Numerical methods for solving linear elliptic PDEs: Direct solvers and high order accurate discretizations

by

Sijia Hao

B.S.,University of Science and Technology of China, 2009

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2015

This thesis entitled:
Numerical methods for solving linear elliptic PDEs:
Direct solvers and high order accurate discretizations
written by Sijia Hao
has been approved for the Department of Applied Mathematics

Per-Gunnar Martinsson

Gregory Beylkin

James Bremer

Bengt Fornberg

Zydrunas Gimbutas

Date ____________________

Hao, Sijia (Ph.D., Applied Mathematics)

Numerical methods for solving linear elliptic PDEs:

Direct solvers and high order accurate discretizations

Thesis directed by Professor Per-Gunnar Martinsson

This dissertation describes fast and highly accurate techniques for solving linear elliptic boundary value problems associated with elliptic partial differential equations such as Laplace's, Helmholtz equations and time-harmonic Maxwell's equation. It is necessary to develop efficient methods to solve these problems numerically.

The techniques we develop in this dissertation are applicable to most linear elliptic PDEs, especially to the Helmholtz equation. This equation models frequency domain wave-propagation, and has proven to be particularly difficult to solve using existing methodology, in particular in situations where the computational domain extends over dozens or hundreds of wave-lengths. One of the difficulties is that the linear systems arising upon discretization are ill-conditioned that have proven difficult to solve using iterative solvers. The other is that errors are aggregated over each wave-length such that a large number of discretization nodes are required to achieve certain accuracy. The objective of this dissertation is to overcome these difficulties by exploring three sets of ideas: 1) high order discretization, 2) direct solvers and 3) local mesh refinements.

In terms of "high-order" discretization, the solutions to the PDEs are approximated by high-order polynomials. We typically use local Legendre or Chebyshev grids capable of resolving polynomials up to degree between 10 and 40. For solving the linear system formed upon high-order discretization, there exist a broad range of schemes. Most schemes are "iterative" methods such as multigrid and preconditioned Krylov methods. In contrast, we in this thesis mainly focus on "direct" solvers in the sense that they construct an approximation to the inverse of the coefficient matrix. Such techniques tend to be more robust, versatile and stable compared to the iterative techniques. Finally, local mesh refinement techniques are developed to effectively deal with sharp gradients, localized singularities and regions of poor resolution. A variety of two-dimensional and three-dimensional problems are solved using the three techniques described above.

Dedication

I dedicate this dissertation to my family, especially my parents and my sister, who have encouraged and supported me for so many years.

Acknowledgements

First and foremost, I would like to thank my advisor Gunnar Martisson, who has supported and advised me through my time at University of Colorado. Without his knowledge and help, this dissertation would never have been possible.

Next, I would like to thank all my committee members for their time and interest to my work. A special thank you to Gregory Beylkin who has provided valuable suggestions on my dissertation.

The faculty in the Department of Applied Mathematics are awesome. The Grandview Gang, especially Tom Manteuffel and Steve McCormick have provided guidance in my first two years of study.

During my six years life at Boulder, I am fortunate to make some great friends who helped ease the stress. Kuo Liu and Lei Tang are generous to share their experience. Ying Zhao, Qian Li and Liang Zhang have been so supportive of me and for that, I am grateful.

Finally, thank you to my sister and parents who were always there to encourage me.

Contents

Tables

Figures

Figure

Chapter 1

Introduction

The dissertation describes fast techniques for solving linear elliptic boundary value problems (BVPs) associated with elliptic partial differential operators arise frequently in engineering, mathematics, and physics. Equations like Stokes, Helmholtz and time-harmonic Maxwell are used to model a diverse range of physical phenomena. It is necessary to find efficient techniques to solve these problems numerically. Typically there are two approaches to solving these partial differential equations (PDEs). The first and most common approach is to directly discretize the differential operator via finite difference, finite element and spectral methods. The second approach is to convert the BVPs to boundary integral equations (BIEs) once the fundamental solution is known. The BIEs are then discretized by boundary element, Galerkin or Nyström method. The resulting reduced dimensionality and geometric simplicity allows for high-order accurate numerical solutions with much more efficiency than standard finite difference or finite element discretization.

The techniques we are developing in the dissertation are applicable to most linear elliptic PDEs, but for concreteness, we will mainly focus on the Helmholtz equation. This equation models frequency domain wave-propagation, and has proven to be particularly difficult to solve using existing methodology, in particular in situations where the computational domain extends over dozens or hundreds of wave-lengths. In this situation, the physics of the problem is inherently ill-conditioned, and the linear systems arising upon discretization have proven difficult to solve using iterative solvers. Moreover, due to the aggregation of errors over each wave-length, it is necessary to make the local discretization error tiny in order to have any accuracy in the final answer. Three sets of ideas for overcoming these difficulties are explored:

(1) ***High order discretization.*** Due to the need for tiny local discretization errors, we will in this dissertation exclusively work with so called "high-order" discretizations, in which the solution to the PDE is on each "patch" in the discretization approximated by a high-order polynomial. We typically use local Legendre or Chebyshev grids capable of resolving polynomials up to degree p between 10 and 40. Traditionally, such high order discretizations lead to high expense for forming the linear system, as well as to difficulties in getting iterative solvers to converge. To overcome these problems, we develop novel techniques for evaluating the matrix elements, as well as direct solvers to overcome issues of convergence. Brief introduction to applications of these ideas to BIEs with weakly singular kernels are explored in Section 1.1.1; applications to composite spectral collocation methods are briefly discussed in Section 1.3.1. The first discretization technique is relied on earlier work [69, 2, 64, 66] while the second is similar to earlier work on multidomain pseudospectral methods in [95, 31, 101], especially in Pfeiffer *et al* [84].

(2) ***Direct solvers.*** Consider a linear system

$$\mathsf{A}\mathbf{u} = \mathbf{b}, \tag{1.1}$$

arising from the discretization of a linear boundary value problem. The $N \times N$ matrix A may be either dense of sparse, depending on whether the differential operator was discretized directly, or a boundary integral equation formulation was used. A *direct solver* (as opposed to an *iterative solver*) constructs an operator T such that

$$||\mathsf{A}^{-1} - \mathsf{T}|| \leq \epsilon,$$

where ϵ is a given computational tolerance. Then an approximate solution to (1.1) is obtained by simply evaluating

$$\mathbf{u}_{\text{approx}} = \mathsf{T}\,\mathbf{b}.$$

Direct solvers offer several advantages over iterative ones:

(1) a couple of orders of magnitude speed-up for problems involving multiple right hand sides;

(2) the ability to solve relatively ill-conditioned problems;

(3) increased reliability and robustness.

We are interested in developing *fast* direct solvers. By "fast", we mean that the computational cost is of order $O(N \log^q N)$ where q is a small integer, normally $q = 0, 1$ or 2. Most of the fast schemes rely on rank-deficiencies in the off-diagonal blocks of the matrix. For example, for many one-dimensional BIEs, the matrix to be inverted is *Hierarchically Semi-Separable* (HSS), allowing the direct solver to be accelerated to linear or close to linear complexity. These direct solvers are described in [36, 42], drawing on the early work [77, 20, 8, 80, 93]. We will adopt this technique to reduce computational cost in solving problems in 2D on domains with corners as well as problems involving many 3D scatterers that are well separated (under the constraint that each scatterer is rotationally symmetric), see Sections 1.1.3 and 1.2.

It is also possible to adapt direct solvers to solve the linear systems associated with certain novel multidomain spectral collocation discretizations. We have demonstrated via extensive numerical experimentation that the solution operators in this context can also be represented in the HSS format. Section 1.3.3 outlines a solution strategy that accelerates the scheme for problems in 3D from the $O(N^2)$ complexity that would be attained by an established "multifrontal" solver [29] to $O(N^{4/3})$. This acceleration is similar to techniques developed for classical nested dissection for finite-element and finite-difference matrices in [19, 89, 41].

(3) ***Local mesh refinement.*** In numerical simulation of PDEs, a frequently encountered objective is to effectively deal with sharp gradients, localized singularities and regions of

poor resolution. A natural way is to do local mesh refinement that does not require much à priori knowledge of the solution. Such an algorithm would take an error estimator as input and produce a new discrete model or mesh that reduces the error as an output. However, driving adaptive refinement at a "high level" is never an easy task: it depends on the experienced selection of tolerances and refinement criteria that are highly problem-dependent [61]. In the thesis, we present two local mesh refinement schemes. In Section 1.1.2, we describe an efficient refinement technique for solving BIEs in 2D on domains with corners. Another mesh refinement algorithm adapted to a direct solver via spectral collocation method is proposed in Section 1.3.2. The adaptive algorithm is working in progress.

The three sets of ideas have been explored in a number of manuscripts. Rather than discussing each topic individually, we in this introduction organize the work into three broad "themes," which will be described in Sections 1.1, 1.2, and 1.3. Section 1.4 provides an overview of the specific contributions of each Chapter.

1.1 Theme I: High order methods for boundary integral equations in the plane

Consider the so called "sound-soft" scattering problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega^{\mathrm{c}}, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \\ \dfrac{\partial u(\boldsymbol{x})}{\partial r} - i\kappa u(\boldsymbol{x}) = O(1/r) & r := |\boldsymbol{x}| \to \infty, \end{cases} \tag{1.2}$$

where the "wave number" κ is a real non-negative number. It is known [26] that (1) has a unique solution for every incoming field v. Following standard practice, we reformulate (1) as second kind Fredholm BIE using a so called "combined field technique" [26, 81]. We then look for a solution u of the form

$$u(\boldsymbol{x}) = \int_\Gamma G_\kappa(\boldsymbol{x}, \boldsymbol{x}')\, \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in \Omega^c, \tag{1.3}$$

where G_κ is a combination of the single and double layer kernels,

$$G_\kappa(\boldsymbol{x}, \boldsymbol{x}') = \frac{\partial \phi_\kappa(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} + i\kappa\, \phi(\boldsymbol{x}, \boldsymbol{x}') \tag{1.4}$$

and where ϕ_κ is the free space fundamental solution

$$\phi_\kappa(\boldsymbol{x}, \boldsymbol{x}') = \frac{e^{i\kappa|\boldsymbol{x}-\boldsymbol{x}'|}}{|\boldsymbol{x}-\boldsymbol{x}'|}. \tag{1.5}$$

To obtain an equation for σ, we take the limit in (2) as $\boldsymbol{x}$ approaches the boundary Γ, and find that σ must satisfy the integral equation

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma G_\kappa(\boldsymbol{x}, \boldsymbol{x}')\, \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma. \tag{1.6}$$

1.1.1 High-order Nyström discretization and quadrature schemes

We start with an **underlying** quadrature scheme on $[0, T]$, defined by nodes $\{x_i\}_{i=1}^N$ ordered by $0 \leq x_1 < x_2 < \cdots < x_N < T$, and corresponding weights $\{w_i\}_{i=1}^N$. The Nyström discretization is to use the nodes $\{x_i\}_{i=1}^N$ as **collocation points** where (5) is enforced and to construct a linear system that relates a given data vector $\mathsf{f} = \{f_i\}_{i=1}^N$ to an unknown solution vector $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^N$ to solve. Our first aim is to construct the matrix element $\{a_{i,j}\}_{i,j=1}^N$ such that

$$\int_0^T G_\kappa(\boldsymbol{x}_i, \boldsymbol{x}')\sigma(\boldsymbol{x}')\, d\boldsymbol{x}' \approx \sum_{j=1}^N a_{i,j}\, \sigma(\boldsymbol{x}_j)\ , \tag{1.7}$$

holds to high accuracy. If the kernel G_κ in (5) is smooth, as occurs for the Laplace double-layer operator ($\kappa = 0$), then standard trapezoidal rule and Gaussian quadrature approximate the integral to high accuracy, and the matrix elements are given by

$$a_{i,j} = G_\kappa(x_i, x_j) w_j \tag{1.8}$$

It is less obvious how to construct the matrix $\mathsf{A} = \{a_{i,j}\}$ such that (2.4) holds to high order accuracy in the case where k has a logarithmic singularity.

Fortunately there exist high-order quadratures to integrate these weakly singular kernels, for example, Kapur-Rokhlin, Alpert, modified Gaussian and Kress quadrature, described in earlier work [69, 2, 64, 66]. In Chapter 2, we summarize their key features in view of the differences of numerical construction, requirement to know the kernel explicitly or not, easiness to implement, *etc.* Furthermore, details on how to construct Nyström discretization with these four quadratures and comparisons of their performance are given, where we also provide numerical results for solving problem (1) using different schemes for different wave numbers.

1.1.2 Local mesh refinement

Suppose the domain where we solve the problem (1) is a piecewise smooth contour in the plane but has some corners, for example a tear-shaped domain shown in Figure 1.1. Challenges arise since the integrand exhibits complicated singular behavior near the corner points. To overcome this difficulty, we start with a standard Nyström discretization using a panel based quadrature and partition the contour into two disjoint parts, $\Gamma = \Gamma_1 \bigcup \Gamma_2$, in such a way that Γ_1 is a small piece containing the corner. The piece Γ_2 is smooth, and can be discretized into panels rather coarsely. For the piece Γ_1, we use a simplistic refinement strategy where we recursively cut the panel nearest the corner in half. Once the innermost panel is small enough that its contribution can be ignored, it is simply discarded and the refinement stops.

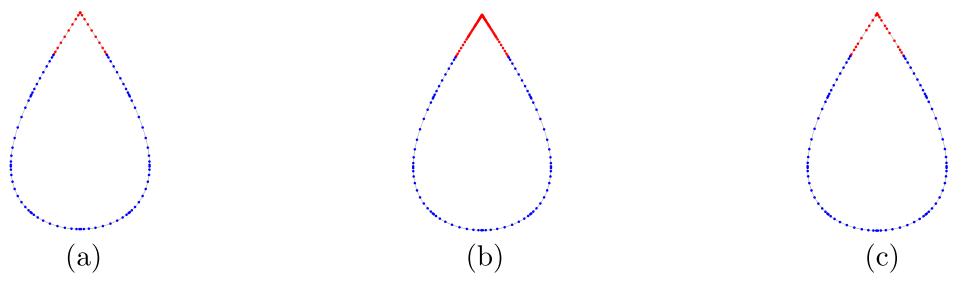


Figure 1.1: The boundary Γ considered in the numerical experiments. (a) The original Gaussian grid before refinement, red nodes in Γ_1 and blue in Γ_2. (b) The locally refined grid. (c) The grid after local compression.

1.1.3 Direct solver

The apparent drawback of a simplistic refinement process is that it can dramatically increase the number of degrees of freedom required in the Nyström discretization. Remarkable progress has been made in [9, 54, 58] to “squeeze out” the degrees of freedom added by the local refinement near the corner via purpose-built local compression techniques. In Chapter 3 we propose a different local compression scheme via a general direct solver [36]. A local compression process is then applied such that a set of skeleton points are automatically picked from among the large number of points used in the refinement. The degrees of freedom are then largely reduced. Most important, the algorithm in Chapter 3 can be executed in time that scales linearly with the number of degrees of freedom added.

1.2 Theme II: Acoustic scattering involving rotationally symmetric scatterers

This section describes the second theme: a robust and highly accurate numerical method for modeling frequency domain acoustic scattering on domain external to a single scatter and a group of scatterers in three dimensions. Specifically, for problem with multiple scatterers, let $\Gamma = \cup_{p=1}^{m}\Gamma_p$ denote the union of m smooth, disjoint, rotationally symmetric surfaces in $\mathbb{R}^3$. We solve problem (1) on the domain exterior to Γ, denoted by Ω, for example the one shown in Figure 5.1. The ability to solve scattering problems in complex geometries to this high level of accuracy has only recently become possible, due to improvements both in constructing high-order Nyström discretizations of the BIEs as described in Section 1.1 and in robust solvers for the resulting linear systems.

1.2.1 A single rotationally symmetric scatter.

We first develop a solution procedure draws an earlier work [86, 103] describing techniques for solving scattering problems involving a single scatterer. The high-order accuracy technique relies on decoupling the BIEs defined on the surface of the scattering body as a sequence of equations defined on a generating contour and discretizing the BIEs using the Nyström method based on

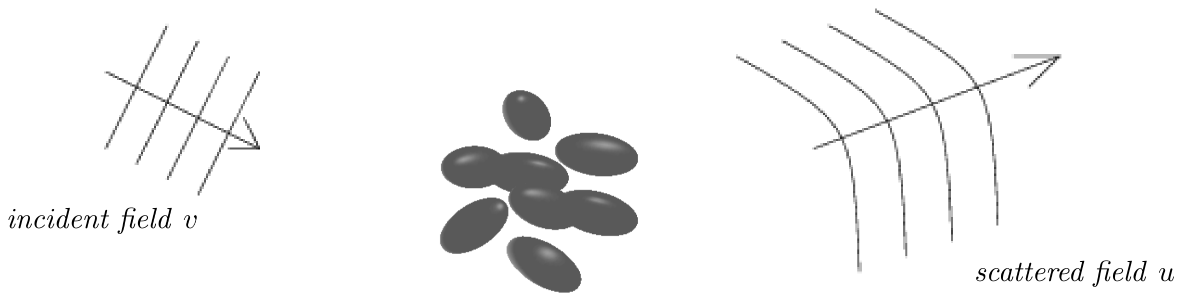


Figure 1.2: Geometry of scattering problem. An incident field v propagates in a medium with constant wave-speed and hits a scattering surface $\Gamma = \bigcup_{p=1}^{m} \Gamma_p$ (shown for $m = 8$). A charge distribution σ is induced on the surface Γ and generates an outgoing field u.

a high-order accurate composite Gaussian quadrature rule. Then the scattering matrix for an individual rotationally symmetric body can very efficiently be constructed and solved by iterative solvers, such as GMRES.

In view of asymptotic complexity, let N denote the total numbers of degree of freedom, the cost of solving the linear system for a single right-hand side is $O(N^2)$. The subsequent solve requires only $O(N^{3/2})$ operations for additional right hand sides. Details can be found in Chapter 4.

1.2.2 Multibody scattering.

We then in Chapter 5 proceed to the general case of m disjoint scattering surfaces where each scatterer is discretized using the tensor product procedure described in Chapter 4. For notational simplicity, we assume that each scatterer is discretized using the same n number of nodes, for a total degree of freedom of $N = mn$. We then seek to construct matrix blocks $\{\mathsf{A}_{p,q}\}_{p,q=1}^{m}$ such that the Nyström discretization of (5) associated with this quadrature rule takes the form

$$\begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} & \cdots & \mathsf{A}_{1,m} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} & \cdots & \mathsf{A}_{2,m} \\ \vdots & \vdots & & \vdots \\ \mathsf{A}_{m,1} & \mathsf{A}_{m,2} & \cdots & \mathsf{A}_{m,m} \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \vdots \\ \boldsymbol{\sigma}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \end{bmatrix}, \tag{1.9}$$

where each block $\mathsf{A}_{p,q}$ is of size $n \times n$. The diagonal blocks $\mathsf{A}_{p,p}$ are constructed using the technique described in Section 5.3.2. Next observe that in the off-diagonal blocks, the "naive" formula resulted from standard Gaussian quadrature works well since the kernel $G_\kappa(\boldsymbol{x}, \boldsymbol{x}')$ is smooth since $\boldsymbol{x}$ and $\boldsymbol{x}'$ belong to different scatterers. We then solve the linear system (17) using the iterative solver GMRES [88], accelerated by a block-diagonal pre-conditioner. Each matrix-vector multiplication is accelerated using the Fast Multipole Method (FMM) [43, 21]; specifically the implementation [39] by Zydrunas Gimbutas and Leslie Greengard.

Furthermore, when the scatterers are not tightly packed, we explore that the off-diagonal blocks of the linear system are typically rank-deficient. Therefore, the multibody scattering scheme can be accelerated by reducing the size of the linear system (17) before applying an iterative solver.

Chapter 5 describes the implementation details of the high accurate numerical scheme described above. Numerical results illustrate the efficiency and robustness of the method. In particular, implementing this accelerated scheme greatly reduces the degree of freedom especially for Helmholtz equation.

1.3 Theme III: Scattering in variable wave-speed media in 2D and 3D

This section describes a numerical method for solving boundary value problems, such as

$$\begin{cases} -\nabla(c(\boldsymbol{x})\nabla u(\boldsymbol{x})) + b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases} \tag{1.10}$$

where Ω is rectangular domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary $\Gamma = \partial\Omega$, where c and b are smooth functions, where $c > 0$ and $b \geq 0$, and where f is a given function.

1.3.1 Direct solver via composite spectral discretization

We first describe a direct solver which draws an earlier work [79] for solving variable coefficient elliptic PDEs (1.10) in the plane via a composite spectral collocation method. The scheme combines a spectral multidomain technique with a hierarchical direct solver. More specifically, the scheme starts with splitting the domain into a hierarchical tree of patches where on each patch the solution to (1.10) are approximated via tabulation on a tensor product grid of Chebyshev points. A local solution operator for each "leaf" patch is then built, and solution operators for larger patches are built via a hierarchical merge procedure in a upwards sweep over the tree from smaller to larger patches. Once the solution operators for all patches have been computed, the Dirichlet data tabulated on Γ are taken as input to construct tabulated values of the solution vector at all internal grid points. This solve stage involves a single downwards pass through the tree, going from larger patches to smaller.

Note that once the solution operators have been built, solves can be executed extremely rapidly, requiring only $O(N \log N)$ operations while the build stage has $O(N^{1.5})$ asymptotic complexity. Details describing the direct solver are given in Chapter 6.

1.3.2 Local mesh refinement scheme

If there exists a localized region $\tilde{\Omega} \subset \Omega$ where $c(\boldsymbol{x})$ and/or $b(\boldsymbol{x})$ vary dramatically, or the solution $u(\boldsymbol{x})$ exhibits singularities, it is beneficial to use a finer discretization in that region. We propose a refinement scheme that retains the high-order accuracy by continuously splitting the the patches into smaller ones with a distance criteria. Further details on this topic are presented in Chapter 6. Numerical experiments show that the proposed algorithm is capable of retaining high-order accuracy with a few levels of refinements.

1.3.3 Accelerated 3D direct solver

We also develop a fast direct solver for solutions to linear systems arising from three dimensional elliptic equations. The construction of the direct solver for 3D problems is similar to that of the 2D problems, which consists of spectral discretization, build stage and solve stage. The build stage has $O(N^2)$ asymptotic complexity, which is dominated by inverting dense matrices of size $O(N^{2/3}) \times O(N^{2/3})$ at top levels. The solve stage takes $O(N^{4/3})$ operations.

However, an acceleration technique to compute the solution operators efficiently can be implemented by exploring the internal structure of the matrix representation T of the DtN operator. We make the following claims:

- The off-diagonal blocks of T have low numerical rank.
- The diagonal blocks of T are *Hierarchically Semi Separable* (HSS) matrices with low HSS rank.

By HSS, we mean a matrix is amenable to a telescoping block factorization. The computational cost of the build stage could then be reduced to $O(N^{4/3})$ since arithmetic operations involving dense HSS matrices of size $M \times M$ can often be executed in $O(M)$ operations. Details on properties, the factored representation of HSS matrices as well as the associated fast algorithms are provided in [36, 46, 47] and Chapter 7.

1.4 Structure of dissertation and overview of principle contributions

Beside the introductory and conclusion chapters, the dissertation consists of six chapters, four of which have been published to referred journals. Each chapter is independent and can be read by itself. To improve readability and accessibility, in this section we briefly summarize the key contributions of each chapter.

Chapter 2: The material in this chapter has appeared as:

> S. Hao, A. Barnett, P.G. Martinsson, and P. Young, *High-order accurate methods for Nyström discretization of integral equations on smooth curves in the plane*, Advances in Computational Mathematics, 40(1), pp. 245-272, 2014.

This paper can be viewed as a survey which summarizes existing high-order quadrature rules for integrating functions with logarithmic kernels. It also provides details to deriving Nyström implementation formulas and also demonstrates their performance with numerical experiments. Principle contributions of this paper include:

- Nyström implementation formulas of four different quadratures: Kapur-Rokhlin, Alpert, Kolm-Rokhlin and Kress are carefully derived in this paper. Although earlier work [70, 2, 64, 66, 70] present the construction of these high-order quadrature rules, many of them are lack of details of how to implement them in Nyström discretization of BIEs. The formulas we derived are compact and can be easily implemented into numerical simulation.
- The paper presents formal comparison between the accuracy of different approaches on various problem settings. We also make informal remarks on the relative advantages and disadvantages of different quadrature rules. Our remarks are primarily informed by the numerical experiments. These remarks could give some advice which quadratures should be used given certain problems.

Chapter 3: The material in this chapter has appeared as:

A. Gillman, S. Hao, P.G. Martinsson, *A simplied technique for the efficient and high-order accurate discretization of boundary integral equations in 2D on domains with corners*, Journal of Computational Physics, 256(1), pp. 214–219, 2014.

This paper implements one of the techniques described in Chapter 2, composite Gaussian rule, to solve BIEs on two-dimensional domains with corners, but focuses on eliminating the "superfluous" degrees of freedom added by refinement. Principle contributions include:

- This paper describes the recent developed techniques for computing solutions to BIEs defined on piecewise smooth contour in the plane with corners. In our approach, the BIE is discretized with standard Nyström method where composite Gaussian quadrature is used to deal with log-singularity of the kernel functions. Singularity near the corner points can be resolved by refining the computation mesh near the corners until given computational tolerance ϵ reached.

- When dealing with the challenges arise with the refinement process where the number of degrees of freedom dramatically increases, instead of using the purpose-built local process introduced in [10, 55, 59, 53], we accomplish the "squeezing out" task via employing the general purpose direct solver of [36, 42, 63, 78] to compress the corner and eliminate the "extra" degrees of freedom. Moreover, numerical experiments show that this local compression process can be executed in time that scales linearly with the number of degrees of freedom added.

Chapter 4: The material in this chapter has appeared as:

P. Young, S. Hao, and P. G. Martinsson, *A high-order Nyström discretization scheme for boundary integral equations defined on rotationally symmetric surfaces*, J. Comput. Phys. 231 (2012), no. 11, 41424159.

This paper implements high-order panel-based Gaussian quadrature rule to solve scattering problem on rotationally symmetric surfaces in 3D. Principle contributions include:

- The paper describes a high-order Nyström discretization of the BIEs (1) that provides far higher accuracy and speed than previously published methods. The scheme presented uses the Fourier transform to reduce the original BIE defined on a surface to a sequence of BIEs defined on a generating curve for the surface. Nyström discretization with high-order Gaussian quadrature rule is used to discretize the BIEs on the generating curve. The reduction in dimensionality, along with the use of high-order accurate quadratures, leads to small linear systems that can be inverted directly via, *e.g.*, Gaussian elimination. This makes the scheme particularly fast in environments involving multiple right hand sides.
- Complications arise when evaluating the kernel functions of the axisymmetric formulation. The fact is that each kernel is defined as a Fourier integral of the original kernel function in the azimuthal variable, that cannot be evaluated analytically, and would be too expensive to approximate via standard quadrature techniques. Fortunately, for BIEs associated with the Laplace and Helmholtz equations, we develop techniques to evaluate the kernel in the reduced equations very rapidly by exploiting recursion relations for Legendre function.

Chapter 5: The material in this chapter has appeared as:

> S. Hao, P.G. Martinsson, P. Young, *An efficient and high-order accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers*, Computers and Mathematics with Applications, 69(4), 2015, pp 304 - 318.

This paper extends the algorithm described in Chapter 4 to solving problems involving multiple rotational symmetric bodies. Principle contributions include:

- In this paper we combine several recently developed techniques to obtain a solver capable of solving scattering problems on complex multibody geometries in three dimensions. The solver is robust and highly accurate that it solves the problem to eight digits of accuracy or more with moderate numbers of discretization nodes. In particular, the solver is capable of resolving domain involving cavities.

- This paper also presents an accelerated algorithm for problems where the scatterers are well-separated. In these situations, the off-diagonal blocks of the coefficient matrix are rank deficient. The number of degrees of freedom in the global system can be greatly reduced by applying skeletonization scheme introduced in [76]. The algorithm is capable of solving even very large scale problems to high accuracy. Extensive numerical experiments confirm the performance that a dozen of degrees of freedom can be reduced if the accelerated algorithm is implemented.

Chapter 6: This chapter describes a high-order local mesh refinement scheme for variable coefficient elliptic PDEs in the plane, where the regularity of the solution changes across the domain. The algorithm relies on the composite spectral scheme [79] on uniform grid. Specific contributions include:

- The variable coefficient elliptic PDEs in the plane are discretized via a composite spectral collocation method. While Chebyshev nodes are ideal for the leaf computations, however, there are benefits to using Gaussian nodes that justify the trouble to retabulate the operators. One of the advantages is to allow for interpolation between different discretizations and make the direct solver be easily extended to local refinement when necessary. An interpolation technique is carefully presented in this chapter which interpolates the potential values at Chebyshev nodes on the boundary of each patch to its values at Gaussian nodes.

- This chapter presents a refinement scheme when the positions need for refinement are known in advance. The boxes containing these points are split into smaller ones according to a distance criteria. A direct solver is then executed after the hierarchical tree is built which solves the problem in a single sweep. After all the solution operators are constructed for every box in the hierarchical tree, solutions can be obtained immediately for multiple righthand-sides without recomputing the solution operators.

Chapter 7: This chapter describes a numerical scheme to solve elliptic PDEs with variable coefficients on three-dimensional domains. The method is an extension to the direct solver presented in Chapter 6.

- The principal contribution of the present work is a technique that exploits internal structure in the dense matrices representing DtN maps to improve on the computational complexity from $O(N^2)$ to $O(N^{4/3})$. The fact is that the diagonal blocks of the DtN operators are of HSS (Hierarchically Semi Separable) form which enable matrix inversion to be performed in linear time.

- Numerical experiments presented in this chapter were carried on a personal workstation. The algorithm performs well for solving three-dimensional Laplace and low-frequency Helmholtz problems. However, with these experiments, we observe that memory constraints become far more limiting for 3D problems than for problems in 2D.

Chapter 2

High-order accurate methods for Nyström discretization of integral equations on smooth curves in the plane

S. Hao, A. H. Barnett, P.G. Martinsson, P. Young

Note: *The work described in this chapter was carried out in collaboration with Professors Alexander Barnett of Dartmouth and Per-Gunnar Martinsson of the University of Colorado and Patrick Young. It appeared in the Advances in Computational Mathematics (40(1), pages 245-272, 2014) under the title: "High-order accurate methods for Nyström discretization of integral equations on smooth curves in the plane " .*

Abstract: Boundary integral equations and Nyström discretization provide a powerful tool for the solution of Laplace and Helmholtz boundary value problems. However, often a weakly-singular kernel arises, in which case specialized quadratures that modify the matrix entries near the diagonal are needed to reach a high accuracy. We describe the construction of four different quadratures which handle logarithmically-singular kernels. Only smooth boundaries are considered, but some of the techniques extend straightforwardly to the case of corners. Three are modifications of the global periodic trapezoidal rule, due to Kapur–Rokhlin, to Alpert, and to Kress. The fourth is a modification to a quadrature based on Gauss–Legendre panels due to Kolm–Rokhlin; this formulation allows adaptivity. We compare in numerical experiments the convergence of the four schemes in various settings, including low- and high-frequency planar Helmholtz problems, and 3D axisymmetric Laplace problems. We also find striking differences in performance in an iterative setting. We summarize the relative advantages of the schemes.

2.1 Introduction

Linear elliptic boundary value problems (BVPs) where the partial differential equation has constant or piecewise-constant coefficients arise frequently in engineering, mathematics, and physics. For the Laplace equation, applications include electrostatics, heat and fluid flow, and probability; for the Helmholtz equation they include the scattering of waves in acoustics, electromagnetics, optics, and quantum mechanics. Because the fundamental solution (free-space Green's function) is known, one may solve such problems using boundary integral equations (BIEs). In this approach, a BVP in two dimensions (2D) is converted via so-called jump relations to an integral equation for an unknown function living on a 1D curve [27]. The resulting reduced dimensionality and geometric simplicity allows for high-order accurate numerical solutions with much more efficiency than standard finite-difference or finite element discretizations [4].

The BIEs that arise in this setting often take the second-kind form

$$\sigma(x) + \int_0^T k(x,x')\sigma(x')\,dx' = f(x), \qquad x \in [0,T], \tag{2.1}$$

where $[0,T]$ is an interval, where f is a given smooth T-periodic function, and where k is a (doubly) T-periodic kernel function that is smooth away from the diagonal and has a logarithmic singularity as $x' \to x$. (The term "smooth" refers to C^∞ in this manuscript.) In order to solve a BIE such as (2.1) numerically, it must be turned into a linear system with a finite number N unknowns. This is most easily done via the Nyström method [82, 71]. (There do exist other discretization methods such as Galerkin and collocation [71]; while their relative merits in higher dimensional settings are still debated, for curves in the plane there seems to be little to compete with Nyström [27, Sec. 3.5].) However, since the fundamental solution in 2D has a logarithmic singularity, generic integral operators of interest inherit this singularity at the diagonal, giving them (at most weakly-singular) kernels which we write in the standard "periodized-log" form

$$k(x,x') = \varphi(x,x')\log\left(4\sin^2\frac{\pi(x-x')}{T}\right) + \psi(x,x') \tag{2.2}$$

for some smooth, doubly T-periodic functions φ and ψ. In this note we focus on a variety of

	split into φ, ψ explicit	split into φ, ψ unknown
global (periodic trapezoidal rule)	• Kress[†] [70]	• Kapur–Rokhlin [64] • Alpert [2] ○ QBX* [65]
panel-based (Gauss-Legendre nodes)	○ Helsing [52, 56]	• Modified Gaussian (Kolm-Rokhlin) [66] ○ QBX* [65]

Table 2.1: Classification of Nyström quadrature schemes for logarithmically-singular kernels on smooth 1D curves. Schemes tested in this work are marked by a solid bullet ("•"). Schemes are amenable to the FMM unless indicated with a †. Finally, $*$ indicates that other analytic knowledge is required, namely a local expansion for the PDE.

high-order quadrature schemes for the Nyström solution of such 1D integral equations.

We first review the Nyström method, and classify some quadrature schemes for weakly-singular kernels.

2.1.1 Overview of Nyström discretization

One starts with an **underlying** quadrature scheme on $[0, T]$, defined by nodes $\{x_i\}_{i=1}^N$ ordered by $0 \leq x_1 < x_2 < x_3 < \cdots < x_N < T$, and corresponding weights $\{w_i\}_{i=1}^N$. This means that for g a smooth T-periodic function,

$$\int_0^T g(x)dx \;\approx\; \sum_{i=1}^N w_i g(x_i)$$

holds to high accuracy. More specifically, the error converges to zero to high order in N. Such quadratures fall into two popular types: either a **global** rule on $[0, T]$, such as the periodic trapezoidal rule [71, Sec. 12.1] (which has equally-spaced nodes and equal weights), or a **panel-based** (composite) rule which is the union of simple quadrature rules on disjoint intervals (or **panels**) which cover $[0, T]$. An example of the latter is composite Gauss–Legendre quadrature. The two types are shown in Figure 2.1 (a) and (b). Global rules may be excellent—for instance, if g is analytic in a neighborhood of the real axis, the periodic trapezoidal rule has exponential convergence [71, Thm. 12.6]—yet panel-based rules can be more useful in practice because they are very simple to make adaptive: one may split a panel into two smaller panels until a local convergence criterion is met.

The Nyström method for discretizing (2.1) constructs a linear system that relates a given data vector $\mathsf{f} = \{f_i\}_{i=1}^N$ where $f_i = f(x_i)$ to an unknown solution vector $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^N$ where $\sigma_i \approx \sigma(x_i)$. Informally speaking, the idea is to use the nodes $\{x_i\}_{i=1}^N$ as **collocation points** where (2.1) is enforced:

$$\sigma(x_i) + \int_0^T k(x_i, x')\sigma(x')\,dx' = f(x_i), \qquad i = 1, \ldots, N. \tag{2.3}$$

Then matrix elements $\{a_{i,j}\}_{i,j=1}^N$ are constructed such that, for smooth T-periodic σ,

$$\int_0^T k(x_i, x')\sigma(x')\,dx' \approx \sum_{j=1}^N a_{i,j}\,\sigma(x_j)\ . \tag{2.4}$$

Combining (2.3) and (2.4) we obtain a square linear system that relates $\boldsymbol{\sigma}$ to f:

$$\sigma_i + \sum_{j=1}^N a_{i,j}\,\sigma_j = f_i, \qquad i = 1, \ldots, N\ . \tag{2.5}$$

In a sum such as (2.5), it is convenient to think of x_j as the **source** node, and x_i as the **target** node. We write (2.5) in matrix form as

$$\boldsymbol{\sigma} + \mathsf{A}\boldsymbol{\sigma} = \mathsf{f}\ . \tag{2.6}$$

A high-order approximation to $\sigma(x)$ for general $x \in [0, T]$ may then be constructed by interpolation through the values $\boldsymbol{\sigma}$; the preferred way to do this is via the kernel itself, which is known as "Nyström interpolation" [71, p. 202].

If the kernel k is smooth, as occurs for the Laplace double-layer operator, then the matrix elements

$$a_{i,j} = k(x_i, x_j)w_j \tag{2.7}$$

lead to an error convergence rate that is provably the same order as the underlying quadrature scheme [71, Sec. 12.2]. It is less obvious how to construct the matrix $\mathsf{A} = \{a_{i,j}\}$ such that (2.4) holds to high order accuracy in the case where k has a logarithmic singularity, as in (2.2). The purpose of this note is to describe and compare several techniques for this latter task. Note that it is the assumption that the solution $\sigma(x)$ is smooth (i.e. well approximated by high-order interpolation schemes)

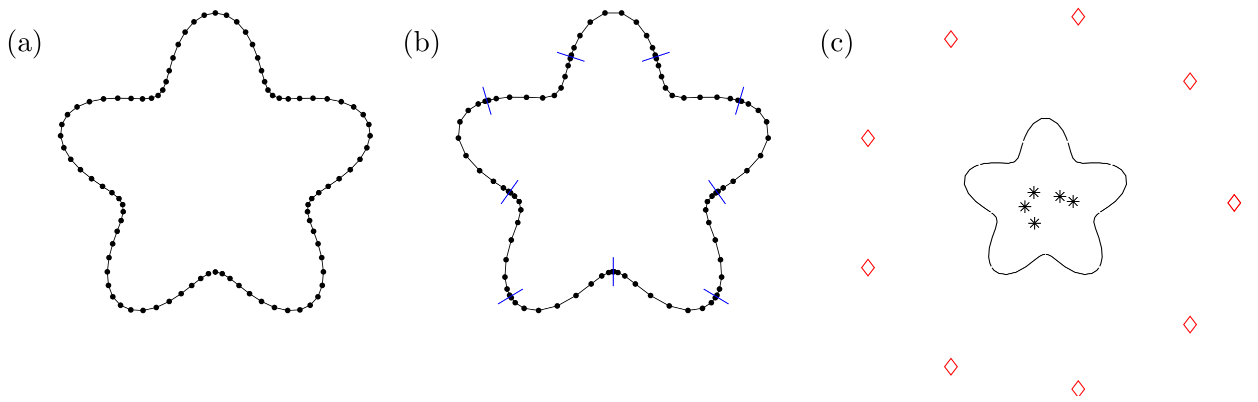


Figure 2.1: Example smooth planar curve discretized with $N = 90$ points via (a) periodic trapezoidal rule nodes and (b) panel-based rule (10-point Gauss–Legendre; the panel ends are shown by line segments). In both cases the parametrization is polar angle $t \in [0, 2\pi]$ and the curve is the radial function $f(t) = 9/20 - (1/9)\cos(5t)$. (c) Geometry for 2D Helmholtz numerical examples in section 2.7.2 and 2.7.3. The curve is as in (a) and (b). Stars show source locations that generate the potential, while diamonds show testing locations.

2.1.2 Types of singular quadrature schemes

We now overview the schemes presented in this work. It is desirable for a scheme for the singular kernel case to have **almost all** elements be given by (2.7), for the following reason. When N is large (greater than 10^4, say), solving (2.6) via standard dense linear algebra starts to become impractical, since $O(N^3)$ effort is needed. Iterative methods are preferred which converge to a solution using a small number of matrix-vector products; in the last couple of decades so-called **fast algorithms** have arisen to perform such a matrix-vector product involving a dense $N \times N$ matrix in only $O(N)$ or $O(N \log N)$ time. The most well-known is probably the fast multipole method (FMM) of Rokhlin–Greengard [43], but others exist [6, 28, 102]. They use potential theory to evaluate all N sums of the form

$$\sum_{j=1}^{N} k(x_i, x_j)\, q_j, \qquad i = 1, \ldots, N \tag{2.8}$$

where the q_j are interpreted as charge strengths. Choosing $q_j = w_j \sigma_j$ turns this into a fast algorithm to evaluate $\mathsf{A}\boldsymbol{\sigma}$ given $\boldsymbol{\sigma}$, in the case where A is a standard Nyström matrix (2.7).

Definition 1 We say that a quadrature scheme is *FMM-compatible* provided that only $O(N)$ elements $\{a_{i,j}\}$ differ from the standard formula (2.7).

An FMM-compatible scheme can easily be combined with any fast summation scheme for the sum (2.8) without compromising its asymptotic speed. Usually, for FMM-compatible schemes, the elements which differ from (2.7) will lie in a band about the diagonal; the width of the band depends only on the order of the scheme (not on N). All the schemes we discuss are FMM-compatible, apart from that of Kress (which is not to say that Kress quadrature is incompatible with fast summation; merely that a standard FMM will not work out of the box).

Another important distinction is the one between (a) schemes in which the analytic split (2.2) into two smooth kernels must be **explicitly known** (i.e. the functions φ and ψ are independently evaluated), and (b) schemes which merely need to access the overall kernel function k. The latter schemes are more flexible, since in applications the split is not always readily available. However, as we will see, this flexibility comes with a penalty in terms of accuracy.

The following schemes will be described:

- **Kapur–Rokhlin** (section 2.3). This is the simplest scheme to implement, based upon an underlying periodic trapezoidal rule. The weights, but not the node locations, are modified near the diagonal. No explicit split is needed.
- **Alpert** (section 2.4). Also based upon the periodic trapezoidal rule, and also not needing an explicit split, this scheme replaces the equi-spaced nodes near the diagonal with an optimal set of auxiliary nodes, at which new kernel evaluations are needed.
- **Modified Gaussian** (section 2.5). Here the underlying quadrature is Gauss–Legendre panels, and new kernel evaluations are needed at each set of auxiliary nodes chosen for each target node in the panel. These auxiliary nodes are chosen using the algorithm of Kolm–Rokhlin [66]. No explicit split is needed.
- **Kress** (section 2.6). This scheme uses an explicit split to create a spectrally-accurate product quadrature based upon the periodic trapezoidal rule nodes. All of the matrix elements differ from the standard form (2.7), thus the scheme is not FMM-compatible. We include it as a benchmark where possible.

Table 2.1 classifies these schemes (and a couple of others), according to whether they have underlying global (periodic trapezoidal rule) or panel-based quadrature, whether the split into the two smooth functions need be explicitly known or not, and whether they are FMM-compatible.

In section 5.6 we present numerical tests comparing the accuracy of these quadratures in 1D, 2D, and 3D axisymmetric settings. We also demonstrate that some schemes have negative effects on the convergence rate in an iterative setting. We compare the advantages of the schemes and draw some conclusions in section 2.8.

2.1.3 Related work and schemes not compared

The methods described in this paper rely on earlier work [70, 2, 64, 66] describing high-order quadrature rules for integrands with weakly singular kernels. It appears likely that these rules were designed in part to facilitate Nyström discretization of BIEs, but, with the exception of Kress [70], the original papers leave most details out. (Kress describes the Nyström implementation but does not motivate the quadrature formula; hence we derive this in section 2.6.) Some later papers reference the use (e.g. [11, 76]) of high order quadratures but provide few details. In particular, there appears to have been no formal comparison between the accuracy of different approaches.

There are several schemes that we do not have the space to include in our comparison. One of the most promising is the recent scheme of Helsing for Laplace [52] and Helmholtz [56] problems, which is panel-based but uses an explicit split in the style of Kress, and thus needs no extra kernel evaluations. We also note the recent QBX scheme [65] (quadrature by expansion) which makes use of off-curve evaluations and local expansions of the PDE.

2.2 A brief review of Lagrange interpolation

This section reviews some well-known (see, e.g., [4, Sec 3.1]) facts about polynomial interpolation that will be used repeatedly in the text.

For a given set of distinct nodes $\{x_j\}_{j=1}^N$ and function values $\{y_j\}_{j=1}^N$, the Lagrange interpolation polynomial $L(x)$ is the unique polynomial of degree no greater than $N-1$ that passes through the N points $\{(x_j,\, y_j)\}_{j=1}^N$. It is given by

$$L(x) = \sum_{j=1}^{N} y_j\, L_j(x),$$

where

$$L_j(x) = \prod_{\substack{i=1\\ i\neq j}}^{N} \left(\frac{x - x_i}{x_j - x_i}\right). \tag{2.9}$$

While polynomial interpolation can in general be highly unstable, it is perfectly safe and accurate as long as the interpolation nodes are chosen well. For instance, for the case where the nodes

$\{x_j\}_{j=1}^N$ are the nodes associated with standard Gaussian quadrature on an interval $I = [0, b]$, it is known [4, Thm. 3.2] that for any $f \in C^N(I)$

$$\left| f(s) - \sum_{j=1}^{N} L_j(s)\, f(x_j) \right| \le C\, b^N \qquad s \in I,$$

where

$$C = \left(\sup_{s \in [0,\, b]} |f^{(N)}(s)| \right) /N!$$

2.3 Nyström discretization using the Kapur-Rokhlin quadrature rule

2.3.1 The Kapur–Rokhlin correction to the trapezoidal rule

Recall that the standard N+1-point trapezoidal rule that approximates the integral of a function $g \in C^\infty[0, T]$ is $h[g(0)/2 + g(h) + \cdots + g(T-h) + g(T)/2]$, where the node spacing is

$$h = \frac{T}{N} \ ,$$

and that it has only 2nd-order accuracy [71, Thm. 12.1]. The idea of Kapur–Rokhlin [64] is to modify this rule to make it high-order accurate, by changing a small number of weights near the interval ends, and by adding some extra equi-spaced evaluation nodes $x_j = hj$ for $-k \le j < 0$ and $N < j \le N + k$, for some small integer $k > 0$, i.e. nodes lying just **beyond** the ends. They achieve this goal for functions $g \in C^\infty[0, T]$, but also for the case where one or more endpoint behavior is singular, as in

$$g(x) = \varphi(x)s(x) + \psi(x) \ , \tag{2.10}$$

where $\varphi(x), \psi(x) \in C^\infty[0, T]$ and $s(x) \in C(0, T)$ is a known function with integrable singularity at zero, or at T, or at both zero and T. Accessing extra nodes outside of the interval suppresses the rapid growth with order of the weight magnitudes that plagued previous corrected trapezoidal rules. However, it is then maybe unsurprising that their results need the additional assumption that $\varphi, \psi \in C^\infty[-hk, T + hk]$.

Since we are interested in methods for kernels of the form (2.2), we specialize to a periodic integrand with the logarithmic singularity at $x = 0$ (and therefore also at $x = T$, making both endpoints singular),

$$g(x) = \varphi(x) \log \left|\sin \frac{x\pi}{T}\right| + \psi(x) \ . \tag{2.11}$$

The mth-order Kapur–Rokhlin rule T_m^{N+1} which corrects for a log singularity at both left and right endpoints is,

$$T_m^{N+1}(g) = h\Bigg[\sum_{\substack{\ell=-m\\ \ell\neq 0}}^{m} \gamma_\ell\, g(\ell h) + g(h) + g(2h) + \cdots + g(T-h) + \sum_{\substack{\ell=-m\\ \ell\neq 0}}^{m} \gamma_{-\ell}\, g(T+\ell h)\Bigg] \tag{2.12}$$

Notice that the left endpoint correction weights $\{\gamma_\ell\}_{\ell=-m,\ell\neq 0}^{m}$ are also used (in reverse order) at the right endpoint. The convergence theorems from [64] then imply that, for any fixed T-periodic $\varphi, \psi \in C^\infty(\mathbb{R})$,

$$\left|\int_0^T g(x)\, dx - T_m^{N+1}(g)\right| = O(h^m) \qquad \text{as } N \to \infty \ . \tag{2.13}$$

The apparent drawback that function values are needed at nodes ℓh for $-m \le \ell \le -1$, which lie outside the integration interval, is not a problem in our case of periodic functions, since by periodicity these function values are known. This enables us to rewrite (2.12) as

$$T_m^{N+1}(g) = h\Bigg[\sum_{\ell=1}^{m}(\gamma_\ell+\gamma_{-\ell})g(\ell h)+g(h)+g(2h)+\cdots+g(T-h)+\sum_{\ell=-m}^{-1}(\gamma_\ell+\gamma_{-\ell})g(T+\ell h)\Bigg] \tag{2.14}$$

which involves only the $N-1$ nodes interior to $[0,T]$.

For orders $m = 2, 6, 10$ the values of γ_ℓ are given in the left-hand column of [64, Table 6]. (These are constructed by solving certain linear systems, see [64, Sec. 4.5.1].) In our periodic case, only the values $\gamma_\ell + \gamma_{-\ell}$ are needed; for convenience we give them in appendix A. Notice that they are of magnitude 2 for $m = 2$, of magnitude around 20 for $m = 6$, and of magnitude around 400 for $m = 10$, and alternate in sign in each case. This highlights the statement of Kapur–Rokhlin that they were only partially able to suppress the growth of weights in the case of singular endpoints [64].

2.3.2 A Nyström scheme

We now can construct numbers $a_{i,j}$ such that (2.4) holds. We start with the underlying periodic trapezoidal rule, with equi-spaced nodes $\{x_i\}_{i=1}^{N}$ with $x_j = hj$, $h = T/N$. We introduce a discrete offset function $\ell(i,j)$ between two nodes x_i and x_j defined by

$$\ell(i,j) \equiv j - i \pmod N, \qquad -N/2 < \ell(i,j) \leq N/2 \ ,$$

and note that for each $i \in \{1,\ldots,N\}$, and each $|\ell| < N/2$, there is a unique $j \in \{1,\ldots,N\}$ such that $\ell(i,j) = \ell$. Two nodes x_i and x_j are said to be "close" if $|\ell(i,j)| \leq m$. Moreover, we call x_i and x_j "well-separated" if they are not close.

We now apply (2.14) to the integral (2.4), which by periodicity of σ and the kernel, we may rewrite with limits x_i to $x_i + T$ so that the log singularity appears at the endpoints. We also extend the definition of the nodes $x_j = hj$ for all integers j, and get,

$$\begin{aligned}\int_0^T k(x_i,x')\sigma(x')\,dx' &= \int_{x_i}^{x_i+T} k(x_i,x')\sigma(x')\,dx' \\ &\approx h \sum_{j=i+1}^{i+N-1} k(x_i,x_j)\sigma(x_j) + h \sum_{\substack{\ell=-m \\ \ell\neq 0}}^{m} (\gamma_\ell + \gamma_{-\ell}) k(x_i,x_{i+\ell})\,\sigma(x_{i+\ell}) \ .\end{aligned}$$

Wrapping indices back into $\{1,\ldots,N\}$, the entries of the coefficient matrix A are seen to be,

$$a_{i,j} = \begin{cases} 0 & \text{if } i = j, \\ h\,k(x_i,x_j) & \text{if } x_i \text{ and } x_j \text{ are "well-separated"}, \\ h\,(1 + \gamma_{\ell(i,j)} + \gamma_{-\ell(i,j)})\,k(x_i,x_j) & \text{if } x_i \text{ and } x_j \text{ are "close", and } i \neq j. \end{cases} \tag{2.15}$$

Notice that this is the elementwise product of a circulant matrix with the kernel matrix $k(x_i,x_j)$, and that diagonal values are ignored. Only $O(N)$ elements (those closest to the diagonal) differ from the standard Nyström formula (2.7).

2.4 Nyström discretization using the Alpert quadrature rule

2.4.1 The Alpert correction to the trapezoidal rule

Alpert quadrature is another correction to the trapezoidal rule that is high-order accurate for integrands of the form (2.10) on $(0,T)$. The main difference with Kapur–Rokhlin is that Alpert quadrature uses node locations **off** the equi-spaced grid $x_j = hj$, but within the interval $(0,T)$.

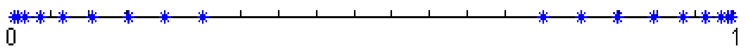


Figure 2.2: Example of Alpert quadrature scheme of order $l = 10$ on the interval $[0, 1]$. The original trapezoidal rule had 20 points including both endpoints, i.e. $N = 19$ and $h = 1/19$. Correction nodes $\{\chi_p h\}_{p=1}^m$ and $\{1 - \chi_p h\}_{p=1}^m$ for $m = 10$ and $a = 6$, are denoted by stars.

Specifically, for the function g in (2.11) with log singularities at both ends, we denote by $S_l^{N+1}(g)$ the lth-order Alpert quadrature rule based on an $N+1$-point trapezoidal grid, defined by the formula

$$S_l^{N+1}(g) = h \sum_{p=1}^{m} w_p \, g(\chi_p \, h) + h \sum_{j=a}^{N-a} g(jh) + h \sum_{p=1}^{m} w_p \, g(T - \chi_p \, h). \tag{2.16}$$

There are $N - 2a + 1$ internal equi-spaced nodes with spacing $h = T/N$ and equal weights; these are common to the trapezoidal rule. There are also m new "correction" nodes at each end which replace the a original nodes at each end in the trapezoidal rule. The label "lth-order" is actually slightly too strong: the scheme is proven [2, Cor. 3.8] to have error convergence of order $O(h^l |\log h|)$ as $h \to 0$. The number m of new nodes needed per end is either $l-1$ or l. For each order l, the integer a is chosen by experiments to be the smallest integer leading to positive correction nodes and weights. The following table shows the values of m and a for log-singular kernels for convergence orders $l = 2$, 6, 10, 16:

Convergence order	$h^2 \|\log h\|$	$h^6 \|\log h\|$	$h^{10} \|\log h\|$	$h^{16} \|\log h\|$
Number of correction points m	1	5	10	15
Width of correction window a	1	3	6	10

The corresponding node locations $\chi_1, \ldots \chi_m$ and weights $w_1, \ldots w_m$ are listed in Appendix B; and illustrated for the case $l = 10$ in Figure 2.2. The Alpert quadrature nodes and weights are determined by solving certain non-linear systems of equations, see e.g. [2, Eqns. (39) & (40)]. Numerically, these equations can be solved via schemes based on modified Newton iterations, see [2, Sec. 5]. (The quadratures listed in Appendix B are adapted from [2, Table 8].)

2.4.2 A Nyström scheme

Recall that we wish to construct a matrix $a_{i,j}$ that when applied to the vector of values $\{\sigma(x_j)\}_{j=1}^N$ approximates the action of the integral operator on the function σ, evaluated at each target node x_i. To do this via the lth-order Alpert scheme with parameter a, we start by using periodicity to shift the domain of the integral in (2.4) to $(x_i, x_i + T)$, as in section (2.3.2). Since the $2m$ auxiliary Alpert nodes lie symmetrically about the singularity location x_i, for simplicity we will treat them as a single set by defining $\chi_{p+m} = -\chi_p$ and $w_{p+m} = w_p$ for $p = 1, \dots, m$. Then the rule (2.16) gives

$$\int_0^T k(x_i, x')\sigma(x')\,dx' \;\approx\; h\sum_{p=a}^{N-a} k(x_i, x_i{+}ph)\,\sigma(x_i{+}ph){+}h\sum_{p=1}^{2m} w_p k(x_i, x_i{+}\chi_p h)\,\sigma(x_i{+}\chi_p h)\;. \quad (2.17)$$

The values $\{\chi_p\}_{p=1}^{2m}$ are not integers, so no auxiliary nodes coincide with any equi-spaced nodes $\{x_j\}_{j=1}^N$ at which the vector of σ values is given. Hence we must interpolate σ to the auxiliary nodes $\{x_i + \chi_p h\}_{p=1}^{2m}$. We do this using local Lagrange interpolation through M equi-spaced nodes surrounding the auxiliary source point $x_i + \chi_p h$. For $M > l$ the interpolation error is higher order than that of the Alpert scheme; we actually use $M = l + 3$ since the error is then negligible.

Remark 1 While high-order Lagrange interpolation through equi-spaced points is generally a bad idea due to the Runge phenomenon [96], here we will be careful to ensure that the evaluation point always lies nearly at the center of the interpolation grid, and there is no stability problem.

For each auxiliary node p, let the nth interpolation node index offset relative to i be

$$o_n^{(p)} := \lfloor \chi_p - M/2 \rfloor + n\ ,$$

and let the whole set be $O^{(p)} := \{o_n^{(p)}\}_{n=1}^M$. For $q \in O^{(p)}$, let the function $n_p(q) := q - \lfloor \chi_p - M/2 \rfloor$ return the node number of an index offset of q. Finally, let

$$L_n^{(p)}(x) = \prod_{\substack{k=1\\k\neq n}}^{M} \left(\frac{x - o_k^{(p)}}{o_n^{(p)} - o_k^{(p)}} \right)$$

be the nth Lagrange basis polynomial for auxiliary node p. Applying this interpolation in σ gives for the second sum in (2.17),

$$h \sum_{p=1}^{2m} w_p k(x_i, x_i + \chi_p h) \sum_{n=1}^{M} L_n^{(p)}(\chi_p) \sigma(x_{i+o_n^{(p)}}) \ .$$

Note that all node indices in the above will be cyclically folded back into the set $\{1, \ldots, N\}$.

Recalling the notation $\ell(i,j)$ from section 2.3.2, we now find the coefficient matrix A has entries

$$a_{i,j} = b_{i,j} + c_{i,j}, \tag{2.18}$$

where the first sum in (2.17) gives

$$b_{i,j} = \begin{cases} 0 & \text{if } |\ell(i,j)| < a, \\ h\, k(x_i, x_j) & \text{if } |\ell(i,j)| \geq a, \end{cases} \tag{2.19}$$

the standard Nyström matrix (2.7) with a diagonal band set to zero, and the auxiliary nodes give

$$c_{i,j} \ = \ h \sum_{\substack{p=1 \\ O^{(p)} \ni \ell(i,j)}}^{2m} w_p\, k(x_i, x_i + \chi_p h)\, L^{(p)}_{n_p(\ell(i,j))}(\chi_p) \ . \tag{2.20}$$

Notice that the bandwidth of matrix $c_{i,j}$ does not exceed $a + M/2$, which is roughly l, and thus only $O(N)$ elements of $a_{i,j}$ differ from those of the standard Nyström matrix.

2.5 Nyström discretization using modified Gaussian quadrature

We now turn to a scheme with panel-based underlying quadrature, namely the composite Gauss–Legendre rule. We recall that for any interval $I = [0, b]$, the single-panel n-point Gauss–Legendre rule has nodes $\{x_j\}_{j=1}^n \subset I$ and weights $\{w_j\}_{j=1}^n \subset (0, \infty)$ such that the identity

$$\int_0^b f(x)\, dx = \sum_{j=1}^{n} w_j f(x_j) \tag{2.21}$$

holds for every polynomial f of degree at most $2n - 1$. For analytic f the rule is exponentially convergent in n with rate given by the largest ellipse with foci 0 and b in which f is analytic [97, Ch. 19].

2.5.1 Modified Gaussian quadratures of Kolm–Rokhlin

Suppose that given an interval $[0,b]$ and a target point $t \in [0,b]$, we seek to approximate integrals of the form

$$\int_0^b \big(\varphi(s)\, S(t,s) + \psi(s)\big)\, ds, \tag{2.22}$$

where φ and ψ are smooth functions over $[0,b]$ and $S(t,s)$ has a known singularity as $s \to t$. Since the integrand is non-smooth, standard Gauss–Legendre quadrature would be inaccurate if applied to evaluate (2.22). Instead, we seek an m-node "modified Gaussian" quadrature rule with weights $\{v_k\}_{k=1}^m$ and nodes $\{y_k\}_{k=1}^m \subset [0,b]$ that evaluates the integral (2.22) to high accuracy. In particular, we use a quadrature formula of the form

$$\int_0^b \big(\varphi(s)\, S(t,s) + \psi(s)\big)\, ds \approx \sum_{k=1}^m v_k \big(\varphi(y_k)\, S(t,y_k) + \psi(y_k)\big) \tag{2.23}$$

which holds when φ and ψ are polynomials of degree n. It is crucial to note that $\{v_k\}_{k=1}^m$ and $\{y_k\}_{k=1}^m$ depend on the target location t; unless t values are very close then new sets are needed for **each** different t.

Next consider the problem of evaluating (2.22) in the case where the target point t is near the region of integration but not actually inside it, for instance $t \in [-b,0) \cup (b,\, 2b]$. In this case, the integrand is smooth but has a nearby singularity: this reduces the convergence rate and means that its practical accuracy would be low for the fixed n value we prefer. In this case, we can approximate the integral (2.22) with another set of modified Gaussian quadrature weights $\{\hat{v}_k\}_{k=1}^{m'}$ and nodes $\{\hat{y}_k\}_{k=1}^{m'} \subset [0,b]$ giving a quadrature formula analogous to (2.23). In fact, such weights and nodes can be found that hold to high accuracy for all targets t in intervals of the form $[-10^{-p+1}, -10^{-p}]$.

The nodes and weights of the Kolm-Rokhlin quadrature rules are constructed by solving certain non-linear systems of equations via modified Newton iterations, see [66] and [24]. In the numerical experiments in section 5.6, we use a rule with $n = 10$, $m = 20$, and $m' = 24$. This rule leads to fairly high accuracy, but is not of so high order that clustering of the end points becomes an issue in double precision arithmetic. Since the full tables of quadrature weights get long in this case, we provide them as text files at [49].

2.5.2 A Nyström scheme

We partition the domain of integration as

$$[0, T] = \bigcup_{p=1}^{N_P} \Omega_p,$$

where the Ω_p's are non-overlapping subintervals called panels. For simplicity, we for now assume that the panels are equi-sized so that $\Omega_p = \left[\frac{T(p-1)}{N_P}, \frac{Tp}{N_P}\right]$. Note that an adaptive version would have variable-sized panels. On each panel, we place the nodes of an n-point Gaussian quadrature to obtain a total of $N = n\,N_P$ nodes. Let $\{x_i\}_{i=1}^N$ and $\{w_i\}_{i=1}^N$ denote the nodes and weights of the resulting composite Gaussian rule.

Now consider the task of approximating the integral (2.4) at a target point x_i. We decompose the integral as

$$\int_0^T k(x_i, x')\,\sigma(x')\,dx' = \sum_{q=1}^{N_P} \int_{\Omega_q} k(x_i, x')\,\sigma(x')\,dx'. \tag{2.24}$$

We will construct an approximation for each panel-wise integral

$$\int_{\Omega_q} k(x_i, x')\sigma(x')\,dx', \tag{2.25}$$

expressed in terms of the values of σ at the Gaussian nodes in the source panel Ω_q. There are three distinct cases:

Case 1: x_i belongs to the source panel Ω_q: The integrand is now singular in the domain of integration, but we can exploit that σ is still smooth and can be approximated via polynomial interpolation on this single panel. Using the set of n interpolation nodes $\{x_k : x_k \in \Omega_q\}$, let L_j be the Lagrange basis function corresponding to node j. Then,

$$\sigma(x') \;\approx \sum_{j\,:\,x_j \in \Omega_q} L_j(x')\,\sigma(x_j)\ . \tag{2.26}$$

Inserting (2.26) into the integral in (2.25) we find

$$\int_{\Omega_q} k(x_i, x')\sigma(x')\,dx' \;\approx \sum_{j\,:\,x_j \in \Omega_q} \left(\int_{\Omega_q} k(x_i, x') L_j(x')\,dx' \right) \sigma(x_j).$$

Let $\{v_{i,k}\}_{k=1}^m$ and $\{y_{i,k}\}_{k=1}^m$ be the modified Gaussian weights and nodes in the rule (2.23) on the interval Ω_q associated with the target $t = x_i$. Using this rule,

$$\int_{\Omega_q} k(x_i, x')\sigma(x')\, dx' \;\approx \sum_{j\,:\,x_j \in \Omega_q} \left(\sum_{k=1}^{m} v_{i,k}\, k(x_i, y_{i,k})\, L_j(y_{i,k}) \right) \sigma(x_j). \tag{2.27}$$

Note that the expression in brackets gives the matrix element $a_{i,j}$. Here the auxiliary nodes $y_{i,k}$ play a similar role to the auxiliary Alpert nodes $x_i + \chi_p h$ from section 2.4: new kernel evaluations are needed at each of these nodes.

Case 2: x_i belongs to a panel Ω_p adjacent to source panel Ω_q: In this case, the kernel k is smooth, but has a singularity closer to Ω_q than the size of one panel, so standard Gaussian quadrature would still be inaccurate. We therefore proceed as in Case 1: we replace σ by its polynomial interpolant and then integrate using the modified quadratures described in Section 2.5.1. The end result is a formula similar to (2.27) but with the sum including m' rather than m terms, and with $\hat{v}_{i,k}$ and $\hat{y}_{i,k}$ replacing $v_{i,k}$ and $y_{i,k}$, respectively.

Case 3: x_i is well-separated from the source panel Ω_q: By "well-separated" we mean that x_i and Ω_q are at least one panel size apart in the parameter x. (Note that if the curve geometry involves close-to-touching parts, then this might not be a sufficient criterion for being well-separated in $\mathbb{R}^2$; in practice this would best be handled by adaptivity.) In this case, both the kernel k and the potential σ are smooth, so the original Gaussian rule will be accurate,

$$\int_{\Omega_q} k(x_i, x')\sigma(x')\, dx' \approx \sum_{j\,:\,x_j \in \Omega_q} w_j k(x_i, x_j)\sigma(x_j)\ . \tag{2.28}$$

Combining (2.28) and (2.27), we find that the Nyström matrix elements $a_{i,j}$ are given by

$$a_{i,j} = \begin{cases} \sum_{k=1}^{m} v_{i,k}\, k(x_i, y_{i,k})\, L_j(y_{i,k}), & \text{if } x_i \text{ and } x_j \text{ are in the same panel,} \\ \sum_{k=1}^{m'} \hat{v}_{i,k}\, k(x_i, \hat{y}_{i,k})\, L_j(\hat{y}_{i,k}), & \text{if } x_i \text{ and } x_j \text{ are in adjacent panels,} \\ k(x_i, x_j)\, w_j, & \text{if } x_i \text{ and } x_j \text{ are in well-separated panels.} \end{cases}$$

2.6 Nyström discretization using the Kress quadrature rule

The final scheme that we present returns to an underlying periodic trapezoidal rule, but demands separate knowledge of the smooth kernel functions φ and ψ appearing in (2.2). We first review spectrally-accurate product quadratures, which is an old idea, but which we do not find well explained in the standard literature.

2.6.1 Product quadratures

For simplicity, and to match the notation of [70], we fix the period $T = 2\pi$ and take N to be even. The nodes are thus $x_j = 2\pi j/N$, $j = 1, \ldots, N$.

A **product quadrature** approximates the integral of the product of a general smooth 2π-periodic real function f with a fixed known (and possibly singular) 2π-periodic real function g, by a periodic trapezoidal rule with modified weights w_j,

$$\int_0^{2\pi} f(s)g(s)\,ds \;\approx\; \sum_{j=1}^{N} w_j f(x_j)\ . \tag{2.29}$$

Using the Fourier series $f(s) = \sum_{n\in\mathbb{Z}} f_n e^{ins}$, and similar for g, we recognize the integral as an inner product and use Parseval,

$$\int_0^{2\pi} f(s)g(s)\,ds \;=\; 2\pi \sum_{n\in\mathbb{Z}} f_n \overline{g_n}\ . \tag{2.30}$$

Since f is smooth, $|f_n|$ decays to zero with a high order as $|n| \to \infty$. Thus we can make two approximations. Firstly, we truncate the infinite sum to $\sum'_{|n|\le N/2}$, where the prime indicates that the extreme terms $n = \pm N/2$ are given a factor of $1/2$. Secondly, we use the periodic trapezoidal rule to evaluate the Fourier coefficients of f, i.e.

$$f_n \;=\; \frac{1}{2\pi}\int_0^{2\pi} e^{-ins} f(s)ds \;\approx\; \frac{1}{N}\sum_{j=1}^{N} e^{-inx_j} f(x_j)\ . \tag{2.31}$$

Although the latter introduces aliasing (one may check that the latter sum is exactly $f_n + f_{n+N} + f_{n-N} + f_{n+2N} + f_{n-2N} + \cdots$), the decay of f_n means that errors decay to high order with N. Substituting (2.31) into the truncated version of (2.30) gives

$$\int_0^{2\pi} f(s)g(s)\,ds \;\approx\; 2\pi \sideset{}{'}\sum_{|n|\le N/2} \overline{g_n}\frac{1}{N}\sum_{j=1}^{N} e^{-inx_j} f(x_j) \;\approx\; \sum_{j=1}^{N}\left(\frac{2\pi}{N}\sideset{}{'}\sum_{|n|\le N/2} e^{-inx_j}\overline{g_n}\right) f(x_j) \tag{2.32}$$

The bracketed expression gives the weights in (2.29). Since g is real (hence $g_{-n} = \overline{g_n}$),

$$w_j \;=\; \frac{2\pi}{N} \sum_{|n|\le N/2}{}' e^{-inx_j}\overline{g_n} \;=\; \frac{2\pi}{N}\left[g_0 + \sum_{n=1}^{N/2-1} 2\mathrm{Re}(g_n e^{inx_j}) + \mathrm{Re}(g_{N/2}e^{iNx_j/2})\right], \quad j = 1,\ldots,N. \tag{2.33}$$

2.6.2 The Kress quadrature

To derive the scheme of Kress (originally due to Martensen–Kussmaul; see references in [70]) we note the Fourier series (proved in [71, Thm. 8.21]),

$$g(s) = \log\left(4\sin^2\frac{s}{2}\right) \qquad \Leftrightarrow \qquad g_n = \begin{cases} 0, & n = 0, \\ -1/|n|, & n \neq 0. \end{cases} \tag{2.34}$$

Translating g by a displacement $t \in \mathbb{R}$ corresponds to multiplication of g_n by e^{-int}. Substituting this displaced series into (2.33) and simplifying gives

$$\int_0^{2\pi} \log\left(4\sin^2\frac{t-s}{2}\right)\varphi(s)\,ds \;\approx\; \sum_{j=1}^{N} R_j^{(N/2)}(t)\,\varphi(x_j)\ , \tag{2.35}$$

where the weights, which depend on the target location t, are

$$R_j^{(N/2)}(t) \;=\; -\frac{4\pi}{N}\left[\sum_{n=1}^{N/2-1}\frac{1}{n}\cos n(x_j-t) \;+\; \frac{1}{N}\cos\frac{N}{2}(x_j-t)\right], \quad j = 1,\ldots,N. \tag{2.36}$$

This matches [70, (3.1)] (note the convention on the number of nodes differs by a factor 2).

When a smooth function is also present, we use the periodic trapezoidal rule for it, to get

$$\int_0^{2\pi} \log\left(4\sin^2\frac{t-s}{2}\right)\varphi(s) + \psi(s)\,ds \;\approx\; \sum_{j=1}^{N} R_j^{(N/2)}(t)\,\varphi(x_j) + \frac{2\pi}{N}\sum_{j=1}^{N}\psi(x_j)\ . \tag{2.37}$$

Assuming the separation into φ and ψ is known, this gives a high-order accurate quadrature; in fact for φ and ψ analytic, it is exponentially convergent [70].

2.6.3 A Nyström scheme

We use the above Kress quadrature to approximate the integral (2.4) where the kernel has the form (2.2) with $T = 2\pi$, and the functions $\varphi(x, x')$ and $\psi(x, x')$ are separately known. Applying

(2.37), with $h = 2\pi/N$, gives

$$\int_0^{2\pi} k(x_i, x')\sigma(x')\,dx' \;\approx\; \sum_{j=1}^{N} R_j^{(N/2)}(x_i)\varphi(x_i, x_j)\sigma(x_j) + h\sum_{j=1}^{N} \psi(x_i, x_j)\sigma(x_j)\ . \tag{2.38}$$

Using the symbol $R_j^{(N/2)} := R_j^{(N/2)}(0)$, and noticing that $R_j^{(N/2)}(x_i)$ depends only on $|i-j|$, we find that the entries of the coefficient matrix A are,

$$a_{i,j} \;=\; R_{|i-j|}^{(N/2)}\varphi(x_i, x_j) + h\,\psi(x_i, x_j)\ . \tag{2.39}$$

Note that $R_{|i-j|}^{(N/2)}$ is a dense circulant matrix, and all N^2 elements differ from the standard Nyström matrix (2.7). Since φ and ψ do not usually have fast potential-theory based algorithms to apply them, the Kress scheme is not FMM-compatible.

2.7 Numerical experiments

We now describe numerical experiments in 1D, 2D and 3D applications that illustrate the performance of the quadratures described in sections 2.3-2.6. The experiments were carried out on a Macbook Pro with 2.4GHz Intel Core 2 Duo and 4GB of RAM, and executed in a MATLAB environment. Once the Nyström matrix is filled, the linear system (2.6) is solved via MATLAB's backslash (`mldivide`) command. In all examples below, the errors reported are relative errors measured in the L^∞-norm, $||u_\epsilon - u||_\infty/||u||_\infty$, where u is the reference solution and u_ϵ is the numerical solution. For each experiment we compare the performance of the different quadratures. Specifically, we compare the rules of Kapur–Rokhlin of orders 2, 6, and 10; Alpert of orders 2, 6, 10, and 16; modified Gaussian with $n = 10$ points per panel; and (where convenient) Kress. Our implementation of the modified Gaussian rule uses $m = 20$ auxiliary nodes for source and target on the same panel, and $m' = 24$ when on adjacent panels.

The quadrature nodes and weights used are provided in appendices and at the website [49].

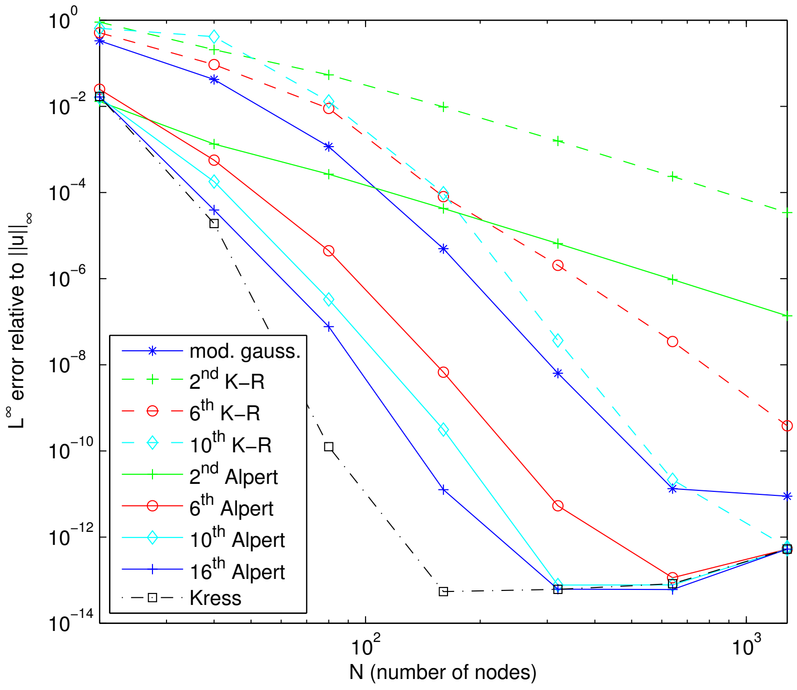


Figure 2.3: Error results for solving the integral equation (2.40) in Section 2.7.1.

2.7.1 A 1D integral equation example

We solve the one-dimensional integral equation

$$u(x) + \int_0^{2\pi} k(x,x')u(x')dx' = f(x), \qquad x \in [0,\, 2\pi] \tag{2.40}$$

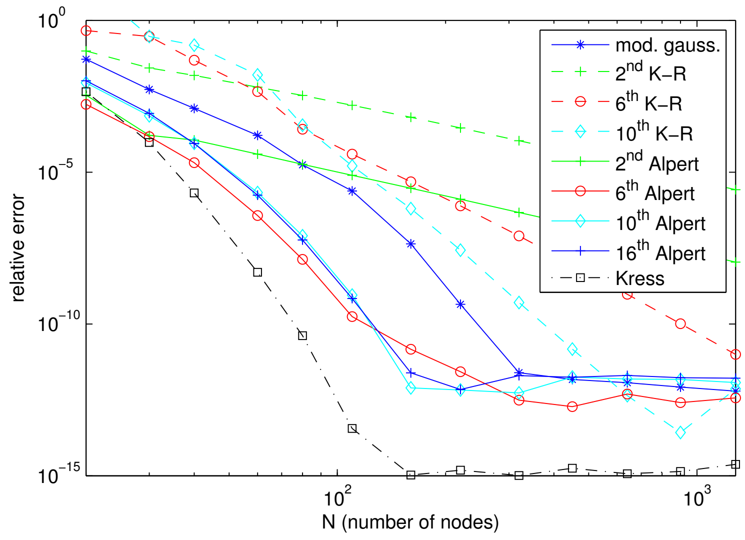
associated with a simple kernel function having a periodic log singularity at the diagonal,

$$k(x,x') = \frac{1}{2}\log\left|\sin\frac{x-x'}{2}\right| = \frac{1}{4}\log\left(4\sin^2\frac{t-s}{2}\right) - \frac{1}{2}\log 2\ , \tag{2.41}$$

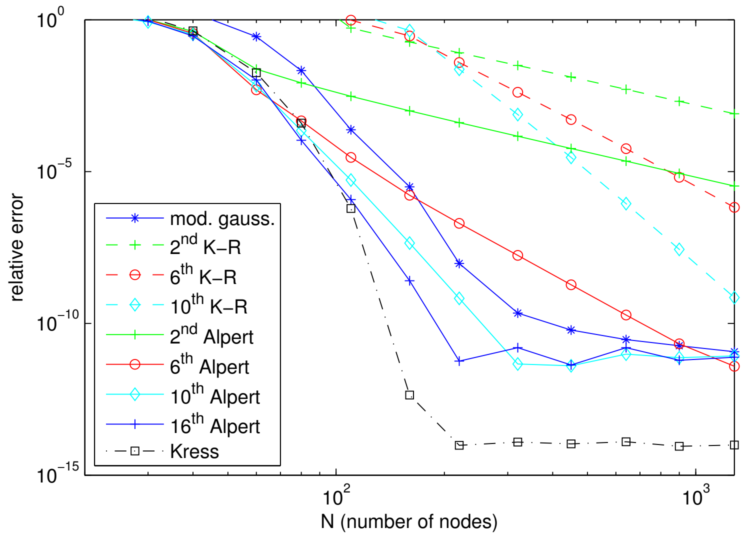
thus the smooth functions $\varphi(x,x') = 1/4$ and $\psi(x,x') = -(1/2)\log 2$ are constant. This kernel is similar to that arising from the Laplace single-layer operator on the unit circle. Using (2.34) one may check that the above integral operator has exact eigenvalues $-\pi\log 2$ (simple) and $-\pi/(2n)$, $n = 1, 2, \ldots$ (each doubly-degenerate). Thus the exact condition number of the problem (2.40) is $((\pi\log 2) - 1)/(1 - \pi/4) \approx 5.5$. We choose the real-analytic periodic right-hand side $f(x) = \sin(3x)\, e^{\cos(5x)}$. The solution u has $\|u\|_\infty \approx 6.1$. We estimate errors by comparing to the Kress solution at $N = 2560$. (In passing we note that the exact solution to (2.40) could be written analytically as a Fourier series since the Fourier series of f is known in terms of modified Bessel functions.) When a solution is computed on panel-based nodes, we use interpolation back to the uniform trapezoidal grid by evaluating the Lagrange basis on the $n = 10$ nodes on each panel.

In Figure 2.3, the errors in the L^∞-norm divided by $\|u\|_\infty$ are presented for $N = 20, 40, 80, \ldots, 1280$. We see that the rules of order 2, 6, and 10 have the expected convergence rates, but that Alpert has prefactors a factor 10^2 to 10^5 smaller the Kapur–Rokhlin. We also see that Kress is the most efficient at any desired accuracy, followed by the three highest-order Alpert schemes. These four schemes flatten out at 13 digits, but errors start to grow again for larger N, believed due to the larger linear system. Note that modified Gaussian performs roughly as well as 6th-order Alpert with twice the number of points, and that it flattens out at around 11 digits.

(a) 0.5 λ diameter
($\omega = 2.8$)



(b) 5 λ diameter
($\omega = 28$)



(c) 50 λ diameter
($\omega = 280$)

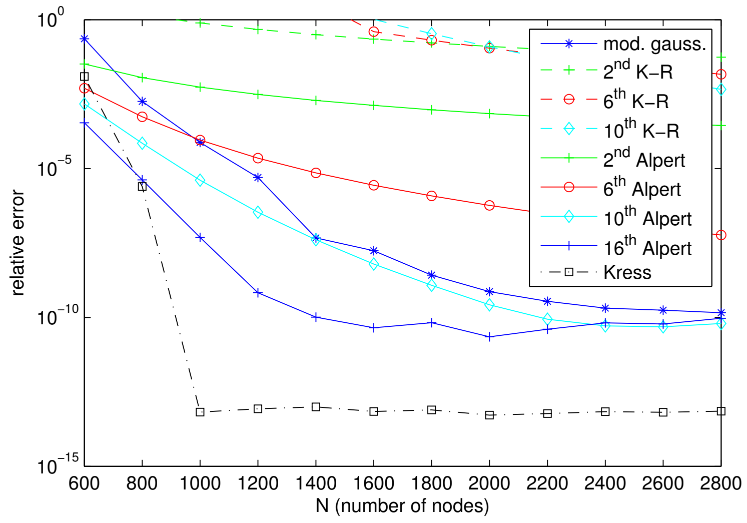


Figure 2.4: Error results for the exterior planar Helmholtz problem (2.42) in Section 2.7.2 solved on the starfish domain of Figure 2.1.

2.7.2 Combined field discretization of the Helmholtz equation in $\mathbb{R}^2$

In this section, we solve the Dirichlet problem for the Helmholtz equation exterior to a smooth domain $\Omega \subset \mathbb{R}^2$ with boundary Γ,

$$-\Delta u - \omega^2 u = 0, \qquad \text{in } \; E = \Omega^c, \tag{2.42}$$

$$u = f, \qquad \text{on } \; \Gamma, \tag{2.43}$$

where $\omega > 0$ is the wavenumber. u satisfies the Sommerfeld radiation condition

$$\lim_{r\to\infty} r^{1/2}\left(\frac{\partial u}{\partial r} - i\omega u\right) = 0, \tag{2.44}$$

where $r = |\mathbf{x}|$ and the limit holds uniformly in all directions. A common approach [27, Ch. 3] is to represent the solution to (2.42) via both the single and double layer acoustic potentials,

$$\begin{aligned} u(\mathbf{x}) &= \int_\Gamma k(\mathbf{x},\mathbf{x}')\,\sigma(\mathbf{x}')\,dl(\mathbf{x}') \\ &= \int_\Gamma \left(\frac{\partial \phi(\mathbf{x},\mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} - i\omega\,\phi(\mathbf{x},\mathbf{x}')\right)\sigma(\mathbf{x}')\,dl(\mathbf{x}'), \qquad \mathbf{x} \in E, \end{aligned} \tag{2.45}$$

where $\phi(\mathbf{x},\mathbf{x}') = \frac{i}{4}H_0^{(1)}(\omega|\mathbf{x}-\mathbf{x}'|)$ and $H_0^{(1)}$ is the Hankel function of the first kind of order zero; $\mathbf{n}$ is the normal vector pointing outward to Γ, and dl the arclength measure on Γ. The motivation for the combined representation (2.45) is to obtain the unique solvability to problem (2.42-2.43) for all $\omega > 0$. The corresponding boundary integral equation we need to solve is

$$\frac{1}{2}\sigma(\mathbf{x}) + \int_\Gamma k(\mathbf{x},\mathbf{x}')\sigma(\mathbf{x}')\,dl(\mathbf{x}') = f(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \tag{2.46}$$

where $k(\mathbf{x},\mathbf{x}') = \frac{\partial\phi(\mathbf{x},\mathbf{x}')}{\partial\mathbf{n}(\mathbf{x}')} - i\omega\,\phi(\mathbf{x},\mathbf{x}')$.

To convert (2.46) into an integral equation on the real line, we need to parametrize Γ by a vector-valued smooth function $\boldsymbol{\tau} : [0,T] \to \mathbb{R}^2$. By changing variable, (2.46) becomes

$$\frac{1}{2}\sigma(\boldsymbol{\tau}(t)) + \int_0^T k(\boldsymbol{\tau}(t),\boldsymbol{\tau}(s))\,\sigma(\boldsymbol{\tau}(s))\,|d\boldsymbol{\tau}/ds|\,ds = f(\boldsymbol{\tau}(t)), \qquad t \in [0,T]. \tag{2.47}$$

To keep our formula uncluttered, we rewrite the kernel as

$$m(t,s) = k(\boldsymbol{\tau}(t),\boldsymbol{\tau}(s))\,|d\boldsymbol{\tau}/ds|, \tag{2.48}$$

as well as the functions

$$\sigma(t) = \sigma(\boldsymbol{\tau}(t)) \quad \text{and} \quad f(t) = f(\boldsymbol{\tau}(t)).$$

Thus we may write the integral equation in standard form,

$$\sigma(t) + 2\int_0^T m(t,s)\,\sigma(s)\,ds = 2f(s), \qquad t \in [0,T], \tag{2.49}$$

and apply the techniques of this paper to it. All the quadrature schemes apart from that of Kress are now easy to implement by evaluation of $m(t,s)$. For the Kress scheme, some additional work is required to split this kernel into analytic functions φ and ψ, see [27, p. 68], or [57, eq. (27)].

We assess the accuracy of each quadrature rule for the smooth domain shown in Figure 2.1(c), varying N and the wavenumber ω. Specifically, we varied wavenumbers such that there are 0.5, 5 and 50 wavelengths across the domain's diameter. The right-hand side is generated by a sum of five point sources inside Ω, with random strengths; thus the exact exterior solution is known. Errors are taken to be the maximum relative error at the set of measurement points shown in Figure 2.1(c). Notice that sources and measurement points are both far from Γ, thus no challenges due to close evaluation arise here.

The results are shown in Figure 2.4. At the lowest frequency, results are similar to Figure 2.3, except that errors bottom out at slightly higher accuracies, probably because of the smoothing effect of evaluation of u at distant measurement points. In terms of the error level each scheme saturates at, Kress has a 2-3 digit advantage over the others at all frequencies. At the highest frequency, Figure 2.4(c), there are about 165 wavelengths around the perimeter Γ and we find that the point at which the Kress quadrature saturates ($N = 1000$) corresponds to around 6 points per wavelength. At 10 points per wavelength (apparently a standard choice in the engineering community, and roughly halfway along the N axis in our plot), 16th-order Alpert has converged at 10 digits, while modified Gaussian and 10th-order Alpert are similar at 8 digits. The other schemes are not competitive.

scheme	mod. Gauss	2nd K-R	6th K-R	10th K-R	2nd Alpert	6th Alpert	10th Alpert	16th Alpert	Kress
cond #	3.95	3.52	3.68	169	3.52	3.52	3.52	3.52	3.52
# iters	14	14	22	206	14	14	14	14	14

Table 2.2: Condition numbers of the Nyström system matrix $(\frac{1}{2}\mathsf{I} + \mathsf{A})$, and numbers of GMRES iterations to reach residual error 10^{-12}, for all the quadrature schemes.

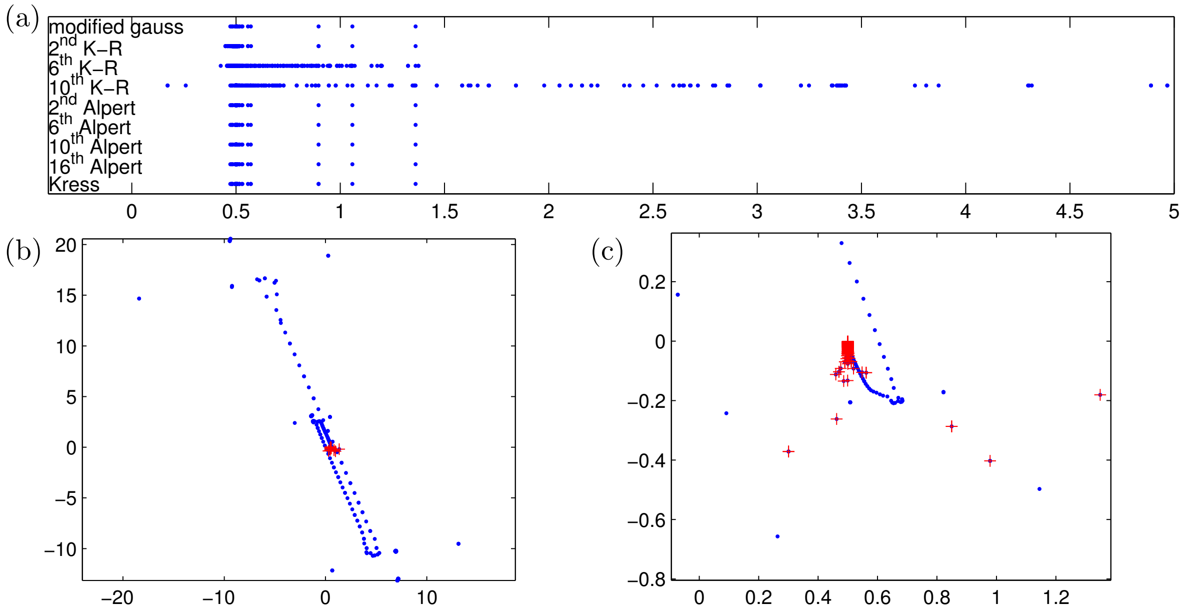


Figure 2.5: (a) Magnitude of eigenvalues of the matrix $(\frac{1}{2}\mathsf{I} + \mathsf{A})$ associated with the Nyström discretization of the Helmholtz BVP (2.46). The system size is $N = 640$ and the wave-number ω corresponds to a contour of size 0.5 wave-lengths. (b) Eigenvalues in the complex plane associated with 10th-order Kapur–Rokhlin (dots) and Kress (crosses) quadratures. (c) Same plot as (b), but zoomed in to the origin.

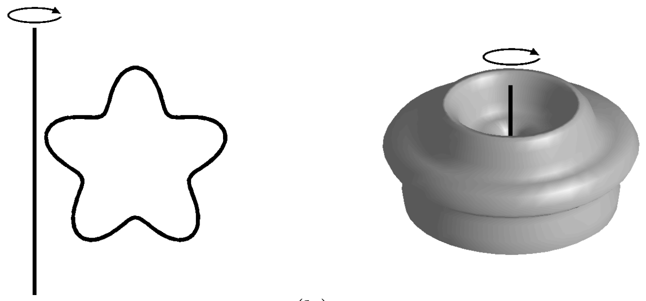
2.7.3 Effect of quadrature scheme on iterative solution efficiency

When the number of unknowns N becomes large, iterative solution becomes an important tool. One standard iterative method for nonsymmetric systems is GMRES [87]. In Table 2.2 we compare the numbers of GMRES iterations needed to solve the linear system (2.4) arising from the low-frequency Helmholtz BVP in section 2.7.2 (0.5 wavelengths across), when the matrix was constructed via the various quadrature schemes. We also give the condition number of the system. We see that most schemes result in 14 iterations and a condition number of around 3.5; this reflects the fact that the underlying integral equation is Fredholm 2nd-kind and well-conditioned. However, 6th-order and particularly 10th-order Kapur–Rokhlin require many more iterations (a factor 15 more in the latter case). In the latter case the condition number has also now become two orders of magnitude larger. In practical settings this could have negative consequences for both solution time and accuracy.

In order to understand the cause of the above, in Figure 2.5 we study the spectra of the system matrix; since the operator is of the form Id/2 + compact, the matrix eigenvalues should also cluster at 1/2. Indeed, this is the case for all of the schemes. However, as subfigure (a) shows, 6th- and 10th-order Kapur–Rokhlin contain many additional eigenvalues that appear unrelated to those of the operator. In the 10th-order case, these create a wide "spray" of many eigenvalues with much larger magnitudes. Examining the latter in the complex plane in subfigures (b) and (c), we see that spurious eigenvalues (there are around 200 of them) fall on both sides of the origin, and appear to fill densely diagonal lines passing very close to the origin. It is well known that the GMRES convergence rate can be related to the size of the potential at the origin for a certain electrostatic problem in the plane [?]: if $S \subset \mathbb{C}$ is a compact set approximating the matrix spectrum, then one enforces zero potential on S while fixing unit total charge on S. Thus when large parts of the spectrum nearly surround the origin, as the spurious eigenvalues here appear to do, the potential at the origin is nearly zero, causing a drastic decrease in convergence rate. We believe that this explains the slow convergence of high-order Kapur–Rokhlin rules relative to the other schemes.

The corresponding spurious eigenvectors are oscillatory, typically alternating in sign from node to node. Thus we believe this pollution of the spectrum arises from the large alternating weights γ_l in these rules. Note that one spurious eigenvalue falls quite near the origin; this mechanism could induce an arbitrarily large condition number, even though the integral equation condition number is small. Although an isolated eigenvalue does not have impact on the GMRES convergence rate, the increased condition number can result in unnecessary loss of accuracy. Although we do not test the very large N values where iterative methods become essential, we expect that our conclusions apply also to larger N.

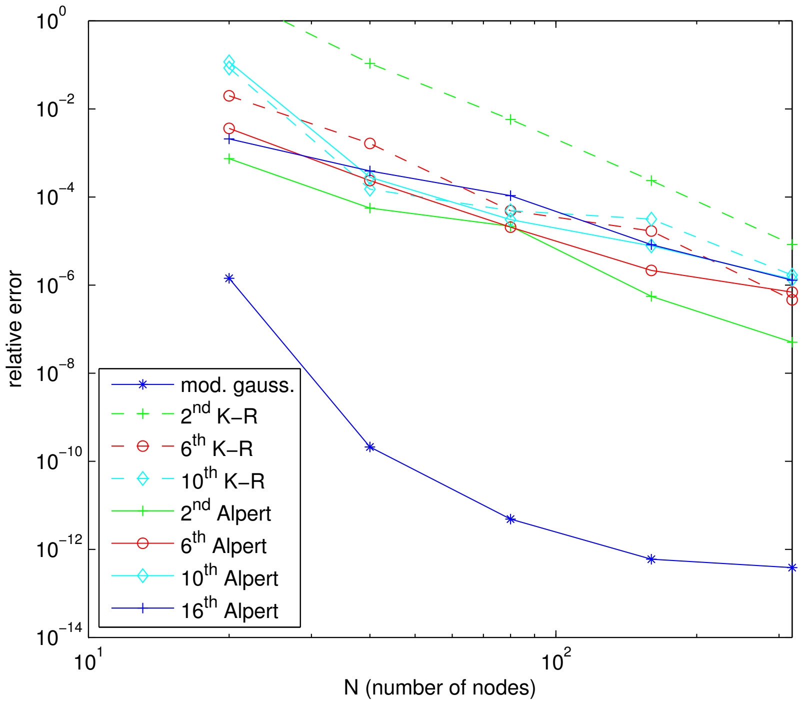
(a)



(b)

Figure 2.6: Domains used in numerical examples in Section 2.7.4. All items are rotated about the vertical axis. (a) A sphere. (b) A starfish torus.

(a)



(b)

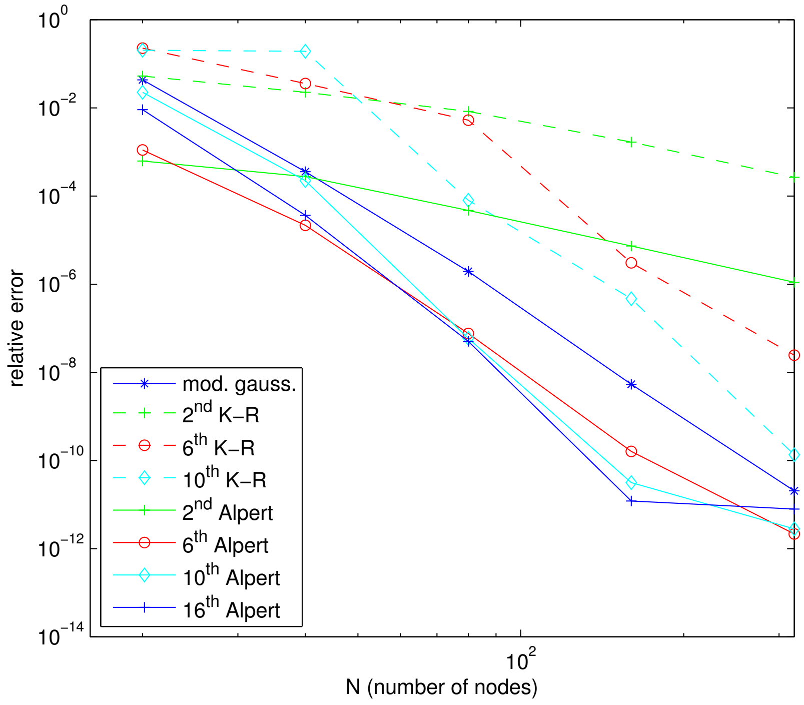


Figure 2.7: Error results for the 3D interior Dirichlet Laplace problem from section 2.7.4 solved on the axisymmetric domains (a) and (b) respectively shown in Figure 2.6.

2.7.4 The Laplace BVP on axisymmetric surfaces in $\mathbb{R}^3$

In this section, we compare quadratures rules applied on kernels associated with BIEs on rotationally symmetric surfaces in $\mathbb{R}^3$. Specifically, we considered integral equations of the form

$$\sigma(\mathbf{x}) + \int_\Gamma k(\mathbf{x},\mathbf{x}')\,\sigma(\mathbf{x}')\,dA(\mathbf{x}') = f(\mathbf{x}), \quad x \in \Gamma, \tag{2.50}$$

under the assumptions that Γ is a surface in $\mathbb{R}^3$ obtained by rotating a curve γ about an axis and the kernel function k is invariant under rotation about the symmetry axis. Figure 2.6 depicts domains used in numerical examples: the generating curves γ are shown in the left figures and the axisymmetric surfaces Γ are shown in the right ones. The BIE (2.50) on rotationally symmetric surfaces can via a Fourier transform be recast as a sequence of equations defined on the generating curve in cylindrical coordinates, i.e.

$$\sigma_n(r,z) + \sqrt{2\pi}\int_\Gamma k_n(r,z,r',z')\,\sigma_n(r',z')\,r'\,dl(r',z') = f_n(r,z), \quad (r,z)\in\gamma, \quad n\in\mathbb{Z}, \tag{2.51}$$

where σ_n, f_n, and k_n denote the Fourier coefficients of σ, f, and k, respectively. Details on how to truncate the Fourier series and construct the coefficient matrices for Laplace problem and Helmholtz problem can be found in [104]. In the following experiments, we consider the BIE (2.50) which arises from the interior Dirichlet Laplace problem, in which case

$$k(\mathbf{x},\mathbf{x}') = \frac{\mathbf{n}(\mathbf{x}')\cdot(\mathbf{x}-\mathbf{x'})}{4\pi|\mathbf{x}-\mathbf{x'}|^3}. \tag{2.52}$$

As we recast the BIE defined on Γ to a sequence of equations defined on the generating curve γ, it is easy to see that the kernel function k_n has a logarithmic singularity as $(r',z') \to (r,z)$. In this experiment, 101 Fourier modes were used. For the axisymmetric kernels k_n, analytic splits that extract the singularity are known [75, 68], but are slightly tricky to implement, in particular for contours involving "poles" such as, e.g., the one shown in Figure 2.6(a). We therefore did not implement the quadrature of Kress for this example (even though it likely would perform very well).

Equation (2.50) was solved for Dirichlet data f corresponding to an exact solution u generated by point charges placed outside the domain. The errors reported reflect the maximum of the pointwise errors (compared to the known analytic solution) sampled at a set of target points inside the domain.

The results are presented in Figure 2.7. The most apparent feature in (a) is that, because the curve γ is open, the schemes based on the periodic trapezoidal rule fail to give high-order convergence; rather, it appears to be approximately 3rd-order. Panel-based schemes are able to handle open intervals as easily as periodic ones, thus modified Gaussian performs well: it reaches 12-digit accuracy with only around $N = 100$ points. (We remark that the problems associated with an open curve are in this case artificial and can be overcome by a better problem formulation. We deliberately chose to use a simplistic formulation to simulate an "open curve" problem.) In (b), all functions are again periodic since γ is closed; modified Gaussian performs similarly to the three highest-order Alpert schemes with around 1.5 to 2 times the number of points.

2.8 Concluding remarks

To conclude, we make some informal remarks on the relative advantages and disadvantages of the different quadrature rules that we have discussed. Our remarks are informed primarily by the numerical experiments in section 5.6.

Comparing the three schemes based upon nodes equi-spaced in parameter (Kapur–Rokhlin, Alpert, and Kress), we see that Kress always excels due to its superalgebraic convergence; it reaches a small saturation error of 10^{-13} to 10^{-15} at only 6 points per wavelength. However, the analytic split required for Kress is not always available or convenient, and Kress is not amendable to standard FMM-style fast matrix algebra. Kapur–Rokhlin and Alpert show their expected algebraic convergence rates at orders 2, 6, and 10, and both seem to saturate at around 10^{-12}. However, Alpert outperforms Kapur–Rokhlin since its prefactor is much lower, resulting in 2-8 extra digits of accuracy at the same N. Another way to compare these two is that Kapur–Rokhlin requires around 6 to 10 times the number of unknowns as Alpert to reach comparable accuracy. The difference in a high-frequency problem is striking, as in Figure 2.4. The performance of 16th-order Alpert is usually less than 1 digit better than 10th-order Alpert, apart from at high frequency when it can be up to 3 digits better.

Turning to the panel-based modified Gaussian scheme, we see that in the low-frequency settings it behaves like 10th-order Alpert but requires around 1.5 to 2 times the N to reach similar accuracy. This may be related to the fact that Gauss–Legendre panels would need a factor $\pi/2$ higher N than the trapezoidal rule to achieve the same largest spacing between nodes; this is the price to pay for a panel-based scheme. However, for medium and high frequencies, 10th-order Alpert has little distinguishable advantage over modified Gaussian. Both are able to reach around 10 digit accuracy at 15 points per wavelength. Modified Gaussian errors seem to saturate at around 10^{-11} to 10^{-12}. It therefore provides a good all-round choice, especially when adaptivity is anticipated, or global parametrizations are not readily constructed. One disadvantage relative to the other schemes is that the auxiliary nodes require kernel evaluations that are very close to the singularity (10^{-7} or less; for Alpert the minimum is only around 10^{-3}).

We have not tested situations in which adaptive quadrature becomes essential; in such cases modified Gaussian would excel. However, a hint of the convenience of modified Gaussian is given by its effortless handling of an open curve in Figure 2.7(a) where the other (periodic) schemes become low-order (corner-style reparametrizations would be needed to fix this [27, Sec. 3.5]).

In addition, we have showed that, in an iterative solution setting, higher-order Kapur–Rokhlin can lead to much slower GMRES convergence than any of the other schemes. We believe this is because it introduces many large eigenvalues into the spectrum, unrelated to those of the underlying operator. Thus 10th-order Kapur–Rokhlin should be used with caution. With that said, Kapur–Rokhlin is without doubt the simplest to implement of the four schemes, since no interpolation or new kernel evaluations are needed.

We have chosen to not report computational times in this note since our MATLAB implementations are far from optimized for speed. However, it should be mentioned that both the Alpert method and the method based on modified Gaussian quadrature require a substantial number (between $20N$ and $30N$ in the examples reported) of additional kernel evaluations.

For simplicity, in this note we limited our attention to the case of smooth contours, but both the Alpert and the modified Gaussian rule can with certain modifications be applied to contours with corners, see, e.g., [83, 14, 15, 11, 60, 56]. We plan to include the other recent schemes shown in Table 2.1, and curves with corners, in future comparisons.

Chapter 3

A simplified technique for the efficient and highly accurate discretization of boundary integral equations in 2D on domains with corners

A. Gillman, S. Hao, P.G. Martinsson

Note: *The work described in this chapter was carried out in collaboration with Adrianna Gillman of Dartmouth and Professor Per-Gunnar Martinsson of the University of Colorado. It appeared in the Journal of Computational Physics (256(1), pages 214–219, 2014) under the title: "A simplified technique for the efficient and highly accurate discretization of boundary integral equations in 2D on domains with corners " .*

3.1 Background

This note comments on some recently developed techniques for computing an approximate solution to a Boundary Integral Equation (BIE) like

$$\alpha\, q(\boldsymbol{x}) + \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\, q(\boldsymbol{y})\, ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x}\in\Gamma, \tag{3.1}$$

where Γ is a piecewise smooth contour in the plane, and where K is one of the standard kernels of potential theory such as, e.g., the single or double layer kernels associated with the Laplace or Helmholtz equations. A challenge in solving (3.1) is that its integrand exhibits complicated singular behavior near the corner points of Γ. A classical technique for dealing with this difficulty has been to expand the unknown q near the corner using specialized basis functions that incorporate analytical knowledge about the singularity [16]. Recently, however, a remarkable observation has been made

[10, 55, 59, 53] that there exist general purpose techniques that do not require any analytical à priori knowledge other than that the integrand of (3.1) be absolutely integrable.

In a nutshell, the idea of [10, 55, 59, 53] is to use a standard Nyström discretization of (3.1) designed for a smooth contour. The discretization should use a panel based (i.e. non-global) quadrature rule such as, e.g., a composite Gaussian rule. Then simply refine the computational mesh near any corner. For any given computational tolerance ε (setting $\varepsilon = 10^{-10}$ or smaller is often entirely manageable), continue the refinement until the contribution from any panels directly touching a corner is bounded by ε (this is possible since the integrand in (3.1) is absolutely integrable). Then simply omit the two panels nearest to the corner from the discretization. Observe that on any remaining panel, the function q is smooth enough to be accurately represented by the interpolant implied by the chosen quadrature rule.

The apparent drawback of a simplistic refinement process like the one described is that it can dramatically increase the number of degrees of freedom required in the Nyström discretization. A key insight of [10, 55, 59, 53] is that the "superfluous" degrees of freedom added by the refinement can be eliminated from the linear system via a *strictly local* process. Moreover, this local process can be executed in time that scales linearly with the number of degrees of freedom added. The end result is a linear system discretizing (3.1) that has about as many degrees of freedom as one would have needed had the corner not been present in the first place. (For the case of regular polygonal domains, the compression can even be performed in sublinear time [53].)

The task of "squeezing out" the degrees of freedom added by the local refinement near the corner is in [10, 55, 59, 53] executed via purpose-built local compression techniques that can be somewhat challenging to implement. The purpose of this note is to demonstrate that this compression step can be executed via the general purpose direct solvers described in [36, 42, 63, 78].

3.2 A linear algebraic observation

The compression technique that allows us to eliminate the superfluous degrees of freedom is based on the observation that certain off-diagonal blocks of the coefficient matrix resulting from the discretization of (3.1) have low numerical rank. Critically, the important ranks do not depend on how many degrees of freedom are used in the refinement near the corner. To illustrate how such rank-deficiencies can be exploited, consider in general the task of solving the linear system

$$\begin{bmatrix} \mathsf{A}_{11} & \mathsf{A}_{12} \\ \mathsf{A}_{21} & \mathsf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}, \tag{3.2}$$

where A_{11} is of size $n_1 \times n_1$ and A_{22} is of size $n_2 \times n_2$. Now assume that A_{12} and A_{21} each are of rank k. Think of n_1 as a large number (e.g. the number of degrees of freedom used in the refinement of the corner, say $n_1 \sim 10^3$), and k as a small number (often in the range $20 - 50$). Then A_{12} and A_{21} admit factorizations

$$\begin{array}{ccccc} \mathsf{A}_{12} & = & \mathsf{U}_1 & \mathsf{B}_{12} \\ n_1 \times n_2 & & n_1 \times k & k \times n_2 \end{array} \quad \text{and} \quad \begin{array}{ccccc} \mathsf{A}_{21} & = & \mathsf{B}_{21} & \mathsf{V}_1^*, \\ n_2 \times n_1 & & n_2 \times k & k \times n_1 \end{array} \tag{3.3}$$

where U_1 and V_1 are well-conditioned matrices. We further assume that the data vector $\mathbf{f}_1$ belongs to the same k-dimensional space as the columns of A_{12} (if it does not, then the space can be extended as needed),

$$\mathbf{f}_1 = \mathsf{U}_1 \tilde{\mathbf{f}}_1. \tag{3.4}$$

When (3.3) and (26) hold, the linear system (3.2) with $n_1 + n_2$ unknowns is in a certain sense equivalent to the smaller system

$$\begin{bmatrix} \mathsf{D}_{11} & \mathsf{B}_{12} \\ \mathsf{B}_{21} & \mathsf{A}_{22} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_1 \\ \mathbf{q}_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}_1 \\ \mathbf{f}_2 \end{bmatrix} \tag{3.5}$$

with only $k + n_2$ unknowns. In (3.5), D_{11} and $\tilde{\mathbf{q}}_1$ are defined by

$$\mathsf{D}_{11} = \left(\mathsf{V}_1^* \mathsf{A}_{11}^{-1} \mathsf{U}_1\right)^{-1} \qquad \text{and} \qquad \tilde{\mathbf{q}}_1 = \mathsf{V}_1^* \mathbf{q}_1. \tag{3.6}$$

When we say that (3.2) and (3.5) are "equivalent" we mean that the solution $\{\mathbf{q}_1, \mathbf{q}_2\}$ of the larger system (3.2) can be obtained from the solution $\{\tilde{\mathbf{q}}_1, \mathbf{q}_2\}$ of the smaller system (3.5) via the formula

$$\mathbf{q}_1 = \mathsf{A}_{11}^{-1}\,\mathsf{U}_1\,\mathsf{D}_{11}\,\tilde{\mathbf{q}}_1. \tag{3.7}$$

To be precise, the equivalence holds when A_{11} and $\mathsf{V}_1^*\mathsf{A}_{11}^{-1}\mathsf{U}_1$ are both non-singular.

3.3 Matrix skeletons

For the low-rank factorizations (3.3), it is convenient to use a so-called *interpolative decomposition* (ID) [22] in which B_{12} is a $k \times n_2$ matrix consisting of k rows of A_{12} and B_{21} is an $n_2 \times k$ matrix consisting of k columns of A_{21}. The matrices U_1 and V_1 each hold a $k \times k$ identity matrix as a submatrix, and have no entries whose magnitude exceeds 1.

The advantage of using an ID is that the matrices A_{12} and A_{21} *need never be formed.* Instead, a local computation determines the index vectors pointing out which columns and rows are needed. and then only those entries need to be computed to create the off-diagonal blocks B_{12} and B_{21} in (3.5). Moreover, when skeletonization is used, the vector $\tilde{\mathbf{f}}_1$ can be formed by evaluating the vector $\mathbf{f}_1$ only at the k nodes associated with the spanning rows of A_{12}.

A strictly local technique for computing U_1 and V_1, and determining the corresponding index vectors, is described in Section 6.2 of [36].

3.4 Outline of the solution process

To describe the refinement and the local compression, we consider a contour with a single corner, like the one in Figure 3.1(a). We partition the contour into two disjoint parts, $\Gamma = \Gamma_1 \cup \Gamma_2$, in such a way that Γ_1 is a small piece containing the corner. The piece Γ_2 is smooth, and can be discretized into panels rather coarsely (since we use a high order rule, high accuracy does not require many points). For the piece Γ_1, we use a simplistic refinement strategy where we recursively cut the panel nearest the corner in half. Once the innermost panel is small enough that its contribution can be ignored (recall that we assume that the integrand is integrable), it is simply discarded and the refinement stops. The Nyström discretization now results in a linear system like (3.2). Observe that the block A_{11} can be very large since we may need thousands of points (or more [53]) to fully resolve the singularity near the corner. The key observation is now that the rank k of A_{12} and A_{21} is essentially independent of how finely the corner has been refined. This allows us to employ the direct solver of [36, 42, 63, 78] to compress the corner and eliminate the "extra" degrees of freedom used to resolve the singularity. The output of the compression is (i) a set of k collocation points inside Γ_1 that are automatically picked by the algorithm from among the n_1 points used in the refinement and (ii) a $k \times k$ dense matrix D_{11} that represents self-interaction among the k remaining points. The cost of this compression is $O(n_1\, k^2)$. Once the compression has been completed, all that remains is to solve the smaller system (3.5) to obtain the solution $\{\tilde{\mathbf{q}}_1,\, \mathbf{q}_2\}$, and, if required, reconstructing the full vector $\mathbf{q}_1$ via (3.7).

For a contour with multiple corners, simply repeat the local compression for each corner. Note that the compression processes are independent, meaning that they can be executed in parallel on a multi-core machine. The authors of [10, 55, 59, 53] have demonstrated the astonishing power of this observation by solving numerical examples involving tens of thousands of corners to close to full double precision accuracy.

Note: To achieve optimal accuracy, it is important to scale the matrix elements in the Nyström discretization as described in [12, 10]. The idea is to scale the vectors in the discretization by the quadrature weights so that for a panel Γ_p corresponding to an index vector I_p, we have $||q||_{L^2(\Gamma_p)} \approx ||\mathbf{q}(I_p)||_{\ell^2}$. The matrix elements in the coefficient matrix are scaled analogously so that $||\alpha\, q(\cdot) + \int_\Gamma k(\cdot, \boldsymbol{y})\, q(\boldsymbol{y})\, ds(\boldsymbol{y})||_{L^2(\Gamma_p)} \approx ||[\mathsf{A}\mathbf{q}](I_p)||_{\ell^2}$. The idea is to not give disproportionate weight to a region of the contour that ends up with a high density of discretization points due to a local refinement.

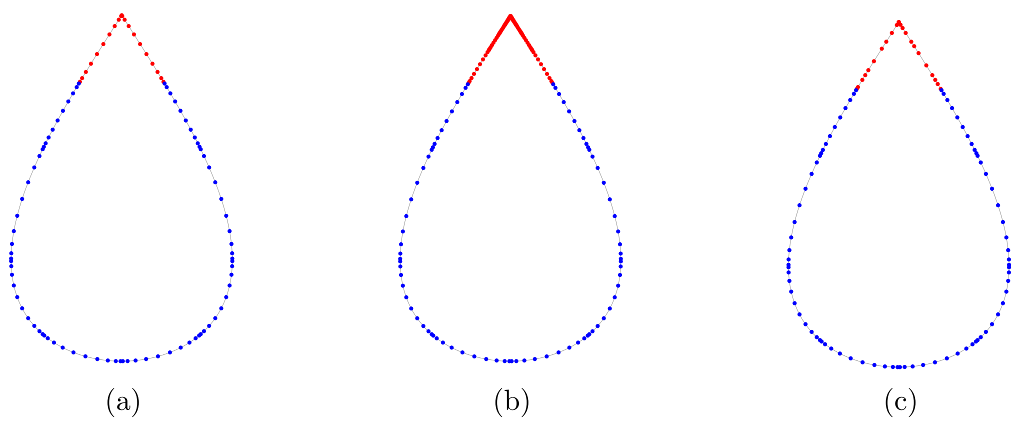


Figure 3.1: The boundary Γ_{tear} considered in the numerical experiments. (a) The original Gaussian grid before refinement, red nodes in Γ_1 and blue in Γ_2. (b) The locally refined grid. (c) The grid after local compression.

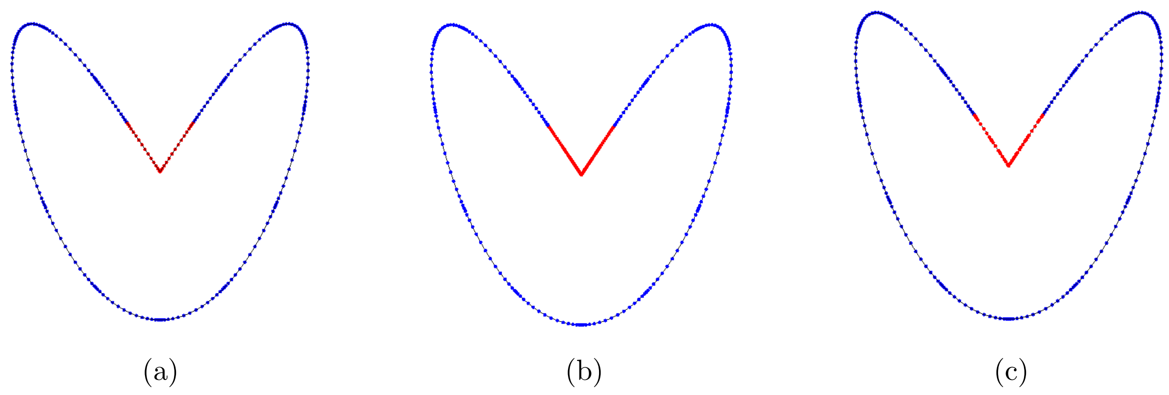


Figure 3.2: The boundary Γ_{pacman} considered in the numerical experiments. (a) The original Gaussian grid before refinement, red nodes in Γ_1 and blue in Γ_2. (b) The locally refined grid. (c) The grid after local compression.

3.5 Generalizations

The general idea of first discretizing very finely to fully resolve complicated behavior, and then compressing the resulting linear system to reduce the number of degrees of freedom to one appropriate for resolving macro-scale interactions can be applied not only to corners, but to many situations where the solution to a PDE or an integral equation exhibits complicated, but localized, behavior. For an application to a BIE defined on a contour that comes very close to touching itself (but does not actually touch), see [59, Sec. 10.3]. For an application to modeling of composite materials, see, e.g., [38, Sec. 4].

3.6 Numerical illustration

Let Ω denote a bounded domain with piece-wise smooth boundary Γ. On the exterior domain Ω^c, we consider a Dirichlet boundary value problem

$$\begin{cases} \Delta u(\boldsymbol{x}) + \omega^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega^c \\ u(\boldsymbol{x}) = g(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma = \partial\Omega \end{cases} \tag{D}$$

and a Neumann boundary value problem

$$\begin{cases} \Delta u(\boldsymbol{x}) + \omega^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega^c \\ \frac{\partial u}{\partial \boldsymbol{\nu}}(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma = \partial\Omega, \end{cases} \tag{N}$$

where ω is the wavenumber, $\boldsymbol{\nu}(\boldsymbol{x})$ represents the normal vector to Γ at the point $\boldsymbol{x} \in \Gamma$ pointing into Ω^c. Both (D) and (N) are coupled with the radiation condition $\sqrt{|\boldsymbol{x}|}\left(\frac{\partial}{\partial|x|} - \mathrm{i}\omega\right) u(\boldsymbol{x}) \to 0$ as $|\boldsymbol{x}| \to \infty$.

For the Dirichlet problem (D), we make the ansatz that the solution u can be represented by

$$u(\boldsymbol{x}) = \int_\Gamma \left(\frac{\mathrm{i}}{4}\frac{\partial}{\partial \boldsymbol{\nu}(\boldsymbol{y})} H_0(\omega|\boldsymbol{x} - \boldsymbol{y}|)\right) q(\boldsymbol{y}) \partial s(\boldsymbol{y})$$

where $\frac{\mathrm{i}}{4} H_0(\omega|\boldsymbol{x} - \boldsymbol{y}|)$ is the fundamental solution to the Helmholtz equation. By enforcing the boundary condition, the boundary charge distribution q is given by the solution of the integral equation

$$\frac{1}{2}q(\boldsymbol{x}) + \frac{\mathrm{i}}{4}\int_\Gamma \frac{\partial}{\partial \boldsymbol{\nu}(\boldsymbol{y})}\left(H_0(\omega|\boldsymbol{x}-\boldsymbol{y}|)\right) q(\boldsymbol{y})\partial s(\boldsymbol{y}) = g(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \Gamma. \tag{8}$$

For the Neumann problem (N), we make the ansatz that the solution u can be represented by

$$u(\boldsymbol{x}) = \int_\Gamma \frac{\mathrm{i}}{4} H_0(\omega|\boldsymbol{x}-\boldsymbol{y}|)q(\boldsymbol{y})\partial s(\boldsymbol{y}).$$

Now, the boundary charge distribution q is given determined by the equation

$$-\frac{1}{2}q(\boldsymbol{x}) + \frac{\mathrm{i}}{4}\int_\Gamma \frac{\partial}{\partial \boldsymbol{\nu}(\boldsymbol{x})}\left(H_0(\omega|\boldsymbol{x}-\boldsymbol{y}|)\right) q(\boldsymbol{y})\partial s(\boldsymbol{y}) = f(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \Gamma. \tag{9}$$

The equations (8) and (9) were discretized using a Nyström technique based on a 16-point composite Gaussian quadrature [67]. Certain matrix elements near the diagonal were modified to maintain high accuracy in spite of the (weakly) singular kernels as described in [50]. All experiments were run on a Lenovo laptop computer with 8GB of RAM and a 2.6GHz Intel i5-2540M processor. The compression technique was implemented rather crudely in MATLAB, which means that significant further gains in speed should be achievable.

For the Dirichlet problem, we considered a tear shaped geometry given by

$$\boldsymbol{x}(t) = \Big(2|\sin(\pi t)|, -\tan(\pi/4)\sin(2\pi t)\Big)$$

for $t \in [-0.5, 0.5]$ with arc-length approximately 1.3π, cf. Figure 3.1. For the Neumann problem, we considered a "pacman" shaped geometry given by $\boldsymbol{x}(t) = \Big(\text{sign}(2\pi \text{t})\sin(3\pi \text{t}), \tan(3\pi/4)\sin(2\pi \text{t})\Big)$ for $t \in [-0.5, 0.5]$ with arc-length of approximately 2.4π, cf Figure 3.2. We used the values $\omega = 1, 10$, and 100 (meaning the tear is 0.65, 6.5, and 65 wavelengths; while the pacman is $1.2, 12$, and 120 wavelengths). The boundary data is given by the incident wave $u^{\text{inc}}(\boldsymbol{x}) = e^{\mathrm{i}\omega x_2}$, so $g(\boldsymbol{x}) = u^{\text{inc}}(\boldsymbol{x})$ and $f(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{\nu}(\boldsymbol{x})}u^{\text{inc}}(\boldsymbol{x})$. We measured the error in the computed boundary charge q by

$$E_{\text{charge}} = \frac{\|q - q_{\text{exact}}\|_{L^2(\Gamma)}}{\|q_{\text{exact}}\|_{L^2(\Gamma)}}$$

and the error in the potential u (evaluated on the boundary of a circle S with radius 3 enclosing Ω) via

$$E_{\text{pot}} = \frac{\|u - u_{\text{exact}}\|_{L^2(S)}}{\|u_{\text{exact}}\|_{L^2(S)}}.$$

Since the exact solution was not available, we measured against a very highly over-resolved reference solution.

The computed solutions q for our two problems are shown in Figure 3.3. They both belong to $L^2(\Gamma)$, but the solution associated with the Neumann problem on the pacman geometry is unbounded.

Rows 1, 5, 9, 13, 17, and 21 of Table 3.1 reports the errors E_{charge} and E_{pot} when there is no refinement of the corner for the Dirichlet and Neumann boundary value problems. In all other experiments, the corner is discretized with 1024 points and is compressed for three prescribed tolerances ϵ. (We also tried solving the Dirichlet problem on the pacman geometry, but found that for such domains nine digits of accuracy was obtained *without* local refinement in the corner.) The remainder of Table 3.1 reports the size of the original system ($N \times N$), the size of the compressed system ($N_{\text{compressed}} \times N_{\text{compressed}}$), the number of skeleton nodes in the corner k, the time T_{compress} in seconds for the compression. We observe that very high accuracy can be attained for every wave-number. We also observe that the rank k of interaction between the corner patch (Γ_1) and the rest of the domain (Γ_2) is always very small, it depends only weakly on the requested accuracy, and it hardly depends at all on the wave-number.

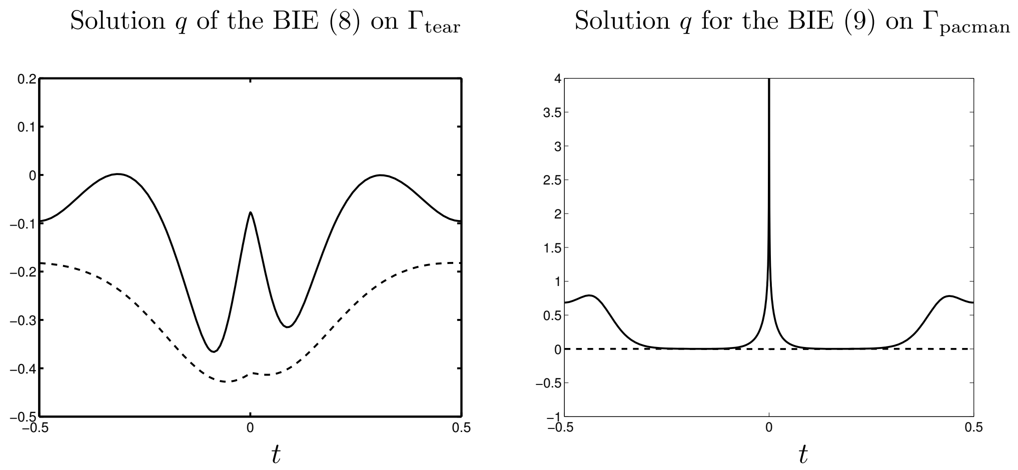


Figure 3.3: Plots of the solutions q for the BIEs (8) and (9) associated with, respectively, the Dirichlet problem (D) and the Neumann problem (N). The solid line is the real part, and the dotted line is the imaginary part.

3.7 Acknowledgements

The authors are grateful to James Bremer for the use of his high order quadrature.

	ω	N	ϵ	$N_{\text{compressed}}$	k	T_{compress}	E_{charge}	E_{pot}
Dirichlet on Γ_{tear}	1	128	—	—	—	—	$2.6e-01$	$5.2e-05$
		1152	$1e-7$	154	26	0.45	$1.2e-06$	$2.1e-07$
		1152	$1e-10$	167	39	0.61	$1.2e-09$	$8.0e-11$
		1152	$1e-12$	175	47	0.73	$9.3e-12$	$3.2e-13$
	10	576	—	—	—	—	$4.8e-01$	$5.1e-05$
		1600	$1e-7$	603	27	0.67	$1.3e-05$	$6.5e-07$
		1600	$1e-10$	616	40	0.75	$2.4e-08$	$3.0e-10$
		1600	$1e-12$	624	48	0.85	$2.8e-10$	$6.14e-12$
	100	1856	—	—	—	—	$1.1e-01$	$8.3e-05$
		2880	$1e-7$	1888	32	0.53	$1.2e-06$	$1.3e-06$
		2880	$1e-10$	1898	42	0.76	$1.0e-09$	$8.0e-10$
		2880	$1e-12$	1906	50	0.90	$4.6e-11$	$7.4e-12$
Neumann on Γ_{pacman}	1	224	—	—	—	—	$6.3e-01$	$1.0e-02$
		1248	$1e-7$	251	27	0.52	$4.0e-05$	$6.8e-07$
		1248	$1e-10$	264	40	0.70	$3.3e-08$	$5.8e-10$
		1248	$1e-12$	273	49	0.84	$1.1e-09$	$4.6e-12$
	10	576	—	—	—	—	$2.6e-01$	$1.7e-02$
		1600	$1e-7$	607	31	0.58	$4.4e-05$	$1.2e-06$
		1600	$1e-10$	619	43	0.76	$2.7e-08$	$7.6e-10$
		1600	$1e-12$	627	51	0.94	$4.3e-10$	$1.1e-11$
	100	1856	—	—	—	—	$6.0e-02$	$4.5e-03$
		3660	$1e-7$	2369	33	0.64	$4.5e-06$	$1.1e-06$
		3660	$1e-10$	2381	45	0.84	$1.4e-08$	$9.6e-10$
		3660	$1e-12$	2389	53	1.1	$1.3e-10$	$3.9e-12$

Table 3.1: Results from solving the external Helmholtz boundary value problems (D) and (N) on the geometries in Figures 3.1(a) and 3.2(a) for three different values of the wave-number ω. The errors E_{charge} and E_{pot} report the relative errors in the computed charge distribution q, and the evaluated potential, respectively. k is the rank of interaction (to precision ϵ) between the corner piece in Γ_1 and the rest of the contour, and N_{compress} is the size of the compressed system. T_{compress} is the time in seconds required for compressing the corner.

Chapter 4

A high-order Nyström discretization scheme for Boundary Integral Equations Defined on Rotationally Symmetric Surfaces

P. Young, S. Hao, P.G. Martinsson

Note: *The work described in this chapter was carried out in collaboration with Patrick Young and Professor Per-Gunnar Martinsson of the University of Colorado. It appeared in the Journal of Computational Physics (231(11), pages 4142-4159, 2012) under the title: "A high-order Nyström discretization scheme for Boundary Integral Equations Defined on Rotationally Symmetric Surfaces "*

Abstract: A scheme for rapidly and accurately computing solutions to boundary integral equations (BIEs) on rotationally symmetric surfaces in $\mathbb{R}^3$ is presented. The scheme uses the Fourier transform to reduce the original BIE defined on a surface to a sequence of BIEs defined on a generating curve for the surface. It can handle loads that are not necessarily rotationally symmetric. Nyström discretization is used to discretize the BIEs on the generating curve. The quadrature used is a high-order Gaussian rule that is modified near the diagonal to retain high-order accuracy for singular kernels. The reduction in dimensionality, along with the use of high-order accurate quadratures, leads to small linear systems that can be inverted directly via, *e.g.*, Gaussian elimination. This makes the scheme particularly fast in environments involving multiple right hand sides. It is demonstrated that for BIEs associated with the Laplace and Helmholtz equations, the kernel in the reduced equations can be evaluated very rapidly by exploiting recursion relations for Legendre functions. Numerical examples illustrate the performance of the scheme; in particular, it is demonstrated that for a BIE associated with Laplace's equation on a surface discretized using 320 800 points, the set-up phase of the algorithm takes 1 minute on a standard laptop, and then solves can be executed in 0.5 seconds.

4.1 Introduction

The premise of the paper is that it is much easier to solve a Boundary Integral Equation (BIE) defined on a curve in $\mathbb{R}^2$ than one defined on a surface in $\mathbb{R}^3$. With the development of high order accurate Nyström discretization techniques [64, 2, 67, 11, 50], it has become possible to attain close to double precision accuracy in 2D using only a very moderate number of degrees of freedom. This opens up the possibility of solving a BIE on a rotationally symmetric surface with the same efficiency since such an equation can in principle be written as a sequence of BIEs defined on a generating curve. However, there is a practical obstacle: The kernels in the BIEs on the generating curve are given via Fourier integrals that cannot be evaluated analytically. The principal contribution of the present paper is to describe a set of fast methods for constructing approximations to these kernels.

4.1.1 Problem formulation

This paper presents a numerical technique for solving boundary integral equations (BIEs) defined on axisymmetric surfaces in $\mathbb{R}^3$. Specifically, we consider second kind Fredholm equations of the form

$$\sigma(\boldsymbol{x}) + \int_{\Gamma} k(\boldsymbol{x},\boldsymbol{x}')\,\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}') = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma, \tag{1}$$

under two assumptions: First, that Γ is a surface in $\mathbb{R}^3$ obtained by rotating a curve γ about an axis. Second, that the kernel k is invariant under rotation about the symmetry axis in the sense that

$$k(\boldsymbol{x},\boldsymbol{x}') = k(\theta-\theta', r, z, r', z'), \tag{2}$$

where $(r,\, z,\, \theta)$ and $(r',\, z',\, \theta')$ are cylindrical coordinates for $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively,

$$\boldsymbol{x} = (r\,\cos\theta,\, r\,\sin\theta,\, z), \tag{3}$$

$$\boldsymbol{x}' = (r'\,\cos\theta',\, r'\,\sin\theta',\, z'), \tag{4}$$

see Figure 5.2. Under these assumptions, the equation (1), which is defined on the two-dimensional surface Γ, can via a Fourier transform in the azimuthal variable be recast as a sequence of equations defined on the one-dimensional curve γ. To be precise, letting σ_n, f_n, and k_n denote the Fourier coefficients of σ, f, and k, respectively (so that (11), (12), and (13) hold), the equation (1) is equivalent to the sequence of equations

$$\sigma_n(r,z) + \sqrt{2\pi}\int_\gamma k_n(r,z,r',z')\,\sigma_n(r',z')\,r'\,dl(r',z') = f_n(r,z), \quad (r,z)\in\gamma,\ \ n\in\mathbb{Z}. \tag{5}$$

Whenever f can be represented with a moderate number of Fourier modes, the formula (5) provides an efficient technique for computing the corresponding modes of σ. The conversion of (1) to (5) appears in, *e.g.*, [86], and is described in detail in Section 4.2. Note that the conversion procedure does *not* require the data function f to be rotationally symmetric.

4.1.2 Applications and prior work

Equations like (1) arise in many areas of mathematical physics and engineering, commonly as reformulations of elliptic partial differential equations. Advantages of a BIE approach include a reduction in dimensionality, often a radical improvement in the conditioning of the mathematical equation to be solved, a natural way of handling problems defined on exterior domains, and a relative ease in implementing high-order discretization schemes, see, *e.g.*, [3].

The observation that BIEs on rotationally symmetric surfaces can conveniently be solved by recasting them as a sequence of BIEs on a generating curve has previously been exploited in the context of stress analysis [5], scattering [30, 73, 92, 98, 99], and potential theory [45, 85, 86, 91]. Most of these approaches have relied on collocation or Galerkin discretizations and have generally used low-order accurate discretizations.

4.1.3 Kernel evaluations

A complication of the axisymmetric formulation is the need to determine the kernels k_n in (5). Each kernel k_n is defined as a Fourier integral of the original kernel function k in the azimuthal variable θ in (2), cf. (8), that cannot be evaluated analytically, and would be too expensive to approximate via standard quadrature techniques. The FFT can be used to a accelerate the computation in certain regimes. For many points, however, the function which is to be transformed is sharply peaked, and the FFT would at these points yield inaccurate results.

4.1.4 Principal contributions of present work

This paper resolves the difficulty of computing the kernel functions k_n described in Section 4.1.3. For the case of kernels associated with the Laplace equation, it provides analytic recursion relations that are valid precisely in the regions where the FFT loses accuracy. The kernels associated with the Helmholtz equation can then be obtained via a perturbation technique.

The paper also describes a high-order Nyström discretization of the BIEs (5) that provides far higher accuracy and speed than previously published methods. The discretization scheme converges fast enough that for simple generating curves, a relative accuracy of 10^{-10} is obtained using as few as a hundred points, cf. Section 5.6. The rapid convergence of the discretization leads to linear systems of small size that can be solved **directly** via, *e.g.*, Gaussian elimination, making the algorithm particularly effective in environments involving multiple right hand sides or when the linear system is challenging for iterative solvers (as happens for many scattering problems).

Finally, the efficient techniques for evaluating the fundamental solutions to the Laplace and Helmholtz equations in an axisymmetric environment have applications beyond solving boundary integral equations, for details see Section 4.7.

4.1.5 Asymptotic costs

To describe the asymptotic complexity of the method, we let N_{tot} denote the total number of discretization points, and assume that as N_{tot} grows, the number of Fourier modes required to resolve the solution scales proportionate to the number of discretization points required along the generating curve γ. Then the asymptotic cost of solving (1) for a single right-hand side f is $O(N_{\text{tot}}^2)$. If additional right hand sides are given, any subsequent solve requires only $O(N_{\text{tot}}^{3/2})$ operations.

Numerical experiments presented in Section 5.6 indicate that the constants of proportionality in the two estimates are moderate. For instance, in a simulation with $N_{\text{tot}} = 320\,800$, the scheme requires 1 minute for the first solve, and 0.49 seconds for each additional right hand side (on a standard laptop).

Observe that since a high-order discretization scheme is used, even complicated geometries can be resolved to high accuracy with a moderate number N_{tot} of points.

4.1.6 Outline

Section 4.2 provides details on the conversion of a BIE on a rotationally symmetric surface to a sequence of BIEs on a generating curve. Section 4.3 describes a high order methods for discretizing a BIE on a curve. Section 4.4 summarizes the algorithm and estimates its computational costs. Section 4.5 describes how to rapidly evaluate the kernels associated with the Laplace equation, and then Section 4.6 deals with the Helmholtz case. Section 4.7 describes other applications of the kernel evaluation techniques. Section 5.6 illustrates the performance of the proposed method via numerical experiments.

4.2 Fourier Representation of BIE

Consider the BIE (1) under the assumptions on rotational symmetry stated in Section 4.1.1 (i.e. Γ is a rotationally symmetric surface generated by a curve γ and that k is a rotationally symmetric kernel). Cylindrical coordinates (r, z, θ) are introduced as specified in (3). We write $\Gamma = \gamma \times \mathbb{T}$ where $\mathbb{T}$ is the one-dimensional torus, usually parameterized by $\theta \in (-\pi,\, \pi]$.

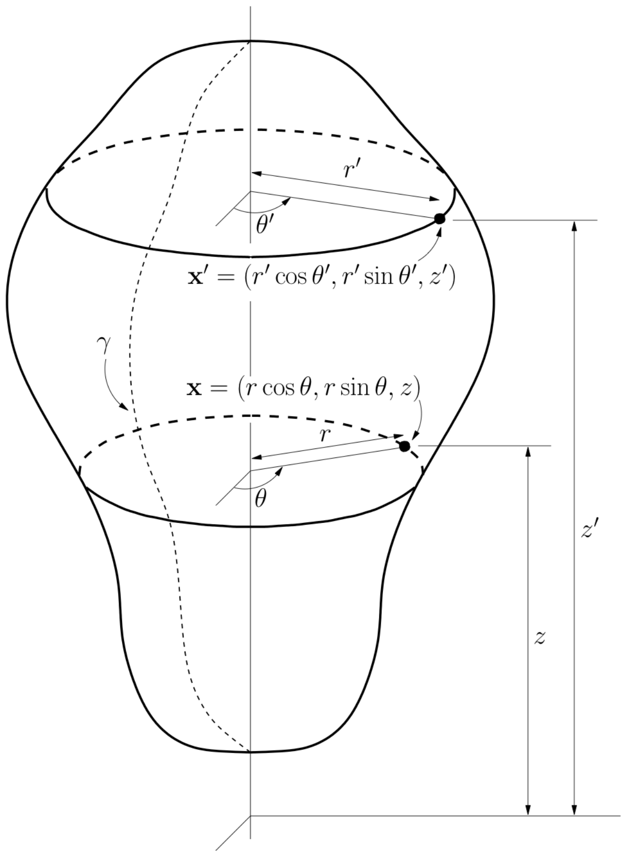

r'
θ'
$\mathbf{x}' = (r'\cos\theta', r'\sin\theta', z')$
γ
$\mathbf{x} = (r\cos\theta, r\sin\theta, z)$
r
θ
z'
z

Figure 4.1: The axisymmetric domain Γ generated by the curve γ.

4.2.1 Separation of Variables

We define for $n \in \mathbb{Z}$ the functions f_n, σ_n, and k_n via

$$f_n(r,z) = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, f(\theta,r,z)\, d\theta, \tag{6}$$

$$\sigma_n(r,z) = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, \sigma(\theta,r,z)\, d\theta, \tag{7}$$

$$k_n(r,z,r',z') = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, k(\theta,r,z,r',z')\, d\theta. \tag{8}$$

Formulas (6), (7), and (8) define f_n, σ_n, and k_n as the coefficients in the Fourier series of the functions f, σ, and k about the azimuthal variable,

$$f(\boldsymbol{x}) = \sum_{n\in\mathbb{Z}} \frac{e^{in\theta}}{\sqrt{2\pi}}\, f_n(r,z), \tag{9}$$

$$\sigma(\boldsymbol{x}) = \sum_{n\in\mathbb{Z}} \frac{e^{in\theta}}{\sqrt{2\pi}}\, \sigma_n(r,z), \tag{10}$$

$$k(\boldsymbol{x},\boldsymbol{x}') = k(\theta-\theta',r,z,r',z') = \sum_{n\in\mathbb{Z}} \frac{e^{in(\theta-\theta')}}{\sqrt{2\pi}}\, k_n(r,z,r',z'). \tag{11}$$

To determine the Fourier representation of (1), we multiply the equation by $e^{-in\theta}/\sqrt{2\pi}$ and integrate θ over $\mathbb{T}$. Equation (1) can then be said to be equivalent to the sequence of equations

$$\sigma_n(r,z) + \int_{\gamma\times\mathbb{T}} \left[\int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, k(\boldsymbol{x},\boldsymbol{x}')\, d\theta\right] \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') = f_n(r,z), \qquad n\in\mathbb{Z}. \tag{12}$$

Invoking (13), we evaluate the bracketed factor in (12) as

$$\begin{aligned}\int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, k(\boldsymbol{x},\boldsymbol{x}')\, d\theta &= \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}}\, k(\theta-\theta',r,z,r',z')\, d\theta \\ &= e^{-in\theta'} \int_{\mathbb{T}} \frac{e^{-in(\theta-\theta')}}{\sqrt{2\pi}}\, k(\theta-\theta',r,z,r',z')\, d\theta = e^{-in\theta'}\, k_n(r,z,r',z'). \end{aligned} \tag{13}$$

Inserting (13) into (12) and executing the integration of θ' over $\mathbb{T}$, we find that (1) is equivalent to the sequence of equations

$$\sigma_n(r,z) + \sqrt{2\pi}\int_{\gamma} k_n(r,z,r',z')\, \sigma_n(r',z')\, r'\, dl(r',z') = f_n(r,z), \quad n\in\mathbb{Z}. \tag{14}$$

For future reference, we define for $n\in\mathbb{Z}$ the boundary integral operators $\mathcal{K}_n$ via

$$[\mathcal{K}_n\, \sigma_n](r,z) = \sqrt{2\pi}\int_{\gamma} k_n(r,z,r',z')\, \sigma_n(r',z')\, r'\, dl(r',z'). \tag{15}$$

Then equation (14) can be written

$$\big(I + \mathcal{K}_n\big)\,\sigma_n = f_n, \qquad n \in \mathbb{Z}. \tag{16}$$

When each operator $I + \mathcal{K}_n$ is continuously invertible, we write the solution of (1) as

$$\sigma(r, z, \theta) = \sum_{n\in\mathbb{Z}} \frac{e^{in\theta}}{\sqrt{2\pi}} [(I + \mathcal{K}_n)^{-1} f_n](r, z). \tag{17}$$

4.2.2 Truncation of the Fourier series

When evaluating the solution operator (9) in practice, we will choose a truncation parameter N, and evaluate only the lowest $2N + 1$ Fourier modes. If N is chosen so that the given function f is well-represented by its lowest $2N + 1$ Fourier modes, then in typical environments the solution obtained by truncating the sum (9) will also be accurate. To substantiate this claim, suppose that ε is a given tolerance, and that N has been chosen so that

$$||f - \sum_{n=-N}^{N} \frac{e^{in\theta}}{\sqrt{2\pi}} f_n|| \leq \varepsilon, \tag{18}$$

We define an approximate solution via

$$\sigma_{\text{approx}} = \sum_{n=-N}^{N} \frac{e^{in\theta}}{\sqrt{2\pi}} (I + \mathcal{K}_n)^{-1} f_n. \tag{19}$$

From Parseval's identity, we then find that the error in the solution satisfies

$$\begin{aligned} ||\sigma - \sigma_{\text{approx}}||^2 &= \sum_{|n|>N} ||(I + \mathcal{K}_n)^{-1} f_n||^2 \leq \sum_{|n|>N} ||(I + \mathcal{K}_n)^{-1}||^2\, ||f_n||^2 \\ &\leq \left(\max_{|n|>N} ||(I + \mathcal{K}_n)^{-1}||^2\right) \sum_{|n|>N} ||f_n||^2 \leq \left(\max_{|n|>N} ||(I + \mathcal{K}_n)^{-1}||^2\right) \varepsilon^2. \end{aligned}$$

It is typically the case that the kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ has enough smoothness that the Fourier modes $k_n(r, z, r', z')$ decay as $n \to \infty$. Then $||\mathcal{K}_n|| \to 0$ as $n \to \infty$ and $||(I + \mathcal{K}_n)^{-1}|| \to 1$. Thus, an accurate approximation of f leads to an approximation in σ that is of the same order of accuracy. Figure 4.4 illustrates how fast this convergence is for Laplace's equation (note that in the case illustrated, the original equation is $\frac{1}{2}\sigma + \mathcal{K}\sigma = f$, and it is shown that $||(\frac{1}{2}I + \mathcal{K}_n)^{-1}|| \to 1/2$).

4.3 Nyström discretization of BIEs on the generating curve

We discretize the BIEs (5) defined on the generating curve γ using a Nyström scheme. In describing the scheme, we keep the formulas uncluttered by discussing a generic integral equation

$$\sigma(\boldsymbol{x}) + \int_\gamma k(\boldsymbol{x},\boldsymbol{x}')\,\sigma(\boldsymbol{x}')\,dl(\boldsymbol{x}') = f(\boldsymbol{x}), \qquad \boldsymbol{x}\in\gamma,$$

where γ is a simple smooth curve in the plane, and k is a weakly singular kernel function.

4.3.1 Quadrature nodes

Consider a quadrature rule on γ with nodes $\{\boldsymbol{x}_i\}_{i=1}^I \subset \gamma$ and weights $\{w_i\}_{i=1}^I$. In other words, for a sufficiently smooth function φ on γ,

$$\int_\gamma \varphi(\boldsymbol{x})\,dl(\boldsymbol{x}) \approx \sum_{i=1}^I \varphi(\boldsymbol{x}_i)\,w_i. \tag{20}$$

For the experiments in this paper, we use a composite Gaussian rule with 10 points per panel. Such a rule admits for local refinement, and can easily be modified to accommodate contours with corners that are only piece-wise smooth.

4.3.2 A simplistic Nyström scheme

The Nyström discretization of (20) corresponding to a quadrature with nodes $\{\boldsymbol{x}_i\}_{i=1}^I$ takes the form

$$\sigma_i + \sum_{j=1}^I a_{i,j}\,\sigma_j = f(\boldsymbol{x}_i), \qquad i = 1,\,2,\,3,\,\ldots,\,I, \tag{21}$$

where $\{a_{i,j}\}_{i,j=1}^I$ are coefficients such that

$$\int_\gamma k(\boldsymbol{x}_i,\boldsymbol{x}')\,\sigma(\boldsymbol{x}')\,dl(\boldsymbol{x}') \approx \sum_{j=1}^I a_{i,j}\,\sigma(\boldsymbol{x}_j), \qquad i = 1,\,2,\,3,\,\ldots,\,I. \tag{22}$$

The solution of (9) is a vector $\boldsymbol{\sigma} = [\sigma_i]_{i=1}^I$ such that each σ_i is an approximation to $\sigma(\boldsymbol{x}_i)$.

A simplistic way to construct coefficients $a_{i,j}$ so that (22) holds is to simply apply the rule (20) to each function $\boldsymbol{x}' \mapsto k(\boldsymbol{x}_i,\boldsymbol{x}')\,\sigma(\boldsymbol{x}')$ whence

$$a_{i,j} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)\,w_j. \tag{23}$$

This generally results in low accuracy since the kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ has a singularity at the diagonal. However, the formula (23) has the great advantage that constructing each $a_{i,j}$ costs no more than a kernel evaluation; we seek to preserve this property for as many elements as possible.

4.3.3 High-order accurate Nyström discretization

It is possible to construct a high-order discretization that preserves the simple formula (23) for the vast majority of coefficients $a_{i,j}$ [50]. The only ones that need to be modified are those for which the target point $\boldsymbol{x}_i$ is *near*[1] to the panel τ holding $\boldsymbol{x}_j$. In this case, $a_{i,j}$ is conceptually constructed as follows: First map the pointwise values $\{\sigma_j\}_{\boldsymbol{x}_j \in \tau}$ to a polynomial interpolant on τ, then integrate this polynomial against the singular kernel k using the quadratures of [67]. Operationally, the end result is that $a_{i,j}$ is given by

$$a_{i,j} = \begin{cases} \sum_{p=1}^{m} k(\boldsymbol{x}_i, \boldsymbol{y}_{i,j,p})\, v_{i,j,p} & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are near,} \\ k(\boldsymbol{x}_i, \boldsymbol{x}_j)\, w_j & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are not near.} \end{cases} \tag{24}$$

In (24), m is a small integer (roughly equal to the order of the Gaussian quadrature), the numbers $v_{i,j,p}$ are coefficients that depend on γ but not on k, and $\boldsymbol{y}_{i,j,p}$ are points on γ located in the same panel as $\boldsymbol{x}_j$ (in fact, $\boldsymbol{y}_{i,j,p} = \boldsymbol{y}_{i,j',p}$ when j and j' belong to the same panel, so the number of kernel evaluations required is less than it seems). For details, see [50].

4.4 The full algorithm

4.4.1 Overview

At this point, we have shown how to convert a BIE defined on an axisymmetric surface in $\mathbb{R}^3$ to a sequence of equations defined on a curve in $\mathbb{R}^2$ (Section 4.2), and then how to discretize each of these reduced equations (Section 4.3). Putting the components together, we obtain the following algorithm for solving (1) to within some preset tolerance ε:

[1] To be precise, we say that $\boldsymbol{x}_i$ is *near* a panel if is inside a circle of two times the radius of the smallest circle enclosing the source panel.

i) Given f, determine a truncation parameter N such that $||f - \sum_{n=-N}^{N} \frac{e^{in\theta}}{\sqrt{2\pi}} f_n|| \leq \varepsilon$.

ii) Fix a quadrature rule for γ with nodes $\{(r_i, z_i)\}_{i=1}^{I} \subset \gamma$ and form for each Fourier mode $n = -N,\ -N+1,\ -N+2,\ \ldots,\ N$ the corresponding Nyström discretization as described in Section 4.3. The number of nodes I must be picked to meet the computational tolerance ε. Denote the resulting coefficient matrices $\{\mathsf{A}^{(n)}\}_{n=-N}^{N}$.

iii) Evaluate via the FFT the terms $\{f_n(r_i, z_i)\}_{n=-N}^{N}$ (as defined by (6)) for $i = 1,\ 2,\ 3,\ \ldots,\ I$.

iv) Solve the equation $(\mathsf{I} + \mathsf{A}^{(n)})\,\sigma_n = f_n$ for $n = -N,\ -N+1,\ -N+2,\ \ldots,\ N$.

v) Construct σ_{approx} using formula (19) evaluated via the FFT.

The construction of the matrices $\mathsf{A}^{(n)}$ in Step ii can be accelerated using the FFT (as described in Section 4.4.2), but even with such acceleration, it is typically by a wide margin the most expensive part of the algorithm. However, this step needs to be performed only once for any given geometry. The method therefore becomes particularly efficient when (1) needs to be solved for a sequence of right-hand sides. In this case, it may be worth the cost to pre-compute the inverse (or LU-factorization) of each matrix $\mathsf{I} + \mathsf{A}^{(n)}$.

4.4.2 Cost of computing the coefficient matrices

For each of the $2N+1$ Fourier modes, we need to construct an $I \times I$ matrix $\mathsf{A}^{(n)}$ with entries $a_{i,j}^{(n)}$. These entries are derived from the kernel functions k_n defined by (8). Note that whenever $(r, z) \neq (r', z')$, the function $\theta \mapsto k(\theta, r, z, r', z')$ is C^∞, but that as $(r', z') \to (r, z)$ it develops a progressively sharper peak around $\theta = 0$.

For two nodes (r_i, z_i) and (r_j, z_j) that are "not near" (in the sense defined in Section 4.3.3) the matrix entries are given by the formula

$$a_{i,j}^{(n)} = k_n(r_i, z_i, r_j, z_j)\, w_j \tag{25}$$

where k_n is given by (8). Using the FFT, all $2N+1$ entries can be evaluated at once in $O(N \log N)$ operations. The FFT implicitly evaluates the integrals (8) via a trapezoidal rule which is highly accurate since the points (r_i, z_i) and (r_j, z_j) are well-separated on the curve γ.

For two nodes (r_i, z_i) and (r_j, z_j) that are not well-separated on γ, evaluating $a_{i,j}^{(n)}$ is dicier. The first complication is that we must now use the corrected formula, cf. (24),

$$a_{i,j}^{(n)} = \sum_{p=1}^{m} k_n(r_i, z_i, r_{i,j,p}, z_{i,j,p})\, v_{i,j,p}. \tag{26}$$

The second complication is that the FFT acceleration for computing the kernels $\{k_n\}_{n=-N}^{N}$ jointly no longer works since the integrand in (8) is too peaked for the simplistic trapezoidal rule implicit in the FFT. Fortunately, it turns out that for the kernels we are most interested in (the single and double layer kernels associated with the Laplace and Helmholtz equations), the sequence $\{k_n\}_{n=-N}^{N}$ can be evaluated very efficiently via certain recurrence relations as described in Sections 4.5 and 4.6 (for the Laplace and Helmholtz equations, respectively). Happily, the analytic formulas are stable precisely in the region where the FFT becomes inaccurate.

4.4.3 Computational Costs

The asymptotic cost of the algorithm described in Section 4.4.1 will be expressed in terms of the number N of Fourier modes required, and the number I of discretization points required along γ. The total cost can be split into three components:

(1) *Cost of forming the matrices* $\{\mathsf{A}^{(n)}\}_{n=-N}^{N}$*:* We need to form $2N+1$ matrices, each of size $I \times I$. For $O(I^2)$ entries in each matrix, the formula (25) applies and using the FFT, all $2N+1$ entries can be computed at once at cost $O(N \log N)$. For $O(I)$ entries close to the diagonal, the formula (26) applies, and all the $2N+1$ entries can be computed at once at cost $O(N)$ using the recursion relations in Sections 4.5 and 4.6. The total cost of this step is therefore $O(I^2 N \log N)$.

(2) *Cost of transforming functions from physical space to Fourier space and back:* The boundary data f must be decomposed into Fourier modes $\{f_n\}_{n=-N}^{N}$, and after the linear systems $(\mathsf{I}+\mathsf{A}^{(n)})\sigma_n = f_n$ have been solved, the Fourier modes $\{\sigma_n\}_{n=-N}^{N}$ must be transformed back to physical space. The asymptotic cost is $O(IN\log(N))$.

(3) *Cost of solving the linear systems* $(\mathsf{I}+\mathsf{A}^{(n)})\,\sigma_n = f_n$*:* Using a direct solver such as Gaussian elimination, the asymptotic cost is $O(I^3N)$.

We make some practical observations:

- The cost of executing FFTs is minimal and is dwarfed by the remaining costs.
- The scheme is highly efficient in situations where the same equation needs to be solved for a sequence of different right hand sides. In this situation, one factors the matrices $\mathsf{I}+\mathsf{A}^{(n)}$ once at cost $O(I^3N)$, and then the cost of processing an additional right hand side is only $O(I^2N + IN\log N)$ with a very small constant of proportionality.
- To elucidate the computational costs, let us express them in terms of the total number of discretization points N_{tot} under the simplifying assumption that $I \sim N$. Since $N_{\text{tot}} = IN$, we find $I \sim N_{\text{tot}}^{1/2}$ and $N \sim N_{\text{tot}}^{1/2}$. Then:

Cost of setting up linear systems:	$O(N_{\text{tot}}^{3/2}\log N_{\text{tot}})$
Cost of the first solve:	$O(N_{\text{tot}}^{2})$
Cost of subsequent solves:	$O(N_{\text{tot}}^{3/2})$

  We observe that even though the asymptotic cost of forming the linear systems is less than the cost of factoring the matrices, the set-up phase tends to dominate unless N_{tot} is large. Moreover, the $O(N_{\text{tot}}^{3/2})$ cost of the subsequent solves has a very small constant of proportionality.

- The high order discretization employed achieves high accuracy with a small number of points. In practical terms, this means that despite the $O(N_{\text{tot}}^2)$ scaling, the scheme is very fast even for moderately complicated geometries.

- The system matrices $\mathsf{I} + \mathsf{A}^{(n)}$ often have internal structure that allow them to be inverted using "fast methods" such as, *e.g.*, those in [76]. The cost of inversion and application can in fact be accelerated to near optimal complexity.

4.5 Accelerations for the Single and Double Layer Kernels Associated with Laplace's Equation

This section describes an efficient technique based on recursion relations for evaluating the kernel k_n, cf. (8), when k is either the single or double layer kernel associated with Laplace's equation.

4.5.1 The Double Layer Kernels of Laplace's Equation

Let $D \subset \mathbb{R}^3$ be a bounded domain whose boundary is given by a smooth surface Γ, let $E = \bar{D}^{\text{c}}$ denote the domain exterior to D, and let $\boldsymbol{n}$ be the outward unit normal to D. Consider the interior and exterior Dirichlet problems of potential theory [44],

$$\Delta u = 0 \ \text{ in } \ D, \quad u = f \ \text{ on } \ \Gamma, \qquad \text{(interior Dirichlet problem)} \tag{27}$$

$$\Delta u = 0 \ \text{ in } \ E, \quad u = f \ \text{ on } \ \Gamma. \qquad \text{(exterior Dirichlet problem)} \tag{28}$$

The solutions to (27) and (28) can be written in the respective forms

$$u(\boldsymbol{x}) = \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{x}') \cdot (\boldsymbol{x} - \boldsymbol{x}')}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|^3} \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in D, \tag{29}$$

$$u(\boldsymbol{x}) = \int_\Gamma \left(-\frac{\boldsymbol{n}(\boldsymbol{x}') \cdot (\boldsymbol{x} - \boldsymbol{x}')}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|^3} + \frac{1}{4\pi|\boldsymbol{x} - \boldsymbol{x}_0|} \right) \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in E, \tag{30}$$

where σ is a boundary charge distribution that can be determined using the boundary conditions. The point $\boldsymbol{x}_0$ can be placed at any suitable location in D. The resulting equations are

$$-\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{x}')\cdot(\boldsymbol{x}-\boldsymbol{x}')}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|^3}\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}') = f(\boldsymbol{x}), \tag{31}$$

$$-\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \left(-\frac{\boldsymbol{n}(\boldsymbol{x}')\cdot(\boldsymbol{x}-\boldsymbol{x}')}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|^3} + \frac{1}{4\pi|\boldsymbol{x}-\boldsymbol{x}_0|}\right)\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}') = f(\boldsymbol{x}), \tag{32}$$

where $\boldsymbol{x} \in \Gamma$ in (31) and (32).

Remark 2 There are other integral formulations for the solution to Laplace's equation. The double layer formulation presented here is a good choice in that it provides an integral operator that leads to well conditioned linear systems. However, the methodology of this chapter is equally applicable to single-layer formulations that lead to first kind Fredholm BIEs.

4.5.2 Separation of Variables

Using the procedure given in Section 4.2, if $\Gamma = \gamma \times \mathbb{T}$, then (27) and (28) can be recast as a series of BIEs defined along γ. We express $\boldsymbol{n}$ in cylindrical coordinates as

$$\boldsymbol{n}(\boldsymbol{x}') = (n_{r'}\cos\theta', n_{r'}\sin\theta', n_{z'}).$$

Further,

$$\begin{aligned}|\boldsymbol{x}-\boldsymbol{x}'|^2 &= (r\cos\theta - r'\cos\theta')^2 + (r\sin\theta - r'\sin\theta')^2 + (z-z')^2\\ &= r^2 + (r')^2 - 2rr'(\sin\theta\sin\theta' + \cos\theta\cos\theta') + (z-z')^2\\ &= r^2 + (r')^2 - 2rr'\cos(\theta-\theta') + (z-z')^2\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{n}(\boldsymbol{x}')\cdot(\boldsymbol{x}-\boldsymbol{x}') &= (n_{r'}\cos\theta', n_{r'}\sin\theta', n_{z'})\cdot(r\cos\theta - r'\cos\theta', r\sin\theta - r'\sin\theta', z-z')\\ &= n_{r'}r(\sin\theta\sin\theta' + \cos\theta\cos\theta') - n_{r'}r' + n_{z'}(z-z')\\ &= n_{r'}(r\cos(\theta-\theta') - r') + n_{z'}(z-z').\end{aligned}$$

Then for a point $\boldsymbol{x}' \in \Gamma$, the kernel of the internal Dirichlet problem can be expanded as

$$\frac{\boldsymbol{n}(\boldsymbol{x}') \cdot (\boldsymbol{x} - \boldsymbol{x}')}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|^3} = \frac{1}{\sqrt{2\pi}} \sum_{n\in\mathbb{Z}} e^{in(\theta-\theta')} d_n^{(i)}(r, z, r', z'),$$

where

$$d_n^{(i)}(r, z, r', z') = \frac{1}{\sqrt{32\pi^3}} \int_{\mathbb{T}} e^{-in\theta} \left[\frac{n_{r'}(r\cos\theta - r') + n_{z'}(z - z')}{(r^2 + (r')^2 - 2rr'\cos\theta + (z - z')^2)^{3/2}} \right] d\theta.$$

Similarly, the kernel of the external Dirichlet problem can be written as

$$-\frac{\boldsymbol{n}(\boldsymbol{x}') \cdot (\boldsymbol{x} - \boldsymbol{x}')}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|^3} + \frac{1}{4\pi|\boldsymbol{x} - \boldsymbol{x}_0|} = \frac{1}{\sqrt{2\pi}} \sum_{n\in\mathbb{Z}} e^{in(\theta-\theta')} d_n^{(e)}(r, z, r', z'),$$

with

$$\begin{aligned} d_n^{(e)}(r, z, r', z') = \frac{1}{\sqrt{32\pi^3}} \int_{\mathbb{T}} e^{-in\theta} \Bigg(&-\frac{n_{r'}(r\cos\theta - r') + n_{z'}(z - z')}{(r^2 + (r')^2 - 2rr'\cos\theta + (z - z')^2)^{3/2}} + \\ &+ \frac{1}{(r^2 + r_0^2 - 2rr_0\cos\theta + (z - z_0)^2)^{1/2}} \Bigg) d\theta, \end{aligned}$$

where $\boldsymbol{x}_0$ has been written in cylindrical coordinates as $(r_0\cos(\theta_0), r_0\sin(\theta_0), z_0)$. With the expansions of the kernels available, the procedure described in Section 4.4 can be used to solve (31) and (32) by solving

$$\sigma_n(r, z) + \sqrt{2\pi} \int_\gamma d_n^{(i)}(r, r', z, z')\sigma_n(r', z')\, r'\, dl(r', z') = f_n(r, z) \tag{33}$$

and

$$\sigma_n(r, z) + \sqrt{2\pi} \int_\gamma d_n^{(e)}(r, r', z, z')\sigma_n(r', z')\, r'\, dl(r', z') = f_n(r, z), \tag{34}$$

respectively for $n = -N, -N+1, \dots, N$. Note that the kernels $d_n^{(i)}$ and $d_n^{(e)}$ contain a log-singularity as $(r', z') \to (r, z)$.

4.5.3 Evaluation of Kernels

The values of $d_n^{(i)}$ and $d_n^{(e)}$ for $n = -N, -N+1, \dots, N$ need to be computed efficiently and with high accuracy to construct the Nyström discretization of (33) and (34). Note that the integrands of $d_n^{(i)}$ and $d_n^{(e)}$ are real valued and even functions on the interval $[-\pi, \pi]$. Therefore, $d_n^{(i)}$ can be written as

$$d_n^{(i)}(r, z, r', z') = \frac{1}{\sqrt{32\pi^3}} \int_{\mathbb{T}} \left[\frac{n_{r'}(r\cos t - r') + n_{z'}(z - z')}{(r^2 + (r')^2 - 2rr'\cos t + (z - z')^2)^{3/2}} \right] \cos(nt)\, dt. \tag{35}$$

Note that $d_n^{(e)}$ can be written in a similar form.

This integrand is oscillatory and increasingly peaked at the origin as (r', z') approaches (r, z). As long as r' and r as well as z' and z are well separated, the integrand does not experience peaks near the origin, and as mentioned before, the FFT provides a fast and accurate way for calculating $d_n^{(i)}$ and $d_n^{(e)}$.

In regimes where the integrand is peaked, the FFT no longer provides a means of evaluating $d_n^{(i)}$ and $d_n^{(e)}$ with the desired accuracy. One possible solution to this issue is applying adaptive quadrature to fully resolve the peak. However, this must be done for each value of n required and becomes prohibitively expensive if N is large.

Fortunately, an analytical solution to (35) exists. As noted in [25], the single-layer kernel can be expanded with respect to the azimuthal variable as

$$\begin{aligned} s(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|} &= \frac{1}{4\pi(r^2 + (r')^2 - 2rr'\cos(\theta - \theta') + (z - z')^2)^{1/2}} \\ &= \frac{1}{\sqrt{2\pi}} \sum_{n\in\mathbb{Z}} e^{in(\theta-\theta')} s_n(r, z, r', z'), \end{aligned}$$

where

$$\begin{aligned} s_n(r, z, r', z') &= \frac{1}{\sqrt{32\pi^3}} \int_{\mathbb{T}} \frac{\cos(nt)}{(r^2 + (r')^2 - 2rr'\cos(t) + (z - z')^2)^{1/2}}\, dt \\ &= \frac{1}{\sqrt{8\pi^3 rr'}} \int_{\mathbb{T}} \frac{\cos(nt)}{\sqrt{8(\chi - \cos(t))}}\, dt \\ &= \frac{1}{\sqrt{8\pi^3 rr'}} \mathcal{Q}_{n-1/2}(\chi), \end{aligned}$$

$\mathcal{Q}_{n-1/2}$ is the half-integer degree Legendre function of the second kind, and

$$\chi = \frac{r^2 + (r')^2 + (z - z')^2}{2rr'}.$$

To find an analytical form for (35), first note that in cylindrical coordinates the double-layer kernel can be written in terms of the single-layer kernel,

$$\begin{aligned}\frac{\boldsymbol{n}(\boldsymbol{x}')\cdot(\boldsymbol{x}-\boldsymbol{x}')}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|^3} =& \frac{n_{r'}(r\cos(\theta-\theta')-r')+n_{z'}(z-z')}{4\pi(r^2+(r')^2-2rr'\cos(\theta-\theta')+(z-z')^2)^{3/2}}\\ =&\frac{1}{4\pi}\left[n_{r'}\frac{\partial}{\partial r'}\left(\frac{1}{(r^2+(r')^2-2rr'\cos(\theta-\theta')+(z-z')^2)^{1/2}}\right)+\right.\\ &\left.+\,n_{z'}\frac{\partial}{\partial z'}\left(\frac{1}{(r^2+(r')^2-2rr'\cos(\theta-\theta')+(z-z')^2)^{1/2}}\right)\right].\end{aligned}$$

The coefficients of the Fourier series expansion of the double-layer kernel are then given by $d_n^{(i)}$, which can be written using the previous equation as

$$\begin{aligned}d_n^{(i)}(r,z,r',z') =& n_{r'}\int_{\mathbb{T}}\frac{\partial}{\partial r'}\left(\frac{\cos(nt)}{(32\pi^3(r^2+(r')^2-2rr'\cos(t)+(z-z')^2))^{1/2}}\right)dt+\\ &+n_{z'}\int_{\mathbb{T}}\frac{\partial}{\partial z'}\left(\frac{\cos(nt)}{(32\pi^3(r^2+(r')^2-2rr'\cos(t)+(z-z')^2))^{1/2}}\right)dt\\ =&n_{r'}\frac{\partial}{\partial r'}\left(\frac{1}{\sqrt{8\pi^3rr'}}\mathcal{Q}_{n-1/2}(\chi)\right)+n_{z'}\frac{\partial}{\partial z'}\left(\frac{1}{\sqrt{8\pi^3rr'}}\mathcal{Q}_{n-1/2}(\chi)\right)\\ =&\frac{1}{\sqrt{8\pi^3rr'}}\left[n_{r'}\left(\frac{\partial\mathcal{Q}_{n-1/2}(\chi)}{\partial\chi}\frac{\partial\chi}{\partial r'}-\frac{\mathcal{Q}_{n-1/2}(\chi)}{2r'}\right)+n_{z'}\frac{\partial\mathcal{Q}_{n-1/2}(\chi)}{\partial\chi}\frac{\partial\chi}{\partial z'}\right].\end{aligned}$$

To utilize this form of $d_n^{(i)}$, set $\mu=\sqrt{\frac{2}{\chi+1}}$ and note that

$$\begin{aligned}&\frac{\partial\chi}{\partial r'}=\frac{(r')^2-r^2-(z-z')^2}{2r(r')^2},\\ &\frac{\partial\chi}{\partial z'}=\frac{z'-z}{rr'},\\ &\mathcal{Q}_{-1/2}(\chi)=\mu K(\mu),\\ &\mathcal{Q}_{1/2}(\chi)=\chi\mu K(\mu)-\sqrt{2(\chi+1)}E(\mu),\\ &\mathcal{Q}_{-n-1/2}(\chi)=\mathcal{Q}_{n-1/2}(\chi),\\ &\mathcal{Q}_{n-1/2}(\chi)=4\frac{n-1}{2n-1}\chi\mathcal{Q}_{n-3/2}(\chi)-\frac{2n-3}{2n-1}\mathcal{Q}_{n-5/2}(\chi),\\ &\frac{\partial\mathcal{Q}_{n-1/2}(\chi)}{\partial\chi}=\frac{2n-1}{2(\chi^2-1)}\left(\chi\mathcal{Q}_{n-1/2}-\mathcal{Q}_{n-3/2}\right),\end{aligned}$$

where K and E are the complete elliptic integrals of the first and second kinds, respectively. The first two relations follow immediately from the definition of χ and the relations for the Legendre functions of the second kind can be found in [1]. With these relations in hand, the calculation of $d_n^{(i)}$ for $n=-N,-N+1,\ldots,N$ can be done accurately and efficiently when r' and r as well as z' and z are in close proximity. The calculation of $d_n^{(e)}$ can be done analogously.

Remark 3 Note that the forward recursion relation for the Legendre functions $\mathcal{Q}_{n-1/2}(\chi)$ is unstable when $\chi > 1$. In practice, the instability is mild when χ is near 1 and the recursion relation can still be employed to accurately compute values in this regime. Additionally, if stability becomes an issue, Miller's algorithm [32] can be used to calculate the values of the Legendre functions using the backwards recursion relation, which is stable for $\chi > 1$.

4.6 Fast Kernel Evaluation for the Helmholtz Equation

Section 4.5 describes how to efficiently evaluate the kernels k_n as defined by (8) for kernels associated with Laplace's equation. This section generalizes these methods to a broad class of kernels that includes the single and double layer kernels associated with the Helmholtz equation.

4.6.1 Rapid Kernel Calculation via Convolution

Consider a kernel of the form

$$f(\boldsymbol{x}, \boldsymbol{x}') = s(\boldsymbol{x}, \boldsymbol{x}')\, g(\boldsymbol{x}, \boldsymbol{x}'), \tag{36}$$

where

$$s(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{4\pi|\boldsymbol{x} - \boldsymbol{x}'|}$$

is the single layer kernel of Laplace's equation and $g(\boldsymbol{x}, \boldsymbol{x}')$ is a smooth function for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^3$. Common examples of kernels that take this form include the fundamental solution of the Helmholtz equation and screened Coulomb (Yukawa) potentials.

Letting

$$\boldsymbol{x} = (r\,\cos\theta,\, r\,\sin\theta,\, z),$$
$$\boldsymbol{x}' = (r'\,\cos\theta',\, r'\,\sin\theta',\, z'),$$

we are interested in calculating the Fourier expansion of (36) in terms of the azimuthal variable. When $g(\boldsymbol{x}, \boldsymbol{x}') = 1$ (the Laplace kernel), we know how to rapidly compute these Fourier coefficients rapidly and efficiently. However, when g takes a nontrivial form, this is not generally true; there is no known analytical formula for calculating the Fourier coefficients of (36).

We will now describe an efficient technique for calculating the the Fourier coefficients of (36), when the function g is sufficiently smooth. For a fixed value of (r, z) and (r', z'), the functions s and g are periodic in the azimuthal variable over the interval $\mathbb{T}$. Dropping the dependence of s and g on (r, z) and (r', z') for notational clarity, we define $t = \theta - \theta' \in \mathbb{T}$ and the Fourier series expansions of s and g as

$$s(t) = \sum_{n\in\mathbb{Z}} \frac{e^{int}}{\sqrt{2\pi}} s_n, \tag{37}$$

$$g(t) = \sum_{n\in\mathbb{Z}} \frac{e^{int}}{\sqrt{2\pi}} g_n, \tag{38}$$

where

$$s_n = \int_{\mathbb{T}} \frac{e^{-int}}{\sqrt{2\pi}} s(t)\, dt, \tag{39}$$

$$g_n = \int_{\mathbb{T}} \frac{e^{-int}}{\sqrt{2\pi}} g(t)\, dt. \tag{40}$$

The values given by (39) can be calculated as described in Section 4.5, while the values given by (40) can be rapidly and accurately computed using the FFT.

Assuming the Fourier series defined by (37) and (38) are uniformly convergent, we find

$$f_n = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{T}} s(t)\, g(t)\, e^{-int}\, dt = \sum_{k\in\mathbb{Z}} s_k\, g_{n-k} = [s_k * g_k](n),$$

where $s_k * g_k$ is the discrete convolution of the sequences defined by (39) and (40).

In a practical setting, the Fourier series are truncated to finite length. Assuming that we have kept $-N, -N+1, \ldots, N$ terms, directly calculating the convolution would require $O(N^2)$ operations. Fortunately, this computation can be accelerated to $O(N \log N)$ operations by employing the discrete convolution theorem and the FFT [17].

Letting $\mathcal{D}$ denote the discrete Fourier transform (DFT), the discrete convolution theorem states that the convolution of two periodic sequences $\{a_n\}$ and $\{b_n\}$ is related by

$$\mathcal{D}\{a_n * b_n\}_k = \alpha A_k B_k,$$

where $\{A_n\} = \mathcal{D}\{a_n\}$, $\{B_n\} = \mathcal{D}\{b_n\}$, and α is a known constant depending upon the length of the periodic sequences and the definition of the DFT. Thus, we can rapidly calculate the convolution of two periodic sequences by taking the FFT of each sequence, computing the pointwise product of the result, and then applying the inverse FFT.

Of course, the sequences that we need to convolute are not periodic. Applying the discrete convolution to the sequences defined by (39) and (40) will not be exact, but the error incurred will be small assuming that the Fourier coefficients decay rapidly and that N is large enough. To see this, assume that

$$s(t) = \sum_{n=-N}^{N} \frac{e^{int}}{\sqrt{2\pi}} s_n,$$

$$g(t) = \sum_{n=-N}^{N} \frac{e^{int}}{\sqrt{2\pi}} g_n.$$

Then the exact Fourier representation of f can be found by taking the product of these two series, which will be of length $4N + 1$. As is well known, the coefficients of this product is given by the discrete convolution of the sequences containing the coefficients of the two series, and these sequences must first be padded with $2N$ zeros. Thus, we can effectively calculate the Fourier coefficients of the function given by (36) by calculating $2N + 1$ Fourier coefficients of s and g, padding these sequences with zeros, calculating the discrete convolution of these sequences, and truncating the resulting sequence. In practice, padding may not even be required if the Fourier coefficients of s and g decay sufficiently fast.

Note that the procedure described in this section is quite general. The azimuthal Fourier coefficients of many kernels that can be represented as the product of a singular function and a smooth function can be found, assuming that there is an accurate technique for determining the coefficients of the singular function.

4.6.2 Application to the Helmholtz Equation

In this section, we will apply the fast kernel calculation technique described in Section 4.6.1 to the exterior Dirichlet problem for the Helmholtz equation. Let $D \subset \mathbb{R}^3$ be a bounded domain whose boundary is given by a smooth surface Γ, let $E = \bar{D}^{\mathrm{c}}$ denote the domain exterior to D, and let $\boldsymbol{n}$ and be the outward unit normal to D. The partial differential equation representing this problem is given by

$$\Delta u + k^2 u = 0 \text{ in } E, \quad u = f \text{ on } \Gamma, \tag{41}$$

where $k > 0$ is the wavenumber, and u satisfies the Sommerfeld radiation condition

$$\lim_{r \to \infty} r \left(\frac{\partial u}{\partial r} - i\, k\, u \right) = 0, \tag{42}$$

where $r = |\boldsymbol{x}|$ and the limit holds uniformly in all directions $\boldsymbol{x}/|\boldsymbol{x}|$. Let the single and double layer potentials for the Helmholtz equation be given by

$$\phi(\boldsymbol{x}, \boldsymbol{x}') = \frac{e^{ik|\boldsymbol{x}-\boldsymbol{x}'|}}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|}, \qquad \text{(single layer)} \tag{43}$$

$$\frac{\partial \phi(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} = \frac{\boldsymbol{n}(\boldsymbol{x}') \cdot (\boldsymbol{x}-\boldsymbol{x}')}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|^3} \left[\left(1 - ik|\boldsymbol{x}-\boldsymbol{x}'|\right) e^{ik|\boldsymbol{x}-\boldsymbol{x}'|} \right]. \qquad \text{(double layer)} \tag{44}$$

The solution to (41) can be written in terms of the double layer potential,

$$u(\boldsymbol{x}) = \int_\Gamma \frac{\partial \phi(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in E,$$

where σ is a boundary charge density that can be determined using the boundary conditions. The resulting boundary integral equation is given by

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\partial \phi(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') = f(\boldsymbol{x}). \tag{45}$$

As is well known, (45) is not always uniquely solvable, even though (41) is uniquely solvable for all $k > 0$. A common solution to this is to represent the solution to (41) as a combined single and double layer potential,

$$u(\boldsymbol{x}) = \int_\Gamma \left(\frac{\partial \phi(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} - i\, \nu\, \phi(\boldsymbol{x}, \boldsymbol{x}') \right) \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in E,$$

where $\nu > 0$. We have freedom in choosing ν, see *e.g.*, [18, 72], for some analysis on this choice. The boundary integral equation we need to solve is

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \left(\frac{\partial \phi(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} - i\, \nu\, \phi(\boldsymbol{x}, \boldsymbol{x}') \right) \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') = f(\boldsymbol{x}). \tag{46}$$

4.7 Fast evaluation of fundamental solutions in cylindrical coordinates

The techniques for kernel evaluations described in Sections 4.4.2, 4.5, and 4.6 are useful not only for solving BIEs, but for solving the Laplace and Helmholtz equations in a variety of contexts where cylindrical coordinates are effective. To illustrate, observe that the free space equation

$$-\Delta u(\boldsymbol{x}) - k^2\, u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^3 \tag{47}$$

has the solution

$$u(\boldsymbol{x}) = \int_{\mathbb{R}^3} \phi^{(k)}(\boldsymbol{x}, \boldsymbol{x}')\, f(\boldsymbol{x}')\, dA(\boldsymbol{x}'). \qquad \boldsymbol{x} \in \mathbb{R}^3, \tag{48}$$

where $\phi^{(k)}$ is the fundamental solution

$$\phi^{(k)}(\boldsymbol{x}) = \frac{e^{ik|\boldsymbol{x}|}}{4\pi|\boldsymbol{x}|}.$$

In cylindrical coordinates, we write (48) as

$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} \frac{e^{in\theta}}{\sqrt{2\pi}} \int_H \phi_n^{(k)}(r, z, r', z') f_n(r', z') dA(r', z') \tag{49}$$

where $H = \{(r, z) \in \mathbb{R}^2 : r \geq 0\}$ is a half-plane, and where u_n, f_n, and $\phi_n^{(k)}$ are the Fourier coefficients defined by

$$u(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} \frac{e^{in\theta}}{\sqrt{2\pi}} u_n(r, z),$$

$$f(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} \frac{e^{in\theta}}{\sqrt{2\pi}} f_n(r, z),$$

$$\phi^{(k)}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{n=-\infty}^{\infty} \frac{e^{in(\theta-\theta')}}{\sqrt{2\pi}} \phi_n^{(k)}(r, z, r', z').$$

The kernel $\phi_n^{(k)}$ in (49) can be evaluated efficiently using the techniques of Sections 4.4.2, 4.5 and 4.6. If f has a rapidly convergent Fourier series, and if "fast" summation (e.g. the Fast Multipole Method) is used to evaluate the integrals in (49), then very efficient solvers result.

More generally, we observe that the equation (47) can be expressed

$$-\frac{\partial^2 u_n}{\partial^2 r} - \frac{1}{r}\frac{\partial u_n}{\partial r} - \frac{\partial^2 u_n}{\partial^2 z} + \left(\frac{n^2}{r^2} - k^2\right) u_n = f_n, \qquad n \in \mathbb{Z}. \tag{50}$$

and that the function $\phi_n^{(k)}$ is the Green's function of (50).

4.8 Numerical Results

This section describes several numerical experiments performed to assess the efficiency and accuracy of the numerical scheme outlined in Section 4.4.1. The geometries investigated are described in Figure 5.3. The generating curves were parameterized by arc length, and split into $N_{\rm P}$ panels of equal length. A 10-point Gaussian quadrature has been used along each panel, with the modified quadratures of [67] used to handle the integrable singularities in the kernel. The algorithm was implemented in FORTRAN, using BLAS, LAPACK, and the FFT library provided by Intel's MKL library. All numerical experiments in this section have been carried out on a Macbook Pro with a 2.4 GHz Intel Core 2 Duo and 4GB of RAM.

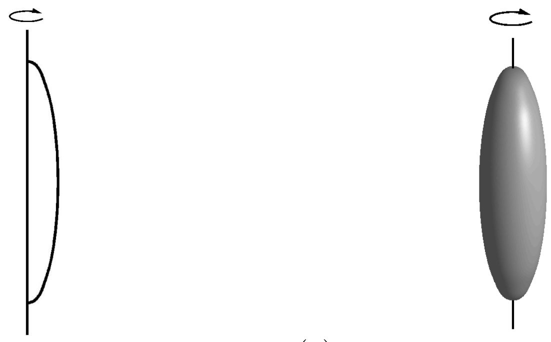
4.8.1 Laplace's equation

We solved Laplace's equation in the domain interior to the surfaces shown in Figure 5.3. The solution was represented via the double layer Ansatz (29) leading to the BIE (31). The kernels d_n defined via (35) were evaluated using the techniques described in Section 4.5.3. Tn this case $\mathsf{A}^{(n)} = \mathsf{A}^{(-n)}$, and so we need only to invert $N+1$ matrices. Further, the FFT used here is complex-valued, and a real-valued FFT would yield a significant decrease in computation time.

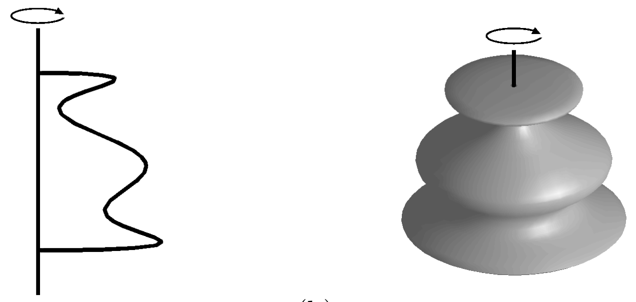
To investigate the speed of the proposed method, we solved a sequence of problems on the domain in Figure 5.3(a). The timing results are given in Table 4.1. The reported results include:

$N_{\rm P}$	the number of panels used to discretize the contour (each panel has $I/N_{\rm P}$ nodes)
N	the Fourier truncation parameter (we keep $2N+1$ modes)
$T_{\rm mat}$	time to construct the linear systems (utilizing the recursion relation)
$T_{\rm inv}$	time to invert the linear systems
$T_{\rm fft}$	time to Fourier transform the right hand side and the solution
$T_{\rm apply}$	time to apply the inverse to the right hand side

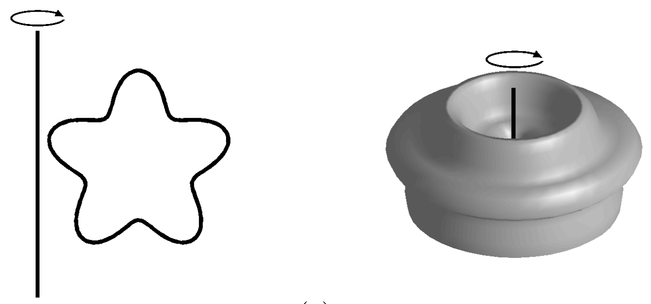
The most expensive component of the calculation is the kernel evaluation required to form the coefficient matrices $\mathsf{A}^{(n)}$. Table 4.2 compares the use of the recursion relation in evaluating the



(a)



(b)



(c)

Figure 4.2: Domains used in numerical examples. All items are rotated about the vertical axis. (a) An ellipse. (b) A wavy block. (c) A starfish torus.

kernel when it is near-singular to using an adaptive Gaussian quadrature. The efficiency of the recursion relation is evident.

Figure 4.3 plots the time to construct the linear systems as the number of degrees of freedom $N_{\text{tot}} = I(2N+1)$ increases, for the case when $I \approx 2N+1$. The estimated asymptotic costs given in this Figure match well with the estimates derived in Section 4.4.3. It is also clear that as N_{tot} grows, the cost of inversion will eventually dominate. We remark that the asymptotic scaling of this cost can be lowered by using fast techniques for the inversion of boundary integral operators,

N_{P}	$2N+1$	T_{mat}	T_{inv}	T_{fft}	T_{apply}
5	25	1.70E-02	1.42E-03	7.81E-05	3.83E-05
10	25	3.64E-02	6.15E-03	1.68E-04	2.66E-04
20	25	9.73E-02	3.52E-02	3.69E-04	2.10E-03
40	25	3.09E-01	2.35E-01	6.69E-04	4.82E-03
80	25	1.20E+00	1.88E+00	1.36E-03	2.85E-02
5	51	2.83E-02	3.02E-03	2.38E-04	1.13E-04
10	51	6.71E-02	1.23E-02	4.48E-04	6.58E-04
20	51	2.02E-01	7.42E-02	9.17E-04	2.63E-03
40	51	7.28E-01	4.94E-01	1.92E-03	1.03E-02
80	51	3.07E+00	3.73E+00	3.59E-03	6.17E-02
5	101	5.08E-02	5.48E-03	7.27E-04	2.38E-04
10	101	1.35E-01	2.27E-02	1.41E-03	1.33E-03
20	101	4.39E-01	1.32E-01	2.73E-03	4.60E-03
40	101	1.98E+00	1.04E+00	6.20E-03	2.14E-02
80	101	7.13E+00	7.04E+00	1.12E-02	1.12E-01
5	201	1.07E-01	1.06E-02	1.80E-03	6.36E-04
10	201	3.33E-01	4.96E-02	3.83E-03	2.79E-03
20	201	1.12E+00	2.73E-01	7.05E-03	9.46E-03
40	201	4.63E+00	1.88E+00	1.46E-02	4.15E-02
80	201	1.71E+01	1.41E+01	2.93E-02	2.15E-01
5	401	1.87E-01	2.15E-02	3.43E-03	1.48E-03
10	401	5.85E-01	9.51E-02	6.59E-03	5.05E-03
20	401	2.15E+00	5.42E-01	1.33E-02	1.91E-02
40	401	8.40E+00	3.71E+00	2.78E-02	7.98E-02
80	401	3.15E+01	2.83E+01	5.56E-02	4.34E-01

Table 4.1: Timing results in seconds performed for the domain given in Figure 5.3(a) for the interior Dirichlet problem.

but that little gain would be achieved for the problem sizes considered here.

We observe that the largest problem reported in Table 4.1 involves 320 800 degrees of freedom. The method requires 1 minute of pre-computation for this example, and is then capable of computing a solution u from a given data function f in 0.49 seconds.

To test the accuracy of the approach, we have solved a both interior and exterior Dirichlet problems on each of the domains given in Figure 5.3. Exact solutions were generated by placing a few random point charges outside of the domain where the solution was calculated. The solution was evaluated at points defined on a sphere encompassing (or interior to) the boundary. The errors

$2N+1$	Composite Quadrature	Recursion Relation
25	1.9	0.017
50	3.1	0.028
100	6.6	0.051
200	18.9	0.107

Table 4.2: Timing comparison in seconds for constructing the matrices $(\mathsf{I} + \mathsf{A}^{(n)})$ using composite Gaussian quadrature and the recursion relation described in Section 4.5.3 to evaluate k_n for diagonal and near diagonal blocks. The FFT is used to evaluate k_n at all other entries. $2N+1$ is the total number of Fourier modes used. 5 panels were used to discretize the boundary.

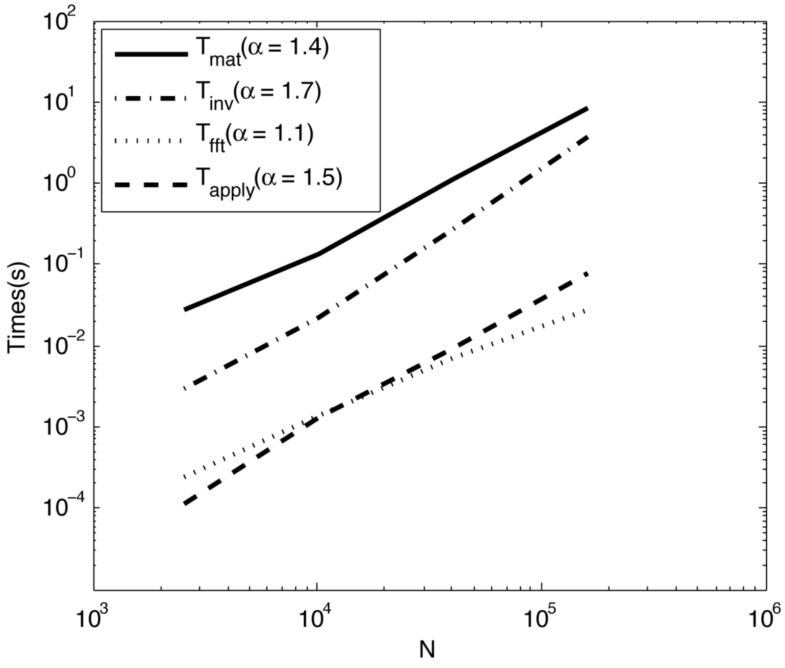


Figure 4.3: Timings of the algorithm as the number of degrees of freedom $N_{\text{tot}} = I(2N+1)$ increases. The timings reported here are for the case $I \approx 2N+1$. The numbers in parentheses provide estimates of the asymptotic complexity, *i.e.* the best fit to a curve $T = C\, N_{\text{tot}}^{\alpha}$.

reported in Tables 4.3-4.5 are relative errors measured in the l^{∞}-norm, $||u_{\epsilon} - u||_{\infty}/||u||_{\infty}$, where u is the exact potential and u_{ϵ} is the potential obtained from the numerical solution.

For all geometries, 10 digits of accuracy has been obtained from a discretization involving a relatively small number of degrees of freedom, due to the rapid convergence of the Gaussian quadrature. This is especially advantageous, as the most expensive component of the algorithm is the construction of the linear systems, the majority of the cost being directly related to the number of panels used. Further, the number of Fourier modes required to obtain 10 digits of accuracy is on the order of 100 modes. Although not investigated here, the discretization technique naturally lends itself to nonuniform refinement in the rz-plane, allowing one to resolve features of the generating curve that require finer resolution.

The number of correct digits obtained as the number of panels and number of Fourier modes increases eventually stalls. This is a result of a loss of precision in determining the kernels, as well as cancelation errors incurred when evaluating interactions between nearby points. This is especially prominent with the use of Gaussian quadratures, as points cluster near the ends of the panels. If more digits are required, high precision arithmetic can be employed in the setup phase of the algorithm.

N_{P}	$2N+1$				
-	25	51	101	201	401
5	3.9506E-04	4.6172E-04	4.6199E-04	4.6203E-04	4.6204E-04
10	1.3140E-05	1.1091E-08	4.8475E-09	4.8480E-09	4.8481E-09
20	1.7232E-05	7.7964E-09	4.7197E-12	4.7232E-12	4.7237E-12
40	2.7527E-05	2.7147E-08	2.8818E-14	5.7173E-14	5.7658E-14
80	2.118E-05	9.4821E-09	2.1529E-13	2.0392E-13	2.0356E-13

Table 4.3: Error in internal Dirichlet problem solved on domain (a) in Figure 5.3.

N_{P}	$2N+1$				
-	25	51	101	201	401
5	8.6992E-04	1.3615E-03	1.3620E-03	1.3621E-03	1.3621E-03
10	2.2610E-04	9.6399E-05	9.6751E-05	9.6751E-05	9.6752E-05
20	2.6291E-04	4.6053E-07	2.4794E-07	2.4794E-07	2.4794E-07
40	3.1714E-04	2.8922E-07	2.2875E-11	2.3601E-11	2.3605E-11
80	3.0404E-04	3.5955E-07	3.3708E-11	3.3138E-11	3.3150E-11

Table 4.4: Error in external Dirichlet problem solved on domain (b) in Figure 5.3.

Finally, we investigated the conditioning of the numerical procedure. Figure 4.4 shows the smallest and largest singular values of the matrices $\{\frac{1}{2}\mathsf{I} + \mathsf{A}^{(n)}\}_{n=-200}^{200}$ on the domain shown in

$N_{\rm P}$	$2N+1$				
-	25	51	101	201	401
5	4.3633E-04	7.9169E-05	7.8970E-05	7.8970E-05	7.8971E-05
10	3.9007E-04	6.8504E-07	2.0274E-08	2.0272E-08	2.0272E-08
20	3.8803E-04	6.4014E-07	3.2138E-11	3.1624E-11	3.1625E-11
40	3.8456E-04	6.4098E-07	6.5742E-12	3.4529E-12	3.4530E-12
80	3.9828E-04	6.4486E-07	6.8987E-12	3.1914E-12	3.1913E-12

Table 4.5: Error in external Dirichlet problem solved on domain (c) in Figure 5.3.

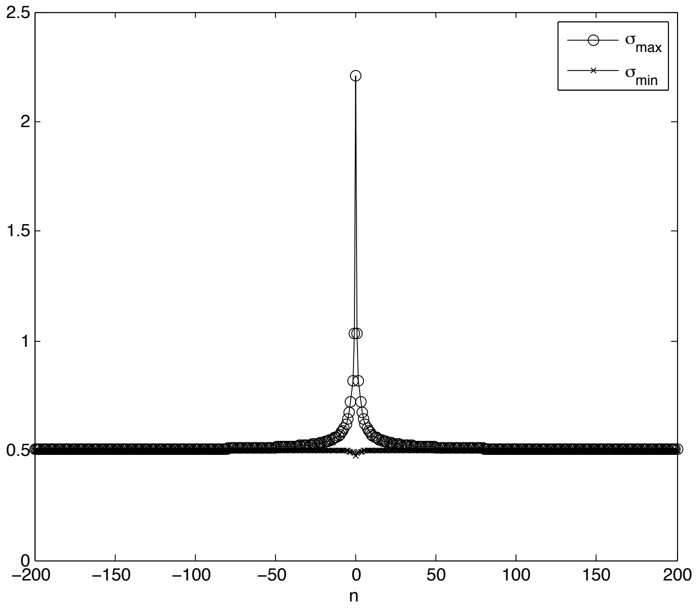


Figure 4.4: Maximum and minimum singular values for the matrices resulting from an 80 panel discretization of a sphere using 400 Fourier modes, where n is the the matrix associated with the $n^{\rm th}$ Fourier mode.

Figure 5.3(a). The convergence of both the smallest and the largest singular values to 1/2 follow from the convergence $||\mathsf{A}^{(n)}|| \to 0$, cf. Section 4.2.2. Figure 4.4 indicates both that all matrices involved are well-conditioned, and that truncation of the Fourier series is generally safe.

4.8.2 Helmholtz Equation

In this section, we repeat many of the experiments reported in Section 4.8.1, but now for the Helmholtz equation on an exterior domain with the associated "combined field" BIE formulation (46). The algorithm employed to solve the integral equation is the same as described in Section 4.4, with the caveat that the kernels are calculated using the fast procedure described in Section 4.6.1.

Table 4.6 presents timing results, with all variables defined as in Section 4.8.1.

N_{P}	$2N+1$	T_{mat}	T_{inv}	T_{fft}	T_{apply}
5	25	4.51E-02	3.78E-03	2.59E-04	1.77E-04
10	25	1.18E-01	2.03E-02	4.42E-04	8.38E-04
20	25	3.77E-01	1.48E-01	8.47E-04	3.05E-03
40	25	1.27E+00	9.71E-01	1.71E-03	1.19E-02
80	25	5.37E+00	8.20E+00	3.68E-03	8.17E-02
5	51	8.06E-02	6.91E-03	5.09E-04	3.66E-04
10	51	2.25E-01	4.27E-02	1.10E-03	1.00E-03
20	51	7.53E-01	2.85E-01	2.08E-03	6.13E-03
40	51	2.69E+00	1.95E+00	3.83E-03	2.79E-02
80	51	1.01E+01	1.47E+01	7.55E-03	1.40E-01
5	101	1.57E-01	1.36E-02	1.33E-03	8.13E-04
10	101	4.74E-01	8.20E-02	2.55E-03	3.05E-03
20	101	1.58E+00	5.27E-01	5.03E-03	1.17E-02
40	101	5.95E+00	3.81E+00	1.01E-02	5.67E-02
80	101	2.11E+01	2.72E+01	1.90E-02	2.39E-01
5	201	3.02E-01	2.50E-02	2.56E-03	1.64E-03
10	201	9.40E-01	1.56E-01	5.09E-03	5.71E-03
20	201	3.23E+00	1.02E+00	1.01E-02	2.15E-02
40	201	1.19E+01	7.32E+00	2.05E-02	9.94E-02
80	201	4.35E+01	5.38E+01	4.04E-02	4.67E-01
5	401	5.90E-01	5.046E-02	5.66E-03	3.48E-03
10	401	1.86E+00	2.97E-01	1.05E-02	1.06E-02
20	401	6.60E+00	2.04E+00	2.20E-02	4.75E-02
40	401	2.40E+01	1.46E+01	4.42E-02	1.98E-01

Table 4.6: Timing results in seconds performed for a spherical domain.

The largest problem size considered here has 80 panels and 201 Fourier modes, leading to 160 800 unknowns discretizing the surface. Note that this problem size is slightly smaller than the largest considered in Section 4.8.1, due to memory constraints. This is because the matrices now contain complex entries, and thus use twice the memory compared with the Laplace case. The total running time of the algorithm is 97 seconds for this problem size. If given additional right hand sides, we can solve them in 0.51 seconds. We also remark that the asymptotic scaling of the cost of this algorithm is identical to that of the Laplace case, it simply takes more operations (roughly twice as many) to calculate the kernels and perform the required matrix operations.

We have assessed the accuracy of the algorithm for various domains, discretization parameters, and Fourier modes. The boundary conditions used are determined by placing point charges inside the domains, and we evaluate the solution at random points placed on a sphere that encompasses the boundary. In the combined field BIE (46), we set the parameter $\nu = k$.

$N_{\rm P}$	$2N+1$				
-	25	51	101	201	401
5	4.2306E-04	4.0187E-04	4.0185E-04	4.0185E-04	4.0185E-04
10	3.4791E-06	2.3738E-06	2.3759E-06	2.3760E-06	2.376E-06
20	7.8645E-06	4.5707E-09	7.1732E-11	7.1730E-11	7.173E-11
40	1.3908E-05	1.5980E-08	2.9812E-13	3.1500E-13	3.148E-13
80	1.0164E-05	5.4190E-09	4.9276E-13	4.8895E-13	-

Table 4.7: Relative error in external Helmholtz problem for the domain in Figure 5.3(a). The domain is 1 wavelength in length (the major axis).

$N_{\rm P}$	$2N+1$				
-	25	51	101	201	401
5	2.0516E+00	3.4665E+00	4.1762E+00	4.4320E+00	4.4951E+00
10	2.3847E-01	2.4301E-01	2.4310E-01	2.4312E-01	2.4313E-01
20	1.7792E-02	8.8147E-06	8.8075E-06	8.8081E-06	8.8082E-06
40	1.7054E-02	5.4967E-08	3.7994E-10	3.7999E-10	3.7998E-10
80	1.7302E-02	1.6689E-08	1.8777E-11	1.8782E-11	-

Table 4.8: Relative error in external Helmholtz problem for the domain in Figure 5.3(a). The domain is 25 wavelengths in length (the major axis).

First, we consider the ellipsoidal domain given in Figure 5.3(a). The major axis of this ellipse has a diameter of 2, and its minor axes have a diameter of 1/2. Table 4.7 lists the accuracy achieved. We achieve 9 digits of accuracy in this problem with 20 panels and 51 Fourier modes. We have not padded the two sequences in the convolution procedure described in Section 4.6.1, but as more modes are used the tails of the sequence rapidly approach zero, increasing the accuracy of the convolution algorithm utilized to calculate the kernels. Table 4.8 displays the same data, but with the wavenumber increased so that there are 25 wavelengths along the length of the ellipsoid. We see a minor decrease in accuracy as we would expect, but it only takes 40 panels and 51 Fourier modes to achieve 8 digits of accuracy.

N_P	$2N+1$				
-	25	51	101	201	401
5	2.6874E+00	2.5719E+00	2.5826E+00	2.2374E+00	1.9047E+00
10	1.6351E+00	4.2972E+00	4.6017E+00	1.0380E+00	1.0328E+00
20	4.3666E+00	2.6963E-03	2.6966E-03	2.6967E-03	2.6967E-03
40	4.4498E+00	9.1729E-08	4.1650E-08	4.1661E-08	4.1664E-08
80	4.3692E+00	7.3799E-08	1.4811E-10	1.4812E-10	-

Table 4.9: Relative error in external Helmholtz problem for the domain in Figure 5.3(b). The domain is 10 wavelengths in length (the major axis).

N_P	$2N+1$				
-	25	51	101	201	401
5	1.1140E+01	3.2400E+01	3.3885E+01	4.4322E+01	1.6151E+01
10	1.9626E+01	7.7739E+01	6.2931E+01	3.6335E-01	3.6344E-01
20	2.6155E+01	4.9485E+01	2.5796E+01	4.8239E-05	4.8245E-05
40	3.3316E+01	5.0645E+01	2.4330E+01	1.3841E-09	1.3846E-09
80	1.6966E+01	6.0163E+01	2.4354E+01	1.6510E-10	-

Table 4.10: Relative error in external Helmholtz problem for the domain in Figure 5.3(c). The domain is 10 wavelengths in length (the major axis).

We now consider the more complex domains given in Figure 5.3(b) and 5.3(c). There are a wavy shaped block and a starfish shaped block with the outer diameter of size roughly 1.5 and 1.0, respectively. The accuracy for various values of N_P and N are given in Table 4.9 and 4.10. We achieve 8 digits of accuracy with 40 panels and 51 Fourier modes for the wavy block and 9 digits of accuracy with 40 panels and 201 Fourier modes for the starfish block.

Chapter 5

An efficient and highly accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers

S. Hao, P.G. Martinsson

Note: *The work described in this chapter was carried out in collaboration with Professor Per-Gunnar Martinsson of the University of Colorado. It appeared in the Advances in Computers and Mathematics with Applications (69(4), pages 304 - 318, 2015) under the title: "An efficient and highly accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers ".*

Abstract: A numerical method for solving the equations modeling acoustic scattering is presented. The method is capable of handling several dozen scatterers, each of which is several wave-lengths long, on a personal work station. Even for geometries involving cavities, solutions accurate to seven digits or better were obtained. The method relies on a Boundary Integral Equation formulation of the scattering problem, discretized using a high-order accurate Nyström method. A hybrid iterative/direct solver is used in which a local scattering matrix for each body is computed, and then GMRES, accelerated by the Fast Multipole Method, is used to handle reflections between the scatterers. The main limitation of the method described is that it currently applies only to scattering bodies that are rotationally symmetric.

5.1 Introduction

The manuscript presents a robust and highly accurate numerical method for modeling frequency domain acoustic scattering on a domain external to a group of scatterers in three dimensions.

The solver is designed for the special case where each scatterer is rotationally symmetric, and relies on a Boundary Integral Equation (BIE) formulation of the scattering problem.

The contribution of the manuscript is to combine several recently developed techniques to obtain a solver capable of solving scattering problems on complex multibody geometries in three dimensions to seven digits of accuracy or more. In particular, the solver is capable of resolving domains involving cavities such as, e.g., the geometry shown Figure 5.5(a).

The solution technique proposed involves the following steps:

(1) *Reformulation.* The problem is written mathematically as a BIE on the surface of the scattering bodies using the "combined field" formulation [26, 81]. See Section 5.2 for details.

(2) *Discretization.* The BIE is discretized using the Nyström method based on a high-order accurate composite Gaussian quadrature rule. Despite the fact that the kernel in the BIE is singular, high accuracy can be maintained using the correction techniques of [67, 51]. Following [86], we exploit the rotational symmetry of each body to decouple the local equations as a sequence of equations defined on a generating contour [90, 92, 73, 99, 98]. This dimension reduction technique requires an efficient method for evaluating the fundamental solution of the Helmholtz equation in cylindrical coordinates (the so called "toroidal harmonics"); we use the technique described in [103]. See Section 6.2 for details.

(3) *Iterative solver.* The dense linear system resulting from the Nyström discretization of the BIE is solved using the iterative solver GMRES [88], combined with a block-diagonal pre-conditioner, as in, e.g., [62, Sec. 6.4]. This pre-conditioner exploits that a highly accurate discrete approximation to the scattering matrix for each individual scatterer can be computed efficiently. See Section 5.4 for details.

(4) *Fast matrix-vector multiplication.* The application of the coefficient matrix in the iterative solver is accelerated using the Fast Multipole Method (FMM) [43], specifically the version

for the Helmholtz equation developed by Gimbutas and Greengard [39].

(5) *Skeletonization.* In situations where the individual scatterers are not packed very tightly, the number of degrees of freedom in the global system can be greatly reduced by exploiting rank deficiencies in the off-diagonal blocks of the coefficient matrix. Specifically, we use a variation of the scheme introduced in [23], and further developed in [76]. Randomized methods are used to accelerate the computation of low-rank approximations to large dense matrices [48]. See Section 5.5 for details.

The present work draws on several recent papers describing techniques for multibody scattering, including [62], which applies a very similar technique to acoustic scattering in two dimensions. [40] addresses the harder problem of electro-magnetic scattering in 3D (as opposed to the acoustic scattering considered here), but uses classical scattering matrices expressed in spherical harmonics. This is a more restrictive frame-work than the one used in [62] for problems in 2D, and in the present work for problems in 3D. The more general model for a compressed scattering matrix that we use here allows for larger scatterers to be handled, and also permits it to handle scatterers closely packed together. For a deeper discussion of different ways of representing compressed scattering matrices, see [13].

To describe the asymptotic cost of the method presented, let m denote the number of scatterers, let n denote the total number of discretization nodes on a single scatterer and let I denote the number of iterations required in our pre-conditioned iterative solver to achieve convergence. The cost of building all local scattering matrices is then $O(mn^2)$, and the cost of solving the linear system consists of the time T_{FMM} required for applying the coefficient matrices using the FMM, and the time T_{precond} required for applying the block-diagonal preconditioner. These scale as $T_{\text{FMM}} \sim Imn$ and $T_{\text{precond}} \sim Imn^{3/2}$ (cf. Remark 7), but for practical problem sizes, the execution time is completely dominated by the FMM. For this reason, we implemented a "skeletonization" compression scheme [23] that reduces the cost of executing the FMM from Imn to Imk, where k is a numerically determined "rank of interaction". We provide numerical examples in Section 5.6

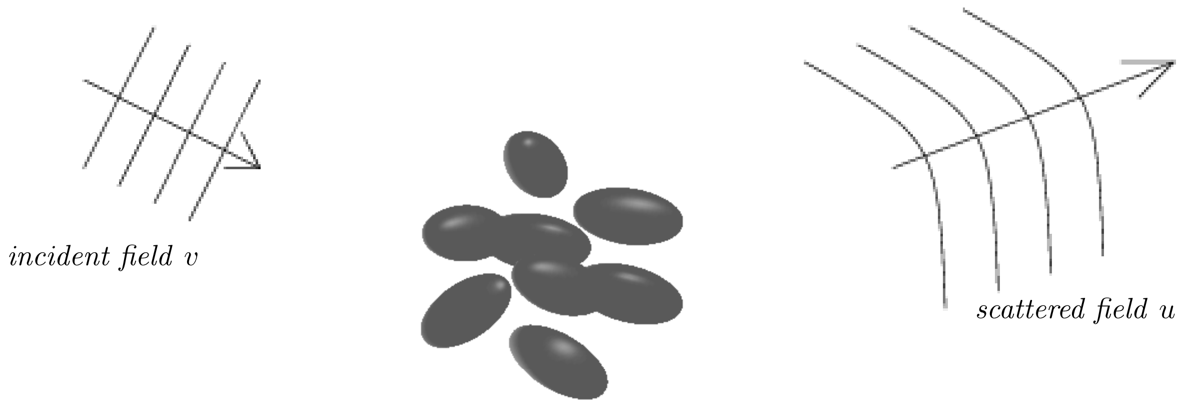


Figure 5.1: Geometry of scattering problem. An incident field v propagates in a medium with constant wave-speed and hits a scattering surface $\Gamma = \bigcup_{p=1}^{m}\Gamma_p$ (shown for $m = 8$). A charge distribution σ is induced on the surface Γ and generates an outgoing field u.

that demonstrate that when the scatterers are moderately well separated, k can by smaller than n by one or two orders of magnitude, leading to dramatic practical acceleration.

5.2 Mathematical formulation of the scattering problem

Let $\{\Gamma_p\}_{p=1}^{m}$ denote a collection of m smooth, disjoint, rotationally symmetric surfaces in $\mathbb{R}^3$, let $\Gamma = \cup_{p=1}^{m}\Gamma_p$ denote their union, and let Ω denote the domain exterior to Γ. Our task is to compute the "scattered field" u generated by an incident field v that hits the scattering surface Γ, see Figure 5.1. For concreteness, we consider the so called "sound-soft" scattering problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega^{\mathrm{c}}, \\ u(\boldsymbol{x}) = \ -v(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \\ \dfrac{\partial u(\boldsymbol{x})}{\partial r} - i\kappa u(\boldsymbol{x}) = O(1/r) & r := |\boldsymbol{x}| \to \infty. \end{cases} \tag{1}$$

We assume that the "wave number" κ is a real non-negative number. It is known [26] that (1) has a unique solution for every incoming field v.

Following standard practice, we reformulate (1) as second kind Fredholm Boundary Integral Equation (BIE) using a so called "combined field technique" [26, 81]. We then look for a solution

u of the form

$$u(\boldsymbol{x}) = \int_\Gamma G_\kappa(\boldsymbol{x}, \boldsymbol{x}')\,\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}'), \quad \boldsymbol{x} \in \Omega^c, \tag{2}$$

where G_κ is a combination of the single and double layer kernels,

$$G_\kappa(\boldsymbol{x}, \boldsymbol{x}') = \frac{\partial \phi_\kappa(\boldsymbol{x}, \boldsymbol{x}')}{\partial \boldsymbol{n}(\boldsymbol{x}')} + i\kappa\,\phi_\kappa(\boldsymbol{x}, \boldsymbol{x}') \tag{3}$$

and where ϕ_κ is the free space fundamental solution

$$\phi_\kappa(\boldsymbol{x}, \boldsymbol{x}') = \frac{e^{i\kappa|\boldsymbol{x}-\boldsymbol{x}'|}}{4\pi|\boldsymbol{x}-\boldsymbol{x}'|}. \tag{4}$$

Equation (2) introduces a new unknown function σ, which we refer to as a "boundary charge distribution". To obtain an equation for σ, we take the limit in (2) as $\boldsymbol{x}$ approaches the boundary Γ, and find that σ must satisfy the integral equation

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma G_\kappa(\boldsymbol{x}, \boldsymbol{x}')\,\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}') = -v(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma. \tag{5}$$

The combined field equation (5) is known to be a second kind Fredholm equation whenever Γ is smooth. Like the orignal boundary value problem (1), it is known to be well posed for every κ, see [26, Theorem. 3.9], [81, Sec. 3.2.2] (in particular, it does not suffer from the problem of "artificial resonances" that plague many alternative formulations).

5.3 Discretization of rotationally symmetric scattering bodies

In Section 5.2 we formulated the scattering problem as the BIE (5) defined on the scattering surface Γ. In this section, we show how to discretize (5) to obtain a system of linear algebraic equations $\mathsf{A}\boldsymbol{\sigma} = -\mathbf{v}$. We use a Nyström technique that combines high accuracy, and (relative) ease of implementation. Section 5.3.1 gives a general overview of the Nyström method, Section 5.3.2 describes how rotational symmetry can be exploited to relatively easily discretize a single body to high order, and then Section 5.3.3 describes how to generalize the procedure to a multibody scattering problem.

5.3.1 Nyström discretization

The Nyström method provides a way of discretizing a BIE on a surface Γ from a quadrature rule for the surface that is valid for smooth functions. To illustrate, suppose that we are given *nodes* $\{\boldsymbol{x}_i\}_{i=1}^n$ and *weights* $\{w_i\}_{i=1}^n$ such that

$$\int_\Gamma \varphi(\boldsymbol{x})\, dS(\boldsymbol{x}) \approx \sum_{i=1}^n \varphi(\boldsymbol{x}_i)\, w_i, \qquad \text{for } \varphi \text{ smooth.} \tag{6}$$

The idea is then to first use the discretization nodes $\{\boldsymbol{x}_i\}_{i=1}^n$ as collocation points; in other words, we require that

$$\frac{1}{2}\sigma(\boldsymbol{x}_i) + \int_\Gamma G_\kappa(\boldsymbol{x}_i, \boldsymbol{x}')\, \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') = -v(\boldsymbol{x}_i), \quad i = 1, 2, 3, \dots, n. \tag{7}$$

Next, suppose that we can somehow (this can require some work) construct an $n \times n$ matrix A such that for any sufficiently smooth function φ, the integral in (7) can be approximated from the function values $\{\sigma(\boldsymbol{x}_i)\}_{i=1}^n$

$$\frac{1}{2}\sigma(\boldsymbol{x}_i) + \int_\Gamma G_\kappa(\boldsymbol{x}_i, \boldsymbol{x}')\, \sigma(\boldsymbol{x}')\, dA(\boldsymbol{x}') \approx \sum_{j=1}^n \mathsf{A}(i,j)\, \sigma(\boldsymbol{x}_j) \qquad \text{for } \sigma \text{ smooth.} \tag{8}$$

Then a system of n equations for the n unknowns $\{\sigma(\boldsymbol{x}_i)\}_{i=1}^n$ is obtained by inserting the approximation (8) into (7). Specifically, given a data vector $\mathbf{v} \in \mathbb{C}^n$ given by $\mathbf{v}(i) = v(\boldsymbol{x}_i)$, we seek to determine a vector $\boldsymbol{\sigma} \in \mathbb{C}^n$ of approximations $\boldsymbol{\sigma}(i) \approx \sigma(\boldsymbol{x}_i)$ by solving the linear system

$$\sum_{j=1}^n \mathsf{A}(i,j)\, \boldsymbol{\sigma}(j) = -\mathbf{v}(i), \quad i = 1, 2, 3, \dots, n. \tag{9}$$

The task of constructing a matrix A such that (8) holds is complicated by the fact that the kernel $G_\kappa(\boldsymbol{x}, \boldsymbol{x}')$ has a singularity as $\boldsymbol{x}' \to \boldsymbol{x}$. Had this not been the case, one could simply have applied the rule (6) to the integral in (7) to obtain

$$\mathsf{A}(i,j) = G_\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)\, w_j. \tag{10}$$

In Sections 5.3.2 and 5.3.3 we will describe how to construct a basic quadrature rule $\{\boldsymbol{x}_i, w_i\}_{i=1}^n$ that is suitable for the geometry under consideration, and also how to construct a matrix A such that (8) holds to high accuracy despite the singular kernel. It turns out to be possible to do so while having almost all elements of A given by the simple formula (10) — only matrix elements $\mathsf{A}(i,j)$ for which $||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is "small" need to be modified. As we will see in Section 5.4, this will greatly help when forming fast algorithms for evaluating the matrix-vector product $\boldsymbol{\sigma} \mapsto \mathsf{A}\boldsymbol{\sigma}$.

5.3.2 A single rotationally symmetric scatterer

We first consider the case where the scattering surface Γ is a single rotationally symmetric surface. We let γ denote a generating curve of Γ, and can then view Γ as a tensor product between γ and the circle $\mathbb{T}$, so that $\Gamma = \gamma \times \mathbb{T}$, see Figure 5.2. The idea is now to use a composite Gaussian rule to discretize γ, and a trapezoidal rule with equispaced nodes to discretize $\mathbb{T}$, and then take the tensor product between these rules to obtain the global rule $\{\boldsymbol{x}_i, w_i\}_{i=1}^{n}$ for Γ.

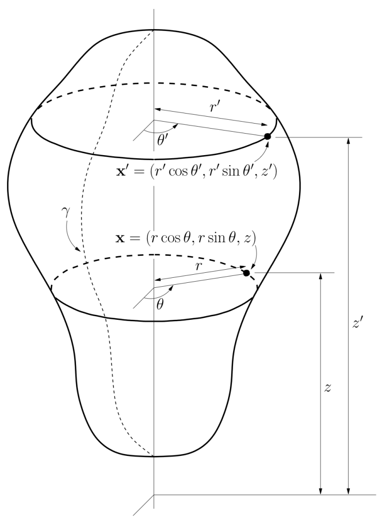

r'
θ'
$\mathbf{x}' = (r'\cos\theta', r'\sin\theta', z')$
γ
$\mathbf{x} = (r\cos\theta, r\sin\theta, z)$
r
θ
z'
z

Figure 5.2: The axisymmetric domain Γ generated by the curve γ.

Remark 4 (Convergence order) Suppose that φ is a smooth (C^∞) function on Γ. Then since φ is periodic in the azimuthal direction, the Trapezoidal rule converges super-algebraically fast. If we use p-point Gaussian quadrature on r intervals to discretize the generating curve γ, then the error in (6) scales as $(1/r)^{2p-1}$ as $r, p \to \infty$.

The technique for constructing a matrix A such that (8) holds is based on the observation that when Γ is a rotationally symmetric surface, the equation (5) is diagonalized by the Fourier transform. The process is somewhat involved and we will here give only a brief overview of the key techniques, for details we refer to [103]. The first step is to introduce cylindrical coordinates $\boldsymbol{x} = (r, \theta, z)$ with the z-axis being the symmetry axis of Γ, and let v_p, σ_p, and $G_{\kappa,p}$ denote the Fourier coefficients of the functions v, σ, and G_κ:

$$v(\boldsymbol{x}) = \sum_{p\in\mathbb{Z}} \frac{e^{ip\theta}}{\sqrt{2\pi}}\, v_p(r,z), \tag{11}$$

$$\sigma(\boldsymbol{x}) = \sum_{p\in\mathbb{Z}} \frac{e^{ip\theta}}{\sqrt{2\pi}}\, \sigma_p(r,z), \tag{12}$$

$$G_\kappa(\boldsymbol{x},\boldsymbol{x}') = G_\kappa(\theta-\theta', r, z, r', z') = \sum_{p\in\mathbb{Z}} \frac{e^{ip(\theta-\theta')}}{\sqrt{2\pi}}\, G_{\kappa,p}(r,z,r',z'). \tag{13}$$

Then (5) is equivalent to the sequence of equations

$$\frac{1}{2}\sigma_p(\boldsymbol{y}) + \sqrt{2\pi}\int_\gamma G_{\kappa,p}(\boldsymbol{y},\boldsymbol{y}')\,\sigma_p(\boldsymbol{y}')\,dA(\boldsymbol{y}') = -v_p(\boldsymbol{y}), \quad \boldsymbol{y}\in\gamma,\ p\in\mathbb{Z}. \tag{14}$$

Converting the BIE (5) defined on a surface Γ to the sequence of BIEs (14) defined on the curve γ has a crucial advantage in that constructing high-order Nyström discretizations of BIEs with weakly singular kernels is well-understood and computationally cheap for curves, but remains a challenge for surfaces. We use the modified quadrature of [67], as described in [103, 51].

Beyond ease of discretization, the other key benefit of the formulation (14) is that for each Fourier mode p, the coefficient matrix arising from discretization of (14) is small enough that it can often easily be inverted by brute force. For instance, for the geometries shown in Figure 5.3, it is sufficient to use at most a couple of hundred nodes along γ to achieve ten digits accuracy. To put it another way, the Fourier conversion allows to write the matrix A as a product

$$\mathsf{A} = \mathsf{F}^*\,\tilde{\mathsf{A}}\,\mathsf{F} \tag{15}$$

where F is the discrete Fourier transform (in the azimuthal variable), and $\tilde{\mathsf{A}}$ is a block-diagonal matrix, where each diagonal block corresponds to one Fourier mode, and is relatively small. We can pre-compute and store the block diagonal matrix $\tilde{\mathsf{A}}^{-1}$, and then very rapidly apply the inverse

$$\mathsf{A}^{-1} = \mathsf{F}^* \, \tilde{\mathsf{A}}^{-1} \, \mathsf{F}, \tag{16}$$

by using the FFT to apply F and F^*.

One complication to the procedure outlined in this section is that while the kernel G_κ in (5) is given by the simple formula (3), the kernels $G_{\kappa,p}$ must be evaluated computationally. Techniques for doing so rapidly have been developed, and are described in [103].

Remark 5 (Cost of precomputation) To state the asymptotic cost of the algorithm, let N_{G} ("G" for Gaussian) denote the number of points on the generating curve γ of each scatter and let N_{F} ("F" for Fourier) denote the number of points used to discretize $\mathbb{T}$. The total number of degrees of freedom of each scatter is $n = N_{\mathrm{G}} N_{\mathrm{F}}$. Under the simplifying assumption that $N_{\mathrm{G}} \sim N_{\mathrm{F}}$, the cost of forming the block diagonal matrix $\tilde{\mathsf{A}}$ is $O(n^{3/2} \log n)$, while the cost of inverting $\tilde{\mathsf{A}}$ is $O(n^2)$, see [103]. Applying F and F^* is done via the FFT in negligible time.

5.3.3 Multibody scattering

Having described how to discretize the single-body scattering problem in Section 5.3.2, we now proceed to the general case of m disjoint scattering surfaces $\Gamma = \cup_{p=1}^{m} \Gamma_p$. We assume that each scatterer is discretized using the tensor product procedure described in Section 5.3.2. For notational simplicity, we assume that each scatterer is discretized using the same n number of nodes, for a total of $N = mn$ discretization nodes $\{\boldsymbol{x}_i\}_{i=1}^{N}$ with associated weights $\{w_i\}_{i=1}^{N}$. We then seek to construct matrix blocks $\{\mathsf{A}_{p,q}\}_{p,q=1}^{m}$ such that the Nyström discretization of (5) associated with this quadrature rule takes the form

$$\begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} & \cdots & \mathsf{A}_{1,m} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} & \cdots & \mathsf{A}_{2,m} \\ \vdots & \vdots & & \vdots \\ \mathsf{A}_{m,1} & \mathsf{A}_{m,2} & \cdots & \mathsf{A}_{m,m} \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \vdots \\ \boldsymbol{\sigma}_m \end{bmatrix} = - \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_m \end{bmatrix}, \tag{17}$$

where each block $\mathsf{A}_{p,q}$ is of size $n \times n$. The diagonal blocks $\mathsf{A}_{p,p}$ are constructed using the technique described in Section 5.3.2. Next observe that in the off-diagonal blocks, the "naive" formula (10) works well since the kernel $G_\kappa(\boldsymbol{x}, \boldsymbol{x}')$ is smooth when $\boldsymbol{x}$ and $\boldsymbol{x}'$ belong to different scatterers.

Remark 6 In this paper, we avoid considering the complications of scatterers that touch or are very close. The procedure described works well as long as the minimal distance between scatterers is not small compared to the resolution of the quadrature rules used. This means that if two scatterers are moderately close, high accuracy can be maintained by discretizing these two scatterers more finely.

5.4 A block-diagonal pre-conditioner for the multibody scattering problem

We solve the linear system (17) using the iterative solver GMRES [88], accelerated by a block-diagonal pre-conditioner. To formalize, let us decompose the system matrix as

$$\mathsf{A} = \mathsf{D} + \mathsf{B},$$

where

$$\mathsf{D} = \begin{bmatrix} \mathsf{A}_{1,1} & 0 & 0 & \cdots \\ 0 & \mathsf{A}_{2,2} & 0 & \cdots \\ 0 & 0 & \mathsf{A}_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \qquad \text{and} \qquad \mathsf{B} = \begin{bmatrix} 0 & \mathsf{A}_{1,2} & \mathsf{A}_{1,3} & \cdots \\ \mathsf{A}_{2,1} & 0 & \mathsf{A}_{2,3} & \cdots \\ \mathsf{A}_{3,1} & \mathsf{A}_{3,2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}.$$

Then we use GMRES to solve the linear system

$$\boldsymbol{\sigma} + \mathsf{D}^{-1}\mathsf{B}\boldsymbol{\sigma} = -\mathsf{D}^{-1}\mathbf{v}. \tag{18}$$

We apply the matrix B using the Fast Multipole Method [43, 21]; specifically the implementation [39] by Zydrunas Gimbutas and Leslie Greengard.

Remark 7 The cost of evaluating the term $\mathsf{D}^{-1}\mathsf{B}\boldsymbol{\sigma}$ in (18) consists of two parts: applying B to vector $\boldsymbol{\sigma}$ via FMM costs $O(mn)$ operations and applying the block-diagonal pre-conditioner costs $O(mn^{3/2})$ operations. Observe that the matrix D^{-1} can be precomputed since each matrix $\mathsf{A}_{p,p}^{-1}$ is itself block-diagonal in the local Fourier basis, cf. formula (16). Applying $\mathsf{A}_{p,p}^{-1}$ to a vector $\mathbf{w} \in \mathbb{C}^n$ is executed as follows: (1) form $\mathsf{F}\mathbf{w}$ using the FFT at cost $O(n \log n)$, (2) for each Fourier mode apply D^{-1} to $\mathsf{F}\mathbf{w}$ at cost $O(n^{3/2})$, and (3) use the FFT to apply F^* to $\mathsf{D}^{-1}\mathsf{F}\mathbf{w}$.

5.5 Accelerated multibody scattering

In situations where the scatterers are not tightly packed, it is often possible to substantially reduce the size of the linear system (18) before applying an iterative solver. We use a technique that was introduced in [23] for problems in two dimensions, which exploits that when the scatterers are somewhat separated, the off-diagonal blocks $\mathsf{A}_{p,q}$ are typically rank deficient. Specifically, we assume that for some finite precision ε (say $\varepsilon = 10^{-10}$), each such block admits a factorization

$$\begin{array}{ccccccc} \mathsf{A}_{p,q} & = & \mathsf{U}_p & \tilde{\mathsf{A}}_{p,q} & \mathsf{V}_q^* & + & \mathsf{R}_{p,q} \\ n \times n & & n \times k & k \times k & k \times n & & n \times n \end{array} \tag{19}$$

where n is the number of nodes originally used to discretize a single scatterer, and k is the numerical rank of the factorization. The remainder term $\mathsf{R}_{p,q}$ satisfies $||\mathsf{R}_{p,q}|| \leq \varepsilon$ in some suitable matrix norm (we typically use the Frobenius norm since it is simple to compute).

Now write the linear system (18) in block form as

$$\boldsymbol{\sigma}_p + \sum_{q \neq p} \mathsf{A}_{p,p}^{-1} \mathsf{A}_{p,q} \boldsymbol{\sigma}_q = -\mathsf{A}_{p,p}^{-1} \mathbf{v}_p, \qquad p = 1,\, 2,\, 3,\, \dots,\, m. \tag{20}$$

We left multiply (20) by V_p^*, and insert the factorization (19) to obtain

$$\mathsf{V}_p^* \boldsymbol{\sigma}_p + \sum_{q \neq p} \mathsf{V}_p^* \mathsf{A}_{p,p}^{-1} \mathsf{U}_p \tilde{\mathsf{A}}_{p,q} \mathsf{V}_q^* \boldsymbol{\sigma}_q = -\mathsf{V}_p^* \mathsf{A}_{p,p}^{-1} \mathbf{v}_p, \qquad p = 1,\, 2,\, 3,\, \dots,\, m. \tag{21}$$

We now define quantities $\{\tilde{\boldsymbol{\sigma}}_p\}_{p=1}^m$, $\{\tilde{\mathbf{v}}_p\}_{p=1}^m$, and $\{\tilde{\mathsf{S}}_p\}_{p=1}^m$ via

$$\tilde{\boldsymbol{\sigma}}_p = \mathsf{V}_p^* \boldsymbol{\sigma}_p, \qquad \tilde{\mathbf{v}}_p = \mathsf{V}_p^* \mathsf{A}_{p,p}^{-1} \mathbf{v}_p \qquad \mathsf{S}_{p,p} = \mathsf{V}_p^* \mathsf{A}_{p,p}^{-1} \mathsf{U}_p, \qquad \text{for } p = 1,\, 2,\, 3,\, \dots,\, m. \tag{22}$$

Then the system (21) can be written

$$\tilde{\boldsymbol{\sigma}}_p + \sum_{q \neq p} \mathsf{S}_p \tilde{\mathsf{A}}_{p,q} \tilde{\boldsymbol{\sigma}}_q = -\tilde{\mathbf{v}}_p, \qquad p = 1,\, 2,\, 3,\, \dots,\, m. \tag{23}$$

To write (23) in block form, we introduce matrices

$$\mathsf{S} = \begin{bmatrix} \mathsf{S}_1 & 0 & 0 & \cdots \\ 0 & \mathsf{S}_2 & 0 & \cdots \\ 0 & 0 & \mathsf{S}_3 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \qquad \text{and} \qquad \tilde{\mathsf{B}} = \begin{bmatrix} 0 & \tilde{\mathsf{A}}_{1,2} & \tilde{\mathsf{A}}_{1,3} & \cdots \\ \tilde{\mathsf{A}}_{2,1} & 0 & \tilde{\mathsf{A}}_{2,3} & \cdots \\ \tilde{\mathsf{A}}_{3,1} & \tilde{\mathsf{A}}_{3,2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}, \tag{24}$$

whence equation (23) takes the form, cf. (18),

$$\tilde{\boldsymbol{\sigma}} + \mathsf{S}\tilde{\mathsf{B}}\tilde{\boldsymbol{\sigma}} = -\tilde{\mathbf{v}}. \tag{25}$$

The process of first forming the linear system (25), and then solving it using GMRES is very computationally efficient when the following techniques are used:

- The matrices $\{\mathsf{U}_p, \mathsf{V}_p\}_{p=1}^m$ in the factorizations (19) can be computed via a purely local procedure in $O(n^2k)$ operations, independent of the number of scatterers m. The idea is to use representation techniques from scattering theory to construct a local basis for all possible incoming harmonic fields (to within precision ε), see [42, Sec. 5.1] or [36, Sec. 6.2].

- In constructing the factorization (19), the so called *interpolatory decomposition* [23] should be used. Then each matrix U_p and each matrix V_p contains the $k \times k$ identity matrix I_k. Specifically, there exists for each k an index vector $\tilde{I}_p \subset \{1, 2, \dots, n\}$ such that $\mathsf{U}(\tilde{I}_p, :) = \mathsf{V}(\tilde{I}_p, :) = \mathsf{I}_k$. Then each off-diagonal block $\tilde{\mathsf{A}}_{p,q}$ is given as a *submatrix* $\tilde{\mathsf{A}}_{p,q} = \mathsf{A}_{p,q}(\tilde{I}_p, \tilde{I}_q)$. In consequence, the matrix $\tilde{\mathsf{B}}$ is a sub-matrix of B and can be rapidly applied using the FMM in $O(mk)$ operations.

- In evaluating the formula $\mathsf{S}_{p,p} = \mathsf{V}_p^* \mathsf{A}_{p,p}^{-1} \mathsf{U}_p$, we exploit that $\mathsf{A}_{p,p}^{-1}$ can be applied rapidly in Fourier space, cf. (16), to reduce the complexity of this step from $O(n^3)$ to $O(n^{3/2}k)$ if $\mathsf{A}_{p,p}^{-1}$ was precomputed and stored and to $O(n^2k)$ if $\mathsf{A}_{p,p}^{-1}$ is computed at this step.

Remark 8 Efficient techniques for computing interpolative decompositions are described in [23]. More recently, techniques based on randomized sampling have proven to be highly efficient on modern computing platforms, in particular for problems in potential theory where the low-rank matrices to be approximated have very rapidly decaying singular values. We use the specific technique described in [48].

5.6 Numerical examples

This section describes numerical experiments to assess the performance of the numerical scheme outlined in previous sections. All the experiments are carried out on a personal work-station with an Intel Xeon E-1660 3.3GHz 6-core CPU, and 128GB of RAM. The experiments explore (1) the accuracy of the algorithm, (2) the computational cost, (3) the performance of block-diagonal pre-conditioner and (4) the performance of the acceleration scheme when scatterers are separated suitably. In all the experiments below, we measure accuracy against a known analytic solution u_{exact}. This solution is generated by randomly placing one point source inside each scatterer, and then solving (1) with the Dirichlet data v set to equal the field generated by these radiating sources. Let $\mathbf{u}_{\text{exact}}$ and $\mathbf{u}_{\text{approx}}$ denote the vectors holding the exact and the computed solutions at a set of 10 randomly chosen target points, placed at random on a sphere that is concentric to the smallest sphere holding all scatterers, but of twice the radius. The relative error, measured in ℓ^∞-norm, is then given by

$$E_\infty^{\text{rel}} = \frac{||\mathbf{u}_{\text{approx}} - \mathbf{u}_{\text{exact}}||_\infty}{||\mathbf{u}_{\text{exact}}||_\infty} = \frac{\max_i |\mathbf{u}_{\text{approx}}(i) - \mathbf{u}_{\text{exact}}(i)|}{\max_j |\mathbf{u}_{\text{exact}}(j)|}.$$

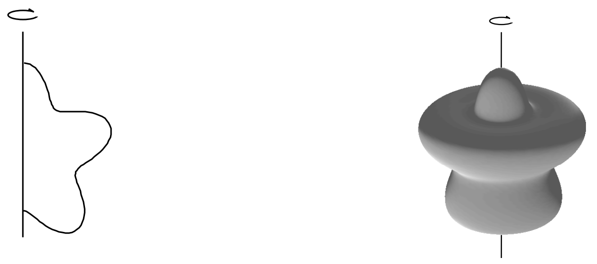
In addition to E_∞^{rel}, we report:

n	number of nodes discretizing each body (in form of $n = N_{\text{G}} \times N_{\text{F}}$, cf. Section 5.3.2)
N	total degree of freedom $N = m \times n$, where m is the number of scatterers
$N_{\text{compressed}}$	number of skeleton points after applying the compression scheme, cf. Section 5.5
T_{pre}	time (in seconds) of precomputation
T_{solve}	total time (in seconds) to solve for the surface charges σ via GMRES
T_{compress}	time (in seconds) to do compression in the accelerated scheme
I	number of GMRES iterations required to reduce the residual to 10^{-9}.

All the numerical experiments in this section are executed on domains composed of the three sample scatterers shown in Figure 5.3.

(a)

(b)



(c)

Figure 5.3: Domains used in numerical examples. All items are rotated about their symmetry axis. (a) An ellipsoid. (b) A bowl-shaped cavity. (c) A starfish-shaped cavity.

5.6.1 Laplace's equation

We first solve the Laplace equation exterior to the domains shown in Figures 5.4 and 5.5(a) (Examples 1 and 2, respectively). A combination of the single and double layer kernels is chosen to represent the potential outside the domain. The integral equation to be solved is

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{1}{4\pi}\left(\frac{1}{|\boldsymbol{x}-\boldsymbol{x}'|} + \frac{\boldsymbol{n}(\boldsymbol{x}')\cdot(\boldsymbol{x}-\boldsymbol{x}')}{|\boldsymbol{x}-\boldsymbol{x}'|^3}\right)\sigma(\boldsymbol{x}')\,dA(\boldsymbol{x}') = f(\boldsymbol{x}), \quad \boldsymbol{x}\in\Gamma.$$

5.6.1.1 Example 1

This example solves the exterior Laplace equation on the domain depicted in Figure 5.4. The domain consists of 125 ellipsoids contained in the box $[0, 10.2]^3$, where each ellipse has a major axis of length 2 and a minor axis of length 1. The minimal distance between any two ellipsoids is 0.05. We did not apply the compression technique since the scatterers are packed tightly. We compare the performance of the algorithm with and without using block-diagonal pre-conditioner in Table 5.1 and find that for this example, the pre-conditioning does not make any real difference. The scheme quickly reaches 9 digits of accuracy with 10 100 discretization nodes per scatterer, with an overall solve time of about 40 minutes.

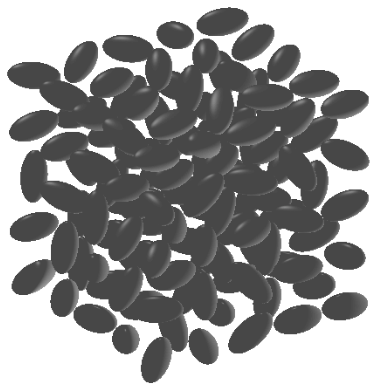


Figure 5.4: Domain contains 125 randomly oriented ellipsoids. Distance between any two ellipsoids is 0.05.

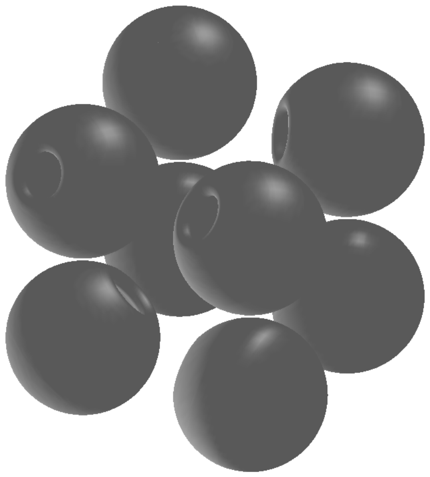
5.6.1.2 Example 2

This time the domain consists of 8 bowl-shaped cavities contained in the box $[0, 4.1]^3$ in Figure 5.5(a). The minimal distance between any two cavities is 0.5. Results are shown in Table 5.2. The scheme achieves 8 digits of accuracy with 400 discretization nodes on the generating curve and 201 Fourier modes. Again, the pre-conditioning is superfluous.

Remark 9 All examples described in this section involve geometries where all the scatterers are

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_∞^{rel}
156 250	50×25	1.09e+00	31 /33	3.16e+02 /3.29e+02	9.731e-05
312 500	100×25	3.44e+00	31 /33	6.84e+02 /6.82e+02	9.203e-05
625 000	200×25	1.29e+01	31 /34	1.10e+03 /1.18e+03	9.814e-05
318 750	50×51	1.53e+00	31 /33	6.29e+02 /7.44e+02	1.571e-06
637 500	100×51	4.36e+00	31 /34	1.18e+03 /1.23e+03	1.529e-06
1 275 000	200×51	1.36e+01	32 /34	2.70e+03 /2.53e+03	1.711e-06
631 250	50×101	2.44e+00	31 /34	1.11e+03 /1.22e+03	2.165e-08
1 262 500	100×101	6.11e+00	32 /34	2.45e+03 /2.60e+03	1.182e-09

Table 5.1: Example 1: exterior Laplace problem solved on the domain in Figure 5.4.


(a)

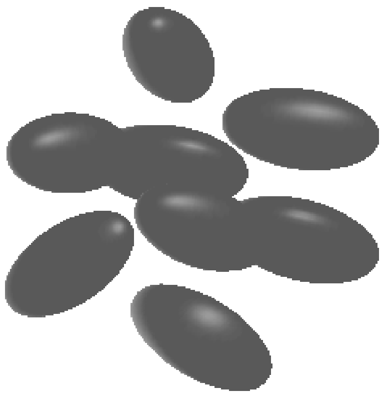

(b)

Figure 5.5: (a) Domain contains 8 bowl-shaped cavities. Distance between any two cavities is 0.5. (b) Domain contains 8 randomly oriented ellipsoids. Distance between any two ellipsoids is 0.05.

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_{∞}^{rel}
20 400	50×51	2.09e-01	398 /402	4.65e+02 /6.05e+02	1.251e-04
40 800	100×51	4.55e-01	20 /23	4.94e+01 /6.09e+01	3.909e-05
81 600	200×51	9.83e-01	20 /23	1.05e+02 /1.14e+02	3.164e-05
40 400	50×101	2.25e-01	20 /23	4.72e+01 /6.17e+01	5.850e-05
80 800	100×101	4.49e-01	20 /23	9.50e+01 /1.13e+02	1.627e-05
161 600	200×101	1.35e+00	20 /24	2.05e+02 /2.39e+02	6.825e-06
80 400	50 × 201	2.93e-01	20 /23	9.13e+01 /1.12e+02	5.704e-05
160 800	100 × 201	7.05e-01	20 /24	1.96e+02 /2.40e+02	8.000e-06
321 600	200 × 201	1.97e+00	20 /24	4.43e+02 /5.25e+02	1.931e-07
643 200	400 × 201	5.78e+00	21 /24	7.68e+02 /8.19e+02	1.726e-08

Table 5.2: Example 2: exterior Laplace problem solved on the domain in Figure 5.5(a).

copies of the basic shapes shown in Figure 5.3. In our experience, this restriction on the geometry does not in any way change the overall accuracy or efficiency of the solver. The only advantage we benefit from is that the pre-computation gets faster, as only a small number of scattering matrices need to be pre-computed. However, it is clear from the numbers given that even for a fully general geometry (without repetitions), the pre-computation time would be dominated by the time required for the FMM.

5.6.2 Helmholtz Equation

We now consider the exterior Helmholtz problem (1). We represent the potential by a combination of the single and double layer kernels, see (3), and end up with the "combined field" integral equation (5).

5.6.2.1 Example 3

The domain in this experiment contains 8 ellipsoids in the box $[0, 4.05]^3$, whose minimal distance between any two is 0.05. The wavelength is 10π so that the scatterers are approximately 10 wavelengths in size and the whole region is about $20 \times 20 \times 20$ wavelengths in size. Results are presented in Table 5.3. We also compare the results without using block-diagonal pre-conditioner in the same table. Around twice of the iteration numbers are required resulting in twice of the computation time. Table 5.4 reports the results from an analogous experiment, but now the wavenumber increases such that each scatterer contains 20 wavelengths.

5.6.2.2 Example 4

This example solves the exterior Helmholtz problem on the cavity domain in Figure 5.5(a). Tables 5.5 and 5.6 show the results from experiments involving cavities of diameters 2 and 5 wavelengths, respectively. In this case, computing the actual scattering matrix for each scatterer was *essential*, without using these to pre-condition the problem, we did not observe any convergence in GMRES.

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_∞^{rel}
20 400	50×51	1.58e-01	35 /67	7.71e+02 /1.56e+03	1.364e-03
40 800	100×51	4.20e-01	36 /67	1.75e+03 /3.43e+03	1.183e-03
81 600	200×51	1.26e+00	36 /68	3.52e+03 /6.85e+03	1.639e-04
40 400	50×101	2.64e-01	36 /68	1.71e+03 /3.35e+03	1.312e-03
80 800	100×101	6.05e-01	36 /68	3.45e+03 /6.76e+03	1.839e-06
161 600	200×101	1.87e+00	37 /69	6.18e+03 /1.19e+04	5.126e-08
80 400	50×201	4.61e-01	36 /69	3.40e+03 /6.70e+03	1.312e-03
160 800	100×201	1.09e+00	37 /69	6.07e+03 /1.18e+04	1.851e-06
321 600	200×201	3.11e+00	37 /69	1.20e+04 /1.97e+04	1.039e-09

Table 5.3: Example 3: exterior Helmholtz problem solved on the domain in Figure 5.5(b). Each ellipsoid is 10 wavelengths in diameter.

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_∞^{rel}
20 400	50×51	2.03e-01	58 /119	3.59e+03 /8.10e+03	4.362e+00
40 800	100×51	4.44e-01	39 /102	3.98e+03 /1.11e+04	1.071e+00
81 600	200×51	1.36e+00	39 /106	6.72e+03 /1.92e+04	1.008e+00
40 400	50×101	2.78e-01	54 /94	5.43e+03 /1.02e+04	5.039e+00
80 800	100×101	6.18e-01	36 /82	6.11e+03 /1.46e+04	8.919e-04
161 600	200×101	1.93e+00	36 /83	9.44e+03 /2.32e+04	5.129e-07
80 400	50×201	4.28e-01	55 /95	9.19e+03 /2.41e+04	5.031e+00
160 800	100×201	1.07e+00	36 /83	9.49e+03 /2.31e+04	8.916e-04
321 600	200×201	3.10e+00	37 /83	1.45e+04 /3.57e+04	8.781e-09

Table 5.4: Example 3: exterior Helmholtz problem solved on the domain in Figure 5.5(b). Each ellipsoid is 20 wavelengths in diameter.

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_∞^{rel}
40 800	100×51	4.29e-01	59 /181	2.17e+03 /6.73e+03	1.127e-02
81 600	200×51	1.28e+00	60 / −	4.23e+03 / −	1.131e-02
80 800	100×101	6.83e-01	60 / −	4.18e+03 / −	3.953e-03
161 600	200×101	1.90e+00	60 / −	8.93e+03 / −	3.802e-04
323 200	400×101	6.07e+00	61 / −	1.91e+04 / −	3.813e-04
160 800	100×201	1.09e+00	60 / −	8.35e+03 / −	4.788e-05
321 600	200×201	3.07e+00	61 / −	1.88e+04 / −	5.488e-06
643 200	400×201	9.61e+00	61 / −	4.03e+04 / −	8.713e-08

Table 5.5: Example 4: exterior Helmholtz problem solved on the domain in Figure 5.5(a). Each cavity is 2 wavelength in diameter.

N	n	T_{pre}	I (precond /no precond)	T_{solve} (precond /no precond)	E_{∞}^{rel}
80 800	100×101	6.54e-01	62 /304	5.17e+03 / 2.64e+04	1.555e-03
161 600	200×101	1.82e+00	63 / –	9.88e+03 / –	1.518e-04
323 200	400×101	6.46e+00	64 / –	2.19e+04 / –	3.813e-04
160 800	100×201	1.09e+00	63 / –	9.95e+03 / –	1.861e-03
321 600	200×201	3.00e+00	64 / –	2.19e+04 / –	2.235e-05
643 200	400×201	1.09e+01	64 / –	4.11e+04 / –	8.145e-06
641 600	200×401	5.02e+00	64 / –	4.07e+04 / –	2.485e-05
1 283 200	400×401	1.98e+01	65 / –	9.75e+04 / –	6.884e-07

Table 5.6: Example 4: exterior Helmholtz problem solved on the domain in Figure 5.5(a). Each cavity is 5 wavelength in diameter.

5.6.3 Accelerated scheme

In this section, we provide two examples illustrating the efficiency of the accelerated scheme in Section 5.5 when applied to the geometries shown in Figures 5.6 (for the Laplace and Helmholtz equations) and 5.8 (for the Helmholtz equation). Recall that the idea here is to discretize each scatterer finely enough to fully resolve the local incoming and outgoing fields. This requires a somewhat large n number of points per scatterer, which for a system with p scatterers leads to a global coefficient matrix of size $np \times np$. Using the compression technique described in Section 5.5, we compute a "reduced" system of size $kp \times kp$, where now k is the (computed) rank of interaction between the scatterers. The number k is largely independent of the local geometry of a scatterer (an accurate upper bound can be derived by considering the speed of convergence when expanding the fundamental solution in terms of spherical harmonics). These examples illustrate representative sizes of k and n, and investigate whether the convergence of GMRES is affected by the compression.

5.6.3.1 Example 5

We apply the accelerated scheme in Section 5.5 to solve the Laplace's equation on the domain exterior to the bodies depicted in Figure 5.6. This geometry contains $p = 50$ different shaped scatterers (ellipsoids, bowls, and rotated "starfish") and is contained in the box $[0, 18] \times [0, 18] \times [0, 6]$. The minimal distance between any two bodies is 4.0. In this example, we have three different shapes of scatterers, and the relevant numbers n and k are given in Figure 5.7. The results obtained when solving the full $np \times np$ system are shown in Table 5.7, while the ones resulting from working with the compressed $kp \times kp$ system are shown in Table 5.8. We see that the compression did not substantially alter either the convergence speed of GMRES, or the final accuracy. Since the time for matrix-vector multiplications is dramatically reduced, the total solve time was reduced between one and two orders of magnitude.

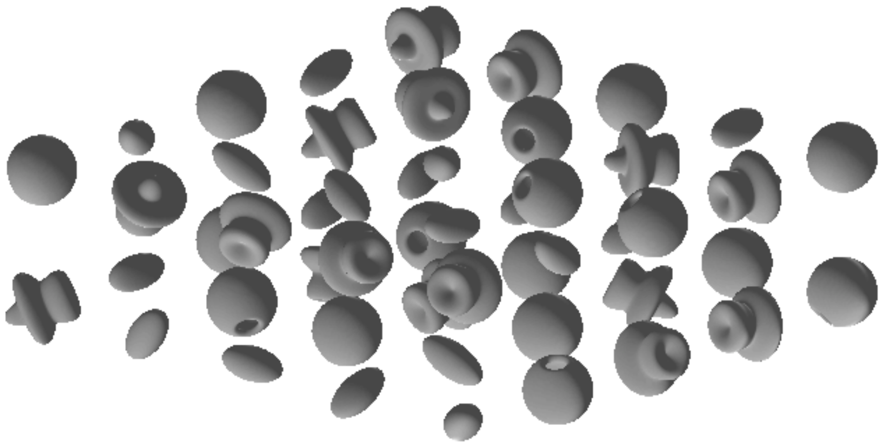


Figure 5.6: Domain contains 50 randomly oriented scatters.

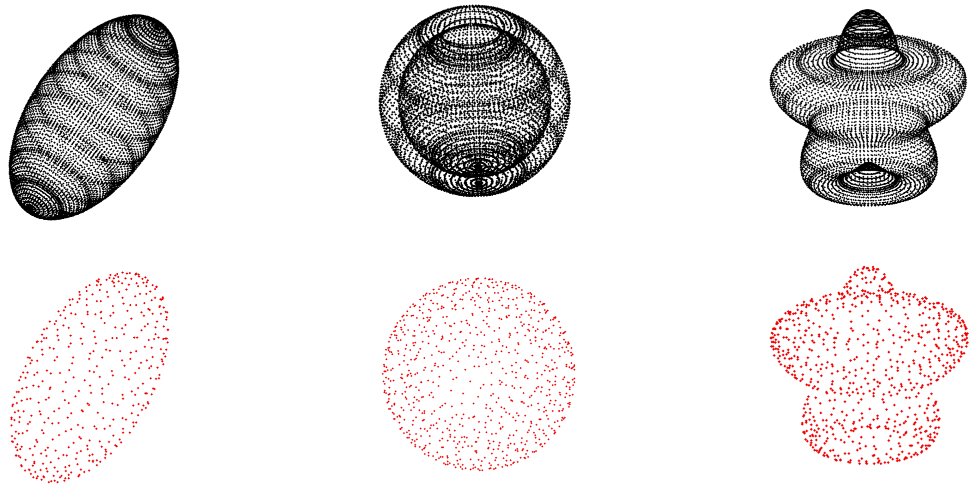


Figure 5.7: Example of skeletonization of three different scatterers before and after compression. With $n = 10\,100$ original discretization points (denoted by black dots), after compression (a) for an ellipsoid, only $k_a = 435$ points survive (denoted by red dots); (b) for a bowl-shaped cavity domain, only $k_b = 826$ points survive; (c) for a starfish-shaped cavity, only $k_c = 803$ points survive.

N	n	T_{pre}	I	T_{solve}	E_{∞}^{rel}
127 500	50×51	2.29e+00	18	1.52e+02	2.908e-05
255 000	100×51	4.70e+00	18	2.94e+02	2.329e-05
510 000	200×51	1.22e+01	18	5.85e+02	2.034e-05
252 500	50×101	3.23e+00	19	2.85e+02	3.677e-05
505 000	100×101	7.08e+00	19	5.29e+02	1.705e-06
1 010 000	200×101	1.93e+01	19	1.06e+03	4.128e-07
502 500	50×201	5.07e+00	19	5.02e+02	3.674e-05
1 050 000	100×201	1.28e+01	19	9.88e+02	1.673e-06
2 010 000	200×201	3.63e+01	19	2.07e+03	1.568e-08

Table 5.7: (Example 5) Results from solving an exterior Laplace problem on the domain in Figure 5.6 with $p = 50$ scatterers. Here the system with the full $np \times np$ coefficient matrix is solved (no compression).

N	n	$N_{\text{compressed}}$	(k_a, k_b, k_c)	T_{compress}	I	T_{solve}	E_{∞}^{rel}
127 500	50×51	30 286	(411,797,746)	3.33e+01	18	3.85e+01	3.042e-05
255 000	100×51	33 876	(434,824,805)	7.00e+01	19	4.25e+01	1.458e-05
510 000	200×51	35 042	(449,847,838)	1.46e+02	19	4.26e+01	1.285e-05
252 500	50×101	32 186	(413,795,752)	6.66e+01	19	3.94e+01	3.008e-05
505 000	100×101	33 894	(435,826,803)	1.40e+02	19	4.04e+01	9.134e-06
1 010 000	200×101	35 094	(451,846,840)	3.20e+02	19	4.12e+01	5.287e-07
502 500	50×201	32 286	(414,797,754)	1.33e+02	19	3.98e+01	3.013e-05
1 050 000	100×201	33 798	(437,830,802)	3.00e+02	19	4.06e+01	9.130e-06
2 010 000	200×201	35 194	(453,848,842)	5.78e+02	19	4.21e+01	4.725e-08

Table 5.8: (Example 5) Results from solving an exterior Laplace problem on the domain in Figure 5.6 using the accelerated scheme with a reduced size coefficient matrix. The ranks k_a, k_b, and k_c for the three “species” of scatterers are given.

5.6.3.2 Example 6

The accelerated scheme is applied to solve Helmholtz equation on domain containing 64 randomly placed ellipsoids depicted in Figure 5.8. The minimal distance between any two bodies is 6.0. Each ellipsoid is 5 wavelengths in diameter. The results for solving this problem without compression are given in Table 5.9, and with compression in Table 5.10. Again, we see that the convergence speed of GMRES is largely unaffected, and that the accelerated scheme is much faster.

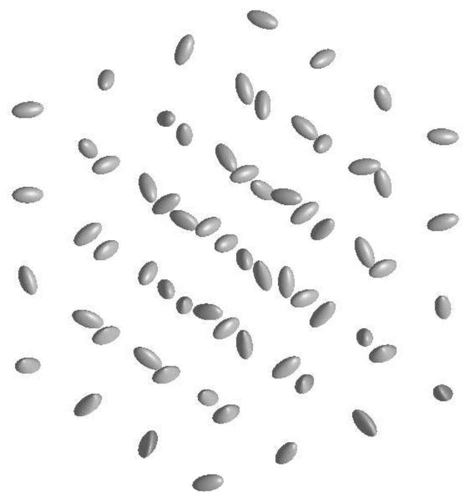


Figure 5.8: Domain contains 64 randomly oriented ellipsoids, where the minimal distance between any two is 6.0.

5.6.3.3 Example 7

The accelerated scheme is applied to solve the Helmholtz equation on the domain in Figure 5.6. Each scatterer is two wavelengths in diameter. The results obtained when solving the original $np \times np$ system are given in Table 5.11, and the ones from the small $kp \times kp$ system are given in Table 5.12. The tables substantiate our claim regarding the efficiency of the acceleration scheme. Note that in Table 5.11, due to limitation of the memory, only estimations of the run time are reported when four million discretization nodes were used.

N	n	T_{init}	I	T_{solve}	E_{∞}^{rel}
80 000	50×25	4.41e-01	28	3.60e+03	7.009e-03
160 000	100×25	8.44e-01	28	5.69e+03	5.755e-03
163 200	50×51	8.22e-01	28	5.78e+03	1.239e-04
326 400	100×51	1.65e+00	29	8.75e+03	4.806e-05
652 800	200×51	3.36e+00	29	1.54e+04	5.552e-05
323 200	50×101	1.58e+00	29	8.64e+03	8.223e-06
646 400	100×101	3.24e+00	29	1.69e+04	1.354e-07
1 292 800	200×101	6.67e+00	29	3.01e+04	2.823e-08

Table 5.9: (Example 6) Results from solving an exterior Helmholtz problem on the domain in Figure 5.8 with $p = 64$ scatterers without compression (the system with the full $np \times np$ coefficient matrix is solved). Each ellipsoid is 5 wavelengths in diameter.

N	n	$N_{\text{compressed}}$	k	$T_{\text{compressed}}$	I	T_{solve}	E_{∞}^{rel}
80 000	50×25	61 184	956	1.92e+01	28	4.42e+03	2.339e-02
160 000	100×25	75 648	1182	6.58e+01	29	4.79e+03	8.656e-03
163 200	50×51	87 744	1371	8.50e+01	29	4.92e+03	2.798e-04
326 400	100×51	100 288	1567	2.83e+02	30	5.25e+03	5.892e-05
652 800	200×51	105 216	1644	9.06e+02	30	5.51e+03	6.056e-05
323 200	50×101	91 648	1432	2.40e+02	30	5.09e+03	9.485e-06
646 400	100×101	102 400	1552	8.55e+02	31	5.50e+03	2.150e-07
1 292 800	200×101	106 944	1671	2.91e+03	31	5.73e+03	8.441e-08

Table 5.10: (Example 6) Results from solving an exterior Helmholtz problem on the domain in Figure 5.8 using the accelerated scheme. Each ellipsoid is 5 wavelengths in diameter.

N	n	T_{init}	I	T_{solve}	E_{∞}^{rel}
252 500	50×101	5.33e+00	50	1.01e+04	3.211e-03
505 000	100×101	1.07e+01	50	2.04e+04	2.260e-03
1 010 000	200×101	2.21e+01	51	4.16e+04	8.211e-04
502 500	50×201	1.02e+01	51	2.15e+04	8.273e-03
1 005 000	100×201	2.01e+01	51	4.20e+04	3.914e-03
2 010 000	200×201	3.90e+01	51	8.42e+04	5.044e-06
4 020 000	400×201	–	–	$\sim$ 48h	–
2 005 000	100×401	3.89e+01	51	8.30e+04	4.244e-04
4 010 000	200×401	–	–	$\sim$ 48h	–

Table 5.11: (Example 7) Results from solving the exterior Helmholtz problem on the domain in Figure 5.6 with $p = 50$ scatterers, using the full $np \times np$ coefficient matrix (no compression). Each scatterer is 2 wavelengths in diameter.

N	n	$N_{\text{compressed}}$	(k_a, k_b, k_c)	$T_{\text{compressed}}$	I	T_{solve}	E_{∞}^{rel}
252 500	50×101	53 390	(775,1254,1211)	2.26e+02	52	2.48e+03	4.941e-03
505 000	100×101	57 934	(823.1358,1337)	5.17e+02	53	2.72e+03	2.026e-03
1 010 000	200×101	60 512	(856,1420,1399)	1.14e+03	54	2.89e+03	4.865e-04
502 500	50×201	54 538	(789,1283,1238)	4.89e+02	53	2.63e+03	9.276e-03
1 005 000	100×201	59 036	(838,1384,1363)	1.10e+03	54	2.90e+03	4.392e-03
2 010 000	200×201	61 488	(872,1443,1419)	2.70e+03	56	3.10e+03	7.709e-06
4 020 000	400×201	61 664	(888,1428,1427)	1.50e+04	57	3.31e+03	1.856e-06
2 005 000	100×401	60 106	(853,1409,1388)	2.58e+03	56	3.04e+03	9.632e-04
4 010 000	200×401	61 818	(885,1441,1427)	1.54e+04	57	3.32e+03	2.452e-07

Table 5.12: (Example 7) Results from solving an exterior Helmholtz problem on the domain in Figure 5.6 using the accelerated scheme with a reduced size coefficient matrix. Each scatterer is 2 wavelengths in diameter. The ranks k_a, k_b, and k_c, for each of the three types of scatterer is given.

5.7 Conclusions and Future work

We have presented a highly accurate numerical scheme for solving acoustic scattering problems on domains involving multiple scatterers in three dimensions, under the assumption that each scatterer is axisymmetric. The algorithm relies on a boundary integral equation formulation of the scattering problem, combined with a highly accurate Nyström discretization technique. For each scatterer, a scattering matrix is constructed via an explicit inversion scheme. Then these individual scattering matrices are used as a block-diagonal pre-conditioner to GMRES to solve the very large system of linear equations. The Fast Multiple Method is used to accelerate the evaluation of all inter-body interactions. Numerical experiments show that while the block-diagonal pre-conditioner does not make almost any different for "zero-frequency" scattering problems (governed by Laplace's equation), it dramatically improves the convergence speed at intermediate frequencies.

Furthermore, for problems where the scatterers are well-separated, we present an accelerated scheme capable of solving even very large scale problems to high accuracy on a basic personal work station. In one numerical example in Section 5.6, the numbers of degrees of freedom required to solve the Laplace equation to eight digits of accuracy on a complex geometry could be reduced by a factor of 57 resulting in a reduction of the total computation time from 35 minutes to 10 minutes (9 minutes for compression and 42 seconds for solving the linear system). For a Helmholtz problem the reduction of computation time is even more significant: the numbers of degrees of freedom to reach seven digits of accuracy was in one example reduced by a factor of 65; consequently the overall computation time is reduced from 48 hours to 5 hours (4 hours for compression and 1 hour for solving the linear system).

The scheme presented assumes that each scatterer is rotationally symmetric; this property is used both to achieve higher accuracy in the discretization, and to accelerate all computations (by using the FFT in the azimuthal direction). It appears conceptually straight-forward to use the techniques of [12, 59] to generalize the method presented to handle scatterers with edges (generated by "corners" in the generating curve). The idea is to use local refinement to resolve the singular behavior of solutions near the corner, and then eliminate the added “superfluous” degrees of freedom added by the refinement via a local compression technique, see [34].

Chapter 6

A composite spectral scheme for 2D variable coefficient elliptic PDEs with local mesh refinement

6.1 Introduction

The chapter describes a numerical method for solving boundary value problems in situations where non-uniform grids are necessary due to change of regularity of the solution across the domain. We focus on the problems of the form

$$\begin{cases} -\nabla(c(\boldsymbol{x})\nabla u(\boldsymbol{x})) + b(\boldsymbol{x})\,u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases} \tag{1}$$

where Ω is domain in the plane with boundary $\Gamma = \partial\Omega$, where b and c are smooth, where $c > 0$ and $b \geq 0$, and where f is a given function that can be either smooth or non-smooth.

6.1.1 Outline of solution procedure

We solve (1) by combining the spectral discretization, hierarchical direct solve and local mesh refinement scheme. Basically, there are three steps:

(1) Discretization: The domain is split into small rectangular patches. On each patch, the solution u to (1) is approximated via tabulation on a tensor product grid of Chebyshev points to high-order accuracy. The elliptic operator in (1) is then approximated via spectral differential matrix defined on each local grid.

(2) Local mesh refinement: If the positions required for refinement is known in advance, we partition the patches containing those points into smaller patches based on how close these points are to the four corners of the holding patches.

(3) Direct solver: To solve the linear system hierarchically, we build a hierarchical tree by organizing the patches in binary way. The direct solver involves two way paths along the tree, upwards and downwards:

- Upwards pass: we construct the local solution operator and Dirichlet-to-Neumann (DtN) operator for each leaf patch. For a parent patch, the solution operator and DtN operator are constructed by glueing together the DtN operators of its children.
- Downwards pass: we take a vector of Dirichlet boundary data as input and construct the tabulated values of u at all internal grid points via local solution operator, moving downwards through the hierarchical tree.

6.1.2 Outline of the chapter

This chapter is organized as follows: Section 6.2 introduces notations and spectral interpolation and differentiation. Section 7.3 describes how to compute the solution operator and the DtN operator for a leaf in the tree. Section 7.4.1 describes how the DtN operator for a larger patch consisting of two small patches can be computed if the DtN operators for the smaller patches are given. Section 6.5 describes the full hierarchical scheme. Implementation details of mesh refinement are given in Section 6.6. In Section 7.7 we illustrate the performance of the local refinement scheme with a variety of numerical examples.

6.2 Spectral discretization

This section introduces notations for spectral differentiation on tensor product grids of Chebyshev nodes on the square domain $[-a,\, a]^2$. This material is classical, see, e.g., Trefethen [94].

Let p denote a positive integer. The *Chebyshev* nodes on $[-a,\, a]$ are points

$$t_i = a\,\cos\left((i-1)\pi/(p-1)\right), \quad i = 1, 2, \ldots, p.$$

Let $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$ denote the set of points of the form $(t_i,\, t_j)$ for $1 \leq i, j \leq p$. Let $\mathbb{P}_p$ denote the linear space of sums of tensor products of polynomials of degree $p-1$ or less. $\mathbb{P}_p$ has dimension p^2. Given a vector $\mathbf{u} \in \mathbb{R}^{p^2}$, there is a unique function $u \in \mathbb{P}_p$ such that $u(\boldsymbol{x}_k) = \mathbf{u}(k)$ for $1 \leq k \leq p^2$. We next define $\mathsf{D}^{(1)}$, $\mathsf{D}^{(2)}$ and L as the matrix equivalents of the differentiation operators $\partial/\partial x_1$, $\partial/\partial x_2$, and $-\Delta$, respectively. To be precise, these $p^2 \times p^2$ matrices are the unique matrices for which

$$[\mathsf{L}\mathbf{u}](k) = [-\Delta u](\boldsymbol{x}_k), \quad k = 1, 2, \ldots, p^2, \tag{2}$$

$$[\mathsf{D}^{(1)}\mathbf{u}](k) = [\partial_1 u](\boldsymbol{x}_k), \quad k = 1, 2, \ldots, p^2, \tag{3}$$

$$[\mathsf{D}^{(2)}\mathbf{u}](k) = [\partial_2 u](\boldsymbol{x}_k), \quad k = 1, 2, \ldots, p^2. \tag{4}$$

6.3 Leaf computation

This section describes the construction of a discrete approximation to the Dirichlet-to-Neumann operator associated with the an elliptic equation such as (1) for a square patch Ω. We in this section assume that the patch is small enough that it can readily be handled via a "single-domain" (non-composite) spectral method using a tensor product grid of Chebyshev nodes on Ω. In addition to the DtN operator, we also construct a solution operator to (1) that maps the Dirichlet data on the nodes on the boundary of Ω to the value of u at all internal interpolation nodes.

6.3.1 Notation and basic equilibrium conditions

Let Ω denote a square patch. Let $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$ denote the nodes in a tensor product grid of $p \times p$ Chebyshev nodes. Partition the index set

$$\{1, 2, \ldots, p^2\} = I_{\text{ext}} \cup I_{\text{int}}$$

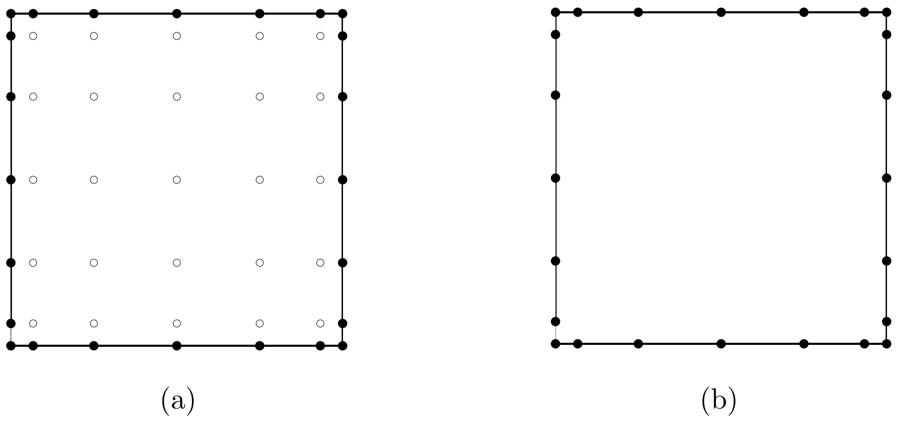


Figure 6.1: Notation for the leaf computation in Section 7.3. (a) A leaf before elimination of interior (white) nodes. (b) A leaf after elimination of interior nodes.

in such a way that I_{ext} contains all nodes on the boundary of Ω, and I_{int} denotes the set of interior nodes, see Figure 6.1. Let u be a function that satisfies (1) on Ω and let $\tilde{\mathbf{u}} \in \mathbb{R}^{p^2}$ denote a vector holding tabulated values of u on nodes $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$, in other words

$$\tilde{\mathbf{u}} = [u(\boldsymbol{x}_k)]_{k=1}^{p^2}.$$

The operator (1) is then locally approximated via the $p^2 \times p^2$ matrix

$$\mathsf{A} = -\mathsf{D}^{(1)}\mathsf{C}\mathsf{D}^{(1)} - \mathsf{D}^{(2)}\mathsf{C}\mathsf{D}^{(2)} + \mathsf{B}, \tag{5}$$

where B and C are diagonal matrices with diagonal entries $\{b(\boldsymbol{x}_k)\}_{k=1}^{p^2}$ and $\{c(\boldsymbol{x}_k)\}_{k=1}^{p^2}$. The equation we enforce on Ω is that $\mathsf{A}\tilde{\mathbf{u}}$ should evaluate to zero at all internal nodes, i.e.

$$\mathsf{A}_{\text{int,int}}\,\tilde{\mathbf{u}}_{\text{int}} + \mathsf{A}_{\text{int,ext}}\,\tilde{\mathbf{u}}_{\text{ext}} = 0, \tag{6}$$

where

$$\mathsf{A}_{\text{int,int}} = \mathsf{A}(I_{\text{int}}, I_{\text{int}}), \quad \mathsf{A}_{\text{int,ext}} = \mathsf{A}(I_{\text{int}}, I_{\text{ext}}).$$

Solving (6) for $\tilde{\mathbf{u}}_{\text{i}}$, we obtain

$$\tilde{\mathbf{u}}_{\text{int}} = \mathsf{U}\,\tilde{\mathbf{u}}_{\text{ext}}, \tag{7}$$

where

$$\mathsf{U} = -(\mathsf{A}_{\text{int,int}})^{-1}\,\mathsf{A}_{\text{int,ext}}. \tag{8}$$

U is defined as the solution operator, which is of size $(p-1)^2 \times (4p-4)$.

6.3.2 Constructing the DtN operator

We will next construct a matrix that maps given Dirichlet conditions on the boundary of the square Ω to fluxes on the boundary. As a first step, we define vectors $\{\tilde{\mathbf{v}}_{\text{s}}, \tilde{\mathbf{v}}_{\text{e}}, \tilde{\mathbf{v}}_{\text{n}}, \tilde{\mathbf{v}}_{\text{w}}\}$ that each hold the boundary fluxes on the four sides (south, east, north, west). To be precise

$$\tilde{\mathbf{v}}_{\text{s}} = \{\partial_2 u(\boldsymbol{x}_k)\}_{k\in I_{\text{s}}},\ \tilde{\mathbf{v}}_{\text{e}} = \{\partial_1 u(\boldsymbol{x}_k)\}_{k\in I_{\text{e}}},\ \tilde{\mathbf{v}}_{\text{n}} = \{\partial_2 u(\boldsymbol{x}_k)\}_{k\in I_{\text{n}}},\ \tilde{\mathbf{v}}_{\text{w}} = \{\partial_1 u(\boldsymbol{x}_k)\}_{k\in I_{\text{w}}},$$

where I_{s}, I_{e}, I_{n}, I_{w}, are index vectors that each mark the p nodes on the respective sides. We aggregate these vectors into a $4p \times 1$ vector

$$\tilde{\mathbf{v}} = \begin{bmatrix} \tilde{\mathbf{v}}_{\text{s}} \\ \tilde{\mathbf{v}}_{\text{e}} \\ \tilde{\mathbf{v}}_{\text{n}} \\ \tilde{\mathbf{v}}_{\text{w}} \end{bmatrix}.$$

Note that $\tilde{\mathbf{v}}$ represents an *outgoing flux* on certain boxes and an *incoming flux* on others. This is a deliberate choice to avoid problems with signs when matching fluxes of touching boxes. Furthermore, note that each corner node contributes to *two* entries in $\tilde{\mathbf{v}}$. We define the short-hands

$$\tilde{\mathbf{u}}_{\text{int}} = \tilde{\mathbf{u}}(I_{\text{int}}) \quad \text{and} \quad \tilde{\mathbf{u}}_{\text{ext}} = \tilde{\mathbf{u}}(I_{\text{ext}}).$$

With L, $\mathsf{D}^{(1)}$ and $\mathsf{D}^{(2)}$ denoting spectral differentiation matrices corresponding to the operators $-\Delta$, ∂_1 and ∂_2 respectively, as defined in Section 6.2, we use the short-hands

$$\mathsf{D}^{(k)}_{\text{int,ext}} = \mathsf{D}^{(k)}(I_{\text{int}}, I_{\text{ext}}), \quad \text{for } k = 1, 2$$

to denote the part of the differentiation matrix $\mathsf{D}^{(1)}$ and $\mathsf{D}^{(2)}$ that map exterior nodes to interior nodes, etc.

We are now ready to construct a matrix V of size $4p \times 4(p-1)$ that maps Dirichlet data to Neumann data, so that

$$\tilde{\mathbf{v}} = \mathsf{V}\,\tilde{\mathbf{u}}_{\text{ext}}. \tag{9}$$

Conceptually we proceed as follows: Given the potential $\tilde{\mathbf{u}}_{\text{ext}}$ on the boundary, we reconstruct $\tilde{\mathbf{u}}_{\text{int}}$ in the interior via (7). Since the potential is then known on all Chebyshev nodes in Ω, we can determine the gradient on each boundary via spectral differentiation on the entire domain. To formalize, the gradient on the south boundary can be written

$$\tilde{\mathbf{v}}_{\text{s}} = \mathsf{D}^{(2)}_{\text{s,ext}}\tilde{\mathbf{u}}_{\text{ext}} + \mathsf{D}^{(2)}_{\text{s,int}}\tilde{\mathbf{u}}_{\text{int}} = \mathsf{D}^{(2)}_{\text{s,ext}}\tilde{\mathbf{u}}_{\text{ext}} + \mathsf{D}^{(2)}_{\text{s,int}}\mathsf{U}\,\tilde{\mathbf{u}}_{\text{ext}}. \tag{10}$$

The gradient on the other sides can be determined similarly. In this way we define a DtN operator V of size $4p \times (4p-4)$ via, cf. (9),

$$\mathsf{V} = \begin{bmatrix} \mathsf{D}^{(2)}_{\text{s,ext}} + \mathsf{D}^{(2)}_{\text{s,int}}\mathsf{U} \\ \mathsf{D}^{(1)}_{\text{e,ext}} + \mathsf{D}^{(1)}_{\text{e,int}}\mathsf{U} \\ \mathsf{D}^{(2)}_{\text{n,ext}} + \mathsf{D}^{(2)}_{\text{n,int}}\mathsf{U} \\ \mathsf{D}^{(1)}_{\text{w,ext}} + \mathsf{D}^{(1)}_{\text{w,int}}\mathsf{U}. \end{bmatrix} \tag{11}$$

6.3.3 Interpolating the solution and DtN operators to Gaussian nodes

In practical implementations, we prefer to tabulate the boundary solution u on *Gaussian nodes* instead of on Chebyshev nodes. Using Gaussian nodes makes the computation much easier when we merge two boxes into a bigger one since the Gaussian nodes do not lie on the corners of the boxes.

In addition to the $p \times p$ tensor product Chebyshev nodes $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$ on Ω that we defined in Section 6.3.1, we now also introduce a vector $\{\mathbf{z}_\ell\}_{\ell=1}^{4q}$ of $4q$ Gaussian discretization nodes lying on the four edges of Ω. The sets of boundary Gaussian and Chebyshev nodes are both ordered in a counter-clockwise fashion, starting from the “southwest” order.

Let P_{C2G} and P_{G2C} denote the standard interpolation operators that map between a set of p Chebyshev nodes and a set of q Gaussian nodes, respectively. In other words, P_{C2G} and P_{G2C} are the unique matrices such that

$$[\phi(\mathbf{z}_i)]_{i=1}^{q} = \mathsf{P}_{\text{C2G}}\,[\phi(\mathbf{x}_j)]_{j=1}^{p}, \qquad \text{for every polynomial } \phi \text{ of degree at most } p-1$$

$$[\phi(\mathbf{x}_i)]_{i=1}^{p} = \mathsf{P}_{\text{G2C}}\,[\phi(\mathbf{z}_j)]_{j=1}^{q}, \qquad \text{for every polynomial } \phi \text{ of degree at most } q-1.$$

The idea is now to build large interpolation operators L_{C2G} and L_{G2C} that interpolate between the $4q$ boundary Gaussian nodes and the $4(p-1)$ boundary Chebyshev nodes, and then form the “Gauss-to-Gauss” DtN matrix via

$$\underset{4q\times 4q}{\mathsf{T}} = \underset{4q\times 4p}{\mathsf{L}_{\text{C2G}}} \circ \underset{4p\times 4(p-1)}{\mathsf{V}} \circ \underset{4(p-1)\times 4q}{\mathsf{L}_{\text{G2C}}}, \tag{12}$$

where V is the "Chebyshev-to-Chebyshev" DtN matrix defined in Section 7.3.2. The $4q \times 4p$ matrix $\mathsf{L}_{\rm C2G}$ is obtained simply by forming the block-diagonal matrix

$$\mathsf{L}_{\rm C2G} = \begin{bmatrix} \mathsf{P}_{\rm C2G} & & & \\ & \mathsf{P}_{\rm C2G} & & \\ & & \mathsf{P}_{\rm C2G} & \\ & & & \mathsf{P}_{\rm C2G} \end{bmatrix}. \tag{13}$$

Building the $4(p-1) \times 4q$ matrix $\mathsf{L}_{\rm G2C}$ requires us to decide how to assign values to the 4 corner nodes. For instance, the node on the "southest" corner can be assigned a value either by interpolation from the q Gaussian nodes on the south side, or from the q Gaussian nodes on the east side. (To be precise, these are *extrapolations*, but only by a tiny amount, and quite stable.) We chose to form the average of these two values, which leads to the definition

$$\mathsf{L}_{\rm G2C} = \left[\begin{array}{c|c|c|c} \frac{1}{2}\mathsf{P}_{\rm G2C}(1,:) & & & \frac{1}{2}\mathsf{P}_{\rm G2C}(p,:) \\ \hline \mathsf{P}_{\rm G2C}(2:(p-1),:) & & & \\ \hline \frac{1}{2}\mathsf{P}_{\rm G2C}(p,:) & \frac{1}{2}\mathsf{P}_{\rm G2C}(1,:) & & \\ \hline & \mathsf{P}_{\rm G2C}(2:(p-1),:) & & \\ \hline & \frac{1}{2}\mathsf{P}_{\rm G2C}(p,:) & \frac{1}{2}\mathsf{P}_{\rm G2C}(1,:) & \\ \hline & & \mathsf{P}_{\rm G2C}(2:(p-1),:) & \\ \hline & & \frac{1}{2}\mathsf{P}_{\rm G2C}(p,:) & \frac{1}{2}\mathsf{P}_{\rm G2C}(1,:) \\ \hline & & & \mathsf{P}_{\rm G2C}(2:(p-1),:) \end{array}\right]. \tag{14}$$

Finally, we define for future reference a solution operator S of size $p^2 \times 4q$ that maps Dirichlet data specified on the Gaussian nodes, to a solution defined on the Chebyshev nodes on the patch. This map is easily formed by combining the big interpolation matrix $\mathsf{L}_{\rm G2C}$ with the Chebyshev solution operator U defined in Section 6.3.1,

$$\underset{p^2\times 4q}{\mathsf{S}} = \underset{p^2\times 4(p-1)}{\mathsf{U}} \circ \underset{4(p-1)\times 4q}{\mathsf{L}_{\mathsf{G2C}}}. \tag{15}$$

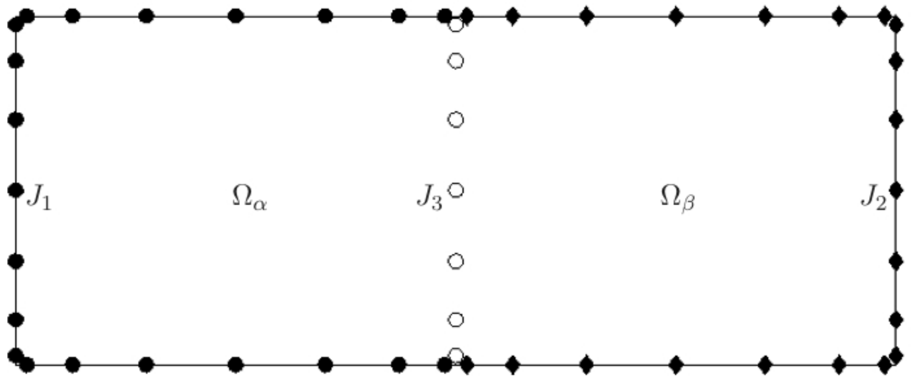


Figure 6.2: Notation for the merge operation described in Section 7.4.1. The rectangular domain Ω is formed by two squares Ω_α and Ω_β. The sets J_1 (boundary nodes of Ω_α that are not boundary nodes of Ω_β) and J_2 (boundary nodes of Ω_β that are not boundary nodes of Ω_α) form the exterior nodes (black), while J_3 (boundary nodes of both Ω_α and Ω_β that are not boundary nodes of the union box) consists of the interior nodes (white).

6.4 The merge operation

Let Ω denote a rectangular domain consisting of the union of the two smaller rectangular domains,

$$\Omega = \Omega_\alpha \cup \Omega_\beta,$$

as shown in Figure 7.1. Moreover, suppose that approximations to the DtN operators for Ω_α and Ω_β are available, represented as matrices T^α and T^β that map boundary values of u to boundary values of $\partial_1 u$ and $\partial_2 u$. This section describes how to compute a solution operator S that maps the value of a function u that is tabulated on the boundary of Ω to the value of u on interpolation nodes on the internal boundary, as well as DtN operators T that maps boundary values of u on the boundary of Ω to values of the $\partial_1 u$ and $\partial_2 u$ tabulated on the boundary.

6.4.1 Notation

We start with introducing some notations: Let Ω denote a box with children Ω_α and Ω_β. For concreteness, let us assume that Ω_α and Ω_β share a vertical edge. We partition the points on $\partial\Omega_\alpha$ and $\partial\Omega_\beta$ into three sets:

J_1	Boundary nodes of Ω_α that are not boundary nodes of Ω_β.
J_2	Boundary nodes of Ω_β that are not boundary nodes of Ω_α.
J_3	Boundary nodes of both Ω_α and Ω_β that are *not* boundary nodes of the union box Ω.

Figure 7.1 illustrates the definitions of the J_j's. Let u denote a solution to (1), with tabulated values $\mathbf{u}_j$ and boundary fluxes $\mathbf{v}_j$ restricted to the nodes in the set "j". Set

$$\mathbf{u}_{\mathrm{i}} = \mathbf{u}_3, \qquad \text{and} \qquad \mathbf{u}_{\mathrm{e}} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}. \tag{16}$$

Finally, let T^α and T^β denote the operators that map values of the potential values on the boundaries to values of $\partial_1 u$ or $\partial_2 u$ on the boundaries of the boxes Ω_α and Ω_β, as described in Section 7.3.2.

We partition these matrices according to the numbering of nodes in Figure 7.1,

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathsf{T}^{\alpha}_{1,1} & \mathsf{T}^{\alpha}_{1,3} \\ \mathsf{T}^{\alpha}_{3,1} & \mathsf{T}^{\alpha}_{3,3} \end{bmatrix} \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_3 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathsf{v}_2 \\ \mathsf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathsf{T}^{\beta}_{2,2} & \mathsf{T}^{\beta}_{2,3} \\ \mathsf{T}^{\beta}_{3,2} & \mathsf{T}^{\beta}_{3,3} \end{bmatrix} \begin{bmatrix} \mathsf{u}_2 \\ \mathsf{u}_3 \end{bmatrix}. \tag{17}$$

6.4.2 Equilibrium condition

Suppose that we are given a tabulation of boundary values of a function u that satisfies (1) on Ω. In other words, we are given the vectors u_1 and u_2. We can then reconstruct the values of the potential on the interior boundary (tabulated in the vector u_3) by using information in (17). Simply observe that there are two equations specifying the normal derivative across the internal boundary (tabulated in v_3), and combine these equations:

$$\mathsf{T}^{\alpha}_{3,1}\mathsf{u}_1 + \mathsf{T}^{\alpha}_{3,3}\mathsf{u}_3 = \mathsf{v}_3 = \mathsf{T}^{\beta}_{3,2}\mathsf{u}_2 + \mathsf{T}^{\beta}_{3,3}\mathsf{u}_3. \tag{18}$$

Solving (18) for u_3 we get

$$\mathsf{u}_3 = \big(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\big)^{-1}\big(\mathsf{T}^{\beta}_{3,2}\mathsf{u}_2 - \mathsf{T}^{\alpha}_{3,1}\mathsf{u}_1\big). \tag{19}$$

Now set

$$\mathsf{S} = \big(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\big)^{-1}\big[-\mathsf{T}^{\alpha}_{3,1} \mid \mathsf{T}^{\beta}_{3,2}\big], \tag{20}$$

to find that (19) is (in view of (16)) precisely the desired formula

$$\mathsf{u}_{\rm i} = \mathsf{S}\,\mathsf{u}_{\rm e}. \tag{21}$$

6.4.3 Constructing the DtN operators for the union box

Our next object is to construct a DtN matrix T such that

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_2 \end{bmatrix} = \mathsf{T} \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \end{bmatrix}. \tag{22}$$

To this end, observe from (17) that

$$\mathsf{v}_1 = \mathsf{T}^{\alpha}_{1,1}\mathsf{u}_1 + \mathsf{T}^{\alpha}_{1,3}\mathsf{u}_3, \tag{23}$$

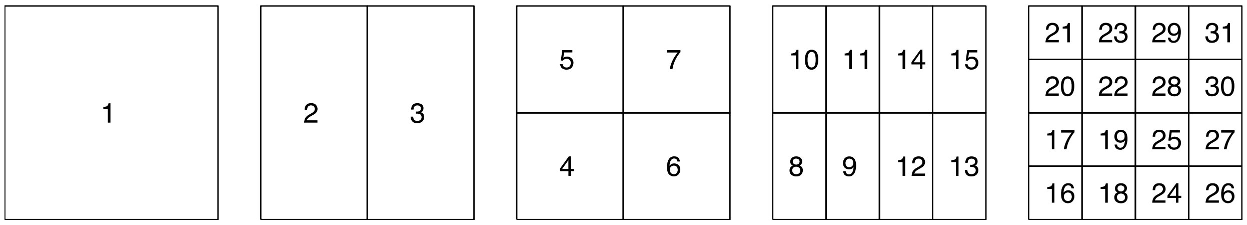


Figure 6.3: The square domain Ω is split into 4×4 leaf boxes. These are then gathered into a binary tree of successively larger boxes as described in Section 6.5.1. One possible enumeration of the boxes in the tree is shown, but note that the only restriction is that if box τ is the parent of box σ, then $\tau < \sigma$.

$$\mathsf{v}_2 = \mathsf{T}^{\beta}_{2,3}\mathsf{u}_2 + \mathsf{T}^{\beta}_{2,3}\mathsf{u}_3. \tag{24}$$

By eliminating u_3 from the above two equations using (19), we obtain

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_2 \end{bmatrix} = \left(\begin{bmatrix} \mathsf{T}^{\alpha}_{1,1} & 0 \\ 0 & \mathsf{T}^{\beta}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\alpha}_{1,3} \\ \mathsf{T}^{\beta}_{2,3} \end{bmatrix} \left(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\right)^{-1} \left[-\mathsf{T}^{\alpha}_{3,1} \mid \mathsf{T}^{\beta}_{3,2}\right] \right) \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \end{bmatrix}. \tag{25}$$

In other words,

$$\mathsf{T} = \begin{bmatrix} \mathsf{T}^{\alpha}_{1,1} & 0 \\ 0 & \mathsf{T}^{\beta}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\alpha}_{1,3} \\ \mathsf{T}^{\beta}_{2,3} \end{bmatrix} \left(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\right)^{-1} \left[-\mathsf{T}^{\alpha}_{3,1} \mid \mathsf{T}^{\beta}_{3,2}\right]. \tag{26}$$

6.5 The full hierarchical scheme

At this point, we know how the construct the solution and DtN operators for a leaf patch (Section 7.3), and how to merge two DtN operators of neighboring patches to form the DtN operator of their union (Section 7.4.1). We are ready to describe the full hierarchical scheme for solving the Dirichlet problem (1). We start by describing the scheme for a uniform mesh, and will then in Section 6.6 proceed to describe the modifications required to handle locally refined meshes.

6.5.1 The algorithm

To begin with, we partition the domain Ω into a collection of rectangular boxes, called *leaf* boxes. These should be small enough that a small spectral mesh with $p \times p$ nodes (for, say, $p = 20$) accurately interpolates both any potential solution u of (1) and its derivatives $\partial_1 u$, $\partial_2 u$, and $-\Delta u$. Let vector $\mathbf{u}$ denote the tabulated approximations to u at the Gaussian nodes $\{\mathbf{z}_\ell\}_{\ell=1}^N$ on all interior edges. Next construct a binary tree on the collection of boxes by hierarchically merging them, making sure that all boxes on the same level are roughly of the same size, cf. Figure 6.3. The boxes should be ordered so that if τ is a parent of a box σ, then $\tau < \sigma$. We also assume that the root of the tree (i.e. the full box Ω) has index $\tau = 1$.

With each box τ, we define two index vectors I_i^τ and I_e^τ as follows:

I_e^τ A list of all indices of the nodes on the boundary of τ.

I_i^τ For a leaf τ, I_i^τ is a list of all interior nodes of τ.

For a parent τ, I_i^τ is a list of all its interior nodes that are not interior nodes of its children.

Next we execute a "build stage" in which for every box τ, we construct the following two matrices:

S^τ The matrix that maps the values of $\mathbf{u}$ on $\partial\Omega_\tau$ to the values of $\mathbf{u}$ on the interior nodes of Ω_τ. In other words, $\mathbf{u}(I_\mathrm{i}^\tau) = \mathsf{S}^\tau\,\mathbf{u}(I_\mathrm{e}^\tau)$.

T^τ The matrix that maps the values of $\mathbf{u}$ on $\partial\Omega_\tau$ to the values of $\mathbf{v}$ (tabulating du/dn) on $\partial\Omega_\tau$. In other words, $\mathbf{v}(I_\mathrm{e}^\tau) = \mathsf{T}^\tau\,\mathbf{u}(I_\mathrm{e}^\tau)$.

The build stage consists of a single sweep over all nodes in the tree. Any bottom-up ordering in which any parent box is processed after its children can be used. For each leaf box τ, an approximation to the local DtN map T^τ is constructed using the procedure described in Section 7.3. For a parent box τ with children σ_1 and σ_2, the matrices S^τ and T^τ are formed from the DtN operators T^{σ_1} and T^{σ_2} via the process described in Section 7.4.1. Figure 6.4 summarizes the build stage.

Once all the matrices $\{\mathsf{S}^\tau\}_\tau$ have been formed, the vector $\mathbf{u}$ holding approximations to the solution u of (1) can be constructed for all discretization points by starting at the root box Ω and moving down the tree toward the leaf boxes. The values of $\mathbf{u}$ for the points on the boundary of Ω can be obtained by tabulating the boundary function f. When any box τ is processed, the value of $\mathbf{u}$ is known for all nodes on its boundary (i.e. those listed in I_e^τ). The matrix S^τ directly maps these values to the values of $\mathbf{u}$ on the nodes in the interior of τ (i.e. those listed in $I_{\rm i}^{\tau}$). When all nodes have been processed, approximations to u have been constructed for all tabulation nodes on interior edges. Figure 6.5 summarizes the solve stage.

```
PRE-COMPUTATION

This program constructs the global Dirichlet-to-Neumann operator for (1).
It also constructs all matrices $\mathsf{S}^{\tau}$ required for constructing $u$ at any interior point.
It is assumed that if node $\tau$ is a parent of node $\sigma$, then $\tau < \sigma$.
________________________________________________________________

for $\tau = N_{\rm boxes}, N_{\rm boxes}-1, N_{\rm boxes}-2, \dots, 1$
    if ($\tau$ is a leaf)
        Construct $\mathsf{S}^{\tau}$ and $\mathsf{T}^{\tau}$ via the process described in Section 7.3.
    else
        Let $\sigma_1$ and $\sigma_2$ be the children of $\tau$.
        Partition $I_{\rm e}^{\sigma_1}$ and $I_{\rm e}^{\sigma_2}$ into vectors $I_1$, $I_2$, and $I_3$ as shown in Figure 7.1.
        $\mathsf{S}^{\tau} = \big(\mathsf{T}^{\sigma_1}_{3,3} - \mathsf{T}^{\sigma_2}_{3,3}\big)^{-1}\big[-\mathsf{T}^{\sigma_1}_{3,1} \,\big|\, \mathsf{T}^{\sigma_2}_{3,2}\big]$
        $\mathsf{T}^{\tau} = \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,1} & 0 \\ 0 & \mathsf{T}^{\sigma_2}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,3} \\ \mathsf{T}^{\sigma_2}_{2,3} \end{bmatrix} \mathsf{S}^{\tau}.$
        Delete $\mathsf{T}^{\sigma_1}$ and $\mathsf{T}^{\sigma_2}$.
    end if
end for
```

Figure 6.4: Pre-computation

6.6 Mesh refinement

Oftentimes, there are situations where a refinement algorithm is needed to retain the accuracy. For example, these situations can be when the boundary data is non-smooth, when the solution exhibits sharp gradient somewhere, or when the solutions develops local singularities around corners. To resolve these problems, a refinement criteria is demanding to identify the regions where the grid needs to be refined. In this section, we describe how the scheme described in Section 6.5.1 can be modified to allow for local refinement. We restrict attention to the case where the user specifies the refined grid. (The development of automatic grid refinement strategies is a work in progress.)

The core technical difficulty that we need to address is that upon refinement, the Gaussian interpolation nodes on the edges of domains no longer line up perfectly, which necessitates

```
Solver

This program constructs an approximation $\mathbf{u}$ to the solution $u$ of (1).
It assumes that all matrices $\mathsf{S}^{\tau}$ have already been constructed in a pre-computation.
It is assumed that if node $\tau$ is a parent of node $\sigma$, then $\tau < \sigma$.
________________________________________________________________

$\mathbf{u}(\ell) = f(\mathbf{z}_{\ell})$ for all $\ell \in I_{\mathrm{e}}^{1}$.
for $\tau = 1, 2, 3, \ldots, N_{\mathrm{boxes}}$
    $\mathbf{u}(I_{\mathrm{i}}^{\tau}) = \mathsf{S}^{\tau}\,\mathbf{u}(I_{\mathrm{e}}^{\tau})$.
end for
```

Figure 6.5: Solve stage.

modifications to the merge process described in Section 7.4.1.

6.6.1 Refinement criteria

Suppose that $(x_0,\, y_0)$ denotes a point where the solution exhibits local loss of regularity. For instance, $(x_0,\, y_0)$ could be a corner of the domain, or a point on the boundary where the prescribed boundary data is non-smooth. Our refinement criteria is then to split any box that contains the point $(x_0,\, y_0)$ itself into four equi-sized smaller boxes, unless this box is smaller than some preset minimum size L_{min}. In addition to splitting any box that contains the singular point, we also split any boxes that are "close". Specifically, for any box Ω_τ, let d denote the distance from (x_0, y_0) to Ω_τ, and let $2\,l_\tau$ denote the side-length of Ω_τ. We then split Ω_τ into four equi-sized boxes if

$$d < \alpha\, l_\tau$$

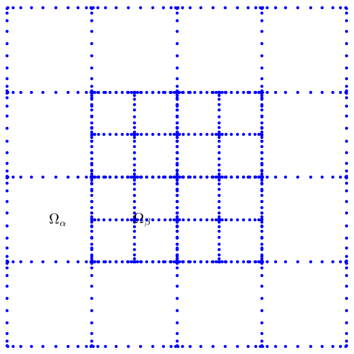
for some fixed parameter α. We proceed recursively, stopping while boxes reach the pre-scribed minimal size L_{min}. Empirically, we found that $\alpha = \sqrt{2}$ is often a good choice for the refinement parameter.

6.6.2 New data structures

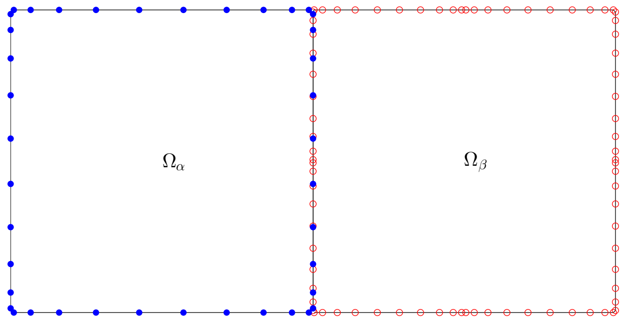
Note that after introducing refinement, any one edge can now have different, overlapping sets of nodes associated with refined boxes. Therefore, we need to introduce an extended set of Gaussian nodes $\{\boldsymbol{z}_i\}_{i=1}^N$ on all interior edges of Ω such that for a refined box Ω_τ^{ref}, it contains both the coarser Gaussian nodes on $\partial\Omega_\tau^{\text{ref}}$ as defined in Section 6.3.3 and the boundary nodes of its children. Let $\{\mathbf{u}_i\}_{i=1}^N$ be the vector of tabulated approximate values at these points so that

$$\mathbf{u}_i \approx u(\boldsymbol{z}_i) \quad \text{for} \quad i = 1, \ldots, N.$$

6.6.3 Interpolation of solution and DtN operators



(a)



(b)

Figure 6.6: (a) Grid with refinement at the center; (b) J_α denote the indices of Gaussian nodes on $\partial\Omega_\alpha$ denoted by blue dots, while J_β denote the indices of Gaussian nodes on $\partial\Omega_\beta$ denoted by red circles.

In this section, we describe the technique to handle the issues when the tabulation nodes for two touching boxes do not coincide. This typically happens when one of the two neighboring boxes is refined or when they have different levels of refinements. Figure 6.6 depicts an example involving two adjacent boxes Ω_α and Ω_β of equal size. One box, Ω_α, is a leaf node, while the other, Ω_β, was refined and has four children. On the shared edge of Ω_α there are q Gaussian nodes (in the figure, $q = 10$), while on Ω_β there are two panels with q Gaussian nodes each. To eliminate the troubles when merging such boxes, we keep the nodes on the shared edge consistent by applying interpolation operators. For a better illustration, we first introduce some notations.

Let J_α and J_β denote the boundary nodes of Ω_α and Ω_β respectively, which are subsets of the extended set $\{\mathbf{z}_i\}_{i=1}^N$. Assume on the shared edge of Ω_α and Ω_β there are s and t nodes respectively. For simplicity, we assume the number of nodes on the other edges of the two boxes are the same, say r nodes. Figure 6.6 illustrates the definitions of J_α and J_β. For this specific example we have $t > s$ since Ω_β contains refined nodes on the shared edge.

Recall that for uniform grid we define the solution operator in (15) which maps the potential value tabulated on exterior Gaussian nodes to interior Chebyshev nodes. For leaf boxes that do not touch any refined boxes, calculations of the solution operators remain the same. However for those boxes adjacent to refined boxes, the solution operator now maps the refined boundary nodes to interior Chebyshev nodes by applying an interpolation matrix. Like what we did in Section 6.3.3, we construct two interpolation operators P_1 and P_2 that map between nodes $\{\mathbf{z}_i\}_{i\in J_\alpha}$ and $\{\mathbf{z}_j\}_{j\in J_\beta}$. In other words,

$$[\phi(\mathbf{z}_i)]_{i\in J_\alpha} = \mathsf{P}_1\,[\phi(\mathbf{z}_j)]_{j\in J_\beta}, \qquad \text{for every polynomial } \phi \text{ of degree at most } t-1$$

$$[\varphi(\mathbf{z}_j)]_{j\in J_\beta} = \mathsf{P}_2\,[\varphi(\mathbf{z}_i)]_{i\in J_\alpha}, \qquad \text{for every polynomial } \varphi \text{ of degree at most } s-1.$$

Let L_1 be a 4×4 block matrix having P_1 and three identity matrices on the diagonal, such that

$$\underset{(s+r)\times 1}{\mathbf{u}(J_\alpha)} = \underset{(s+r)\times(t+r)}{\mathsf{L}_1}\;\underset{(t+r)\times 1}{\mathbf{u}(J_\beta)}\,. \tag{27}$$

Then the solution operator for Ω_α is computed via

$$\underset{p^2\times(t+r)}{\mathsf{S}^{\alpha,\text{ref}}} = \underset{p^2\times(s+r)}{\mathsf{S}^{\alpha}} \circ \underset{(s+r)\times(t+r)}{\mathsf{L}_1}\,. \tag{28}$$

The DtN operator can be constructed in a similar way by applying two interpolation matrices

$$\underset{(t+r)\times(t+r)}{\mathsf{T}^{\alpha,\text{ref}}} = \underset{(t+r)\times(s+r)}{\mathsf{L}_2} \circ \underset{(s+r)\times(s+r)}{\mathsf{T}^{\alpha}} \circ \underset{(s+r)\times(t+r)}{\mathsf{L}_1}\,, \tag{29}$$

where L_2 is a block matrix of size $(t+r)\times(s+r)$ such that

$$\underset{(t+r)\times 1}{\mathbf{u}(J_\beta)} = \underset{(t+r)\times(s+r)}{\mathsf{L}_2}\;\underset{(s+r)\times 1}{\mathbf{u}(J_\alpha)}\,. \tag{30}$$

Remark 10 In practice, standard Lagrange interpolation is unstable to approximate P_2 since s is usually a smaller number than t. Hence, we approximate P_2 by Barycentric Lagrange interpolation [7] which is more stable. But we stick to Lagrange interpolation to approximate P_1.

6.7 Numerical experiments

This section describes numerical experiments to explore the performance the refinement scheme outlined in previous sections. All the experiments are carried out on a desktop with 2.8GHz Intel Core 2 Duo and 12GB of RAM, and executed in a MATLAB environment.

The performance of the scheme are illustrated by solving variety of problems. In Section 6.7.1 we apply the refinement scheme to Helmholtz problems on square domain with non-smooth or non-continuous boundary data. Section 6.7.2 presents results from applying the scheme to solving problems on domains with corners such as a tunnel-shaped domain. In Section 6.7.3 and 6.7.4 we solve the variable coefficient problem and free space problem where the scattering term exhibits non-smoothness.

In all the experiments below, to check for convergence of the potential, we compute the solution at some chosen interior points $\hat{\boldsymbol{x}}$ pointwisely via

$$E_{\text{int}}^{N} = |u^{N}(\hat{\boldsymbol{x}}) - u^{2N}(\hat{\boldsymbol{x}})|,$$

where u^N denote the solution at the $N \times N$ Chebyshev grid. To check the convergence of the normal derivative on the boundary, we compute the total fluxes across the boundary of the domain via

$$E_{\text{flux}}^{N} = |f_N - f_{2N}|,$$

where $f_N = \sum_{\mathbf{z}_i \in \Gamma} v(\mathbf{z}_i)\omega_i$ with $v(\mathbf{z}_i)$ denoting the derivative of u evaluated at Gaussian nodes $\mathbf{z}_i$ on Γ and weights ω_i. Moreover, we fix the refinement parameter $\alpha = \sqrt{2}$ based on experimental performance.

6.7.1 Helmholtz equation on square domain with non-smooth boundary data

We first solve the Helmholtz equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases} \tag{31}$$

in domain $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Let the wavenumber $\kappa = 62.8$ such that there are 10×10 wavelength across the domain. In Example 1 and 2, we let the boundary data $f(\boldsymbol{x})$ to be non-smooth functions while in Example 3 and 4 $f(\boldsymbol{x})$ are set to be non-continuous functions.

In all these experiments, the number of Gaussian nodes per leaf box is fixed at $q = 20$, and 21×21 tensor product grids of Chebyshev nodes are used for leaf computations. The number of panels each side is fixed at $N_p = 8$.

6.7.1.1 Example 1

In this example, we increase the numbers of refinement positions. On each of the four sides of the square domain Ω, let the Dirichlet boundary data be given

$$f(x) = \begin{cases} |x-0.3|^{\beta} & x \leq 0.5, \\ |x-0.7|^{\beta} & x > 0.5. \end{cases} \tag{32}$$

Therefore there are totally 16 positions to be refined and the refinement grid with four levels of refinement is shown in Figure 6.7. In Figure 6.8 we plot the boundary data with respect to different values of β. We present the convergence of the error $\mathbf{u}$ and fluxes with respect to the number of levels of refinements in Figures 6.9.

6.7.1.2 Example 2

In this example, the boundary data on each of the four sides of the domain is given by

$$f(x) = |x-0.5| - 0.5, \tag{33}$$

such that the grid is refined at the center point of each side. In Figure 6.10 we plot the approximated solution as well as the Dirichlet boundary data. The convergence of error versus the levels of refinement is shown in Figure 6.11.

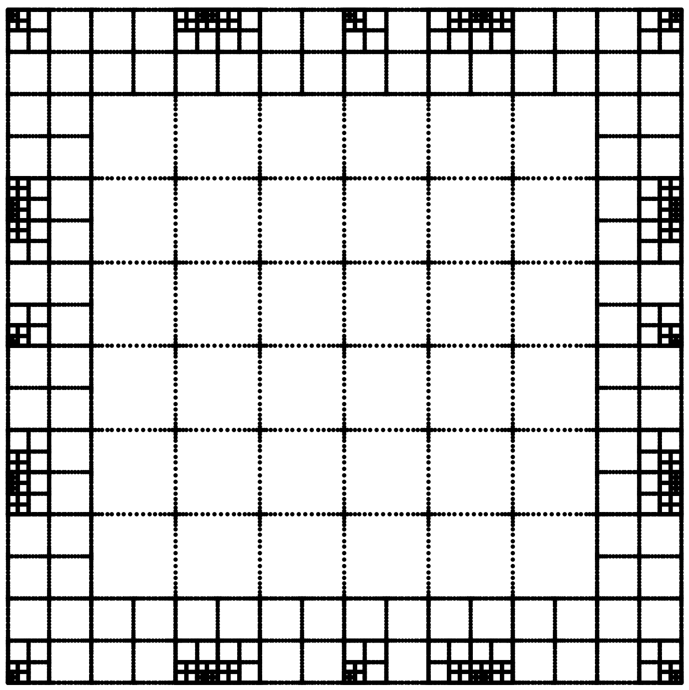
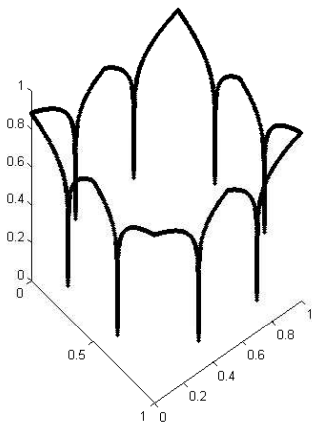
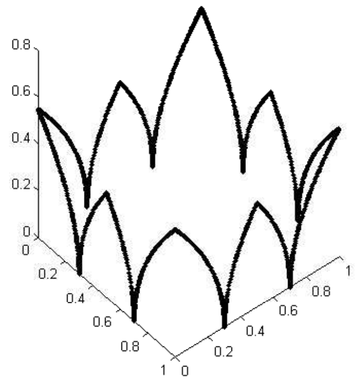


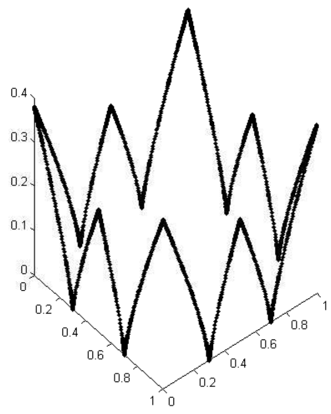
Figure 6.7: (Example 1) Grid of the exterior Gaussian nodes with four levels of refinement.



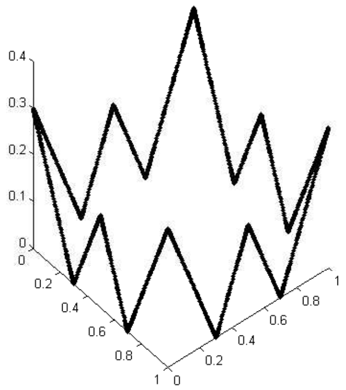
$\beta = 0.1$



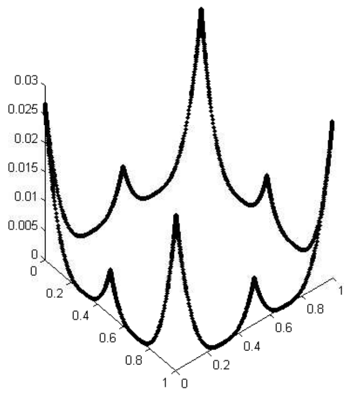
$\beta = 0.5$



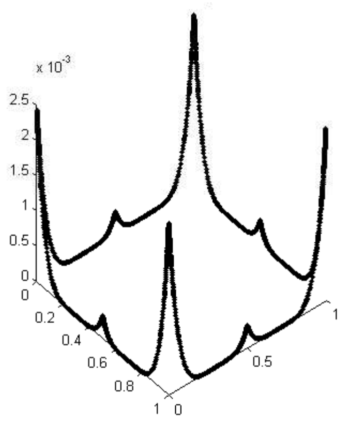
$\beta = 0.8$



$\beta = 1$

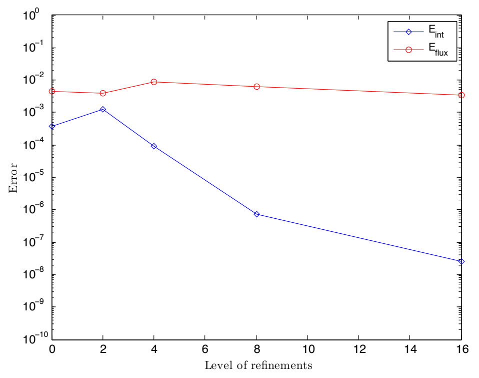


$\beta = 3$

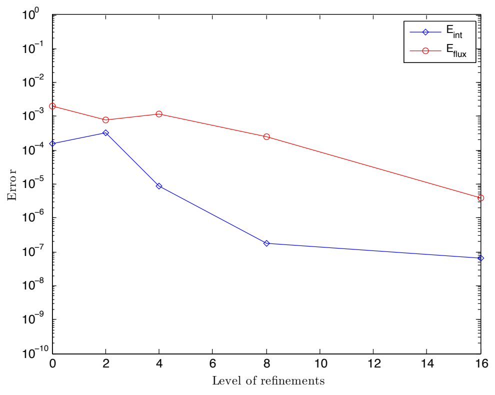


$\beta = 5$

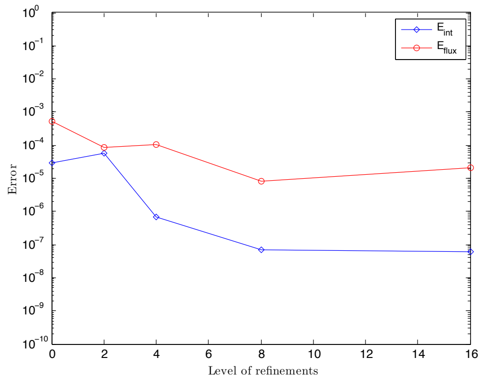
Figure 6.8: (Example 1) Dirichlet data $f(\boldsymbol{x})$ for various β given in (32).



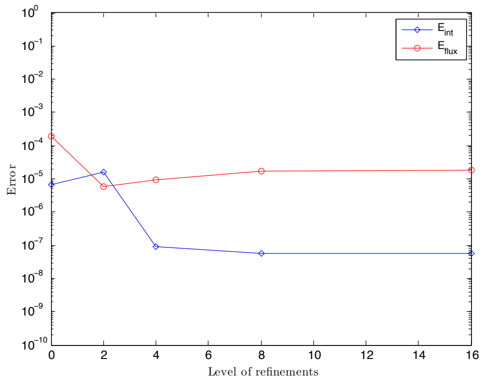
$\beta = 0.1$



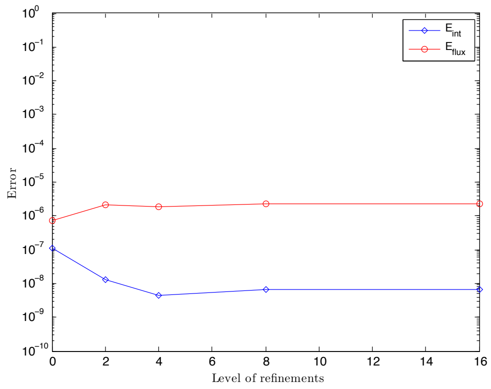
$\beta = 0.5$



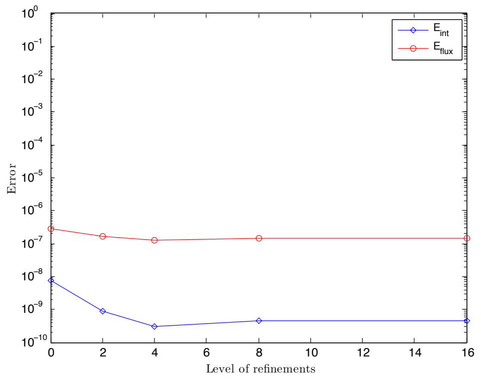
$\beta = 0.8$



$\beta = 1$



$\beta = 3$



$\beta = 5$

Figure 6.9: (Example 1) Convergence of the errors from solving the interior Helmholtz problem (31) on domain $\Omega = [0, 1]^2$ with boundary data given in (32) with respect to the number of levels of refinement.

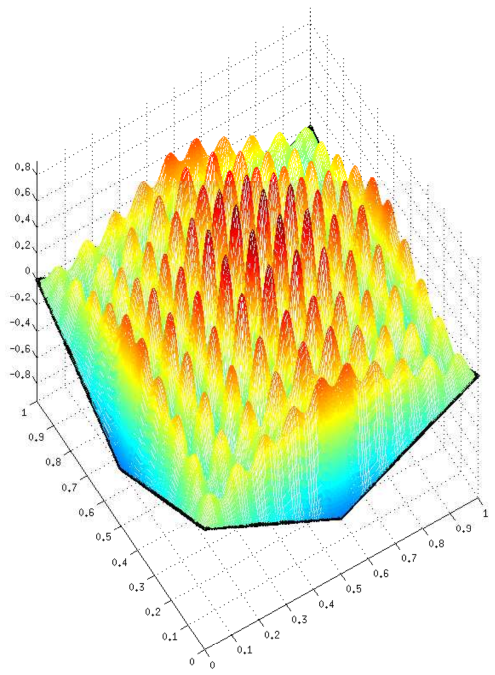


Figure 6.10: (Example 2) Approximated solution and Dirichlet boundary data given in (33).

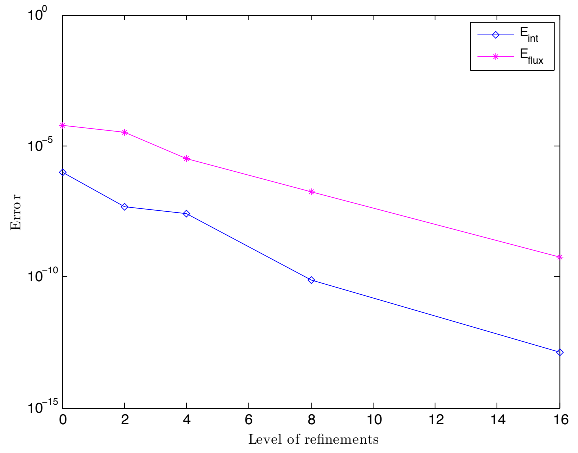


Figure 6.11: (Example 2) Convergence of the errors from solving the interior Helmholtz problem (31) on domain $\Omega = [0,\, 1]^2$ with boundary data given in (33) with respect to the number of levels of refinement.

6.7.1.3 Example 3

In this example, the Dirichlet condition on the boundary is not only non-smooth, but also non-continuous. Specifically, the boundary data of each side is given by the function

$$f(x) = \begin{cases} -x & 0 \leq x \leq 0.5, \\ 1 - x & 0.5 < x \leq 1, \end{cases} \tag{34}$$

such that the refinement is carried out at the center point of each side. In Figure 6.12 we plot the approximated solution as well as the Dirichlet boundary data. The convergence of error versus the levels of refinement is shown in Figure 6.13.

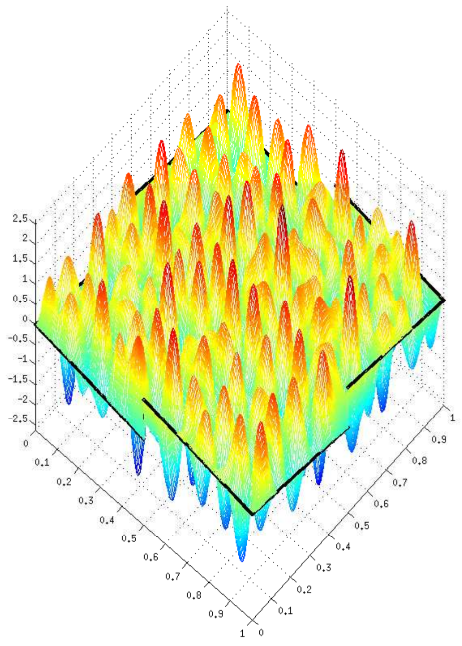


Figure 6.12: (Example 3) Approximated solution and Dirichlet boundary data given in (34).

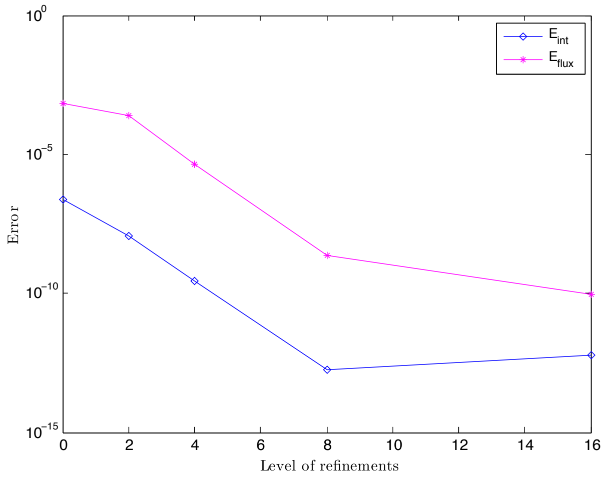


Figure 6.13: (Example 3) Convergence of the errors from solving the interior Helmholtz problem (31) on domain $\Omega = [0,\, 1]^2$ with boundary data given in (34) with respect to the number of levels of refinement.

6.7.2 Helmholtz equation on domain with corners

6.7.2.1 Example 4

In this section, we illustrate the performance of the refinement scheme by solving interior Helmholtz equation (31) on an on a tunnel-shaped domain as shown in Figure 6.14. The number of Gaussian nodes per leaf box is fixed at $q = 20$, and 21×21 tensor product grids of Chebyshev nodes are used for leaf computations. The number of panels each side is fixed at $N_p = 8$. The wavenumber is set as $\kappa = 3.37$ such at the domain is 15 wavelength in length. The pointwise errors were estimated at the interior points $\hat{\boldsymbol{x}} = (-5, 0)$. We pick the Dirichlet boundary data f that generates a sort of plane wave and is given by

$$f(\boldsymbol{x}) = \cos(34\, x_1). \qquad (35)$$

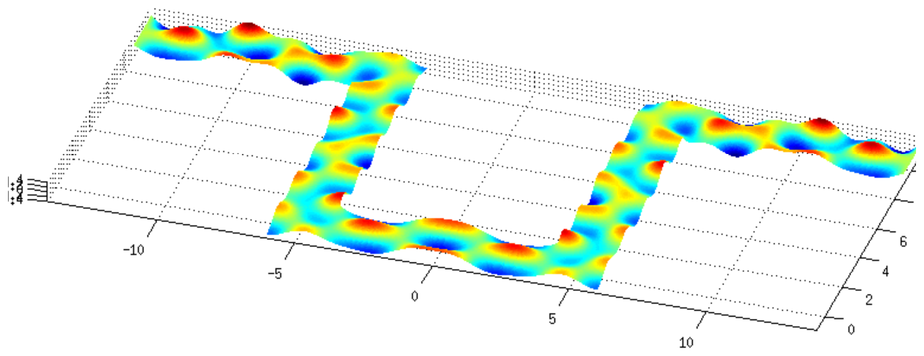


Figure 6.14: (Example 4) Approximated solution of problem (31) with Dirichlet boundary data (35) on a tunnel-shaped domain.

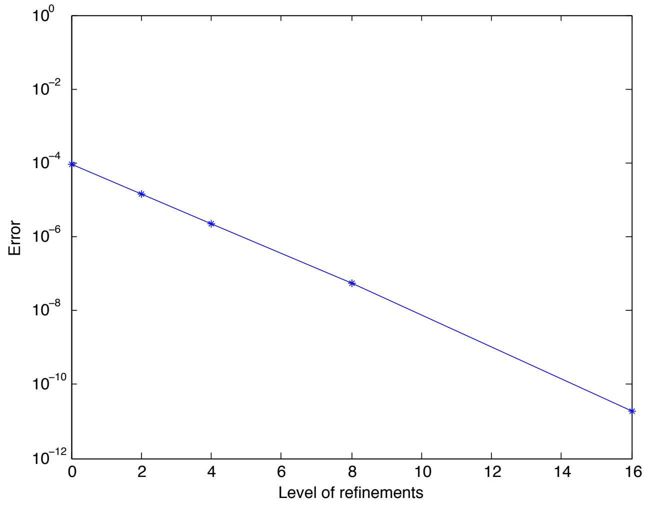


Figure 6.15: (Example 4) Convergence of the relative errors from solving problem (31) on a tunnel-shaped domain with respect to the number of levels of refinement.

6.7.3 Variable coefficient problem on square domain

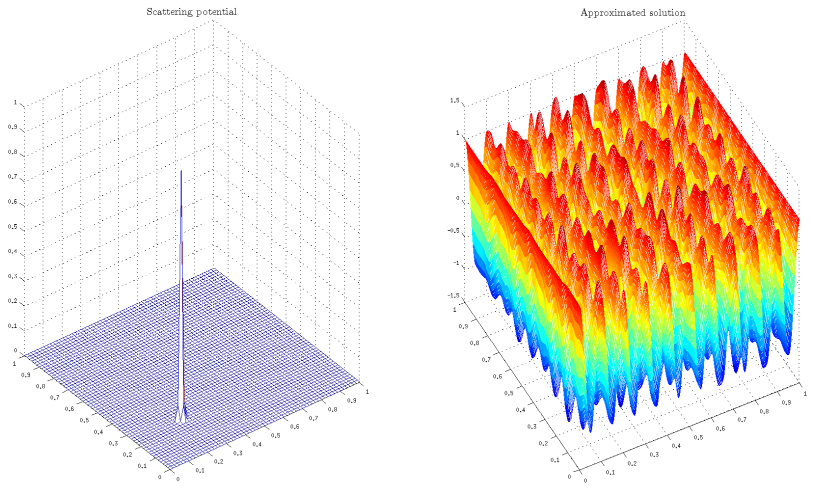
In this section, we solve the variable coefficient problems on the square domain $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2(1-b(\boldsymbol{x}))\,u(\boldsymbol{x}) = 0, & \boldsymbol{x}\in\Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x}\in\Gamma. \end{cases} \tag{36}$$

where $b(\boldsymbol{x})$ are functions exhibiting sharp gradient at points where we need to implement our refinement scheme.

6.7.3.1 Example 5

In this example, let $b(\boldsymbol{x}) = e^{-10000|\boldsymbol{x}-\tilde{\boldsymbol{x}}|^2}$, where $\tilde{x} = (0.25,\, 0.25)$ and the boundary data be $f(\boldsymbol{x}) = \cos(\kappa x_1)$. The Helmholtz parameter was kept fixed at $\kappa = 62.8$, corresponding to the domain size of 10×10 wave lengths. Again we compute the pointwise errors at $\hat{\boldsymbol{x}} = (0.75,\, 0.25)$ and compute the total fluxes towards the south boundary.



(a) (b)

Figure 6.16: (Example 5)(a) Scattering potential $b(\boldsymbol{x}) = e^{-10000|\boldsymbol{x}-\hat{\boldsymbol{x}}|^2}$, (b) approximated solution.

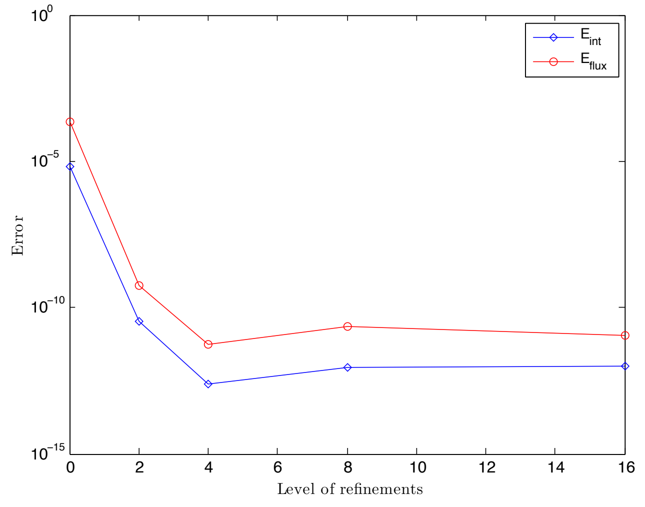


Figure 6.17: (Example 5) Convergence of the errors from solving problem (36) on a square domain with respect to the number of levels of refinement.

6.7.4 Free space scattering problem

In this section, we consider the free space problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u(\boldsymbol{x}) = f(\boldsymbol{x}), \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u(\boldsymbol{x}) - ik\,u(\boldsymbol{x})\big) = 0. \end{cases} \tag{37}$$

We assume b is compactly supported inside the domain Ω, and that f is supported outside Ω. Details on the how to construct the solution using spectral composite method and integral equation method are fully described in [33].

Example 6 Consider the free space scattering problem (37) with

$$b(\boldsymbol{x}) = \begin{cases} \dfrac{1}{a^4}(a^2 - x_1^2)(a^2 - x_2^2), & |x_1| \le a \text{ and } |x_2| \le a, \\ 0, & \text{elsewhere}, \end{cases} \tag{38}$$

where $a = 0.24$. The scatter field as well as the approximated solution are depicted in Figure 6.19. The grid with three levels of refinement along the edges of $b(\boldsymbol{x})$. The accuracy results for different numbers of wavelength are presented in Table 6.1 - 6.4. We compute the pointwise errors at points both inside the square domain $\Omega = [-0.5,\,0.5]^2$ where spectral composite method is executed and outside the domain. Specifically, the interior error E_{int} is computed at $\hat{\boldsymbol{x}}_1 = (0.25,\,0)$ and exterior error E_{ext} is computed at $\hat{\boldsymbol{x}}_2 = (1,\,0.5)$. N_{wave} denotes the number of wavelength across each side of Ω while N is the total number of Chebyshev points to discretize Ω. In Figure 6.20 we plot the set up time and solve time against the number of Chebyshev discretization points of Ω, where the set up time presents linear scale.

6.8 Concluding remarks

We have described a high-order local refinement scheme for variable coefficient elliptic PDEs in the plane, where the regularity of the solution changes across the domain. The algorithm relies on the composite spectral scheme [79] on uniform grid. When the positions needed for refinement are known in advance, the boxes containing these points are split into smaller ones according to the refinement criteria presented (Section 6.6). A direct solver is then executed after the hierarchical

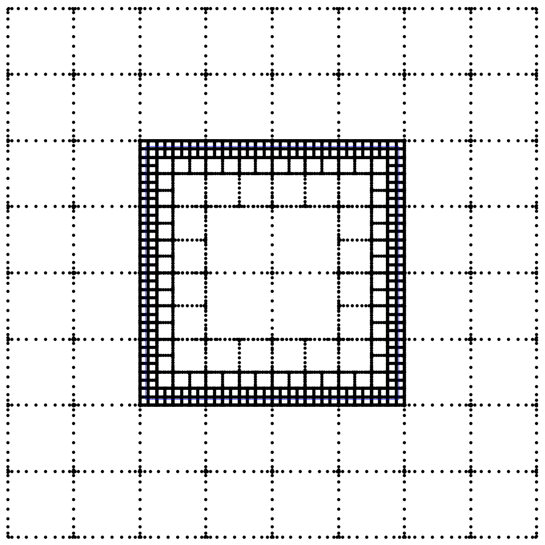
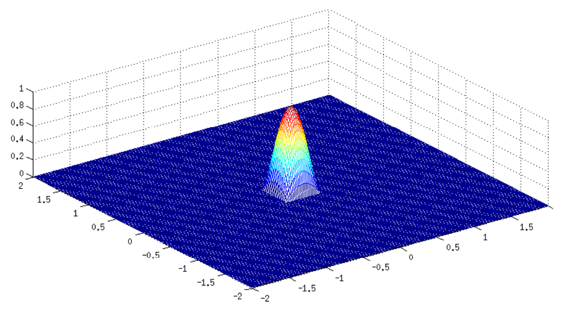
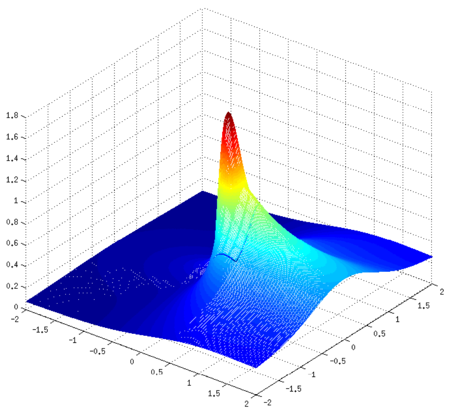


Figure 6.18: (Example 6) Grid of the exterior Gaussian nodes with three levels of refinement. The edges of $b(\boldsymbol{x})$ where refinements are executed are denoted by blue lines.



(a)



(b)

Figure 6.19: (Example 6)(a) Scatter potential $b(\boldsymbol{x})$ defined in (38), (b) approximated solution.

N_{wave}	$N_{\text{ref}} = 0$		$N_{\text{ref}} = 2$		$N_{\text{ref}} = 4$		$N_{\text{ref}} = 6$		$N_{\text{ref}} = 8$	
	N	E_{int}	N	E_{int}	N	E_{int}	N	E_{int}	N	E_{int}
3.2	1681	$3.00e-04$	9425	$8.48e-05$	52017	$1.27e-05$	230129	$1.05e-05$	942577	$1.04e-05$
	6561	$8.81e-05$	25921	$3.86e-06$	114977	$1.86e-06$	471201	$2.48e-08$	1 896 097	$3.62e-08$
	25921	$8.29e-05$	68513	$9.76e-06$	246625	$5.09e-07$	959073	$1.92e-08$	3 812 737	$6.23e-10$
6.4	1681	$2.13e-03$	9 425	$1.16e-03$	52 017	$1.64e-03$	230 129	$1.62e-03$	942577	$1.62e-03$
	6561	$4.52e-04$	25 921	$2.00e-05$	114 977	$2.00e-05$	471 201	$1.39e-05$	1 896 097	$1.37e-05$
	25921	$2.46e-04$	68513	$3.46e-05$	246625	$1.77e-06$	959073	$6.83e-08$	3 812 737	$6.45e-09$
12.7	1681	$5.80e-01$	9425	$7.75e-01$	52017	$4.41e-01$	230129	$4.43e-01$	942577	$4.43e-01$
	6561	$1.80e-03$	25921	$5.31e-03$	114977	$4.64e-03$	471201	$4.63e-03$	1 896 097	$4.63e-03$
	25921	$5.56e-04$	68513	$6.48e-05$	246625	$9.27e-06$	959073	$9.43e-06$	3 812 737	$9.45e-06$

Table 6.1: (Example 6) Interior accuracies resulting from solving free space scattering problem (37) and (38) for different wavenumbers. 11×11 tensor product grids of Chebyshev nodes are used for leaf computations.

N_{wave}	$N_{\text{ref}} = 0$		$N_{\text{ref}} = 2$		$N_{\text{ref}} = 4$		$N_{\text{ref}} = 6$		$N_{\text{ref}} = 8$	
	N	E_{ext}	N	E_{ext}	N	E_{ext}	N	E_{ext}	N	E_{ext}
3.2	1681	$3.59e-04$	9425	$9.30e-05$	52017	$1.14e-05$	230129	$2.18e-06$	942577	$1.53e-06$
	6561	$1.01e-04$	25921	$2.06e-06$	114977	$2.49e-06$	471201	$1.09e-07$	1 896 097	$3.11e-08$
	25921	$9.04e-05$	68513	$1.29e-05$	246625	$6.69e-07$	959073	$2.45e-08$	3 812 737	$2.10e-10$
6.4	1681	$6.27e-03$	9 425	$7.01e-03$	52 017	$7.57e-03$	230 129	$7.53e-03$	942577	$7.53e-03$
	6561	$3.01e-04$	25 921	$1.53e-05$	114 977	$1.22e-05$	471 201	$5.18e-06$	1 896 097	$4.95e-06$
	25921	$2.93e-04$	68513	$3.86e-05$	246625	$2.09e-06$	959073	$1.54e-07$	3 812 737	$8.04e-08$
12.7	1681	$1.18e+00$	9425	$1.18e+00$	52017	$1.14e+00$	230129	$1.14e+00$	942577	$1.14e+00$
	6561	$1.65e-02$	25921	$1.50e-02$	114977	$1.52e-02$	471201	$1.52e-02$	1 896 097	$1.52e-02$
	25921	$8.81e-04$	68513	$9.83e-05$	246625	$1.33e-05$	959073	$8.42e-06$	3 812 737	$8.24e-06$

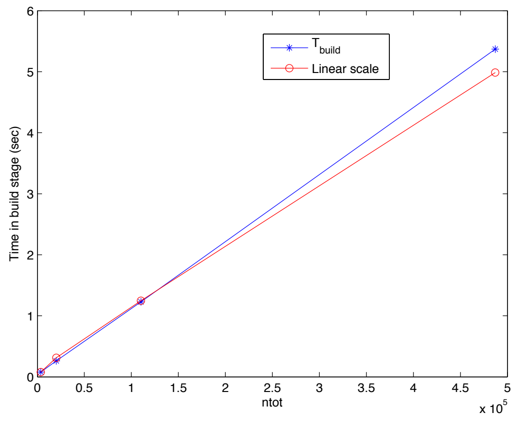
Table 6.2: (Example 6) Exterior accuracies resulting from solving free space scattering problem (37) and (38) for different wavenumbers. 11×11 tensor product grids of Chebyshev nodes are used for leaf computations.

N_{wave}	$N_{\text{ref}} = 0$		$N_{\text{ref}} = 2$		$N_{\text{ref}} = 4$		$N_{\text{ref}} = 6$	
	N	E_{int}	N	E_{int}	N	E_{int}	N	E_{int}
3.2	6561	$6.34e-05$	34785	$6.44e-06$	190017	$5.18e-07$	839169	$2.79e-07$
	25921	$4.28e-05$	96481	$8.05e-06$	421057	$6.66e-07$	1719361	$1.14e-08$
	103041	$6.23e-06$	258273	$7.09e-07$	907425	$5.61e-09$	3504033	$7.49e-09$
6.4	6561	$1.81e-04$	34785	$5.70e-05$	190017	$3.90e-05$	839169	$4.08e-05$
	25921	$1.24e-04$	96481	$2.63e-05$	421057	$2.56e-06$	1719361	$6.29e-07$
	103041	$1.50e-05$	258273	$2.16e-06$	907425	$2.44e-08$	3504033	$2.56e-08$
12.7	6561	$5.15e-01$	34785	$5.02e-01$	190017	$5.07e-01$	839169	$5.04e-01$
	25921	$5.35e-04$	96481	$7.07e-04$	421057	$7.32e-04$	1719361	$7.30e-04$
	103041	$6.39e-05$	258273	$4.26e-06$	907425	$8.77e-07$	3504033	$8.29e-07$

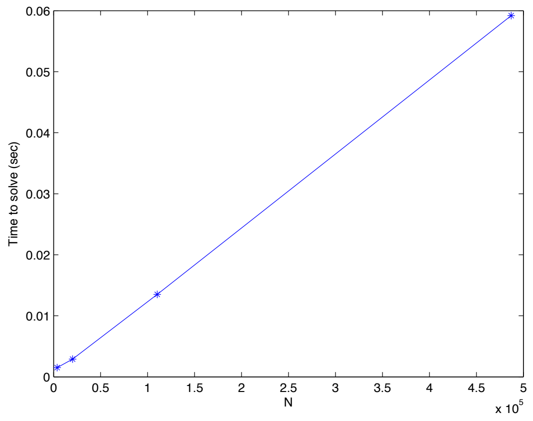
Table 6.3: (Example 6) Interior accuracies resulting from solving free space scattering problem (37) and (38) for different wavenumbers. 21×21 tensor product grids of Chebyshev nodes are used for leaf computations.

N_{wave}	$N_{\text{ref}} = 0$		$N_{\text{ref}} = 2$		$N_{\text{ref}} = 4$		$N_{\text{ref}} = 6$	
	N	E_{ext}	N	E_{ext}	N	E_{ext}	N	E_{ext}
3.2	6561	$6.55e-05$	34785	$5.93e-06$	190017	$4.89e-07$	839169	$6.41e-07$
	25921	$5.09e-05$	96481	$1.03e-05$	421057	$8.59e-07$	1719361	$1.47e-08$
	103041	$5.39e-06$	258273	$8.68e-07$	907425	$8.29e-09$	3504033	$9.79e-09$
6.4	6561	$3.61e-04$	34785	$5.52e-04$	190017	$5.35e-04$	839169	$5.37e-04$
	25921	$1.59e-04$	96481	$3.00e-05$	421057	$4.15e-06$	1719361	$1.66e-06$
	103041	$1.97e-05$	258273	$2.70e-06$	907425	$2.30e-08$	3504033	$3.01e-08$
12.7	6561	$4.10e-02$	34785	$3.92e-02$	190017	$3.99e-02$	839169	$3.98e-02$
	25921	$8.77e-04$	96481	$7.88e-04$	421057	$8.03e-04$	1719361	$8.02e-04$
	103041	$6.36e-05$	258273	$8.49e-06$	907425	$3.16e-06$	3504033	$3.15e-06$

Table 6.4: (Example 6) Exterior accuracies resulting from solving free space scattering problem (37) and (38) for different wavenumbers. 21×21 tensor product grids of Chebyshev nodes are used for leaf computations.



(a) T_{build}



(b) T_{solve}

Figure 6.20: (Example 6) Computational time T_{build} and solve time T_{solve} are plotted against the total degrees of freedom for refinement levels varying from $0, 2, 4, 6$. The domain is of size 3.2×3.2 wave-lengths and $p = 15$.

tree is built which solves the problem in a single sweep. After all the solution operators are constructed for every box in the hierarchical tree, solutions can be obtained almost instantaneously for multiple righthand-sides without recomputing the solution operators.

Numerical experiments indicate the method is able to retain the high-order accuracy. For Helmholtz problem involving non-smooth Dirichlet boundary data on a domain of size 10×10 wavelengths, the method takes 8 to 16 levels of refinements to achieve 9 to 12 digits of accuracy. For variable coefficient problem where the $b(\boldsymbol{x})$ exhibits sharp gradient, the scheme only takes 4 levels refinement to get 12 digits of accuracy.

The refinement scheme presented only works for situations when the positions where the regularity of the solutions changes known in advance. We expect to develop an adaptive refinement scheme that can be applied to a broader range of problems.

Chapter 7

Spectral method for solving three-dimensional scattering problem

7.1 Introduction

This chapter describe an $O(N^{4/3})$ algorithm to solve boundary value problem

$$\begin{cases} A\,u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{1}$$

where $\Omega = [0,\,1]^3$ is a three-dimensional rectangular domain with boundary Γ, and A is an elliptic partial differential operator

$$\begin{aligned}[Au](\boldsymbol{x}) =& -c_{11}(\boldsymbol{x})[\partial_1^2\,u](\boldsymbol{x}) - c_{22}(\boldsymbol{x})[\partial_2^2\,u](\boldsymbol{x}) - c_{33}(\boldsymbol{x})[\partial_3^2\,u](\boldsymbol{x}) - 2c_{12}(\boldsymbol{x})[\partial_1\partial_2\,u](\boldsymbol{x}) - 2c_{13}(\boldsymbol{x})[\partial_1\partial_3\,u](\boldsymbol{x}) \\ & - 2c_{23}(\boldsymbol{x})[\partial_2\partial_3\,u](\boldsymbol{x}) + c_1(\boldsymbol{x})[\partial_1\,u](\boldsymbol{x}) + c_2(\boldsymbol{x})[\partial_2\,u](\boldsymbol{x}) + c_3(\boldsymbol{x})[\partial_3\,u](\boldsymbol{x}) + c(\boldsymbol{x})u(\boldsymbol{x}).\end{aligned} \tag{2}$$

Note that to enforce the above equation to be an elliptic equation, we need to guarantee the coercivity:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ 0 & c_{22} & c_{23} \\ 0 & 0 & c_{33} \end{bmatrix} > 0.$$

The scheme presented in this chapter is an extension of the early work [79, 35]. Similar to the scheme described in Chapter 6, the method is based on a composite spectral discretization. In a single sweep, we construct approximations to the solution operators. One of the advantages of the proposed method compared to iterative solvers (e.g. , GMRES or multigrid) is that once all the solutions operators have been built and stored, solutions can be obtained extremely rapidly, usually within a few seconds to minutes, for different right-hand sides. Another advantage is that the proposed method is able to solve problems for which iterative solvers converge slowly or not at all.

The direct solver is composite of three steps: 1) discretization, 2) build stage and 3) solver stage. Specifically, the three-dimensional domain Ω is split into small rectangular domains (called patches). On each patch, the solution u to (1) is approximated via tabulation on a tensor product grid of Chebyshev points to high-order accuracy. The elliptic operator in (1) is then approximated via spectral differential matrix defined on each local grid. At build stage, we construct the local solution operator and Dirichlet-to-Neumann (DtN) operator for each leaf patch. For a parent patch, the solution operator and DtN operator are constructed by glueing together the DtN operators of its children. At solve stage, we take a vector of Dirichlet boundary data as input and construct the tabulated values of u at all internal grid points via local solution operator, moving downwards through the hierarchical tree.

The complexity of the direct solver stated above is dominated by building DtN operators at top levels, which is of $O(N^2)$ complexity. The novelty of the proposed method is to explore the internal structures of the dense matrices representing DtN operators at top levels. The acceleration technique, which is similar to the work [74, 89, 100], improves the complexity to $O(N^{4/3})$.

Remark 11 Although in this chapter we only apply the method to solving PDEs on simple rectangular domains, it can be easily generalized to other domains $\Omega \subseteq \mathbb{R}^3$, including curved domains. Details on how to extend the methodology to two-dimensional domains that can be mapped to a rectangle or a union of rectangles with smooth parameterizations can be found in sections 6.3 and 6.4 of [79].

7.1.1 Outline

Section 7.2 describes the Gaussian grid on the surface of the domain which forms the computational grid. Section 7.3 describes how to compute the solution operator and the DtN operator for a leaf box. Section 7.4 describes how to compute the DtN operator for parent boxes as well as the full hierarchical algorithm. Sections 7.5 and 7.6 describe how to improve the asymptotic computational complexity at build stage from $O(N^2)$ to $O(N^{4/3})$ by exploring the internal structures of the DtN operators. In Section 7.7, we illustrate the performance of the algorithm with a set of numerical examples. Section 7.8 summarizes the key features of the algorithm, as well as limitation and future work.

7.2 Discretization

We start with partitioning the domain Ω into a collection of three-dimensional rectangular domains, called leaf boxes. On each face of the leaf box, place $q \times q$ tensor product grid of Gaussian interpolation nodes to interpolate the potential function u as well as its first, second and third derivatives to high-order of accuracy. Let $\{\mathbf{z}_\ell\}_{\ell=1}^{N}$ denote the collection of Gaussian nodes on the boundaries. Let u be the vector holding approximation to the solution function u tabulated on $\{\mathbf{z}_\ell\}_{\ell=1}^{N}$, in other words,

$$\mathsf{u}_\ell \approx u(\mathbf{z}_\ell).$$

Furthermore, let v denote a $6q^2$-length vector holding approximated values of the boundary flux of u, say

$$\mathsf{v}_\ell \approx \begin{cases} \partial_1 u(\mathbf{z}_\ell), & \text{when } \mathbf{z}_\ell \text{ lies on the right and left faces,} \\ \partial_2 u(\mathbf{z}_\ell), & \text{when } \mathbf{z}_\ell \text{ lies on the front and back faces,} \\ \partial_3 u(\mathbf{z}_\ell), & \text{when } \mathbf{z}_\ell \text{ lies on the bottom and up faces.} \end{cases}$$

7.3 Leaf computation

This section describes a spectral method for computing approximations to Dirichlet-to-Neumann operators associated with a leaf box Ω_τ. In other words, we seek a matrix T^τ such that

$$\mathsf{v} = \mathsf{T}^\tau \mathsf{u}(J_\mathrm{e}^\tau), \tag{3}$$

where J_e^τ denote the set of Gaussian nodes lying on the boundaries of Ω_τ. To this end, we first discretize the leaf boxes using tensor product grid of Chebyshev nodes. This approximates the smooth function u and its derivatives to spectral accuracy. Then we construct a solution operator that maps the potential values tabulated on exterior Chebyshev nodes to values on interior nodes, from where we construct the DtN operator that maps the potential values to boundary fluxes on Chebyshev nodes. Applying purpose specified interpolation matrices enable us to re-tabulate boundary fluxes to the exterior Gaussian nodes.

7.3.1 Spectral discretization.

On each leaf box, we construct approximations to the smooth function u satisfying (1) and its derivatives using a classic spectral collocation method as described in [94]. Specifically, for a small number p, we place a $p \times p \times p$ tensor product grid of Chebyshev nodes denoted by $\{\boldsymbol{x}_k\}_{k=1}^{p^3}$. Let $\tilde{\mathsf{u}} \in \mathbb{R}^{p^3}$ denote a vector holding approximations to u at $\{\boldsymbol{x}_k\}_{k=1}^{p^3}$ and let $\mathsf{D}^{(1)}$, $\mathsf{D}^{(2)}$, $\mathsf{D}^{(3)}$ denote the spectral differentiation matrices corresponding to the partial differential operators $\partial/\partial x_1$, $\partial/\partial x_2$ and $\partial/\partial x_3$. Then the operator in (1) can be approximated via a $p^3 \times p^3$ matrix

$$\begin{aligned}\mathsf{A} = &- \mathsf{C}_{11}(\mathsf{D}^{(1)})^2 - \mathsf{C}_{22}(\mathsf{D}^{(2)})^2 - \mathsf{C}_{33}(\mathsf{D}^{(3)})^2 - 2\mathsf{C}_{12}\mathsf{D}^{(1)}\mathsf{D}^{(2)} - 2\mathsf{C}_{13}\mathsf{D}^{(1)}\mathsf{D}^{(3)} - 2\mathsf{C}_{23}\mathsf{D}^{(2)}\mathsf{D}^{(3)} \\ &+ \mathsf{C}_1\mathsf{D}^{(1)} + \mathsf{C}_2\mathsf{D}^{(2)} + \mathsf{C}_3\mathsf{D}^{(3)} + \mathsf{C},\end{aligned} \tag{4}$$

where C_{ij} are coefficient matrices with diagonal entries $\{c_{i,j}(\boldsymbol{x}_k)\}$ as well as C_i and C.

Partition the index set of Chebyshev nodes as

$$\{1, 2, \ldots, p^3\} = I_\mathrm{e} \cup I_\mathrm{i}$$

where I_e contains the $6(p-1)^2$ exterior nodes on the boundaries and I_i contains the $(p-2)^3$ interior nodes of Ω_τ. Then partition the vector $\tilde{\mathsf{u}}$ according to its value at internal and exterior nodes via

$$\tilde{\mathsf{u}}_\mathrm{i} = \tilde{\mathsf{u}}(I_\mathrm{i}) \quad \text{and} \quad \tilde{\mathsf{u}}_\mathrm{e} = \tilde{\mathsf{u}}(I_\mathrm{e}).$$

We also partition the A into four parts via

$$\mathsf{A}_{\mathrm{i,i}} = \mathsf{A}(I_{\mathrm{i}}, I_{\mathrm{i}}), \quad \mathsf{A}_{\mathrm{i,e}} = \mathsf{A}(I_{\mathrm{i}}, I_{\mathrm{e}}), \quad \mathsf{A}_{\mathrm{e,i}} = \mathsf{A}(I_{\mathrm{e}}, I_{\mathrm{i}}), \quad \mathsf{A}_{\mathrm{e,e}} = \mathsf{A}(I_{\mathrm{e}}, I_{\mathrm{e}}).$$

We also introduce notations relating to Gaussian nodes that will be used in next section. Specifically, on each face of the rectangular box, we place $q \times q$ Gaussian nodes denoted by $\{\mathbf{z}_\ell\}_{\ell=1}^{6q^2}$. Let $\mathsf{u} \in \mathbb{R}^{6q^2}$ denote a vector holding approximations to u at these Gaussian nodes.

7.3.2 Constructing the DtN operator.

On each leaf box, as what we do in Chapter 6, we construct an approximation to the Dirichlet-to-Neumann operator which maps the Dirichlet boundary data defined on the exterior nodes of Ω_τ to its normal derivatives across the boundary. Specifically the DtN operator T is defined as

$$T : u|_{\partial\Omega_\tau} \mapsto u_n|_{\partial\Omega_\tau}.$$

To this end, we first compute the solution operator. Specifically, in (1) we enforce

$$\mathsf{A}\tilde{\mathsf{u}} = 0$$

at all interior nodes of Ω_τ. Then the matrix representation of the equation in (1) can be partitioned as

$$\begin{bmatrix} \mathsf{A}_{\mathrm{i,i}} & \mathsf{A}_{\mathrm{i,e}} \end{bmatrix} \begin{bmatrix} \tilde{\mathsf{u}}_{\mathrm{i}} \\ \tilde{\mathsf{u}}_{\mathrm{e}} \end{bmatrix} = 0.$$

Therefore, the potential values at interior nodes of Ω_τ can be constructed from its values at exterior nodes by applying the solution operator

$$\tilde{\mathsf{u}}_{\mathrm{i}} = \mathsf{S}\, \tilde{\mathsf{u}}_{\mathrm{e}}, \tag{5}$$

where $\mathsf{S} = -\mathsf{A}_{\mathrm{i,i}}^{-1}\mathsf{A}_{\mathrm{i,e}}$.

Construction of DtN operator consists for three steps:

Step 1 - re-tabulation from Gaussian nodes to Chebyshev nodes: On each surface of Ω, form a unique polynomial of degree no more than $q-1$ that interpolates q potential values at

Gaussian nodes. Evaluate the kronecker tensor product of these polynomials at p^2 exterior Cheybshev nodes. Let L_{G2C} denote the matrix that achieves the interpolation.

Step 2 - spectral differentiation on Chebyshev grids: We can determine the gradient on each surface via spectral differentiation on the entire domain since the potential is known on all Chebyshev nodes in Ω. Specifically, let $\tilde{\mathsf{v}}_{\text{bot}}$ denote the boundary flux on the bottom surface, then the gradient on that surface can be formed via

$$\tilde{\mathsf{v}}_{\text{bot}} = \mathsf{D}^{(3)}_{\text{bot,e}}\tilde{\mathsf{u}}_{\text{e}} + \mathsf{D}^{(3)}_{\text{bot,i}}\tilde{\mathsf{u}}_{\text{i}} = (\mathsf{D}^{(3)}_{\text{bot,e}} + \mathsf{D}^{(3)}_{\text{bot,i}}\mathsf{S})\,\tilde{\mathsf{u}}_{\text{e}},$$

where $\mathsf{D}^{(3)}_{\text{bot,e}}$ is a sub-matrix of $\mathsf{D}^{(3)}$ with entries corresponding to nodes at bottom face and all exterior nodes. Boundary fluxes on the other five faces can be computed in the same way. Denote V the DtN operator that maps boundary potential values to boundary fluxes at Chebyshev nodes.

Step 3 - re-tabulation from Cheyshev nodes back to Gaussian nodes: Now the boundary fluxes are known at Chebyshev exterior nodes. Similar to Step 1, we form a unique polynomial of degree no more than $p-1$ that interpolates the potential values at Chebyshev boundary nodes. Evaluating the kronecker tensor product of the polynomials at q^2 Gaussian nodes at each surface gives the vector v. Denote the corresponding interpolation matrix by L_{C2G}.

Putting everything together, we form the DtN operator T as a product of three matrices

$$\mathsf{T} = \mathsf{L}_{\text{C2G}} \circ \mathsf{V} \circ \mathsf{L}_{\text{G2C}}. \tag{6}$$

7.4 Direct solver via composite spectral method

In this section, we present how to construct the DtN operator for a rectangular domain Ω_τ which is the union of two smaller domains, i.e.

$$\Omega_\tau = \Omega_{\sigma_1} \cup \Omega_{\sigma_2}$$

when the approximations to the DtN operators for Ω_{σ_1} and Ω_{σ_2} are available.

7.4.1 Merge two DtN operators.

For a box Ω_τ with two children boxes Ω_{σ_1} and Ω_{σ_2}, see Figure 7.1, the DtN operator T^τ can be formed by "merging" T^{σ_1} and T^{σ_2}. We start with partitioning the nodes on $\partial\Omega_{\sigma_1}$ and $\partial\Omega_{\sigma_2}$ into three sets:

J_1	Boundary nodes of Ω_{σ_1} but not boundary nodes of Ω_{σ_2}.
J_2	Boundary nodes of Ω_{σ_2} but not boundary nodes of Ω_{σ_1}.
J_3	Boundary nodes shared by Ω_{σ_1} and Ω_{σ_2} but are not boundary nodes of Ω_τ.

Let u denote the vector holding tabulated solution values and v denotes the vector holding boundary fluxes values as in Section 7.2. Letting u_i and u_e denote the solution tabulated on the interior and exterior nodes of Ω_τ, we have

$$\mathsf{u}_\mathrm{i} = \mathsf{u}_3, \quad \text{and} \quad \mathsf{u}_\mathrm{e} = \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \end{bmatrix}.$$

By definition of DtN operators, we have

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,1} & \mathsf{T}^{\sigma_1}_{1,3} \\ \mathsf{T}^{\sigma_1}_{3,1} & \mathsf{T}^{\sigma_1}_{3,3} \end{bmatrix} \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_3 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathsf{v}_2 \\ \mathsf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathsf{T}^{\sigma_2}_{2,2} & \mathsf{T}^{\sigma_2}_{2,3} \\ \mathsf{T}^{\sigma_2}_{3,2} & \mathsf{T}^{\sigma_2}_{3,3} \end{bmatrix} \begin{bmatrix} \mathsf{u}_2 \\ \mathsf{u}_3 \end{bmatrix}. \tag{7}$$

Noting that u_3 and v_3 are shared in both of the equations, we have

$$\mathsf{T}^{\sigma_1}_{3,1}\mathsf{u}_1 + \mathsf{T}^{\sigma_1}_{3,3}\mathsf{u}_3 = \mathsf{T}^{\sigma_2}_{3,2}\mathsf{u}_2 + \mathsf{T}^{\sigma_2}_{3,3}\mathsf{u}_3.$$

Therefore,

$$\mathsf{u}_3 = (\mathsf{T}^{\sigma_1}_{3,3} - \mathsf{T}^{\sigma_2}_{3,3})^{-1}(-\mathsf{T}^{\sigma_1}_{3,1}\mathsf{u}_1 + \mathsf{T}^{\sigma_2}_{3,2}\mathsf{u}_2) = \mathsf{S}^\tau \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \end{bmatrix}, \tag{8}$$

where

$$\mathsf{S}^\tau = (\mathsf{T}^{\sigma_1}_{3,3} - \mathsf{T}^{\sigma_2}_{3,3})^{-1}[-\mathsf{T}^{\sigma_1}_{3,1} \mid \mathsf{T}^{\sigma_2}_{3,2}]. \tag{9}$$

Likewise, we construct the DtN operator of Ω_τ by solving the equilibrium equation on v_3 and obtain

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_2 \end{bmatrix} = \mathsf{T}^\tau \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \end{bmatrix},$$

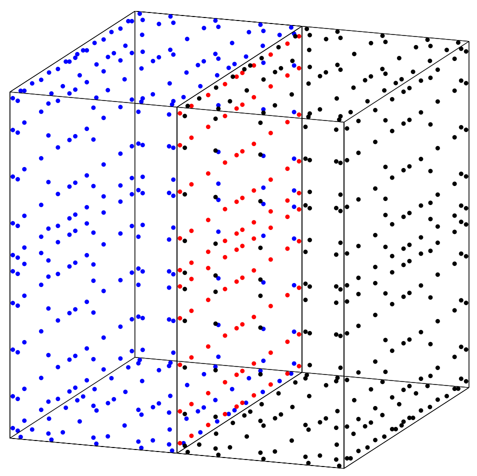


Figure 7.1: The three-dimensional rectangular domain $\Omega_\tau = \Omega_{\sigma_1} \cup \Omega_{\sigma_2}$. J_1 are denoted by blue dots while J_2 are denoted by black dots. Red nodes present the interior nodes of Ω_τ denoted by J_3.

where

$$\mathsf{T}^{\tau} = \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,1} & 0 \\ 0 & \mathsf{T}^{\sigma_2}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,3} \\ \mathsf{T}^{\sigma_2}_{2,3} \end{bmatrix} \mathsf{S}^{\tau}. \tag{10}$$

7.4.2 The full hierarchical scheme.

Now we know how to construct the DtN operator for a leaf (Section 7.3), and how to merge the DtN operators of two neighboring patches to form the DtN operator of their union (Section 7.4.1), we are ready to describe the full hierarchical scheme for solving (1). To start with, partition the domain Ω into into a hierarchical tree, called leaf boxes. Form a hierarchical binary tree on the collection of leaf boxes, making sure that the boxes on the same level are roughly of the same size. Order the boxes in the way that if box τ is a parent of σ then $\tau < \sigma$. Let L denote the number of levels, starting from 0 where Ω has index $\tau = 1$. We assume the root of the tree (i.e. the full box Ω) has index $\tau = 1$. Figure 7.2 shows the boxes on levels 0 to 3.

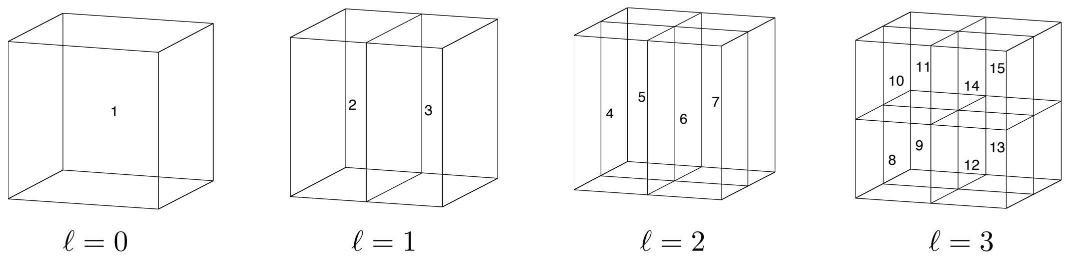


Figure 7.2: The rectangular domain Ω is split into $2 \times 2 \times 2$ leaf boxes. These are then organized into a binary tree of successively larger boxes as described in Section 7.2. Note that if box τ is the parent of a box σ, we have $\tau < \sigma$.

The algorithm is composed of three steps:

(1) Discretization: The domain is partitioned into a hierarchical tree where in each leaf box a $p \times p \times p$ tensor product grid of Chebyshev nodes are placed to interpolate smooth potential function u to spectral accuracy. The elliptic operator in (1) is then approximated via spectral differentiation matrices.

(2) Upwards pass: For each leaf box, a solution operator and DtN operator are constructed as

shown in Section 7.3. Then we do a single sweep moving upwards, constructing the DtN operator of a parent box by "merging" the DtN operators of its two children.

(3) Downwards pass: After collecting all the solution and DtN operators for all boxes on each level, we take a vector holding the Dirichlet boundary data as input and apply the solution operator to map the solution values on the exterior nodes to its values on the interior nodes. Moving downwards along the hierarchical tree, we finally obtain the approximated values of the solution on the Chebyshev grid points.

7.5 Fast algorithms of compressible matrices

The cost of the algorithm presented in Section 7.4.2 is dominated by constructing DtN operators at top levels. The matrix operations involve inverting dense matrices of size $O(N^{2/3}) \times O(N^{2/3})$ where N is total number of discretization nodes, resulting in $O(N^2)$ total cost. However there are internal structures in these dense matrices that can be explored to accelerate the computation. Specifically, the off-diagonal blocks of these matrices are rank-deficient to high precision and the diagonal blocks can be represented as *Hierarchical Semi-Separable (HSS)* matrices. By HSS, we mean a matrix is amenable to a telescoping block factorization. In this section, we give definitions to HSS matrices and briefly describe their properties.

7.5.1 Compressible matrices.

We first give definition of block separable matrix. Let A be a matrix of size $np \times np$ such that it is partitioned into $p \times p$ blocks, each of size $m \times m$, *i.e.*

$$\mathsf{A} = \begin{bmatrix} \mathsf{D}_1 & \mathsf{A}_{1,2} & \dots & \mathsf{A}_{1,p} \\ \mathsf{A}_{2,1} & \mathsf{D}_2 & \dots & \mathsf{A}_{2,p} \\ \vdots & \vdots & & \vdots \\ \mathsf{A}_{p,1} & \mathsf{A}_{p,2} & \dots & \mathsf{D}_p \end{bmatrix} \tag{11}$$

A is called block separable if each off-diagonal block admits the factorization

$$\underset{m\times m}{\mathsf{A}_{\sigma,\tau}} = \underset{m\times k}{\mathsf{U}_{\sigma}} \underset{k\times k}{\tilde{\mathsf{A}}_{\sigma,\tau}} \underset{k\times m}{\mathsf{V}^{*}_{\tau}}, \quad \sigma,\tau \in \{1,2,\dots,p\}, \quad \sigma \neq \tau, \tag{12}$$

where U and V are $m \times k$ matrices. Therefore, the matrix A admits the factorization

$$\underset{mp\times mp}{\mathsf{A}} = \underset{mp\times kp}{\mathsf{U}} \underset{kp\times kp}{\tilde{\mathsf{A}}} \underset{kp\times mp}{\mathsf{V}^{*}} + \underset{mp\times mp}{\mathsf{D}}, \tag{13}$$

where $\mathsf{U} = \text{diag}(\mathsf{U}_1, \mathsf{U}_2, \dots, \mathsf{U}_\mathsf{p})$, $\mathsf{V} = \text{diag}(\mathsf{V}_1, \mathsf{V}_2, \dots, \mathsf{V}_\mathsf{p})$ and $\mathsf{D} = \text{diag}(\mathsf{D}_1, \mathsf{D}_2, \dots, \mathsf{D}_\mathsf{p})$. Moreover,

$$\tilde{\mathsf{A}} = \begin{bmatrix} 0 & \tilde{\mathsf{A}}_{1,2} & \tilde{\mathsf{A}}_{1,3} & \dots \\ \tilde{\mathsf{A}}_{2,1} & 0 & \tilde{\mathsf{A}}_{2,3} & \dots \\ \tilde{\mathsf{A}}_{3,1} & \tilde{\mathsf{A}}_{3,2} & 0 & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}.$$

Remark 12 In constructing the factorization (12), we use the so-called *interpolative decomposition*

$$\mathsf{B} = \mathsf{B}(:, J)\mathsf{X},$$

where J is an index vector pointing to k columns of B, and the $k \times m$ matrix X is well-conditioned and has a $k \times k$ identity matrix as a submatrix. In this sense, U and V are matrices with k columns and rows that form column basis and row basis for the range of A, respectively. The factorization costs $O(mk^2)$ asymptotically.

7.5.1.1 Hierarchically semi-separable (HSS) matrices.

To give the definition of HSS matrices, we first define the binary tree structure associated with the discretization nodes with index $I = [1, 2, \dots M]$. Basically, let I denote the root of the tree and partition the index set into two roughly equi-sized subsets level by level. Specifically, we call an index set as leaf node if there is no further split. For a non-leaf node τ we call two nodes σ_1 and σ_2 as the children of τ if $I_\tau = I_{\sigma_1} \cup I_{\sigma_2}$, and call τ the parent node of σ_1 and σ_2.

By now, we are ready to give definition of hierarchical semi-separable with respect to a given binary tree associated with index set I. Let $\ell = 0, 1, \dots, L$ denote the levels from the coarsest level to the finest level. A matrix is called a HSS matrix if it satisfies two conditions:

(1) For each leaf node pair $\{\tau, \tau'\}$ $(\tau \neq \tau')$ on level L, there exists integer k such that the off-diagonal blocks $A_{\tau,\tau'}$ admits factorization

$$\underset{m\times m}{\mathsf{A}_{\tau,\tau'}} = \underset{m\times k}{\mathsf{U}_{\tau}} \; \underset{k\times k}{\tilde{\mathsf{A}}_{\tau,\tau'}} \; \underset{k\times m}{\mathsf{V}^*_{\tau}} \,. \tag{14}$$

(2) For off-diagonal blocks on level $\ell = L-1, L-2, \ldots, 1$, the rank-deficiency property at level ℓ can be constructed based on the next finer level $\ell+1$. Specifically, for any distinct non-leaf nodes τ and τ' on level ℓ with children σ_1, σ_2 and σ_1' and σ_2', define

$$\mathsf{A}_{\tau,\tau'} = \begin{bmatrix} \tilde{\mathsf{A}}_{\sigma_1,\sigma_1'} & \tilde{\mathsf{A}}_{\sigma_1,\sigma_2'} \\ \tilde{\mathsf{A}}_{\sigma_2,\sigma_1'} & \tilde{\mathsf{A}}_{\sigma_2,\sigma_2'} \end{bmatrix}.$$

There exists factorization

$$\underset{2k\times 2k}{\mathsf{A}_{\tau,\tau'}} = \underset{2k\times k}{\mathsf{U}_{\tau}} \; \underset{k\times k}{\tilde{\mathsf{A}}_{\tau,\tau'}} \; \underset{k\times 2k}{\mathsf{V}^*_{\tau}} \,. \tag{15}$$

Define

$$\mathsf{D}_{\tau} = \mathsf{A}(I_{\tau}, I_{\tau})$$

for each leaf node τ, and define

$$\mathsf{B}_{\tau} = \begin{bmatrix} 0 & \tilde{\mathsf{A}}_{\sigma_1,\sigma_2} \\ \tilde{\mathsf{A}}_{\sigma_2,\sigma_1} & 0 \end{bmatrix}$$

for non-leaf node τ with children σ_1 and σ_2. An HSS matrix A can then be fully described if

- for every leaf node, we are given the diagonal matrices D_{τ}, as well as column basis and row basis U_{τ} and V_{τ};
- for every non-leaf node, we are given the interaction matrix B_{τ} between the children of τ, as well as column basis and row basis U_{τ} and V_{τ}.

In other words, the HSS matrix A admits telescoping factorization given U, V, D and B hierarchically:

(1) $\tilde{\mathsf{A}}^{(0)} = \mathsf{B}^{(0)}$,

(2) $\underset{k2^\ell\times k2^\ell}{\tilde{\mathsf{A}}^{(\ell)}} = \underset{k2^\ell\times k2^{\ell-1}}{\mathsf{U}^{(\ell)}} \quad \underset{k2^{\ell-1}\times k2^{\ell-1}}{\tilde{\mathsf{A}}^{(\ell-1)}} \quad \underset{k2^{\ell-1}\times k2^\ell}{(\mathsf{V}^{(\ell)})^*} + \underset{k2^\ell\times k2^\ell}{\mathsf{B}^{(\ell)}}$ for $\ell = 1, 2, \ldots, L-1$,

(3) $\underset{m2^L\times n2^L}{\mathsf{A}} = \underset{m2^L\times k2^L}{\mathsf{U}^{(L)}} \quad \underset{k2^L\times k2^L}{\tilde{\mathsf{A}}^{(L-1)}} \quad \underset{k2^L\times n2^L}{(\mathsf{V}^{(L)})^*} + \underset{m2^L\times m2^L}{\mathsf{D}^{(L)}}.$

7.5.2 Fast algorithms on HSS matrices.

Fast algorithms on how to add two HSS matrices, apply HSS matrices to vector and invert HSS matrices are presented in [37]. Here we briefly summarize the fast matrix-vector multiplication and inversion algorithms.

7.5.2.1 Fast matrix inversion algorithm

The inverse of a HSS matrix can be rapidly constructed using a variation of the classical Sherman-Morrison-Woodbury formula by exploring the low-rank deficiency hierarchically. The total computational cost is reduced by turning the task of inverting matrix of size $mp \times mp$ to inverting matrix of size $kp \times kp$ at each level. Firstly, we define several matrices on each node τ

$$\hat{\mathsf{D}}_\tau = (\mathsf{V}_\tau^* \tilde{\mathsf{D}}_\tau^{-1} \mathsf{U}_\tau)^{-1}, \tag{16}$$

$$\mathsf{E}_\tau = \tilde{\mathsf{D}}_\tau^{-1} \mathsf{U}_\tau \hat{\mathsf{D}}_\tau, \tag{17}$$

$$\mathsf{F}_\tau = (\hat{\mathsf{D}}_\tau \mathsf{V}_\tau^* \tilde{\mathsf{D}}_\tau^{-1})^*, \tag{18}$$

$$\mathsf{G}_\tau = \tilde{\mathsf{D}}_\tau^{-1} - \tilde{\mathsf{D}}_\tau^{-1} \mathsf{U}_\tau \hat{\mathsf{D}}_\tau \mathsf{V}_\tau^* \tilde{\mathsf{D}}_\tau^{-1}. \tag{19}$$

In above equations, we define

$$\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$$

if τ is a leaf node and define

$$\tilde{\mathsf{D}}_\tau = \begin{bmatrix} \hat{\mathsf{D}}_{\sigma_1} & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & \hat{\mathsf{D}}_{\sigma_2} \end{bmatrix}$$

if τ is a non-leaf node. Note that all the matrices $\{\hat{\mathsf{D}}_\tau, \mathsf{E}_\tau, \mathsf{F}_\tau, \mathsf{G}_\tau\}_\tau$ can be computed cheaply level by level. Using the formula

$$\mathsf{A}^{-1} = \mathsf{E}(\tilde{\mathsf{A}} + \hat{\mathsf{D}})^{-1}\mathsf{F}^* + \mathsf{G}, \tag{20}$$

we can express A^{-1} hierarchically via

$$(\tilde{\mathsf{A}}^{(\ell)} + \hat{\mathsf{D}}^{(\ell)})^{-1} = \mathsf{E}^{(\ell-1)}(\tilde{\mathsf{A}}^{(\ell-1)} + \hat{\mathsf{D}}^{(\ell-1)})^{-1}(\mathsf{F}^{(\ell-1)})^{*} + \mathsf{G}^{(\ell-1)} \tag{21}$$

for $\ell = L, L-1, \ldots, 2$, and

$$(\tilde{\mathsf{A}}^{(1)} + \hat{\mathsf{D}}^{(1)})^{-1} = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix}^{-1} = \mathsf{G}^{(0)} = \mathsf{G}_1. \tag{22}$$

Algorithm 3 summarizes inversion procedure of HSS matrices.

7.5.2.2 Fast matrix-vector multiplication algorithm.

In this section, we briefly describe how to compute $\mathbf{y} = \mathsf{A}^{-1}\mathbf{x}$, for given $\mathbf{x}$, using the compressed representation of A^{-1} resulting from the inversion algorithm. Using equations (21) and (22), $\mathbf{y}$ can be computed hierarchically as

$$\begin{aligned}
\mathbf{y} = \mathsf{A}^{-1}\mathbf{x} &= (\tilde{\mathsf{A}}^{(L)} + \hat{\mathsf{D}}^{(L)})^{-1}\mathbf{x} \\
&= \mathsf{E}^{(L-1)}(\tilde{\mathsf{A}}^{(L-1)} + \hat{\mathsf{D}}^{(L-1)})^{-1}(\mathsf{F}^{(L-1)})^{*}\mathbf{x} + \mathsf{G}^{(L-1)}\mathbf{x} \\
&= \mathsf{E}^{(L-1)}\mathsf{E}^{(L-2)}(\tilde{\mathsf{A}}^{(L-2)} + \hat{\mathsf{D}}^{(L-2)})^{-1}(\mathsf{F}^{(L-2)})^{*}(\mathsf{F}^{(L-1)})^{*}\mathbf{x} + \mathsf{G}^{(L-2)}\mathsf{G}^{(L-1)}\mathbf{x} \\
&= \mathsf{E}^{(L-1)}\mathsf{E}^{(L-2)}\ldots\mathsf{E}^{(1)}\mathsf{G}_1(\mathsf{F}^{(1)})^{*}\ldots(\mathsf{F}^{(L-2)})^{*}(\mathsf{F}^{(L-1)})^{*}\mathbf{x} + \mathsf{G}^{(1)}\ldots\mathsf{G}^{(L-2)}\mathsf{G}^{(L-1)}\mathbf{x}.
\end{aligned}$$

Details on fast matrix-vector multiplication algorithm are given in Algorithm 4.

7.6 Accelerating the direct solver

In this section, we describe how to accelerate the direct solver in Section 7.4.2 using the fast algorithms described in Section 7.5.2 to achieve $O(N^{4/3})$ complexity. We claim that the $O(N^2)$ computational cost of the direct solver is dominated by executing matrix inversion at top levels which is

$$\mathsf{T}^{\tau} = \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,1} & 0 \\ 0 & \mathsf{T}^{\sigma_2}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,3} \\ \mathsf{T}^{\sigma_2}_{2,3} \end{bmatrix} (\mathsf{T}^{\sigma_1}_{3,3} - \mathsf{T}^{\sigma_2}_{3,3})^{-1}[-\mathsf{T}^{\sigma_1}_{3,1} \mid \mathsf{T}^{\sigma_2}_{3,2}],$$

```
ALGORITHM 3 (inversion of an HSS matrix)

Given factors $\{\mathsf{U}_\tau,\, \mathsf{V}_\tau,\, \mathsf{D}_\tau,\, \mathsf{B}_\tau\}_\tau$ representing an HBS matrix $\mathsf{H}$, this algorithm constructs factors
$\{\mathsf{E}_\tau,\, \mathsf{F}_\tau,\, \mathsf{G}_\tau\}_\tau$ representing $\mathsf{H}^{-1}$.

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \dots, 1$
    loop over all boxes $\tau$ on level $\ell$,
        if $\tau$ is a leaf node
            $\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$
        else
            Let $\sigma_1$ and $\sigma_2$ denote the children of $\tau$.
            $\tilde{\mathsf{D}}_\tau = \begin{bmatrix} \hat{\mathsf{D}}_{\sigma_1} & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & \hat{\mathsf{D}}_{\sigma_2} \end{bmatrix}$
        end if
        $\hat{\mathsf{D}}_\tau = \left(\mathsf{V}_\tau^*\, \tilde{\mathsf{D}}_\tau^{-1}\, \mathsf{U}_\tau\right)^{-1}$.
        $\mathsf{E}_\tau = \tilde{\mathsf{D}}_\tau^{-1}\, \mathsf{U}_\tau\, \hat{\mathsf{D}}_\tau$.
        $\mathsf{F}_\tau^* = \hat{\mathsf{D}}_\tau\, \mathsf{V}_\tau^*\, \tilde{\mathsf{D}}_\tau^{-1}$.
        $\mathsf{G}_\tau = \tilde{\mathsf{D}}_\tau^{-1} - \tilde{\mathsf{D}}_\tau^{-1}\, \mathsf{U}_\tau\, \hat{\mathsf{D}}_\tau\, \mathsf{V}_\tau^*\, \tilde{\mathsf{D}}_\tau^{-1}$.
    end loop
end loop
$\mathsf{G}_1 = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix}^{-1}$.
```

```
ALGORITHM 4 (application of the inverse of an HSS matrix)

Given $\mathbf{x}$, compute $\mathbf{y} = \mathsf{H}^{-1}\,\mathbf{x}$ using the factors $\{\mathsf{E}_\tau,\, \mathsf{F}_\tau,\, \mathsf{G}_\tau\}_\tau$ resulting from Algorithm 3.

loop over all leaf boxes $\tau$
    $\hat{\mathbf{x}}_\tau = \mathsf{F}_\tau^*\, \mathbf{x}(I_\tau)$.
end loop
loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \dots, 1$
    loop over all parent boxes $\tau$ on level $\ell$,
        Let $\sigma_1$ and $\sigma_2$ denote the children of $\tau$.
        $\hat{\mathbf{x}}_\tau = \mathsf{F}_\tau^* \begin{bmatrix} \hat{\mathbf{x}}_{\sigma_1} \\ \hat{\mathbf{x}}_{\sigma_2} \end{bmatrix}$.
    end loop
end loop
$\begin{bmatrix} \hat{\mathbf{y}}_2 \\ \hat{\mathbf{y}}_3 \end{bmatrix} = \mathsf{G}_1 \begin{bmatrix} \hat{\mathbf{x}}_2 \\ \hat{\mathbf{x}}_3 \end{bmatrix}$.
loop over all levels, coarser to finer, $\ell = 1,\, 2,\, \dots,\, L-1$
    loop over all parent boxes $\tau$ on level $\ell$
        Let $\sigma_1$ and $\sigma_2$ denote the children of $\tau$.
        $\begin{bmatrix} \hat{\mathbf{y}}_{\sigma_1} \\ \hat{\mathbf{y}}_{\sigma_2} \end{bmatrix} = \mathsf{E}_\tau\, \hat{\mathbf{x}}_\tau + \mathsf{G}_\tau \begin{bmatrix} \hat{\mathbf{x}}_{\sigma_1} \\ \hat{\mathbf{x}}_{\sigma_2} \end{bmatrix}$.
    end loop
end loop
loop over all leaf boxes $\tau$
    $\mathbf{y}(I_\tau) = \mathsf{E}_\tau\, \hat{\mathbf{q}}_\tau + \mathsf{G}_\tau\, \mathbf{x}(I_\tau)$.
end loop
```

where matrices $\mathsf{T}_{3,3}^{\sigma_1}$ and $\mathsf{T}_{3,3}^{\sigma_2}$ are of size $O(N^{2/3}) \times O(N^{2/3})$. Exploring the inner structures of the DtN operator, we claim that the off-diagonal blocks are low numerical rank and the diagonal blocks are HSS matrices with low HSS rank.

7.6.1 Memory efficiency.

In cases where memory efficiency is more important than time efficiency, we keep matrices $\mathsf{T}_{1,1}^{\sigma_1}$, $\mathsf{T}_{2,2}^{\sigma_2}$, $\mathsf{T}_{3,3}^{\sigma_1}$ and $\mathsf{T}_{3,3}^{\sigma_2}$ in dense form and perform low-rank computations on $\mathsf{T}_{1,3}^{\sigma_1}, \mathsf{T}_{3,1}^{\sigma_1}, \mathsf{T}_{2,3}^{\sigma_2}$ and $\mathsf{T}_{3,2}^{\sigma_2}$. The cause of rank deficiency is that for PDEs with non-oscillatory solutions, the corresponding DtN operators have smooth kernels. Technically, we compute QR factorizations to each blocks of $\mathsf{T}_{1,3}^{\sigma_1}, \mathsf{T}_{3,1}^{\sigma_1}, \mathsf{T}_{2,3}^{\sigma_2}$ and $\mathsf{T}_{3,2}^{\sigma_2}$ corresponding to interactions between different faces. For Laplace's and low-frequency Helmholtz problems, storing the QR factors takes much less memory than storing the whole dense matrices.

7.6.2 Memory and Time efficiency.

In the case for time efficiency, note that $\mathsf{T}_{1,1}^{\sigma_1}$ and $\mathsf{T}_{2,2}^{\sigma_2}$ are stored as HSS matrices as well as $\mathsf{T}_{3,3}^{\sigma_1}$ and $\mathsf{T}_{3,3}^{\sigma_2}$. Assume the low rank factors associated with $\mathsf{T}_{1,3}^{\sigma_1}, \mathsf{T}_{3,1}^{\sigma_1}, \mathsf{T}_{2,3}^{\sigma_2}$ and $\mathsf{T}_{3,2}^{\sigma_2}$ are $\{\mathsf{Q}_{1,3}, \mathsf{R}_{1,3}\}$, $\{\mathsf{Q}_{3,1}, \mathsf{R}_{3,1}\}$, $\{\mathsf{Q}_{2,3}, \mathsf{R}_{2,3}\}$ and $\{\mathsf{Q}_{3,2}, \mathsf{R}_{3,2}\}$. We execute the following computations to accelerate the build stage:

(1) Add two HSS matrices $\mathsf{T}_{3,3}^{\sigma_1}$ and $-\mathsf{T}_{3,3}^{\sigma_2}$ resulting a new HSS matrix $\mathsf{T}_{3,3}^{\sigma_1} - \mathsf{T}_{3,3}^{\sigma_2}$.

(2) Invert the HSS matrix $\mathsf{T}_{3,3}^{\sigma_1} - \mathsf{T}_{3,3}^{\sigma_2}$.

(3) Apply the thin low rank factor $\begin{bmatrix} \mathsf{R}_{1,3} \\ \mathsf{R}_{2,3} \end{bmatrix}$ to the inverse (in HSS form). The resulting matrix together with another low rank factor $[-\mathsf{Q}_{3,1} \mid \mathsf{Q}_{3,2}]$ form the low rank approximation to the solution operator S^{τ}.

(4) Performing matrix products $\mathsf{T}_{1,3}^{\sigma_1}\mathsf{S}^{\tau}$ and $\mathsf{T}_{2,3}^{\sigma_2}\mathsf{S}^{\tau}$ are analogous, just exploiting all factors are low rank.

(5) Perform a low-rank update to the block-diagonal matrix $\begin{bmatrix} \mathsf{T}^{\sigma_1}_{1,1} & 0 \\ 0 & \mathsf{T}^{\sigma_2}_{2,2} \end{bmatrix}$, whose blocks are provided in HSS-form to construct the new HSS matrix T^{τ}.

7.6.3 Complexity analysis.

There are two kinds of hierarchical structures involved in this algorithm. One is the hierarchical structure of the direct solver, described in Section 7.2. Within this "outer" tree, we store many matrices associated with large patches in a structured format that has its own hierarchical tree. This hierarchical structure is involved in the HSS representation of the DtN operators as described in Section 7.5.1.1.

7.6.3.1 Cost of fast algorithm of HSS matrices.

To investigate the cost of the inversion algorithm described in Section 7.5.2.1, notice that we need to compute $\hat{\mathsf{D}}^{\ell}_{\tau}, \mathsf{E}^{\ell}_{\tau}, \mathsf{F}^{\ell}_{\tau}, \mathsf{G}^{\ell}_{\tau}$ on each level ℓ. The cost is dominated by performing dense matrix inversion of matrices of size $2k \times 2k$ on each level, where there are 2^{ℓ} nodes totally. Therefore the total cost is

$$T_{\text{inv}} = \sum_{\ell=1}^{L} 2^{\ell}(2k)^3 \sim 2^{L+4}k^3 \sim Mk^2.$$

Moreover, the total cost of fast matrix-vector multiplication is

$$T_{\text{multi}} = \sum_{\ell=1}^{L} 2^{\ell}(2k)^2 \sim 2^{L+3}k^2 \sim Mk.$$

7.6.3.2 Cost of direct solver.

Let N be the total number of degrees of freedom to discretize the three-dimensional rectangular domain Ω and let L be the number of levels in the octree. Therefore there are 8^L leaf boxes. If on each leaf box, p^3 Chebyshev discretization nodes are placed to discretize the potential, then $N = 8^L p^3$.

Leaf computation: Since computing the DtN operators for each leaf box involves inverting dense matrices of $p^3 \times p^3$, the total cost for all the bottom level is

$$T_{\text{leaf}} = \frac{N}{p^3} \times (p^3)^3 \sim Np^6.$$

Merge operation: For each box on level ℓ, the operators T and S are constructed via (9) and (10), where the computation cost is dominated by inverting matrices of size $4^{-\ell}\,N^{2/3} \times 4^{-\ell}\,N^{2/3}$. Note that there are 8^ℓ boxes on each level. If the matrices are inverted *densely*, then the cost on each level ℓ of the merge operation is

$$8^\ell \times (4^{-\ell}N^{2/3})^3 \sim 2^{-3\ell}N^2.$$

The total cost for all merge operation has complexity

$$T_{\text{merge}} = \sum_{\ell=0}^{L-1} 2^{-3\ell}N^2 \sim N^2.$$

If we store the matrices in HSS form and execute fast inversion algorithms, then the cost on each level is

$$8^\ell \times (4^{-\ell}N^{2/3}k^2) \sim 2^\ell N^{2/3}k^2,$$

where k is the numerical rank in HSS compression and

$$k \sim 2^{-\ell}N^{1/3}. \tag{23}$$

Therefore all the merge procedures cost

$$T_{\text{merge}} = \sum_{\ell=0}^{L-1}(2^\ell N^{2/3})(2^{-\ell}N^{1/3})^2 \sim N^{4/3}.$$

Solve stage: The cost of solve stage is dominated by matrix-vector multiplication of applying the solution operator to the solution values on the exterior nodes. For each leaf box on the bottom level, the solution operator is of size $(p-2)^3 \times 6\,p^2$, while for any non-leaf box on level ℓ, the solution operator of size approximately $4^{-\ell}N^{2/3} \times 4^{-\ell}N^{2/3}$. Therefore the total cost of solve stage is

$$T_{\text{solve}} = \frac{N}{p^3} \times p^3\,p^2 + \sum_{\ell=0}^{L-1} 8^\ell(4^{-\ell}N^{2/3})^2 \sim Np^2 + \sum_{\ell=0}^{L-1} 2^{-\ell}N^{4/3} \sim Np^2 + N^{4/3}.$$

Remark 13 Notice that the estimation of the numerical rank in (23) is an upper bound at top levels. In practice, the performance should be better since at lower levels, the numerical ranks do not reach this upper bound.

Remark 14 The above derivation of the complexity on merge operator involving HSS structures is theoretically true when the number of wavelength across the domain is fixed. In practice, this asymptotic estimate works well for Laplace's and low-frequency Helmholtz problems. In the case when the number of wavelength increases along with the total number of discretization points, the above analysis is not going to hold anymore.

7.7 Numerical experiments

This section presents numerical experiments to explore the performance of the accelerated direct solver for solving three-dimensional elliptical equations. All the experiments are carried out on a personal work-station with an Intel Xeon E-1660 3.3GHz 6-core CPU, and 128GB of RAM. The experiments serves two purposes. The first is to systematically measure the speed and memory requirements (the amount of RAM used in the build stage in GB) for different problems. The second is to measure the accuracy of the algorithm. Specifically, we report:

N_{tot}	Total number of Chebyshev discretization nodes
T_{build}	Time for building the solution operator
T_{solve}	Time to solve for solution at interior nodes once solution operator is built
R	Amount of memory required at build stage to store the solution operator.

Moreover, for all the experiments in this section, we choose the compression parameter $\epsilon = 10^{-5}$ which is small enough to obtain at least eight digits of accuracy. Finally, due to the overhead cost of HSS construction, we only execute HSS compression stated in Section 7.6 at the highest level.

Example 7.1: Laplace problem with known exact solution

We first consider the Laplace problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{24}$$

on domain $\Omega = [0, 1]^3$. The number of Chebyshev nodes are fixed $p = 5$ on each panel while the number of panels on each side is increased. The boundary data is chosen to coincide with the known solution

$$u_{\text{exact}}(\boldsymbol{x}) = \frac{1}{4\pi|\boldsymbol{x} - \hat{\boldsymbol{x}}|}$$

where $\hat{\boldsymbol{x}} = (-2, -1, 0)$. To measure the accuracy, we present both the error E^{∞} in ℓ^{∞}-norm as well as the relative error E^{rel} given by

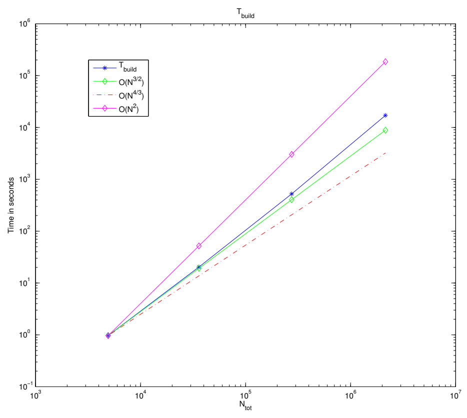
$$E^{\infty} = ||\mathbf{u}_{\text{approx}} - \mathbf{u}_{\text{exact}}||_{\ell^{\infty}} \quad \text{and} \quad E^{\text{rel}} = \frac{||\mathbf{u}_{\text{approx}} - \mathbf{u}_{\text{exact}}||_{\ell^{\infty}}}{||\mathbf{u}_{\text{exact}}||_{\ell^{\infty}}},$$

where $\mathbf{u}_{\text{exact}}$ and $\mathbf{u}_{\text{approx}}$ denote the vectors holding the exact and the computed solutions evaluated at all interior Chebyshev nodes. Speed, memory and accuracy results are shown in Table 7.1. With only 20 seconds of build time and 0.032 second of solve time, we are able to obtain 7 digits of accuracy. Note that the off-diagonal blocks of DtN operators at top levels are stored in low-rank form which saves roughly 1.6 times of the memory compared to dense form. If no low-rank computation is executed, the computation runs out of memory (over 128GB) at degrees of freedom 2 146 689. Figure 7.3 shows the scales of time spent at build stage and solve stage.

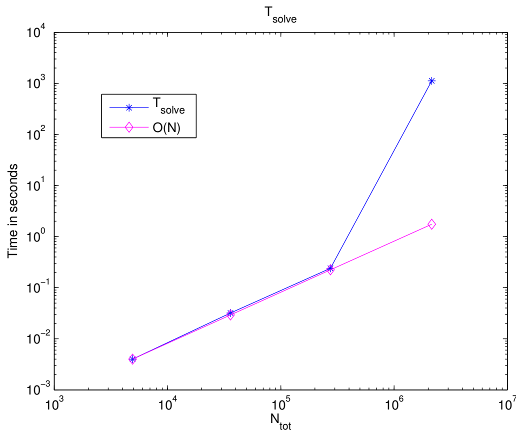
N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	E^{∞}	E^{rel}
4 913	0.04	0.97	0.004	1.20e-06	3.38e-05
35 937	0.52	20.34	0.032	1.45e-08	4.08e-07
274 625	6.33	522.78	0.24	5.48e-08	1.54e-07
2 146 689	76.59	17103.21	1121.0	6.51e-09	1.83e-07

Table 7.1: Results for solving Laplace's equation (24) in Example 7.1 with known exact solution.

Example 7.2: Helmholtz problems with known exact solution



(a)



(b)

Figure 7.3: (Example 7.1)(a) Time at build stage in seconds, (b) time at solve stage.

We next consider a Helmholtz problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \\ \dfrac{\partial u(\boldsymbol{x})}{\partial r} - i\kappa u(\boldsymbol{x}) = O(1/r) & r := |\boldsymbol{x}| \to \infty, \end{cases} \tag{25}$$

on domain $\Omega = [0,\, 1]^3$. In this example and the following examples, we fixed the number of panels on each side to be eight and change the number Chebyshev nodes on each panel. The boundary data is chosen to coincide with the known solution

$$u_{\text{exact}}(\boldsymbol{x}) = \frac{e^{i\kappa|\boldsymbol{x}-\hat{\boldsymbol{x}}|}}{4\pi|\boldsymbol{x}-\hat{\boldsymbol{x}}|}$$

where $\hat{\boldsymbol{x}} = (-2,\, -1,\, 0)$. Accuracy is measured in the same way as Example 7.1. Table 7.7 reports the results when $\kappa = 62.8$ such that there are $10 \times 10 \times 10$ wavelengths across the whole domain. Table 7.7 reports an analogous experiment, but now for a domain of size $20 \times 20 \times 20$ wavelengths.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	E^{∞}	E^{rel}
274 625	6.79	850.5	0.2	1.55e-03	4.28e-02
531 441	15.03	2427.2	0.6	6.51e-05	1.80e-03
912 673	29.51	4967.0	1.1	1.83e-06	5.06e-05
1 442 897	52.80	10133.1	2.7	3.57e-08	9.86e-07
2 146 689	89.15	19831.9	558.8	1.14e-08	3.15e-07

Table 7.2: Results for solving Helmholtz equation (25) in Example 7.2 with $10 \times 10 \times 10$ wavelength across the domain.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	E^{∞}	E^{rel}
274 625	8.65	1034.3	0.2	1.34e+00	3.76e+01
531 441	18.40	2910.6	0.5	1.70e-01	4.78e+00
912 673	34.55	7573.7	1.1	7.50e-03	2.11e-01
1 442 897	59.53	14161.1	2.8	9.45e-04	2.65e-02
2 146 689	97.73	25859.3	978.7	5.26e-05	1.48e-03

Table 7.3: Results for solving Helmholtz equation (25) in Example 7.2 with $20 \times 20 \times 20$ wavelength across the domain.

Example 7.3: Laplace's equation with unknown exact solution

In this example, we solve the Laplace's equation (24) with Dirichlet boundary data given by

$$f(\boldsymbol{x}) = \cos(8\, x_1)(1 - 2\, x_2)\, e^{x_3}. \tag{26}$$

Since we have no knowledge of an exact solution, we report pointwise convergence. Letting u_{N_1} and u_{N_2} denote the value of u computed using N_1 and N_2 degrees of freedom where $N_2 > N_1$, we used

$$E^{\infty} = |u^{N_1}(\hat{\boldsymbol{x}}) - u^{N_2}(\hat{\boldsymbol{x}})| \quad \text{and} \quad E^{\text{rel}} = \frac{|u^{N_1}(\hat{\boldsymbol{x}}) - u^{N_2}(\hat{\boldsymbol{x}})|}{|u^{N_1}(\hat{\boldsymbol{x}})|}$$

as estimations for the pointwise errors at point $\hat{\boldsymbol{x}} = (0.5, 0.25, 0.75)$. Results are reported in Table 7.4.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	$u_{\text{int}}(\boldsymbol{x})$	E^{∞}	E^{rel}
117 649	2.13	84.7	0.06	-0.32055891842	1.10e-06	3.44e-06
274 625	6.09	540.1	0.2	-0.32055781677	9.43e-08	2.94e-07
531 441	14.35	1517.3	0.4	-0.32055772259	2.56e-08	7.98e-08
912 673	29.11	2822.4	0.7	-0.32055769701	1.24e-07	3.87e-07
1 442 897	50.44	9130.9	1.4	-0.32557572862	7.34e-08	2.29e-07
2 146 689	86.12	18076.5	541.5	-0.32055776368		

Table 7.4: Results for solving Laplace's equation (24) with Dirichlet boundary data (26).

Example 7.4: Helmholtz equation with unknown exact solution

In this example, we solve the Helmholtz equation (25) with Dirichlet boundary data given in (26). The wavenumber is set $\kappa = 62.8$ such that there are $10\times10\times10$ wavelength across the domain. Table 7.5 presents the convergence as well as time and memory results. Pointwise errors are computed the same as Example 7.3.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	$u_{\text{int}}(\boldsymbol{x})$	E^{∞}	E^{rel}
274 625	6.58	662.9	0.2	1.93279205417	1.87e+00	9.68e-01
531 441	15.13	1382.9	0.3	3.80381724805	9.55e-02	2.51e-02
912 673	30.25	2992.5	0.7	3.70818462313	2.91e-03	7.84e-04
1 442 897	55.23	7895.9	2.3	3.71109259971	6.16e-05	1.66e-05
2 146 689	89.15	21190.2	789.8	3.71103088491		

Table 7.5: Results for solving Helmholtz equation (25) with Dirichlet boundary data (26).

Example 7.5: Variable coefficient Helmholtz

We solve the variable coefficient problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2(1-b(\boldsymbol{x}))u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{27}$$

where $\Omega = [0,\, 1]^3$, where $\Gamma = \partial\Omega$ and where

$$b(\boldsymbol{x}) = (\sin(4\pi x_1)\, \sin(4\pi x_2)\, \sin(4\pi x_3))^2.$$

The Helmholtz parameter was chosen as $\kappa = 62.8$, corresponding to a domain of size $10 \times 10 \times 10$ wavelengths. The Dirichlet boundary data was given by in (26). Again we measure the pointwise errors since we don't know the exact solution. Results are reported in Table 7.6. We observe that the accuracy as almost as good as constant coefficient case.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	$u_{\text{int}}(\boldsymbol{x})$	E^{∞}	E^{rel}
274 625	6.55	639.3	0.2	10.22765480303	6.13e-02	5.99e-03
531 441	15.15	1443.4	0.3	10.16634235402	8.69e-03	8.55e-04
912 673	30.35	3701.0	0.7	10.17503224623	4.56e-04	4.48e-05
1 442 897	55.39	5639.6	1.4	10.17548843592	1.35e-04	1.33e-05
2 146 689	89.27	20854.3	874.7	10.17535090141		

Table 7.6: Results for solving variable coefficient problem (27).

Example 7.6: Constant convection diffusion problem

Our last example is to solve a convection diffusion problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - 1000\, \partial_3 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{28}$$

on domain $\Omega = [0,\, 1]^3$ with Dirichlet boundary data given in (26).

7.8 Conclusions and Future work

In this chapter, we present a numerical algorithm to solve three-dimensional variable coefficient elliptic equation on rectangular domains, under the assumption that the solution as well as its first, second and third derivatives are smooth functions. The scheme is based on a composite

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	$u_{\text{int}}(\boldsymbol{x})$	E^{∞}	E^{rel}
117 649	2.41	136.0	0.08	-0.90989632585	3.33e-02	3.66e-02
274 625	6.65	524.9	0.1	-0.87662189265	2.30e-03	2.62e-02
531 441	15.33	1806.0	0.3	-0.87432365086	4.36e-05	4.99e-05
912 673	30.45	3524.9	0.7	-0.87428002798	1.23e-05	1.41e-05
1 442 897	55.35	6719.9	1.3	-0.87429233218	3.30e-06	3.77e-06
2 146 689	88.03	19313.7	656.2	-0.87429562890		

Table 7.7: Results for solving constant convection problem (28).

spectral discretization and is an extension of the work in two dimensional case [79, 35]. To improve the asymptotic complexity from $O(N^2)$ to $O(N^{4/3})$, we explore HSS and low-rank structures of matrix representation of the Dirichlet-to-Neumann maps at top levels. Once the solution operator is built, executing solve stage is extremely fast.

The direct solver requires more storage than classical iterative solvers because of the use of high-order discretizations. Right now memory is the major limit to the algorithm since we have to store most of the matrices in dense form in RAM. In Section 7.6.3, we theoretically derive the computational cost at build stage is $O(N^{4/3})$. However, in numerical experiments presented in Section 7.7, due to the limitation of the problems sizes we are not able to achieve this asymptotic complexity. The build time presented by the numerical experiments scale as $O(N^{1.5})$. However, since the computation of the DtN operators are highly localized, the scheme is particularly well suited for implementations on parallel machines with distributed memory. This is a future direction to improve the performance of the algorithm.

Chapter 8

Conclusion

A collection of fast and highly accurate direct solvers for elliptic boundary value problems are presented in this dissertation.

Converting the boundary value problems to boundary integral equations is a widely used method for solving problems whose fundamental solutions are known. The resulting reduced dimensionality and geometric simplicity allow for high-order accurate numerical solutions with much more efficiency than standard finite-difference or finite element discretizations. Recently, a set of high-order Nyström discretization schemes have been developed to resolve the difficulties brought by the logrithmic singularity when utilizing the BIE method to solve elliptic partial differential equations. Numerical examples in Chapter 2 show that these schemes are highly accurate. For example, with only 1600 modified Gaussian discretization nodes the BIE associated with the Helmholtz problem with up to 50 wavelength across the domain can be solved to 11 to 12 digits of accuracy. These discretization schemes also makes it possible to solve BIEs defined on piecewise smooth contours in plane where corners are exhibited to high-order accuracy. In the algorithm presented in Chapter 3, we solve such problems via a general direct solver where the panel based quadrature rules are used to discretize the contour. The degrees of freedom added to refine the corners can be largely reduced via a local compression scheme. Numerical experiments show that the algorithm can be executed in time that scales linearly with the number of degrees of freedom added.

In Chapters 4 and 5, a robust and highly accurate numerical method for modeling frequency domain acoustic scattering on domain external to a single scatter and a group of scatterers in three dimensions is presented where the scatterers are rotational symmetric. The algorithm relies on a boundary integral equation formulation of the scattering problem, combined with a highly accurate Nyström discretization technique. A dense linear system is formed by constructing the scattering matrix for each scatter and is solved via GMRES in which Fast Multiple Method is used to accelerate all inter-body interactions. The algorithm is highly accurate. For example, the scheme quickly reaches 9 digits of accuracy for solving Laplace's equation exterior to a domain having 125 ellipsoids that are lying closely with 10 100 discretization nodes per scatterer, with an overall solve time of about 40 minutes. We also present an accelerated scheme that enables us to reduce the problem size greatly for problems where the scatterers are well-separated thus to fit the problem on a basic personal work station. In one numerical example on solving Helmholtz problem, the numbers of degrees of freedom to reach seven digits of accuracy was in one example reduced by a factor of 65; consequently the overall computation time is reduced from 48 hours to 5 hours.

For variable coefficient problems that are defined on rectangular domains, we developed a fast and accurate scheme combining a spectral multidomain technique with a hierarchical direct solver. One advantage is that once the solution operator has been built, solves can be executed extremely rapidly, making the scheme excel when solving a sequence of equations with the same operator but different boundary data. Each additional solve only takes 1.5 seconds for problems discretized with over one million degrees of freedom. Based on the direct solver, we also proposed a refinement scheme that enables us to retain high-order accuracy for problems exhibiting change of regularity of the solution across the domain. For example, for Helmholtz problem involving non-smooth Dirichlet boundary data on a domain of size 10×10 wavelengths, the method takes 8 to 16 levels of refinements to achieve 9 to 12 digits of accuracy. We also extend the scheme to solving variable coefficient problems on three-dimensional rectangular domains. Similar to the linear system that arises from the discretization of many boundary integral equation, the dense matrices representing DtN maps have internal structure that enables us to improve the computational complexity from $O(N^2)$ to $O(N^{4/3})$. Numerical examples carried on a personal workstation show that the scheme is highly accurate to solve Laplace and low-frequency Helmholtz problems. However, memory constraints become far more limiting than for problems in 2D. We expect the algorithm to show its power on machines with larger memories.

Bibliography

[1] M. Abramowitz and I. Stegun, Handbook of mathematical functions with formulas, graphs, and mathematical tables, Dover, New York, 1965.

[2] B. K. Alpert, Hybrid gauss-trapezoidal quadrature rules, SIAM J. Sci. Comput., 20 (1999), pp. 1551–1584.

[3] K. Atkinson, The numerical solution of integral equations of the second kind, Cambridge University Press, Cambridge, 1997.

[4] K. E. Atkinson, The numerical solution of integral equations of the second kind, Cambridge University Press, Cambridge, 1997.

[5] A. Bakr, The boundary integral equation method in axisymmetric stress analysis problems, Springer-Verlag, Berlin, 1985.

[6] J. Barnes and P. Hut, A hierarchical $o(n \log n)$ force-calculation algorithm, Nature, 324 (1986).

[7] J. Berrut and L. Trefethen, Barycentric lagrange interpolation, SIAM Review, 46 (2004), pp. 501–517.

[8] G. Beylkin, R. Coifman, and V. Rokhlin, Wavelets in numerical analysis, Wavelets and their applications, (1992), pp. 181–210.

[9] J. Bremer, A fast direct solver for the integral equations of scattering theory on planar curves with corners, Journal of Computational Physics, 231 (2012), pp. 1879 – 1899.

[10] J. Bremer, A fast direct solver for the integral equations of scattering theory on planar curves with corners, Journal of Computational Physics, 231 (2012), pp. 1879–1899.

[11] ——, A fast direct solver for the integral equations of scattering theory on planar curves with corners, Journal of Computational Physics, 231 (2012), pp. 1879–1899.

[12] J. Bremer, On the nyström discretization of integral equations on planar curves with corners, Applied and Computational Harmonic Analysis, 32 (2012), pp. 45–64.

[13] J. Bremer, A. Gillman, and P.-G. Martinsson, A high-order accurate accelerated direct solver for acoustic scattering from surfaces, arXiv preprint arXiv:1308.6643, (2013).

[14] J. Bremer and V. Rokhlin, Efficient discretization of Laplace boundary integral equations on polygonal domains, J. Comput. Phys., 229 (2010), pp. 2507–2525.

[15] J. Bremer, V. Rokhlin, and I. Sammis, Universal quadratures for boundary integral equations on two-dimensional domains with corners, Journal of Computational Physics, 229 (2010), pp. 8259 – 8280.

[16] ——, Universal quadratures for boundary integral equations on two-dimensional domains with corners, Journal of Computational Physics, 229 (2010), pp. 8259 – 8280.

[17] B. Briggs and V. Henson, The DFT: An Owner's Manual for the Discrete Fourier Transform, SIAM, Philadelphia, 1995.

[18] O. Bruno and L. Kunyansky, A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, test, and applications, J. Comput. Phys., 169 (2001), pp. 80–110.

[19] S. Chandrasekaran, M. Gu, X. Li, and J. Xia, Some fast algorithms for hierarchically semiseparable matrices, Tech. Rep. 08-24, UCLA/CAM, 2008.

[20] Y. Chen, A fast, direct algorithm for the lippmann-schwinger integral equation in two dimensions, Adv. Comp. Math., 16 (2002), pp. 175–190.

[21] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, A wideband fast multipole method for the Helmholtz equation in three dimensions, J. Comput. Phys., 216 (2006), pp. 300–325.

[22] H. Cheng, Z. Gimbutas, P. Martinsson, and V. Rokhlin, On the compression of low rank matrices, SIAM Journal of Scientific Computing, 26 (2005), pp. 1389–1404.

[23] ——, On the compression of low rank matrices, SIAM Journal of Scientific Computing, 26 (2005), pp. 1389–1404.

[24] H. Cheng, V. Rokhlin, and N. Yarvin, Nonlinear optimization, quadrature, and interpolation, SIAM Journal on Optimization, 9 (1999), pp. 901–923.

[25] H. Cohl and J. Tohline, A compact cylindrical green's function expansion for the solution of potential problems, Astrophys. J., 527 (1999), pp. 86–101.

[26] D. Colton and R. Kress, Inverse Acoustic and Electromagnetic Scattering Theory, Springer-Verlag, New York, 2nd ed., 1998.

[27] D. Colton and R. Kress, Inverse acoustic and electromagnetic scattering theory, vol. 93 of Applied Mathematical Sciences, Springer-Verlag, Berlin, second ed., 1998.

[28] Z.-H. Duan and R. Krasny, An adaptive treecode for computing nonbonded potential energy in classical molecular systems, Journal of Computational Chemistry, 22 (2001), pp. 184–195.

[29] I. Duff, A. Erisman, and J. Reid, Direct Methods for Sparse Matrices, Oxford, 1989.

[30] J. Fleming, A. Wood, and W. W. Jr., Locally corrected nyström method for em scattering by bodies of revolution, J. Comput. Phys., 196 (2004), pp. 41–52.

[31] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, 1996.

[32] A. Gil, J. Segura, and N. Temme, Numerical methods for special functions, SIAM, Philadelphia, 2007.

[33] A. Gillman, A. Barnett, and P.-G. Martinsson, A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media, BIT Numerical Mathematics, (2014), pp. 1–30.

[34] A. Gillman, S. Hao, and P. Martinsson, Short note: A simplified technique for the efficient and highly accurate discretization of boundary integral equations in 2d on domains with corners, Journal of Computational Physics, 256 (2014), pp. 214–219.

[35] A. Gillman and P. G. Martinsson, A direct solver with $o(n)$ complexity for variable coefficient elliptic pdes discretized via a high-order composite spectral collocation method, SIAM Journal on Scientific Computing, 36 (2014), pp. A2023–A2046.

[36] A. Gillman, P. Young, and P.-G. Martinsson, A direct solver $o(n)$ complexity for integral equations on one-dimensional domains, Frontiers of Mathematics in China, 7 (2012), pp. 217–247. 10.1007/s11464-012-0188-3.

[37] A. Gillman, P. Young, and P.-G. Martinsson, A direct solver with o(n) complexity for integral equations on one-dimensional domains, Frontiers of Mathematics in China, 7 (2012), pp. 217–247.

[38] A. Gillman, P. Young, and P.-G. Martinsson, Numerical homogenization via approximation of the solution operator, in Numerical Analysis of Multiscale Computations, B. Engquist, O. Runborg, and Y.-H. R. Tsai, eds., vol. 82 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2012, pp. 187–216.

[39] Z. Gimbutas and L. Greengard, FMMLIB3D, fortran libraries for fast multiple method in three dimensions, 2011. http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html.

[40] Z. Gimbutas and L. Greengard, Fast multi-particle scattering: A hybrid solver for the maxwell equations in microstructured materials., J. Comput. Physics, 232 (2013), pp. 22–32.

[41] L. Grasedyck, R. Kriemann, and S. Le Borne, Domain decomposition based h -lu preconditioning, Numerische Mathematik, 112 (2009), pp. 565–600.

[42] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, Fast direct solvers for integral equations in complex three-dimensional domains, Acta Numer., 18 (2009), pp. 243–275.

[43] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys., 73 (1987), pp. 325–348.

[44] R. Guenther and J. Lee, Partial differential equations of mathematical physics and integral equations, Dover, New York, 1988.

[45] A. Gupta, The boundary integral equation method for potential problems involving axisymmetric geometry and arbitrary boundary conditions, Master's thesis, University of Kentucky, 1979.

[46] W. Hackbusch, A sparse matrix arithmetic based on H-matrices; Part I: Introduction to H-matrices, Computing, 62 (1999), pp. 89–108.

[47] W. Hackbusch, B. Khoromskij, and S. Sauter, On $\mathcal{H}^2$-matrices, in Lectures on Applied Mathematics, Springer Berlin, 2002, pp. 9–29.

[48] N. Halko, P. G. Martinsson, and J. A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev., 53 (2011), pp. 217–288.

[49] S. Hao, A. Barnett, and P. Martinsson, Nyström quadratures for BIEs with weakly singular kernels on 1D domains, 2012. http://amath.colorado.edu/faculty/martinss/Nystrom/.

[50] S. Hao, A. Barnett, P. Martinsson, and P. Young, High-order accurate nystrom discretization of integral equations with weakly singular kernels on smooth curves in the plane, 2011. arXiv.org report #1112.6262.

[51] S. Hao, A. Barnett, P. Martinsson, and P. Young, High-order accurate methods for nyström discretization of integral equations on smooth curves in the plane, Advances in Computational Mathematics, (2013), pp. 1–28.

[52] J. Helsing, Integral equation methods for elliptic problems with boundary conditions of mixed type, J. Comput. Phys., 228 (2009), pp. 8892–8907.

[53] J. Helsing, The effective conductivity of arrays of squares: large random unit cells and extreme contrast ratios, Journal of Computational Physics, 230 (2011), pp. 7533–7547.

[54] J. Helsing, A fast and stable solver for singular integral equations on piecewise smooth curves, SIAM J. Sci. Comput., 33 (2011), pp. 153–174.

[55] J. Helsing, A fast and stable solver for singular integral equations on piecewise smooth curves, SIAM Journal on Scientific Computing, 33 (2011), pp. 153–174.

[56] J. Helsing, Solving integral equations on piecewise smooth boundaries using the RCIP method: a tutorial, 2012. preprint, 34 pages, `arXiv:1207.6737v3`.

[57] J. Helsing and A. Karlsson, An accurate boundary value problem solver applied to scattering from cylinders with corners, arXiv preprint arXiv:1211.2467, (2012).

[58] J. Helsing and R. Ojala, Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning, J. Comput. Phys., 227 (2008), pp. 8820–8840.

[59] J. Helsing and R. Ojala, Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning, Journal of Computational Physics, 227 (2008), pp. 8820 – 8840.

[60] J. Helsing and R. Ojala, Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning, J. Comput. Phys., 227 (2008), pp. 8820–8840.

[61] R. Henderson, Adaptive spectral element methods for turbulence and transition, in High-Order Methods for Computational Physics, T. Barth and H. Deconinck, eds., vol. 9 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 1999, pp. 225–324.

[62] K. Ho and L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, SIAM Journal on Scientific Computing, 34 (2012), pp. 2507–2532.

[63] L. Ho and L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, SIAM J. on Scientific Computing, 34 (2012), pp. A2507–A2532.

[64] S. Kapur and V. Rokhlin, High-order corrected trapezoidal quadrature rules for singular functions, SIAM J. Numer. Anal., 34 (1997), pp. 1331–1356.

[65] A. Klöckner, A. H. Barnett, L. Greengard, and M. O'Neil, Quadrature by expansion: a new method for the evaluation of layer potentials, 2012. submitted.

[66] P. Kolm and V. Rokhlin, Numerical quadratures for singular and hypersingular integrals, Comput. Math. Appl., 41 (2001), pp. 327–352.

[67] ——, Numerical quadratures for singular and hypersingular integrals, Comput. Math. Appl., 41 (2001), pp. 327–352.

[68] R. Kress, On constant-alpha force-free fields in a torus, Journal of Engineering Mathematics, 20 (1986), pp. 323–344.

[69] R. Kress, Boundary integral equations in time-harmonic acoustic scattering, Mathematical and Computer Modelling, 15 (1991), pp. 229–243.

[70] R. Kress, Boundary integral equations in time-harmonic acoustic scattering, Mathl. Comput. Modelling, 15 (1991), pp. 229–243.

[71] R. Kress, Linear Integral Equations, vol. 82 of Applied Mathematical Sciences, Springer, second ed., 1999.

[72] R. Kress and W. Spassov, On the condition of boundary integral operators for the exterior dirichlet problem for the Helmholtz equation, Numer. Math., 42 (1983), pp. 77–95.

[73] A. Kuijpers, G. Verbeek, and J. Verheij, An improved acoustic fourier boundary element method formulation using fast fourier transform integration, J. Acoust. Soc. Am., 102 (1997), pp. 1394–1401.

[74] S. Le Borne, L. Grasedyck, and R. Kriemann, Domain-decomposition based ?-lu preconditioners, in Domain decomposition methods in science and engineering XVI, Springer, 2007, pp. 667–674.

[75] E. Martensen, Über eine methode zum räumlichen neumannschen problem mit einer anwendung für torusartige berandungen, Acta mathematica, 109 (1963), pp. 75–135.

[76] P. Martinsson and V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys., 205 (2004), pp. 1–23.

[77] ——, A fast direct solver for boundary integral equations in two dimensions, Journal of Computational Physics, 205 (2005), pp. 1 – 23.

[78] P. Martinsson and V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comp. Phys., 205 (2005), pp. 1–23.

[79] P.-G. Martinsson, A direct solver for variable coefficient elliptic pdes discretized via a composite spectral collocation method, J. Comput. Phys., 242 (2013), pp. 460–479.

[80] E. Michielssen, A. Boag, and W. Chew, Scattering from elongated objects: direct solution in o(n log2 n) operations, Microwaves, Antennas and Propagation, IEE Proceedings, 143 (1996), pp. 277–283.

[81] J.-C. Nédélec, Acoustic and Electromagnetic Equations: Integral Representations for Harmonic Functions, Springer, New York, 2012.

[82] E. Nyström, Über die praktische Auflösung von Integralgleichungen mit Andwendungen aug Randwertaufgaben, Acta Math., 54 (1930), pp. 185–204.

[83] R. Ojala, Towards an All-Embracing Elliptic Solver in 2D, PhD thesis, Department of Mathematics, Lund University, Sweden, 2011.

[84] H. P. Pfeiffer, L. E. Kidder, M. A. Scheel, and S. A. Teukolsky, A multidomain spectral method for solving elliptic equations, Computer Physics Communications, 152 (2003), pp. 253 – 273.

[85] C. Provatidis, A boundary element method for axisymmetric potential problems with non-axisymmetric boundary conditions using fast fourier transform, Engrg. Comput., 15 (1998), pp. 428–449.

[86] F. Rizzo and D. Shippy, A boundary integral approach to potential and elasticity problems for axisymmetric bodies with arbitrary boundary conditions, Mech. Res. Commun., 6 (1979), pp. 99–103.

[87] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2nd ed. ed., 2003.

[88] Y. Saad and M. H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on scientific and statistical computing, 7 (1986), pp. 856–869.

[89] P. G. Schmitz and L. Ying, A fast direct solver for elliptic problems on general meshes in 2d, Journal of Computational Physics, 231 (2012), pp. 1314 – 1338.

[90] A. F. Seybert, B. Soenarko, F. J. Rizzo, and D. J. Shippy, A special integral equation formulation for acoustic radiation and scattering for axisymmetric bodies and boundary conditions, The Journal of the Acoustical Society of America, 80 (1986).

[91] D. Shippy, F. Rizzo, and A. Gupta, Boundary-integral solution of potential problems involving axisymmetric bodies and nonsymmetric boundary conditions, in Developments in Theoretical and Applied Mechanics, J. Stoneking, ed., 1980, pp. 189–206.

[92] B. Soenarko, A boundary element formuluation for radiation of acoustic waves from axisymmetric bodies with arbitrary boundary conditions, J. Acoust. Soc. Am., 93 (1993), pp. 631–639.

[93] P. Starr and V. Rokhlin, On the numerical solution of two-point boundary value problems ii, Communications on Pure and Applied Mathematics, 47 (1994), pp. 1117–1159.

[94] L. Trefethen, Spectral Methods in Matlab, SIAM, Philadelphia, 2000.

[95] L. N. Trefethen, Spectral methods in MatLab, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[96] ——, Spectral methods in MATLAB, vol. 10 of Software, Environments, and Tools, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[97] L. N. Trefethen, Approximation Theory and Approximation Practice, SIAM, 2012. `http://www.maths.ox.ac.uk/chebfun/ATAP`.

[98] S. Tsinopoulos, J. Agnantiaris, and D. Polyzos, An advanced boundary element/fast fourier transform axisymmetric formulation for acoustic radiation and wave scattering problems, J. Acoust. Soc. Am., 105 (1999), pp. 1517–1526.

[99] W. Wang, N. Atalla, and J. Nicolas, A boundary integral approach for accoustic radiation of axisymmetric bodies with arbitrary boundary conditions valid for all wave numbers, J. Acoust. Soc. Am., 101 (1997), pp. 1468–1478.

[100] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, Fast algorithms for hierarchically semiseparable matrices, Numerical Linear Algebra with Applications, 17 (2010), pp. 953–976.

[101] B. Yang and J. Hesthaven, Multidomain pseudospectral computation of maxwell's equations in 3-d general curvilinear coordinates, Applied Numerical Mathematics, 33 (2000), pp. 281 – 289.

[102] L. Ying, G. Biros, and D. Zorin, A kernel-independent adaptive fast multipole method in two and three dimensions, J. Comput. Phys., 196 (2004), pp. 591–626.

[103] P. Young, S. Hao, and P. G. Martinsson, A high-order nyström discretization scheme for boundary integral equations defined on rotationally symmetric surfaces, J. Comput. Phys., 231 (2012), pp. 4142–4159.

[104] P. M. Young, S. Hao, and P. G. Martinsson, A high-order Nyström discretization scheme for boundary integral equations defined on rotationally symmetric surfaces, J. Comput. Phys., 231 (2012), pp. 4142–4159.

Appendix A

Tables of Kapur–Rokhlin Quadrature weights

2^{nd}-order Kapur–Rokhlin correction weights integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$	
INDEX l	WEIGHTS $\gamma_l + \gamma_{-l}$
1	1.825748064736159e+00
2	-1.325748064736159e+00

6^{th}-order Kapur–Rokhlin correction weights integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$	
INDEX l	WEIGHTS $\gamma_l + \gamma_{-l}$
1	4.967362978287758e+00
2	-1.620501504859126e+01
3	2.585153761832639e+01
4	-2.222599466791883e+01
5	9.930104998037539e+00
6	-1.817995878141594e+00

10th-order Kapur–Rokhlin correction weights integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$	
INDEX l	WEIGHTS $\gamma_l + \gamma_{-l}$
1	7.832432020568779e+00
2	-4.565161670374749e+01
3	1.452168846354677e+02
4	-2.901348302886379e+02
5	3.870862162579900e+02
6	-3.523821383570681e+02
7	2.172421547519342e+02
8	-8.707796087382991e+01
9	2.053584266072635e+01
10	-2.166984103403823e+00

Appendix B

Tables of Alpert Quadrature rules

2^{nd}-order Alpert Quadrature Rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$, with $a = 1$	
NODES	WEIGHTS
1.591549430918953e-01	5.000000000000000e-01

6^{th}-order Alpert Quadrature Rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$, with $a = 3$	
NODES	WEIGHTS
4.004884194926570e-03	1.671879691147102e-02
7.745655373336686e-02	1.636958371447360e-01
3.972849993523248e-01	4.981856569770637e-01
1.075673352915104e+00	8.372266245578912e+00
2.003796927111872e+00	9.841730844088381e+00

10th-order Alpert Quadrature Rule for integrals of the form $\int_0^1 f(x) + g(x) \log(x)\, dx$, with $a = 6$	
NODES	WEIGHTS
1.175089381227308e-03	4.560746882084207e-03
1.877034129831289e-02	3.810606322384757e-02
9.686468391426860e-02	1.293864997289512e-01
3.004818668002884e-01	2.884360381408835e-01
6.901331557173356e-01	4.958111914344961e-01
1.293695738083659e+00	7.077154600594529e-01
2.090187729798780e+00	8.741924365285083e-01
3.016719313149212e+00	9.661361986515218e-01
4.001369747872486e+00	9.957887866078700e-01
5.000025661793423e+00	9.998665787423845e-01

16th-order Alpert Quadrature Rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x)\,dx$, with $a = 10$

NODES	WEIGHTS
8.371529832014113e-04	3.190919086626234e-03
1.239382725542637e-02	2.423621380426338e-02
6.009290785739468e-02	7.740135521653088e-02
1.805991249601928e-01	1.704889420286369e-01
4.142832599028031e-01	3.029123478511309e-01
7.964747731112430e-01	4.652220834914617e-01
1.348993882467059e+00	6.401489637096768e-01
2.073471660264359e+00	8.051212946181061e-01
2.947904939031494e+00	9.362411945698647e-01
3.928129252248612e+00	1.014359775369075e+00
4.957203086563112e+00	1.035167721053657e+00
5.986360113977494e+00	1.020308624984610e+00
6.997957704791519e+00	1.004798397441514e+00
7.999888757524622e+00	1.000395017352309e+00
8.999998754306120e+00	1.000007149422537e+00

Appendix C

Tables of Modified Gaussian Quadrature rules

10 Point Gauss-Legendre Rule for integrals of the form $\int_{-1}^{1} f(x)\,dx$

NODES	WEIGHTS
-9.739065285171716e-01	6.667134430868814e-02
-8.650633666889845e-01	1.494513491505806e-01
-6.794095682990244e-01	2.190863625159820e-01
-4.333953941292472e-01	2.692667193099963e-01
-1.488743389816312e-01	2.955242247147529e-01
1.488743389816312e-01	2.955242247147529e-01
4.333953941292472e-01	2.692667193099963e-01
6.794095682990244e-01	2.190863625159820e-01
8.650633666889845e-01	1.494513491505806e-01
9.739065285171716e-01	6.667134430868814e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x)\log|x_1 - x|\,dx$, where x_1 is a Gauss-Legendre node

NODES	WEIGHTS
-9.981629455677877e-01	4.550772157144354e-03
-9.915520723139890e-01	8.062764683328619e-03
-9.832812993252168e-01	7.845621096866406e-03
-9.767801773920733e-01	4.375212351185101e-03
-9.717169387169078e-01	1.021414662954223e-02
-9.510630103726074e-01	3.157199356768625e-02
-9.075765988474132e-01	5.592493151946541e-02
-8.382582352569804e-01	8.310260847601852e-02
-7.408522006801963e-01	1.118164522164500e-01
-6.147619568252419e-01	1.401105427713687e-01
-4.615244999958006e-01	1.657233639623953e-01
-2.849772954295424e-01	1.863566566231937e-01
-9.117593460489747e-02	1.999093145144455e-01
1.119089520342051e-01	2.046841584582030e-01
3.148842536644393e-01	1.995580161940930e-01
5.075733846631832e-01	1.841025430283230e-01
6.797470718157004e-01	1.586456191174843e-01
8.218833662202629e-01	1.242680229936124e-01
9.258924858821892e-01	8.273794370795576e-02
9.857595961761246e-01	3.643931593123844e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_2 - x| \, dx$, where x_2 is a Gauss-Legendre node

NODES	WEIGHTS
-9.954896691005256e-01	1.141744473788874e-02
-9.775532683688947e-01	2.368593568061651e-02
-9.500346715183706e-01	3.027205199814611e-02
-9.192373372373420e-01	3.021809354380292e-02
-8.916563772395616e-01	2.397183723558556e-02
-8.727728136507039e-01	1.253574079839078e-02
-8.607963163061316e-01	2.070840476545303e-02
-8.201318720954396e-01	6.080709508468810e-02
-7.394732321355052e-01	1.002402801599464e-01
-6.204853512352519e-01	1.371499151597280e-01
-4.667290485167077e-01	1.693838059093582e-01
-2.840823320902124e-01	1.945292086962893e-01
-8.079364608026202e-02	2.103223087093422e-01
1.328455136645940e-01	2.149900928447852e-01
3.451233500669768e-01	2.074984762344433e-01
5.437321547508867e-01	1.877085225595498e-01
7.167077216635750e-01	1.564543949958065e-01
8.534299232009863e-01	1.156104890379952e-01
9.458275339169444e-01	6.859369195724087e-02
9.912353127269481e-01	2.390220989094312e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_3 - x| \, dx$, where x_3 is a Gauss-Legendre node

NODES	WEIGHTS
-9.930122613589740e-01	1.779185041193254e-02
-9.643941806993207e-01	3.870503119897836e-02
-9.175869559770760e-01	5.371120494602663e-02
-8.596474181980754e-01	6.073467932536858e-02
-7.990442708271941e-01	5.901993373645797e-02
-7.443700671611690e-01	4.905519963921684e-02
-7.031684479828371e-01	3.249237036645046e-02
-6.811221147275545e-01	1.335394660596527e-02
-6.579449960254029e-01	4.151626407911676e-02
-5.949471688137100e-01	8.451456165895121e-02
-4.893032793226841e-01	1.262522607368499e-01
-3.441659232382107e-01	1.628408264966550e-01
-1.665388322404095e-01	1.907085686614375e-01
3.344207582228461e-02	2.071802230953481e-01
2.434356263087524e-01	2.105274833603497e-01
4.498696863725133e-01	2.000282912446872e-01
6.389777518528792e-01	1.760212445284564e-01
7.978632877793501e-01	1.399000904426490e-01
9.155180703268415e-01	9.402669072995991e-02
9.837258757826489e-01	4.161927873514264e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_4 - x| \, dx$, where x_4 is a Gauss-Legendre node

NODES	WEIGHTS
-9.903478871133073e-01	2.462513260640712e-02
-9.504025146897784e-01	5.449201732062665e-02
-8.834986023815121e-01	7.799498604905293e-02
-7.974523551287549e-01	9.241688894090601e-02
-7.022255002503461e-01	9.619882322938848e-02
-6.087194789244920e-01	8.902783806614303e-02
-5.275278952351541e-01	7.181973054766198e-02
-4.677586540799037e-01	4.663017060126023e-02
-4.360689210457623e-01	1.794303974050253e-02
-4.121945474875853e-01	4.061799823415495e-02
-3.494226766911471e-01	8.507517518447759e-02
-2.425993523586304e-01	1.277525783357134e-01
-9.646839923908594e-02	1.628510773009247e-01
7.921243716767302e-02	1.863323765408308e-01
2.715178194484646e-01	1.958227701927855e-01
4.658440358656903e-01	1.903138548150517e-01
6.472213975763533e-01	1.700731513381802e-01
8.015601619414859e-01	1.365784674773513e-01
9.168056007307982e-01	9.239595239693155e-02
9.839468743284722e-01	4.103797108164931e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_5 - x| \, dx$, where x_5 is a Gauss-Legendre node

NODES	WEIGHTS
-9.883561797860961e-01	2.974603958509255e-02
-9.398305159297058e-01	6.657945456889164e-02
-8.572399919019390e-01	9.731775484182564e-02
-7.482086250804679e-01	1.190433988432928e-01
-6.228514167093102e-01	1.297088242013777e-01
-4.928317114329241e-01	1.282900896966494e-01
-3.702771193724617e-01	1.148917968875341e-01
-2.666412108172461e-01	9.074932908233864e-02
-1.916083010783277e-01	5.818196361216740e-02
-1.521937160593461e-01	2.224697059733435e-02
-1.233125650067164e-01	4.788826761346366e-02
-5.257959675044444e-02	9.237500180593534e-02
5.877314311857769e-02	1.287410543031414e-01
2.012559739993003e-01	1.541960911507042e-01
3.627988191760868e-01	1.665885274544506e-01
5.297121321076323e-01	1.648585116745725e-01
6.878399330187783e-01	1.491408089644010e-01
8.237603202215137e-01	1.207592726093190e-01
9.259297297557394e-01	8.212177982524418e-02
9.856881498392895e-01	3.657506268226379e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x)\log|x_6 - x|\,dx$, where x_6 is a Gauss-Legendre node

NODES	WEIGHTS
-9.856881498392895e-01	3.657506268226379e-02
-9.259297297557394e-01	8.212177982524418e-02
-8.237603202215137e-01	1.207592726093190e-01
-6.878399330187783e-01	1.491408089644010e-01
-5.297121321076323e-01	1.648585116745725e-01
-3.627988191760868e-01	1.665885274544506e-01
-2.012559739993003e-01	1.541960911507042e-01
-5.877314311857769e-02	1.287410543031414e-01
5.257959675044444e-02	9.237500180593534e-02
1.233125650067164e-01	4.788826761346366e-02
1.521937160593461e-01	2.224697059733435e-02
1.916083010783277e-01	5.818196361216740e-02
2.666412108172461e-01	9.074932908233864e-02
3.702771193724617e-01	1.148917968875341e-01
4.928317114329241e-01	1.282900896966494e-01
6.228514167093102e-01	1.297088242013777e-01
7.482086250804679e-01	1.190433988432928e-01
8.572399919019390e-01	9.731775484182564e-02
9.398305159297058e-01	6.657945456889164e-02
9.883561797860961e-01	2.974603958509255e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x)\log|x_7 - x|\,dx$, where x_7 is a Gauss-Legendre node

NODES	WEIGHTS
-9.839468743284722e-01	4.103797108164931e-02
-9.168056007307982e-01	9.239595239693155e-02
-8.015601619414859e-01	1.365784674773513e-01
-6.472213975763533e-01	1.700731513381802e-01
-4.658440358656903e-01	1.903138548150517e-01
-2.715178194484646e-01	1.958227701927855e-01
-7.921243716767302e-02	1.863323765408308e-01
9.646839923908594e-02	1.628510773009247e-01
2.425993523586304e-01	1.277525783357134e-01
3.494226766911471e-01	8.507517518447759e-02
4.121945474875853e-01	4.061799823415495e-02
4.360689210457623e-01	1.794303974050253e-02
4.677586540799037e-01	4.663017060126023e-02
5.275278952351541e-01	7.181973054766198e-02
6.087194789244920e-01	8.902783806614303e-02
7.022255002503461e-01	9.619882322938848e-02
7.974523551287549e-01	9.241688894090601e-02
8.834986023815121e-01	7.799498604905293e-02
9.504025146897784e-01	5.449201732062665e-02
9.903478871133073e-01	2.462513260640712e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_8 - x| \, dx$, where x_8 is a Gauss-Legendre node

NODES	WEIGHTS
-9.837258757826489e-01	4.161927873514264e-02
-9.155180703268415e-01	9.402669072995991e-02
-7.978632877793501e-01	1.399000904426490e-01
-6.389777518528792e-01	1.760212445284564e-01
-4.498696863725133e-01	2.000282912446872e-01
-2.434356263087524e-01	2.105274833603497e-01
-3.344207582228461e-02	2.071802230953481e-01
1.665388322404095e-01	1.907085686614375e-01
3.441659232382107e-01	1.628408264966550e-01
4.893032793226841e-01	1.262522607368499e-01
5.949471688137100e-01	8.451456165895121e-02
6.579449960254029e-01	4.151626407911676e-02
6.811221147275545e-01	1.335394660596527e-02
7.031684479828371e-01	3.249237036645046e-02
7.443700671611690e-01	4.905519963921684e-02
7.990442708271941e-01	5.901993373645797e-02
8.596474181980754e-01	6.073467932536858e-02
9.175869559770760e-01	5.371120494602663e-02
9.643941806993207e-01	3.870503119897836e-02
9.930122613589740e-01	1.779185041193254e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x) \log |x_9 - x| \, dx$, where x_9 is a Gauss-Legendre node

NODES	WEIGHTS
-9.912353127269481e-01	2.390220989094312e-02
-9.458275339169444e-01	6.859369195724087e-02
-8.534299232009863e-01	1.156104890379952e-01
-7.167077216635750e-01	1.564543949958065e-01
-5.437321547508867e-01	1.877085225595498e-01
-3.451233500669768e-01	2.074984762344433e-01
-1.328455136645940e-01	2.149900928447852e-01
8.079364608026202e-02	2.103223087093422e-01
2.840823320902124e-01	1.945292086962893e-01
4.667290485167077e-01	1.693838059093582e-01
6.204853512352519e-01	1.371499151597280e-01
7.394732321355052e-01	1.002402801599464e-01
8.201318720954396e-01	6.080709508468810e-02
8.607963163061316e-01	2.070840476545303e-02
8.727728136507039e-01	1.253574079839078e-02
8.916563772395616e-01	2.397183723558556e-02
9.192373372373420e-01	3.021809354380292e-02
9.500346715183706e-01	3.027205199814611e-02
9.775532683688947e-01	2.368593568061651e-02
9.954896691005256e-01	1.141744473788874e-02

20 point quadrature rule for integrals of the form $\int_{-1}^{1} f(x) + g(x)\log\lvert x_{10} - x\rvert\,dx$, where x_{10} is a Gauss-Legendre node	
NODES	WEIGHTS
-9.857595961761246e-01	3.643931593123844e-02
-9.258924858821892e-01	8.273794370795576e-02
-8.218833662202629e-01	1.242680229936124e-01
-6.797470718157004e-01	1.586456191174843e-01
-5.075733846631832e-01	1.841025430283230e-01
-3.148842536644393e-01	1.995580161940930e-01
-1.119089520342051e-01	2.046841584582030e-01
9.117593460489747e-02	1.999093145144455e-01
2.849772954295424e-01	1.863566566231937e-01
4.615244999958006e-01	1.657233639623953e-01
6.147619568252419e-01	1.401105427713687e-01
7.408522006801963e-01	1.118164522164500e-01
8.382582352569804e-01	8.310260847601852e-02
9.075765988474132e-01	5.592493151946541e-02
9.510630103726074e-01	3.157199356768625e-02
9.717169387169078e-01	1.021414662954223e-02
9.767801773920733e-01	4.375212351185101e-03
9.832812993252168e-01	7.845621096866406e-03
9.915520723139890e-01	8.062764683328619e-03
9.981629455677877e-01	4.550772157144354e-03

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x + \bar{x})dx$, where $\bar{x} \geq 10^{-1}$	
NODES	WEIGHTS
3.916216329415252e-02	4.880755296918116e-02
8.135233983530081e-02	3.196002785163611e-02
1.123448211344994e-01	3.883416642507362e-02
1.595931983965030e-01	5.148898992140820e-02
2.085759027831349e-01	4.219328148763533e-02
2.426241962027560e-01	3.420686213633789e-02
2.886190312538522e-01	5.512488680719239e-02
3.469021762354675e-01	6.007112809843418e-02
4.072910101569611e-01	6.022350479415180e-02
4.664019722595442e-01	5.735022004401478e-02
5.182120817844112e-01	4.167923417118068e-02
5.501308436771654e-01	3.346089628879600e-02
5.970302980854608e-01	5.574716218423796e-02
6.548457960388209e-01	5.847838243344473e-02
7.119542126106005e-01	5.464156990092474e-02
7.607920420946340e-01	4.092186343704961e-02
7.953017051155684e-01	3.283728166050225e-02
8.303900341517088e-01	3.438233273473095e-02
8.612724919009394e-01	3.022585192226418e-02
8.954049128027080e-01	3.700769701277380e-02
9.315909369155358e-01	3.410213679365162e-02
9.621742249068356e-01	2.665791885274193e-02
9.843663446380599e-01	1.754420526360429e-02
9.970087425823398e-01	7.662283104388867e-03

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-2} \leq \bar{x} \leq 10^{-1}$	
NODES	WEIGHTS
1.940564616937581e-02	2.514022176052795e-02
4.545433992382339e-02	2.703526530535647e-02
7.378866604396420e-02	2.980872487617485e-02
1.054147718077606e-01	3.360626237885489e-02
1.412997888401000e-01	3.829678083416609e-02
1.822325567811081e-01	4.365651045780837e-02
2.287282121202408e-01	4.935846322319046e-02
2.809170925514041e-01	5.495967924055210e-02
3.384320962237970e-01	5.991162198705084e-02
4.003108031244078e-01	6.356960862248889e-02
4.648605571606025e-01	6.506868552467118e-02
5.290714994276687e-01	6.219588235225894e-02
5.829663557386375e-01	3.889986041695310e-02
6.128301889979477e-01	3.573431931940621e-02
6.606072156240962e-01	5.296315368353523e-02
7.139495966128518e-01	5.369033999927759e-02
7.677830914961244e-01	5.340793573367282e-02
8.187382423336450e-01	4.704756013998560e-02
8.587068551739496e-01	3.276576301747068e-02
8.906873285570645e-01	3.449175311880027e-02
9.267772492129903e-01	3.560168848238671e-02
9.592137652582382e-01	2.857367151127661e-02
9.830962712794008e-01	1.894042942442201e-02
9.967621546194148e-01	8.291994770212826e-03

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-3} \leq \bar{x} \leq 10^{-2}$	
NODES	WEIGHTS
7.571097817272427e-03	9.878088201321919e-03
1.800655325976786e-02	1.109316819462674e-02
3.003901004577040e-02	1.313311581321880e-02
4.462882147989575e-02	1.624262442061470e-02
6.295732618092606e-02	2.065168462990214e-02
8.644035241970913e-02	2.657795406825320e-02
1.166164809306920e-01	3.399052299072427e-02
1.546690628394902e-01	4.208214612865170e-02
1.999554346680615e-01	4.732516974042797e-02
2.434683359132119e-01	3.618419415803922e-02
2.800846274146029e-01	4.547346840583578e-02
3.368595257878888e-01	6.463153575242817e-02
4.044418359833648e-01	6.859104457897808e-02
4.685002493634456e-01	5.589917935916451e-02
5.185062817085154e-01	5.199232318335285e-02
5.811314144990846e-01	7.089840644422261e-02
6.545700991450585e-01	7.427400331494240e-02
7.276588861478224e-01	7.125308736931726e-02
7.960626077582168e-01	6.513697474660338e-02
8.572037183403355e-01	5.682298546820264e-02
9.091330485015775e-01	4.678000924507099e-02
9.503131649503738e-01	3.538488886617123e-02
9.795718963793163e-01	2.299723483013955e-02
9.961006479199827e-01	9.993597414733579e-03

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-4} \le \bar{x} \le 10^{-3}$

NODES	WEIGHTS
2.625961371586153e-03	3.441901737135120e-03
6.309383772392260e-03	3.978799794732070e-03
1.073246133489697e-02	4.958449505644980e-03
1.645170499644402e-02	6.620822501994994e-03
2.433800511777796e-02	9.385496468197222e-03
3.582530925992294e-02	1.396512052439178e-02
5.315827372101662e-02	2.119383832447796e-02
7.917327903614484e-02	3.124989308824302e-02
1.162053707416708e-01	4.291481168916344e-02
1.648139164451449e-01	5.400832278279924e-02
2.231934088488800e-01	6.197424674301215e-02
2.864519293820641e-01	6.297221626131570e-02
3.466729491189400e-01	5.794981636764223e-02
4.076175535528108e-01	6.650501614478806e-02
4.800964107543535e-01	7.716379373230733e-02
5.594105009204460e-01	8.047814122759604e-02
6.395390292352857e-01	7.917822434973971e-02
7.167410782176877e-01	7.477646096014055e-02
7.882807127957939e-01	6.793424765652059e-02
8.519356675821297e-01	5.906852968947303e-02
9.058606177202579e-01	4.853108558910315e-02
9.485539755760567e-01	3.666228059710319e-02
9.788566874094059e-01	2.380850649522536e-02
9.959649506960162e-01	1.034186239262945e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-5} \le \bar{x} \le 10^{-4}$

NODES	WEIGHTS
7.759451679242260e-04	1.049591733965263e-03
1.952854410117286e-03	1.314968855711329e-03
3.429053832116395e-03	1.651475072547296e-03
5.301128540262913e-03	2.135645684467029e-03
7.878118775220067e-03	3.165043382856636e-03
1.205537050949829e-02	5.479528688655274e-03
1.965871512055557e-02	1.028817002915096e-02
3.403328641997047e-02	1.923291785614007e-02
5.947430305925957e-02	3.212643438782854e-02
9.873500543531440e-02	4.638626850049229e-02
1.518862681939413e-01	5.960676923068444e-02
2.171724325134259e-01	7.052360405410943e-02
2.919941878735093e-01	7.863451090237836e-02
3.734637353255530e-01	8.381771698595157e-02
4.586710018443288e-01	8.612755554083525e-02
5.448057416999684e-01	8.569938467103264e-02
6.292158981939618e-01	8.271051499695768e-02
7.094415843889587e-01	7.736692567834522e-02
7.832417328632321e-01	6.990012937760461e-02
8.486194141302759e-01	6.056687669667680e-02
9.038469149367938e-01	4.964868706783169e-02
9.474898150194623e-01	3.745026957972177e-02
9.784290662963747e-01	2.429741981889855e-02
9.958843370550371e-01	1.054906616108520e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-6} \leq \bar{x} \leq 10^{-5}$

NODES	WEIGHTS
3.126377187332637e-04	4.136479682893960e-04
7.671264269072188e-04	5.068714387414649e-04
1.359575160544077e-03	7.008932527842778e-04
2.238313285727558e-03	1.110264922990352e-03
3.770276623583326e-03	2.120108385941761e-03
7.146583956092048e-03	5.249076343206215e-03
1.635515250548719e-02	1.450809938905405e-02
3.828062855101241e-02	2.987724029376343e-02
7.628984500206759e-02	4.593298717863718e-02
1.294255336121595e-01	5.987634475538021e-02
1.949876755761554e-01	7.065953519392547e-02
2.693852297828856e-01	7.729918562776261e-02
3.469762441631538e-01	7.556635340171830e-02
4.122748928895491e-01	5.234123638339037e-02
4.662499202239145e-01	6.532130125393047e-02
5.421402737123784e-01	8.188272080198840e-02
6.248832413655412e-01	8.237354882288161e-02
7.053258496784840e-01	7.795795664563893e-02
7.798841313231049e-01	7.076514272025076e-02
8.461534275163378e-01	6.145788741452406e-02
9.022312524979976e-01	5.044339641339403e-02
9.465899812310277e-01	3.807817118430632e-02
9.780549563823810e-01	2.471549011101626e-02
9.958125149101927e-01	1.073289672726758e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-7} \leq \bar{x} \leq 10^{-6}$

NODES	WEIGHTS
1.019234906342863e-04	1.349775051746596e-04
2.506087227631447e-04	1.663411550150506e-04
4.461429005344285e-04	2.328782111562424e-04
7.422845421202523e-04	3.804721779784063e-04
1.289196091156456e-03	7.930350452911450e-04
2.739287668024851e-03	2.600694722423854e-03
9.075168969969708e-03	1.212249113599252e-02
2.968005234555358e-02	2.946708975720586e-02
6.781742979962609e-02	4.647771960691390e-02
1.217792474402805e-01	6.095376889009233e-02
1.886625378438471e-01	7.224844725827559e-02
2.650602155844836e-01	7.986429603884565e-02
3.465113608339080e-01	8.143206462900546e-02
4.178374197420536e-01	5.040529357007135e-02
4.597624982511183e-01	5.592137651001418e-02
5.348065111487157e-01	8.398073572656715e-02
6.194640153146728e-01	8.402586870225486e-02
7.013481004172354e-01	7.922223490159952e-02
7.770386175609082e-01	7.177919251691964e-02
8.442211768916794e-01	6.227551999401272e-02
9.010272836291835e-01	5.108407212719758e-02
9.459409782755001e-01	3.854783279333592e-02
9.777905486554876e-01	2.501496650831813e-02
9.957622871041650e-01	1.086176801402067e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-8} \le \bar{x} \le 10^{-7}$	
NODES	WEIGHTS
3.421721832247593e-05	4.559730842497453e-05
8.533906255442380e-05	5.840391255974745e-05
1.563524616155011e-04	8.761580900682040e-05
2.746612401575526e-04	1.617264666294872e-04
5.408643931265062e-04	4.433543035169213e-04
1.782382096488333e-03	3.116175111368442e-03
1.101243912052365e-02	1.655494413772595e-02
3.553172024884285e-02	3.242539256461602e-02
7.554170435463801e-02	4.734426463929677e-02
1.295711894941649e-01	6.032614603579952e-02
1.953213037793089e-01	7.069975187373848e-02
2.699680545714222e-01	7.806973621204365e-02
3.503697281371090e-01	8.216350598137868e-02
4.330838596494367e-01	8.261286657092808e-02
5.141801680435878e-01	7.883476216668445e-02
5.895097016206093e-01	7.157205125318401e-02
6.582708672338614e-01	6.703064468754417e-02
7.252543617887320e-01	6.706137273719630e-02
7.914154485613720e-01	6.449984116349734e-02
8.528383935857844e-01	5.775434959088197e-02
9.059696536862878e-01	4.812600239023880e-02
9.484664124578303e-01	3.661415869304224e-02
9.787863313133854e-01	2.386304203446463e-02
9.959482975155097e-01	1.038268695581411e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-9} \le \bar{x} \le 10^{-8}$	
NODES	WEIGHTS
6.538987938840374e-06	1.500332421093607e-05
2.613485075847413e-05	2.367234654253158e-05
5.664183720634991e-05	4.007286246706405e-05
1.179374114362569e-04	9.497743501485505e-05
3.299119431334128e-04	4.619067037944727e-04
3.626828607577001e-03	9.985382463808036e-03
2.265102906572155e-02	2.805741744607257e-02
5.896796231680340e-02	4.404106103008398e-02
1.092496277855923e-01	5.548413172821072e-02
1.666701689499393e-01	5.693235996372726e-02
2.196889385898800e-01	5.087307376046002e-02
2.770352260035617e-01	6.593729718379782e-02
3.483163928268329e-01	7.335680008972614e-02
4.153287664837260e-01	5.675029500743735e-02
4.695624219668608e-01	6.117926027541254e-02
5.421129318998841e-01	8.004805067067550e-02
6.238832212055707e-01	8.196991767042605e-02
7.041842972237081e-01	7.800219127200407e-02
7.788817007552110e-01	7.097175077519494e-02
8.453877637047045e-01	6.171193295041172e-02
9.017178251963006e-01	5.068671319716005e-02
9.462999385952402e-01	3.827738423897266e-02
9.779333485180249e-01	2.485063762733620e-02
9.957890687155009e-01	1.079284973329516e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x + \bar{x})dx$, where $10^{-10} \leq \bar{x} \leq 10^{-9}$

NODES	WEIGHTS
6.725520559705825e-06	8.128391913974039e-05
6.986424152770461e-06	-7.773900735768282e-05
1.217363416714366e-05	1.287386499666193e-05
2.677746219601529e-05	1.895577251914526e-05
5.597036348896741e-05	4.732580352158076e-05
2.729343280943077e-04	9.857909615386162e-04
9.445526806263141e-03	1.756872897270054e-02
3.556725025161542e-02	3.439422017906772e-02
7.765556668177810e-02	4.944188361792970e-02
1.336848150648662e-01	6.219733934997792e-02
2.011576917683550e-01	7.228007436918939e-02
2.772736854314979e-01	7.944986391225688e-02
3.590124362607926e-01	8.347646288178011e-02
4.430074035214462e-01	8.380433020121207e-02
5.247388219574510e-01	7.832768209682506e-02
5.961053238782420e-01	6.300796225242940e-02
6.547331131213409e-01	5.923406014585053e-02
7.192258519628951e-01	6.834293563803810e-02
7.874251789073102e-01	6.660337204499726e-02
8.505852012775045e-01	5.911988751082552e-02
9.047824617894323e-01	4.893575310568894e-02
9.479045131744448e-01	3.708256438629509e-02
9.785770588866582e-01	2.411463784693618e-02
9.959104692340199e-01	1.048087156697020e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x + \bar{x})dx$, where $10^{-11} \leq \bar{x} \leq 10^{-10}$

NODES	WEIGHTS
2.828736694877886e-08	1.665602686704325e-05
2.302233157554212e-06	2.577419924039251e-06
5.853587143444178e-06	4.957941112780975e-06
1.451588770083244e-05	1.537074702915107e-05
9.711965099273031e-05	4.640075239797995e-04
9.004761967373848e-03	1.705687938176189e-02
3.442077924035546e-02	3.349724914160473e-02
7.543926781582543e-02	4.820210872119093e-02
1.300373356318913e-01	6.054547286337976e-02
1.955182772803384e-01	6.984354388121057e-02
2.683608546664295e-01	7.498721497014774e-02
3.430029178740901e-01	7.240620145057083e-02
4.085056107803621e-01	5.774925310174693e-02
4.660198270439085e-01	6.238505554837956e-02
5.336124745634699e-01	6.940394677081842e-02
5.985245800106473e-01	5.910843483407385e-02
6.564089719608276e-01	6.059752321454190e-02
7.216666024232565e-01	6.823362237770209e-02
7.893712241343741e-01	6.593839664071163e-02
8.518883782001418e-01	5.853014420243146e-02
9.055688088881344e-01	4.849217100974983e-02
9.483163097840529e-01	3.677417821170115e-02
9.787413692715607e-01	2.392585642844202e-02
9.959413203611228e-01	1.040149939671874e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-12} \le \bar{x} \le 10^{-11}$

NODES	WEIGHTS
6.147063879573664e-07	8.763741095000331e-07
2.102921984985835e-06	1.784696796288373e-05
2.188366117432289e-06	-1.795398395983826e-05
3.482602942694880e-06	5.117514567175025e-06
2.768001888608636e-05	1.698863549284390e-04
8.942779215792784e-03	1.701975216672032e-02
3.432218364237253e-02	3.346025972593909e-02
7.530931328026620e-02	4.817949622196712e-02
1.298983048592572e-01	6.055152664710045e-02
1.954020797117703e-01	6.988313730886592e-02
2.682970870436427e-01	7.504602275463067e-02
3.429540704041702e-01	7.230942674874111e-02
4.080399755202422e-01	5.705952259766429e-02
4.652562798154792e-01	6.265021180818162e-02
5.333220999210325e-01	6.993669694523695e-02
5.986982369433125e-01	5.937130986945129e-02
6.564773600603511e-01	6.026572020863567e-02
7.215159032030418e-01	6.815292696374753e-02
7.892098210760941e-01	6.596804590657802e-02
8.517672777806986e-01	5.857483758149194e-02
9.054906995605498e-01	4.853209199396977e-02
9.482736017320823e-01	3.680469214176019e-02
9.787238593479314e-01	2.394561701705853e-02
9.959379852805677e-01	1.041005152890511e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-13} \le \bar{x} \le 10^{-12}$

NODES	WEIGHTS
4.523740015216508e-08	4.418138082366788e-07
4.281855233588279e-07	4.389108058643120e-07
1.036900153156159e-06	9.539585150737866e-07
7.825849325746907e-06	5.823980947200484e-05
8.617419723953112e-03	1.634464263521301e-02
3.268881163637599e-02	3.129682188728318e-02
6.988441391437043e-02	4.212468617589480e-02
1.142202307676442e-01	4.505120897719191e-02
1.596471081833281e-01	4.769069780026684e-02
2.135336418959620e-01	6.038503382768951e-02
2.781100275296151e-01	6.695343672694180e-02
3.433392803364457e-01	6.163298712826237e-02
4.019960595528027e-01	5.877742624357513e-02
4.656415679416787e-01	6.800053637773440e-02
5.334880548894250e-01	6.516918103589647e-02
5.943298528903542e-01	5.853785375926075e-02
6.562968737815924e-01	6.639396325654251e-02
7.250343344601498e-01	6.948738324081696e-02
7.928820737781136e-01	6.538801703374268e-02
8.546103048745466e-01	5.761503751629250e-02
9.073762310762705e-01	4.761344859555310e-02
9.493253659835347e-01	3.607033097268266e-02
9.791606801267259e-01	2.345690720840071e-02
9.960217573957566e-01	1.019557402722854e-02

24 point quadrature rule for integrals of the form $\int_0^1 f(x) + g(x)\log(x+\bar{x})dx$, where $10^{-14} \leq \bar{x} \leq 10^{-13}$

NODES	WEIGHTS
6.025980282801020e-08	9.079353616441234e-07
6.411245262925473e-08	-8.390389042773805e-07
1.862815529429129e-07	2.782460677485016e-07
2.029190208906422e-06	1.821115881362725e-05
8.902881307076499e-03	1.695809650660321e-02
3.420089035164912e-02	3.336370146025145e-02
7.508687525931594e-02	4.807898681796971e-02
1.295858123029775e-01	6.047672723211479e-02
1.950409815188335e-01	6.986774906175534e-02
2.679751967812604e-01	7.515608233194288e-02
3.428525062164689e-01	7.264249904037610e-02
4.080941369413548e-01	5.672507168477261e-02
4.646644511900009e-01	6.220316364524964e-02
5.328071517215501e-01	7.032362652293805e-02
5.978508749698001e-01	5.742730804758014e-02
6.521214523350964e-01	5.644075454541152e-02
7.134921670665336e-01	6.318643666150391e-02
7.679317896479284e-01	3.945995610428228e-02
8.029718487208403e-01	4.324200884758527e-02
8.551101435866935e-01	5.478223695609097e-02
9.067319102017767e-01	4.740856250832772e-02
9.487765213293372e-01	3.633314063504751e-02
9.788979796532736e-01	2.372788917088821e-02
9.959684838634199e-01	1.033036588606145e-02