde{B} \\ \widetilde{B} & \widetilde{A} & \widetilde{D} & \widetilde{C} \\ \widetilde{C} & \widetilde{B} & \widetilde{A} & \widetilde{D} \\ \hline \widetilde{D} & \widetilde{C} & \widetilde{B} & \widetilde{A} \end{array}\right] =: C_2 \tag{2.59}$$

achieves a double circulant embedding of the double Toeplitz matrix on the left-hand side. The embedding is not quite as simple as for scalar Toeplitz matrices: Although T_2 is a submatrix of C_2, it is not a contiguous block in C_2. In (2.59), if the circulant embedding of each block has been patterned after (2.45), then T_2 is recovered from C_2 by keeping only rows $\{i + qM : 1 \le i \le m,\ 0 \le q \le 2\}$ and columns $\{j + rM : 1 \le j \le n,\ 0 \le r \le 1\}$. This can be compactly expressed as

$$T_2 = (E_{3,4}^T \otimes E_{m,M}^T) C_2 (E_{2,4} \otimes E_{n,M}), \tag{2.60}$$

where $\otimes$ is the Kronecker product[3] and $E_{q,r}$ is composed of the first q columns of the $r \times r$ identity matrix.

A double circulant embedding is useful because C_2 is diagonalized by a 2-D Fourier transform. Let $C_2 \in \mathbb{C}^{MN \times MN}$ be partitioned into $N \times N$ blocks of shape $M \times M$. Its eigenvalue decomposition is

$$C_2 = (F_N \otimes F_M)^{-1} \operatorname{diag}(\boldsymbol{\lambda})(F_N \otimes F_M), \tag{2.61}$$

where the eigenvalue vector is the 2-D DFT of the first column,

$$\boldsymbol{\lambda} = (F_N \otimes F_M) C_2 \boldsymbol{e}_1. \tag{2.62}$$

The 2-D DFT matrix $F_N \otimes F_M$ is the Kronecker product of 1-D DFT matrices F_N

[3]If $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{p \times q}$, then the matrix elements of $A \otimes B \in \mathbb{C}^{mp \times nq}$ are the products $a_{ij} b_{k\ell}$ for all index combinations (i, j, k, ℓ). If $A \otimes B$ is partitioned into blocks of size $p \times q$, the block in position (r, s) is $a_{rs} B$. This is a matrix representation of the rank-4 tensor product of rank-2 tensors with components a_{ij} and $b_{k\ell}$.

and F_M with orders N and M, respectively.

An efficient way of computing the product of $F_N \otimes F_M$ with an arbitrary vector $\boldsymbol{x}$ starts by partitioning $\boldsymbol{x}$ into N subvectors of length M, so $\boldsymbol{x}^T = \left[\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_N^T\right]$. Next, construct the $M \times N$ matrix X that has $\boldsymbol{x}_i$ in its ith column. (In a fast image filtering application, X is the $M \times N$ image.) Then

$$\begin{aligned}(F_N \otimes F_M)\boldsymbol{x} &= \operatorname{vec}(F_M X F_N^T) \\ &= \operatorname{vec}(F_M X F_N),\end{aligned} \tag{2.63}$$

where the linear operator $\operatorname{vec}(\cdot)$ concatenates the columns of its matrix argument into a single vector. In this way the 2-D DFT can be implemented with 1-D FFTs, and its complexity is $O(MN \log MN)$ flops.

This fast product is applied to the multipole algorithm by starting from the factorization

$$\Phi = \operatorname{tridiag}(D_{ij}) + \operatorname{diag}(U_i)\mathcal{T}\operatorname{diag}(V_j). \tag{2.64}$$

For any $P \geq 4p + 1$ and $Q \geq 2s - 1$, the aggregate translation matrix $\mathcal{T}$ can be embedded into a $Q \times Q$ block circulant matrix with $P \times P$ circulant blocks. Using the diagonalization (2.61), $\mathcal{T}$ is factored as

$$\begin{aligned}\mathcal{T} &= (E_Q \otimes E_P)^T (F_Q \otimes F_P)^{-1} \operatorname{diag}(\boldsymbol{\lambda})(F_Q \otimes F_P)(E_Q \otimes E_P) \\ &= (E_Q^T F_Q^{-1} \otimes E_P^T F_P^{-1}) \operatorname{diag}(\boldsymbol{\lambda})(F_Q E_Q \otimes F_P E_P).\end{aligned} \tag{2.65}$$

The embedding is accomplished by the tensor product of matrices E_P and E_Q. E_P is the first $2p+1$ columns of the $P \times P$ identity matrix, and E_Q is the first s columns of the $Q \times Q$ identity matrix. The eigenvalue vector is

$$\boldsymbol{\lambda} = (F_Q \otimes F_P)\boldsymbol{h}, \tag{2.66}$$

where the vector $\boldsymbol{h}$ is defined as

$$\boldsymbol{h}^T := \big[\underbrace{0,\ldots,0,}_{2P}\boldsymbol{h}_{31}^T,\boldsymbol{h}_{41}^T,\ldots,\boldsymbol{h}_{s1}^T,\underbrace{0,\ldots,0,}_{P(Q-2s+1)}\boldsymbol{h}_{1s}^T,\ldots,\boldsymbol{h}_{14}^T,\boldsymbol{h}_{13}^T,\underbrace{0,\ldots,0}_{P}\big]^T. \tag{2.67}$$

The subvectors $\boldsymbol{h}_{ij}$ in this definition are taken from (2.57).

Algorithm 2.2 applies the new factorization to compute $\Phi\boldsymbol{q}$. The functions **fft2** and **ifft2** implement the 2-D Fourier transform and its inverse. Each takes a matrix argument of shape $P \times Q$. Thus in line 4, A is placed in the upper-left corner of a $P \times Q$ zero matrix, and in line 6 the same upper-left corner of B is kept to form the vector $\boldsymbol{b}$.

For clarity, new variables are defined in lines 3, 5, and 6, but clearly no new storage need be allocated at any of these points.

The eigenvalue vector $\boldsymbol{\lambda}$ is passed to Algorithm 2.2 as a $P \times Q$ matrix Λ. That matrix is formed by slicing the length-PQ vector $\boldsymbol{\lambda}$ into subvectors of length P, each of which becomes a column of Λ. Equivalently, (2.66) can be rewritten as

$$\Lambda = F_P\big[\mathbf{0},\mathbf{0},\boldsymbol{h}_{31},\ldots,\boldsymbol{h}_{s1},\underbrace{\mathbf{0},\ldots,\mathbf{0},}_{Q-2s+1}\boldsymbol{h}_{1s},\ldots,\boldsymbol{h}_{13},\mathbf{0}\big]F_Q. \tag{2.68}$$

The eigenvalue decomposition of the block matrix $\mathcal{T}$ has been used in place of steps 2–5 of the flat multipole algorithm on page 81. These steps perform the matrix-vector product $\boldsymbol{b} = \mathcal{T}\boldsymbol{a}$, which translates exterior expansions to interior expansions. Using 2-D FFTs in the multipole–grid algorithm, this computation requires $O(sp\log sp)$ flops, instead of the $O(s^2p + sp\log p)$ operations required by flat multipole, or the $O(s^2p^2)$ operations of the standard matrix-vector product.

The complexity of Algorithm 2.2, including its set up, is $O(sp\log sp+pN+N^2/s)$ time and $O(sp + pN + N^2/s)$ space.

Algorithm 2.2: 1-D Multipole–Grid

```
    Input: Charge amplitudes {q_i},
      Representation ({D_ij}, {U_i}, Λ) of Φ,
      Set of s disks {G_i} on a uniform 1-D grid
    Output: Field values {u_i}

1   for each disk G_i
2       a_i := U_i^H q_i
3   A := [a_1, ..., a_s]
4   A ← fft2(A with appended zero rows and columns)
5   B := ifft2(A ⊙ Λ)
6   b := vec(leading submatrix of B)
7   for each disk G_i
8       u_i := U_i b_i
9       for all disks G_j not well separated from G_i
10          u_i ← u_i + D_ij q_j
```

With a choice of s different from that in Table 2.1, a faster algorithm results. With frequency scaling, for which $kL/N = O(1)$, choosing $s = O(N)$ disks produces an algorithm that costs a total of only $O(N \log N)$ flops and $O(N)$ memory. This choice of s corresponds to a fine-grained classification of the particles in which each disk contains only a small cluster. This choice is also appropriate to accuracy scaling, and Table 2.2 summarizes the costs in either case.

Table 2.2: Multipole–Grid Complexity

Scaling	s	p	Flops	Memory
Accuracy	$O(N)$	$O(\log \epsilon^{-1})$	$O\big(N \log \epsilon^{-1} \log(N \log \epsilon^{-1})\big)$	$O(N \log \epsilon^{-1})$
Frequency	$O(N)$	$O(1)$	$O(N \log N)$	$O(N)$

Ignoring the influence of the accuracy parameter ϵ, these costs are asymptotically identical to the case of equally spaced particles, in which the interaction matrix Φ itself is Toeplitz.

The multipole–grid algorithm extends to regular grids of higher dimension. Pictured in Figure 2.7 is a uniform distribution of charged particles on a rectangle. With a lexicographic ordering applied to the clusters, the block translation matrix $\mathcal{T}$ has three levels of Toeplitz structure, and a fast algorithm for $\boldsymbol{b} = \mathcal{T}\boldsymbol{a}$ uses 3-D FFTs.

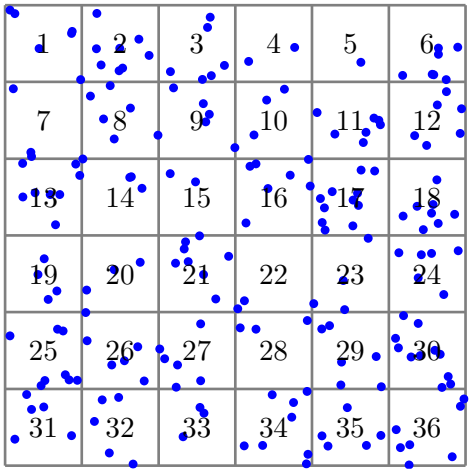


Figure 2.7: Particles are distributed uniformly inside a square in the plane. The particles are classified into clusters by dividing the square with a regular grid. If the particles in each cluster are labeled with consecutive integers, and if the clusters are enumerated as shown here, then $\mathcal{T}$ will have three levels of Toeplitz structure.

The complexity analysis changes for a 2-D particle distribution. If the particles fill a square with side length L, and the square is divided into s square cells of the same size, then the radius of each cluster is $\alpha = L/\sqrt{2s}$. The Nyquist criterion is $(kL)^2/N = O(1)$. Combining these two relations, the spectral expansion cutoff is $p = O(k\alpha) = O(\sqrt{N/s})$. For the 1-D distribution, the cutoff was determined to be $p = O(N/s)$.

If a flat multipole method is applied to the configuration of Figure 2.7, the op-

timal choice of the number of clusters is $s = O(N^{3/5})$, giving the costs displayed in Table 2.3. For a multipole–grid solution, the optimal choice is $s = O(N)$, the same choice made in the 1-D case. The asymptotic cost of the multipole–grid algorithm is independent of the number of space dimensions.

Table 2.3: High Frequency Complexity for Uniform 2-D Distributions.

Method	s	p	Flops	Memory
Flat multipole	$O(N^{3/5})$	$O(N^{1/5})$	$O(N^{7/5} \log N)$	$O(N^{7/5})$
Multipole–grid	$O(N)$	$O(1)$	$O(N \log N)$	$O(N)$

If the particle distribution is not uniform on a rectangle in $\mathbb{E}^d$, then the multipole–grid algorithm loses its effectiveness. For instance, in Figure 2.8, the particles are arranged on an ellipse in $\mathbb{E}^2$. A regular 2-D array of disks could be superimposed on this curved distribution, enabling the application of a multipole–grid method. Again we should choose $s = O(N)$, but most disks will contain no particles. The cost of the algorithm grows to $O(N^{3/2} \log N)$ time and $O(N^{3/2})$ space.

The flat multipole algorithm can, however, be extended in another way that performs well even for highly nonuniform particle distributions.

2.5 Hierarchical Multipole

In flat multipole, particles are collected into disjoint clusters. The guiding principle now will be to organize the particles into a cluster hierarchy. Figure 2.9 is a simple example of such a construction.

The spatial hierarchy is represented with a forest, a collection of trees. The trees that map the cluster of hierarchy of Figure 2.9 are shown in Figure 2.10. In this

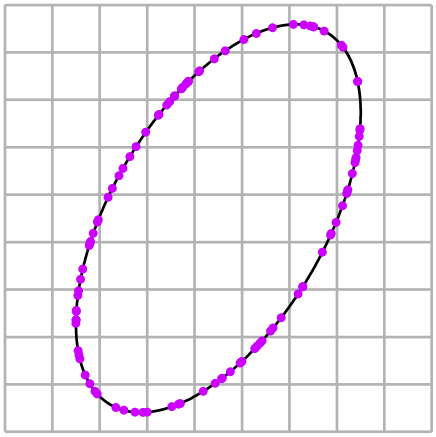


Figure 2.8: Particles are distributed uniformly on an ellipse in the plane. A square grid overlays the ellipse, to facilitate the application of a multipole–grid method. The particles inside each grid cell constitute a single cluster. Because most cells are empty, a hierarchical multipole method is more efficient.

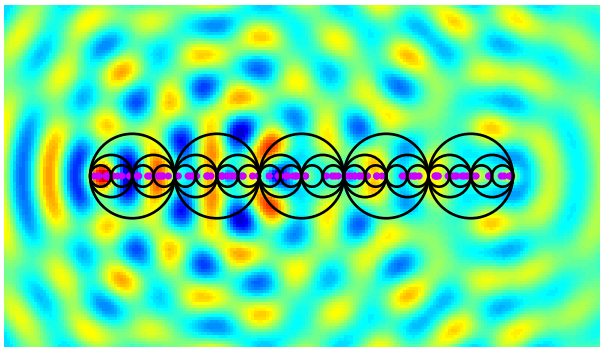


Figure 2.9: Real part of the field generated by 100 point oscillators uniformly distributed on the interval $[0, L]$ of the x-axis. The particles are grouped into a three-level hierarchy of clusters, with 5 large clusters at the top of the hierarchy and 20 small clusters at the bottom.

example the forest consists only of balanced binary trees, but that is not required. For nonuniform particle distributions, some trees may be shallow, with only a few leaves, while others may be deep, with many levels of branching and many leaves. For the moment we will not dwell on the construction of the forest from a given particle distribution, but the usual approach in 2-D problems is to use adaptive quadtrees [19] [110].

As indicated in Figure 2.10, each node in a tree stands for a single cluster of the hierarchy. Each tree is composed of nested particle clusters. With each node is also associated a disk that contains its cluster. This disk, typically the smallest disk that encloses the cluster, is the entity central to an analysis of the multipole expansion of the field generated by the cluster.

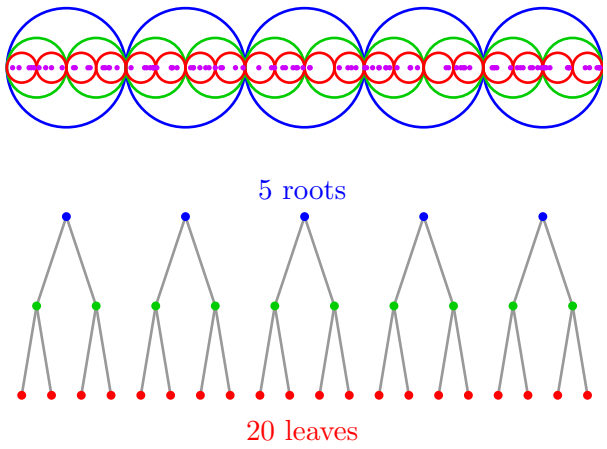


Figure 2.10: Nesting relationships of the particle clusters are described by a set of binary trees. The tree roots correspond to the largest clusters. The tree leaves correspond to the smallest clusters.

The forest of trees can always be merged into a single tree by adding one or more levels to connect the individual roots. Sometimes it is convenient to do this, but typically a hierarchical multipole algorithm is more efficient if the top of a single cluster tree is truncated, creating a forest. Distinguishing the case of a single tree

from the case of multiple trees is unnecessary. Henceforth, I use the word "tree" loosely, understanding it to mean a collection of one or more trees.

In a hierarchical multipole method, the leaves of the tree should be small clusters. Any large leaf cluster would benefit from further division. In our simple example, the leaf disks are all the same size. If the spectral expansions belonging to each leaf disk have length $2p+1$, then each leaf cluster should contain $O(p)$ particles. For general particle distributions, adaptive quadtrees conveniently allow the specification of an upper bound on the number of particles in each leaf.

It is also desirable to impose a lower bound on the number of particles in each leaf. If a leaf cluster were to contain fewer than $2p+1$ particles, then the multipole expansion would be a more complex description of the exterior field than the natural expansion (2.7). Unfortunately, adaptive quadtrees cannot ensure a lower bound. The adaptive FMM of Carrier, Greengard, and Rokhlin [19] uses adaptive quadtrees, but automatically switches to the natural representation when it is more efficient. Another possible solution is abandon adaptive quadtrees for another spatial data structure, such as a binary space partition, that accommodates both upper and lower bounds.

We shall assume that each leaf cluster contains at least—but not many more than—$2p+1$ particles. There are a total of $s = O(N/p)$ leaves.

The hierarchy enables a reduction in the number of spectral expansion translations. Consider the flat particle classification consisting only of the leaf clusters. There are $O(s^2)$ well-separated pairs of leaf disks, and computing the interactions among these pairs is too expensive. The solution in the flat multipole method is to make the clusters larger, so that s takes on the values in Table 2.1. Unfortunately, that approach also increases the work required to compute the interactions among

disk pairs that are not well separated.

The hierarchical multipole method reduces the number of translations in a different way. If two groups of disks are separated by a large gap, then it will be more efficient to represent the interactions among their constituent particles with spectral expansions on a coarser level of clustering. Applying this idea repeatedly to a uniform particle distribution, the number of translations can be reduced from $O(s^2)$ to $O(s)$. Unlike flat multipole, this approach avoids inflating the cost of computing short-range interactions.

The complication introduced by the hierarchy is that new translation operators are needed. The exterior expansions belonging to sibling nodes in the tree must be merged into a single exterior expansion belonging to their parent. Furthermore, the interior expansion of a parent must be split into separate interior expansions for each of its children.

2.5.1 Branch Translations

The new translations link parent disks with their descendants. For every branch in the cluster tree, there is one translation up the tree, from child to parent, and one translation down the tree, from parent to child. As with the translations between well-separated disks (Section 2.3.1), Graf's addition theorem will be used to derive the matrix elements of the new translation operators.

First consider the translation of an exterior spectral expansion up the tree. Figure 2.11 illustrates the relationship between an expansion with origin at the center of disk G,

$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} a_n H_n(k\|\boldsymbol{x}\|)e^{in\theta(\boldsymbol{x})}, \tag{2.69}$$

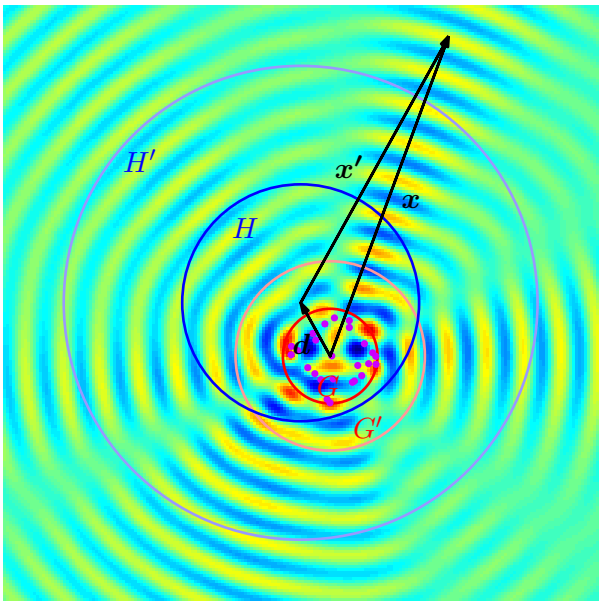


Figure 2.11: Real part of the field generated by 25 point oscillators distributed randomly in the disk G. The spectral expansion (2.10) represents this field in the exterior of G'. Now shift the expansion origin by the displacement vector $\boldsymbol{d}$ to the center of disk $H \supset G$. A new spectral expansion represents the same field in the exterior of H'.

and the equivalent expansion referenced to an origin at the center of the enclosing disk $H \supset G$. Let the point $\boldsymbol{x}$ lie in the exterior of H. If $\boldsymbol{x} = \boldsymbol{d} + \boldsymbol{x'}$, where $\boldsymbol{d}$ is the displacement vector from the center of G to the center of H, then we seek a new representation

$$\begin{aligned} u(\boldsymbol{d}+\boldsymbol{x'}) &= \sum_{n=-\infty}^{\infty} a_n H_n(k\|\boldsymbol{d}+\boldsymbol{x'}\|) e^{in\theta(\boldsymbol{d}+\boldsymbol{x'})} \\ &= \sum_{m=-\infty}^{\infty} a'_m H_m(k\|\boldsymbol{x'}\|) e^{im\theta(\boldsymbol{x'})}, \end{aligned} \tag{2.70}$$

of the same field $u(\boldsymbol{x})$ in (2.69).

Since $\|\boldsymbol{d}\| < \|\boldsymbol{x'}\|$, the addition theorem (2.38) can be applied. After making the

substitutions $\boldsymbol{z} \longrightarrow k\boldsymbol{d}$, $\boldsymbol{w} \longrightarrow k\boldsymbol{x'}$, and $m \rightarrow n - m$, the formula becomes

$$H_n(k\|\boldsymbol{d} + \boldsymbol{x'}\|)e^{in\theta(\boldsymbol{d}+\boldsymbol{x'})} = \sum_{m=-\infty}^{\infty} J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} H_m(k\|\boldsymbol{x'}\|)e^{im\theta(\boldsymbol{x'})}. \quad (2.71)$$

Inserting this into (2.69), we have

$$\begin{aligned} u(\boldsymbol{d} + \boldsymbol{x'}) &= \sum_{n=-\infty}^{\infty} a_n \left(\sum_{m=-\infty}^{\infty} J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} H_m(k\|\boldsymbol{x'}\|)e^{im\theta(\boldsymbol{x'})} \right) \\ &= \sum_{m=-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} a_n J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} \right) H_m(k\|\boldsymbol{x'}\|)e^{im\theta(\boldsymbol{x'})}, \end{aligned} \quad (2.72)$$

so the new coefficients are revealed to be

$$a'_m = \sum_{n=-\infty}^{\infty} a_n J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})}. \quad (2.73)$$

The coefficients $\{a'_m\}$ are a linear shift-invariant transformation of the coefficients $\{a_n\}$.

In an actual implementation, the infinite series must be approximated by finite sums. Let $a_n = 0$ for $|n| > q$, so that the expansion coefficients belonging to G form a vector $\boldsymbol{a} = \left[a_{-q}, \ldots, a_q\right]^T$. If we confine our interest to the coefficients a'_m for $|m| \leq p$, also organized as a vector $\boldsymbol{a'} = \left[a'_{-p}, \ldots, a'_p\right]^T$, then the translation operation assumes the form of a matrix-vector product,

$$\boldsymbol{a'} = S\boldsymbol{a}, \quad (2.74)$$

where S is the Toeplitz matrix with elements

$$\begin{aligned} s_{mn} &= J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} \\ &= J_{m-n}(k\|\boldsymbol{d}\|)e^{-i(m-n)\theta(-\boldsymbol{d})}, \end{aligned} \quad (2.75)$$

where the row index ranges over $-p, \ldots, p$ and the column index over $-q, \ldots, q$.

As with the translation matrix T in (2.41), it is critical to utilize the Toeplitz

structure of S when k is large.

The analysis for the translation of an interior expansion down the tree is nearly identical. Refer to Figure 2.12. The expansion inside H is

$$u(\boldsymbol{x'}) = \sum_{n=-\infty}^{\infty} b'_n J_n(k\|\boldsymbol{x'}\|)e^{in\theta(\boldsymbol{x'})}, \tag{2.76}$$

and it must be connected to an interior expansion referenced to the center of disk G,

$$u(\boldsymbol{d}+\boldsymbol{x}) = \sum_{m=-\infty}^{\infty} b_m J_m(k\|\boldsymbol{x}\|)e^{im\theta(\boldsymbol{x})}, \tag{2.77}$$

where $\boldsymbol{d}$ is the displacement vector from the center of H to the center of G. The coefficients $\{b_m\}$ are a linear transformation of the coefficients $\{b'_n\}$.

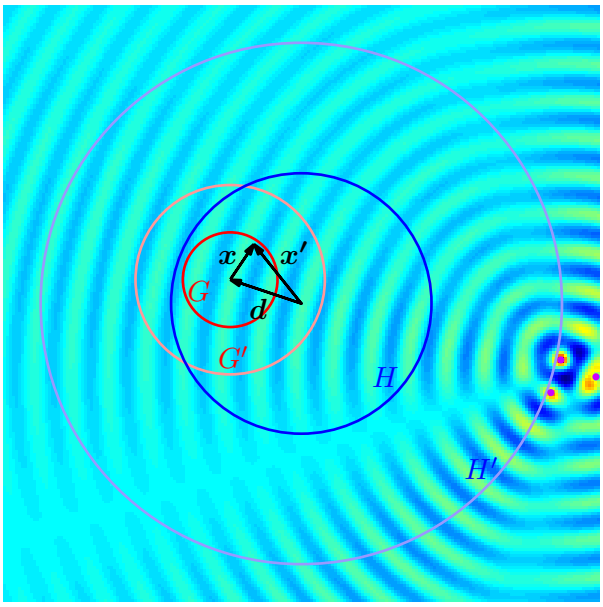


Figure 2.12: Real part of the field generated by 3 point oscillators in the exterior of H'. The spectral expansion (2.28) represents this field in the interior of H. Now shift the expansion origin by the displacement vector $\boldsymbol{d}$ to the center of disk $G \subset H$. A new spectral expansion represents the same field inside G.

To determine the matrix elements of this transformation, we need an addition

theorem that is slightly different from (2.38). That formula remains valid if all Hankel functions H_ν are replaced with Bessel functions J_ν. Furthermore, the restriction $\|\boldsymbol{z}\| < \|\boldsymbol{w}\|$ may be dropped, giving

$$J_n(\|\boldsymbol{z}+\boldsymbol{w}\|)e^{in\theta(\boldsymbol{z}+\boldsymbol{w})} = \sum_{m=-\infty}^{\infty} J_m(\|\boldsymbol{z}\|)e^{im\theta(\boldsymbol{z})}J_{n-m}(\|\boldsymbol{w}\|)e^{i(n-m)\theta(\boldsymbol{w})} \tag{2.78}$$

for arbitrary vectors $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{E}^2$. This is another specialization of Graf's addition theorem.

After making the same substitutions as for the exterior translation, (2.78) is inserted into (2.76), yielding

$$\begin{aligned} u(\boldsymbol{d}+\boldsymbol{x}) &= \sum_{n=-\infty}^{\infty} b'_n \left(\sum_{m=-\infty}^{\infty} J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})}J_m(k\|\boldsymbol{x}\|)e^{im\theta(\boldsymbol{x})} \right) \\ &= \sum_{m=-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} b'_n J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} \right) J_m(k\|\boldsymbol{x}\|)e^{im\theta(\boldsymbol{x})}, \end{aligned} \tag{2.79}$$

and a comparison with (2.77) shows that the new coefficients are

$$b_m = \sum_{n=-\infty}^{\infty} b'_n J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})}. \tag{2.80}$$

After truncation, the translation operation assumes the form of a matrix-vector product,

$$\boldsymbol{b} = R\boldsymbol{b'}, \tag{2.81}$$

where the matrix R has elements

$$\begin{aligned} r_{mn} &= J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})} \\ &= J_{m-n}(k\|\boldsymbol{d}\|)e^{-i(m-n)\theta(-\boldsymbol{d})}. \end{aligned} \tag{2.82}$$

Since r_{mn} is a function of the index displacement $m-n$, R is a Toeplitz matrix.

For any parent–child pair in the tree, the translation from parent to child, with matrix elements (2.82), is closely related to the translation from child to parent,

with matrix elements (2.75). Let $\boldsymbol{d}_{GH}$ be the displacement vector from the center of disk G to the center of its parent disk H. If the cutoffs of the interior and exterior spectral expansions for disks H and G are p and q, respectively, then the elements of the translation matrix $S \in \mathbb{C}^{(2p+1)\times(2q+1)}$ are

$$s_{mn} = J_{n-m}(k\|\boldsymbol{d}_{GH}\|)e^{i(n-m)\theta(\boldsymbol{d}_{GH})}, \tag{2.83}$$

and the elements of the translation matrix $R \in \mathbb{C}^{(2q+1)\times(2p+1)}$ are

$$\begin{aligned} r_{mn} &= J_{n-m}(k\|-\boldsymbol{d}_{GH}\|)e^{i(n-m)\theta(-\boldsymbol{d}_{GH})} \\ &= J_{m-n}(k\|\boldsymbol{d}_{GH}\|)e^{-i(m-n)\theta(\boldsymbol{d}_{GH})} \\ &= \overline{s_{nm}}, \end{aligned} \tag{2.84}$$

so R and S are mutually adjoint: $R = S^H$.

2.5.2 The Multipole Dag

The flat multipole and multipole–grid algorithms follow from different factorizations of the interaction matrix Φ. It is possible to take the same approach with hierarchical multipole. For the uniform linear particle distribution of Figure 2.9, the result is a coherent and illuminating factorization (which I shall not give here). For nonuniform distributions, the factorization is hopelessly cluttered. [But see (4.13).]

Instead of a matrix block partition, a graph can give an alternative description of the same computation. With its added flexibility, the computational graph for hierarchical multipole is easier to digest than the matrix structure. Most importantly, the graph structure is not appreciably different for uniform and nonuniform particle distributions.

The multipole graph is a signal flow dag[4] in which data vectors flow from vertex to vertex, following the edges. Each edge has an associated matrix weight, and on traversing the edge, a data vector is multiplied by this matrix. Thus each edge in the graph stands for a linear system connecting the signals at two vertices.

The dag vertices are data synchronization points. Multiple data vectors incident upon a single vertex are added together, and a copy of this sum is issued to each edge leaving the vertex. Since the graph is a dag, there are no feedback loops in the signal flow.

The vertices and edges of the graph are labeled. The unlabeled graph shows the pattern of data flow, and the labels specify the precise nature of each computation. All necessary computational components have already been assembled.

A vertex label describes the type of data distributed from the vertex. There are four kinds of vertices:

- Particle charges $\boldsymbol{q}_i$ (§2.1)
- Field values $\boldsymbol{u}_i$ (§2.1)
- Exterior expansion coefficients $\boldsymbol{a}_i$ (§2.2.1)
- Interior expansion coefficients $\boldsymbol{b}_i$ (§2.2.2)

The subscript i in each case refers to a specific node in the cluster tree. Each node in the tree has exterior and interior coefficient vectors, but only the leaf nodes have charge and field vectors.

The edge labels are the matrix weights. There are six kinds:

[4]A graph $G = (V, E)$ consists of a set $V = \{v_i\}$ of vertices and a set E of edges that connect the vertices. In a directed graph, the edges are ordered pairs of vertices, so the edge (v_i, v_j) is distinct from the edge (v_j, v_i). In a directed acyclic graph (dag), for each vertex v_i there can be no directed paths that both originate and terminate at v_i.

- Short-range interactions $D_{ij} : \boldsymbol{q}_j \mapsto \boldsymbol{u}_i$ (§2.3.3)
- Interior expansion evaluations $U_i : \boldsymbol{b}_i \mapsto \boldsymbol{u}_i$ (§2.2.2)
- Interior–interior translations $R_{ij} : \boldsymbol{b}_j \mapsto \boldsymbol{b}_i$ (§2.5.1)
- Exterior–interior translations $T_{ij} : \boldsymbol{a}_j \mapsto \boldsymbol{b}_i$ (§2.3.1)
- Exterior–exterior translations $S_{ij} : \boldsymbol{a}_j \mapsto \boldsymbol{a}_i$ (§2.5.1)
- Exterior expansion generations $V_i : \boldsymbol{q}_i \mapsto \boldsymbol{a}_i$ (§2.2.1)

If the subscripts i and j are allowed to stand for any node in the cluster tree, then these operator sequences are sparse. In particular,

$$\begin{aligned} D_{ij} \neq 0 \quad &\Leftrightarrow \quad i, j \text{ are leaf nodes that are not well separated,} \\ U_i, V_i \neq 0 \quad &\Leftrightarrow \quad i \text{ is a leaf node,} \\ R_{ji}, S_{ij} \neq 0 \quad &\Leftrightarrow \quad \text{node } i \text{ is node } j\text{'s parent,} \end{aligned}$$

and a partial characterization of the sparsity pattern of $\{T_{ij}\}$ is

$$T_{ij} \neq 0 \quad \begin{array}{c} \Rightarrow \\ \not\Leftarrow \end{array} \quad i, j \text{ are well-separated nodes.}$$

To the extent that all of these sequences are sparse, the multipole graph itself is sparse.

With the vertices and edges defined in terms of mathematical objects already covered, it is not difficult to finish the graph specification. We consider various sections of the graph before exhibiting the entire dag for the example of Figure 2.9.

Figure 2.13 shows the edges $\{D_{ij}\}$ that give the fields produced by particles in a leaf cluster j at the locations of particles contained in a leaf cluster i not well separated from j. Each leaf cluster appears as two graph vertices here, one associated with the input data $\boldsymbol{q}_i$ and another associated with the output data $\boldsymbol{u}_i$.

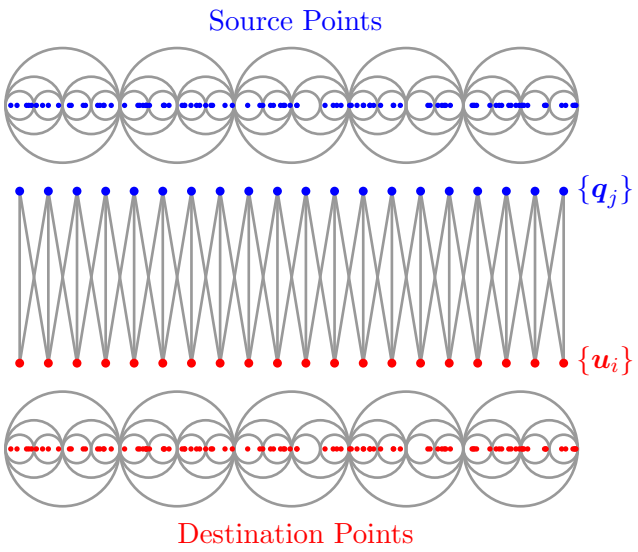


Figure 2.13: Interactions between leaf clusters that are not well separated. For each interior leaf cluster, there are three such interactions: one self-interaction among the cluster's particles and two between the cluster's particles and those in the two adjacent clusters. Each edge, directed from top to bottom, takes a vector input of charges $\boldsymbol{q}_j$ and left-multiplies that vector by the matrix D_{ij} to give an output vector that is accumulated with others to form the field values $\boldsymbol{u}_i$. To emphasize the connection between dag and particles, the source points and their disk hierarchy appear above the input vertices of the dag. The destination points and their disk hierarchy appear below the output vertices of the dag.

Two copies of the cluster tree appear in Figure 2.14. In the bottom half of the figure, node i is labeled with the interior expansion coefficients $\boldsymbol{b}_i$. The branches are labeled with the translation matrices $\{R_{ij}\}$ that split a parent's interior expansion into a separate interior expansion for each of its children. In the top half of the figure, node i is labeled with the exterior coefficients $\boldsymbol{a}_i$, and the branches are labeled with the translation matrices $\{S_{ij}\}$ that merge sibling exterior expansions into a single exterior expansion for the parent.

To avoid edge cycles, the cluster tree must appear twice in the multipole graph. Moreover, in Problem 2.2 the source and destination points are not the same, and separate cluster trees may be constructed for these two sets of points. At the top of

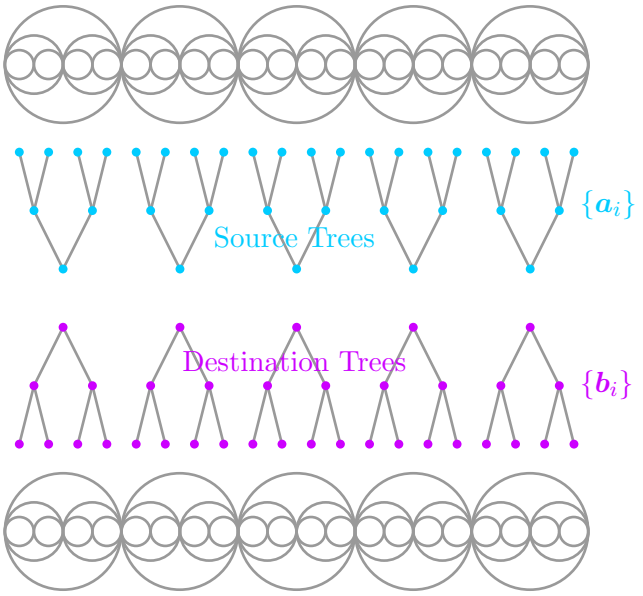


Figure 2.14: Branch translations $\{R_{ij}\}$ and $\{S_{ij}\}$. An edge in the source tree takes a vector $\boldsymbol{a}_j$ of exterior expansion coefficients for node j, premultiplies it by S_{ij}, and outputs a vector that is accumulated with its sibling into the exterior expansion coefficients $\boldsymbol{a}_i$ of j's parent node i. An edge in the destination tree takes a vector $\boldsymbol{b}_j$ of interior expansion coefficients for node j, premultiplies it by R_{ij}, and outputs a vector that is accumulated into the interior expansion coefficients $\boldsymbol{b}_i$ of j's child node i. For further illustration, the nested disks that produce the source and destination trees are shown here as well.

the figure is the source tree, and at the bottom of the figure is the destination tree. We are mainly concerned with Problem 2.1, and the source and destination trees are identical.

Clearly the source and destination trees need to be connected, and those edges are drawn in Figure 2.15. These are the interactions $\{T_{ij}\}$ that map an exterior expansion at node j of the source tree to an interior expansion at node i of the destination tree. This web of connections is the most complicated part of the multipole graph.

In the work of Greengard and Rokhlin [64], the set of clusters $j(i)$ for which

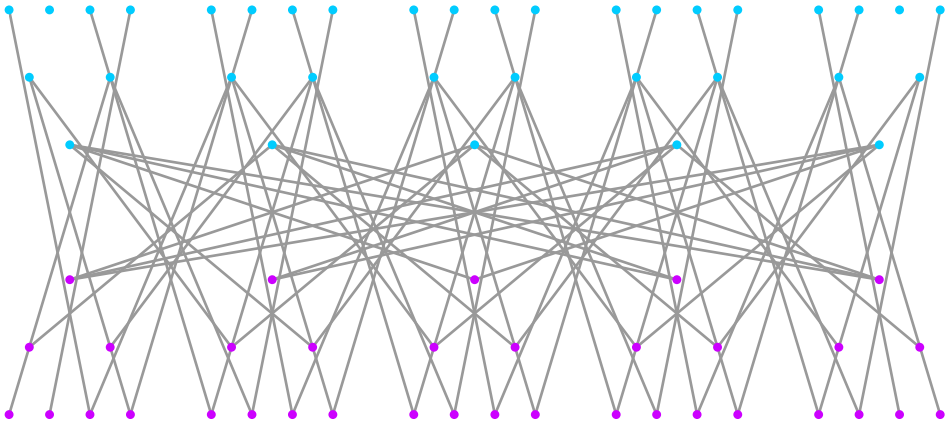


Figure 2.15: Translations between well-separated clusters. The vertices shown are the same as those in Figure 2.14, and the edge topology is the interaction list. Each edge, directed from top to bottom, multiplies the vector $\boldsymbol{a}_j$ at its source vertex by the translation matrix T_{ij}, producing a vector at its destination vertex that is accumulated with the output vectors of all other incident edges to form $\boldsymbol{b}_i$.

$T_{ij} \neq 0$ is called the *interaction list* of node i. More generally, the sparsity pattern of $\{T_{ij}\}$ is the interaction list of the entire tree. The only requirements that these connections must satisfy are

- Any edge between source and destination trees must connect well-separated clusters.
- After including the tree branches, there must be exactly one path connecting any pair of well-separated leaf clusters.

There are many interaction lists that satisfy these conditions, and a good choice will minimize the cost of the algorithm.

The interaction list constructed by Greengard and Rokhlin proved that, for a uniform distribution of s leaf clusters, only $O(s)$ edges are needed. The length of each node's interaction list is bounded by a constant independent of s. In their

construction,

$$T_{ij} \neq 0 \quad \Leftrightarrow \quad \begin{array}{l} i, j \text{ are well-separated nodes, at the same level of} \\ \text{the cluster tree, whose parents are not well sepa-} \\ \text{rated.} \end{array}$$

Without the cluster hierarchy, $O(s^2)$ edges would be needed to connect the well-separated leaf clusters.

In the Greengard–Rokhlin interaction list, $T_{ij} \neq 0$ if and only if $T_{ji} \neq 0$. The sparsity pattern of $\{T_{ij}\}$ is symmetric. In contrast, the interaction list of Figure 2.15 is clearly nonsymmetric.

Given the source and destination trees, an interaction list that is optimal under accuracy scaling will have the fewest possible edges. The Greengard–Rokhlin construction is suboptimal. The interaction list displayed in Figure 2.15, which was computed by another heuristic algorithm, contains 64 edges. The Greengard–Rokhlin interaction list, which only connects clusters at the same level of the tree, contains 66 edges.

With frequency scaling, the optimal solution is harder to characterize, since not all translations require the same work. If each edge connecting source and destination trees is weighted by $(p + q) \log(p + q)$, where p and q are the expansion cutoffs of the connected disks, then an optimal interaction list minimizes the sum of the edge weights.

In Figure 2.16, the graph pieces in Figures 2.13–2.15 are assembled into the entire multipole dag. The only added edges are the simple connections $\{U_i\}$ between the field vertices and the leaves of the destination tree, and $\{V_i\}$ between the charge vertices and the leaves of the source tree.

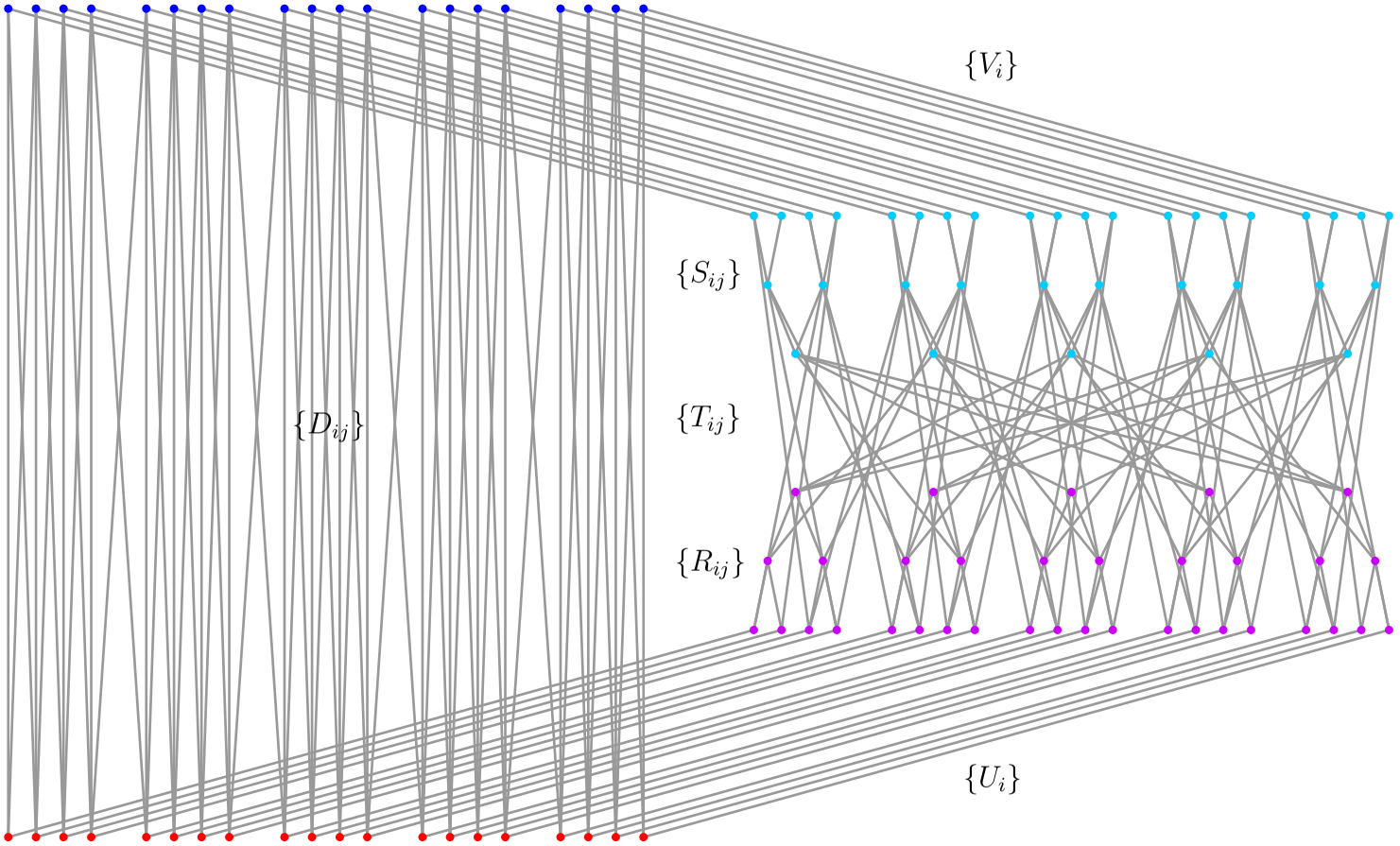


Figure 2.16: The multipole dag that implements $\boldsymbol{u} = \Phi\boldsymbol{q}$. The dag has been topologically sorted, with node priority decreasing from top to bottom. The computation begins with input charges and ends with field values. All intervening computation flows from top to bottom. This figure is assembled from Figures 2.13–2.15, plus the edges $\{V_i\}$ that compute exterior expansion coefficients and the edges $\{U_i\}$ that evaluate interior expansions.

2.5.3 Multilevel Multipole Algorithm

The multipole representation of the interaction matrix Φ is the dag and the six sequences of edge weights $(\{D_{ij}\}, \{U_i\}, \{R_{ij}\}, \{T_{ij}\}, \{S_{ij}\}, \{V_i\})$. The matrices $\{D_{ij}\}$, $\{U_i\}$, and $\{V_i\}$ are unstructured, and slow matrix-vector multiplications will have to suffice when those edges are traversed. But since the matrices $\{T_{ij}\}$, $\{R_{ij}\}$, and $\{S_{ij}\}$ are all Toeplitz, the algorithm of Section 2.3.2 speeds up the computation for those edges.

Recall that an arbitrary Toeplitz matrix $T \in \mathbb{C}^{m\times n}$ has a factorization

$$T = E_m^T F_r^{-1} \operatorname{diag}(\boldsymbol{\lambda}) F_r E_n, \quad r \geq m+n-1, \tag{2.85}$$

where F_r is an order-r DFT matrix and E_n is the first n columns of the $r \times r$ identity matrix.

Applying the pattern (2.85) to each type of Toeplitz matrix, we introduce the following notation for the diagonal forms:

T	$\boldsymbol{\lambda}$
T_{ij}	$\boldsymbol{\lambda}_{ij}$
R_{ij}	$\boldsymbol{\mu}_{ij}$
S_{ij}	$\boldsymbol{\nu}_{ij}$

In lieu of the full matrices on the left-hand side of this table, only the vectors on the right-hand side are stored. The computation of each of these vectors requires an FFT. The formula for $\boldsymbol{\lambda}_{ij}$ has already been given in (2.56) under the assumption that $m = n = 2p+1$. If the expansion lengths in (2.85) are $m = 2p+1$ for node i and $n = 2q+1$ for node j, then the formula for $\boldsymbol{\mu}_{ij}$ is

$$\boldsymbol{\mu}_{ij} = F\boldsymbol{j}_{ij}, \tag{2.86}$$

where

$$\boldsymbol{j}_{ij}^T := \big[J_{-p+q}(k\|\boldsymbol{d}_{ij}\|)e^{-i(-p+q)\theta(-\boldsymbol{d}_{ij})}, \ldots, J_{p+q}(k\|\boldsymbol{d}_{ij}\|)e^{-i(p+q)\theta(-\boldsymbol{d}_{ij})},$$
$$\underbrace{0, \ldots, 0}_{r-m-n+1}, J_{-(p+q)}(k\|\boldsymbol{d}_{ij}\|)e^{i(p+q)\theta(-\boldsymbol{d}_{ij})}, \ldots, J_{-(p-q+1)}(k\|\boldsymbol{d}_{ij}\|)e^{i(p-q+1)\theta(-\boldsymbol{d}_{ij})}\big]. \quad (2.87)$$

This equation uses the symbol i both as subscript index and as $\sqrt{-1}$, but there should be no confusion.

The formula for $\boldsymbol{\lambda}_{ij}$ simply exchanges all Bessel functions J_ν in (2.87) for Hankel functions H_ν.

The formula for $\boldsymbol{\nu}_{ij}$ is the same as (2.86), but the sparsity patterns of $\{\boldsymbol{\mu}_{ij}\}$ and $\{\boldsymbol{\nu}_{ij}\}$ are different. The vector $\boldsymbol{\mu}_{ij}$ is defined by (2.86) only if in the destination tree a branch connects child node i to parent node j. The vector $\boldsymbol{\nu}_{ij}$ is defined only if in the source tree a branch connects parent node i to child node j.

If the source and destination trees are identical, then $\boldsymbol{\nu}_{ij}$ is defined if and only if $\boldsymbol{\mu}_{ji}$ is defined as well. To find the relationship between $\boldsymbol{\nu}_{ij}$ and $\boldsymbol{\mu}_{ji}$, swap subscripts i and j in (2.86). The expansion cutoffs p and q must also be exchanged. Using $\boldsymbol{d}_{ji} = -\boldsymbol{d}_{ij}$ and well-known properties of the discrete Fourier transform, it can be shown that

$$\boldsymbol{\nu}_{ij} = \overline{\boldsymbol{\mu}_{ji}}, \quad (2.88)$$

reflecting the relationship $S_{ij} = R_{ji}^H$ derived in (2.84).

Algorithm 2.3 is a pseudocode encapsulation of the operations in the multipole dag. It relies on two types of depth-first tree traversals. As indicated by the topological sort of the dag in Figure 2.16, child nodes of the source tree have a higher priority than their parents. This is a *postordering* of the nodes: Parent nodes occur only after their children. The nodes of the destination tree are visited in a *pre-*

order, in which a parent occurs before any of its children. Although Algorithm 2.3 is iterative in style, these tree traversals also have a simple recursive implementation.

Algorithm 2.3: Hierarchical Multipole

```
    Input: Charge amplitudes q,
      Representation ({D_ij}, {U_i}, {V_i},
          {λ_ij}, {μ_ij}, {ν_ij}) of Φ
      Source tree T_s,
      Destination tree T_d
    Output: Field values u

1   a := 0,  b := 0
2   for every leaf i ∈ T_s
3       a_i ← V_i q_i
4   for every node j in a postorder traversal of T_s
5       if j is not a root
6           i := j's parent
7           a_i ← a_i + fftconv(ν_ij, a_j)
8       for i ∈ {k : λ_kj ≠ 0}
9           b_i ← b_i + fftconv(λ_ij, a_j)
10  for every node j in a preorder traversal of T_d
11      for each child i of node j
12          b_i ← b_i + fftconv(μ_ij, b_j)
13  for every leaf i ∈ T_d
14      u_i := U_i b_i
15      for every leaf j ∈ T_s not well separated from i
16          u_i ← u_i + D_ij q_j
```

The pseudocode issues three calls to the function **fftconv**, which implements a fast discrete convolution using the FFT. The first argument passed to **fftconv** is assumed to already have been padded and transformed by an FFT. The fast convolution algorithm has already been given in Section 2.3.2, and the pseudocode is listed in Algorithm 2.4.

Algorithm 2.3 encapsulates all the operations in the multipole dag. Dividing the work according to edge type, an equivalent description is:

Algorithm 2.4: Fast Convolution

```
    y = fftconv(λ, x)

1   r := length(λ)
2   n := length(x)
3   y := fft(x with r − n appended zeros)
4   y ← leading r − n + 1 elements of ifft(λ ⊙ y)
```

1. Construct the exterior expansion belonging to each leaf cluster in the source tree: $\boldsymbol{a}_i = V_i \boldsymbol{q}_i$.
2. Traverse the source tree from bottom to top, merging sibling exterior expansions into the parent's exterior expansion: $\boldsymbol{a}_i = \sum_j S_{ij} \boldsymbol{a}_j$.
3. For each node in the destination tree, accumulate into the interior expansion the translations of exterior expansions from the source tree nodes in the interaction list: $\boldsymbol{b}_i = \sum_j T_{ij} \boldsymbol{a}_j$.
4. Traverse the destination tree from top to bottom, accumulating into sibling interior expansions the splitting of the parent's interior expansion: $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + \sum_j R_{ij} \boldsymbol{b}_j$.
5. Evaluate the interior expansion belonging to each leaf cluster in the destination tree: $\boldsymbol{u}_i = U_i \boldsymbol{b}_i$.
6. For each leaf cluster in the destination tree, accumulate into the field vector the contribution of charged particles in neighboring leaf clusters of the source tree: $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i + \sum_j D_{ij} \boldsymbol{q}_j$.

The sums are in every case sparse. The sum in step 4 has at most a single nonzero term, since a node can have no more than one parent.

For a uniform linear distribution of particles, a cost analysis of the method is

straightforward. Let the cluster hierarchy consist of t balanced binary trees with s leaves in total, so that there are $2(s-t)$ branches. As in previous sections, two scalings of the algorithm parameters are interesting. In both, growth of t penalizes the performance. Instead of adding new trees, the existing trees should grow deeper. If $t = O(1)$, then there are $O(s)$ branches, and for an asymptotic analysis it suffices to assume a single balanced binary tree.

Consider the work required in traversing the source tree in Figure 2.16. The number of nodes in level $\ell+1$ is twice the number of nodes in level ℓ. The number of dag edges leaving any node is bounded by a small constant. Under accuracy scaling, all clusters in the tree have the same cutoff $p = O(\log \epsilon^{-1})$. Since the expansion lengths do not change and the translations are spread uniformly throughout the tree, any given node contributes about the same work as any other node. There is therefore about twice as much work on level $\ell + 1$ as on level ℓ, and the total amount of work in performing all translations is about twice the work required on the bottom level alone.

Following the algorithm description given above, the cost of each step is:

1. N particles $\times \dfrac{O(p) \text{ flops}}{\text{particle}} = O(pN)$ flops
2. $O(s)$ branches $\times \dfrac{O(p \log p) \text{ flops}}{\text{branch}} = O(sp \log p)$ flops
3. $O(s)$ translations $\times \dfrac{O(p \log p) \text{ flops}}{\text{translation}} = O(sp \log p)$ flops
4. $O(s)$ branches $\times \dfrac{O(p \log p) \text{ flops}}{\text{branch}} = O(sp \log p)$ flops
5. N particles $\times \dfrac{O(p) \text{ flops}}{\text{particle}} = O(pN)$ flops
6. N particles $\times \dfrac{O(N/s) \text{ flops}}{\text{particle}} = O(N^2/s)$ flops

The sum of these costs is $O(sp \log p + pN + N^2/s)$ flops. To this sum should be added the expense of generating the data input to the algorithm.

The memory required to store the hierarchical representation of Φ is:

1. Space for $\{D_{ij}\} = O(s)$ matrices $\times \dfrac{O(N^2/s^2) \text{ words}}{\text{matrix}} = O(N^2/s)$ words
2. Space for $\{U_i\} = O(s)$ matrices $\times \dfrac{O(pN/s) \text{ words}}{\text{matrix}} = O(pN)$ words
3. Space for $\{V_i\} = O(s)$ matrices $\times \dfrac{O(pN/s) \text{ words}}{\text{matrix}} = O(pN)$ words
4. Space for $\{\boldsymbol{\lambda}_{ij}\} = O(s)$ vectors $\times \dfrac{O(p) \text{ words}}{\text{vector}} = O(sp)$ words
5. Space for $\{\boldsymbol{\mu}_{ij}\} = O(s)$ vectors $\times \dfrac{O(p) \text{ words}}{\text{vector}} = O(sp)$ words
6. Space for $\{\boldsymbol{\nu}_{ij}\} = O(s)$ vectors $\times \dfrac{O(p) \text{ words}}{\text{vector}} = O(sp)$ words

The computational requirement is $O(1)$ flops per word, plus an additional $O(p \log p)$ flops each for the FFTs needed to form the vectors $\{\boldsymbol{\lambda}_{ij}\}$, $\{\boldsymbol{\mu}_{ij}\}$, and $\{\boldsymbol{\nu}_{ij}\}$. The cost of computing the representation is therefore $O(sp + pN + N^2/s)$ words and $O(sp \log p + pN + N^2/s)$ flops.

The computational complexity of constructing the input data for Algorithm 2.3 is the same as the computational complexity of executing the algorithm. The asymptotic minimum of $O(pN)$ space and time is achieved at $s = O(N/p)$, a fact that was anticipated on page 96. Note that we have neglected the cost of constructing the cluster tree and the interaction list. Suboptimal trees and interaction lists may be constructed quite rapidly.

With accuracy scaling, fast translations are actually unnecessary. Using slow matrix-vector products, each factor of $p \log p$ in the flop count is replaced with p^2. The full translation matrices $\{R_{ij}\}$, $\{T_{ij}\}$, and $\{S_{ij}\}$ are also stored, and each one requires $O(p^2)$ words. So if the Toeplitz structure of the translation matrices is ignored, the algorithm requires $O(sp^2 + pN + N^2/s)$ time and space. Again, choosing $s = O(N/p)$ achieves the minimum complexity $O(pN)$.

With frequency scaling, the analysis changes considerably, because the spectral

expansion lengths are not constant but increase in proportion to cluster diameter. For the clusters of Figure 2.9, the expansion length of a parent cluster is twice the expansion length of each of its children. In traversing the source tree, the amount of work per node increases as we ascend the tree. Since the expansion lengths are doubled while the number of nodes is halved, the work required at level ℓ is nearly equal to the work required at level $\ell + 1$. With $\log_2 s$ levels, the total amount of work in performing all translations is a factor of $\log_2 s$ times the work required on the bottom level alone. Since the expansion cutoff at the leaves is $p = O(1)$ under frequency scaling, with the above choice of s we have $s = O(N/p) = O(N)$, and this argument estimates the complexity at $O(N \log N)$ flops.

A more careful analysis shows that the amount of work per node on level ℓ is slightly more than twice the amount of work per node on level $\ell + 1$. The reduction in the number of nodes in going from level $\ell + 1$ to level ℓ is not quite enough to compensate this increase, and the total amount of work per level increases as the tree is ascended. The work per level grows slowly, however, and the complexity increases only by an additional factor of $\log N$.

Adding together the work for each of $\log_2 s$ levels, starting at the leaves, a total of

$$O(sp \log p) + O(\tfrac{s}{2} \cdot 2p \log 2p) + O(\tfrac{s}{4} \cdot 4p \log 4p) + \\ \cdots + O(1 \cdot sp \log sp) = O\big((sp \log sp)(\log s)\big)$$

operations are required in step 2. The same estimate applies to steps 3 and 4. Since $p = O(1)$, the work required to carry out Algorithm 2.3 is $O(s \log^2 s + N + N^2/s)$ flops.

The lengths of the vectors $\{\boldsymbol{\lambda}_{ij}\}$, $\{\boldsymbol{\mu}_{ij}\}$, and $\{\boldsymbol{\nu}_{ij}\}$ increase as the cluster tree is

ascended, and a similar argument shows that we need to allocate $O(s \log s)$ words and expend $O(s \log^2 s)$ flops to prepare these vectors. To set up Algorithm 2.3, a total of $O(s \log s + N + N^2/s)$ words and $O(s \log^2 s + N + N^2/s)$ flops are required. Again, the latter is identical to the algorithm's cost of execution.

By choosing $s = O(N)$, both time and space requirements are minimized. Table 2.4 summarizes the complexities.

Table 2.4: Hierarchical Multipole Complexity

Scaling Law	s	p	Flops	Memory
Accuracy	$O(N/\log \epsilon^{-1})$	$O(\log \epsilon^{-1})$	$O(N \log \epsilon^{-1})$	$O(N \log \epsilon^{-1})$
Frequency	$O(N)$	$O(1)$	$O(N \log^2 N)$	$O(N \log N)$

All these estimates are for a specific distribution of particles—a random but uniform arrangement on a line segment. The strength of the hierarchical multipole method is that the same estimates hold for a uniform distribution on a rectangle, as in Figure 2.7, or on any sufficiently smooth curve, as in Figure 2.8. A cluster hierarchy for the latter example is shown in Figure 2.17.

2.6 Dipole Interactions

The multipole methods in this chapter have so far been developed only to solve Problems 2.1 and 2.2 for the monopole Helmholtz interaction (2.6). The methods readily generalize, however, to allow derivatives with respect to either $\boldsymbol{x}$ or $\boldsymbol{y}$ in the fundamental solution $\Phi(\boldsymbol{x}, \boldsymbol{y})$.

The treatment of derivatives with respect to $\boldsymbol{y}$ allows the introduction of point dipoles, and point multipoles of any higher order, into the particle system. In Chap-

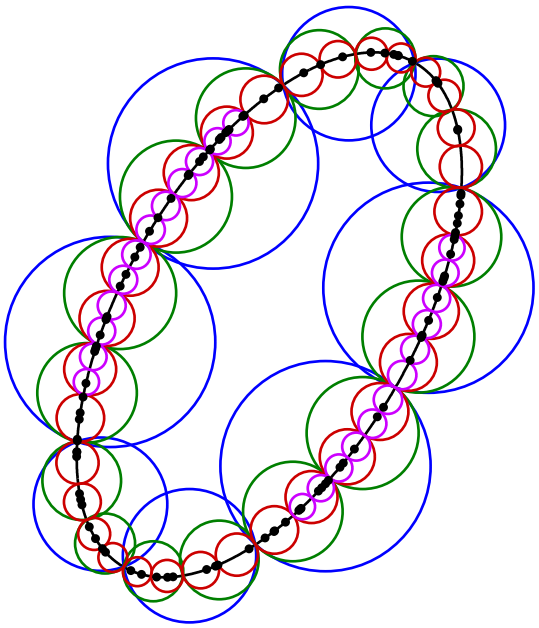


Figure 2.17: A cluster hierarchy for particles distributed uniformly on an ellipse. The cluster disks have been generated automatically for a larger number of particles than shown here. The tree construction code does not require an arc length parametrization of the boundary, and the effect is to generate slightly unbalanced trees of unequal depth. The cluster tree produced by an adaptive quadtree code will have the same character.

ter 3 we discuss a discretization of the EFIE that generates a system of monopoles that lie on the boundary Γ. An analogous discretization of the CSIE generates a particle system that includes both monopoles and dipoles. (The discretization does alter the interaction law over short distances. That modification has little consequence, since multipole does not speed up the computation of short-range interactions anyway.)

The treatment of derivatives with respect to $\boldsymbol{x}$ allows the computation of not only the field but also its derivatives. Particle discretizations of the MFIE and CFIE generate systems of point monopoles, but both the field and its directional derivative must be evaluated at each particle location.

The generalization is possible because derivatives of $\Phi(\boldsymbol{x}, \boldsymbol{y})$ are still radiating solutions of the Helmholtz equation. The same spectral expansions can be used, and the translation operators $\{R_{ij}\}$, $\{T_{ij}\}$, and $\{S_{ij}\}$ are unchanged. For derivatives with respect to $\boldsymbol{x}$, the leaf operators $\{U_i\}$ must be altered. Similarly, the leaf operators $\{V_i\}$ require adjustment if one or more derivatives are taken with respect to $\boldsymbol{y}$.

Consider a collection of N dipoles at the locations $\{\boldsymbol{x}_n\}$ inside a disk with radius α centered on the origin. The dipoles have complex amplitudes $\{\chi_n\}$ and orientations $\{\hat{\boldsymbol{d}}_n\}$. The field generated by the dipoles is

$$u(\boldsymbol{x}) = \sum_{n=1}^{N} \chi_n \hat{\boldsymbol{d}}_n \cdot \nabla_{\boldsymbol{x}_n} \Phi(\boldsymbol{x}, \boldsymbol{x}_n) \quad \text{for} \quad \boldsymbol{x} \in \mathbb{E}^2. \tag{2.89}$$

Following Section 2.2.1, we wish to represent u outside the disk with the infinite series

$$u(\boldsymbol{x}) = \sum_{m=-\infty}^{\infty} a_m H_m(k\|\boldsymbol{x}\|) e^{im\theta(\boldsymbol{x})} \quad \text{for} \quad \|\boldsymbol{x}\| > \beta, \tag{2.90}$$

where $\beta > \alpha$.

With $\|\boldsymbol{x}\| > \|\boldsymbol{y}\|$, Graf's addition theorem gives

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = -\frac{i}{4} \sum_{m=-\infty}^{\infty} J_m(k\|\boldsymbol{y}\|) e^{-im\theta(\boldsymbol{y})} H_m(k\|\boldsymbol{x}\|) e^{im\theta(\boldsymbol{x})}, \tag{2.91}$$

and inserting this into (2.89) reveals an expression for the expansion coefficients,

$$a_m = -\frac{i}{4} \sum_{n=1}^{N} \chi_n \hat{\boldsymbol{d}}_n \cdot \nabla \Big(J_m(k\|\boldsymbol{x}_n\|) e^{-im\theta(\boldsymbol{x}_n)} \Big), \quad m \in \mathbb{Z}. \tag{2.92}$$

The gradient in (2.92) can be simplified using the recurrences

$$\frac{2m}{x} J_m(x) = J_{m+1}(x) + J_{m-1}(x), \tag{2.93a}$$

$$J_m'(x) = \tfrac{1}{2}\big(J_{m-1}(x) - J_{m+1}(x)\big), \tag{2.93b}$$

for the Bessel function and its derivative. The rectangular components of the gra-

dient are

$$\frac{\partial}{\partial x}\Big(J_m(kr)e^{-im\phi}\Big) = \frac{k}{2}\Big(J_{m-1}(kr)e^{-i(m-1)\phi} - J_{m+1}(kr)e^{-i(m+1)\phi}\Big), \tag{2.94a}$$

$$\frac{\partial}{\partial y}\Big(J_m(kr)e^{-im\phi}\Big) = -\frac{ik}{2}\Big(J_{m-1}(kr)e^{-i(m-1)\phi} + J_{m+1}(kr)e^{-i(m+1)\phi}\Big), \tag{2.94b}$$

where both rectangular coordinate (x, y) and polar coordinate (r, ϕ) representations of the same position vector $\boldsymbol{x}$ have been used.

Substituting (2.94) into (2.92), the spectral expansion coefficients are

$$a_m = -\frac{ik}{8}\sum_{n=1}^{N}\chi_n\hat{\boldsymbol{d}}_n \cdot \Big((\hat{\boldsymbol{x}} - i\hat{\boldsymbol{y}})J_{m-1}(k\|\boldsymbol{x}_n\|)e^{-i(m-1)\theta(\boldsymbol{x}_n)} - (\hat{\boldsymbol{x}} + i\hat{\boldsymbol{y}})J_{m+1}(k\|\boldsymbol{x}_n\|)e^{-i(m+1)\theta(\boldsymbol{x}_n)}\Big). \tag{2.95}$$

After truncating the series, discarding all terms with index $|m| > p$, (2.95) is conveniently expressed as a block matrix-vector product,

$$\boldsymbol{a} = \begin{bmatrix} V_- & V_+ \end{bmatrix}\begin{bmatrix} \boldsymbol{\chi}_- \\ \boldsymbol{\chi}_+ \end{bmatrix}, \tag{2.96}$$

where $\boldsymbol{\chi}_\pm := \big[\chi_n\hat{\boldsymbol{d}}_n \cdot (\hat{\boldsymbol{x}} \pm i\hat{\boldsymbol{y}})\big]$ and the matrix elements of $V_\pm \in \mathbb{C}^{(2p+1)\times N}$ are

$$(V_\pm)_{mn} = \pm\frac{ik}{8}J_{m\pm1}(k\|\boldsymbol{x}_n\|)e^{-i(m\pm1)\theta(\boldsymbol{x}_n)}, \tag{2.97}$$

where the row index m assumes successive values $-p, \ldots, p$.

Starting from (2.91) and using the recurrence relations (2.93), similar formulas for the leaf operator V can be derived for quadrupoles and multipoles of any higher order.

The same procedure will produce formulas for the leaf operator U if field derivatives are needed.

2.7 Multipole with Far Fields

The development in this chapter deviates from standard practice and terminology in several respects. In this section, I highlight the following difference: While I use the spectral expansion coefficient vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ as the intermediate variables, the common practice instead uses samples of far fields.

Using polar coordinates (r, ϕ), let us implicitly define[5] the far field pattern $u_\infty : [0, 2\pi] \to \mathbb{C}$ of the radiating field u by

$$\lim_{r\to\infty} u(r, \phi) = u_\infty(\phi)\frac{e^{ikr}}{\sqrt{r}}. \tag{2.98}$$

At infinity, all radiating fields are nonuniform cylindrical waves. Any point in the plane can serve as the center of the cylindrical wave. The far field pattern expresses the angular variation of complex amplitude if the wave center is taken to be the origin of coordinates.

Reproducing the formula (2.73) for the transformation $\{a_m\} \mapsto \{a'_m\}$ of exterior expansion coefficients under a coordinate system displacement $\boldsymbol{d}$,

$$a'_m = \sum_{n=-\infty}^{\infty} a_n J_{n-m}(k\|\boldsymbol{d}\|)e^{i(n-m)\theta(\boldsymbol{d})}, \quad m \in \mathbb{Z}, \tag{2.99}$$

we ought to immediately recognize this as the discrete convolution of two infinite sequences $\{a_m\}$ and $\{j_m\}$, after the identification

$$j_m := J_{-m}(k\|\boldsymbol{d}\|)e^{-im\theta(\boldsymbol{d})}, \quad m \in \mathbb{Z}, \tag{2.100}$$

is made.

The approach taken in Section 2.5.1 is to truncate the sequences $\{a_m\}$ and $\{a'_m\}$

[5]There does not seem to be a standard definition of far field pattern, but all definitions differ only by a constant multiplier. Another definition [15] is $u_\infty(\phi) := \sqrt{i\pi k/2}\lim_{r\to\infty}\sqrt{r}e^{-ikr}u(r, \phi)$.

into the finite vectors $\boldsymbol{a}$ and $\boldsymbol{a'}$. Because (2.99) is a convolution, the translation matrix S is Toeplitz. Because S is Toeplitz, a fast algorithm to compute the matrix-vector product $\boldsymbol{a'} = S\boldsymbol{a}$ is available.

An alternative to that algebraic approach is to perform further analysis of (2.99). In particular, Fourier analysis is well-known to simplify convolutions. Define the Fourier transform $\widetilde{a} : [0, 2\pi] \to \mathbb{C}$ of the infinite sequence $\{a_n\}$ as

$$\widetilde{a}(\phi) := \sum_{n=-\infty}^{\infty} a_n e^{-in\phi}. \tag{2.101}$$

Under this Fourier transform, (2.99) becomes the pointwise multiplication

$$\widetilde{a'}(\phi) = \widetilde{a}(\phi)\widetilde{\jmath}(\phi). \tag{2.102}$$

Moreover, the Fourier transform of (2.100) is

$$\widetilde{\jmath}(\phi) = e^{ik\|\boldsymbol{d}\|\sin(\theta(\boldsymbol{d})+\phi)}, \tag{2.103}$$

a simple result obtained from the generating function for the Bessel functions J_n [1, §9.1.41].

The transform $\widetilde{a}$ is essentially the far field pattern of the exterior wave expansion

$$u(r, \phi) = \sum_{n=-\infty}^{\infty} a_n H_n(kr) e^{in\phi}. \tag{2.104}$$

After substituting into (2.104) the large-argument asymptotic form [1, §9.2.3]

$$\lim_{|z|\to\infty} H_n(z) = \sqrt{\frac{2}{i\pi z}} e^{i(z-n\pi/2)}, \tag{2.105}$$

a comparison with (2.98) shows that the far field pattern is

$$u_\infty(\phi) = \sqrt{\frac{2}{\pi k}} e^{-i\pi/4} \widetilde{a}\Big(\frac{\pi}{2} - \phi\Big). \tag{2.106}$$

The simplicity of (2.102) and (2.103) forms the motivation to work with the function $\widetilde{a}(\phi)$ instead of the infinite sequence $\{a_n\}$.

Like (2.99), the other translations $\{a_m\} \mapsto \{b_m\}$ and $\{b_m\} \mapsto \{b'_m\}$ are also convolutions, and the Fourier transform (2.101) will formally diagonalize them as well.

Unlike $\widetilde{a}$, the Fourier transform of $\{b_m\}$ cannot be interpreted as a far field, because the interior expansion (2.28) is not a radiating field. It represents a field only in a bounded source-free region, and if that field is smoothly extended to all of $\mathbb{E}^2$, then at infinity it is the sum of two nonuniform cylindrical waves. One of those waves, $u_+(\phi)r^{-1/2}e^{ikr}$, is receding from the origin. The other, $u_-(\phi)r^{-1/2}e^{-ikr}$, is traveling toward the origin, and is not present in a radiating field.

The absence of a familiar physical interpretation for $\widetilde{b}$ is not much cause for concern. There is, however, the following serious defect: $\widetilde{b}$ *diverges* for general charge distributions. So too does the Fourier transform of the convolution factor

$$h_m := H_{-m}(k\|\boldsymbol{d}\|)e^{-im\theta(\boldsymbol{d})}, \quad m \in \mathbb{Z}, \tag{2.107}$$

that translates an exterior expansion to an interior expansion. The Fourier analysis cannot be sustained unless the sequences $\{b_m\}$ and $\{h_m\}$ are either truncated or else weighted for large $|m|$ by window sequences that decay sufficiently rapidly.

If the coefficient sequences have to be truncated anyway, then I think that the algebraic approach is easier to grasp.

Chapter 3

Iterative Multipole for Inverse Particle Problems

The main objective of this chapter is to apply the fast multipole method to the following problem:

Problem 3.1 (Inverse Helmholtz Interaction) *A collection of N particles with charge amplitudes $\{q_n\}$ generates a field $u(\boldsymbol{x})$. At the location $\boldsymbol{x}_m$ of particle m, the field is*

$$u_m = \sum_{n \neq m} \Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) q_n, \tag{3.1}$$

where the Helmholtz interaction law is $\Phi(\boldsymbol{x}, \boldsymbol{y}) = -\frac{i}{4} H_0(k\|\boldsymbol{x} - \boldsymbol{y}\|)$. Given the positions $\{\boldsymbol{x}_m\}$ and the field samples $\{u_m\}$, determine the set of charges $\{q_n\}$ that produced the field.

This is the *inverse* problem for the Helmholtz interaction. The direct problem (Problem 2.1) is to compute the matrix-vector product $\boldsymbol{u} = \Phi \boldsymbol{q}$, given a charge vector $\boldsymbol{q}$. In the inverse problem it is $\boldsymbol{u}$ rather than $\boldsymbol{q}$ that is known. Assuming Φ is

nonsingular, $\boldsymbol{q} = \Phi^{-1}\boldsymbol{u}$ is the unique solution.

In Chapter 2 we developed three multipole algorithms for the fast computation of $\Phi\boldsymbol{q}$. With an *iterative* solver we can use the same algorithms to compute $\Phi^{-1}\boldsymbol{u}$.

Although the diagonal entries of Φ are zero, its condition number $\kappa = \|\Phi\|_2\|\Phi^{-1}\|_2$ is usually modest. A typical rate of growth is $\kappa = O(N)$ as $N \to \infty$. That slow growth makes Problem 3.1 numerically well posed, in contrast to many other interesting inverse problems [93].

The good behavior of this inverse problem is a beneficial side effect of the singularity in $\Phi(\boldsymbol{x}, \boldsymbol{y})$. If $\Phi(\boldsymbol{x}, \boldsymbol{y})$ were smooth at $\boldsymbol{x} = \boldsymbol{y}$ instead of being unbounded there, then the inverse problem would be poorly conditioned. Sometimes singularities make a problem easier to solve!

Interest in inverse particle problems is not confined to the Helmholtz interaction. A suitable discretization, such as Nyström's method, can convert any integral equation into an inverse particle problem. The particle interaction law is influenced by both the kernel function and the discretization scheme.

If we are to utilize a fast multipole method tailored to the kernel, then the discretization must not be allowed to spoil things by creating an interaction law that differs too much from that kernel. In the next section we consider how that can be accomplished for a scattering integral equation.

3.1 Particle Discretizations

In Chapter 1 we transformed the elliptic PDE of an obstacle scattering problem into an integral equation on the obstacle boundary. A whole collection of integral

equations can be constructed, but the simplest in form is the EFIE, an integral equation of the first kind,

$$\int_{\Gamma} \Phi(\boldsymbol{x}, \boldsymbol{y}) \sigma(\boldsymbol{y}) \, d\Gamma(\boldsymbol{y}) = (ik\eta)^{-1} u_{\text{inc}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Gamma, \tag{3.2}$$

where the function σ on the boundary must be found.

A spectral discretization of the EFIE was developed in Section 1.6. That discretization does not, however, produce algebraic equations that look like (3.1). The spectral product rule splits the kernel apart into several pieces. Unfortunately, the multipole algorithms of Chapter 2 cannot deal with those pieces. The spectral expansions and translation operators have all been derived from the Helmholtz interaction, rendering the algorithms quite specific to the kernel $\Phi(\boldsymbol{x}, \boldsymbol{y})$.

In this section we construct another high-order discretization that does not so badly disrupt the kernel. The correspondence between our discretization and Problem 3.1 is not exact. The difference is that the particles, which lie at specified points on the boundary Γ, have short-range interactions that obey a different law than the Helmholtz interaction $\Phi(\boldsymbol{x}, \boldsymbol{y})$. At long ranges, though, the Helmholtz interaction law is preserved.

After the discretization, (3.2) will assume the form

$$(\Sigma + \Phi)\boldsymbol{q} = \boldsymbol{u}, \tag{3.3}$$

where Σ is a sparse matrix that corrects the long-range interaction law over small distances $\|\boldsymbol{x} - \boldsymbol{y}\|$. Bounded self-interactions are now permissible, and those terms lie on the main diagonal of Σ. The product $\Sigma\boldsymbol{q}$ can be computed with sparse matrix arithmetic, and the product $\Phi\boldsymbol{q}$ can be computed with a multipole algorithm.

It is possible—for instance by applying a trapezoidal rule to (3.2) but dropping

the node at the kernel singularity—to eliminate Σ from (3.3), yielding Problem 2.1 exactly. But such a discretization has poor accuracy, and the computed solution will converge to the exact solution very slowly as N is increased. We are interested in singular quadrature rules that exhibit higher performance. The perturbation Σ is tolerable as long as it is sufficiently sparse.

Even if $\Sigma = 0$, recall that the fast multipole method already uses block sparse matrix arithmetic to compute the interactions between particles in leaf clusters that are not well separated. The high-order discretization constructed here will only change some of those interactions. Therefore, at least after it has been filled, the nonzero Σ does not add to the workload.

3.1.1 Fejér Rules

In Section 1.6 the parametrization $\boldsymbol{x} = \boldsymbol{\gamma}(s)$ and $\boldsymbol{y} = \boldsymbol{\gamma}(t)$ is substituted into (3.2), and after relabeling functions using the definitions

$$\varphi(t) := \sigma\big(\boldsymbol{\gamma}(t)\big), \tag{3.4a}$$

$$f(s) := (ik\eta)^{-1} u_{\text{inc}}\big(\boldsymbol{\gamma}(s)\big), \tag{3.4b}$$

$$K(s,t) := \Phi\big(\boldsymbol{\gamma}(s), \boldsymbol{\gamma}(t)\big) \|\boldsymbol{\gamma}'(t)\|, \tag{3.4c}$$

the EFIE is recast into the standard form

$$\int_0^1 K(s,t)\varphi(t)\,dt = f(s), \quad s \in [0,1]. \tag{3.5}$$

Then the kernel is broken into more manageable pieces. The quadrature rules, both singular and nonsingular, are global rules that approximate every smooth piece of the integrand with a trigonometric function of N harmonic tones. That trigonometric approximation is easiest if the integrand is sampled at quadrature nodes distributed

uniformly in $[0, 1)$.

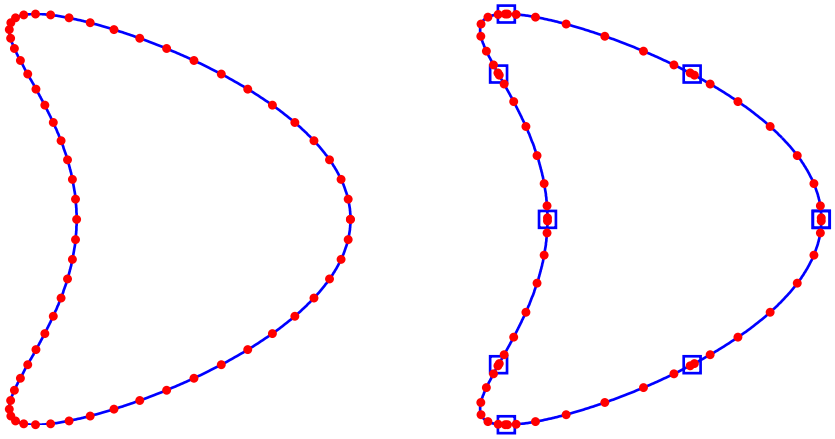
The new quadrature rules are local rules that partition the boundary into a mesh of M arcs. On each arc, any smooth piece of the integrand is approximated by a polynomial of degree $P - 1$. The quadrature nodes are not equally spaced. On each arc, they are an affine transformation of the zeros of the Chebyshev polynomial $T_P(z) := \cos(P \cos^{-1} z)$. The P zeros of T_P are, in increasing order,

$$z_n = -\cos \frac{\pi}{P}\Big(n - \frac{1}{2}\Big), \quad 1 \le n \le P. \tag{3.6}$$

If the arc covers the parametric interval $t \in [a, b]$, then the affine transformation is

$$\zeta(z) := \frac{b-a}{2} z + \frac{a+b}{2}, \quad z \in [-1, 1], \tag{3.7}$$

which simply maps $[-1, 1]$ to $[a, b]$. The quadrature nodes in $[a, b]$ are $t_n = \zeta(z_n)$. Figure 3.1 compares the node distributions of the global and local rules.



(a) (b)

Figure 3.1: In rectangular coordinates, this boundary Γ has parametrization $\boldsymbol{\gamma}(t) = (\cos 2\pi t + 0.65 \cos 4\pi t - 0.65,\ 1.5 \sin 2\pi t)$. (a) Grid for the 64-point spectral rule, with nodes at $t_i = (i-1)/64$, $1 \le i \le 64$. (b) Grid for 8 panels of 8-point Fejér rules. The break points, at the center of the small squares, lie at $t_i = (i-1)/8$, $1 \le i \le 9$.

If the interval $[0,1]$ is partitioned into M subintervals by the points $0 = a_1 < a_2 < \cdots < a_M < a_{M+1} = 1$, then the integral in (3.5) is split into M pieces,

$$\int_0^1 K(s,t)\varphi(t)\,dt = \sum_{m=1}^{M} \int_{a_m}^{a_{m+1}} K(s,t)\varphi(t)\,dt, \tag{3.8}$$

and we may apply a different quadrature rule to each piece. The quadrature weights may be a function of s, but the locations of the quadrature nodes are not allowed to change as s varies.

If the boundary is analytic, then $K(s,t)$ is an analytic function of t on any subinterval that does not contain s. On those subintervals, the kernel does not need to be processed further, and a nonsingular quadrature rule can be applied. The remainder of this section is devoted to that nonsingular rule.

A quadrature rule on $[-1,1]$ using the points (3.6) was studied by Fejér [35], who also considered a rule with nodes at the P local extrema of T_{P-1}. The latter rule is more well-known as a Clenshaw–Curtis rule. Its main advantage is that the local extrema of T_P are also local extrema of T_{2P-1}. An adaptive integration routine that applies a sequence of Clenshaw–Curtis rules with $\{P,\, 2P{-}1,\, 4P{-}3,\, \dots\}$ points can at each step reuse all the integrand evaluations from previous steps. This node recycling offers no advantage, however, for the solution of an integral equation.

The Clenshaw–Curtis rule is closed: It includes nodes at the end points $z = -1$ and $z = 1$. I prefer to use open Fejér rules because then adjacent boundary arcs do not share a node at their end points. Node sharing adds a bookkeeping overhead to the code.

Furthermore, if the boundary Γ has any corners, then I insist that those corners always lie at an arc end point. The open rules then conveniently avoid the singularity in the integrand at the corner. Avoiding the singularity, of course, leads to a low

order of solution accuracy. Rather than trying to incorporate the solution singularity into a high-order singular quadrature rule, I follow the widespread practice of refining the boundary mesh near corners.

The main attraction of a closed rule for the solution of an integral equation by piecewise collocation is that it gives a continuous approximate solution. Using open rules, the approximate solution has jump discontinuities where the arcs join. Those discontinuities are not particularly troubling, however, if the discretization has a high order of accuracy. The magnitude of each jump rapidly falls to $O(\texttt{eps})$, and they are not noticeable in graphs of the computed solution except when P is small.

With a proper choice of basis, the method of undetermined coefficients is a numerically stable way to compute the weights $\{\omega_n\}$ of the Fejér rule,

$$\int_{-1}^{1} g(z)\,dz \approx \sum_{n=1}^{P} \omega_n g(z_n). \tag{3.9}$$

Forcing the rule to be exact for each member of the basis $\{T_0(z), T_1(z), \ldots, T_{P-1}(z)\}$ for the polynomial space $\mathbb{P}_{P-1}[-1,1]$, a linear system

$$\sum_{n=1}^{P} \omega_n \cos\Big(\frac{\pi}{P}(m-1)(n-\tfrac{1}{2})\Big) = (-1)^{m-1} \int_{-1}^{1} T_{m-1}(z)\,dz, \quad 1 \le m \le P, \tag{3.10}$$

is generated for the unknown weights. In vector form, this is $C\boldsymbol{\omega} = \boldsymbol{b}$, where C is a P-point discrete cosine transform (DCT) matrix and the elements of the right-hand side vector $\boldsymbol{b}$ are

$$\begin{aligned} b_m &:= (-1)^{m-1} \int_{-1}^{1} T_{m-1}(z)\,dz \\ &= \begin{cases} \dfrac{1}{m} - \dfrac{1}{m-2} & \text{if } m \text{ is odd,} \\ 0 & \text{if } m \text{ is even.} \end{cases} \end{aligned} \tag{3.11}$$

The inverse of C is $C^{-1} = (1/P)\,\mathrm{diag}(1, 2, \ldots, 2)C^T$, so the weight vector $\boldsymbol{\omega}$ is

$$\boldsymbol{\omega} = (1/P)\,\mathrm{diag}(1, 2, \ldots, 2)C^T\boldsymbol{b}. \tag{3.12}$$

Using a fast DCT algorithm, this can be computed in $O(P \log P)$ flops.

In exact arithmetic, the weights have the even symmetry $\omega_n = \omega_{P+1-n}$. That property can also be applied as a constraint to (3.10), shrinking the linear system by a factor of 2. A fast DCT can be applied to the smaller system as well.

After the change of variable $t = \zeta(z)$ given by (3.7), the integral of any smooth function g on the interval $[a, b]$ can be expressed as

$$\int_a^b g(t)\,dt = \frac{b-a}{2}\int_{-1}^{1} g\big(\zeta(z)\big)\,dz, \tag{3.13}$$

so the weights $\{\omega_n\}$ are scaled by $(b-a)/2$ to give the rule

$$\int_a^b g(t)\,dt \approx \frac{b-a}{2}\sum_{n=1}^{P}\omega_n g(t_n), \tag{3.14}$$

with node locations $t_n = \zeta(z_n)$.

Concatenating the Fejér rules for each arc, the weights are collected into the vector $\boldsymbol{\xi}$ of length $N = MP$,

$$\boldsymbol{\xi}^T := \left[\frac{a_2 - a_1}{2}\boldsymbol{\omega}^T, \ldots, \frac{a_{M+1} - a_M}{2}\boldsymbol{\omega}^T\right]. \tag{3.15}$$

The nodes are collected into the vector $\boldsymbol{\tau}$,

$$\boldsymbol{\tau}^T := \left[\frac{a_2 - a_1}{2}\boldsymbol{z}^T + \frac{a_1 + a_2}{2}, \ldots, \frac{a_{M+1} - a_M}{2}\boldsymbol{z}^T + \frac{a_M + a_{M+1}}{2}\right], \tag{3.16}$$

where $\boldsymbol{z}^T := \big[z_1, \ldots, z_P\big]$ is the vector of Chebyshev polynomial zeros (3.6).

We can now exhibit the product $\boldsymbol{\Phi q}$ in (3.3). Particle n is located at $\boldsymbol{x}_n = \boldsymbol{\gamma}(\tau_n)$ and carries the charge $q_n = \xi_n\|\boldsymbol{\gamma}'(\tau_n)\|\varphi(\tau_n)$. The function φ is the unknown

continuous charge density, and the weight ξ_n and metric coefficient $\|\boldsymbol{\gamma}'(\tau_n)\|$ are absorbed into the discrete charge q_n. The charge and field vectors of the particle discretization are

$$\boldsymbol{q} = \big[\xi_n \|\boldsymbol{\gamma}'(\tau_n)\| \varphi(\tau_n)\big], \tag{3.17a}$$

$$\boldsymbol{u} = (ik\eta)^{-1}\big[u_{\text{inc}}(\boldsymbol{\gamma}(\tau_n))\big]. \tag{3.17b}$$

Next, we construct a sparse matrix Σ of short-range interactions that achieves a high order of accuracy.

3.1.2 Product Fejér Rules

Now we return to the partitioned integral (3.8) to consider the numerical treatment of the arc containing the singularity at $t = s$. Since we have discretized each subintegral with an open quadrature rule, after collocation s cannot lie at any of the subinterval end points $\{a_m\}$. Therefore, for each of the N equations in the collocation system, precisely one of the M subintegrals in (3.8) has a singular kernel.

We begin the treatment of the kernel exactly as in Section 1.6, by splitting the kernel apart to expose the singularity in its most elementary form. The Hankel function H_0 is the source of the singularity in $K(s,t)$. That function can be expressed as

$$H_0(z) = (\log z) A_0(z^2) + B_0(z^2), \tag{3.18}$$

where A_0 and B_0 are entire functions.

Substitution of (3.18) into the singular subintegral splits it into two pieces. The piece containing B_0 is smooth, and is treated with the standard Fejér rule of the previous section. The piece containing A_0 has a logarithmic singularity, and we are

faced with designing a quadrature rule to approximate the integral

$$(\mathcal{L}h)(s) := \int_a^b \log(k\|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|)\, h(t)\, dt \tag{3.19}$$

for an arbitrary smooth function h. The difference between (3.19) and its counterpart (1.96) is that here the domain of integration is $[a, b] \subset [0, 1]$, and the integrand does not have a smooth extension with period $b - a$.

Continuing to parallel the development in Section 1.6, we split the kernel again because we cannot afford to construct a different product rule for each vector function $\boldsymbol{\gamma}$ in (3.19). The global splitting (1.104) that worked in Section 1.6 made use of a product rule developed for a circular boundary. Using a circle makes no sense—and does not work—for the local product rule here. Instead, I choose to split the kernel to expose the deviation of the boundary arc from a straight line segment.

The relationship of the line segment to the boundary Γ is illustrated in Figure 3.2. For the global spectral rule, a single circle informed the splitting for all s. The geometry underpinning the local splitting here changes with s. The segment is cut from the line tangent to Γ at the singularity. It is the Taylor approximation

$$\hat{\boldsymbol{\gamma}}(t) = \boldsymbol{\gamma}(s) + (t - s)\boldsymbol{\gamma}'(s), \quad t \in [a, b], \tag{3.20}$$

so $\|\boldsymbol{\gamma}(s) - \hat{\boldsymbol{\gamma}}(t)\| = O\big((s - t)^2\big)$. The right end point of the segment extends a distance $(b - s)\|\boldsymbol{\gamma}'(s)\|$ from the point of tangency, and the left end point is a distance $(s - a)\|\boldsymbol{\gamma}'(s)\|$ away.

The distance between two points on the tangent line is

$$\begin{aligned} \widehat{R}(s, t) &:= \|\hat{\boldsymbol{\gamma}}(s) - \hat{\boldsymbol{\gamma}}(t)\| \\ &= \|\boldsymbol{\gamma}'(s)\|\, |s - t|. \end{aligned} \tag{3.21}$$

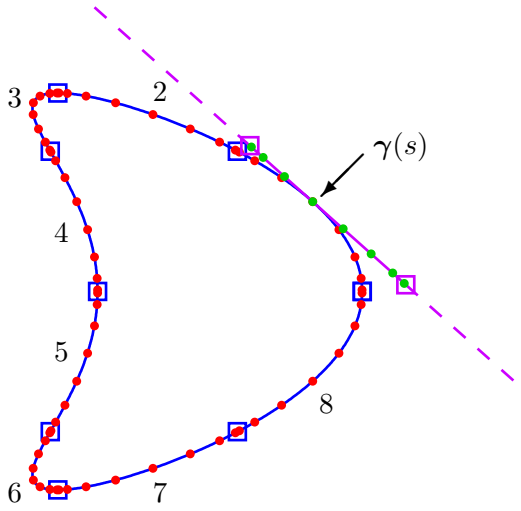


Figure 3.2: When $s = \tau_5$, arc 1 contains the singularity $\boldsymbol{\gamma}(s)$. The singular quadrature rule for that arc is a simple modification of a product Fejér rule for the line segment tangent to Γ at the singularity. On arcs 2–8, the integrand is smooth and the nonsingular rule of Section 3.1.1 is applied.

Inserting this into (3.19), the singular product rule for the tangent line segment is

$$\int_a^b \log(k\|\boldsymbol{\gamma}'(s)\|\,|s-t|)\,h(t)\,dt \approx \sum_{n=1}^{P} w_n(s)h(t_n). \tag{3.22}$$

The method of collocation requires the weights $\{w_n(s)\}$ for each $s \in \{t_m\}$, and those weights form a $P \times P$ matrix W with elements $w_{mn} = w_n(t_m)$.

The left-hand side of (3.22) still contains parameters that will only be known at run time. The weights must have a simple dependence on the end points a and b and on the function $k\|\boldsymbol{\gamma}'(s)\|$. To verify that, let $s = t_m$ in (3.22), and use the affine transformation (3.7) to map the interval $[a, b]$ to the standard domain $[-1, 1]$ of the Chebyshev polynomials. Under that transformation, the left-hand side of (3.22)

becomes

$$\int_a^b \log(k\|\boldsymbol{\gamma}'(t_m)\|\,|t_m - t|)\,h(t)\,dt = \frac{b-a}{2}\int_{-1}^{1} \log|z_m - z|\,h\big(\zeta(z)\big)\,dz + \frac{b-a}{2}\alpha_m \int_{-1}^{1} h\big(\zeta(z)\big)\,dz, \quad (3.23)$$

where α_m is defined by

$$\alpha_m := \log\Big(k\frac{b-a}{2}\|\boldsymbol{\gamma}'(t_m)\|\Big). \quad (3.24)$$

Now if we compute the weight matrix $\Upsilon = \big[\upsilon_n(z_m)\big]$ for the product Fejér rule

$$\int_{-1}^{1} \log|y - z|\,g(z)\,dz \approx \sum_{n=1}^{P} \upsilon_n(y)g(z_n), \quad (3.25)$$

at the singularity locations $y \in \{z_m\}$, then the weight matrix W is

$$W = \frac{b-a}{2}\Big(\Upsilon + \boldsymbol{\alpha}\boldsymbol{\omega}^T\Big), \quad (3.26)$$

where $\boldsymbol{\omega}$ is the vector of standard Fejér weights (3.12) and $\boldsymbol{\alpha} = \big[\alpha_m\big]$. Judging from (3.26), once we have obtained Υ, the weights of the more general rule (3.22) follow easily.

The weights $\{\upsilon_n(z_m)\}$ can be determined by the same procedure used to find the weights $\{\omega_n\}$ in Section 3.1.1. If the rule is forced to be exact for each member of the basis $\{T_0(z), T_1(z), \ldots, T_{P-1}(z)\}$ of $\mathbb{P}_{P-1}[-1, 1]$, then

$$\Upsilon^T = (1/P)\,\mathrm{diag}(1, 2, \ldots, 2)C^T B, \quad (3.27)$$

where C is the DCT matrix and B is a $P \times P$ matrix with elements

$$b_{mn} := \int_{-1}^{1} \log|z_n - z|\,T_{m-1}(z)\,dz. \quad (3.28)$$

Given B and a fast DCT, Υ can be constructed in $O(P^2 \log P)$ flops.

Unlike the situation in Section 1.6, however, the integrals (3.28) apparently do not simplify. I evaluate them with an adaptive integration routine. Since MATLAB does not supply one, I wrote a routine designed to efficiently handle unbounded integrands. Still, the time required to fill B is much greater than the time required for P DCTs.

Since the weights Υ do not depend on Γ, they can be computed in advance and stored on disk. I have done this for various values of P, but I most frequently use the rules for $P = 16$ and $P = 32$. Note that restricting ourselves to only those values of P stored on disk is not a significant limitation, because we can still increase $N = MP$ by dividing the boundary into a finer mesh. There are no restrictions on the number of arcs M.

I have computed the weights Υ for rules as large as $P = 288$. Even at that high order, $\|\Upsilon\|_2 \approx 2.5$, so the weights are well-behaved. In fact, Sloan and Smith [124] proved that a large class of product Fejér rules, of which rule (3.25) is a member, are numerically stable and convergent.

Now we apply this product rule to the integral (3.19) for an arbitrary smooth boundary curve. Split the kernel like

$$\begin{aligned} \log kR &= \log\left(k\widehat{R} \times \frac{R}{\widehat{R}}\right) \\ &= \log k\widehat{R} + \log \frac{R}{\widehat{R}}, \end{aligned} \tag{3.29}$$

which exactly follows (1.104). The second term is smooth for the same reasons listed in Section 1.6, and it is treated by a standard Fejér rule. The first term is treated by the product Fejér rule (3.22).

Let $Z = \Sigma + \Phi$ be the sum of the short-range and long-range interaction matrices, so that the discretization of the EFIE produces the system $Z\boldsymbol{q} = \boldsymbol{u}$. The charge

vector $\boldsymbol{q}$ was specified in (3.17a), and the field vector $\boldsymbol{u}$ in (3.17b).

Since the boundary Γ is partitioned into M arcs, the coefficient matrix Z inherits an $M \times M$ block partition. Each block Z_{mn} is a square matrix of order P. The charge vector has a conforming partition $\boldsymbol{q}^T = \big[\boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_M^T\big]$. The same partition is applied to the quadrature weight vector $\boldsymbol{\xi}$ of (3.15), the quadrature node vector $\boldsymbol{\tau}$ of (3.16), and the vector $\boldsymbol{\alpha}$ of (3.24).

The discretization of this section fills the diagonal blocks $\{Z_{mm}\}$. The structure of those blocks is

$$Z_{mm} = \big(W_m \operatorname{diag}\big((\boldsymbol{\xi}_m)_1^{-1}, \ldots, (\boldsymbol{\xi}_m)_P^{-1}\big) + L_m\big) \odot A_m + B_m, \tag{3.30}$$

where W_m is the product-rule weight matrix W of (3.26), after making the substitutions $a \to a_m$, $b \to a_{m+1}$, and $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\alpha}_m$. The matrices A_m, B_m, and L_m have elements

$$(A_m)_{pq} = -\tfrac{i}{4} A_0\big(k^2 \big\|\boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_p\big) - \boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_q\big)\big\|^2\big), \tag{3.31a}$$

$$(B_m)_{pq} = -\tfrac{i}{4} B_0\big(k^2 \big\|\boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_p\big) - \boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_q\big)\big\|^2\big), \tag{3.31b}$$

$$(L_m)_{pq} = \begin{cases} \log \dfrac{\big\|\boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_p\big) - \boldsymbol{\gamma}\big((\boldsymbol{\tau}_m)_q\big)\big\|}{\big\|\boldsymbol{\gamma}'\big((\boldsymbol{\tau}_m)_p\big)\big\| \, \big|(\boldsymbol{\tau}_m)_p - (\boldsymbol{\tau}_m)_q\big|} & \text{if } p \neq q, \\ 0 & \text{if } p = q, \end{cases} \tag{3.31c}$$

for $1 \leq p \leq P$ and $1 \leq q \leq P$.

This discretization is a *block spectral* discretization. If the boundary is an analytic Jordan curve, the discretization exhibits spectral accuracy if N is increased by adding more grid points to each arc. For some positive constant $\zeta < 1$, the error is $\epsilon = O(\zeta^P)$ as $P \to \infty$. If, on the other hand, N is increased by refining the arc partition, then the discretization has an accuracy of order P, so $\epsilon = O(M^{-P})$ as $M \to \infty$.

3.1.3 Product Rules for Nearly Singular Kernels

The discretization remains deficient in one respect. We have used standard Fejér rules to integrate $K(s,t)\varphi(t)$ on an interval $[a,b]$ whenever s lies outside that interval. If, however, $s = b + \delta$ for a tiny positive number δ, then the integrand is nearly singular. It will display rapid variation at the end point b, and it will be poorly approximated by a low-degree polynomial there.

For a fixed s arbitrarily near the interval $[a,b]$, the P-point nonsingular rule does exhibit spectral convergence, so that a positive constant $\zeta < 1$ exists such that $\epsilon = O(\zeta^P)$ as $P \to \infty$. The problem is that the exponential decay might not be fast enough. As s approaches either a or b, the constant ζ approaches 1.

The solution to this problem is to use a product rule whenever s lies inside an interval $(a-\rho, b+\rho)$ that contains $[a,b]$. The product rules for nearly singular integrands are constructed in exactly the same way as the product rules for singular integrands. The kernel is split, using the line tangent to the boundary at s as a guide.

Let $s \in (b, b+\rho)$, and let $[b,c]$ be the subinterval in the partition that follows the subinterval $[a,b]$. The weights for the nearly singular product rule applied at this value of s depend on the ratio $(c-b)/(b-a)$. Since we can only compute and store a finite number of rules, we must constrain the partition so that this ratio always assumes one of a small set of values. In my code, I require that $(c-b)/(b-a) \in \{\frac{1}{2}, 1, 2\}$. For each value of P, I compute and store one singular product rule and three nearly singular product rules.

The constraint on how Γ may be partitioned appears to be a burden. If the constraint were that adjacent arcs of the partition have arc length ratios in the set $\{\frac{1}{2}, 1, 2\}$, then it would in fact be intolerable. An arc length parametrization of Γ

would have to be computed by solving a nonlinear ordinary differential equation. But the parametrization here is not required to be an arc length parametrization. It can be any convenient, nonsingular parametrization whatsoever, and ensuring that adjacent arcs have parametric length ratios in the set $\{\frac{1}{2}, 1, 2\}$ is easy.

In my code, I also always allocate P Chebyshev points to each arc of the boundary partition. If adjacent arcs were allowed to have unequal point allocations, then more nearly singular quadrature rules would have to be computed. Again, the constraint that each leaf arc have the same number of nodes is a mild one.

Now let us turn to the proper selection of ρ. Given a desired convergence parameter ζ, we can appeal to the theory of approximation for analytic functions [55] [135] to choose the smallest ρ necessary to ensure that $\epsilon = O(\zeta^P)$ as $P \to \infty$.

Let a function $g : \mathbb{C} \to \mathbb{C}$ be analytic on a domain R of the complex plane that contains the cut $C := \{z \in \mathbb{C} : -1 \leq \operatorname{Re} z \leq 1, \operatorname{Im} z = 0\}$. If $\hat{g}$ is the polynomial of degree P that interpolates g at the P real zeros of the Chebyshev polynomial T_P, then on C the approximation $\hat{g}$ converges to g with spectral accuracy as $P \to \infty$. In particular, there exists a positive constant $\zeta < 1$ such that

$$\max_{z \in C} |g(z) - \hat{g}(z)| = O(\zeta^P) \quad \text{as} \quad P \to \infty. \tag{3.32}$$

Approximations that interpolate g at certain other sets of points, such as the Gauss–Legendre points or the extrema of Chebyshev polynomials, also have this property.

The rate parameter ζ depends on the shape and size of the domain R. Let E be the largest ellipse with foci at $z = \pm 1$ that fits into $\overline{R}$. Then ζ^{-1} equals the average of the major and minor diameters of E. For example, the ellipse $E_{1/2}$ with foci at $z = \pm 1$ and eccentricity $4/5$ has a major diameter of $5/2$ and a minor diameter of $3/2$, and the Chebyshev interpolation of any function analytic within $E_{1/2}$ will

converge with a spectral rate parameter of $\zeta \leq 1/2$.

In Section 2.2.1 we selected a separation parameter to ensure that the multipole expansions converge at a rate $o(2^{-|p|})$ as the series cutoff p increases. Although p and P are not connected, we may as well also insist that $\zeta \leq 1/2$, so that the discretization error decays in the worst case like $O(2^{-P})$.

For the nonsingular quadrature rule of Section 3.1.1, the spectral rate parameter is governed by the singularities of the kernel $K(s,t)$ in (3.4c) with $s \in [0,1] \setminus [a,b]$ fixed and $t \in \mathbb{C}$ variable. A singularity exists at $t = s$, and if no other singularities lie within the ellipse of major diameter $(5/4)(b-a)$ with foci at $t = a$ and $t = b$, then choosing $\rho = (b-a)/8$ ensures that $\zeta \leq 1/2$. With this choice of ρ, the interval $(a-\rho, b+\rho)$ is $5/4$ as long as the interval $[a,b]$, just as the major diameter of $E_{1/2}$ is $5/4$ of the distance between its foci.

The point $t = s$ is not necessarily the dominant kernel singularity. All poles and zeros of the function $\|\boldsymbol{\gamma}(s) - \boldsymbol{\gamma}(t)\|$, as well as all poles of the function $\|\boldsymbol{\gamma}'(t)\|$, are kernel singularities. It is impractical, however, to search the complex t-plane for all such points at each value of s. Moreover, when the point $t = s$ is not the controlling singularity, the kernel splitting modeled on the tangent line at $\gamma(s)$ cannot be expected to perform well.

The product rules here are also not designed to handle nearly singular behavior in the kernel when $t = b$ is a corner. The solution $\varphi(t)$ is singular at corner points, so treating nearly singular behavior in the kernel alone is insufficient anyway to ensure rapid spectral convergence. To improve the solution accuracy near corners, I rely on mesh refinement. An example is given in Section 4.4.2, where we compute the scattering of a plane wave from a triangular cylinder.

3.1.4 Particles from Equations of the Second Kind

The discussion has so far been restricted to first-kind integral equations such as the EFIE, which after suitable discretization have the same algebraic structure as the mutual interactions of a collection of point monopoles. Equations of the first kind have a reputation as poorly conditioned problems, but in this case the singularity of $\Phi(\boldsymbol{x}, \boldsymbol{y})$ typically restrains the condition number to grow like $\kappa(\Phi) = O(N)$ as $N \to \infty$.

For smooth closed curves Γ, a second-kind integral equation such as the CFIE or the CSIE is a better option. It can be shown [98] that the condition number of the discrete problem is bounded as the discretization is refined, so $\kappa(\Phi) = O(1)$ as $N \to \infty$. This property suits the application of a broad class of iterative solvers.

The CSIE is transformed into a particle problem in which both monopoles and dipoles are present. The field at point $\boldsymbol{x}$ generated by a unit dipole that has orientation $\hat{\boldsymbol{d}}$ and location $\boldsymbol{y}$ is $\hat{\boldsymbol{d}} \cdot \nabla_{\boldsymbol{y}} \Phi(\boldsymbol{x}, \boldsymbol{y})$. For the CSIE the dipoles are all oriented in the direction of the boundary normal, so $\hat{\boldsymbol{d}} = \hat{\boldsymbol{n}}(\boldsymbol{y})$. At each boundary grid point $\boldsymbol{x}_n$ there is both a monopole with charge q_n and a normally directed dipole with amplitude λq_n, where λ is a coupling constant. The interaction (3.1) is changed to

$$u_m = \sum_{n \neq m} \big(\Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) + \lambda \hat{\boldsymbol{n}}(\boldsymbol{x}_n) \cdot \nabla_{\boldsymbol{x}_n} \Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) \big) q_n. \tag{3.33}$$

We still have a particle problem, but the interaction law is more complicated.

The CFIE is transformed into a particle system consisting of monopoles only, but directional derivatives of the generated field must be computed as well as the field values themselves. The derivative in the direction $\hat{\boldsymbol{d}}$ of the field at point $\boldsymbol{x}$ generated by a unit monopole at location $\boldsymbol{y}$ is $\hat{\boldsymbol{d}} \cdot \nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}, \boldsymbol{y})$. For the CFIE the required derivatives are in a direction normal to the boundary, so $\hat{\boldsymbol{d}} = \hat{\boldsymbol{n}}(\boldsymbol{x})$. The

data u_m is the linear combination $u + \lambda(\partial u/\partial n)$ at the point $\boldsymbol{x}_m$, and

$$u_m = \sum_{n \neq m} \big(\Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) + \lambda \hat{\boldsymbol{n}}(\boldsymbol{x}_m) \cdot \nabla_{\boldsymbol{x}_m} \Phi(\boldsymbol{x}_m, \boldsymbol{x}_n) \big) q_n. \tag{3.34}$$

This potential is slightly different from the CSIE potential in (3.33).

Section 2.6 describes how the fast multipole method for the monopole interaction $\Phi(\boldsymbol{x}, \boldsymbol{y})$ can be modified to accommodate the interaction laws (3.33) and (3.34).

3.2 Iterative Multipole Solvers

Since multipole algorithms give a fast matrix-vector product $\Phi \boldsymbol{q}$, it is natural to embed them into iterative solvers that compute a solution to $\Phi \boldsymbol{q} = \boldsymbol{u}$ using a sequence of matrix-vector products.

If Φ^{-1} is well-approximated by a matrix polynomial $p(\Phi)$ of degree r, then $\hat{\boldsymbol{q}} = p(\Phi)\boldsymbol{u}$ is a good approximation of the solution $\boldsymbol{q} = \Phi^{-1}\boldsymbol{u}$. If $\|\Phi^{-1} - p(\Phi)\| < \epsilon$, then $\|\boldsymbol{q} - \hat{\boldsymbol{q}}\| < \epsilon \|\boldsymbol{u}\|$. The Cayley–Hamilton theorem ensures that Φ^{-1} is *exactly* represented by a polynomial $p(\Phi)$ of degree at most $N - 1$, but an iterative solver is really only attractive if $r \ll N$.

If $p(x) = \sum_{j=0}^{r} c_j x^j$, then the approximate solution

$$\hat{\boldsymbol{q}} = \sum_{j=0}^{r} c_j \Phi^j \boldsymbol{u} \tag{3.35}$$

requires $r + 1$ left multiplications of a vector by Φ. Each of those products can be computed with a fast multipole method.

Krylov solvers begin with $r = 0$ and sequentially increase r, implicitly constructing a new set of good polynomial coefficients $\{c_j\}$ at each step.

3.2.1 Krylov Iterations

We restrict our attention to a family of iterative solvers that select an approximate solution from a nested sequence $K_1 \subseteq K_2 \subseteq K_3 \subseteq \cdots$ of Krylov subspaces $K_i := \operatorname{span}\{\boldsymbol{u}, \Phi\boldsymbol{u}, \Phi^2\boldsymbol{u}, \ldots, \Phi^{i-1}\boldsymbol{u}\}$. Just such a strategy is suggested by (3.35), where clearly $\hat{\boldsymbol{q}} \in K_{r+1}$. (If the user is able to supply a nonzero initial guess of the solution, this structure changes slightly [122].)

The conjugate gradient method (CG) is the oldest member of this family, but it requires that the matrix $\Phi \in \mathbb{C}^{N\times N}$ be Hermitian positive definite, a condition not satisfied by discretizations of scattering integral equations. Many Krylov solvers have been designed for matrices without that structure. In solving scattering problems I have found the greatest success with the generalized minimal residual method (GMRES) [121].

GMRES picks at iteration i an approximate solution vector $\hat{\boldsymbol{q}}_i \in K_i$ that is the least squares solution of the overdetermined system $\Phi\hat{\boldsymbol{q}}_i \approx \boldsymbol{u}$. Like CG, GMRES finds the exact solution in at most N iterations in exact arithmetic, but typically the iteration is stopped when the residual norm $\|\boldsymbol{u} - \Phi\hat{\boldsymbol{q}}_i\|$ falls below a specified threshold.

The principal deficiency of GMRES is that an orthogonal basis $\{\boldsymbol{k}_j\}$ of the Krylov subspaces must be accumulated, so that $K_i = \operatorname{span}\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_i\}$. The basis vector $\boldsymbol{k}_{i+1}$ is computed by extracting the part of $\Phi\boldsymbol{k}_i$ that is orthogonal to K_i. Usually this computation is carried out by a modified Gram–Schmidt process, a recurrence involving all $\boldsymbol{k}_j$ for $j \le i$. Consequently, each iteration requires a growing amount of work and storage. In contrast, CG is neatly described by a three-term recurrence, with a constant stepwise complexity.

One approach to a short-recurrence Krylov solver for nonsymmetric systems is to

apply CG to the normal equations (CGN). The system $\Phi \boldsymbol{q} = \boldsymbol{u}$ is equivalent to the normal equations $\Phi^H \Phi \boldsymbol{q} = \Phi^H \boldsymbol{u}$, which for nonsingular Φ is a Hermitian positive definite system.

The required number of CG iterations has an upper bound proportional to the square root of the condition number $\kappa(\Phi) = \|\Phi\|_2 \|\Phi^{-1}\|_2$. Since $\kappa(\Phi^H \Phi) = \kappa^2(\Phi)$, CGN is apparently a good choice for very well conditioned systems. An integral equation of the second kind typically generates a matrix with bounded condition number, so that the number of CGN iterations is bounded as well. (The bound may, however, depend on the wavenumber k. Under frequency scaling, in which N and k both increase while N/k is held constant, the condition number typically increases like $O(\log N)$.)

On the other hand, CGN has the following undesirable properties:

- It requires a computational routine for evaluating $\Phi^H \boldsymbol{q}$ in addition to one for $\Phi \boldsymbol{q}$.
- It requires two matrix-vector products per iteration, while CG and GMRES require only one.

If, as for the EFIE, the particles are all monopoles, implementation of the conjugate transpose operation is trivial: All occurrences of the imaginary unit i in the computational routine for $\Phi \boldsymbol{q}$ should be replaced with $-i$. If the particles are a mixture of monopoles and dipoles, as in the case of the CSIE, then the implementation is somewhat more complicated. Nevertheless, it is the latter property that is most unreasonable. Even if a fast multipole method is used, these dense matrix-vector multiplications are expensive, and the expense of an additional product usually dwarfs the expense of a long GMRES recurrence.

The prohibitive cost of a matrix-vector product actually makes the choice of a

Krylov solver a bit easier for dense systems than for sparse systems. For a sparse matrix, the matrix-vector product may be cheap, and a short recurrence can yield a significant speed improvement. For these systems, in addition to CGN there is the method of biconjugate gradients (BCG), as well as a collection of more recently developed solvers like CGS [128], QMR [58], and BiCGSTAB [138].

It is worth noting that there exist specializations [56] of BCG and QMR to complex symmetric systems that require only one matrix-vector product per iteration. Moreover, unlike GMRES, they use short recurrences. While Φ is complex symmetric if the particles are all monopoles, the symmetry is broken if dipoles are allowed, or if the short-range interactions in the perturbed particle problem (3.3) are not complex symmetric. Generally, a complex symmetric system is obtained in electromagnetic scattering problems only for certain low-order discretizations of the EFIE. Low-order discretizations are not unusual in practice, however, and complex symmetric QMR is commonly encountered in computational electromagnetics work.

The small condition number of discretized integral operators of the second kind favors the use of CGN. Generally, GMRES will also converge rapidly in such cases, but counterexamples can be constructed. While the convergence behavior of CGN depends on the singular value spectrum of Φ, for GMRES it is the eigenvalues that matter. CGN is likely to do much better only if the eigenvalues are not clustered, but instead occupy an area of the complex plane that covers a large arc of the unit circle [136]. Such distributions appear to be atypical for scattering problems.

3.2.2 Preconditioning

Sometimes a large condition number is unavoidable, and the Krylov iteration converges slowly. Open curves, for instance, do not lend themselves to a second-kind

integral equation. Although a weakly singular integral equation of the first kind for such a boundary may produce a condition number that grows only as $O(N)$, the difference in performance for open and closed curves can be striking. This performance gap can, however, be partially closed.

The best way to speed up a Krylov solver is to construct a cheap and effective *preconditioner.* The idea is to replace the linear system $\Phi \boldsymbol{q} = \boldsymbol{u}$ with an equivalent system $\widehat{\Phi}^{-1}\Phi \boldsymbol{q} = \widehat{\Phi}^{-1}\boldsymbol{u}$ for which

- $\widehat{\Phi}^{-1}\boldsymbol{y}$ is easy to compute for any $\boldsymbol{y}$.
- The iteration for $\widehat{\Phi}^{-1}\Phi$ requires fewer steps than the iteration for Φ.

The name "preconditioner" suggests that the condition number of $\widehat{\Phi}^{-1}\Phi$ should be smaller than the condition number of Φ, but many effective preconditioners do not improve the condition number at all. Instead, compared to the eigenvalue spectrum of Φ, the spectrum of $\widehat{\Phi}^{-1}\Phi$ may consist of fewer and tighter clusters of eigenvalues.

Preconditioners of several types have been used in conjunction with multipole methods. A common starting point is to let $\widehat{\Phi}$ be a sparse approximation of Φ. In Section 3.2.4, I take advantage of MATLAB's `luinc` built-in function to compute an incomplete LU factorization of $\widehat{\Phi}$. The approximate solution of the preconditioner system using that factorization requires sparse forward and backward substitutions.

Other investigators have applied sparse approximate inverse preconditioners to scattering problems [2] [18]. Those preconditioners approximate the inverse $\widehat{\Phi}^{-1}$ explicitly as a sparse matrix, so that the product $\widehat{\Phi}^{-1}\boldsymbol{y}$ requires a one sparse matrix-vector product instead of two sparse triangular solves. The chief motivation for studying these preconditioners is that the data flow of a sparse matrix-vector product is, for implementation on a distributed parallel computer, more favorable than a sparse triangular solve.

3.2.3 Example: Scattering from a Smooth Jordan Curve

To demonstrate the capabilities of multipole, consider the scattering of a plane wave from the analytic curve pictured in Figure 3.3. Since the curve is closed, we choose to solve the CSIE, and we measure the time and memory required for a sequence of problems that increase in size. There are various ways to increase N, but here we choose a frequency scaling that increases the wavenumber k while keeping fixed the number of grid points per wavelength. The data reported in Table 3.1 was measured for the geometric sequence $k \in \{16, 16\sqrt{2}, 32, 32\sqrt{2}, \dots\}$, where powers of $\sqrt{2}$ have been used instead of powers of 2 simply to generate more measurements before exhausting memory.

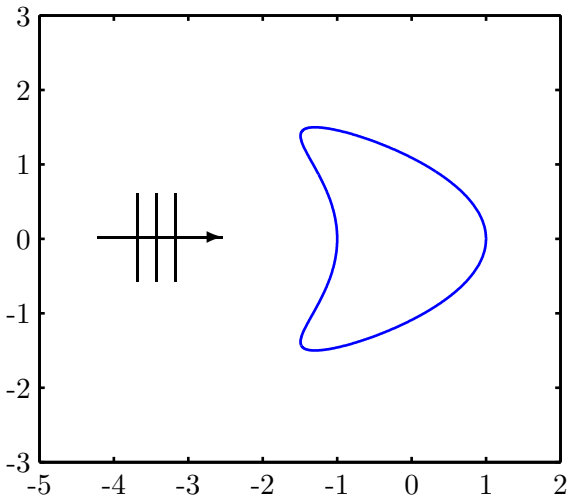


Figure 3.3: A plane wave $u_{\text{inc}}(x, y) = e^{ikx}$ illuminates the impenetrable obstacle defined in Figure 3.1, and borrowed from Colton and Kress [31].

Figure 3.4 shows a cluster hierarchy generated automatically for $k = 64$ with at least 5 grid points per wavelength. There are four levels, with 10 roots at the topmost level. At higher values of k, the tree will have more levels. On each leaf arc, the particle locations are an affine transformation of the zeros of the Chebyshev

polynomial $T_{32}(z)$. The Chebyshev points have been prescribed by the high-order discretization developed in Section 3.1.1. With 32 points, the order of the discretization is 32. If N is increased for fixed k by leaf subdivision, the discretization error decays to zero at the asymptotic rate $O(N^{-32})$.

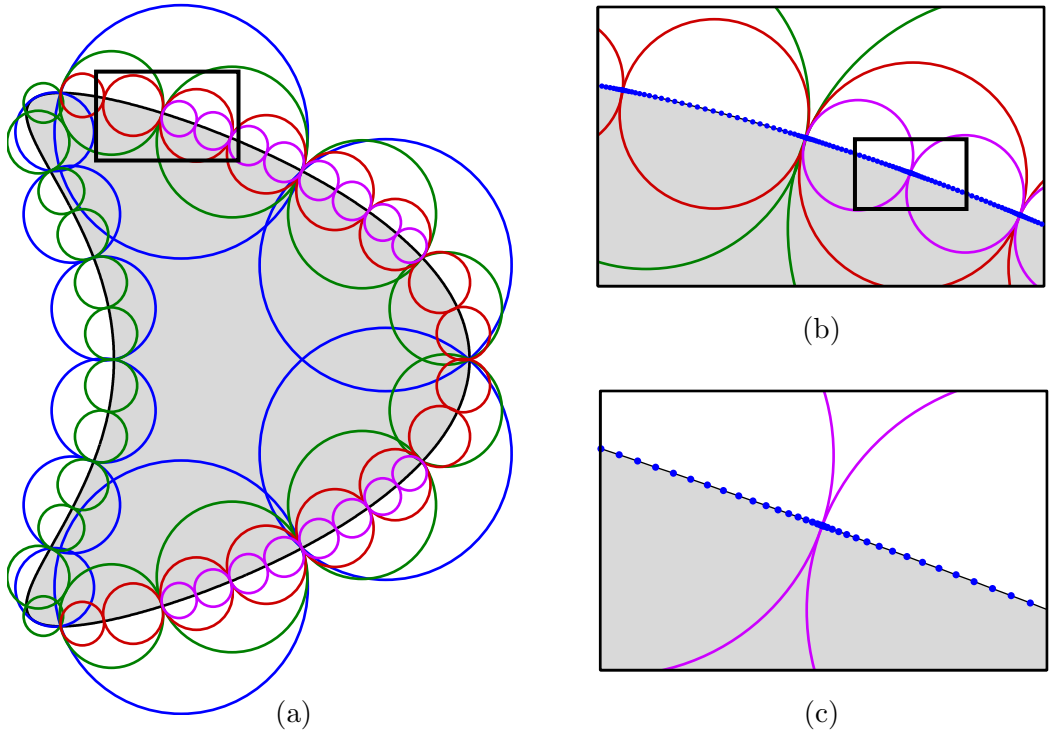


Figure 3.4: (a) Cluster hierarchy for the kite-shaped boundary. (b) Close-up of the interior of the rectangle in (a). The particle locations on the curve are now visible. Each leaf cluster contains 32 particles. (c) Zooming in still further, the increased particle density at the cluster boundaries becomes more clearly visible.

Another parameter under user control is the multipole truncation error. Here the relative accuracy is $\epsilon = 10^{-6}$. At an order of 32, the discretization needs only 5 points per wavelength to meet this specification. The spectral expansion lengths for each disk are determined automatically so that the truncation error is also smaller than ϵ. The iterative solver, unpreconditioned and unrestarted GMRES, is terminated

when the relative residual norm falls below ϵ.

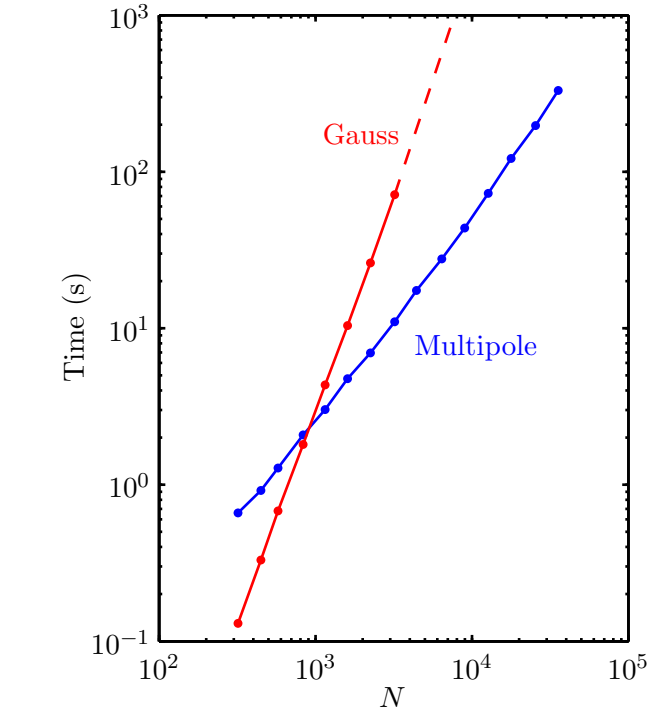
The combination of GMRES and multipole is contrasted with Gaussian elimination in Figure 3.5. Gaussian elimination is only able to cope with relatively small problems before memory is exhausted. For sufficiently small problems, however, Gaussian elimination is faster. The performance crossover is measured for this example to lie below $N = 1000$. At larger values of N, the iterative multipole solver plainly exhibits its superiority.

Only the solve time has been graphed in Figure 3.5(a), so as not to obscure the $O(N^3)$ complexity of Gaussian elimination with the $O(N^2)$ fill time. In Table 3.1, the total time is separated into fill and solve times for each algorithm. The fill time for Gaussian elimination is the time required to form the matrix Φ, while the fill time for the iterative solver is mostly the time needed to compute the translation operators and the sparse matrix of local interactions.

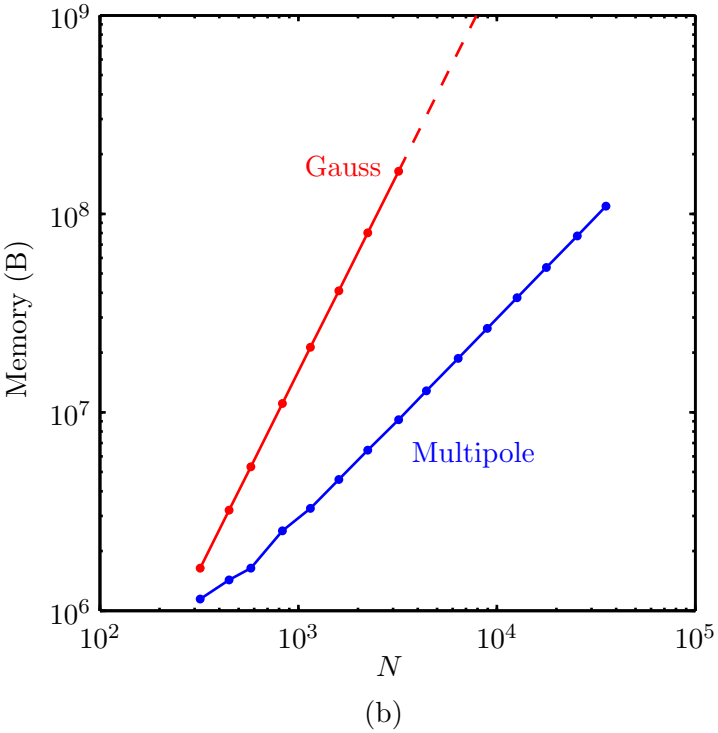
Gaussian elimination timings enclosed in parentheses have been extrapolated from the times at $N = 3200$. Note that, for problems that fit into memory, the fill time for Gaussian elimination dominates the solve time. Including the fill time in Figure 3.5 only makes the performance of multipole more impressive.

The memory requirements are graphed in Figure 3.5(b). At $N = 4416$ there is not enough memory to hold the dense matrix Φ. The reduced storage requirement of multipole allows it to solve a problem of size $N = 35{,}520$, an increase of almost an order of magnitude. More than 20 GB of memory is needed to store a complex double-precision matrix of that size. Multipole fails at the next problem in the sequence, for $k = 2^{23/2}$, which has a size of $N = 50{,}880$. The failure is due to a lack of sufficient memory to hold the multipole operators $\mathcal{M} := \big(\{D_{ij}\}, \{U_i\}, \{R_i\}, \{T_{ij}\}, \{S_i\}, \{V_i\}\big)$.

At the point of failure, the solve time is still only a few minutes. For this par-



(a)



(b)

Figure 3.5: Performance comparison of iterative multipole and Gaussian elimination. The dashed lines extrapolate Gaussian elimination's performance beyond its point of failure. (a) Time spent. (b) Memory used.

Table 3.1: Comparing Multipole to Gaussian Elimination

		Fill Time (s)		Solve Time (s)	
k	N	Multipole	Gauss	Multipole	Gauss
$2^{8/2}$	320	0.99	1.22	0.66	0.13
$2^{9/2}$	448	1.11	2.13	0.92	0.33
$2^{10/2}$	576	1.28	3.37	1.28	0.68
$2^{11/2}$	832	1.97	6.72	2.08	1.81
$2^{12/2}$	1152	2.65	12.49	3.02	4.34
$2^{13/2}$	1600	3.68	23.39	4.76	10.40
$2^{14/2}$	2240	5.22	44.88	6.96	26.16
$2^{15/2}$	3200	7.53	89.52	11.00	71.25
$2^{16/2}$	4416	10.60	(1.7×10^2)	17.42	(1.9×10^2)
$2^{17/2}$	6400	15.69	(3.6×10^2)	27.77	(5.7×10^2)
$2^{18/2}$	8960	22.46	(7.0×10^2)	43.69	(1.6×10^3)
$2^{19/2}$	12672	33.07	(1.4×10^3)	72.75	(4.4×10^3)
$2^{20/2}$	17792	48.66	(2.8×10^3)	121.47	(1.2×10^4)
$2^{21/2}$	25472	73.41	(5.7×10^3)	197.12	(3.6×10^4)
$2^{22/2}$	35520	110.01	(1.1×10^4)	330.80	(9.7×10^4)

ticular boundary, memory is the limiting resource for both multipole and Gaussian elimination. To tackle even larger problems, we can trade time for memory in the multipole implementation. Rather than computing $\mathcal{M}$ once and storing it, some or all of the operators can be recomputed as they are needed at each step of the iteration.

Now let us examine the performance of the iterative solver. Figure 3.6 shows the number of steps required to produce a residual norm of the specified tolerance. The most important thing to notice is that the number of iterations is a small fraction of N, a consequence of the fact that we are solving a well-conditioned problem originating from an integral equation of the second kind. At the same time, the number of required steps is an increasing function of N, which provides some evidence that the spectral condition number of the matrix Φ is also an increasing

function of N.

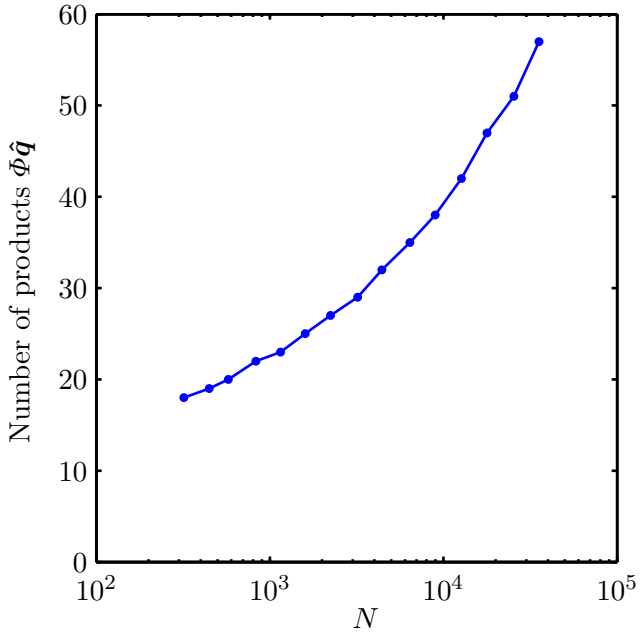


Figure 3.6: Semilogarithmic graph of the number of steps taken by unrestarted GMRES as the problem size N increases.

For a circular boundary of radius α, the condition number can be shown [114] to be proportional to $\log k$. Under frequency scaling, N itself is proportional to k, so the condition number is proportional to $\log N$. Since Figure 3.6 is a semilogarithmic plot, growth linear in $\log N$ should appear as a straight line. The observed growth is faster, at least over the range of N displayed. It appears to be slower than $\log^2 N$, but we will not pursue a more detailed analysis.

In Chapter 2 we estimated the complexity of the hierarchical multipole algorithm to be $O(N \log N)$ flops under frequency scaling. Here we execute the multipole algorithm for the matrix-vector product at each GMRES iteration. Assuming $O(\log N)$ iterations are required for convergence, the solution complexity is $O(N \log^2 N)$ flops.

We should also consider the other vector operations entailed by the GMRES

algorithm. They have a cost of $O(mN)$ flops at step m. The cost increases for successive iterations because of the long Gram–Schmidt recurrence that generates orthonormal bases for the nested Krylov subspaces. After $O(\log N)$ iterations, this overhead totals $O(N \log^2 N)$ flops, which matches the time complexity of the fast matrix-vector products. The memory requirement of the Gram–Schmidt recurrence is $O(N \log N)$, which also matches the space complexity of hierarchical multipole. No other Krylov solver, when used in conjunction with multipole, can give a better asymptotic complexity.

In practice, the cost of the long GMRES recurrence is negligible in comparison with the cost of the multipole computations. A detailed analysis to reveal the constant multipliers hidden by the "big oh" notation would bear this out. Experimental evidence is provided in Figure 3.7, which shows the time that GMRES devotes to each iteration for the largest problem of Table 3.1.

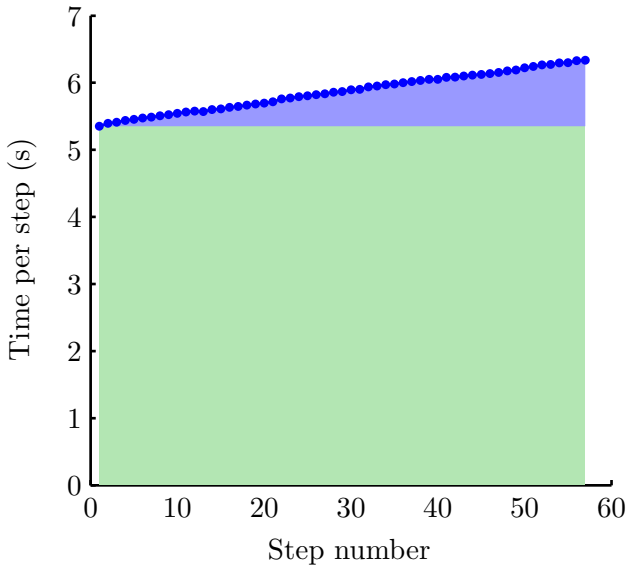


Figure 3.7: The fractional increase in work at each step is small. The area of the rectangle on bottom is the total time spent in the multipole algorithm. The triangle on top is the iteration overhead of GMRES.

The time is indeed an increasing function of step number, but the bulk of the time needed for each step is consumed by the matrix-vector multiplication. The last iteration is only about 20 percent more expensive than the first.

If this added cost becomes too great, one solution is to restart GMRES. Another is to try a Krylov solver that does not incur a growing overhead, but the next figures illustrate that this might not be a good idea.

Figure 3.8 displays the convergence curves of various Krylov solvers applied to the problem with wavenumber $k = 64$. I have experimented with all six of the nonsymmetric Krylov solvers available in MATLAB. Three of these—CGN, BCG, and QMR—require a computational routine to apply the conjugate transpose Φ^H. Since I have not yet built this capability into my multipole code, for this experiment the Krylov solvers all use standard matrix-vector products.

It is perhaps more common to see such curves as plots of relative residual norm versus step number, but that is not the right comparison to make, since different solvers incur different costs at each step. We care about the total computation time, and the number of steps taken in that time is irrelevant.

Figure 3.8(a) confirms that GMRES is a good choice for this boundary, but it also shows that CGS may be an even better selection. The CGS residuals do exhibit a more erratic behavior, and this behavior is commonly observed for CGS. It is not difficult to find boundaries for which this phenomenon renders the performance of CGS quite poor in comparison with GMRES.

Furthermore, the timings graphed in Figure 3.8(a) suffer from an inefficiency in the M-file implementations provided in the MATLAB distribution. With the exception of `lsqr`, the function that implements CGN, all the others compute the residual at each step with a matrix-vector product. The standard implementations [12] up-

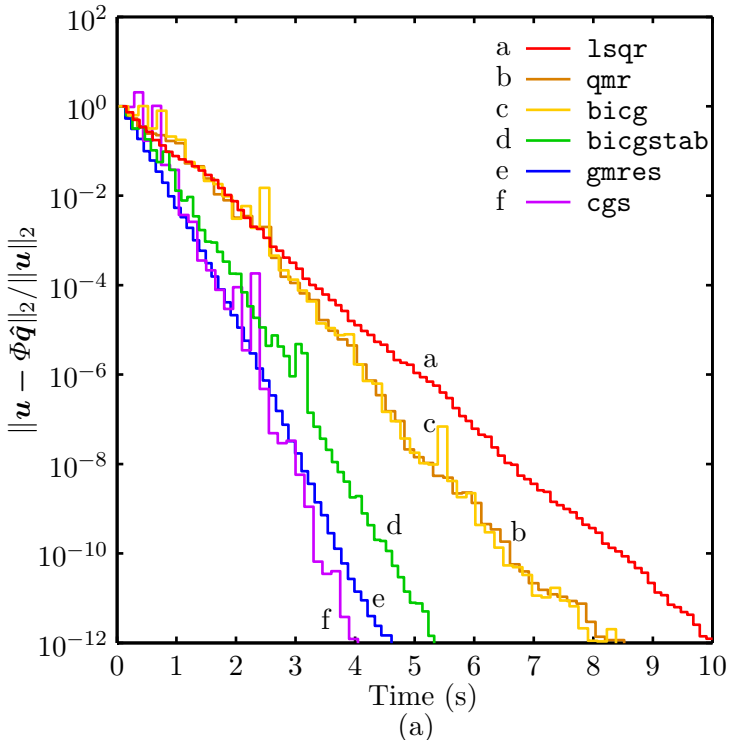


Figure 3.8: Relative residual norm at each step of six Krylov solvers. (a) CGS appears to converge faster than GMRES, but the convergence is more erratic. CGN is clearly the loser. (b) After compensating the timings for MATLAB's inefficient implementation, GMRES is clearly the winner. BiCGSTAB also performs better than CGS.

date the residual at each step through a cheaper computation. Using a matrix-vector product gives a more reliable residual estimate, but when the matrix is dense this extra work can hardly be justified.

I revised the MATLAB function `gmres` to fix this. The new function, `mygmres`, requires only a single matrix-vector product per iteration, while `gmres` requires two. I have not removed this inefficiency from the remaining solvers, but the possible improvement may be judged from Table 3.2.

Table 3.2: Matrix-Vector Products per Krylov Iteration

MATLAB function	Is Now	Could Be
`bicg`	3	2
`bicgstab`	4	2
`cgs`	3	2
`gmres`	2	1
`lsqr`	2	2
`qmr`	3	2

The numbers in Table 3.2 can be used to adjust the curves in Figure 3.8(a), assuming that operations other than matrix-vector products can be neglected. For instance, it is plausible that a new implementation of `bicgstab` would be $4/2 = 2$ times as fast.

Figure 3.8(b) shows that adjustment of the measurements in Figure 3.8(a). It predicts that GMRES is in fact the clear favorite, while the performance of BiCGSTAB and CGS are quite close. Since the residual behavior of BiCGSTAB is smoother, it may be preferred to CGS.

One Krylov solver missing from this brief study is TFQMR [57], a variation of QMR likely to be more competitive with BiCGSTAB. Like CGS and BiCGSTAB,

TFQMR requires two matrix-vector multiplies at each step, and it does not use Φ^H. TFQMR has not been included here simply because MATLAB does not currently provide this function.

3.2.4 Preconditioners for Less Hospitable Boundaries

All of the above iterative solutions have been computed without the aid of a preconditioner. Their demonstrated effectiveness is one of the distinguishing traits of integral equations of the second kind. The growing iteration count in Figure 3.6 suggests, however, that even smooth closed curves may benefit from preconditioning.

When the fast multipole method is used to perform the matrix-vector products at each stage of the iteration, the obvious candidate for a preconditioner $\widehat{\Phi}$ is a sparsification of the interaction matrix Φ that explicitly gives the short-range interactions among leaf clusters that are not well separated. By the time the Krylov solver is called, that sparse matrix has already been constructed, since it is needed in the multipole computation. The nonzero entries of $\widehat{\Phi}$ occupy the blocks $\{D_{ij}\}$ of Section 2.5. Those locations also contain all nonzero entries of the perturbation Σ produced by the discretization. The remaining entries of Φ are represented implicitly as a product of translation matrices, and are not readily available for use in a preconditioner.

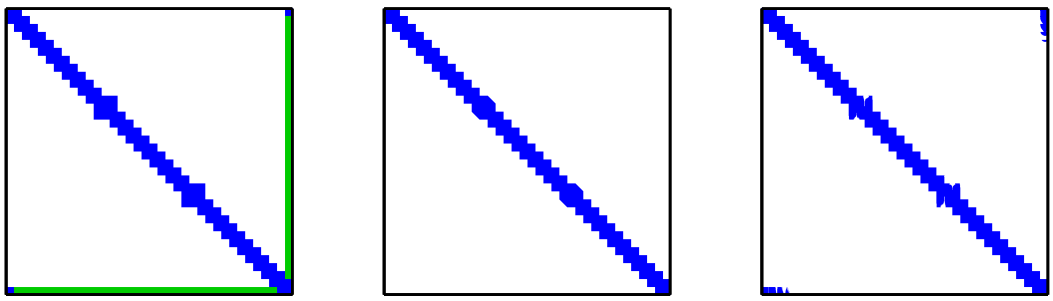
The LU factorization of $\widehat{\Phi}$ can be computed before the Krylov iteration starts. Then the product of $\widehat{\Phi}^{-1}$ and an arbitrary dense vector of length N reduces to two sparse triangular solves. The factorization need not be repeated at each step.

The complete LU factorization usually produces too much *fill-in.* As the rows of $\widehat{\Phi}$ are combined during Gaussian elimination, new nonzero positions are introduced in the triangular factors. This phenomenon can make iterative sparse solvers more

attractive than sparse LU factorization, and such subsidiary iterations applied to $\widehat{\Phi}$ form an important class of preconditioners. If the factorization is preferred, usually the rows and columns of $\widehat{\Phi}$ will be reordered in a way that tries to minimize the amount of fill-in. Another strategy is to perform an *incomplete* factorization, discarding some of the nonzero entries in the computed factors.

The preconditioner that I have applied most successfully is the drop tolerance incomplete LU factorization of $\widehat{\Phi}$. Its performance is controlled by the discard tolerance, which I usually take to be 10^{-2} or 10^{-3}. In computing a solution to an accuracy of $\epsilon < 10^{-6}$, it may pay off to tighten the tolerance to about $\sqrt{\epsilon}$.

Figure 3.9 compares some preconditioner choices, all of which start with $\widehat{\Phi}$. The sparsity pattern of $\widehat{\Phi}$ is shown for the curve of Figure 3.3 with wavenumber $k = 64$. The pattern is nearly the same one produced by a circular boundary, and a similar pattern can be expected for all boundaries that do not differ too much from a circle.



(a) (b) (c)

Figure 3.9: (a) Sparsity pattern of $\widehat{\Phi} \in \mathbb{C}^{1152\times 1152}$ is colored blue. The complete LU factorization produces the green fill-in. The resulting factored preconditioner has 182,272 nonzeros. (b) Nonzero elements of $\widehat{\Phi}$ outside the main band of width 32 are discarded. The LU factorization of the banded approximation generates no fill-in. The resulting factored preconditioner has 110,656 nonzeros. (c) Sparsity pattern of the incomplete LU factorization of $\widehat{\Phi}$, with a drop tolerance of 10^{-3}. Nonzero elements in the strict lower triangle belong to the factor L, and nonzeros in the upper triangle belong to U. The resulting factored preconditioner has 117,583 nonzeros.

It turns out that none of these preconditioners significantly reduces the computation time. The measurements, including the time required for the preconditioner factorization, are given in Table 3.3.

Table 3.3: Solution Time with Three Preconditioners

Preconditioner	Steps	Time (s)
None	24	3.11
Full LU	13	2.50
Band	13	3.92
Incomplete LU	13	2.29

In each case, GMRES has been stopped at a relative residual norm of 10^{-6}. The drop threshold for the incomplete factorization is 10^{-2}. Since the unpreconditioned solver is already fast, there is little incentive to further expand our search for a better preconditioner.

For analytic curves not too different from a circle, preconditioners are unnecessary if a suitable integral equation has been selected. But the class of boundaries for which the standard preconditioners are either unnecessary or else ineffective is small. For boundaries more characteristic of engineering problems, preconditioning is broadly regarded as a worthwhile effort.

I have investigated the following boundary classes, testing the hypothesis that each requires some special care:

1. Curves with corners (e.g., polygons)
2. Curves with a high aspect ratio (e.g., ellipses of high eccentricity)
3. Open curves
4. Closed curves supporting interior resonant modes
5. Small curves ($k \operatorname{diam} \Gamma \ll 1$)
6. Large curves ($k \operatorname{diam} \Gamma \gg 1$)

7. Curves supporting multiple reflections

While it is true that a preconditioner can be of some benefit in each of these cases, it is also true that unpreconditioned iterations often perform well in every instance *except one.* In my experience, class 7 is clearly the most difficult. That is the case of a curve, either open or closed, that can reflect an incident ray many times before it escapes to infinity. For such curves, unpreconditioned iterations do poorly. Unfortunately, standard preconditioners such as those in Figure 3.9 do little to improve matters. Before describing this further, let us dispose of the remaining cases.

In Section 3.2.3 we treated an optically large obstacle (case 6). The last line of Table 3.1 reports timings for a curve with a diameter of more than 975 wavelengths. Figure 3.6 provides evidence that for smooth curves the growth in condition number is sublinear in the wavenumber, $\kappa(\Phi) = o(k)$ as $k \to \infty$. So under frequency scaling the condition number grows without bound at the rate $\kappa(\Phi) = o(N)$ as $N \to \infty$.

Figure 3.10 displays curves of types 1, 3, and 7. Table 3.4 shows the corresponding performance of GMRES both with and without preconditioning. In order to compute the spectral condition number $\kappa(\Phi) = \|\Phi\|_2 \|\Phi^{-1}\|_2$, the matrix size is kept small in each case by limiting the wavenumber k. Since the matrices are small enough to store in physical memory, dense matrix arithmetic has been used instead of fast multipole arithmetic.

Table 3.4: Preconditioning Four Curves

			No Preconditioner		ILU Preconditioner	
Boundary	N	$\kappa(\Phi)$	Steps	Time (s)	Steps	Time (s)
(a) square	1024	21.4	29	1.68	10	1.06
(b) astroid	1280	7.27×10^4	126	9.45	12	1.99
(c) segment	1024	936	68	4.33	25	2.24
(d) spiral	1568	895	319	40.46	281	45.88

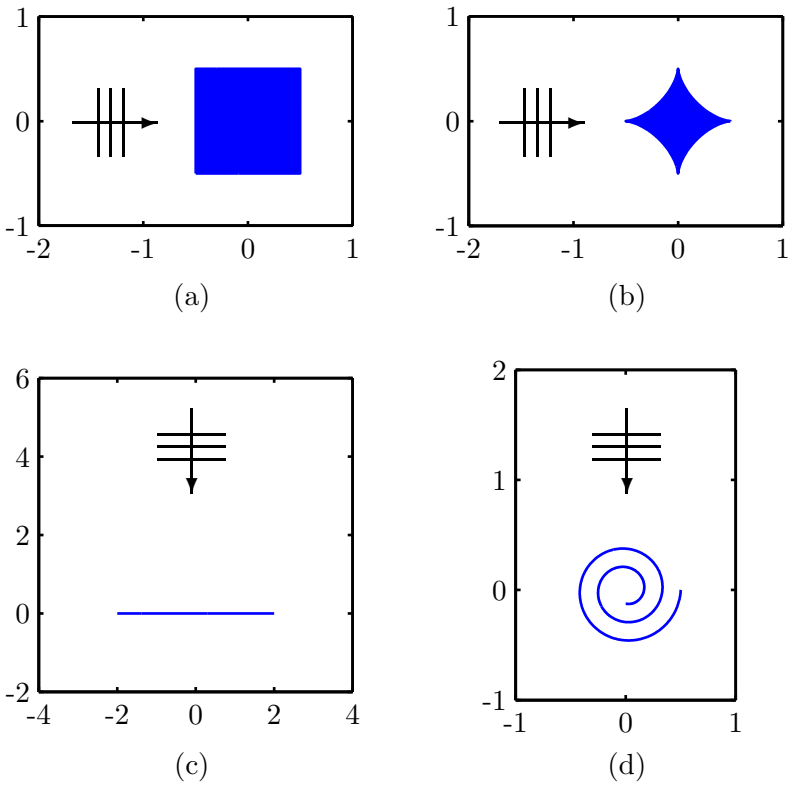


Figure 3.10: Plane wave scattering from four curves Γ. (a) Γ is a square with unit side. (b) Γ is an astroid, a curve with the rectangular parametrization $\boldsymbol{\gamma}(t) = \frac{1}{2}(\cos^3 2\pi t, \sin^3 2\pi t)$. (c) Γ is a straight line segment. (d) Γ is a spiral with a parametrization in polar coordinates (r, ϕ) of $r = \phi/12\pi$ for $\phi \in [3\pi/2, 6\pi]$.

Consider first the scattering from the polygon in Figure 3.10(a). Unlike the boundary in Figure 3.3, this boundary is nonanalytic. In fact, the presence of the corners spoils the compactness of the scattering integral operators. The existence proof contained within the Riesz–Fredholm framework can be repaired [96], but the spectral condition number of the discretized equations is no longer uniformly bounded. Instead, $\kappa(\Phi)$ grows without bound as N increases.

At $N = 1024$, unpreconditioned GMRES requires 29 iterations and 1.68 s to converge to a relative residual norm of $\epsilon = 10^{-6}$. With a drop tolerance of 10^{-2}, an incomplete LU preconditioner improves those figures to 10 iterations and 1.06 s.

The solution time has been reduced by 40 percent, and for larger values of N the absolute savings will be significant.

Note that the improvement in the number of iterations is more impressive: It has been slashed by a factor of three. The disparity between these performance measures exists because construction and application of the preconditioner is not free. The effectiveness of a preconditioner should never be judged solely on the reduction of the number of steps taken by the Krylov method.

The effect of the preconditioner on the eigenvalue distribution of Φ is shown in Figure 3.11. The factored preconditioner is $\widehat{L}\widehat{U} \approx \widehat{\Phi}$, and clearly the eigenvalues of $\widehat{U}^{-1}\widehat{L}^{-1}\Phi$ are clustered more tightly than those of Φ. True to its name, the preconditioner also improves the condition number by a factor of 10, from $\kappa(\Phi) \approx 21.40$ to $\kappa(\widehat{U}^{-1}\widehat{L}^{-1}\Phi) \approx 1.98$. Table 3.4 shows, however, that that does not translate into a factor of 10 improvement in performance.

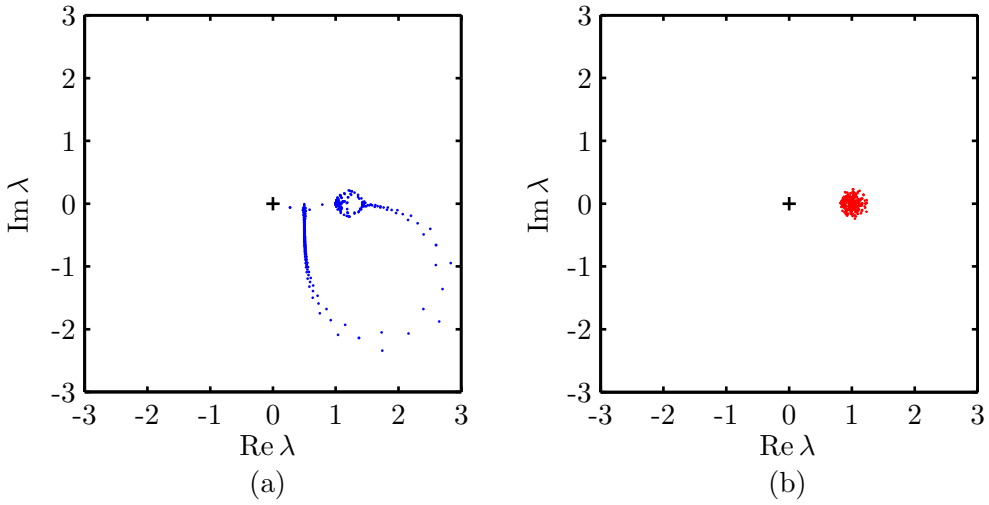


Figure 3.11: Eigenvalues of Φ for the square. Φ is the discretization of the operator $\frac{1}{2}\mathcal{I} - \mathcal{D} - i(k+1)\mathcal{S}$. (a) Before preconditioning. (b) After preconditioning.

As the boundary smoothness is reduced, the preconditioner becomes more valu-

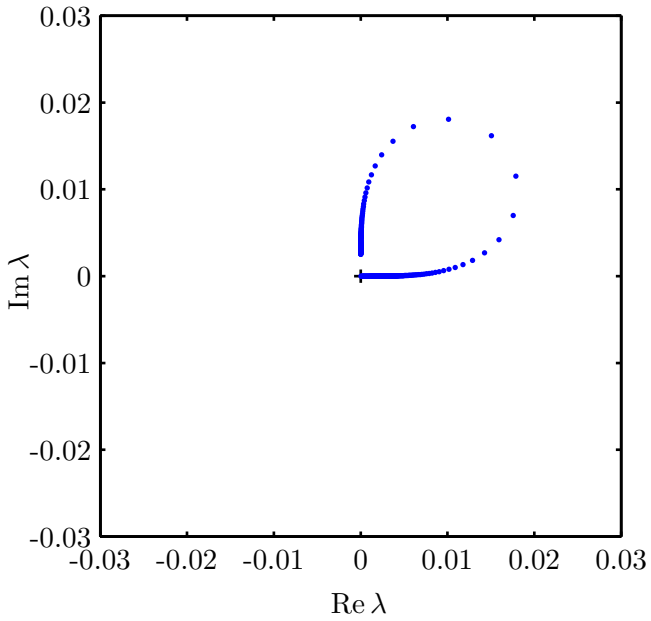


Figure 3.12: Eigenvalues of Φ for the line segment. Φ is the discretization of the operator $-\mathcal{S}$.

able. Figure 3.10(b) shows a boundary with four *cusps*, which are corners with vanishing interior angle. For this curve, the preconditioner gives almost a factor of 5 improvement in solution time.

An integral equation of the second kind is unavailable for open boundaries such as the straight line segment of Figure 3.10(c). Like the solution at corners in Figure 3.10(a), the solution is singular at the segment end points. If we apply the EFIE to this problem, then the convergence of GMRES is expected to be aided by a preconditioner. That is true, as indicated in Table 3.4, but unpreconditioned GMRES also performs quite well. Figure 3.12 shows the eigenvalues of Φ for this case.

Note that the condition number of the discretized EFIE is not nearly as large as it would be if the kernel were smooth. The condition number of the CSIE for the astroid is much larger than the condition number of the EFIE for either open curve in Figure 3.10. Its condition number is larger because the particles (not shown) on

adjacent sides of each cusp are so close together.

Figures 3.13 and 3.14 show the variation of condition number as N increases. The rate of growth appears to be linear for the open curves, and sublinear for the closed curves. Higher growth rates are certainly tolerated when second-order PDEs are discretized with finite differences or finite elements.

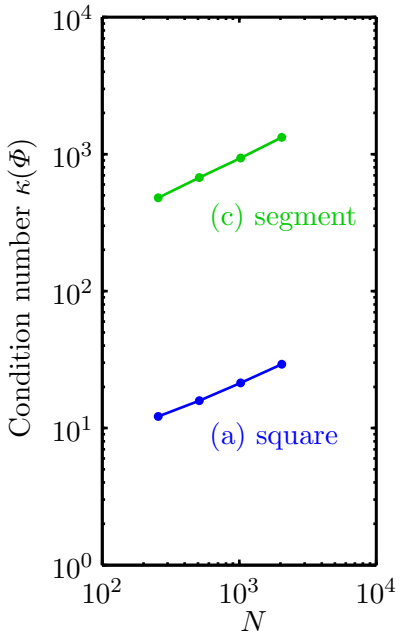
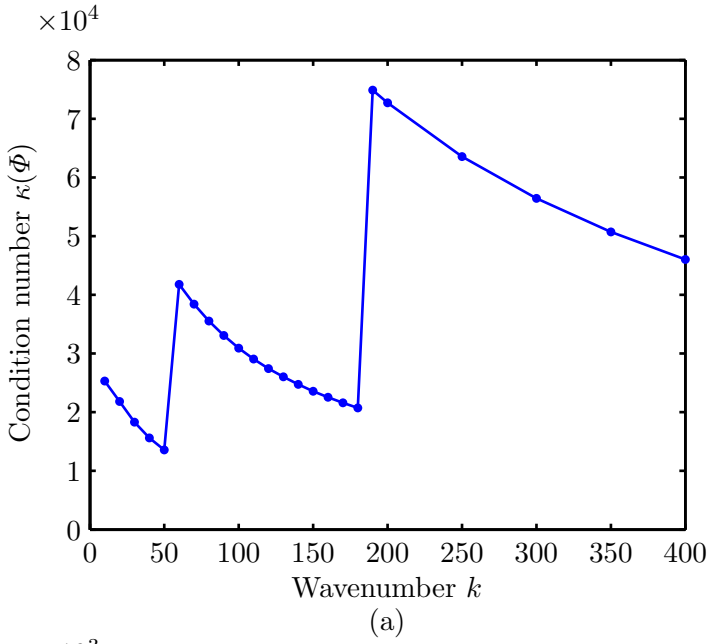
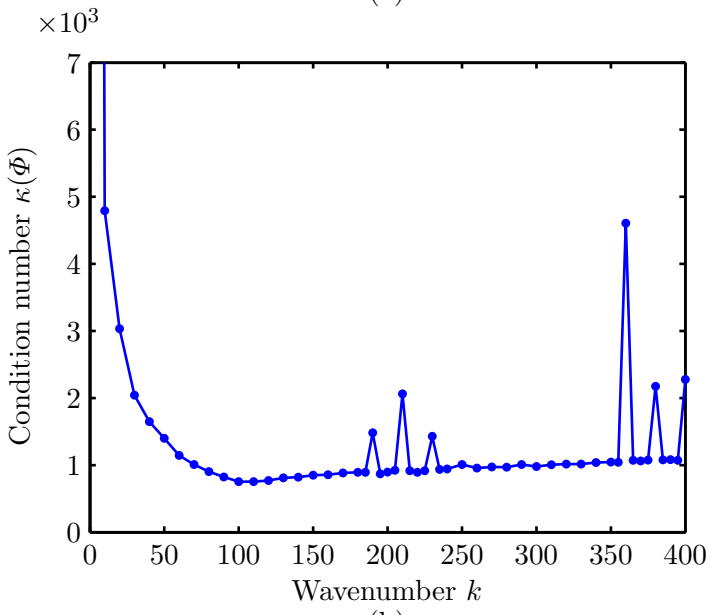


Figure 3.13: Growth of condition number for square and line segment. If the growth were linear in N, then on this logarithmic plot the data points would fall on a line with slope 1. The measured data points are nearly collinear, but the slopes are smaller than 1, indicating sublinear growth. Fitting a line to each data set by least squares, the recorded growth rates are $N^{0.42}$ and $N^{0.49}$ for (a) and (c), respectively.

The last line of Table 3.4 shows that, among these curves, the spiral is most resistant to preconditioning. In spite of the moderate condition number, the iterative solver performs poorly. The reduction in the number of iterations is not enough to compensate the cost of the preconditioner, so even more time is required for the preconditioned iteration. GMRES does much better with the astroid, which has a



(a)



(b)

Figure 3.14: Growth of condition number for astroid and spiral. (a) The two jumps pictured occur when all leaf clusters on the astroid boundary must be split in half to meet the sampling requirement. Between jumps, k increases but N remains constant. (b) Spikes in condition number for the spiral suggest near resonances. The growth as $k \to 0$ suggests that the low frequency problem is poorly conditioned, but the growth occurs slowly.

condition number two orders of magnitude larger than the spiral's.

Figure 3.15 compares the eigenvalue distributions for these curves, and it perhaps indicates why GMRES prefers the astroid. Outliers in the eigenvalue spectrum of the astroid's Φ inflate its condition number, but those eigenvalues are found in the beginning of the Arnoldi iteration that underpins GMRES [136]. Furthermore, if most of the eigenvalues of each problem are collected into a disk, the astroid's eigenvalue disk subtends a smaller angle from the origin than the spiral's eigenvalue disk.

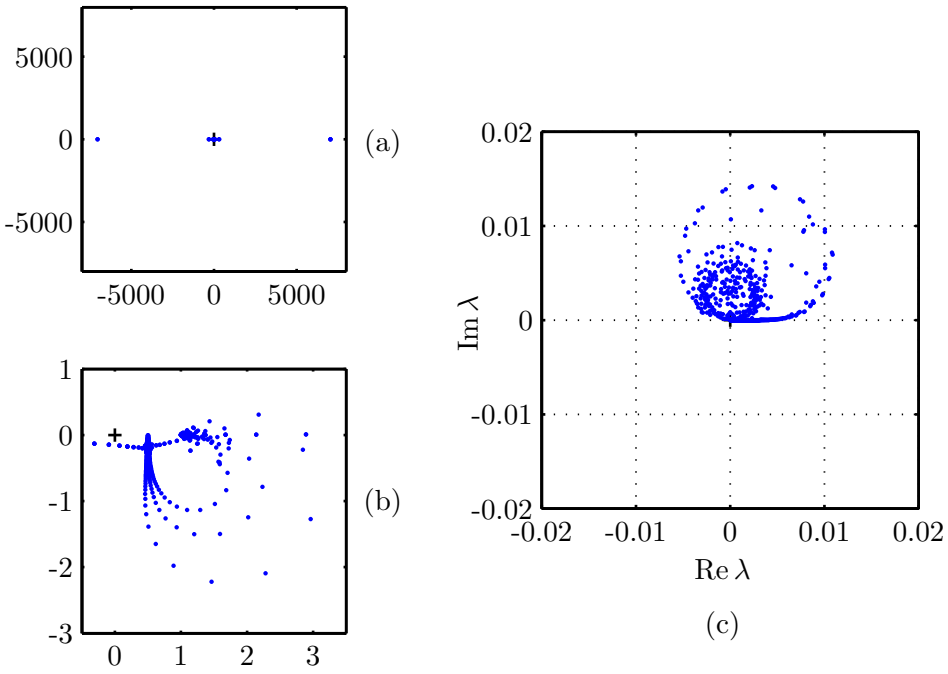


Figure 3.15: Eigenvalues of Φ for astroid and spiral. (a) Eigenvalue spectrum of astroid has outliers that nearly fall on the real axis. (b) Zooming in, this portion of the complex plane contains 1212 eigenvalues, out of the total of 1280. (c) Eigenvalue spectrum of spiral has no outliers or tight clusters.

To understand why curves like the spiral are particularly challenging, consider a ray tracing analysis of the scattering. The ray approximation of high frequency

electromagnetic fields often can quickly yield valuable insights into the behavior of solutions to the Maxwell equations. The ray path drawn in Figure 3.16 suggests two conclusions.

First, a small change in the incidence angle of the ray will substantially alter the locations of the specular reflection points deep inside the spiral. A small deformation of the boundary at any specular reflection point will have a similar effect. The solution of the integral equation, which can be determined from the total field in a neighborhood of the boundary, is therefore sensitive to perturbations in the problem data. Even if the grid points are placed uniformly along the curve, we expect the condition number to be large.

Second, the multiple reflections show how well-separated sections of the boundary can interact strongly. A preconditioner that accounts only for the interactions of neighboring pieces of the curve is not likely to be effective. Since $\widehat{\Phi}$, the starting point for all preconditioners in Figure 3.9, contains only local interactions, it is ineffective. In Chapter 4, I introduce a preconditioner that accounts for strong long-range interactions.

Referring back to the list on page 159, we have yet to consider boundary classes of type 2, 4, or 5. It turns out that if large condition numbers are encountered in these instances, they are almost certainly a result of a poor selection of integral equation or discretization scheme.

Interior resonances of closed curves only cause difficulties for certain integral equations such as the MFIE. Picking the CFIE or CSIE eliminates the problem. Other solutions have been proposed as well [114], but neither CFIE nor CSIE has any serious defect.

Application of the EFIE to small 3-D obstacles has a bad reputation [114], but

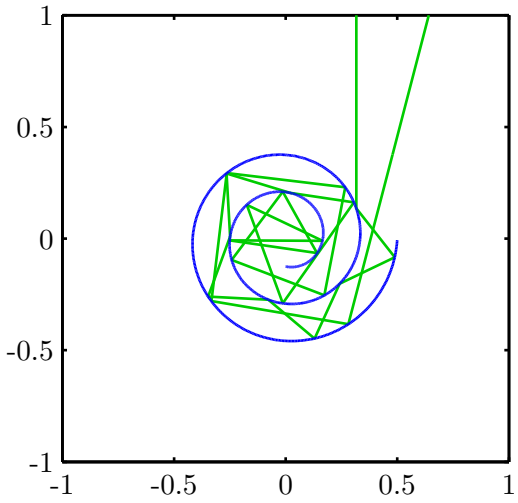


Figure 3.16: A single ray enters the mouth of the spiral and reflects from the boundary many times before finally exiting.

that is because of a poor choice of basis in the method of moments. With a poorly chosen basis, the condition number of the algebraic system can always be made arbitrarily large [98]. By changing the basis, this artificial inflation of the condition number can be avoided.

In the absence of poor user decisions, my numerical experiments show that the condition number depends only weakly on either the diameter or the aspect ratio of the boundary.

For example, consider an ellipse with an axial ratio β of minor diameter to major diameter. For a fixed wavenumber k, there exists a number $\beta_0 > 0$ such that an interior mode cannot exist if $\beta < \beta_0$. For small frequencies or small axial ratios, the EFIE will not suffer from nonuniqueness difficulties. Under those circumstances, a tabulation of condition numbers for both EFIE and CSIE discretizations is given in Table 3.5.

While keeping the number of variables N fixed, the condition number can be

increased by decreasing either k or β. But the growth is sublinear in the reciprocal of either parameter: $\kappa(\Phi) = o(1/k)$ as $k \to 0$ and $\kappa(\Phi) = o(1/\beta)$ as $\beta \to 0$.

Table 3.5: Low Frequencies and High Eccentricities

EFIE

$\beta = 1/16,\ N = 512$		$k = 100,\ N = 1024$	
k	$\kappa(\Phi)$	β	$\kappa(\Phi)$
1/4	2.17×10^5	1/4	2.48×10^3
1/8	2.67×10^5	1/8	5.59×10^3
1/16	3.18×10^5	1/16	14.5×10^3
1/32	3.70×10^5	1/32	34.9×10^3
1/64	4.24×10^5	1/64	76.4×10^3

CSIE

$\beta = 1/16,\ N = 512$		$k = 100,\ N = 1024$	
k	$\kappa(\Phi)$	β	$\kappa(\Phi)$
1/4	15.55	1/4	10.48
1/8	17.80	1/8	11.79
1/16	20.45	1/16	15.70
1/32	23.40	1/32	24.16
1/64	26.55	1/64	38.08

Chapter 4

A Direct Multipole Solver

A need exists for a solver able to effectively treat boundaries, such as the spiral of Figures 3.10(d) and 3.16, that resist preconditioning. In this chapter, I introduce a direct solver created in an effort to meet that need.

A direct solver is a noniterative solver, one that reaches its solution in a predetermined finite number of operations. Gaussian elimination is the direct solver of choice for unstructured dense linear systems, but its $O(N^3)$ complexity is prohibitive when N is large, as in high frequency scattering problems. But, like nearly all discretizations of continuous problems, those scattering problems do have structure. Chapter 2 identified the structure utilized by multipole methods. The challenge here lies in incorporating the multipole structure into a fast and stable direct solver.

Direct multipole solvers have been developed for other problems, and some of them are mentioned in Chapter 6. To the best of my knowledge, no solver of this type has been reported for the general obstacle scattering problem. Jones, Ma, and Rokhlin [88] designed a direct multipole solver for Newton–Coulomb interactions

among particles that lie on 2-D Cantor sets. Among the reported solvers, theirs is closest in its domain of application to Problem 1.1.

Compared to the work of Rokhlin and his colleagues, the solver presented here takes a completely different approach. It uses the same well-developed sparse matrix methods that are applied to finite element discretizations of elliptic PDEs. The construction turns any multipole method for a fast matrix-vector product into a direct solver. It is not limited to any particular interaction law, nor is it limited to two dimensions.

4.1 Sparse Matrix Representation of the Multipole Dag

We begin by reviewing the signal flow dag introduced in Section 2.5.2 and pictured in Figure 2.16. The signal at each graph vertex is a vector. Listed in the order of their priority, the vertices fall into four consecutive groups:

1. Leaf charge vectors $\{\boldsymbol{q}_i\}$
2. Exterior expansion coefficient vectors $\{\boldsymbol{a}_i\}$
3. Interior expansion coefficient vectors $\{\boldsymbol{b}_i\}$
4. Leaf field vectors $\{\boldsymbol{u}_i\}$

Each directed edge is weighted by a matrix. There are six types of edge matrices, all of them dense:

1. Matrices $\{D_{ij}\}$ that evaluate the field $\boldsymbol{u}_i$ at points of leaf cluster i produced by the charge $\boldsymbol{q}_j$ in leaf cluster j, where i and j are not well separated
2. Matrices $\{V_i\}$ that construct exterior expansion coefficients $\boldsymbol{a}_i$ from

the charge $\boldsymbol{q}_i$ contained in leaf cluster i

3. Matrices $\{S_i\}$ that translate the origin of an exterior expansion from the center of disk i to the center of its parent disk

4. Matrices $\{T_{ij}\}$ that translate an exterior expansion with origin at the center of disk j to an interior expansion with origin at the center of disk i, where i and j are well separated

5. Matrices $\{R_i\}$ that translate the origin of an interior expansion from the center of disk i's parent to its own center

6. Matrices $\{U_i\}$ that evaluate the field $\boldsymbol{u}_i$ with interior expansion coefficients $\boldsymbol{b}_i$ at the points of leaf cluster i

The edge matrices $\{S_i\}$, $\{T_{ij}\}$, and $\{R_i\}$ are Toeplitz, and this structure is utilized to reduce the work associated with those edges. No useful structure is extracted from the edge matrices $\{D_{ij}\}$, $\{U_i\}$, and $\{V_i\}$.

The multipole dag suggests an approach to a direct solver. The main idea is to augment the linear system $\Phi\boldsymbol{q} = \boldsymbol{u}$ by letting all vertices in the dag—not just $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{q}_i\}$—be variables.

The new variables are the spectral series coefficients $\{\boldsymbol{a}_i\}$ and $\{\boldsymbol{b}_i\}$. The same number of new equations can be appended to the original system, giving a block linear system that encapsulates all the operations in the dag,

$$\begin{bmatrix} D & & U \\ V & S & \\ & T & R \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}. \tag{4.1}$$

The vector $\boldsymbol{a}$ is formed by concatenating the vertex signals $\{\boldsymbol{a}_i\}$, where i iterates over each node of the source tree. The vector $\boldsymbol{b}$ is formed by concatenating the vertex signals $\{\boldsymbol{b}_i\}$, where i iterates over each node of the destination tree.

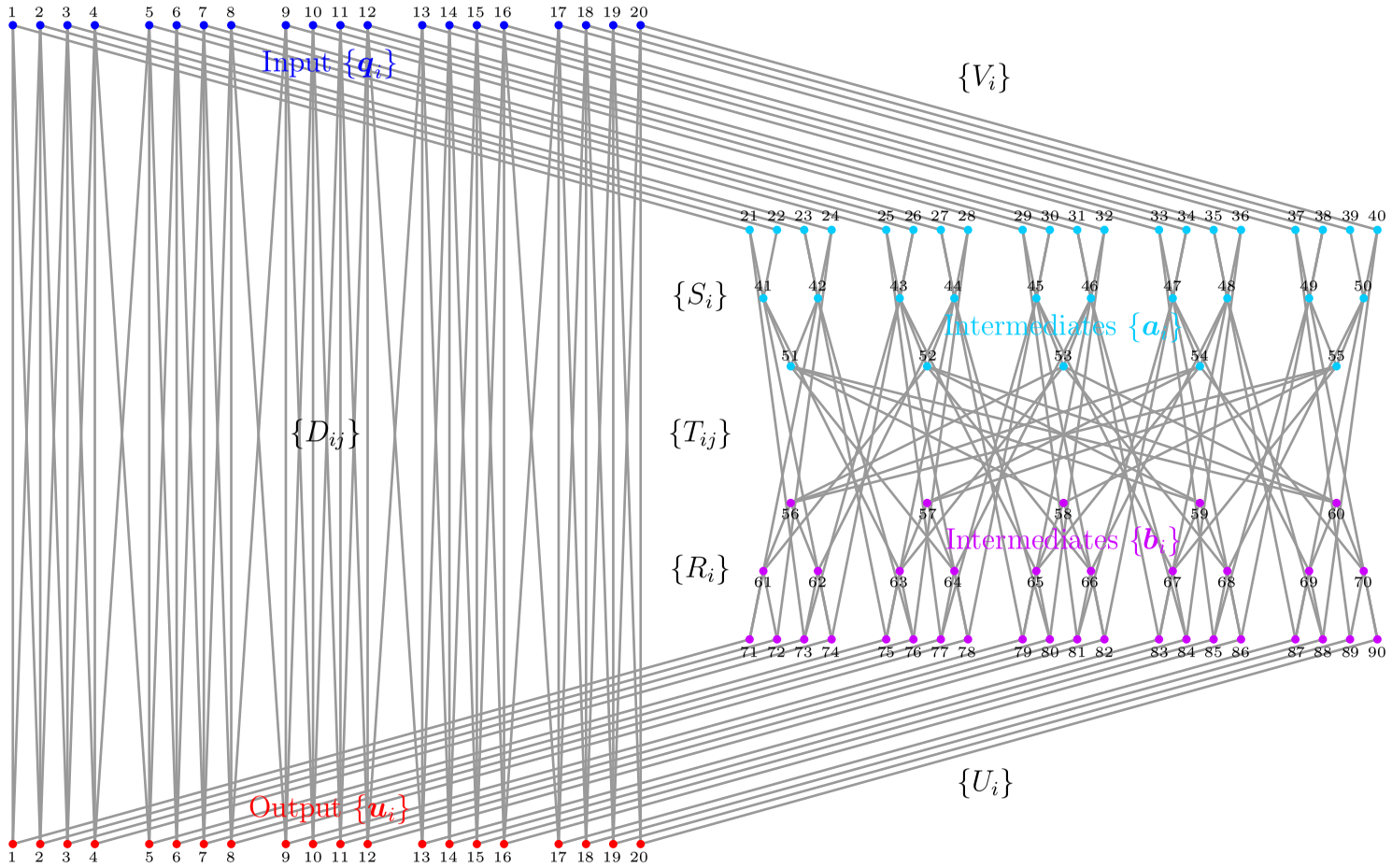


Figure 4.1: The numbers assigned to the input vertices give the positions of $\{\boldsymbol{q}_i\}$ in the vector on the left-hand side of (4.1). The numbers assigned to the output vertices give the positions of $\{\boldsymbol{u}_i\}$ in the vector on the right-hand side. The intermediate variables $\{\boldsymbol{a}_i\}$ and $\{\boldsymbol{b}_i\}$ have been assigned the same positions in both vectors. That is not a requirement, and each of the intermediate vertices can be assigned separate row and column indices.

The submatrix labeled D is the composition of all the short-range interaction edge matrices $\{D_{ij}\}$, where i iterates over each leaf of the destination tree and j iterates over each leaf of the source tree. Since $D_{ij} = 0$ if the clusters belonging to the leaves i and j are well separated, D is sparse. It is identical to the matrix $\widehat{\Phi}$ used in Section 3.2.2 to generate a preconditioner for Φ.

The sparsity pattern of D depends on the ordering assigned to the variables and equations. To illustrate a typical pattern, Figure 4.1 labels the vertices of the dag taken from Figure 2.16. With this vertex order, in which adjacent leaf clusters are numbered consecutively, D is a block tridiagonal matrix.

Like D, each of the other five nonzero submatrices in (4.1) is block sparse. That sparsity reflects the sparsity of the multipole graph. Ordering vertices as in Figure 4.1, the submatrices U and V are block diagonal. The submatrices R, T, and S have the more interesting sparsity patterns shown in Figure 4.2.

In an inverse particle problem, we must determine the charge amplitudes $\boldsymbol{q}$ responsible for generating a given field $\boldsymbol{u}$. The obvious deficiency of (4.1) is that we do not know in advance the subvectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of the right-hand side.

But those unknowns appear in the vector on the left-hand side of (4.1) as well. Subtracting the block column vector $\left[\mathbf{0}; \boldsymbol{a}; \boldsymbol{b}\right]$ from both sides of the equation gives

$$\begin{bmatrix} D & & U \\ V & S & \\ & T & R \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{4.2}$$

which further simplifies to

$$\begin{bmatrix} D & & U \\ V & S-I & \\ & T & R-I \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{4.3}$$

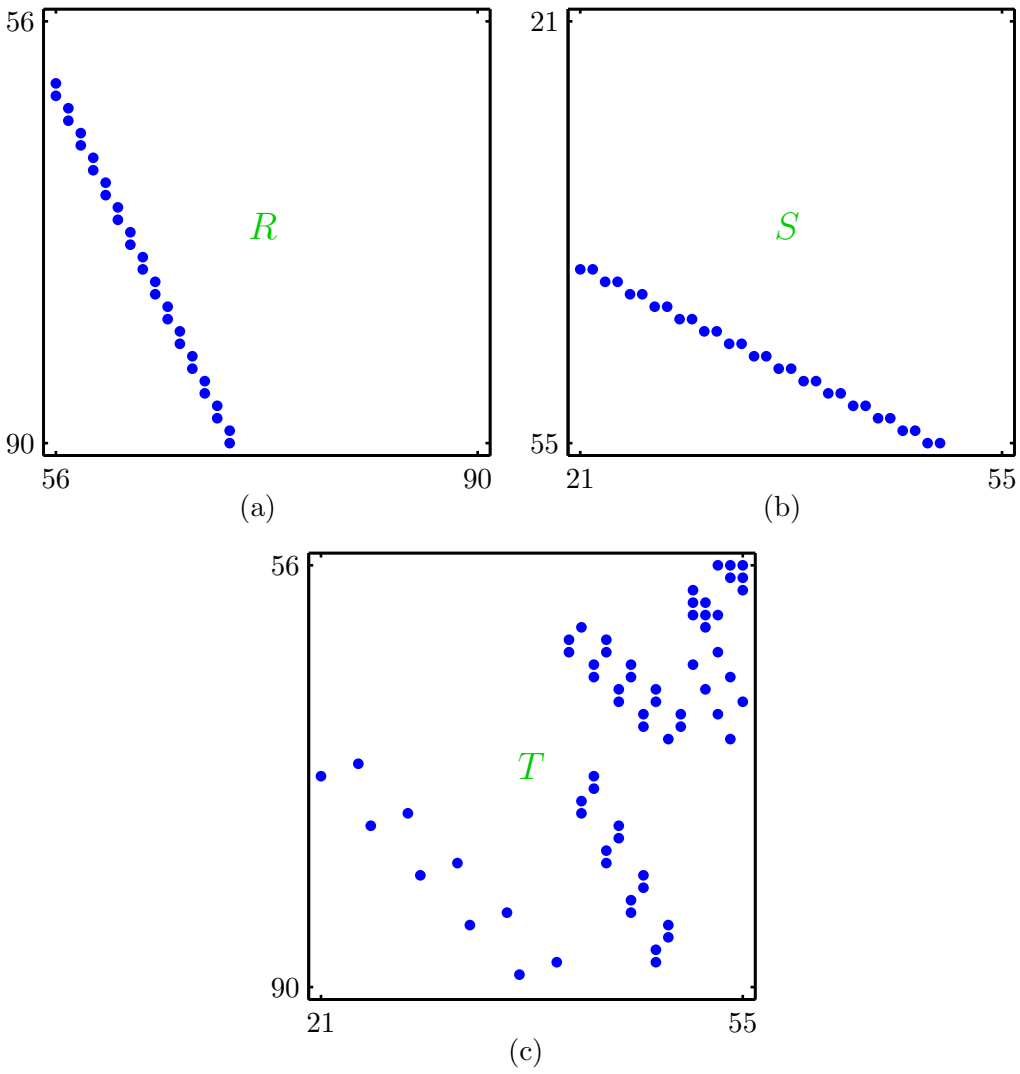


Figure 4.2: Sparsity patterns generated by the numbered dag in Figure 4.1. Each dot stands for a dense block, and the row and column labels are block indices into Ξ. (a) R. (b) S. (c) T.

Now all elements of the right-hand side vector are known, and we can solve the system for $\big[\boldsymbol{q};\boldsymbol{a};\boldsymbol{b}\big]$. In addition to finding the charges $\boldsymbol{q}$, this method simultaneously determines the coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$.

With the definitions

$$\Xi := \begin{bmatrix} D & & U \\ V & S-I & \\ & T & R-I \end{bmatrix}, \tag{4.4a}$$

$$\boldsymbol{x} := \big[\boldsymbol{q};\boldsymbol{a};\boldsymbol{b}\big], \tag{4.4b}$$

$$\boldsymbol{y} := \big[\boldsymbol{u};\boldsymbol{0};\boldsymbol{0}\big], \tag{4.4c}$$

the system (4.3) is $\Xi\boldsymbol{x} = \boldsymbol{y}$, to be solved in lieu of $\Phi\boldsymbol{q} = \boldsymbol{u}$. The effectiveness of working with Ξ instead of Φ remains to be demonstrated. The sparsity of Ξ is its asset, but Ξ is also larger in shape than Φ. These properties influence the computational burden in opposing directions.

I apply a direct sparse solver to (4.3). MATLAB has a built-in sparse solver [60], and several well-regarded libraries offer alternatives [36]. While this strategy is certainly convenient, its efficiency depends strongly on the ordering of the rows and columns of Ξ. Some row and column permutations are investigated in Section 4.2.

One major deficiency of this approach is that a generic sparse solver sacrifices important structure in Ξ. For high frequency scattering problems, iterative multipole must capitalize on the Toeplitz structure of the nonzero blocks of T, S, and R. The sparse solver treats those blocks as unstructured dense matrices. Clearly, a specialized sparse solver that does not ignore the Toeplitz structure is potentially more efficient in space and time.

The neglect of the Toeplitz block structure does not necessarily ensure that the direct solver will be uselessly slow. In the case of a matrix-vector product, the

Toeplitz structure is pivotal to the work reduction from $O(N^2)$ flops to $O(N \log N)$ flops. Ignoring this structure would make multipole a slow $O(N^2)$ algorithm. In contrast, Gaussian elimination requires $O(N^3)$ flops, so there is more room for improvement. Ignoring the Toeplitz structure in Ξ may not be ideal, but it does not prevent a reduction of the complexity from $O(N^3)$ to, say, $O(N^2)$.

As an alternative to a direct sparse solver, we could apply an iterative sparse solver to (4.3), and that would allow us to utilize all of the Toeplitz structure present. Such an approach is different from the multipole solvers of Chapter 3, because the Krylov space now captures approximations of the coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ as well as the charges $\boldsymbol{q}$.

Most intriguing is that, compared to the multipole matrix-vector product, the product on the left-hand side of (4.3) is better suited to a scalable implementation on a distributed parallel architecture. Assignment of the dag vertices in Figure 4.1 to processors is a typical scheduling problem, and balancing the load among the available processors is challenging. By introducing the intermediate variables $\boldsymbol{a}$ and $\boldsymbol{b}$ into the solution space, however, the dag collapses, giving all vertices the same scheduling priority.

The system (4.3) may benefit from preconditioning. The preconditioners for Φ in Chapter 3 were based on D, and those preconditioners can be reused in a preconditioner for Ξ. Moreover, if either U or V is erased from Ξ, then full advantage may be taken of the Toeplitz blocks in S, T, and R during the factorization of the remaining matrix. If V is deleted, then (4.3) becomes

$$\begin{bmatrix} D & & U \\ & S-I & \\ & T & R-I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \end{bmatrix}, \tag{4.5}$$

which is solved in two stages. First, solve

$$\begin{bmatrix} S-I & 0 \\ T & R-I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \end{bmatrix} \tag{4.6}$$

by block forward substitution. The block substitution encourages the use of fast Toeplitz matrix-vector products. Second, solve

$$D\boldsymbol{x}_1 = \boldsymbol{y}_1 - U\boldsymbol{x}_3 \tag{4.7}$$

using $\boldsymbol{x}_3$ from (4.6). An approximate solution should suffice for the preconditioner, so we might use an incomplete factorization of D. Another approach is to solve (4.7) with a subsidiary iteration.

If U is deleted instead of V, a similar method can be devised, and it is tempting to alternate between these two preconditioners at each step of the Krylov iteration.

4.2 Row and Column Permutations

We are free to rearrange the order of variables and equations in (4.3). With permutation matrices P and Q, an equivalent system is

$$(P\Xi Q^T)(Q\boldsymbol{x}) = P\boldsymbol{y}. \tag{4.8}$$

If sparse Gaussian elimination is used to compute a triangular factorization of $P\Xi Q^T$, then a careful choice of P and Q is essential, because these permutations have a strong influence on the amount of fill-in that occurs during the elimination process. If poor permutations are chosen, the sparse factorization can cost more than the dense factorization of Φ.

To see that, consider the symmetric block permutation

$$P = Q = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ I & 0 & 0 \end{bmatrix} \tag{4.9}$$

After this circular shift of both equations and variables, (4.3) becomes

$$\begin{bmatrix} S-I & & V \\ T & R-I & \\ & U & D \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{q} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{u} \end{bmatrix}. \tag{4.10}$$

We have so far specified only that all the variables $\{\boldsymbol{a}_i\}$ occur before any of the variables $\{\boldsymbol{b}_i\}$, and all of the variables $\{\boldsymbol{b}_i\}$ occur before any of the variables $\{\boldsymbol{q}_i\}$. Next we specify partial orderings within each block, maintaining the symmetry of the permutation.

Assemble $\{\boldsymbol{a}_i\}$ into the vector $\boldsymbol{a}$ by following a *postorder* of the source tree. A postorder is a topological order in which every child node must occur before its parent. The ordering of the source tree in Figure 4.1 is an example. With this order, $S - I$ is a sparse lower triangular matrix. Since S alone is strictly lower triangular, each diagonal element of $S - I$ is -1.

Assemble $\{\boldsymbol{b}_i\}$ into the vector $\boldsymbol{b}$ by following a *preorder* of the destination tree. A preorder is a topological order in which every parent node must occur before any of its descendants. The ordering of the destination tree in Figure 4.1 is an example. With this order, $R - I$ is a sparse lower triangular matrix. Since R alone is strictly lower triangular, each diagonal element of $R - I$ is -1.

Since $S - I$ and $R - I$ are triangular matrices with nonzero diagonal elements,

they are nonsingular. The following block factorization of $P\Xi P^T$ is thus possible:

$$\begin{bmatrix} S-I & & V \\ T & R-I & \\ & U & D \end{bmatrix} = \\ \begin{bmatrix} I & & \\ T(S-I)^{-1} & I & \\ & U(R-I)^{-1} & I \end{bmatrix} \begin{bmatrix} S-I & & V \\ & R-I & -T(S-I)^{-1}V \\ & & \widetilde{D} \end{bmatrix}, \quad (4.11)$$

where $\widetilde{D} := D + U(R-I)^{-1}T(S-I)^{-1}V$ is the Schur complement of the block 2×2 leading principal submatrix $\left[\begin{smallmatrix} S-I & 0 \\ T & R-I \end{smallmatrix}\right]$.

After substituting this factorization, (4.10) reduces to the equivalent block triangular linear system,

$$\begin{bmatrix} S-I & & V \\ & R-I & -T(S-I)^{-1}V \\ & & \widetilde{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{q} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{u} \end{bmatrix}. \quad (4.12)$$

The first step in solving this system by block backward substitution is to solve $\widetilde{D}\boldsymbol{q} = \boldsymbol{u}$. Compare this to the original dense system $\Phi\boldsymbol{q} = \boldsymbol{u}$. Clearly, $(\widetilde{D} - \Phi)\boldsymbol{q} = \mathbf{0}$. If we exclude the trivial right-hand side $\boldsymbol{u} = \mathbf{0}$, then $\boldsymbol{q} \neq \mathbf{0}$ and $\widetilde{D} = \Phi$.

We conclude that block Gaussian elimination with the chosen row and column permutation is inefficient. Partway through the computation, we are led back to the system $\Phi\boldsymbol{q} = \boldsymbol{u}$, so the sparse augmented system has only generated more work for us.

An artifact of this analysis is a factorization of the interaction matrix:

$$\Phi = \widetilde{D} = D + U(I-R)^{-1}T(I-S)^{-1}V. \quad (4.13)$$

Since R and S are nilpotent, simple formulas exist for the two inverses in (4.13). If the source tree has μ levels, then $S^\mu = 0$ and

$$(I-S)^{-1} = I + S + S^2 + \cdots + S^{\mu-1}. \quad (4.14)$$

Likewise, if the destination tree has ν levels, then $R^\nu = 0$ and

$$(I - R)^{-1} = I + R + R^2 + \cdots + R^{\nu-1}. \tag{4.15}$$

Using (4.13) to compute $\Phi\boldsymbol{q}$ for a known charge vector $\boldsymbol{q}$ reproduces precisely those steps performed by the fast multipole algorithm.

We have so far considered only two orderings of the rows and columns. Another symmetric permutation that leads to a smaller amount of fill-in is

$$\begin{bmatrix} D & U & \\ & R-I & T \\ V & & S-I \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{b} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \tag{4.16}$$

where variables $\boldsymbol{a}$ are preordered and variables $\boldsymbol{b}$ are postordered on their respective multipole trees. Let Π be the permutation that transforms (4.3) into (4.16). With this ordering, R and S are strictly upper triangular matrices.

I have experimented with various other "natural" orderings and with elaborations on this theme, for instance by interleaving the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$. All other hand-picked permutations have proved inferior to (4.16). That may be due to lack of imagination on my part, and a more inspired choice may embarrass this one.

The sparsity pattern in (4.16) is strongly nonsymmetric—look ahead to Figure 4.6(a)—which suggests that distinct row and column permutations may produce superior orderings. Some sparse solvers capitalize on near structural symmetry [43] [51]. If, however, Ξ originates from an integral equation of the second kind, then another observation suggests that the permutation should be symmetric.

The element of largest modulus in each column of Ξ typically sits on the main diagonal. (Since any Toeplitz structure in R, T, and S is ignored anyway, I use the stabilized basis of Chapter 5, which does not produce large elements in T.) We probably do not want to move these large elements off the main diagonal, because

to ensure numerical stability the solver's pivoting algorithm will tend to move them back. Indeed, for general sparse systems a preliminary computation may be carried out to permute large elements to the main diagonal [44].

Symmetric permutations $P\Xi P^T$ always exchange one diagonal element for another. Unfortunately, the restriction $P = Q$ does not much simplify the choice of permutation.

For a general sparse $N \times N$ matrix, no algorithm of polynomial complexity in N is known for the determination of row and column permutations that minimize the fill-in generated by Gaussian elimination [42]. All codes for the automatic reordering of general sparse matrices are based on suboptimal heuristics.

Nevertheless, the heuristics are sophisticated and can be quite effective. Two common approaches are approximate minimum degree (AMD) [4] [37] and nested dissection [59] [100], and either technique substantially improves on the permutation expressed in (4.16). MATLAB has several AMD implementations, including `colamd`, but it has no function to perform nested dissection. I have used, with much success, the freely available C library METIS [91]. The public-domain MATLAB toolbox MESHPART includes an interface to METIS. It also has routines that implement several other nested dissection methods.

4.2.1 Example: Scattering from a Spiral

To demonstrate the direct multipole solver, we apply it to the scattering problem illustrated in Figure 4.3. The obstacle is a spiral segment, which, as we saw in Chapter 3, is intractable to an iterative multipole solver. This spiral has one more turn than the spiral of Figures 3.10(d) and 3.16, so it is expected to pose an even greater challenge.

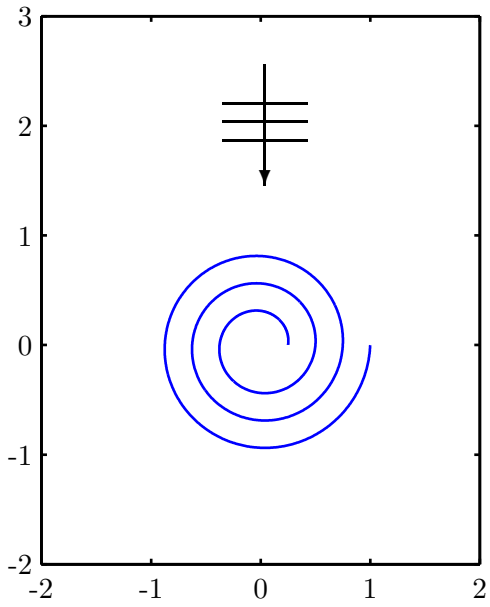


Figure 4.3: A plane wave $u_{\text{inc}}(x, y) = e^{-iky}$ illuminates a section of an Archimedes spiral. The boundary is described in polar coordinates (r, ϕ) by $r = \phi/8\pi$ for $\phi \in [2\pi, 8\pi]$.

The EFIE is discretized by a piecewise product rule of order 32, everywhere maintaining a local boundary grid density of at least 4 points per wavelength $2\pi/k$. For smooth curves, those specifications are sufficient to limit the discretization error to $\epsilon \approx 10^{-3}$.

With $k = 100$, the spiral is automatically partitioned into the cluster hierarchy shown in Figure 4.4. The hierarchy is deep only at the boundary end points. The current density is singular at those points, but the quadrature rule is not designed to accommodate such solution singularities. A common way to improve the solution accuracy is to force the grid density to be higher near the singularities. In this case, after the initial tree generation is finished, the arcs contained in the two leaf disks at the spiral end points are each divided into two child subarcs. Then the new end leaves are divided again, and then again.

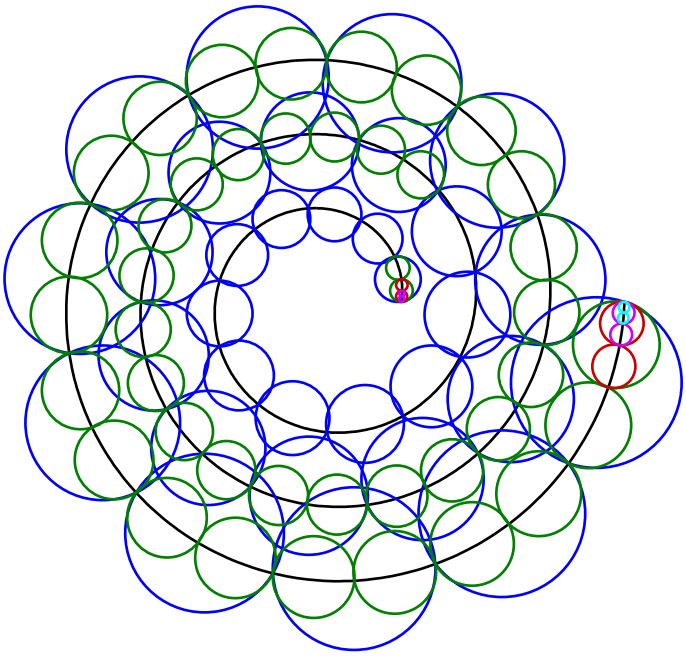


Figure 4.4: Cluster hierarchy generated for the spiral of Figure 4.3 with $k = 100$. The boundary mesh is refined to an extra three levels of depth at the spiral end points.

The dense interaction matrix $\Phi \in \mathbb{C}^{1856\times 1856}$ is displayed in Figure 4.5(a). Note that the matrix appears stretched along its four borders, a consequence of the mesh grading at the end points.

The task of constructing a fast solver is made formidable by the rapid oscillations in the matrix elements. Although MATLAB's entire `jet` colormap has been utilized, the contrast in Figure 4.5(a) is poor because the deep reds and blues that mark the real elements of largest absolute value are packed into a thin band surrounding the main diagonal. To improve the contrast, the `jet` colormap has been reapplied to a submatrix of Φ in Figure 4.5(b), and the oscillations are more clearly visible.

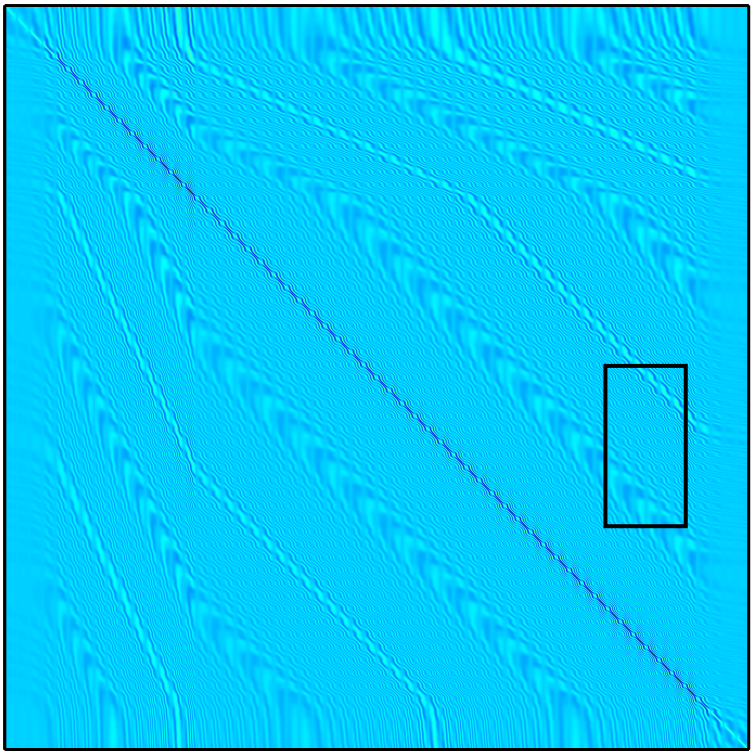
Instead of working with Φ, the approach of (4.16) is to work with $\Pi\Xi\Pi^T$. Its sparsity pattern is plotted in blue in Figure 4.6(a). Also shown is a green pattern

of fill-in produced by Gaussian elimination. The triangular factors of $\Pi\Xi\Pi^T$ have 4,176,522 more nonzero elements than $\Pi\Xi\Pi^T$ itself.

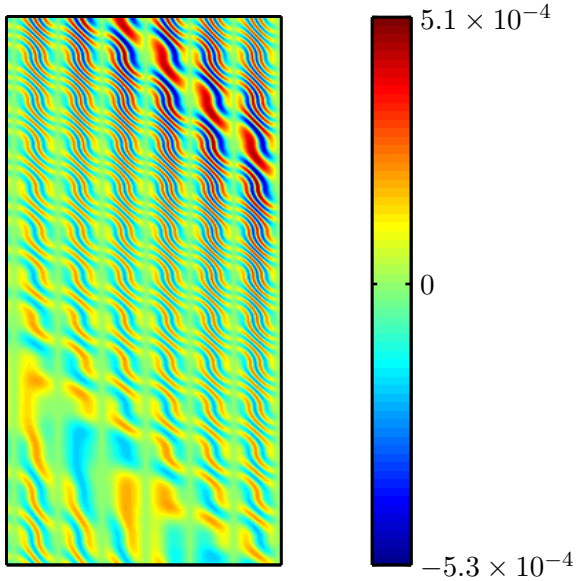
The unknown currents $\boldsymbol{q}$ have been ordered consecutively along the spiral. In the block D of short-range interactions, two fringing bands of nonzero elements run nearly parallel to the main band. That sparsity pattern resembles the one produced by a low-order finite difference discretization of the Helmholtz equation on a domain containing the spiral, assuming the domain is filled with a uniform rectangular grid and the grid points are enumerated in a natural lexicographic order. With respect to the structure of D, this boundary is more nearly two-dimensional than one-dimensional. As k is increased, though, the optical distance between the spiral turns increases. The numerical dimension of the boundary decreases to match its true dimension, and the fringes disappear.

Figure 4.6(b) shows the improvement possible with automatic reordering of the rows and columns of Ξ. It shows the sparsity pattern of $P\Xi P^T$ and its triangular factors, where P has been computed by passing $\Xi+\Xi^T$ to a nested dissection routine in the METIS library. Now the factors consume only 2,062,574 additional nonzero elements, a 50 percent reduction of the fill-in generated by the "natural" ordering of Figure 4.6(a).

The flexibility of choosing a good permutation for the larger sparse system turns out to be sufficient to produce a superior direct solver. Before showing some timings, however, I describe in the next section a compression technique designed to make Ξ as small as possible. That compression has already been applied to Figure 4.6, and it can dramatically speed up the computation.

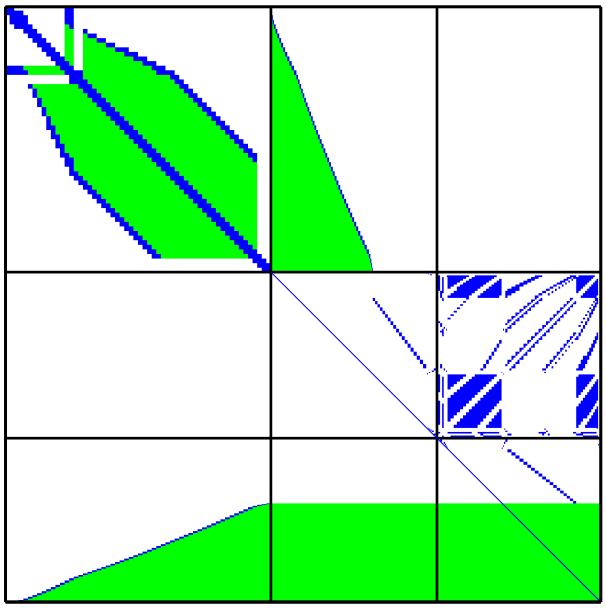


(a)

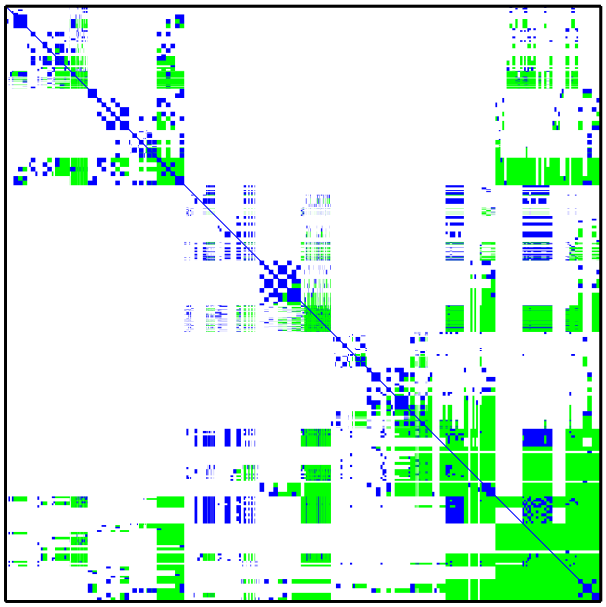


(b)

Figure 4.5: (a) Real part of 1856×1856 interaction matrix Φ at $k = 100$. (b) Real part of the outlined 400×200 submatrix.



(a)



(b)

Figure 4.6: (a) Sparsity pattern of the 4160 × 4160 matrix $\Pi\Xi\Pi^T$ and its triangular factors. The black lines partition the matrix according to (4.16). The block D is the same size as the matrix in Figure 4.5(a). The nonzero main diagonal is difficult to see at this scale, so it has been drawn thickly. (b) Symmetric permutation produced by nested dissection. Again, the main diagonal is drawn thickly to emphasize that no zeros occur there.

4.3 Compressing the Sparse System

It is possible to reduce both the order of the sparse matrix Ξ and its number of nonzero elements. This compression is possible because the multipole field representations (2.10) and (2.28) show some redundancy. While those spectral expansions are efficient for all boundaries, they are suboptimal for any particular boundary.

The suboptimality can be appreciated by considering how the multipole algorithm uses the exterior expansion of a root charge cluster. In polar coordinates (r, ϕ), the field produced by sources inside a disk G_0 with center $r = 0$ and radius α is approximated for $r > 2\alpha$ by a weighted sum of functions $\{H_m(kr)e^{im\phi}\}$. Assume that p of these basis functions are needed to achieve a specified accuracy. Inside each member of the set of well-separated disks $\{G_j\}$ in G_0's interaction list, the exterior expansion is converted into an interior expansion in a local coordinate system (r_j, ϕ_j) attached to disk G_j's center. Because G_0 is a root disk, its exterior expansion serves no further purpose. That expansion, constructed to approximate the field at *all* points well separated from the disk G_0, is actually needed only at points inside $\bigcup_j G_j$. If this collection of disks subtends a small angle with respect to the center of G_0, the field inside $\bigcup_j G_j$ can have fewer than p degrees of freedom.

Rank revealing decompositions of the translation matrices stored in Ξ can detect such a situation and substitute a shorter expansion. That approach is effective not only for exterior expansions of root clusters, but for all interior and exterior expansions.

The details of the compression are illustrated with the assistance of Figure 4.7, which shows two subgraphs of a multipole dag. In Figure 4.7(a), edges lead from the vertex labeled $\boldsymbol{a}_i$ to the three vertices $\boldsymbol{a}_p$, $\boldsymbol{b}_f$, and $\boldsymbol{b}_g$, indicating the flow of an exterior expansion from a leaf node i of the source tree to its parent node p and to

interior expansions at nodes f and g of the destination tree.

Except for $-I$ on the block diagonal, the weight matrices associated with these edges are the only nonzero entries in the block column of Ξ associated with node i. They can be stacked together as

$$\begin{bmatrix} \boldsymbol{a}_p \\ \boldsymbol{b}_f \\ \boldsymbol{b}_g \end{bmatrix} = \begin{bmatrix} S_i \\ T_{f,i} \\ T_{g,i} \end{bmatrix} \boldsymbol{a}_i =: A\boldsymbol{a}_i, \tag{4.17}$$

where $A \in \mathbb{C}^{m\times n}$ has been introduced to simplify later expressions. We plan to approximate A with a matrix of smaller rank.

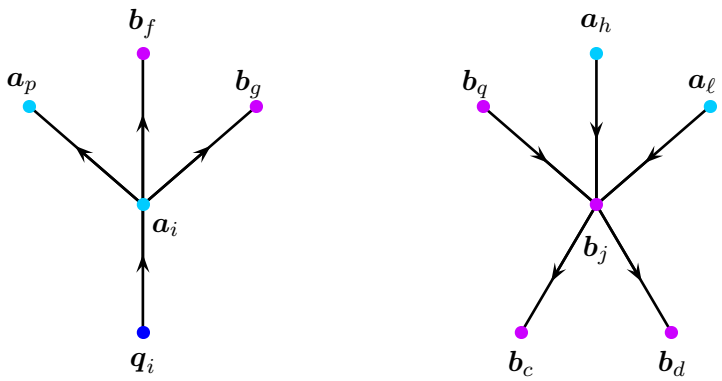


Figure 4.7: Two subgraphs of a multipole dag. (a) Subgraph containing all edges connected to the vertex $\boldsymbol{a}_i$. (b) Subgraph containing of all edges connected to the vertex $\boldsymbol{b}_j$.

If $m \geq n$, then A has a singular value decomposition (SVD) $A = Q_L \Sigma Q_R^H$, where $Q_L \in \mathbb{C}^{m\times n}$ has orthonormal columns, $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix of singular values $\{\sigma_i\}$, and $Q_R \in \mathbb{C}^{n\times n}$ is a unitary matrix of right singular vectors. This reduced SVD is computed in MATLAB with the function call `svd(A,0)`. We assume that the singular values are produced in nonincreasing order, so $\sigma_i \geq \sigma_j$ if $i < j$.

Given a relative accuracy ϵ, the numerical rank of A is defined to be the smallest

number r such that $\sigma_i < \epsilon\sigma_1$ for $i > r$. To this accuracy, a low-rank approximation $\widehat{A}$ of A is obtained by truncating its singular value spectrum at the length r. Let $\widehat{Q}_L$ and $\widehat{Q}_R$ be the leading r columns of Q_L and Q_R, respectively, and let $\widehat{\Sigma}$ be the $r \times r$ leading principal submatrix of Σ. Then the approximation is $\widehat{A} = \widehat{Q}_L \widehat{\Sigma} \widehat{Q}_R^H$.

Now partition $\widehat{Q}_L$ into three block rows conforming with the original row partition of A. Then (4.17) becomes

$$\begin{aligned} \begin{bmatrix} \boldsymbol{a}_p \\ \boldsymbol{b}_f \\ \boldsymbol{b}_g \end{bmatrix} &= \widehat{Q}_L (\widehat{\Sigma} \widehat{Q}_R^H \boldsymbol{a}_i) \\ &= \begin{bmatrix} \widehat{S}_i \\ \widehat{T}_{f,i} \\ \widehat{T}_{g,i} \end{bmatrix} \hat{\boldsymbol{a}}_i, \end{aligned} \tag{4.18}$$

where the new exterior coefficients of node i are $\hat{\boldsymbol{a}}_i := \widehat{\Sigma} \widehat{Q}_R^H \boldsymbol{a}_i$, and the new translation operators are the indicated block rows of $\widehat{Q}_L$.

Although the new translation operators $\widehat{S}_i$, $\widehat{T}_{f,i}$, and $\widehat{T}_{g,i}$ are smaller, any Toeplitz structure is lost. Since I am not using a sparse solver that can utilize such structure anyway, in my code I compress all translation operators.

The net effect of this manipulation is to shorten the vector $\boldsymbol{a}_i$ from a length of n to a length of r, and to simultaneously compress the translation matrices from n columns to r columns. The latter reduces the width of the associated block column of Ξ from n to r, an alteration that makes Ξ nonsquare.

Now the change to $\boldsymbol{a}_i$ must be propagated to the graph edges *incident* on node i, after which Ξ will become square again.

In the subgraph of Figure 4.7(a), there is but a single incident edge, corresponding to the operation $\boldsymbol{a}_i = V_i \boldsymbol{q}_i$. Except for $-I$ on the block diagonal, V_i is the only nonzero entry in the block row of Ξ associated with node i of the source tree. In

the new basis, this operation is

$$\begin{aligned} \hat{\boldsymbol{a}}_i &= \widehat{\Sigma}\widehat{Q}_R^H \boldsymbol{a}_i \\ &= (\widehat{\Sigma}\widehat{Q}_R^H V_i)\boldsymbol{q}_i \\ &= \widehat{V}_i \boldsymbol{q}_i, \end{aligned} \tag{4.19}$$

where $\widehat{V}_i := \widehat{\Sigma}\widehat{Q}_R^H V_i$ is the new leaf operator. It has r rows, and the height of the associated block row of Ξ is thus reduced from n to r.

We have compressed the sparse system Ξ by $n - r$ rows and columns. The only change to the multipole dag is to replace the data vector at vertex i, together with the matrix operators weighting all edges connected to that vertex. Since the change is confined to a small subgraph, further compression of Ξ is possible by repeating this procedure—in parallel, perhaps—for other vertices of the dag.

A similar procedure first compresses a block row of Ξ and propagates the compression to a block column. Figure 4.7(b) shows another subgraph of the multipole dag. The vertex labeled $\boldsymbol{b}_j$ stores the interior expansion coefficients for node j of the destination tree. Three edges feed signals to this vertex, and another two edges carry the data $\boldsymbol{b}_j$ to node j's children, where it is translated into new interior expansion coefficients.

Excluding the matrix $-I$ on the block diagonal, the nonzero entries of the block row of Ξ associated with node j can be collected as

$$\boldsymbol{b}_j = \begin{bmatrix} R_j & T_{j,h} & T_{j,\ell} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix} =: A \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix}, \tag{4.20}$$

where again for convenience the temporary matrix $A \in \mathbb{C}^{m \times n}$ has been introduced. It is obviously not the same as A in (4.17).

Now we assume $m < n$, so that the reduced SVD is $A = Q_L \Sigma Q_R^H$, where $Q_L \in$

$\mathbb{C}^{m\times m}$ is unitary and $Q_R \in \mathbb{C}^{n\times m}$ has orthonormal columns. After truncating the singular values, we have an approximation $\widehat{A} = \widehat{Q}_L \widehat{\Sigma} \widehat{Q}_R^H$ of rank r. Partitioning $\widehat{Q}_R^H$ into three block columns, conformal with the partition of A, (4.20) is replaced with

$$\begin{aligned} \boldsymbol{b}_j &= \widehat{Q}_L \widehat{\Sigma} \widehat{Q}_R^H \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix} \\ &= \widehat{Q}_L \widehat{\Sigma} \begin{bmatrix} \widehat{R}_j & \widehat{T}_{j,h} & \widehat{T}_{j,\ell} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix}, \end{aligned} \tag{4.21}$$

where new translation matrices have been introduced as the block columns of $\widehat{Q}_R^H$. Define the new vector of interior coefficients $\hat{\boldsymbol{b}}_j$ implicitly by

$$\boldsymbol{b}_j = \widehat{Q}_L \widehat{\Sigma} \hat{\boldsymbol{b}}_j. \tag{4.22}$$

Then (4.21) becomes

$$\hat{\boldsymbol{b}}_j = \begin{bmatrix} \widehat{R}_j & \widehat{T}_{j,h} & \widehat{T}_{j,\ell} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix}. \tag{4.23}$$

The length of $\hat{\boldsymbol{b}}_j$ is r, and each of the new translation matrices has r rows instead of m.

The edges leaving the vertex carry out the operations $\boldsymbol{b}_c = R_c \boldsymbol{b}_j$ and $\boldsymbol{b}_d = R_d \boldsymbol{b}_j$. Substituting (4.22), these become

$$\boldsymbol{b}_c = (R_c \widehat{Q}_L \widehat{\Sigma}) \hat{\boldsymbol{b}}_j = \widehat{R}_c \hat{\boldsymbol{b}}_j, \tag{4.24a}$$

$$\boldsymbol{b}_d = (R_d \widehat{Q}_L \widehat{\Sigma}) \hat{\boldsymbol{b}}_j = \widehat{R}_d \hat{\boldsymbol{b}}_j, \tag{4.24b}$$

in which new branch operators $\widehat{R}_c := R_c \widehat{Q}_L \widehat{\Sigma}$ and $\widehat{R}_d := R_d \widehat{Q}_L \widehat{\Sigma}$ have been identified.

Although these new branch operators already have fewer columns, further compression can be achieved through the same manipulations applied to the edges leav-

ing the central vertex of Figure 4.7(a). Concatenating $\widehat{R}_c$ and $\widehat{R}_d$ as in (4.17) gives

$$\begin{bmatrix} \boldsymbol{b}_c \\ \boldsymbol{b}_d \end{bmatrix} = \begin{bmatrix} \widehat{R}_c \\ \widehat{R}_d \end{bmatrix} \hat{\boldsymbol{b}}_j. \tag{4.25}$$

By truncating the SVD of the block matrix on the right-hand side, we obtain an approximation $\widetilde{Q}_L \widetilde{\Sigma} \widetilde{Q}_R^H$ with rank $s \leq r$. With a row partition of $\widetilde{Q}_L$, conformal with the partition of the left-hand side,

$$\begin{aligned} \begin{bmatrix} \boldsymbol{b}_c \\ \boldsymbol{b}_d \end{bmatrix} &= \widetilde{Q}_L (\widetilde{\Sigma} \widetilde{Q}_R^H \hat{\boldsymbol{b}}_j) \\ &= \begin{bmatrix} \widetilde{R}_c \\ \widetilde{R}_d \end{bmatrix} \tilde{\boldsymbol{b}}_j, \end{aligned} \tag{4.26}$$

where $\tilde{\boldsymbol{b}}_j := \widetilde{\Sigma} \widetilde{Q}_R^H \hat{\boldsymbol{b}}_j$ is the second compression of the vertex data, reducing its length from r to s. The updated translation matrices $\widetilde{R}_c$ and $\widetilde{R}_d$ have s columns each.

This compression must be propagated to the translation matrices weighting the incident edges, so that Ξ remains square. The update follows the pattern of (4.19),

$$\begin{aligned} \tilde{\boldsymbol{b}}_j &= \widetilde{\Sigma} \widetilde{Q}_R^H \hat{\boldsymbol{b}}_j \\ &= \widetilde{\Sigma} \widetilde{Q}_R^H \begin{bmatrix} \widehat{R}_j & \widehat{T}_{j,h} & \widehat{T}_{j,\ell} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{R}_j & \widetilde{T}_{j,h} & \widetilde{T}_{j,\ell} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_q \\ \boldsymbol{a}_h \\ \boldsymbol{a}_\ell \end{bmatrix}, \end{aligned} \tag{4.27}$$

where the translation matrices $\widetilde{R}_j$, $\widetilde{T}_{j,h}$, and $\widetilde{T}_{j,\ell}$ now have s rows each.

To review the steps applied to the subgraph in Figure 4.7(b), we first obtained some compression by truncating the SVD of the entering edge matrices, which also modified the exiting edge matrices. Then we obtained more compression by truncating the SVD of the exiting edge matrices, which in turn modified the entering edge matrices. Further compression is *not* possible by iterating this procedure, since the

block of new incident edge matrices,

$$\left[\begin{array}{ccc} \widetilde{R}_j & \widetilde{T}_{j,h} & \widetilde{T}_{j,\ell} \end{array} \right] = \widetilde{\Sigma}\widetilde{Q}_R^H\widehat{Q}_R^H, \tag{4.28}$$

is already a reduced SVD. The columns of the product $\widehat{Q}_R\widetilde{Q}_R$ are necessarily orthonormal, since the columns of each factor are orthonormal.

We can replace the SVD with a cheaper decomposition, such as a rank revealing QR factorization [20] [24] [73]. MATLAB does not offer that factorization, but it does have the column-pivoted QR factorization $AP = QL^H$, which is triggered by calling `qr(A,0)` with three output arguments. The main diagonal elements of the upper triangular factor L^H are returned in order of decreasing modulus. By applying a threshold to those elements, some trailing rows of L^H and trailing columns of Q are discarded. The result is a low-rank approximation $\widehat{A} = \widehat{Q}\widehat{L}^H P^T$ in which P is a permutation matrix, L^H is upper trapezoidal, and the columns of $\widehat{Q}$ are orthonormal.

Although the asymptotic complexity of the SVD matches the complexity of QR, if A is large the QR factorization is faster by a nontrivial length of time. The trade-off is that truncating the column-pivoted QR factorization can overestimate the numerical rank of A, and it will not usually achieve quite as much compression as the SVD. Since the resulting sparse system is larger, the time savings realized by QR compression will in part be erased by the extra work required to factor Ξ. For that reason, in my code I use QR compression only when A is large.

The QR factorization $\widehat{Q}\widehat{L}^H P^H$ is an appropriate substitution for the SVD in the block column compression (4.18). In the block row compression (4.21), however, $\widehat{Q}$ should appear on the right-hand side of the factorization. A row-pivoted LQ factorization $PA = LQ$, easily constructed in MATLAB from the QR factorization of A^H, is a suitable alternative to the SVD.

If instead a QR factorization were used in (4.21), then in (4.24) R_c and R_d would be right-multiplied by only $\widehat{Q}$. It is the presence of $\widehat{\Sigma}$ in the case of the SVD, or $\widehat{L}$ in the case of LQ, that promotes a rank reduction. Furthermore, the usage of only QR—or only LQ—factorizations will invalidate the given argument that iterating the compression of entering and exiting edges is superfluous.

Table 4.1 shows the compression achieved for the spiral of Figure 3.10(d). Details of the discretization are provided in the following section.

Table 4.1: Compression of the Spiral

		Uncompressed Ξ		Compressed Ξ	
k	N	Order	Nonzeros	Order	Nonzeros
$2^{11/2}$	576	$2.87N$	$0.886N^2$	$1.62N$	$0.499N^2$
$2^{12/2}$	736	$3.45N$	$0.810N^2$	$1.85N$	$0.362N^2$
$2^{13/2}$	896	$4.08N$	$0.849N^2$	$2.08N$	$0.299N^2$
$2^{14/2}$	1344	$4.68N$	$0.665N^2$	$2.30N$	$0.172N^2$
$2^{15/2}$	1728	$5.38N$	$0.678N^2$	$2.54N$	$0.140N^2$
$2^{16/2}$	2624	$5.84N$	$0.515N^2$	$2.70N$	$0.098N^2$
$2^{17/2}$	3392	$6.56N$	$0.532N^2$	$2.94N$	$0.089N^2$
$2^{18/2}$	5152	$6.91N$	$0.417N^2$	$3.05N$	$0.065N^2$
$2^{19/2}$	6688	$7.69N$	$0.443N^2$	$3.30N$	$0.061N^2$

4.4 Direct Solver Performance

4.4.1 Frequency Scaling

To demonstrate the efficiency of the direct multipole solver, we compute the plane wave scattering from the $2\frac{1}{4}$-turn spiral of Figure 3.10(d), which is slightly less challenging than the spiral of 3 turns used in Section 4.2.1. In that section, the boundary

mesh was refined three times at the spiral end points. The results of this section are computed with only a single mesh refinement, but the accuracy specification is improved to $\epsilon = 10^{-4}$. Using order-32 rules, 6 grid points per wavelength are needed to achieve that accuracy. To show how the costs grow with N, the scattering problem is solved for a sequence of wavenumbers $k \in \{2^{j/2} : 11 \leq j \leq 21\}$.

Three solvers are compared:

- Gaussian elimination applied to the $N \times N$ dense system $\Phi \boldsymbol{q} = \boldsymbol{u}$
- Iterative multipole applied to the same dense system
- Direct multipole applied to the sparse system $\Xi \boldsymbol{x} = \boldsymbol{y}$

The Krylov solver used for the iterative solution is unrestarted GMRES with no preconditioning. The sparse factorization used for the direct solution is MATLAB's built-in sparse `lu`, which is passed a symmetric permutation of Ξ returned by METIS.

Table 4.2 records the time required by each method. The time reported for Gaussian elimination includes the time to fill the matrix Φ, and the time reported for iterative multipole includes the time spent initializing the multipole data structures after the cluster tree has been constructed. Neither of the last two columns includes the time needed to build the multipole tree, but that cost is negligible in comparison to the expense of the other operations. The figures for the direct multipole solver include the time to fill, compress, and reorder the matrix Ξ.

The times reported are wall clock times, but with the exception of four entries those times are nearly the same as CPU times. At $N = 3392$, Φ fits comfortably into physical memory. At $N = 5152$, however, the dense Gaussian elimination solver already exhausts virtual memory. The memory requirement of either multipole method grows more slowly, so they are each able to make profitable use of the available virtual memory for two large problems. The wall clock times for those

Table 4.2: Solution Time (s) for the Spiral

		Multipole	
N	Gauss	Iterative	Direct
576	4.44	10.93	6.10
736	7.03	17.09	7.90
896	10.44	28.89	11.36
1344	24.04	52.33	22.06
1728	41.03	86.57	37.28
2624	104.08	151.94	61.31
3392	187.36	332.59	120.78
5152		862.21	$240.99 + 3.11$
6688		1151.55	$492.87 + 18.00$
10272		$3472.76 + 12.21$	
13312		$3465.10 + 37.12$	

problems are reported as the sum of the user CPU time and the system time required to manage virtual memory.

Clearly, for this boundary the iterative multipole solver is inferior in execution time to even dense Gaussian elimination. It is, though, able to treat larger problems than either of the other solvers. That lower memory requirement is evident in Table 4.3.

The direct multipole solver becomes faster than dense Gaussian elimination at some point between $N = 896$ and $N = 1344$. It is not faster by a startling amount, but the sparse LU factorization in MATLAB is also not particularly fast. The multifrontal code in UMFPACK [36], for example, typically performs the sparse factorization much more quickly. The dense LU factorization in MATLAB issues a call to LAPACK [9], and it is unreasonable to expect much improvement from any alternatives. If the fastest available sparse solver were compared to the fastest available dense solver, the advantage of the direct multipole method displayed in Table 4.2

would grow.

Table 4.3: Memory (MB) Consumed by the Spiral

		Multipole	
N	Gauss	Iterative	Direct
576	5.31	4.88	7.10
736	8.67	5.88	10.02
896	12.85	7.47	14.85
1344	28.90	10.20	28.35
1728	47.78	14.19	42.15
2624	110.17	24.34	69.63
3392	184.09	43.91	116.13
5152		85.99	206.56
6688		107.40	343.95
10272		215.92	
13312		239.28	

Figure 4.8 marshals some evidence that the complexity of the direct multipole solver is less than the complexity of dense Gaussian elimination. Only the asymptotically dominant cost of those two methods has been plotted. The $O(N^2)$ time needed to fill Φ is not included, but the time spent in compressing Ξ *is* included.

Fitting the data on this logarithmic graph to a straight line produces an experimental complexity estimate of aN^b flops. The cost of the sparse solver is hard to assess theoretically, since it depends strongly on the details of the sparsity pattern. As demonstrated in Section 4.2, a poor choice of row and column permutations can make b as large as 3, irrespective of any compression of Ξ. The data of Figure 4.8 indicate, however, that a good ordering makes $b < 3$ possible.

Since the spectral expansion of a root cluster has length $O(N)$, the translation matrices $\{T_{ij}\}$ at the top of the multipole tree have $O(N^2)$ elements. Therefore, since those translation matrices are stored as dense blocks in Ξ, without compression

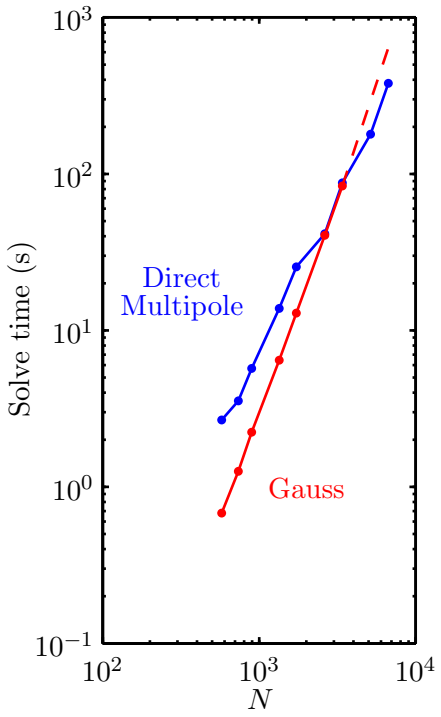


Figure 4.8: Comparison of the asymptotically dominant portion of the solution time for the direct solvers. A least-squares fit to the dense Gaussian elimination data has a slope of 2.72, while a fit to the multipole data improves that slope to 2.01.

the sparse matrix has $\Omega(N^2)$ elements. Unless compression can reduce the storage complexity, the solver will require $\Omega(N^2)$ flops. Figure 4.8 suggests that this lower bound is achievable.

Compared to iterative multipole, the memory consumption of direct multipole is its chief liability. Two factors inflate the memory requirements:

- Toeplitz blocks $\{S_i\}$, $\{T_{ij}\}$, and $\{R_i\}$ are stored explicitly in Ξ, and some of those blocks are large.
- Gaussian elimination partially fills in the sparsity pattern of Ξ.

Iterative multipole can solve problems an order of magnitude larger. For the scattering from a unit circle, Table 4.4 shows the largest problems that either method

can solve before exhausting memory on my PC workstation.

Table 4.4: Circles of Maximum Optical Size

Multipole Solver	Maximum k	Maximum N
Direct	600	4096
Iterative	10,000	65,536

4.4.2 Accuracy Scaling

A different way to grow N is to increase the number of grid points applied to the boundary at a fixed frequency, something that is done to increase the accuracy of the computed solution.

The complexity of the solution by dense Gaussian elimination depends only on N. It is the same for either frequency scaling or accuracy scaling. The complexity of the direct multipole solver, on the other hand, does depend on how N is increased.

To study the direct multipole performance under accuracy scaling, we solve the scattering problem depicted in Figure 4.9. The incident field has wavenumber $k = 100$. The boundary is discretized with order-32 quadrature rules at a density of 16 points per wavelength, which is sufficient on a smooth boundary to obtain a relative accuracy of $\epsilon = 10^{-10}$. This boundary, however, has three corners. To reach the desired level of accuracy, the mesh must be refined by adding grid points near those points. The reported data has been collected for a sequence of boundary grids that are successively finer at the corners.

Table 4.5 shows how the measured solution accuracy $\hat{\epsilon}$ improves as N increases. (To be precise, $\hat{\epsilon}$ is a backward error, determined by averaging the logarithm of

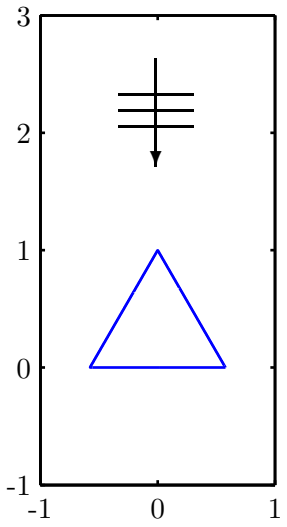


Figure 4.9: A plane wave incident from the angle $\pi/2$ scatters from a perfectly conducting equilateral triangle.

the measured residual over the curve Γ.) The table also lists the size of the sparse matrix Ξ after basis compression. Two measures of its size are given: its order, and the number of nonzero entries in its triangular factors.

The time and space complexities of direct multipole and dense Gaussian elimination are compared in Figure 4.10. The direct multipole method is plainly superior. Its complexity appears to be at most linear in both time and space.

Table 4.6 contains the timing information displayed in Figure 4.10(a), and it also includes the time required by an iterative multipole method. The solver is unrestarted GMRES, preconditioned by the incomplete LU factorization of $\widehat{\Phi}$ with a drop tolerance of 10^{-3}.

The table entries in parentheses are extrapolated values. Dense Gaussian elimination runs out of memory at $N = 3456$. In contrast, neither multipole solver has any trouble fitting these problems into memory.

Table 4.5: Accuracy and Size Measurements for the Triangle

		$\Xi = LU$	
N	$\hat{\epsilon}$	Order	$\texttt{nnz}(L+U)$
1536	$10^{-4.98}$	$2.62N$	$0.56N^2$
1920	$10^{-5.67}$	$2.57N$	$0.41N^2$
2304	$10^{-6.51}$	$2.51N$	$0.32N^2$
2688	$10^{-7.39}$	$2.48N$	$0.26N^2$
3072	$10^{-8.22}$	$2.44N$	$0.22N^2$
3456	$10^{-8.93}$	$2.41N$	$0.19N^2$
3840	$10^{-9.15}$	$2.39N$	$0.18N^2$
4224	$10^{-9.25}$	$2.37N$	$0.15N^2$
4608	$10^{-9.32}$	$2.35N$	$0.14N^2$
4992	$10^{-9.37}$	$2.34N$	$0.12N^2$
5376	$10^{-9.42}$	$2.33N$	$0.11N^2$

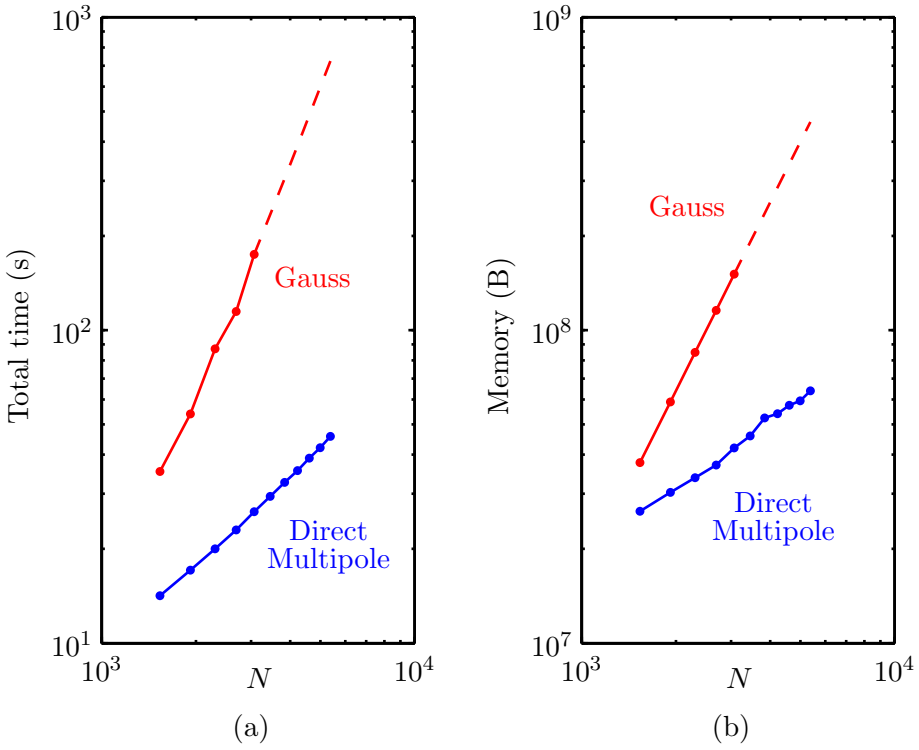


Figure 4.10: (a) Total time, fill plus solve, as a function of problem size N. A least-squares fit to the multipole data has a slope of 0.94. (b) Memory as a function of problem size N. A least-squares fit to the multipole data has a slope of 0.73.

Table 4.6: Solution Time (s) for the Triangle

		Multipole	
N	Gauss	Iterative	Direct
1536	35.36	16.53	14.17
1920	54.00	23.90	17.11
2304	87.12	32.24	20.02
2688	114.78	39.88	23.02
3072	174.55	47.77	26.28
3456	(234)	54.11	29.46
3840	(305)	62.93	32.63
4224	(389)	70.48	35.61
4608	(486)	79.86	38.98
4992	(597)	93.12	42.13
5376	(724)	106.42	45.75

The timings for the direct multipole solver are certainly impressive. In this comparison, however, the iterative solver may be unfairly handicapped. The direct sparse solver used here is not MATLAB's built-in `lu`, but rather UMFPACK, a state-of-the-art library code. The iterative solver is `mygmres`, which is coded as a MATLAB M-file. The multipole algorithm also has an M-file implementation. The overhead of the MATLAB interpreter may be substantial, and a compiled version of the iterative solver may compare more favorably with the direct multipole solver.

4.5 Multipole Preconditioners

I have introduced in this chapter a new direct solver for scattering problems. The direct solver can also be applied to the construction of new preconditioners for iterative solvers. The general strategy is to skip some of the computations in the direct solver, trading accuracy for speed.

Since the direct solver computes the LU factorization of the sparse matrix Ξ, one idea is to compute an incomplete factorization of Ξ.

Furthermore, we might relax any tolerances used to generate Ξ. Multipole expansion lengths can be reduced, giving an approximation $\widehat{\Xi}$ of smaller order and with fewer nonzero elements. Higher thresholds can be employed during the compression, yielding still smaller order and still fewer nonzeros.

I have applied these ideas to the spiral scattering problem treated in Section 4.4.1. In constructing $\widehat{\Xi}$ for the preconditioner factorization, short spectral expansions are used. The expansions have sufficient length only to ensure a truncation error of $\epsilon_{\text{big}} = 10^{-1}$. The basis compression also uses ϵ_{big} as its numerical rank cutoff. The drop tolerance of the incomplete LU factorization is chosen to be 10^{-2}, which is tighter than the basis tolerance. If the drop tolerance is too large, the preconditioner does not perform well.

For the wavenumber $k = 2^{15/2}$, which produces a dense interaction matrix Φ of order $N = 1728$, the relaxed error tolerances generate a sparse matrix $\widehat{\Xi}$ of order 3443 with 284,656 nonzero elements. The tighter tolerances used in the direct solver construct a sparse matrix Ξ of order 4394 with 417,950 nonzero elements.

A row and column permutation of Ξ that works well for the complete factorization is not necessarily the right permutation for an incomplete factorization of $\widehat{\Xi}$. In fact, as illustrated in Figure 4.11, the natural ordering of (4.16) generates many small fill-in values that the incomplete factorization simply discards. That natural ordering turns out to be substantially better than the nested dissection ordering determined by METIS.

The elements colored red in Figure 4.11 are small nonzero elements of $\widehat{\Xi}$ that are dropped from its triangular factors. With a drop tolerance incomplete factorization,

it is possible for the factors to have fewer nonzeros than $\widehat{\Xi}$ itself. That does not happen here, but dropping those small elements helps to offset the fill-in of large values.

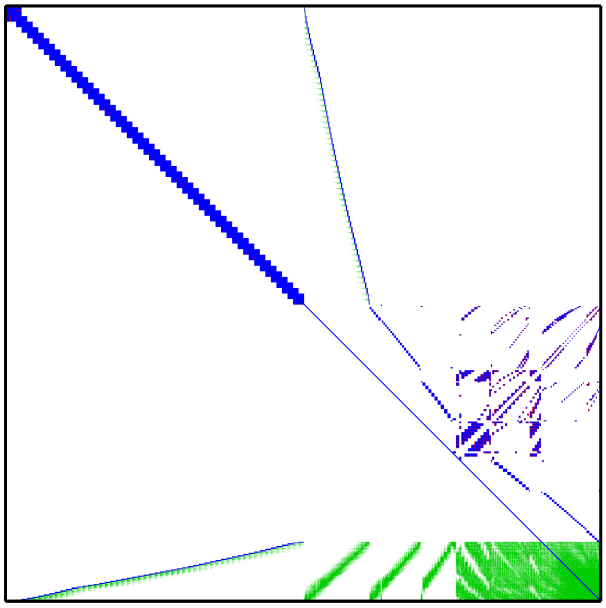
Comparing Figure 4.11(a) with Figure 4.6(a), note that the fringes in the leading principal submatrix D are absent. The spiral here has fewer turns, and the leaf clusters on adjacent turns are well separated. Their interactions are computed with spectral expansions rather than with entries in D.

Unrestarted GMRES is compared once again to the direct multipole solver in Table 4.7. Now, however, in addition to speeding up the matrix-vector multiplications with the usual fast multipole method, the multipole preconditioner described here is also applied. Because of its shorter expansions and reduced fill-in, the iterative solver uses less memory than the direct multipole solver. It still consumes much more memory than the unpreconditioned iteration, as shown in Table 4.3.

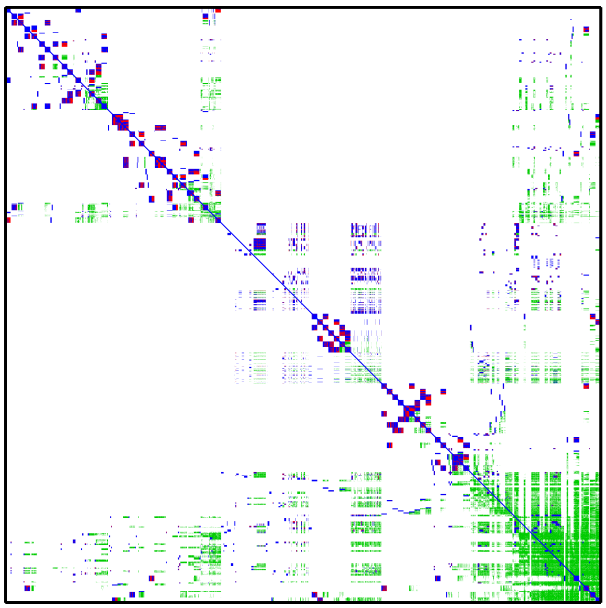
Unlike the preconditioners in Chapter 3, this one is effective against the spiral boundary. With multipole preconditioning, GMRES is faster than the direct multipole solver for large problems. For this boundary, the performance crossover point for these methods occurs at about $N = 1344$. Referring back to Table 4.2, the multipole preconditioner lends an appreciable improvement to GMRES at *all* values of N.

Table 4.8 gives more evidence of the effectiveness of the new preconditioner. Since approximations of the long-range interactions are captured in the multipole preconditioner, it greatly reduces the number of GMRES iterations required to converge to an error level of $\epsilon = 10^{-4}$. An incomplete LU preconditioner that includes only short-range interactions offers scant improvement of the unpreconditioned solver.

Starting with the direct multipole solver, other preconditioners might be devel-



(a)



(b)

Figure 4.11: (a) With the natural ordering (4.16) of $\widehat{\Xi}$, incomplete Gaussian elimination nets 270,407 new nonzero elements. (b) With the nested dissection ordering computed by METIS, the net fill is 371,900 elements.

Table 4.7: Performance of Multipole Preconditioner

	Memory (MB)		Time (s)	
N	MP Direct	MP Precon	MP Direct	MP Precon
576	7.10	7.97	6.10	9.69
736	10.02	11.80	7.90	14.75
896	14.85	13.32	11.36	18.53
1344	28.35	14.91	22.06	22.01
1728	42.15	17.57	37.28	25.38
2624	69.63	31.26	61.31	48.01
3392	116.13	49.47	120.78	90.22
5152	206.56	90.37	$240.99 + 3.11$	195.67
6688	343.95	146.18	$492.87 + 18.00$	$353.11 + 1.22$

Table 4.8: GMRES Steps Taken with Multipole Preconditioning

	GMRES Steps		
N	No Precon	ILU Precon	MP Precon
576	110	49	21
736	139	68	25
896	191	93	28
1344	237	163	28
1728	306	249	27
2624	375	314	37
3392	601	514	55
5152	837	723	79
6688	794	720	92

oped. One idea is to apply a graph coarsening heuristic, like those used in the algebraic multigrid method [16] [137], to eliminate some edges and vertices of the multipole dag. That idea extends the dag compression of Section 4.3, and may lead to a further reduction in the size of $\widehat{\Xi}$.

Chapter 5

Taming Multipole's Numerical Instability

I have so far, apart from a sideways glance on page 83, skirted an issue that commands attention. As anyone who implements a multipole method from Chapter 2 will quickly discover, the algorithm is *numerically unstable*. The computation can, and probably will, go badly awry because of differences between exact and floating-point arithmetic.

The numerical instability is a phenomenon that plagues scattering problems in particular. It does not play a role[1] in the application of multipole methods to the Laplace equation. Fast multipole methods have been constructed for many other interaction kernels, but I am unaware of any unstable methods among them.

In this chapter, I describe how the scattering instability originates and how it can be repaired. Fortunately, the remedy does not require a complete teardown of the algorithm. The history of computational science and engineering is replete

[1]Instability has hampered efforts to speed up the translations [13] [52], but the need for such fast translations is less pressing than in scattering problems.

with examples of fast but unstable algorithms that have not been so lucky, and it is generally a losing strategy to try to build numerical stability into an algorithm as an afterthought.

Much published work on scattering multipole has failed to adequately address the instability. Many budding implementations have probably been thwarted, and as a result the method is not as widespread as it should be.

While acknowledgment of the instability has grown, it continues to be characterized as a phenomenon that should occur only in special circumstances. In particular, the problem is described as a "subwavelength instability" that insinuates itself into scattering solutions only for obstacles that have nonsmooth structure at scales smaller than a wavelength.

5.1 Demonstration of the Instability

Figure 5.1 demonstrates that the instability is not confined to low frequencies. Here a circular cylinder with unit radius is illuminated by a TM plane wave with wavenumber $k = 30$. The circumference of the cylinder is 30 wavelengths, so it is not optically small. The surface current density, as computed by unpreconditioned GMRES with a multipole matrix-vector product, is graphed for each member of a decreasing sequence of multipole truncation error tolerances. As the tolerance is reduced from $\epsilon = 10^{-1}$ to $\epsilon = 10^{-3}$, the solution takes shape nicely. The current is small on the half of the cylinder in shadow, and on the bright side the current oscillates rapidly, reflecting the high frequency of the illuminating field. The solution holds its shape as the tolerance is further tightened, until at $\epsilon = 10^{-9}$ some stray spikes begin to appear. At $\epsilon = 10^{-10}$ it has been completely corrupted. A closer inspection of the

computed values reveals that these errors appear also in the solution at $\epsilon = 10^{-8}$, where they limit the accuracy to about 6 significant digits instead of the requested 8.

The error growth is catastrophic. This is not the type of behavior that results from the gradual accumulation of small rounding errors. Nor is the error specification too close to machine epsilon, `eps` $\approx 10^{-16}$, for us to expect a reliable solution. Clearly a major flaw in the algorithm has been exposed.

The instability portrayed in Figure 5.1 worsens for low frequency problems, so the description "subwavelength instability" is not without merit. If k were decreased in these computations, the instability would obliterate the computed solution at larger values of ϵ. The instability threshold is not a function of k alone, but of both k and ϵ.

I hasten to add that the code responsible for the displayed results has been designed, through tight control of the spectral expansion lengths, to allow as little of the instability as possible to creep in. A naive implementation is likely to do far worse.

We will conduct further analysis on a purely discrete problem, to strip away such things as singular quadrature rules and their associated discretization errors. Consider the charged particle distribution in Figure 5.2. A random collection of N point oscillators contained in the unit disk generates a scalar wavefield that is evaluated at M random positions in a disk well separated from the sources.

The field is $\boldsymbol{u} = \Phi\boldsymbol{q}$. Exposing the matrix and vector elements,

$$u_m = \sum_{n=1}^{N} \Phi(\boldsymbol{y}_m - \boldsymbol{x}_n) q_n, \quad 1 \le m \le M. \tag{5.1}$$

We will compare two ways of evaluating this sum: the standard matrix-vector prod-

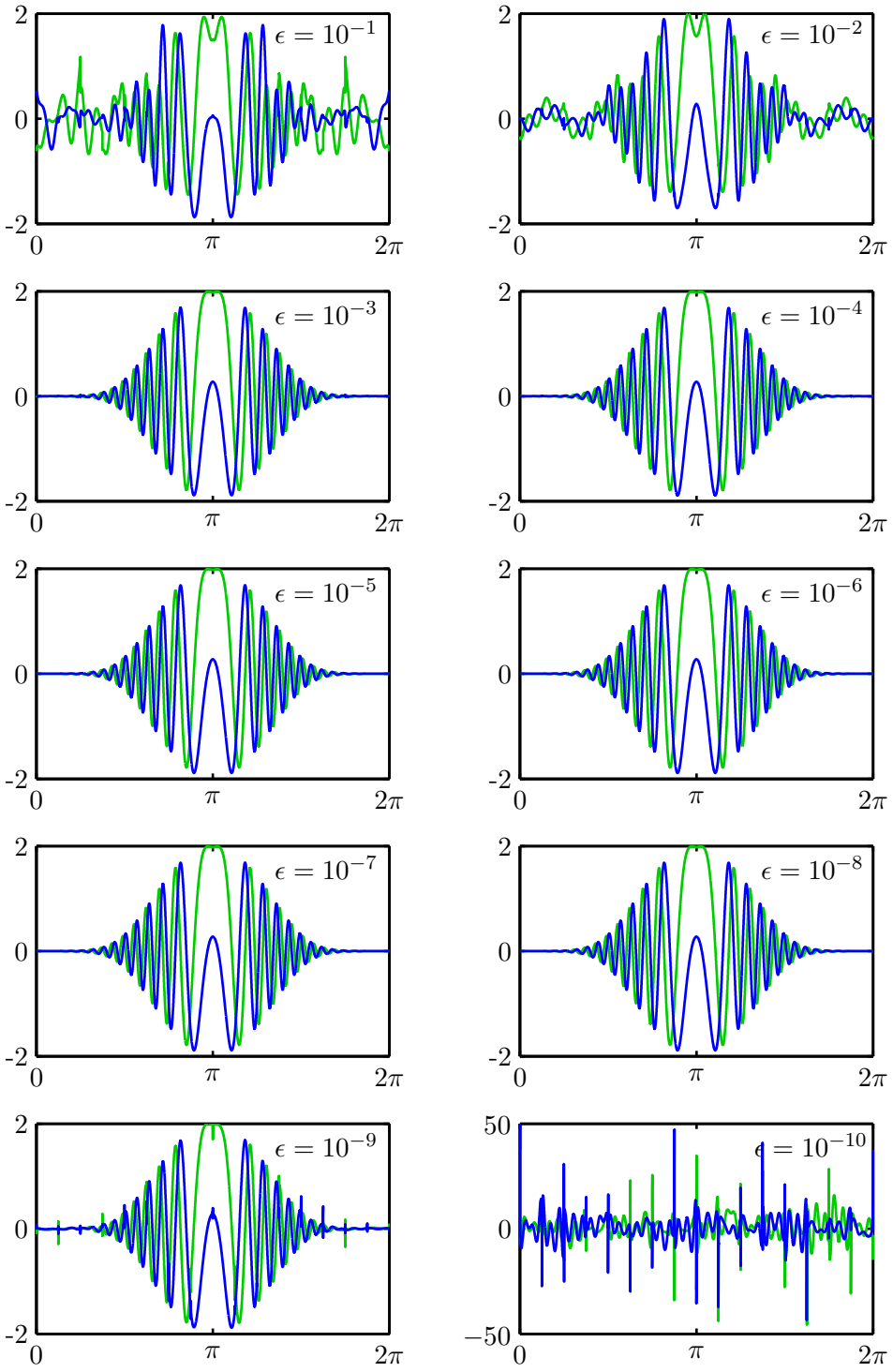


Figure 5.1: Real (—) and imaginary (—) parts of the normalized current density $\eta\sigma/\|\boldsymbol{E}_{\text{inc}}\|$ as a function of angular position ϕ on a circular cylinder. The illumination is a plane wave incident from $\phi = \pi$. The change of vertical scale in the last frame is a telltale sign of numerical instability.

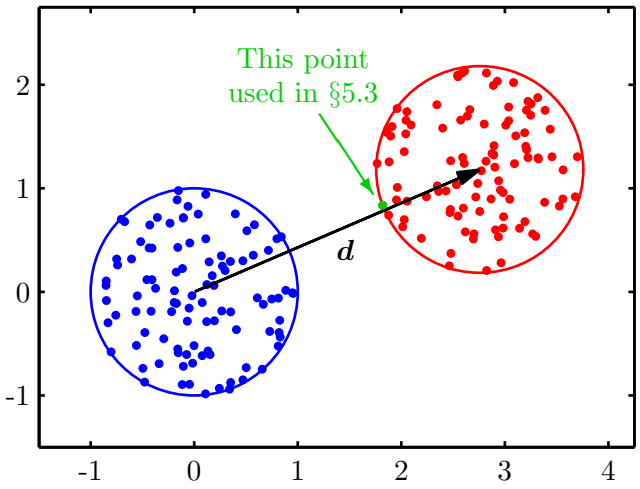


Figure 5.2: $N = 100$ particles at points $\{\boldsymbol{x}_n\}$ generate an oscillating field in $\mathbb{E}^2$. The field is evaluated at $M = 100$ distant points $\{\boldsymbol{y}_m\}$. The vector $\boldsymbol{d}$ is the displacement from the center of the source disk to the center of the destination disk.

uct and the fast multipole method. The standard product is a stable reference against which the numerical stability of the multipole algorithm will be judged.

Since the cluster of destination points is well separated from the cluster of source points, the multipole algorithm is a simple factorization of Φ. To briefly review, a leaf operator V transforms the charge amplitudes $\boldsymbol{q}$ into a vector $\boldsymbol{a}$ of exterior spectral expansion coefficients. Then the exterior coefficients are mapped to interior expansion coefficients $\boldsymbol{b}$ through a Toeplitz matrix T. Finally, another leaf operator U transforms the interior coefficients into the field values $\boldsymbol{u}$. Putting this together, we have

$$\left.\begin{aligned} \boldsymbol{a} &= V\boldsymbol{q} \\ \boldsymbol{b} &= T\boldsymbol{a} \\ \hat{\boldsymbol{u}} &= U\boldsymbol{b} \end{aligned}\right\} \quad \hat{\boldsymbol{u}} = UTV\boldsymbol{q}, \tag{5.2}$$

where the computed field vector has been decorated with a hat to distinguish it from the vector $\boldsymbol{u}$ computed by the standard product.

5.2 High Frequency Instability

In the low frequency case, Φ is numerically rank deficient and the factor T is small, instantly conferring an efficiency advantage to the multipole factorization.

At high frequencies, T is not small, and the factorization efficiency originates elsewhere. Since T has Toeplitz structure, a fast matrix-vector product $T\boldsymbol{a}$ is possible. The algorithm was covered in Section 2.3.2. A fast translation $T\boldsymbol{a}$ is not sufficient by itself to reduce the complexity of $\Phi\boldsymbol{q}$, since we still must compute the matrix-vector products involving U and V. In the multilevel algorithm, however, the matrices U and V are themselves factored into products of leaf operators and branch translation operators. If the source and destination point sets are each organized into a binary cluster tree with 3 levels, then

$$U = \begin{pmatrix} \begin{bmatrix} U_1 R_1 \\ U_2 R_2 \end{bmatrix} R_5 \\ \begin{bmatrix} U_3 R_3 \\ U_4 R_4 \end{bmatrix} R_6 \end{pmatrix}, \tag{5.3a}$$

$$V = \Big(S_5 \begin{bmatrix} S_1 V_1 & S_2 V_2 \end{bmatrix} \quad S_6 \begin{bmatrix} S_3 V_3 & S_4 V_4 \end{bmatrix} \Big). \tag{5.3b}$$

The matrices $\{U_i\}$ and $\{V_i\}$ are small, and all matrices $\{R_i\}$ and $\{S_i\}$ are Toeplitz. With this additional structure, the multipole factorization (5.2) can be faster than (5.1).

Although it is not fast, we will use the single-level factorization $\Phi = UTV$ as a point of comparison because it exhibits the same instability as the faster multilevel factorization. The translation matrix T is the source of the problem at high frequencies.

The solutions $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ have been computed for the particle distribution in Figure 5.2 at the wavenumber $k = 10$. Figure 5.3 shows that no instability is present if

a standard $O(p^2)$ matrix-vector product is used to compute $T\boldsymbol{a}$. If the fast $O(p \log p)$ product is used, and if the spectral expansion is too long, then the algorithm shows its instability. Since the fast product cannot be abandoned, we should adhere to the following piece of advice:

Choose the spectral cutoff p to be as large as necessary, but no larger.

This is good advice anyway from an efficiency standpoint, since needlessly long expansions only add to the workload. But the penalty of numerical instability is much greater.

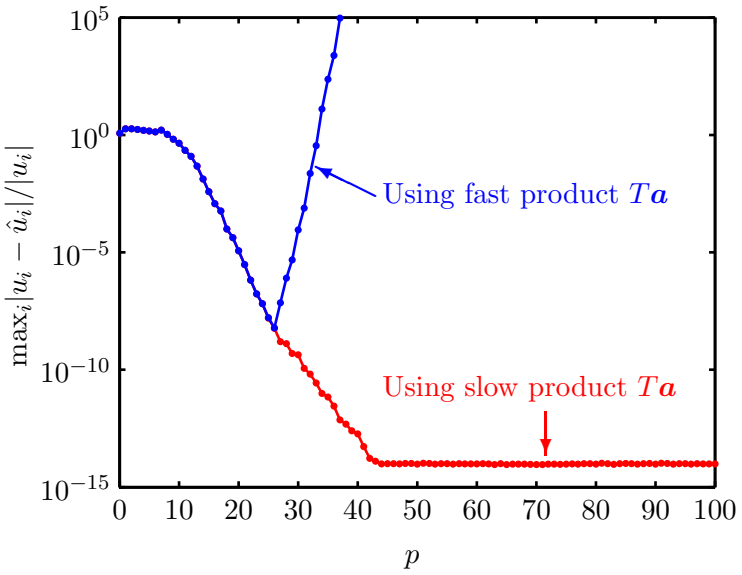


Figure 5.3: Maximum componentwise relative error in the computed field as a function of the spectral cutoff p. This is perhaps an unreasonably tough test. The field $\boldsymbol{u}$ is oscillatory, and some elements may be nearly zero. Only a small *absolute* error can be expected for those elements.

As explained in Chapter 2, the spectral expansion length controls the accuracy of the multipole method. In exact arithmetic, longer expansions always yield higher accuracy. Figure 5.3 shows that this is not true in floating-point arithmetic. Par-

ticularly when the computation time is short, it is tempting to increase p in order to obtain a more accurate solution. That is a risky proposition, and that is how I discovered the instability in my first multipole implementation.

Figure 5.4 shows how the instability threshold changes as the wavenumber is increased. The first frame of that sequence of plots is the same as Figure 5.3. Clearly, as k increases, it is safer to use longer spectral expansions. These graphs also show that as k increases longer expansions are necessary to obtain a given accuracy. The requirement given in Chapter 2 is $p = O(k\alpha + \log \epsilon^{-1})$, where α is the disk radius. As the optical size of the disks increases, more expansion terms are needed before the solution begins to converge. That behavior is exhibited in the rightward progression of the knee at the 0-digit accuracy level. For larger values of p, the error decays at an exponential rate, until rounding errors prohibit it from dropping below `eps`. As expected for a semilogarithmic graph, the plotted data nearly falls on a straight line in the region of exponential convergence.

This experiment shows that the fast translation between well-separated clusters can only be used if the requested accuracy is small enough. If the requested accuracy is too high, then the value of p necessary to achieve that accuracy in exact arithmetic will exceed the instability threshold of the floating-point algorithm. In such an event, I switch to the slow but stable algorithm for computing $T\boldsymbol{a}$. Given two clusters and values of k and ϵ, my code automatically determines which multiplication algorithm should be used.

The question raised by selectively switching to the slower, stable multiplication is: Does that ruin the complexity of the multipole algorithm as a whole? Figure 5.4 gathers evidence that it does not, because as k increases, the accuracy achievable with the fast multiplication also increases. That trend is displayed more explicitly in Table 5.1, which tracks the error level at the onset of instability.

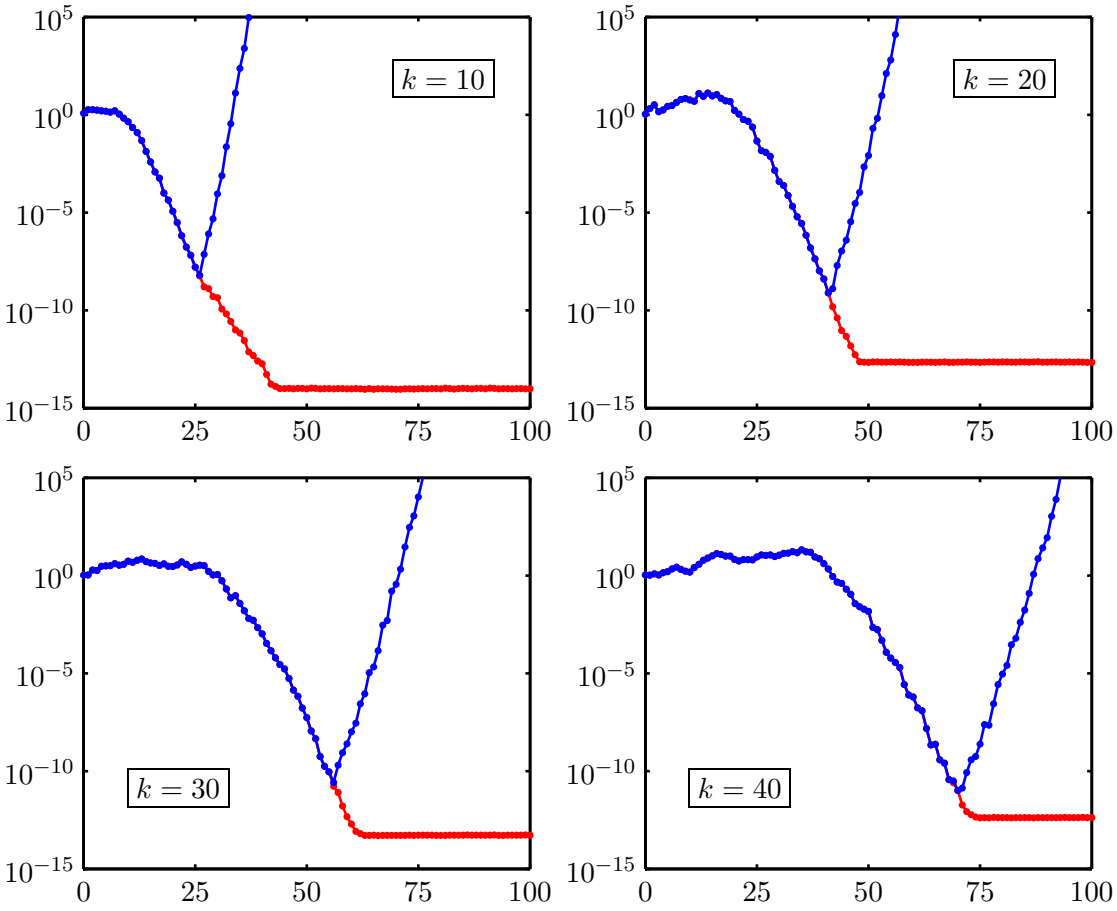


Figure 5.4: Plots of $\max_i |u_i - \hat{u}_i|/|u_i|$ versus p as wavenumber k increases. The point at which the curves separate—beyond which a fast translation should not be used—moves in the direction of increasing p and decreasing ϵ. Were the latter trend in the direction of *increasing* ϵ, the multipole algorithm would probably be unsalvageable.

Table 5.1: Accuracy Obtainable by Fast Translation

k	$\max_i \lvert u_i - \hat{u}_i \rvert / \lvert u_i \rvert$
10	6.1×10^{-09}
20	7.6×10^{-10}
30	2.6×10^{-11}
40	1.0×10^{-11}
50	5.6×10^{-13}
60	1.6×10^{-13}
70	2.7×10^{-13}
80	3.3×10^{-13}
90	3.0×10^{-13}
100	3.6×10^{-13}

As the optical size of the clusters increases while ϵ is fixed, the fast translation becomes more stable. Referring back to the hierarchical multipole example of N interacting particles given in Section 2.5, the largest clusters lie at the upper reaches of the multipole tree. For these clusters, p is large—it may approach or even exceed N—and the $O(p^2)$ translation is worthless. The fast translation *must* be available for these translations. For small clusters, at the lower extremities of the multipole tree, it is quite acceptable to use the $O(p^2)$ translation because p is small.

5.2.1 Why is Fast Translation Unstable?

Through experimentation it is easy enough to isolate the fast translation as the cause of instability. But that algorithm (Algorithm 2.4) consists of nothing more than a few FFTs, and the FFT has a rock-solid stability reputation. An FFT is more stable, in fact, than slow multiplication by the DFT matrix [82, Ch. 24]. So what, exactly, is the problem?

To investigate further, we reexamine the solution computed for the two-cluster

problem pictured in Figure 5.2. Let $k = 10$ and $p = 34$. Referring to Figure 5.3, the fast multiplication $T\boldsymbol{a}$ is unstable for this combination of parameters, and almost all elements of the computed solution vector $\hat{\boldsymbol{u}}$ have no accurate digits whatsoever. Figure 5.5 is a graphical display of the elements of $T \in \mathbb{C}^{69\times 69}$ and $\boldsymbol{a} \in \mathbb{C}^{69}$. The dynamic range of the displayed values is huge, more than 30 orders of magnitude. The scale ranges from about `eps` to $\texttt{eps}^{-1}$, and that is a clue to why we have no significant digits left in the computed solution.

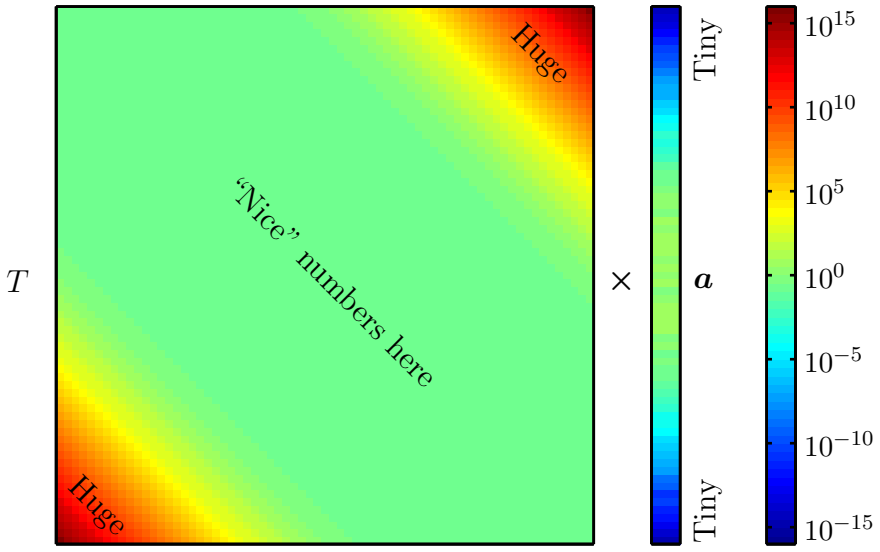


Figure 5.5: Modulus of elements in T and $\boldsymbol{a}$. Note that, in forming the product $T\boldsymbol{a}$, the huge numbers in T *always* get multiplied by tiny numbers in $\boldsymbol{a}$.

The huge elements of T give the matrix a commensurately large norm. Comparing the norm of the interaction matrix with the norms of its factors,

$$\|\Phi\|_2 \approx 11.6 \qquad \text{versus} \qquad \begin{aligned} \|U\|_2 &\approx 3.17, \\ \|T\|_2 &\approx 4.11 \times 10^{15}, \\ \|V\|_2 &\approx 3.23, \end{aligned} \tag{5.4}$$

it is easy to fault the factorization. Since $\|UTV\| \ll \|U\|\|T\|\|V\|$, the factorization

is surely an unstable foundation for any algorithm. But, remarkably, the multipole algorithm manages to be stable when the standard matrix-vector product is used to compute $T\boldsymbol{a}$. To explain, we must turn to a componentwise analysis.

Consider the sensitivity of the product $T\boldsymbol{a}$ to relative perturbations of the elements in T or $\boldsymbol{a}$. The componentwise condition number for $\boldsymbol{b} = T\boldsymbol{a}$ is

$$\begin{aligned} \kappa_a &:= \frac{\| \, |T||\boldsymbol{a}| \, \|_2}{\|T\boldsymbol{a}\|_2} \\ &\approx 1.81, \end{aligned} \tag{5.5}$$

which is surprisingly small. It is so small because, in forming the product[2] $|T||\boldsymbol{a}|$, the huge elements of T always multiply tiny elements in $\boldsymbol{a}$ to give "nice" numbers, something that becomes evident upon studying the positions of small and large numbers in Figure 5.5.

Although the translation is well conditioned in a componentwise sense, the normwise condition number for the same problem is

$$\begin{aligned} \kappa_b &:= \frac{\|T\|_2\|\boldsymbol{a}\|_2}{\|T\boldsymbol{a}\|_2} \\ &\approx 2.75 \times 10^{12}, \end{aligned} \tag{5.6}$$

which is certainly awful. The product $\|T\|_2\|\boldsymbol{a}\|_2$ offers no opportunity to attenuate the huge elements of T.

The standard algorithm for a matrix-vector product is backward stable in a componentwise sense. The cumulative effect of roundoff errors is equivalent to an $O(\texttt{eps})$ relative perturbation of the data elements in the complete absence of rounding errors. If $\hat{\boldsymbol{b}}$ is the computed product, then

$$\hat{\boldsymbol{b}} = (T + \delta T)\boldsymbol{a} \quad \text{where} \quad |\delta T| = O(\texttt{eps}\,|T|). \tag{5.7}$$

[2]The ith element of $|\boldsymbol{a}|$ is $|a_i|$. The (i,j)th element of $|T|$ is $|T_{ij}|$.

Magnifying the tiny perturbation in T by the condition number (5.5) produces only a tiny error in the computed product $\hat{\boldsymbol{b}}$.

If, however, our algorithm for the matrix-vector product were not componentwise backward stable, but only normwise backward stable, then the much larger condition number (5.6) would magnify an $O(\texttt{eps})$ data perturbation, giving a large error in the computed product.

The condition numbers κ_a and κ_b both measure the normwise relative perturbation $\|\boldsymbol{b} - \hat{\boldsymbol{b}}\|_2/\|\boldsymbol{b}\|_2$ of the output. It is not difficult to show, either theoretically or through experiment, that a small normwise perturbation in $\boldsymbol{b}$ is not sufficient to ensure a small perturbation in $\|\boldsymbol{u}\|$. Rather, each *element* of $\boldsymbol{b}$ should have a small relative perturbation. Like T and $\boldsymbol{a}$, the elements of $\boldsymbol{b}$ span a large dynamic range, and its smallest elements can have a large relative error without influencing the relative normwise error.

Each element of $\boldsymbol{b}$ has a condition number that measures its sensitivity to relative perturbations in the elements of T or $\boldsymbol{a}$. For the data of Figure 5.5, the largest of those condition numbers is

$$\begin{aligned} \kappa_c &:= \max_i \frac{i\text{th component of } |T||\boldsymbol{a}|}{i\text{th component of } |T\boldsymbol{a}|} \\ &\approx 7.91, \end{aligned} \tag{5.8}$$

which, like (5.5), is small.

A rounding analysis gives the following forward error estimates for the elements of $\boldsymbol{b}$:

$$\text{Slow translation:} \qquad |\boldsymbol{b} - \hat{\boldsymbol{b}}| = O(\texttt{eps}\,|T||\boldsymbol{a}|) \tag{5.9a}$$

$$\text{Fast translation:} \qquad |\boldsymbol{b} - \hat{\boldsymbol{b}}| = O(\texttt{eps}\,\|\boldsymbol{t}\|\|\boldsymbol{a}\|) \tag{5.9b}$$

In the latter estimate, $\boldsymbol{t}$ is the main crossdiagonal of T, and since $\|\boldsymbol{t}\|_2 \approx 5.65 \times 10^{15}$ the fast algorithm is not stable with respect to the condition number (5.8). In my code, the switch from a fast translation to a slow translation is made when $\|\boldsymbol{t}\|_2$ exceeds a threshold that depends on the chosen accuracy ϵ.

The simple example in Figure 5.6 computes $\hat{\boldsymbol{x}} = F^{-1}F\boldsymbol{x}$ for a vector $\boldsymbol{x}$ with large dynamic range. The FFT mixes the small and large elements of $\boldsymbol{x}$, and significant digits in the small elements are lost during floating-point addition with larger elements. The error in each component of $\hat{\boldsymbol{x}}$ is $O(\texttt{eps}\,\|\boldsymbol{x}\|)$, showing that processing with the FFT can introduce large relative errors in small components. That does not contradict the fact that the FFT is *normwise* backward stable.

```
>> x = (1/2).^(0:40);
>> y = real(ifft(fft(x)));
>> [x(end); y(end)]

ans =

    1.0e-12 *

    0.90949470177293
    0.90958135332607
```

Figure 5.6: A short MATLAB session. The second line of input is an algorithm for the identity operation, a problem that is perfectly conditioned by any measure. The first line defines the input vector $\boldsymbol{x} \in \mathbb{R}^{41}$, which has elements $\boldsymbol{x}_i = 2^{1-i}$. The output produced in response to the third line shows that the computed solution $\boldsymbol{y}$ has a large relative error in its smallest component. Only 3 digits are correct.

Another demonstration, closer in spirit to the fast multipole translation, is the well-known application of the FFT to the rapid computation of discrete convolutions.

Let two sequences be

$$f_n = \begin{cases} \alpha^{|n|} & \text{if } -q \le n \le q, \\ 0 & \text{otherwise,} \end{cases} \tag{5.10a}$$

$$g_n = \begin{cases} \beta^{|n|} & \text{if } -r \le n \le r, \\ 0 & \text{otherwise.} \end{cases} \tag{5.10b}$$

The convolution of f and g is defined elementally as

$$(f \star g)_n := \sum_{m=-\infty}^{\infty} f_m g_{n-m}, \quad -\infty \le n \le \infty. \tag{5.11}$$

The sequences have bounded support, and an equivalent matrix-vector product is $G\boldsymbol{f}$, where $G \in \mathbb{R}^{(2q+2r+1)\times(2q+1)}$ is a banded Toeplitz matrix, as in Figure 5.7. The leading $2r+1$ elements of the first column of G comprise the vector $\boldsymbol{g}$.

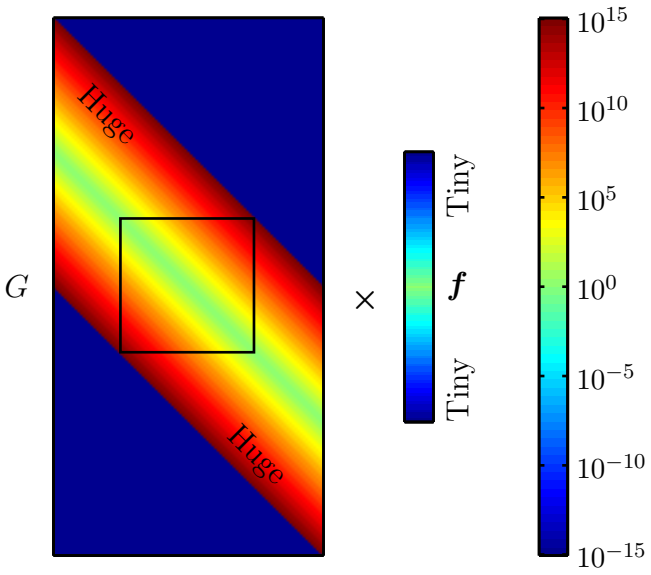


Figure 5.7: Graphical representation of the convolution of sequence (5.10a) with $\alpha = 1/2$, $q = 50$ and the sequence (5.10b) with $\beta = 2$, $r = 50$. Compare the outlined square submatrix of G with the matrix T in Figure 5.5.

In the example of Figure 5.7, replacing the matrix-vector product with a fast convolution produces an output vector whose smallest components are in error by more than 25 percent. There are no correct digits in those elements. The largest elements, on the other hand, have a full complement of correct digits.

The breakdown of the multipole algorithm is evidently caused by the combination of two factors:

1. The fast translation algorithm is normwise backward stable but not componentwise backward stable.
2. The translation is well conditioned in a componentwise sense but poorly conditioned in a normwise sense.

In light of the latter factor, it becomes less clear that the fast translation algorithm is to blame. We have described it as numerically unstable, but that is not a label fairly attached to an algorithm that is normwise backward stable. It seems more judicious to blame the multipole factorization for presenting the fast translation algorithm with a poorly conditioned system in the first place. After all, the overall problem, to compute $\boldsymbol{u} = \Phi\boldsymbol{q}$, is well conditioned. The factorization has fashioned a poorly conditioned subproblem, so the fault must really lie with the multipole algorithm structure at an earlier stage. The following section uncovers the root of the trouble.

5.3 Low Frequency Instability

For small enough wavenumbers, the instability of the multipole algorithm manifests itself in a different way. As shown in Figure 5.5, the elements grow exponentially in modulus as the top-right and bottom-left corners of T are approached. That growth is exacerbated at small wavenumbers, and the largest elements of T can overflow.

Furthermore, if the spectral expansions are too long at low frequencies, the overflow in T may be preempted by underflow in U or V.

This instability is unrelated to the fast translation algorithm, and the strategy in the previous section of switching to a slow translation does not ensure stability at low frequencies.

Figure 5.8 exhibits the accuracy obtainable for the particle problem of Figure 5.2 at low frequencies. For each of several values of k, the multipole expansion length is increased until at least one element of T overflows. Table 5.2 summarizes the results.

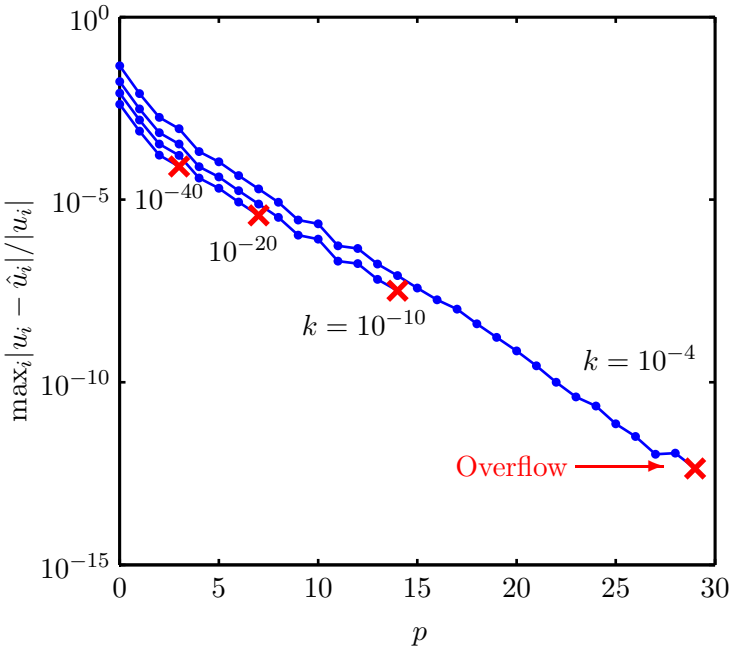


Figure 5.8: Floating-point range exceptions halt the computation at low frequencies. The minimum attainable error is determined by the optical size $k\alpha$ of the disks.

The problem underlying both high and low frequency instabilities is that the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and the matrices U, T, and V all have elements that span too great a range. The fault ultimately rests with the chosen bases for the exterior and interior

Table 5.2: Accuracy Obtainable with Unstable Basis

k	$\max_i \lvert u_i - \hat{u}_i \rvert / \lvert u_i \rvert$
10^{-40}	8.1×10^{-05}
10^{-20}	3.7×10^{-06}
10^{-10}	3.2×10^{-08}
10^{-4}	4.4×10^{-13}

spectral representations.

The multipole algorithm is numerically unstable

because

both interior and exterior bases are unstable.

Recall from Section 2.2.1 that the basis for the spectral expansion on the exterior of a disk with center $\mathbf{0}$ is

$$\varphi_n(\boldsymbol{x}) := H_n(k\|\boldsymbol{x}\|)e^{in\theta(\boldsymbol{x})}, \quad n \in \mathbb{Z}, \tag{5.12}$$

and, from Section 2.2.2, the basis on the disk interior is

$$\psi_n(\boldsymbol{x}) := J_n(k\|\boldsymbol{x}\|)e^{in\theta(\boldsymbol{x})}, \quad n \in \mathbb{Z}. \tag{5.13}$$

Through the Bessel function addition theorem, these functions also appear in the matrix elements of U, T, and V. Those elements are

$$U_{mn} = \psi_n(\boldsymbol{y}_m - \boldsymbol{d}), \tag{5.14a}$$

$$T_{mn} = \varphi_{n-m}(\boldsymbol{d}), \tag{5.14b}$$

$$V_{mn} = -\tfrac{i}{4}\overline{\psi_m(\boldsymbol{x}_n)}, \tag{5.14c}$$

if the center of the source disk in Figure 5.2 lies at the origin of coordinates, and if $\boldsymbol{d}$ is the displacement vector from the origin to the center of the destination disk.

Figure 5.9 graphs the Bessel functions $J_\nu(x)$ and $Y_\nu(x)$ for $\nu, x \in [0, 30]$. The scale problems are reflected in the asymptotic behavior of the basis functions. As $|n| \to \infty$, $|J_n|$ decays faster than any exponential function $\zeta^{|n|}$ with $\zeta < 1$. Meanwhile, $|H_n|$ grows faster than any exponential function $\zeta^{|n|}$ with $\zeta > 1$. In more detail, the asymptotic behavior of the basis functions is [1, §9.3.1]

$$\psi_n(\boldsymbol{x}) = O\left(\frac{1}{\sqrt{n}}\left(\frac{ek\|\boldsymbol{x}\|}{2n}\right)^n\right) \quad \text{as } n \to \infty, \tag{5.15a}$$

$$\varphi_n(\boldsymbol{x}) = O\left(\frac{1}{\sqrt{n}}\left(\frac{ek\|\boldsymbol{x}\|}{2n}\right)^{-n}\right) \quad \text{as } n \to \infty. \tag{5.15b}$$

The behavior as $n \to -\infty$ is the same, since $|\psi_{-n}| = |\psi_n|$ and $|\varphi_{-n}| = |\varphi_n|$.

To understand the trouble this causes, pick a single point $\boldsymbol{x}$, highlighted in Figure 5.2, and evaluate each term in the exterior spectral expansion of the field at that point. The expansion is

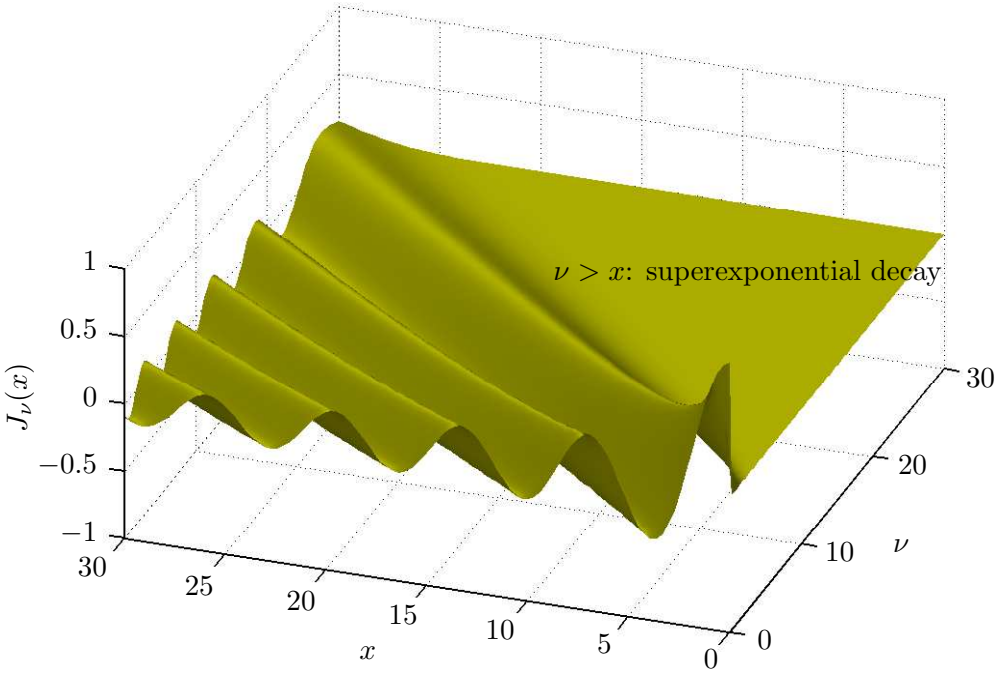
$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} a_n \varphi_n(\boldsymbol{x}), \tag{5.16}$$

and Figure 5.10 plots the modulus the factors a_n and φ_n, together with the modulus of their product.

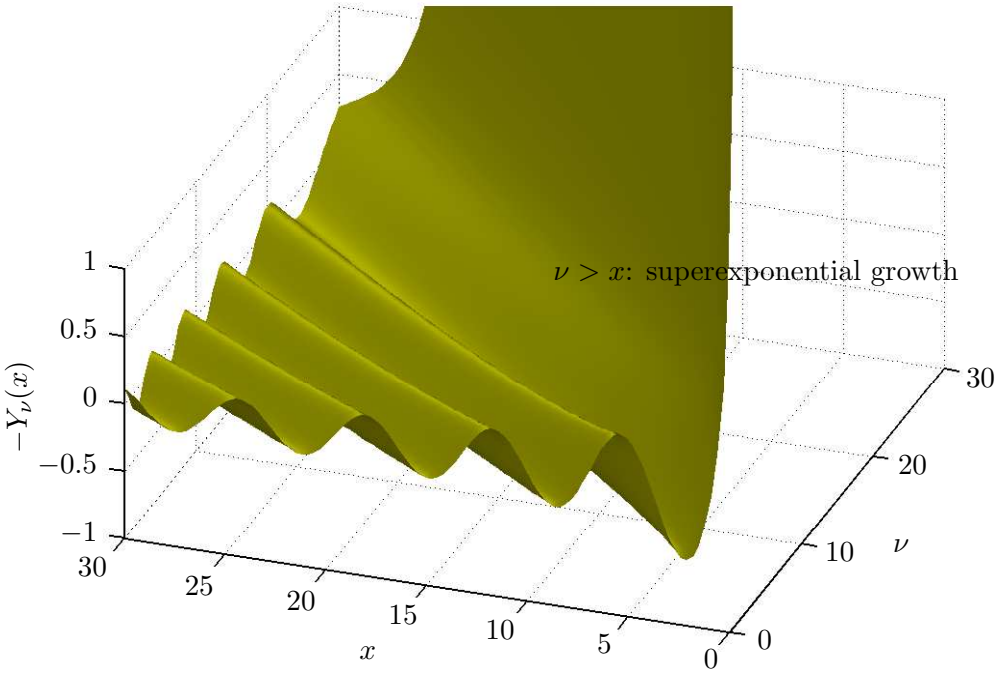
The vertical scale in Figure 5.10 covers practically the entire range of double-precision floating-point numbers. While the terms of the expansion fall on a shallow line on this logarithmic scale, for large n the terms are computed as the product of a huge number and a tiny number. To compensate superexponential growth in the exterior basis functions, the expansion coefficients $\{a_n\}$ decay at a superexponential rate. At $n = 26$, the growing tension between φ_n and a_n finally snaps when a_n underflows.

The behavior of the interior spectral expansion,

$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} b_n \psi_n(\boldsymbol{x} - \boldsymbol{d}), \tag{5.17}$$



(a)



(b)

Figure 5.9: (a) Bessel function $J_\nu(x)$. (b) Bessel function $Y_\nu(x)$.

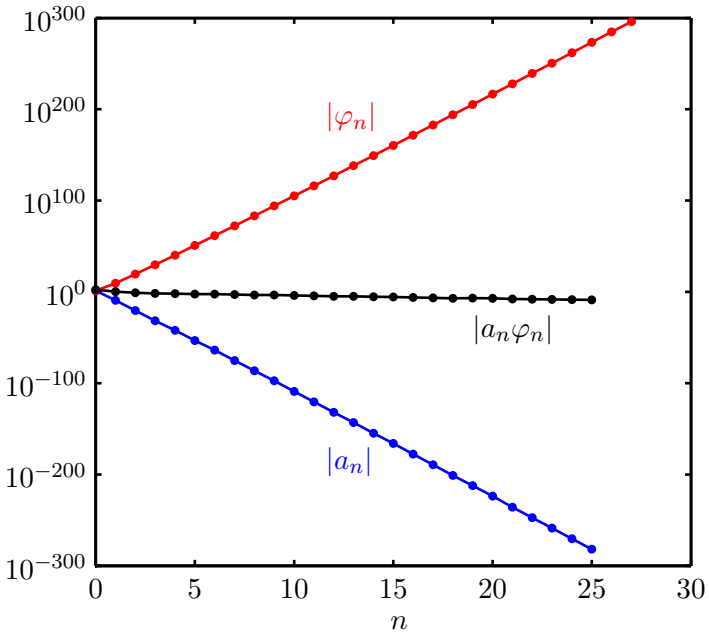


Figure 5.10: Composition of the terms in the exterior spectral expansion. Growth in φ_n is compensated by decay in a_n, at least until the range limits `realmin` and `realmax` of the floating-point number system are breached.

is the same. To compensate superexponential decay in the interior basis functions, the expansion coefficients $\{b_n\}$ grow at a superexponential rate.

5.3.1 Scaling the Multipole Basis

The easiest remedy for the arithmetic exceptions at low frequencies is to scale the basis functions, better governing their asymptotic rate of change. I use a normalization that restrains the limiting behavior to an exponential law.

For a disk with radius α, the new basis functions are

$$\widetilde{\psi}_n(\boldsymbol{x}) := \psi_n(\boldsymbol{x}) H_n(k\alpha), \tag{5.18a}$$

$$\widetilde{\varphi}_n(\boldsymbol{x}) := \frac{\varphi_n(\boldsymbol{x})}{H_n(k\alpha)}, \tag{5.18b}$$

and these functions have the asymptotic behavior

$$\widetilde{\psi}_n(\boldsymbol{x}) = O\left(\left(\frac{\|\boldsymbol{x}\|}{\alpha}\right)^{|n|}\right) \quad \text{as } |n| \to \infty, \tag{5.19a}$$

$$\widetilde{\varphi}_n(\boldsymbol{x}) = O\left(\left(\frac{\alpha}{\|\boldsymbol{x}\|}\right)^{|n|}\right) \quad \text{as } |n| \to \infty, \tag{5.19b}$$

so both interior and exterior basis functions exhibit exponential decay. This is illustrated for the unit disk in Figure 5.11.

The new basis eliminates range exceptions even for wavenumbers that approach the underflow limit `realmin`. This stability is illustrated by taking $k = 10^{-200}$ with the random charge distribution of Figure 5.2. Table 5.3 shows that with the new basis a relative error approaching `eps` is possible. Without scaling, only single-term expansions are allowed, and the achievable error is no better than 8×10^{-4}.

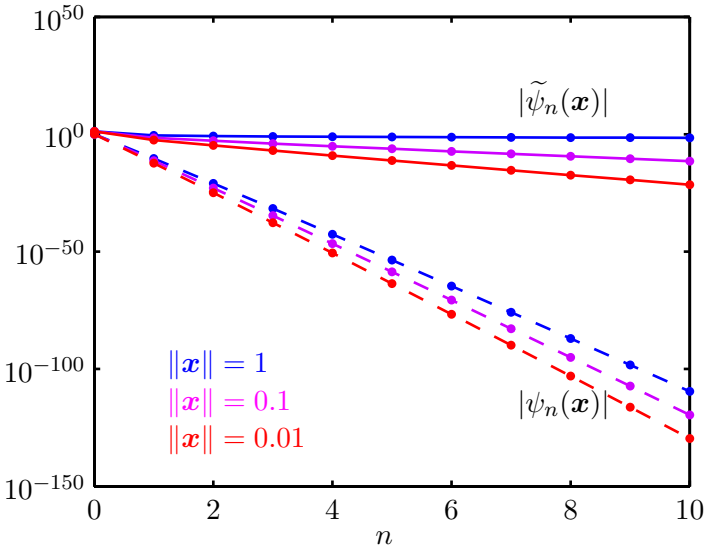
Table 5.3: Scaled Basis Performance at $k = 10^{-200}$

	$\max_i \ \lvert u_i - \hat{u}_i\rvert/\lvert u_i\rvert$	
p	Before Scaling	After Scaling
0	8.1×10^{-04}	8.1×10^{-04}
1	Fails	1.5×10^{-04}
2	Fails	3.3×10^{-05}
3	Fails	1.6×10^{-05}
⋮	⋮	⋮
32	Fails	1.2×10^{-15}

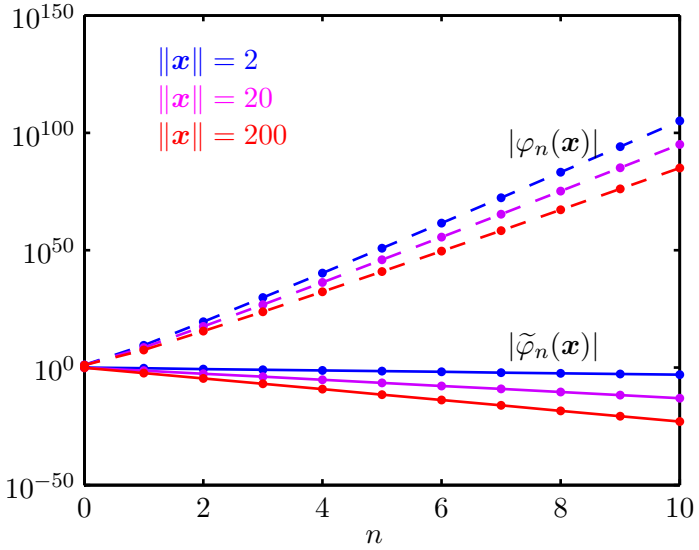
The numerical evaluation of the new basis functions must not literally follow the formulas (5.18), because that procedure produces range exceptions in intermediate quantities. Instead, I compute them as follows:

$$\widetilde{\psi}_n(\boldsymbol{x}) = \exp\Big[\log H_n(k\alpha) + \log J_n(k\|\boldsymbol{x}\|) + in\theta(\boldsymbol{x})\Big] \tag{5.20a}$$

$$\widetilde{\varphi}_n(\boldsymbol{x}) = \exp\Big[-\log H_n(k\alpha) + \log H_n(k\|\boldsymbol{x}\|) + in\theta(\boldsymbol{x})\Big] \tag{5.20b}$$



(a)



(b)

Figure 5.11: New basis functions for $k = 10^{-10}$ and $\alpha = 1$. (a) Superexponential decay in functions $\{\psi_n\}$ (dashed lines) is replaced with exponential decay in functions $\{\widetilde{\psi}_n\}$ (solid lines). (b) Superexponential growth in functions $\{\varphi_n\}$ (dashed lines) is replaced with exponential decay in functions $\{\widetilde{\varphi}_n\}$ (solid lines).

This approach requires computational routines that return the complex logarithms $\log J_n(x)$ and $\log H_n(x)$. The exponential functions in (5.20a) and (5.20b) make irrelevant the choice of branch cut. Specialized code is required because $J_n(x)$ underflows well before $\log J_n(x)$ does. Likewise, $H_n(x)$ overflows well before $\log H_n(x)$ does. The implementation should return an accurate value of $\log H_n(x)$ even if $H_n(x)$ overflows.

Many other stable bases can be chosen, but a considerable advantage of the choice made here is that the Bessel function addition theorem can still be used to derive a formula for the translation matrix elements in the new basis. The translation from an expansion on the exterior of a disk with radius α to an expansion on the interior of a well-separated disk with radius β is

$$b_m = \sum_{n=-\infty}^{\infty} a_n H_{m-n}(k\|\boldsymbol{d}\|)e^{-i(m-n)\theta(-\boldsymbol{d})}, \quad m \in \mathbb{Z}, \tag{5.21}$$

which after renormalization becomes

$$\widetilde{b}_m = \sum_{n=-\infty}^{\infty} \widetilde{a}_n \big(H_n(k\alpha)H_m(k\beta)\big)^{-1} H_{m-n}(k\|\boldsymbol{d}\|)e^{-i(m-n)\theta(-\boldsymbol{d})}, \quad m \in \mathbb{Z}. \tag{5.22}$$

After series truncation, this operation becomes $\widetilde{\boldsymbol{b}} = \widetilde{T}\widetilde{\boldsymbol{a}}$. Note that $\widetilde{T}$ is a row and column scaling of T, and the scaling gives the elements of $\widetilde{T}$ a much smaller dynamic range than the elements of T. Again, to avoid overflowing intermediate quantities, the matrix elements should be computed as

$$\widetilde{T}_{mn} = \exp\Big[-\log H_n(k\alpha) - \log H_m(k\beta) + \log H_{m-n}(k\|\boldsymbol{d}\|) - i(m-n)\theta(-\boldsymbol{d})\Big]. \tag{5.23}$$

Similar changes can be applied to the leaf operators U and V, and to the branch operators R and S.

Unfortunately, scaling the rows and columns of the translation matrix T eliminates its Toeplitz structure. The fast translation algorithm of Section 2.3.2 is no

longer available. But since the scaling is only necessary for optically small charge clusters, for which p is small, a slow $O(p^2)$ translation for those clusters will not damage the overall multipole complexity of $O(N \log^{\varsigma} N)$ flops.

Chapter 6

Conclusion

Before closing with a discussion of extensions and related work, I comment briefly on the restriction to 2-D scattering problems. After all, that restriction limits the immediate value of the work to the engineering community.

My interest in numerical scattering originated with the computation of the mutual element impedances of large antenna arrays. In the course of my studies, I continually simplified the scattering geometry, trading realistic engineering complexity for a stronger mathematical footing. The product of that shift emphasizes accuracy and speed, the currency of numerical analysis. I believe that it makes more sense to build engineering complexity on top of this foundation than to try to build speed and accuracy into a slow and inaccurate treatment of general 3-D problems with multiple materials and complicated interfaces.

That is not to say that the problems I have been solving are simple. While the results presented here have been restricted to connected 2-D metallic obstacles, some of those obstacles span thousands of wavelengths. Because they require so

many variables, the associated scattering problems are challenging. The greatest difficulty in three dimensions is the same: Solutions are rapidly oscillating functions that must be sampled at a large number of points.

6.1 Extensions

Laplace and Poisson

In the limiting case $k \to 0$ of zero oscillation frequency, the Helmholtz equation reduces to the Laplace equation, which brings into scope an impressive assortment of applications. In empty regions of space, Newtonian gravitational potentials and Coulomb electrostatic potentials are both solutions of the Laplace equation. Multipole has been used to speed up the computation of parasitic capacitances [111] and inductances [89] in high-speed integrated circuits. In the field of computational chemistry, multipole has also been used to speed up the computation of long-range interatomic forces in large molecular simulations [41] [123].

The Poisson equation—the nonhomogeneous Laplace equation—is important in the modeling of solid-state electronic devices. It is also an important component of many numerical treatments of the equations that govern fluid flow [3, §5.9–5.12]. Multipole has been used to speed up vortex methods for fluid simulation [120] [134].

Multiple Obstacles

It is merely a programming exercise to extend my existing 2-D code to treat a disconnected boundary Γ. An interesting example of such multiple obstacles is given by Greengard and Moura [69], who compute the flow of electric current through

composite materials. That application is magnetostatic ($k = 0$), but high frequency computations are also important in materials science. With the extension to multiple obstacles, the Helmholtz solver described here can be applied to scattering structures in photonic bandgap materials [140] [26].

Other Boundary Conditions

Another extension of the code might treat various other boundary conditions. If the Dirichlet condition is replaced with a Neumann condition, then a new integral operator $\mathcal{D}'$ arises from the CSIE formulation,

$$(\mathcal{D}'\chi)(\boldsymbol{x}) := \frac{\partial}{\partial n(\boldsymbol{x})} \int_\Gamma \frac{\partial \Phi(\boldsymbol{x}, \boldsymbol{y})}{\partial n(\boldsymbol{y})} \chi(\boldsymbol{y})\, d\Gamma(\boldsymbol{y}) \quad \text{where} \quad \boldsymbol{x} \in \Gamma. \tag{6.1}$$

This operator evaluates the normal derivative of the field produced by a dipole layer χ. It is a hypersingular integral operator: The formal exchange of the outer differentiation with the integration produces an integrand that is nonintegrable. Nevertheless, a high-order discretization for (6.1) can be developed in the same way that we developed rules for the single layer operator $\mathcal{S}$. Kress [97] has extended the global spectral rule of Section 1.6 to cover $\mathcal{D}'$.

Three Dimensions

Rokhlin [118] has extended scattering multipole from $\mathbb{E}^2$ to $\mathbb{E}^3$. As in his presentation [117] of the 2-D algorithm, he mentions the hierarchical algorithm in an offhand way. As they wrote code, other researchers filled in some important details. Those efforts were conducted more or less simultaneously by Chew et al. [125] [126] at the University of Illinois, by Dembart and Yip [39] at Boeing, and by Wandzura et al. [29] [75] at HRL Laboratories.

The basis functions for scalar 3-D spectral expansions are the natural generalization of the 2-D basis functions. In two dimensions, the expansions represent fields inside or outside a disk. In three dimensions, they represent fields inside or outside a ball. The 3-D expansions are double series, indexed by two integers m and n. In spherical coordinates (r, θ, ϕ), the exterior basis functions $\{\varphi_{mn}\}$ and interior basis functions $\{\psi_{mn}\}$ are

$$\varphi_{mn}(r, \theta, \phi) := h_m(kr) Y_{mn}(\theta, \phi), \tag{6.2a}$$

$$\psi_{mn}(r, \theta, \phi) := j_m(kr) Y_{mn}(\theta, \phi), \tag{6.2b}$$

for nonnegative integers m and integers $-m \leq n \leq m$. The functions h_m and j_m are spherical Bessel functions [1, Ch. 10]. The functions Y_{mn} are spherical harmonics, defined by

$$Y_{mn}(\theta, \phi) := c_{mn} P_m^n(\cos\theta) e^{in\phi}, \tag{6.3}$$

where P_m^n is an associated Legendre function [1, Ch. 8] and c_{mn} is a normalization constant. These basis functions are, like the 2-D basis functions, numerically unstable. Stabilization measures patterned after those in Chapter 5 can, however, restore stability.

Unfortunately, the 3-D translation operators are substantially more complicated than their 2-D counterparts. In two dimensions, each matrix element requires little more than a single Bessel function evaluation. In three dimensions, the matrix elements are sums of Wigner 3-j symbols [53]. Ongoing research [74] aims to make the computation as efficient as possible. Meanwhile, in MATLAB a user has easy access, through functions like `besselj` and `besselh`, to Amos's venerable FORTRAN Bessel routines [6] [7]. But, as with many of the less common special functions, there is no access to Wigner 3-j symbols, or to their close relatives the Clebsch–Gordon coefficients.

For translations that do not require special treatment to avoid numerical instability, the expensive evaluation of the matrix elements can be bypassed if the coefficients $\{a_{mn}\}$ and $\{b_{mn}\}$ of the spectral expansions are suppressed in favor of their spherical Fourier transforms. The transform of the exterior coefficients is

$$\widetilde{a}(\theta,\phi) := \sum_{m=0}^{\infty} \sum_{n=-m}^{m} a_{mn} Y_{mn}(\theta,\phi), \tag{6.4}$$

which is essentially the far field pattern of the expansion. As in the 2-D case, the interior coefficients $\{b_{mn}\}$ must be weighted for large values of m to prevent their Fourier transform from diverging.

The Fourier transforms $\widetilde{a}$ and $\widetilde{b}$ are sampled at a finite set of points on the unit sphere. As in two dimensions, the bandwidth of those functions increases with the optical size of the charge cluster, so bandlimited interpolations must be performed as the multipole tree is traversed. While in two dimensions the standard bandlimited interpolation of far field patterns does not dominate the complexity, in three dimensions it does. Recent work [84] has produced a faster bandlimited interpolation algorithm on the sphere. The method relies on a subsidiary fast multipole method for particles distributed on a line [45] [141]. Another approach [94] [32] oversamples the transforms, but replaces the global bandlimited interpolations with local polynomial interpolations.

Instead of using spherical Fourier transforms to diagonalize the translations of infinite coefficient sequences, we could try to unravel the algebraic structure of the linear transformations of the finite-length coefficient vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. While the 2-D translation matrices are Toeplitz matrices, the 3-D translation matrices are only block Toeplitz. If the 2-D complexity $O(N \log^{\varsigma} N)$ is to be preserved, some remaining structure in the translation matrices must be utilized. If only the block Toeplitz structure is used, the 3-D algorithm will require $O(N^{3/2} \log^{\varsigma} N)$ flops, where

ς is a small constant.

Turning to vector scattering problems, in the context of a Galerkin discretization of the 3-D EFIE, Coifman, Rokhlin, and Wandzura [29] briefly consider the modifications necessary to treat the vector field solution of Problem 1.2. One approach is to represent each rectangular component of the electric field with the scalar basis (6.2). But since the components are coupled in order to satisfy $\nabla \cdot \boldsymbol{E} = \boldsymbol{0}$, that approach is not as efficient as one that uses the vector bases

$$\big\{\nabla \times \hat{\boldsymbol{r}}\varphi_{mn}, \nabla \times \nabla \times \hat{\boldsymbol{r}}\varphi_{mn} : m \geq 0, -m \leq n \leq m\big\}, \tag{6.5a}$$

$$\big\{\nabla \times \hat{\boldsymbol{r}}\psi_{mn}, \nabla \times \nabla \times \hat{\boldsymbol{r}}\psi_{mn} : m \geq 0, -m \leq n \leq m\big\}, \tag{6.5b}$$

on the exterior and interior of a ball, where φ_{mn} and ψ_{mn} are taken from (6.2).

Other Constant-Coefficient PDEs

In principle, a fast multipole method can be developed for any shift-invariant linear differential operator. Gimbutas and Rokhlin [61] use a generic tensor product multipole basis capable of approximating any analytic field with spectral accuracy. The translation matrix elements are then constructed by brute force.

In practice, for more than one space dimension a substantial gain in efficiency can be realized if the multipole basis is specialized to the differential operator. As in Chapter 2, that is typically done with a separation of variables analysis of the PDE on the interior and exterior of a simple domain. Addition theorems provide formulas for the translation matrix elements.

Table 6.1 lists a selection of differential operators for which the multipole construction has been carried out. Subscripts have been used to indicate partial derivatives with respect to rectangular coordinates. The constant k is everywhere a positive

real number, and f is a prescribed source function. Listed next to the equations are the fundamental solutions that satisfy the usual boundary condition at infinity. Those solutions are given in polar coordinates, up to a constant multiplier, for a point source at the origin.

Table 6.1: Selected Fast Multipole Methods

Two Space Dimensions

Name	Equation	Fund. Solution	References
Poisson	$u_{xx}+u_{yy} = f$	$\log r$	[116]
Helmholtz	$u_{xx}+u_{yy}+k^2u = f$	$H_0(kr)$	[117]

Three Space Dimensions

Name	Equation	Fund. Solution	References
Poisson	$u_{xx}+u_{yy}+u_{zz} = f$	r^{-1}	[66] [70]
Helmholtz	$u_{xx}+u_{yy}+u_{zz}+k^2u = f$	$r^{-1}e^{ikr}$	[118]
Modified Helmholtz	$u_{xx}+u_{yy}+u_{zz}-k^2u = f$	$r^{-1}e^{-kr}$	[14] [72]
Heat	$u_{xx}+u_{yy}+u_{zz}-ku_t = f$	$t^{-3/2}e^{-(k/4)r^2/t}$	[71]

The 3-D wave equation $u_{xx} + u_{yy} + u_{zz} - c^{-2}u_{tt} = f$ has a fundamental solution $r^{-2}\delta(r - ct)$. That interaction has a character distinct from those in Table 6.1, all of which are analytic away from the origin. Ergin, Shanker, and Michielssen [54] have nevertheless developed a multipole method for the wave equation. The particles taking part in the interaction are points in space, but their time waveforms are not impulse functions. Because it does not have a sharp trailing edge [10, §I.6], the fundamental solution of the 2-D wave equation presents an even greater difficulty [102].

As indicated in Section 1.3.1, another approach to the wave equation relies on Fourier synthesis of solutions to the Helmholtz equation. For each member of a finite set of wavenumbers, the multipole algorithms of Chapter 2 may be applied.

Variable-Coefficient PDEs

Multipole methods can be profitably used in two different numerical treatments of a variable-coefficient PDE such as

$$\Delta u(\boldsymbol{x}) + k^2\big(1 + r(\boldsymbol{x})\big)u(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \mathbb{E}^2, \tag{6.6}$$

in which r is the electric susceptibility and $\sqrt{1+r}$ is the index of refraction of a material that fills the plane. For electromagnetic scattering from bounded obstacles, the function r vanishes outside a bounded set G_-. Transmission boundary conditions apply at any jump discontinuities of r, such as on the obstacle boundaries and on any internal material interfaces.

One approach uses a multipole method for a nearby constant-coefficient equation as a preconditioner for an iterative variable-coefficient PDE solver [132]. The constant-coefficient approximation may be obtained by averaging r over the obstacle interior G_-. Boundary integral equations can also be established if the nonhomogeneous material is approximated by a piecewise homogeneous structure, so that the averaging of r is performed separately over each subdomain of a partition of G_-.

A more typical approach to (6.6) uses volume integral equations. If r is smooth on the whole plane, even at the boundary of G_-, then u may be represented as the field generated by an unknown current density J,

$$u(\boldsymbol{x}) = -ik\eta \int_{G_-} \Phi(\boldsymbol{x}, \boldsymbol{y}) J(\boldsymbol{y})\, d\boldsymbol{y} \quad \text{for} \quad \boldsymbol{x} \in \mathbb{E}^2. \tag{6.7}$$

The current J is, like q, smooth on the whole plane, but it is nonzero only on G_-. Substitution of (6.7) into (6.6) gives the Lippmann–Schwinger equation [31], a

volume integral equation of the second kind,

$$J(\boldsymbol{x}) + k^2 r(\boldsymbol{x}) \int_{G_-} \Phi(\boldsymbol{x}, \boldsymbol{y}) J(\boldsymbol{y})\, d\boldsymbol{y} = -(ik\eta)^{-1} f(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in G_-. \tag{6.8}$$

A suitable discretization of (6.8), patterned after the one in Section 3.1, produces an algebraic system to which a multipole method can be applied. The construction of a high-order discretization will be made easier if in the Lippmann–Schwinger equation G_- is replaced with a rectangular region $R \supset G_-$, but then (6.8) will no longer be a second-kind equation.

In electromagnetics problems, r is usually discontinuous at the obstacle boundary Γ. Then the volume sources J should be supplemented by surface distributions on Γ. If r is piecewise constant in $\mathbb{E}^2$, then surface distributions can be placed on all the curves of discontinuity, and no volume sources are required.

Note that the motivation to transform the PDE into an integral equation is weakened if volume integrals are necessary. For both differential equation and integral equation, the computational grid fills a 2-D region. Low-order finite difference and finite element methods are, by virtue of their sparsity, likely to be the preferred approaches.

6.2 Related Work

This work addresses the solution of scattering problems such as Problem 1.1 by the fast multipole method. Its three main contributions are

- The block spectral singular quadrature rules of Chapter 3
- The direct solver of Chapter 4
- The basis stabilization of Chapter 5

My research has been influenced by a number of sources, a few of which I now briefly describe.

6.2.1 Singular Quadrature Rules

I had been reading Colton and Kress [31] and was impressed with the spectral singular quadrature rule they apply to 2-D obstacle scattering problems. (That rule is given in Section 1.6 for the EFIE.) Colton and Kress themselves attribute the rule to previous work by Martenson [103] and Kussmaul [99]. I am continually puzzled that the rule is not presented in more texts on computational electromagnetics. For example, Peterson, Ray, and Mittra [114] have written one of the best such books, but in its lengthy treatment of scattering integral equations in two dimensions, it never progresses beyond crude discretizations that facet the boundary and produce piecewise constant or piecewise linear solutions.

At the same time, I was learning about the fast multipole method by reading the papers of Rokhlin, Greengard, and their collaborators. (Greengard and Rokhlin's treatment [64] of Newton–Coulomb particle interactions in the plane is probably the best place to start.) It became clear that the spectral discretization used by Colton and Kress is incompatible with existing multipole methods.

Although they are conceptually simpler than the rules I have developed, the singular quadratures rules applied to scattering integral equations by Rokhlin [119], and subsequently by Kapur and Rokhlin [90], are numerically unstable at high orders. Like Newton–Cotes rules, those rules use equally spaced quadrature nodes. And just like high-order Newton–Cotes rules, the quadrature weights become large and oscillatory. Acute rounding errors occur in floating-point arithmetic. Of course, low-order Newton–Cotes rules are used all the time, and so too the rules produced

by Rokhlin and his colleagues suffer no ill effects at sufficiently low orders.

Work has been carried out [130] [133] to stabilize the rules at high orders. Order of accuracy can be traded for stability. Instead of making a P-point rule exact for functions in a P-dimensional linear space, the rule is designed to be exact for a linear space of lower dimension. The remaining degrees of freedom are used to minimize the norm of the weight vector. The same approach can be taken with high-order Newton–Cotes rules. Nobody does it, because much better high-order rules are available.

I have not presented a comparison of those rules with the new block spectral rules. In my limited experiments to date, the block spectral rules at low orders outperform Kapur and Rokhlin's rule [90, column 1 of Table 6] of the same order. The block spectral rules can reach much higher orders. (I have experimented with rules of order 288.) The block spectral rules are easier to generate. (They do not require variable-precision arithmetic.) On the other hand, the kernel splitting certainly makes the application of the block spectral rules more cumbersome.

The block spectral rules are more easily applied to boundaries with corners. The rules are based on piecewise polynomial interpolations, and the break points of the interpolant can include any corners. Mesh refinement is accomplished by adding more break points near the corners.

6.2.2 Stabilization

After reading Rokhlin's papers, I was unprepared to expect severe numerical instability in my multipole implementation. To my dismay, I found that the instability in fact seriously limited the scope of problems I wished to solve. Arriving at an effective way to manage that behavior was a long and frustrating process. With

the solution of Chapter 5 implemented, however, I no longer have to worry about stability issues. I am free to concentrate on solver efficiency.

After I settled on the basis renormalization presented in Chapter 5, I discovered some work of Zhao and Chew [143] [144] [27, Ch. 5], who had already reached a similar conclusion.

As indicated in Table 6.2, however, our scalings are not the same. The new basis functions are given for the exterior and interior of a disk with radius α.

Table 6.2: Two Basis Renormalizations

Author(s)	Exterior Basis	Interior Basis
Zhao and Chew [143]	$(k\alpha)^n H_n(kr)e^{in\phi}$	$(k\alpha)^{-n} J_n(kr)e^{in\phi}$
Pals	$H_n(k\alpha)^{-1} H_n(kr)e^{in\phi}$	$H_n(k\alpha) J_n(kr)e^{in\phi}$

The renormalization presented here is stronger. Zhao and Chew's polynomial scaling does not eliminate the superexponential growth of the exterior basis functions $\{\varphi_n\}$ as $|n| \to \infty$. Nor does it eliminate the superexponential decay of the interior basis functions $\{\psi_n\}$. It may nevertheless be quite sufficient. I have not yet performed any comparisons of the two scalings.

In Chapter 5, I distinguish between high and low frequency instabilities. I do not treat the high frequency instability with basis renormalization. It is only necessary in that case to replace some fast translation operations with slow translations.

The high frequency instability in particular puzzled me for a while. The instability is clearly linked to the FFT, but the FFT is known [82, Ch. 24] to be stable. I have endeavored to uncover the heart of the matter, and I do not think the instabilities have been analyzed anywhere else in the detail appearing in Chapter 5.

6.2.3 Direct Solver

The direct solver in Chapter 4 draws some inspiration from work by Chandrasekaran and Gu [22] on fast and stable solvers for dense linear systems with coefficient matrices that are the sum of a banded matrix and a semiseparable matrix. A *semiseparable* matrix is constructed by joining the upper and lower triangular parts of two low-rank matrices.

For the high-order numerical solution of two-point boundary value problems, Greengard and Rokhlin [65] present a direct solver for the block diagonal plus semiseparable system produced by a discretization of an equivalent volume integral equation. Starr and Rokhlin [131] extend the method to first-order ODE systems, which after discretization also give block diagonal plus semiseparable matrices. Starr [130] extends the solver still further to cover weakly singular integral equations

$$\lambda\varphi(s) - \int_0^1 K(s,t)\varphi(t)\,dt = f(t), \quad s \in [0,1], \tag{6.9}$$

in which the kernel $K(s,t)$ is smooth and nonoscillatory away from the diagonal $s = t$. The coefficient matrices in that case are not banded plus semiseparable, but have a hierarchical low rank structure away from the main diagonal.

The Greengard–Starr–Rokhlin solver is based on the recursive application of formulas related to the Sherman–Morrison–Woodbury formula,

$$(A + UV^H)^{-1} = A^{-1} - A^{-1}U(I + V^H A^{-1} U)^{-1} V^H A^{-1} \tag{6.10}$$

Solving an updated linear system using this formula is known to be numerically unstable [82, Prob. 26.2], although a full characterization of its roundoff behavior has not been given. The Greengard–Starr–Rokhlin solver also assumes that each submatrix on the block diagonal is nonsingular, and Eidelman and Gohberg [48]

give a simple example for which that assumption is violated. But it is the reliance on the Woodbury formula that makes the instability pervasive.

Chandrasekaran and Gu's direct solver is a stable alternative to the Greengard–Starr–Rokhlin solver. It is also a stable alternative to the Woodbury formula when A is banded and U and V are skinny matrices.

The solution technique employed by Chandrasekaran and Gu has been extended to cover dense matrices that arise in the study of time-varying linear systems with sparse state structure [21]. In their book [40], Dewilde and van der Veen have given a factorization algorithm for such matrices. Eidelman and Gohberg [47] [46] have analyzed matrix structures generated by time-varying systems with scalar input and output. They have also made recent contributions [49] to the algebra of vector I/O systems with sparse state structure.

Chandrasekaran et al. [23] have further generalized their solver to treat matrices with a hierarchical structure that is nearly the same as the one studied by Starr in his dissertation.

Hackbusch and his colleagues have also systematically studied dense matrices with hierarchical structure [76] [77]. There are two connections with the research presented here. First, their $\mathcal{H}^2$ matrices [79] have exactly the structure utilized by a fast multipole method for a nonoscillatory 1-D interaction. Second, their notion of weak admissibility [78] simplifies that structure so that it is exactly the one treated by the hierarchical solver of Chandrasekaran and his collaborators.

The three generations of solvers constructed by Chandrasekaran et al. all work by sequentially deflating the dense linear system after eliminating matrix elements by the application of unitary transforms from the left- and right-hand sides. In each case—banded plus semiseparable, sequentially semiseparable, hierarchically

semiseparable—a different fast and stable solver can be obtained by the method used in Chapter 4. The fast matrix-vector product is expressed as a dag, the dag is embedded into a large sparse matrix, and sparse Gaussian elimination with partial pivoting solves the system.

That idea can be illustrated with the Woodbury formula. If A is banded and U and V are skinny matrices, then in lieu of using (6.10) to solve $(A + UV^H)\boldsymbol{x} = \boldsymbol{b}$, we can solve the sparse system

$$\begin{bmatrix} A & U \\ V^H & -I \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}, \tag{6.11}$$

where $\boldsymbol{y} := V^H\boldsymbol{x}$ is the vector of intermediate variables introduced to form the sparse system.

For block diagonal plus semiseparable matrices, Eidelman and Gohberg [50] present a direct solver that utilizes the solution of a larger banded system. That sparse system is not, however, the same one that would be obtained if the construction in Chapter 4 were applied to a block diagonal plus semiseparable matrix.

The sparse matrix approach is more flexible than the Chandrasekaran–Gu solver. For instance, in (6.11) A is not restricted to have band structure, and arbitrary sparsity patterns can be just as easily accommodated. I need the flexibility of the sparse matrix approach because

- The multipole dag is more unstructured than the dag of a hierarchically semiseparable matrix.
- At high frequencies, the off-diagonal matrix blocks do not have low rank.

For eligible matrices, though, the Chandrasekaran–Gu solver is likely to be faster than the sparse matrix approach.

I am aware of no other fast direct solver that treats general impenetrable obstacles at high frequencies. For *elongated* impenetrable obstacles, other attempts [108] [17] at a direct solver have been made. For penetrable obstacles, direct solution of the Lippmann–Schwinger equation has been met with greater success [28] [101] [25].

6.3 Future Directions

The results presented in this dissertation are the product of a research program that can continue in a number of directions. In conclusion, I highlight three of them.

Parallel Iterative Solver

At the end of Section 4.1, I discuss the attraction of the sparse matrix representation of the multipole dag from the standpoint of a parallel iterative solver. Those ideas should be tested in the near future. The implementation appears to be painless, and certainly it is much simpler than existing parallel multipole solvers [142] [139] [27, Ch. 4].

If, in addition to the solution vector $\boldsymbol{q}$, the spectral expansion coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ are also approximated in the Krylov space, then no interprocessor communication is needed to execute the multipole matrix-vector multiplication. Only the communication required by the Krylov iteration stands in the way of ideal parallel scaling.

Unanswered questions: Is the condition number of the sparse matrix significantly worse than that of the dense interaction matrix? How effective is the fast preconditioner obtained by dropping U or V from the sparse matrix?

B-Spline Quadrature

The block spectral quadrature rules developed in Chapter 3 strike me as deficient in one way. If I were to start over from scratch, I think I would try basing the quadrature on a B-spline [38] polynomial interpolation rather than on polynomial interpolation at Chebyshev nodes.

Under accuracy scaling, in which grid points are added to reduce the discretization error, I have no complaint with the Chebyshev nodes. In computational electromagnetics, however, we are usually more interested in frequency scaling, in which grid points are added to keep up with the moving Nyquist barrier as frequency is increased. In that circumstance, the clustering of the Chebyshev grid points becomes a liability. N must be large enough so that, even where the boundary grid is least dense, the Nyquist rate is exceeded.

Splines encourage the use of a more uniformly spaced boundary grid. In a P-point Chebyshev interpolation, all P degrees of freedom are devoted to increasing the polynomial degree. In a P-point spline interpolation, the polynomial degree is less than P. The remaining degrees of freedom are spent on suppressing the Runge oscillations [55] [135] well-known to plague uniformly spaced polynomial interpolations. Unlike other splines, the B-spline basis functions have compact support, necessary for the construction of a discretization compatible with the fast multipole method.

If two boundary grids, one of N_{Cheby} Chebyshev nodes and another of N_{Spline} equally spaced spline nodes, are designed to both satisfy the Nyquist criterion, then

$$N_{\text{Cheby}} = \frac{\pi}{2} N_{\text{Spline}} \tag{6.12}$$

expresses the expected asymptotic inefficiency of the Chebyshev points.

I confess that I find the steps taken in Chapter 5 to be an inelegant—though effective—solution to scattering multipole's numerical instability. I have little stomach for implementing the ideas in three dimensions, where the translation matrix elements are tougher to compute.

The poor asymptotic behavior of the spherical wave functions (6.2) is a barrier to more widespread deployment of the fast multipole method. A global renormalization cures the instability, but distorts the structure of the translations, destroying the algorithm's efficiency.

Is there a basis that is both fast and numerically stable?

Some recent work indicates that there is. To speed up the translations among well-separated particle clusters for the 3-D Laplace interaction, Greengard and Rokhlin [70] have replaced the spherical boundary of the exterior spectral expansions with a planar boundary. They subsequently suggest that a similar measure will cure the instability of the multipole method for Helmholtz interactions.

To construct the new basis functions, the Helmholtz equation is solved on the half-space by separation of variables. That solution does not take the form of an infinite double series, but rather an improper double integral. The integral is the continuous summation of plane waves, some propagating and some evanescent. Its discretization is more complicated: The series only requires truncation, while the integral requires a quadrature rule on an unbounded domain.

The flaw with the approach outlined by Greengard et al. [68] is that the plane wave basis is only applied selectively. While they switch between the spherical wave basis and the plane wave basis, the spherical wave basis—both exterior and

interior—needs to be abandoned once and for all. Darve and Havé [33] realize that, and their recent work may embody the best solution.

A plane wave basis would also benefit the direct solver. Since the plane wave expansion must be taken in several directions, Ξ will have a larger order. But since the translation matrices are diagonal in that basis, the matrix Ξ becomes more sparse, and there is no leftover Toeplitz structure that fails to be utilized by a sparse Gaussian elimination code.

Bibliography

[1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions : with Formulas, Graphs, and Mathematical Tables*. U.S. Government Printing Office, Washington, DC, corrected edition, 1972.

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

[3] William F. Ames. *Numerical Methods for Partial Differential Equations*. Academic Press, Boston, third edition, 1992.

[4] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[5] S. Amini and A. Profit. Analysis of the truncation errors in the fast multipole method for scattering problems. *Journal of Computational and Applied Mathematics*, 115(1–2):23–33, 2000.

[6] D. E. Amos. Algorithm 644 : A portable package for Bessel functions of a complex argument and nonnegative order. *ACM Transactions on Mathematical Software*, 12(3):265–273, 1986.

[7] D. E. Amos. A remark on algorithm 644 : A portable package for Bessel functions of a complex argument and nonnegative order. *ACM Transactions on Mathematical Software*, 21(4):388–393, 1995.

[8] C. R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific Computing*, 13(4):923–947, 1992.

[9] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society of Industrial and Applied Mathematics, Philadelphia, third edition, 1999.

[10] Bevan B. Baker and E. T. Copson. *The Mathematical Theory of Huygens' Principle*. Clarendon Press, Oxford, second edition, 1950.

[11] Josh Barnes and Piet Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[12] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods*. Society of Industrial and Applied Mathematics, Philadelphia, second edition, 1994.

[13] C. Leonard Berman. Grid–multipole calculations. *SIAM Journal on Scientific Computing*, 16(5):1082–1091, 1995.

[14] Alexander H. Boschitsch, Marcia O. Fenley, and Wilma K. Olson. A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions. *Journal of Computational Physics*, 151(1):212–241, 1999.

[15] J. J. Bowman, T. B. A. Senior, and P. L. E. Uslenghi, editors. *Electromagnetic and Acoustic Scattering by Simple Shapes*. Hemisphere Publishing, New York, revised edition, 1987.

[16] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial*. Society of Industrial and Applied Mathematics, Philadelphia, second edition, 2000.

[17] Francis X. Canning and Kevin Rogovin. Fast direct solution of standard moment-method matrices. *IEEE Antennas and Propagation Magazine*, 40(3):15–26, 1998.

[18] B. Carpentieri, I. S. Duff, and L. Giraud. Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism. *Numerical Linear Algebra with Applications*, 7:667–685, 2000.

[19] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):669–686, 1988.

[20] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88/89:67–82, 1987.

[21] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen. Fast stable solver for sequentially semi-separable linear systems of equations. *Lecture Notes in Computer Science*, 2552:545–554, 2002. (Proceedings of 9th International Conference on High Performance Computing, Bangalore, India).

[22] S. Chandrasekaran and M. Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM Journal on Matrix Analysis and Applications*, 25(2):373–384, 2003.

[23] S. Chandrasekaran, M. Gu, and T. Pals. Fast and stable algorithms for hierarchically semi-separable representations. Technical report, Department of Mathematics, University of California, Berkeley, 2004.

[24] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. On rank-revealing factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15(2):592–622, 1994.

[25] Yu Chen. A fast direct algorithm for the Lippmann–Schwinger integral equation in two dimensions. *Advances in Computational Mathematics*, 16(2–3):175–190, 2002.

[26] H. Cheng, W. Y. Crutchfield, M. Doery, and L. Greengard. Fast, accurate integral equation methods for the analysis of photonic crystal fibers. I: Theory. *Optics Express*, 12(16):3791–3805, 2004.

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

[28] Weng Cho Chew and Cai-Cheng Lu. The use of Huygens' principle for solving the volume integral equation of scattering. *IEEE Transactions on Antennas and Propagation*, 41(7):897–904, 1993.

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

[30] David Colton and Rainer Kress. *Integral Equation Methods in Scattering Theory*. Wiley, New York, 1983.

[31] David Colton and Rainer Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer, New York, second edition, 1998.

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

[33] Eric Darve and Pascal Havé. Efficient fast multipole method for low-frequency scattering. *Journal of Computational Physics*, 197(1):341–363, 2004.

[34] Philip J. Davis. *Circulant Matrices*. Wiley, New York, 1979.

[35] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press, Orlando, second edition, 1984.

[36] Timothy A. Davis. Algorithm 832 : UMFPACK V4.3 – An unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, 2004.

[37] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):353–376, 2004.

[38] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[39] B. Dembart and E. Yip. A 3D fast multipole method for electromagnetics with multiple levels. In *Review of Progress in Applied Computational Electromagnetics*, volume 1, pages 621–628. Applied Computational Electromagnetics Society, 1995.

[40] Patrick Dewilde and Alle-Jan van der Veen. *Time-Varying Systems and Computations*. Kluwer, Boston, 1998.

[41] Hong-Qiang Ding, Naoki Karasawa, and William A. Goddard, III. Atomic level simulations on a million particles: The cell multipole method for Coulomb and London nonbond interactions. *Journal of Chemical Physics*, 97(6):4309–4315, 1992.

[42] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, New York, 1986.

[43] I. S. Duff and J. K. Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal on Scientific and Statistical Computing*, 5(3):633–641, 1984.

[44] Iain S. Duff and Jacko Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 20(4):889–901, 1999.

[45] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM Journal on Numerical Analysis*, 33(5):1689–1711, 1996.

[46] Y. Eidelman and I. Gohberg. Linear complexity inversion algorithms for a class of structured matrices. *Integral Equations and Operator Theory*, 35(1):28–52, 1999.

[47] Y. Eidelman and I. Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34(3):293–324, 1999.

[48] Y. Eidelman and I. Gohberg. Algorithms for inversion of diagonal plus semiseparable operator matrices. *Integral Equations and Operator Theory*, 44(2):172–211, 2002.

[49] Y. Eidelman and I. Gohberg. A modification of the Dewilde–van der Veen method for inversion of finite structured matrices. *Linear Algebra and its Applications*, 343:419–450, 2002.

[50] Y. Eidelman and I. Gohberg. Fast inversion algorithms for a class of structured operator matrices. *Linear Algebra and its Applications*, 371:153–190, 2003.

[51] Stanley C. Eisenstat and Joseph W. H. Liu. Exploiting structural symmetry in unsymmetric sparse symbolic factorization. *SIAM Journal on Matrix Analysis and Applications*, 13(1):202–211, 1992.

[52] William D. Elliott and John A. Board, Jr. Fast Fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.

[53] Michael A. Epton and Benjamin Dembart. Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. *SIAM Journal on Scientific Computing*, 16(4):865–897, 1995.

[54] A. Arif Ergin, Balasubramaniam Shanker, and Eric Michielssen. The plane-wave time-domain algorithm for the fast analysis of transient wave phenomena. *IEEE Antennas and Propagation Magazine*, 41(4):39–52, 1999.

[55] Bengt Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, New York, 1996.

[56] Roland W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM Journal on Scientific and Statistical Computing*, 13(1):425–448, 1992.

[57] Roland W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM Journal on Scientific Computing*, 14(2):470–482, 1993.

[58] Roland W. Freund and Noël M. Nachtigal. QMR : A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(3):315–339, 1991.

[59] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–367, 1973.

[60] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in MATLAB : Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, 1992.

[61] Zydrunas Gimbutas and Vladimir Rokhlin. A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24(3):796–817, 2002.

[62] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[63] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, San Diego, sixth edition, 2000. Edited by Alan Jeffrey and Daniel Zwillinger.

[64] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[65] L. Greengard and V. Rokhlin. On the numerical solution of two-point boundary value problems. *Communications on Pure and Applied Mathematics*, 44(4):419–452, 1991.

[66] Leslie Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[67] Leslie Greengard. Fast algorithms for classical physics. *Science*, 265(5174):909–914, 1994.

[68] Leslie Greengard, Jingfang Huang, Vladimir Rokhlin, and Stephen Wandzura. Accelerating fast multipole methods for the Helmholtz equation at low frequencies. *IEEE Computational Science & Engineering*, 5(3):32–38, 1998.

[69] Leslie Greengard and Monique Moura. On the numerical evaluation of electrostatic fields in composite materials. *Acta Numerica*, pages 379–410, 1994.

[70] Leslie Greengard and Vladimir Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, pages 229–269, 1997.

[71] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[72] Leslie F. Greengard and Jingfang Huang. A new version of the fast multipole method for screened Coulomb interactions in three dimensions. *Journal of Computational Physics*, 180(2):642–658, 2002.

[73] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996.

[74] Nail A. Gumerov and Ramani Duraiswami. Recursions for the computation of multipole translation and rotation coefficients for the 3-D Helmholtz equation. *SIAM Journal on Scientific Computing*, 25(4):1344–1381, 2003.

[75] Mark F. Gyure and Mark A. Stalzer. A prescription for the multilevel Helmholtz FMM. *IEEE Computational Science & Engineering*, 5(3):39–47, 1998.

[76] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[77] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[78] W. Hackbusch, B. N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73(3):207–243, 2004.

[79] Wolfgang Hackbusch, Boris Khoromskij, and Stefan A. Sauter. On $\mathcal{H}^2$ matrices. In Hans-Joachim Bungartz, Ronald H. W. Hoppe, and Christophe Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29. Springer, New York, 2000.

[80] Roger F. Harrington. *Time-Harmonic Electromagnetic Fields*. McGraw-Hill, New York, 1961.

[81] Roger F. Harrington. *Field Computation by Moment Methods*. Macmillan, New York, 1968.

[82] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society of Industrial and Applied Mathematics, Philadelphia, second edition, 2002.

[83] John David Jackson. *Classical Electrodynamics*. Wiley, New York, second edition, 1975. (Third edition published by Wiley in 1999).

[84] Rüdiger Jakob-Chien and Bradley K. Alpert. A fast spherical filter with uniform resolution. *Journal of Computational Physics*, 136(2):580–584, 1997.

[85] Shidong Jiang and Vladimir Rokhlin. Second kind integral equations for the classical potential theory on open surfaces I: Analytical apparatus. *Journal of Computational Physics*, 191(1):40–74, 2003.

[86] Shidong Jiang and Vladimir Rokhlin. Second kind integral equations for the classical potential theory on open surfaces II. *Journal of Computational Physics*, 195(1):1–16, 2004.

[87] D. S. Jones. *Acoustic and Electromagnetic Waves*. Oxford University Press, New York, 1986.

[88] P. Jones, J. Ma, and V. Rokhlin. A fast direct algorithm for the solution of the Laplace equation on regions with fractal boundaries. *Journal of Computational Physics*, 113(1):35–51, 1994.

[89] Mattan Kamon, Michael J. Tsuk, and Jacob K. White. FASTHENRY: A multipole-accelerated 3-D inductance extraction program. *IEEE Transactions on Microwave Theory and Techniques*, 42(9):1750–1758, 1994.

[90] Sharad Kapur and Vladimir Rokhlin. High-order corrected trapezoidal quadrature rules for singular functions. *SIAM Journal on Numerical Analysis*, 34(4):1331–1356, 1997.

[91] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[92] Oliver Dimon Kellogg. *Foundations of Potential Theory*. J. Springer, Berlin, 1929. (Republished by Dover Publications, New York, 1953).

[93] Andreas Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*. Springer, New York, 1996.

[94] S. Koc, Jiming Song, and W. C. Chew. Error analysis for the numerical evaluation of the diagonal forms of the scalar spherical addition theorem. *SIAM Journal on Numerical Analysis*, 36(3):906–921, 1999.

[95] R. Kress. Minimizing the condition number of boundary integral operators in acoustic and electromagnetic scattering. *Quarterly Journal of Mechanics and Applied Mathematics*, 38(2):323–341, 1985.

[96] Rainer Kress. A Nyström method for boundary integral equations in domains with corners. *Numerische Mathematik*, 58:145–161, 1990.

[97] Rainer Kress. On the numerical solution of a hypersingular integral equation in scattering theory. *Journal of Computational and Applied Mathematics*, 61:345–360, 1995.

[98] Rainer Kress. *Linear Integral Equations*. Springer, New York, second edition, 1999.

[99] R. Kussmaul. Ein numerisches Verfahren zur Lösung des Neumannschen Aussenraumproblems für die Helmholtzsche Schwingungsgleichung. *Computing*, 4:246–273, 1969.

[100] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979.

[101] Cai-Cheng Lu and Weng Cho Chew. The use of Huygens' equivalence principle for solving 3-D volume integral equation of scattering. *IEEE Transactions on Antennas and Propagation*, 43(5):500–507, 1995.

[102] M. Y. Lu, J. G. Wang, A. A. Ergin, and E. Michielssen. Fast evaluation of two-dimensional transient wave fields. *Journal of Computational Physics*, 158(2):161–185, 2000.

[103] E. Martensen. Über eine Methode zum räumlichen Neumannschen Problem mit einer Andwendung für torusartige Berandungen. *Acta Mathematica*, 109:75–135, 1963.

[104] A. W. Maue. Zur Formulierung eines Allgemeinen Beugungsproblems durch eine Integralgleichung. *Zeitschrift für Physik*, 126:601–618, 1949.

[105] J. R. Mautz and R. F. Harrington. H-field, E-field, and combined-field solutions for conducting bodies of revolution. *Archiv für Elektronik und Übertragungstechnik*, 32:157–163, 1978.

[106] Joseph R. Mautz and Roger F. Harrington. A combined-source solution for radiation and scattering from a perfectly conducting body. *IEEE Transactions on Antennas and Propagation*, 27(4):445–454, 1979.

[107] J. Meixner. Die Kantenbedingung in der Theorie der Beugung Elektromagnetischer Wellen an Volkommen Leitenden Ebenen Schirmen. *Annalen der Physik*, 6:2–9, 1949.

[108] E. Michielssen, A. Boag, and W. C. Chew. Scattering from elongated objects: Direct solution in $O(N \log^2 N)$ operations. *IEE Proceedings – Microwaves, Antennas, and Propagation*, 143(4):277–283, 1996.

[109] S. G. Mikhlin. *Linear Integral Equations*. Hindustan Publishing, Delhi, 1960.

[110] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White. Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory. *SIAM Journal on Scientific Computing*, 15(3):713–735, 1994.

[111] K. Nabors and J. White. FastCap – A multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, 1991.

[112] Arch W. Naylor and George R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag, New York, 1982.

[113] Shinichiro Ohnuki and Weng Cho Chew. Numerical accuracy of multipole expansion for 2-D MLFMA. *IEEE Transactions on Antennas and Propagation*, 51(8):1883–1890, 2003.

[114] Andrew F. Peterson, Scott L. Ray, and Raj Mittra. *Computational Methods for Electromagnetics*. IEEE Press, New York, 1998.

[115] A. J. Poggio and E. K. Miller. Integral equation solutions of three-dimensional scattering problems. In R. Mittra, editor, *Computer Techniques for Electromagnetics*, pages 159–264. Pergamon Press, New York, 1973.

[116] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

[117] V. Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics*, 86(2):414–439, 1990.

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

[119] Vladimir Rokhlin. End-point corrected trapezoidal quadrature rules for singular functions. *Computers & Mathematics with Applications*, 20(7):51–62, 1990.

[120] G. Russo and J. A. Strain. Fast triangulated vortex methods for the 2D Euler equations. *Journal of Computational Physics*, 111(2):291–323, 1994.

[121] Youcef Saad and Martin H. Schultz. GMRES : A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[122] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society of Industrial and Applied Mathematics, Philadelphia, second edition, 2003.

[123] H. Schwichtenberg, G. Winter, and H. Wallmeier. Acceleration of molecular mechanic simulation by parallelization and fast multipole techniques. *Parallel Computing*, 25(5):535–546, 1999.

[124] Ian H. Sloan and W. E. Smith. Product integration with Clenshaw–Curtis and related points : Convergence properties. *Numerische Mathematik*, 30(4):415–428, 1978.

[125] J. M. Song and W. C. Chew. Multilevel fast multipole algorithm for solving combined field integral equations of electromagnetic scattering. *Microwave and Optical Technology Letters*, 10(1):14–19, 1995.

[126] J. M. Song, C. C. Lu, W. C. Chew, and S. W. Lee. Fast illinois solver code (fisc). *IEEE Antennas and Propagation Magazine*, 40(3):27–33, 1998.

[127] Jiming Song and Weng Cho Chew. Error analysis for the truncation of multipole expansion of vector Green's functions. *IEEE Microwave and Optical Components Letters*, 11(7):311–313, 2001.

[128] Peter Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36–52, 1989.

[129] Ivar Stakgold. *Green's Functions and Boundary Value Problems*. Wiley, New York, second edition, 1998.

[130] H. P. Starr. *Rapid Solution of One-Dimensional Integral and Differential Equations*. PhD thesis, Yale University, 1993.

[131] Page Starr and Vladimir Rokhlin. On the numerical solution of 2-point boundary value problems. 2. *Communications on Pure and Applied Mathematics*, 47(8):1117–1159, 1994.

[132] John Strain. Fast spectrally-accurate solution of variable-coefficient elliptic problems. *Proceedings of the American Mathematical Society*, 122(3):843–850, 1994.

[133] John Strain. Locally corrected multidimensional quadrature rules for singular functions. *SIAM Journal on Scientific Computing*, 16(4):992–1017, 1995.

[134] John Strain. Fast adaptive 2D vortex methods. *Journal of Computational Physics*, 132(1):108–122, 1997.

[135] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society of Industrial and Applied Mathematics, Philadelphia, 2000.

[136] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. Society of Industrial and Applied Mathematics, Philadelphia, 1997.

[137] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic, San Diego, 2001.

[138] H. A. van der Vorst. Bi-CGSTAB : A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[139] Sanjay Velamparambil, Weng Cho Chew, and Jiming Song. 10 million unknowns: Is it that big? *IEEE Antennas and Propagation Magazine*, 45(2):43–58, 2003.

[140] Stephanos Venakides, Mansoor A. Haider, and Vassilis Papanicolaou. Boundary integral calculations of two-dimensional electromagnetic scattering by photonic crystal Fabry–Perot structures. *SIAM Journal on Applied Mathematics*, 60(5):1686–1706, 2000.

[141] Norman Yarvin and Vladimir Rokhlin. A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics. *Journal of Computational Physics*, 147(2):594–609, 1998.

[142] Yanhong Yuan and Prith Banerjee. A parallel implementation of a fast multipole based 3-D capacitance extraction program on distributed memory multicomputers. *Journal of Parallel and Distributed Computing*, 61(12):1751–1774, 2001.

[143] Jun-Sheng Zhao and Weng Cho Chew. MLFMA for solving integral equations of 2-D electromagnetic problems from static to electrodynamic. *Microwave and Optical Technology Letters*, 20(5):306–311, 1999.

[144] Jun-Sheng Zhao and Weng Cho Chew. Three-dimensional multilevel fast multipole algorithm from static to electrodynamic. *Microwave and Optical Technology Letters*, 26(1):43–48, 2000.