

RICE UNIVERSITY
Numerical methods for boundary integral equations

By

Yabin Zhang

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE



Adrianna Gillman

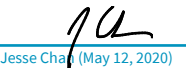
Assistant Professor of Computational and
Applied Mathematics

Beatrice Riviere

Beatrice Riviere (May 13, 2020)

Beatrice Riviere

Noah Harding Chair and Professor of
Computational and Applied Mathematics



Jesse Chan

Assistant Professor of Computational and
Applied Mathematics

Ilinca Stanciulescu

Ilinca Stanciulescu (May 13, 2020)

Ilinca Stanciulescu

Associate Professor of Civil Engineering

HOUSTON, TEXAS

May 2020

ABSTRACT

Numerical methods for boundary integral equations

by

Yabin Zhang

This thesis focuses on numerical methods for boundary integral equation (BIE) formulations of partial differential equations (PDEs). The work contains three parts: the first two consider numerical solution methods for boundary integral equations in wave scattering and Stokes flow, respectively. The last part proposes an adaptive discretization technique for BIEs in 2D.

The proposed work is based on previous developments in fast direct solution techniques for BIEs. Such methods exploit the rank deficiency in the off-diagonal blocks of the discretized system and build an approximation to the inverse with linear cost with respect to the number of unknowns on the domain boundary for two-dimensional problems. Once the approximation of the inverse is constructed, applying it to any given vector is very cheap, making the methods ideal for problems with lots of right-hand-sides. The two direct solvers presented in this thesis are driven by applications. The scattering solver is built to assist practitioners in designing acoustic and optic devices to manipulate waves. Its efficiency in handling multiple incident angles and minor modifications in the scatterer's structure will be handy in an optimal design setting. The Stokes solver helps practitioners to simulate objects such as bacteria and vesicles in viscous flow. To accurately capture the interaction between the objects

and the confining wall, the discretization of the wall often needs to be locally refined in the region approached by the objects. This makes standard fast direct solvers too expensive to be useful, as the linear system changes for each time step. The proposed approach avoids this by pre-constructing a fast direct solver for the wall independently of time and updating the original solver to accommodate any refinements in discretization.

The last part of the thesis presents an adaptive discretization technique for two-dimensional BIEs. Standard adaptive discretization method often requires a sequence of global boundary density solves each on a finer mesh and terminates with the final mesh which resolves the boundary density to the user prescribed tolerance. The global density solves make the cost of the standard approach relatively high. The proposed alternative reduces the cost by replacing global solves with local solves for an approximate of the true density.

Acknowledgements

I would like to express my gratitude to people who supported and helped me completing this piece of work as well as my graduate studies.

First of all, I would like to thank my PhD advisor, Adrianna Gillman, for her patient and insightful guidance in the last five years and her great support for my pursuing an academic career. I would also like to thank my other committee members, Jesse Chan, Beatrice Riviere, and Ilinca Stanciulescu for their support and valuable feedback on the presented thesis work. Preparing and defending a thesis in this pandemic was a challenge, but their quick replies and understanding made this much easier.

I would also want to give special thanks to Alex Barnett and Manas Rachh for the fruitful conversations on the research projects and Shravan Veerapaneni, Hanliang Guo and Hai Zhu for the collaboration on the Stokes project.

Additionally, I would like to thank the administration staff at CAAM and the applied mathematics department at CU Boulder who helped me with my 6-month stay at Boulder as a visiting graduate student.

Finally, I would like to give my personal thanks to my friends and family members for their love and support.

Contents

| | |
|--|-----------|
| Abstract | ii |
| List of Illustrations | ix |
| List of Tables | xv |
| 1 Introduction | 1 |
| 1.1 Thesis outline | 2 |
| 1.2 BIE formulation for elliptic BVPs | 2 |
| 1.3 Discretization and fast solvers for BIE | 4 |
| 1.4 Use BIE to simulate applications | 10 |
| 1.4.1 Quasi-periodic scattering in layered media | 11 |
| 1.4.2 Stokes flow in confined geometry | 12 |
| 1.5 An adaptive discretization strategy | 13 |
| 2 Tools and solvers in literature | 14 |
| 2.1 Interpolative decomposition (ID) for rank-deficient matrices | 14 |
| 2.2 Far-field and near-field separation for boundary integral operator | 16 |
| 2.3 Hierarchically block-separable (HBS) solver for BIEs | 19 |
| 2.3.1 Block-separable matrix | 19 |
| 2.3.2 Hierarchically block-separable matrix | 22 |
| 2.3.3 HBS representation for the coefficient matrix | 24 |
| 2.4 A fast direct solver for locally-perturbed geometry | 25 |
| 2.4.1 Locally perturbed geometry and extended linear system | 26 |

| | | |
|-------|--------------------------------|----|
| 2.4.2 | A fast direct solver | 28 |
|-------|--------------------------------|----|

3 A fast direct solver for quasi-periodic scattering in layered media **30**

| | | |
|-------|---|----|
| 3.1 | Introduction | 31 |
| 3.1.1 | Related work | 33 |
| 3.1.2 | High level view of the solution technique | 34 |
| 3.1.3 | Outline | 36 |
| 3.2 | Periodizing scheme | 36 |
| 3.2.1 | Integral operators | 37 |
| 3.2.2 | Integral formulation | 41 |
| 3.2.3 | The linear system | 44 |
| 3.3 | The fast direct solver | 49 |
| 3.3.1 | Fast inversion of \mathbf{A} | 51 |
| 3.3.2 | Low rank factorization of $\hat{\mathbf{A}}$ | 53 |
| 3.3.3 | The Bloch phase and incident angle dependence | 59 |
| 3.3.4 | Extensions | 60 |
| 3.4 | Numerical examples | 61 |
| 3.4.1 | Scaling experiment | 63 |
| 3.4.2 | Sweep over multiple incident angles | 64 |
| 3.4.3 | Local change to the geometry | 67 |
| 3.5 | Summary | 68 |

4 A fast direct solver for modeling Stokes flow in confined geometry **72**

| | | |
|-----|---|----|
| 4.1 | Boundary integral formulation | 74 |
|-----|---|----|

| | | |
|-------|---|----|
| 4.1.1 | The steady-state problem | 75 |
| 4.1.2 | Time-dependent problem formulation | 79 |
| 4.1.3 | Numerical simulation and local discretization refinements | 80 |
| 4.2 | A fast direct solver for BIEs with locally refined discretization | 81 |
| 4.2.1 | An alternative extended system formulation | 82 |
| 4.2.2 | Low-rank approximation for the update matrix | 84 |
| 4.2.3 | Solution evaluation | 86 |
| 4.3 | Numerical experiments | 88 |
| 4.3.1 | Scaling test | 90 |
| 4.3.2 | Complex geometry test | 92 |
| 4.3.3 | Other tests | 92 |
| 4.4 | Summary | 94 |

5 An adaptive discretization technique for BIEs on the plane **95**

| | | |
|-------|---|-----|
| 5.1 | Introduction | 96 |
| 5.1.1 | Existing adaptive discretization strategies in literature | 97 |
| 5.2 | An adaptive scheme based on the true boundary density | 103 |
| 5.2.1 | Initialization | 103 |
| 5.2.2 | Refining and coarsening condition | 104 |
| 5.2.3 | Convergence error and exit condition | 106 |
| 5.2.4 | Full Algorithm and cost analysis | 108 |
| 5.3 | A new adaptive scheme | 110 |
| 5.3.1 | Locally updated artificial density | 110 |
| 5.3.2 | Exit condition | 111 |
| 5.3.3 | Full algorithm and efficient implementation | 112 |

| | | |
|----------|---|------------|
| 5.4 | Numerical results | 116 |
| 5.4.1 | Comparison with the true density guided algorithm | 117 |
| 5.4.2 | Same geometry with different right-hand-side functions | 118 |
| 5.4.3 | Complex geometry | 120 |
| 5.5 | Summary | 123 |
| 6 | Conclusions | 124 |
| | Bibliography | 126 |
| A | Appendix | 142 |
| A.1 | Efficient construction of \mathbf{S}_2 | 142 |
| A.2 | Definitions for the Stokes boundary integral operators | 144 |
| A.3 | Numerical tests for Laplace and Helmholtz BVPs on locally perturbed geometries | 145 |
| A.3.1 | A local change in the geometry | 147 |
| A.3.2 | A Laplace problem with a locally refined discretization | 148 |
| A.3.3 | A Helmholtz problem with a locally refined discretization | 149 |
| A.4 | Definition of matrix M_n | 153 |

Illustrations

| | |
|--|----|
| 1.2.1 Model geometry for the Laplace boundary value problem example. . . | 3 |
| 1.3.1 Example of a geometrically ill-conditioned problem. | 9 |
| 2.2.1 An illustration of near-field and far-field separation. (a) The boundary geometry with Γ_b in bold line. (b) The proxy circle (dotted blue line) that separates $\Gamma \setminus \Gamma_b$ into the far-field $\Gamma_{far(b)}$ and the near-field $\Gamma_{near(b)}$ with respect to b | 17 |
| 2.3.1 Illustration of a block-separable factorization as in (2.3.3) . Matrix \mathbf{A} is a block-separable matrix with $4 \times 4 = 16$ sub-blocks. \mathbf{U} , \mathbf{V} , and \mathbf{D} are all block-diagonal matrices. Matrix $\hat{\mathbf{A}}$ has zero blocks on the diagonal. | 21 |
| 2.3.2 Illustration of the inverse factorization of a block separable matrix \mathbf{A} (2.3.5) . Matrix \mathbf{A} is a block-separable matrix with $4 \times 4 = 16$ sub-blocks and can be factored as (2.3.3). \mathbf{E} , $\tilde{\mathbf{D}}$, \mathbf{F} , and \mathbf{G} are all block-diagonal matrices. Matrix $(\tilde{\mathbf{A}} + \hat{\mathbf{D}})^{-1}$ is the only dense matrix remained. | 22 |
| 2.3.3 Numbering of box nodes in a fully populated binary tree with $L = 3$ levels. The root is the original index vector $I = I_1 = [1, 2, \dots, 400]$ of the discretization points in the parameter space of Γ | 23 |

| | | |
|-------|--|----|
| 2.3.4 | Illustration of the block structure of the telescoping factors for a 3-level HBS representation. The telescoping factorization is given in (2.3.9). | 24 |
| 2.4.1 | A sample locally perturbed geometry where the original boundary is $\Gamma_o = \Gamma_k \cup \Gamma_c$, the portion of the boundary being removed is Γ_c , the portion of the original boundary remaining is Γ_k and the newly added boundary piece is Γ_p | 26 |
| 2.4.2 | (a) The star geometry with the portion of the boundary to be refined boxed. (b) The three Gaussian panels in the boxed region from the original discretization. (c) The six Gaussian panels that replaced the original three panels. | 29 |
| 3.1.1 | A five layered periodic geometry. 7 periods are shown. | 32 |
| 3.2.1 | This figure illustrates a five layered periodic geometry with artificial walls and proxy circles. Only three periods of the infinite periodic geometry are shown. The period contained within the unit cell is in black while the other two periods are in blue. Figure (a) illustrates the notation for the unit cell with left, right, upper, and lower boundary L , R , U , and D shown in red lines. Figure (b) illustrates the proxy circles P_i for each layer. The color of the proxy circles alternates between green and magenta. | 38 |
| 3.3.1 | Illustration of the dyadic refinement partitioning of Γ_i with 5 levels of refinement and geometries for compressing \mathbf{A}_{ii}^m . (a) Illustration of the proxy surface (dashed circle) used to compress neighbor interactions when γ_l is far from Γ_i^m . (c) Illustration of the proxy surface (dashed circle) and near points (bold blue curve on Γ_i^m) when γ_l is touching Γ_i^m | 55 |

| | |
|--|----|
| 3.3.2 Illustration of the proxy surface for compressing $\mathbf{A}_{i,i+1}$ | 59 |
| 3.4.1 Three periods of the interface geometries γ_1 and γ_2 as defined in equation (3.4.1). | 65 |
| 3.4.2 Illustration of the real part of the total field of the solution to (3.1.1) for a geometry with 10 interfaces where the wave number alternates between 40 and $40\sqrt{2}$. The shown solution is for $\theta^{inc} = -0.845\pi$. The total number of discretization points was set to $N = 121, 136$, resulting in a flux error estimate of $2.3e - 8$. Seven periods in the geometry are shown. | 70 |
| 3.4.3 The three different “corner” geometries in the 11-layer structure. Three periods are shown. See Figure 3.4.2 for the full structure. . . . | 71 |
| 3.4.4 Illustration of 5 periods of (a) the original 11-layer structure and (b) the new structure obtained from replacing the fourth interface with a different geometry. The modified interface is in red box. | 71 |
| 4.0.1 Example of a complex confining wall geometry from [1]. The geometry is extracted from a generic microscopic image of the cross-section of a Fallopian tube. | 73 |
| 4.1.1 Model geometry for the Stokes BVP. | 76 |
| 4.2.1 (a) The proxy circle for Γ_p shown in dash blue line divides Γ_k into far (in green) and near (in red) with respect to Γ_p (b) The interaction between the far-filed part of Γ_k and Γ_p can be captured by the interaction between Γ_k^{far} and a smaller proxy circle for Γ_p shown in dash green. | 86 |
| 4.2.2 Dyadic partition for Γ_k^{far} used in the compression of \mathbf{A}_{kp}^{far} | 87 |

| | |
|---|----|
| 4.3.1 (a) The pipe geometry with the portion of the boundary to be refined boxed. The coordinates of the four sharp corners of the pipe is also shown. (b) The two Gaussian panels in the boxed region from the original discretization. (c) The four Gaussian panels that replaced the original two panels. | 91 |
| 4.3.2 (a) The bumpy pipe geometry with the portion of the boundary to be refined highlighted in red. (b) The two Gaussian panels to be removed from the original discretization. (c) The four Gaussian panels to be added to replace the original two panels. | 93 |
| 5.1.1 An example where the geometric approach leads to under-discretization. The boundary geometry and the location of the exterior source are given in (a). (b) and (d) plot the arc length function and the boundary density on a reference grid with 300 Gaussian panels against parameterization variable t . The relative errors for the arc length, curvature, and boundary density for discretization with different number of panels are plotted on a log scale in (c). The errors of the density in (c) are evaluated by comparing the solved density to the density solved from the reference grid. | 99 |

5.1.2 An example where the geometric approach leads to over-discretization. The boundary geometry and the location of the exterior source are given in (a). (b) and (d) plot the arc length function and the boundary density on a reference grid with 300 Gaussian panels against parameterization variable t . The relative errors for the arc length, curvature, and boundary density for discretization with different number of panels are plotted on a log scale in (c). The errors of the density in (c) are evaluated by comparing the solved density to the density solved from the reference grid. 100

5.1.3 Double disc geometry hit by plane wave with incident angle $\theta^{inc} = -\frac{\pi}{2}$. The discs have radius equal to 1, and the minimum distance between the two is chosen to be 0.05. 101

5.1.4 The boundary density and real part of the scatter field for solving the Helmholtz BVP with the different wave numbers: (a-b) $-\omega = 5$, (c-d) $-\omega = 10$, (e-f) $-\omega = 20$. The boundary data is due to a plane wave with incident angle $\theta^{inc} = -\frac{\pi}{2}$. The boundary is discretized with 100 uniformly distributed 10th order Gaussian panels. 102

5.4.1 (a) A butterfly geometry with interior target locations and exterior source locations. The geometry is generated by applying a corner smoothing scheme present in [2] to a butterfly shaped polygon. (b) zooms in the circled region in (a) to show that there is no sharp corners for this geometry. 119

5.4.2 (a) A fish geometry with interior target locations and two choices of exterior source locations (i) and (ii) for the right-hand-side function $f(\mathbf{x})$. (b) The right-hand-side function $f(\mathbf{x}(t))$ plotted against parameterization variable t for the two choices of exterior source locations (i) and (ii). 121

5.4.3 The final mesh produced by the new algorithm for $f(\mathbf{x})$ generated by choices of exterior source locations (i) and (ii). More panels are placed at the fish head (circled region) for (ii). Desired accuracy is set to $\epsilon = 10^{-8}$, and the initial mesh integrates the arc length function to $\epsilon_{init} = 10^{-1}$ 121

5.4.4 The “Fallopian tube piece” geometry. 122

A.3.1 The square with nose geometry. A nose of height d is smoothly attached to the a square. 148

A.3.2(a) The sunflower geometry with the portion of the boundary to be refined in red. (b) The three Gaussian panels in the boxed region from the original discretization. (c) Six Gaussian panels replacing the original three panels. 149

Tables

| | | |
|-----|---|----|
| 3.1 | Time in seconds and flux error estimates for applying the direct solver to a 3- and 9-layer geometry where the interfaces alternate between γ_1 and γ_2 defined in (3.4.1). N_i denotes the number of discretization points for each boundary charge density on the interface. The wave number alternates between 10 and $10\sqrt{2}$ | 66 |
| 3.2 | Time in seconds for solving 287 incident angles and 24 distinct Bloch phases on an 11-layer geometry shown in Figure 3.4.2. The incident angles are sampled from $[-0.89\pi, -0.11\pi]$ | 67 |
| 3.3 | Time in seconds for constructing and applying the fast direct solver to an 11-layer geometry (first column), a geometry that has the fourth interface changed (second column) and the wave number for the second layer changed from $40\sqrt{2}$ to 30 (third column). N_{total} is the number of discretization points on the interfaces in the unit cell. | 68 |
| 4.1 | Timing results for Stokes boundary value problem on the pipe with refined panel geometry. A separate HBS for Γ_p is constructed for test values with $N_p > 2000$, shown in bold font. | 91 |
| 4.2 | Timing results for applying the solver to the Stokes BVP on the bumpy pipe geometry. $Nk = 2528$, $Nc = 32$, and \mathbf{A}_{pp} is evaluated and inverted via dense linear algebra for all test cases. | 93 |

| | | |
|-----|---|-----|
| 5.1 | Results for applying (a) the artificial density algorithm and (b) the true density algorithm to a Laplace BVP defined on the butterfly geometry given in Figure 5.4.1. All tests start with a two-panel initial mesh which integrates the arc length function to one digit of accuracy. | 119 |
| 5.2 | Results for solving a Laplace BVP defined on the butterfly geometry by uniform Gaussian panel quadrature. | 120 |
| 5.3 | Results for applying the artificial density guided adaptive discretization algorithm to the “Fallopian tube piece” geometry. . . . | 122 |
| A.1 | Times for applying the solution techniques to (A.3.2) on the geometry in Figure A.3.2 with local refinement. | 151 |
| A.2 | Times for applying the solution technique to an interior Laplace-Dirichlet BVP on the square with thinning nose geometry. . . | 152 |
| A.3 | Times for applying the solution techniques to an interior Laplace-Dirichlet BVP on the square with fixed nose geometry. . . . | 152 |
| A.4 | Times for applying the solution techniques to an interior Laplace-Dirichlet BVP on the sunflower geometry in Figure A.3.2 with local refinement. | 152 |

Chapter 1

Introduction

This thesis presents numerical methods for solving two-dimensional elliptic boundary value problems (BVPs) recast as boundary integral equations (BIEs). The BIE formulation of a BVP involves only unknowns defined on the boundary of the domain, often referred to as *the boundary density*, thus reducing the dimension of the problem by one. Upon discretization, one is left with a dense linear system to solve. To obtain an accurate solution with acceptable computational cost, a good BIE formulation, an appropriate discretization, and an efficient numerical method for solving the linear system are all necessary. The first two parts of the proposed work adopt existing BIE formulation for BVP problems in wave scattering and Stokes flow, respectively, and construct fast solvers for the linear systems. The solvers are direct in the sense that they build an approximation to the inverse operator. Different from traditional inversion algorithms, such as Gauss elimination, which may be as expensive as $O(N^3)$ for dense systems with N unknowns, the solvers presented in this thesis are “fast” and scale linearly with respect to N . The last part of the thesis presents a new adaptive discretization technique for BIEs. The current standard technique for discretizing BIEs on complex geometries is to use an adaptive panel-based quadrature, where more panels are placed near the complex part of the geometry, such as high-curvature or sharp corners, to obtain accurate solution. The major novelty of the proposed scheme is that it does not require any global solves for the boundary density on any intermediate mesh.

1.1 Thesis outline

This thesis is organized as follows. The current chapter (chapter 1) introduces the basic idea and notations for BIE formulations and motivates the three projects. Section 1.2 briefly summarizes how to re-formulate a BVP as a BIE. Section 1.3 discusses the Nyström discretization for the boundary integral operator and solution methods for the resulting linear system. Section 1.4 motivates and introduces two applications: quasi-periodic scattering in layered media and simulation of bodies in confined Stokes flow. Finally, Section 1.5 elaborates on how an adaptive discretization technique would be useful for numerical solutions to BIEs.

Chapter 2 reviews some concepts and tools in literature, which are used in the new work presented in the later chapters. The following three chapters focus on the projects that comprise the thesis: chapter 3 presents a fast direct solver for quasi-periodic scattering in multilayer media; chapter 4 presents a fast direct solver for simulating confined Stokes flow which requires time-dependent local refinements to the confining wall discretization; chapter 5 presents an adaptive discretization technique for two-dimensional BIEs. Finally, chapter 6 concludes the thesis and discusses future directions.

1.2 BIE formulation for elliptic BVPs

A variety of problems in science and engineering can be formulated as BIEs. Examples include constant coefficient Laplace equation, Helmholtz equation, and Stokes flow. Since the unknown is only defined on the boundary of the domain, often referred to as *the boundary (charge) density*, the formulation reduces the dimension of the problem by one. Figure 1.2.1 gives an example for a 2D domain Ω enclosed by a simple curve

Γ . The unknown is the boundary charge density defined on Γ , which is a 1D object.

The integral formulation for a given elliptic BVP is not unique. In fact, different formulations for classic problems can be found in literature. Two kinds of BIE formulation for the Laplace problem with Dirichlet boundary condition are derived below as an example. The derivation applies to other equations as well. More complete theory can be found in standard text such as [3] and [4].

Consider the following BVP defined on the geometry given in Figure 1.2.1:

$$\begin{cases} -\Delta u(\mathbf{x}) = 0 & \text{for } \mathbf{x} \in \Omega & (1.2.1) \\ u(\mathbf{x}) = f(\mathbf{x}) & \text{for } \mathbf{x} \in \Gamma, & (1.2.2) \end{cases}$$

where $\Omega \subset \mathbb{R}^n$ and $\Gamma = \partial\Omega$.

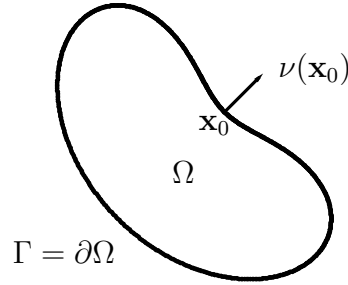


Figure 1.2.1 : Model geometry for the Laplace boundary value problem example.

- *Single-layer formulation* The solution can be represented as a single-layer potential

$$u(\mathbf{x}) = \int_{\Gamma} G(\mathbf{x}, \mathbf{y}) \tau(\mathbf{y}) dl(\mathbf{y}), \quad (1.2.3)$$

where $\tau(\mathbf{x})$ is some unknown boundary charge density, and $G(\mathbf{x}, \mathbf{y})$ is the Green's function $G(\mathbf{x}, \mathbf{y}) = -\frac{1}{2\pi} \log \left(\frac{1}{|\mathbf{x}-\mathbf{y}|} \right)$.

This representation naturally satisfies the PDE (1.2.1). To ensure the solution (1.2.3) also satisfies the boundary condition (1.2.2), one takes the limit of $u(\mathbf{x})$

as \mathbf{x} approaches the boundary Γ and sets the limit to be $f(\mathbf{x})$. This leads to the following BIE

$$\int_{\Gamma} G(\mathbf{x}, \mathbf{y}) \tau(\mathbf{y}) dl(\mathbf{y}) = f(\mathbf{x}), \text{ for } \mathbf{x} \in \Gamma. \quad (1.2.4)$$

- *Double-layer formulation* The solution can also be represented as a double-layer potential

$$u(\mathbf{x}) = \int_{\Gamma} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y}), \quad (1.2.5)$$

and by taking the limit as \mathbf{x} approaches the boundary, one ends up with a different BIE

$$-\frac{1}{2}\sigma(\mathbf{x}) + \int_{\Gamma} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y}) = f(\mathbf{x}), \text{ for } \mathbf{x} \in \Gamma. \quad (1.2.6)$$

The term $-1/2\sigma(\mathbf{x})$ comes from the jump condition, which can be found in [5, 3] and many other texts on Laplace equation.

In the example above, the single-layer formulation leads to an integral operator of the first-kind. The double-layer formulation, on the other hand, gives a second-kind Fredholm operator, provided the boundary geometry is smooth. A second-kind integral operator is favorable, since theoretical results including solution existence and computational stability often exist for the second kind [3, 6, 4].

1.3 Discretization and fast solvers for BIE

The first step towards a numerical solution to a BIE is to discretize the boundary integral equation. The work in this thesis adopts the *Nyström method* [3, 4, 7], while other methods such as the boundary element method [8, 9, 10], which discretizes

the weak form of the problem, are also available but are not discussed in this text. The basic idea for the Nyström method is to use quadrature to approximate the integral in the given integral equation. Let λ denote a scalar and $K(\mathbf{x}, \mathbf{y})$ denote a kernel function, consider a general form of integral equation defined on a closed and bounded set $D \subset \mathbb{R}^m$ for some $m \geq 1$.

$$\lambda\sigma(\mathbf{x}) + \int_D K(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) dy = f(\mathbf{x}). \quad (1.3.1)$$

The integral on Γ in (1.3.1) can be approximated by a quadrature rule with weights $\{\omega_j\}_{j=1}^N$ and nodes $\{\mathbf{z}_j\}_{j=1}^N$. The quadrature approximation leads to an equation for the density approximation σ_N

$$\lambda\sigma_N(\mathbf{x}) + \sum_{j=1}^N \omega_j K(\mathbf{x}, \mathbf{z}_j)\sigma_N(\mathbf{z}_j) = f(\mathbf{x}). \quad (1.3.2)$$

Seeking the solution to (1.3.2) at the quadrature nodes yields the linear system

$$\lambda\sigma_N(\mathbf{z}_i) + \sum_{j=1}^N \omega_j K(\mathbf{z}_i, \mathbf{z}_j)\sigma_N(\mathbf{z}_j) = f(\mathbf{z}_i), \text{ for } i = 1, \dots, N, \quad (1.3.3)$$

of which the unknown is a vector $\sigma_N = [\sigma_N(\mathbf{z}_1), \dots, \sigma_N(\mathbf{z}_N)]^T$. Then $\sigma_N(\mathbf{x})$ defined by

$$\sigma_N(\mathbf{x}) = \frac{1}{\lambda} \left(f(\mathbf{x}) - \sum_{j=1}^N \omega_j K(\mathbf{x}, \mathbf{z}_j)\sigma_N(\mathbf{z}_j) \right) \quad (1.3.4)$$

is a solution to (1.3.2). Comprehensive error analysis and convergence theorems of the Nyström method can be found in chapter 4 of [4].

For BIEs, D is the boundary of the problem domain $D = \partial\Omega = \Gamma$. In this thesis, the boundaries are parameterized closed curves of the form $\mathbf{x} = \gamma(t)$, where

the parameterization variable t lives in $[0, T]$ and $\gamma(0) = \gamma(T)$. Then the BIE

$$\lambda\sigma(\mathbf{x}) + \int_{\Gamma} K(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y})dl(\mathbf{y}) = f(\mathbf{x})$$

can be written as

$$\lambda\sigma(t) + \int_0^T K(t, s)\sigma(s)|\dot{\gamma}(s)| ds = f(t),$$

where $\sigma(t) = \sigma(\gamma(t))$, $K(t, s) = K(\gamma(t), \gamma(s))$, and $f(t) = f(\gamma(t))$. The dot notation is used to denote the derivative with respect to t : $\dot{\gamma}(t) = \left(\frac{d\gamma_1}{dt}(t), \frac{d\gamma_2}{dt}(t)\right)$. Let $\{t_j\}_{j=1}^N$ and $\{w_j\}_{j=1}^N$ be a quadrature rule for integrating functions defined on $[0, T]$. The corresponding linear system is

$$\lambda\sigma_i + \sum_{j=1}^N w_j K(t_i, t_j)\sigma_j|\dot{\gamma}_j| = f_i, \text{ for } i = 1, \dots, N, \quad (1.3.5)$$

where $\sigma_i = \sigma(t_i)$, $|\dot{\gamma}_i| = |\dot{\gamma}(t_i)|$, and $f_i = f(t_i)$

The linear system (1.3.5) can be written more compactly as

$$\mathbf{A}\sigma = (\lambda\mathbf{I} + \mathbf{K})\sigma = \mathbf{f}. \quad (1.3.6)$$

The matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ with entry $K_{ij} = K(t_i, t_j)|\dot{\gamma}_j|$ comes directly from the discretization of the boundary integral operator scaled by the arc-length evaluation, and $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix. The matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is often referred to as the *coefficient matrix*.

The integral operator in the BIE may be weakly singular or singular. The kernel $K(\mathbf{x}, \mathbf{y})$ may not be absolutely integrable, and the integral may need to be defined in the limit sense as $|\mathbf{x} - \mathbf{y}| \rightarrow 0^+$. Special quadrature rules have been developed to deal

with such complexity. Some examples are high-order quadrature schemes for singular integral developed in [11, 12] and quadrature schemes for boundaries with corners in [13, 14, 15, 16, 17]. This remains an active area of research.

The linear system arising from the discretization of BIEs is dense. This stopped the BIE formulation from being widely used for approximating solutions to PDEs until the 1980s, when the fast summation technique called the *fast multipole method (FMM)* [18] was developed. This thesis describes a method as “fast” if its complexity is asymptotically less than traditional methods. For example, given a desired tolerance ϵ , the cost of FMM is $O(N \log 1/\epsilon)$ for a matrix-vector product, while dense linear algebra matrix-vector multiplication is $O(N^2)$. The huge reduction in cost is obtained by exploiting the fact that the off-diagonal blocks of the coefficient matrix is rank-deficient. The FMM can be used to accelerate the matrix-vector multiplications for an iterative solver to obtain a solution to the discretized BIE. Work by Hackbusch et al. at the end of the twentieth century, introduced the concept of *data-sparse* matrix [19, 20, 21, 22] and made the observation that the off-diagonal deficiency property can be used to build fast matrix factorizations and inversions. The cost of early implementation of the inversion scheme is $O(N(\log N)^p)$, for some small integer p . This is significantly faster than traditional inversion schemes for general dense matrices, such as Gaussian elimination. Work by Martinsson and Rokhlin [23] reduced the cost to $O(N)$ by separating the forward “compression” and inversion into individual steps. This enables other algorithms to be incorporated into the solution algorithm for BIEs: for example, randomized linear algebra based matrix factorization methods can be used to aid the forward compression [24, 25]. Later work on fast inversion methods, such as the *hierarchically semi-separable (HSS)* matrix method [26, 27, 28, 29], the *hierarchically block-separable (HBS)* matrix method [30], and

the *HODLR* method [31] took advantage of the earlier development and managed to obtain solution with $O(N)$ or $O(N \log N)$ cost for BIEs defined on planar curves.

Generally speaking, the fast solution techniques for BIEs mentioned above belong to one of the two categories:

- (i). *A fast matrix-vector multiplication scheme coupled with an iterative solver.*

Examples of fast matrix-vector multiplication schemes include the FMM [18] and hierarchical matrix method [19, 21, 22]. A popular choice of iterative solver is GMRES, with or without generic or problem-specific preconditioner [32, 33, 34, 35, 36].

- (ii). *A fast direct solver.* Most common examples are hierarchical matrices whose inverse operator can be constructed in a fast manner [23, 26, 30, 37].

The iterative approach (i) is ideal for problems of which the coefficient matrix has a clustered spectrum. For other problems, the method may have convergence issue or converges slowly. For example, Figure 1.3.1 from [38] gives an example of a problem, of which the singular values of the coefficient matrix lies on the circle of radius one in the complex plane but not clustered. The problem is well-conditioned but a FMM accelerated GMRES takes 248 iterations to reach an accuracy of 10^{-8} for each incident wave. Complex geometries may also lead to ill-conditioned problems. Method (i) often performs poorly for such problems, as well as problems that are physically ill-conditioned, e.g. the Helmholtz equation with a wave number close to an eigenvalue of the Laplacian.

The direct approach (ii) is an ideal alternative for these problems. Direct solvers allow for highly accurate solutions even when the conditioning of the problem is bad. In addition, fast direct solvers are also very efficient in handling problems

with multiple right-hand-sides. The approximate inverse needs to be constructed once. Then solutions to each different right-hand-side can be obtained by inexpensive matrix-vector multiplications.

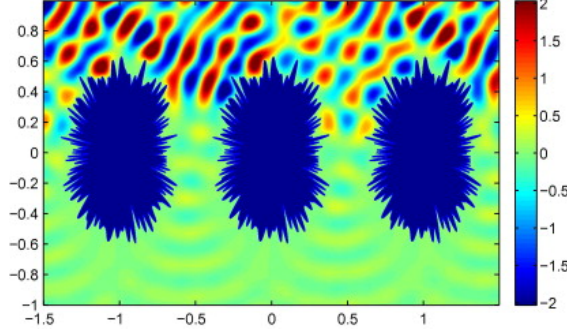


Figure 1.3.1 : Example of a geometrically ill-conditioned problem.

Plot of the total scatter field off an infinite array of objects (in dark blue), which is solved as a test problem in [38]. The geometric complexity of the surface causes the spectrum of the discretized integral operator to be scattered around the origin. Iterative methods converge slowly.

The fast direct solvers presented in this manuscript are based upon previous work on hierarchical matrix methods, for example the \mathcal{H} matrix and \mathcal{H}^2 -matrix approaches such as *HSS* and *HBS* [26, 30, 37]. These methods are related to each other and share the same steps for obtaining a solution to a discretized BIE. Let ϵ be a user prescribed tolerance. A fast direct solver obtains the solution to (1.3.6) by

- (1) constructing a compressed representation of the coefficient matrix $\tilde{\mathbf{A}}$ with $\|\tilde{\mathbf{A}} - \mathbf{A}\| < \epsilon$;
- (2) constructing an approximate inverse $\tilde{\mathbf{B}}$ with $\|\tilde{\mathbf{B}} - \tilde{\mathbf{A}}^{-1}\| < \epsilon$;
- (3) applying the approximate inverse $\tilde{\mathbf{B}}$ to the right-hand-side vector.

In order to be useful, the operator $\tilde{\mathbf{B}}$ is constructed so that it can be applied rapidly, i.e. with $O(N)$ cost.

Steps (1) and (2) are both independent of the right-hand-side vector \mathbf{f} , and they constitute the *precomputation* of the solver. Step (3) depends on the right-hand-side vector and is referred to as the *solve* step.

Once the boundary density is found, solution at any point in the domain can be evaluated via a convolution. For the Laplace Dirichlet BVP example, this is given by equation(1.2.3) or (1.2.5), depending on which BIE formulation is used. Fast summation methods can be used to reduce the cost of evaluating the convolution. If the location of the target point is close to or on the boundary, *near-field evaluation* schemes are necessary to obtain accurate solution[39, 40, 41].

1.4 Use BIE to simulate applications

The major goal of the presented work is to extend the range of applications that can be modeled via the boundary integral approach. Two particular applications are considered. Section 1.4.1 discusses wave scattering off a layered structure that models the optical devices specially designed for wave manipulation. Section 1.4.2 focuses on modeling Stokes flow in confined geometry, which often plays a key role in numerical simulations of bacteria or micro-swimmers in biology and fluid mechanics.

The author is interested in these two problems for the following reasons. First of all, both problems raise lots of interest in their respective research community as these problems are bottlenecks in simulations run by practitioners and engineers. For both problems, nice boundary integral formulations exist, but fast algorithms working with these formulations remain to be developed. The proposed fast direct solvers will allow the practitioners to actually exploit the nice properties of the existing BIE formulations to run large size simulations with manageable cost.

1.4.1 Quasi-periodic scattering in layered media

Wave scattering off multilayered quasi-periodic structures arises in the design of optical and electromagnetic devices. Practitioners combine different materials, metal or non-metal, together into a layered structure to manipulate waves. Examples include solar cells and dielectric gratings [42, 43, 44, 45, 46, 47]. To study the wave diffraction from these structures, one often needs to solve the Helmholtz equation with different wave numbers for each of the layers.

In [48], Cho and Barnett construct a BIE formulation that is robust even at so-called Wood’s anomalies. The proposed work adopts this formulation and constructs a new fast direct solution technique, whose computational cost scales linearly with respect to the number of points on the layer interfaces. For many of the applications, multiple incident waves need to be solved for the same structure. For example, a Bragg diagram created from the solution of 200 incident angles is desirable in many engineering applications. A fast direct solver is ideal in this setting, since once the approximate inverse is constructed, solving extra right-hand-sides will be cheap.

The direct solver is built in a way that local changes in the multilayer structure lead to local changes in the solver as well. For example, if the shape of one interface between two layers is changed or the material property of one particular layer becomes different, one can easily redefine the corresponding parts of the solver and reuses the rest of it instead of building a new solver from scratch. This feature becomes very handy in the optimal design setting: when the geometry and material properties are close to the optimal choice, the changes in the scattering problem are localized to a few layer.

Chapter 3 discusses the physical problem, BIE formulation, and solution technique in detail.

1.4.2 Stokes flow in confined geometry

An interesting problem in biology is to study the dynamics of objects inside a confined region. Particular applications include modeling blood coagulation, developing microfluidic devices, and understanding the behavior of bacteria and other swimmers. For such problems, the confining wall is static while the interior bodies (bacteria, swimmers, etc.) evolve in time. These problems can be formulated as BIE defined on the static wall coupled with extra equations modeling the dynamics of the bodies. And a time-stepping scheme is used to evolve the bodies in time.

One particular interesting case is when the geometry of the confining wall is complicated and the swimmers appear only in certain region of the enclosed domain for each time step. The motivating example in [49] simulates a group of bacteria swimming through a complicated microfluidic device. The swimmers cumulate in several clusters and only appear in a few regions of the device for each time step. The shape of the device is complicated and requires a large number of discretization points to resolve. If a cluster of swimmers get close to a part of the device's boundary, extra refinement becomes necessary on this part of the boundary to capture the density's behavior. As the swimmers moving through the device in time, the regions that need extra refinement change. Chapter 4 proposes a solution technique built specially to accommodate such time-evolving discretization refinement: a fast direct solver is built for the static wall geometry where no swimmers exist, and then for each future time step the changes due to refinements in the regions approached by swimmers are captured via a low-rank update, which allows the solver for the discretization at this particular time step to be applied with linear cost. The proposed work is an extension of the previous work by the author on building a fast direct solver for locally-perturbed geometry [50] and a collaboration with the authors of [49].

1.5 An adaptive discretization strategy

As briefly mentioned in section 1.2, finding the appropriate quadrature rule for the boundary curve Γ is among the numerical challenges when approximating solutions to PDEs using BIEs. The geometry needs to be resolved well enough to obtain desired accuracy; however, over-resolution should be avoided for computational efficiency. Global quadrature schemes such as (composite) Trapezoidal rule often struggle to handle complex geometries and are hard to generalize to higher dimensions. Panel-based quadratures, such as Gaussian quadrature, offer more flexibility and generalize to higher dimensions easily. Particularly for complex geometries, the balance point between solution accuracy and computational efficiency may be found by coupling panel based quadrature with an adaptive scheme that identifies and refines the parts of the boundary as needed.

Chapter 5 presents a panel-based adaptive discretization technique for two-dimensional BIEs. The proposed technique solves small local systems to update a quantity called *the artificial density* and uses it to choose which part of the boundary to refine. The technique is more efficient compared to approaches which identify the relative complex parts of the boundary by looking at the globally solved boundary density.

Chapter 2

Tools and solvers in literature

The fast direct solvers and discretization technique proposed in this thesis build upon previous work on numerical solutions for BIE and other numerical methods such as random linear algebra based matrix factorization. This chapter goes through some tools and ideas that are used repeatedly in the development of the new solvers and discretization scheme.

The chapter is organized as follows. Section 2.1 reviews the definition of interpolative decomposition, which is used in compressing the low-rank subblocks of the coefficient matrix and other rank-deficient matrices. Section 2.2 summarizes a commonly used idea in potential theory, the separation of near-field and far-field interaction. Section 2.3 reviews a particular fast direct solver, the HBS solver [30], since it is used to build the proposed new solvers in both Chapter 3 and 4. Finally, Section 2.4 reviews the author's previous work on solving BIEs with locally perturbed geometry [50]. The Stokes solver in Chapter 4 builds upon the idea presented in this section and extends the technique to accelerate numerical simulation of Stokes flow.

2.1 Interpolative decomposition (ID) for rank-deficient matrices

Fast numerical solution techniques for BIE, both iterative and direct, are based on the rank-deficiency of the off-diagonal blocks of the coefficient matrix. Given a user-

specified tolerance ϵ , these off-diagonal blocks can be *compressed*, i.e. approximated by a low-rank factorization. This will be referred to as the ϵ -rank factorization for the rest of the thesis. The formal definition is as follows.

Definition 2.1 \mathbf{LR} is an ϵ -rank factorization of matrix $\mathbf{S} \in \mathbb{R}^{M \times N}$ with rank $k = k(\epsilon)$, if $\mathbf{L} \in \mathbb{R}^{M \times k}$ and $\mathbf{R} \in \mathbb{R}^{k \times N}$ satisfy $\|\mathbf{S} - \mathbf{LR}\| < \epsilon\|\mathbf{S}\|$.

Examples of ϵ -rank factorization techniques include truncated singular value decomposition (SVD) [51], rank revealing QR factorizations [52], CUR decompositions [53], and interpolative decompositions (ID). Both the HBS solver by Gillman et al. in [30] and the new solvers presented in this thesis use the ID approach. Thereby, the formal definition of ID is given below.

Definition 2.2 Given a tolerance ϵ , a matrix \mathbf{S} of dimension $M \times N$ can be approximated by a *interpolative decomposition (ID)* of the form

$$\mathbf{S} \approx \mathbf{PS}(J, :), \quad (2.1.1)$$

where $J = (J_1, \dots, J_k)$ is a reordering of a size- k sub-collection of the row indices of \mathbf{S} , and \mathbf{P} is a matrix of dimension $M \times k$ that contains an identity matrix of size $k \times k$. The value of k depends on the user-specified tolerance ϵ , and the approximation satisfies $\|\mathbf{S} - \mathbf{PS}(J, :)\| < \epsilon\|\mathbf{S}\|$.

The matrix \mathbf{P} is referred to as the *interpolation matrix*, and the vector J is called the *row index vector*. Analogously, \mathbf{S} can have an ID decomposition of the form $\mathbf{S} \approx \mathbf{S}(:, J)\mathbf{P}$, where the *column index vector* J is a sub-collection of the column indices of \mathbf{S} . When applied to the coefficient matrix in (1.3.6), the row or column index corresponds to discretization point on the boundary. The discretization points

selected by J are called *the skeleton points*.

A complete theory and an $O(N \log N)$ technique for obtaining the ID for a $N \times N$ matrix is presented in Cheng et al. [54]. The HBS solver and the new solver both adopt the randomized algorithm by in Martinsson et al. [55].

2.2 Far-field and near-field separation for boundary integral operator

This section discusses a commonly used idea in fast algorithms for BIEs: the separation of self, near-field, and far-field interaction. Consider subinterval $b \subsetneq [0, 2\pi]$ with corresponding boundary segment $\Gamma_b \subsetneq \Gamma$ as illustrated in Figure 2.2.1(a). The rest of the boundary geometry $\Gamma \setminus \Gamma_b$ can be classified into two categories based on the distance to Γ_b . One way to do this is to draw a circle centered at Γ_b and with radius $r_{\text{rel}} > 1$ times larger than the smallest circle enclosing Γ_b . This circle is referred to as *the proxy circle*. $r_{\text{rel}} = 1.75$ is often a good choice in practice. The *near-field* of b $\Gamma_{\text{near}(b)}$ is defined as the part of the boundary that lies within the circle minus b itself, and the *far-field* of b $\Gamma_{\text{far}(b)}$ is the part outside of the circle. Figure 2.2.1 (b) illustrates the proxy circle as well as the near-field and far-field of b for a sample geometry.

Recall the double layer Laplace BIE from section 1.2, the row equation for \mathbf{x} is

$$-\frac{1}{2}\sigma(\mathbf{x}) + \int_{\Gamma} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y}) = f(\mathbf{x}).$$

The integral with respect to \mathbf{y} can be broken into an integral on Γ_b plus the integral

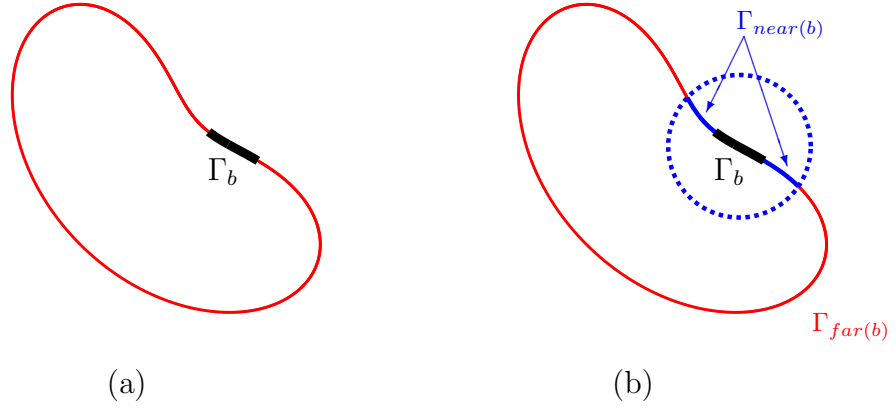


Figure 2.2.1 : An illustration of near-field and far-field separation. (a) The boundary geometry with Γ_b in bold line. (b) The proxy circle (dotted blue line) that separates $\Gamma \setminus \Gamma_b$ into the far-field $\Gamma_{far(b)}$ and the near-field $\Gamma_{near(b)}$ with respect to b .

on the near-field and the far-field of b

$$\begin{aligned}
 & \underbrace{-\frac{1}{2}\sigma(\mathbf{x}) + \int_{\Gamma_b} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y})}_{\text{self}} + \underbrace{\int_{\Gamma_{near(b)}} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y})}_{\text{near-field}} \\
 & \quad + \underbrace{\int_{\Gamma_{far(b)}} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \nu(\mathbf{y})} \sigma(\mathbf{y}) dl(\mathbf{y})}_{\text{far-field}} = f(\mathbf{x}).
 \end{aligned}$$

The corresponding block row equation in the linear system will be of the format

$$\mathbf{A}_{b,b}\sigma_b + \mathbf{A}_{b,near(b)}\sigma_{near(b)} + \mathbf{A}_{b,far(b)}\sigma_{far(b)} = \mathbf{f}_b.$$

Here $\mathbf{A}_{b,b}$ is a diagonal subblock of the coefficient matrix, while $\mathbf{A}_{b,near(b)}$ and $\mathbf{A}_{b,far(b)}$ are off-diagonal subblocks. This separation is very useful from the following perspectives:

- The low-rank off-diagonal blocks are separated from the full-rank diagonal blocks, which allows the off-diagonal blocks corresponding to the near- and far-field interaction to be approximated by ϵ -rank factorizations.

- The further separation of “non-self” interactions into near-field and far-field is also significant. Although both the near-field and the far-field interactions are low-rank, the far-field interaction has even lower ranks compared to the near-field, as the far-field is shielded away from Γ_b by a nontrivial distance. From potential theory, the potential at points on Γ_b due to the far-field $\Gamma_{far(b)}$ can be represented as a linear combination of basis functions defined on the proxy circle.

The separation and using basis functions on the proxy circle to represent far-field interactions are often exploited in both building BIE formulation for problems as well as fast solution methods for BIEs. For example, in the periodizing scheme for the multilayer scattering problem (Chapter 3 section 3.2.1), basis functions on the proxy circle are defined to capture the contribution from non-neighbor copies of the periodic scatterer. As for fast solvers, the near-field and far-field separation can be used to reduce the cost of compressing the off-diagonal blocks. Let subscript c denote the collection of points on $\Gamma_c := \Gamma \setminus \Gamma_b$, the coefficient matrix contains the off-diagonal block $\mathbf{A}_{bc} = [\mathbf{A}_{b,near(b)} \mid \mathbf{A}_{b,far(b)}]$. One way to obtain a low-rank factorization of \mathbf{A}_{bc} is to first evaluate the matrix and then apply ID to find the skeleton row index J_b and interpolation matrix \mathbf{P}_b so that $\mathbf{A}_{bc} \approx \mathbf{P}_b \mathbf{A}_{bc}(J_b, :)$. This approach is numerically expensive, since (row-based) ID is applied to a matrix with lots of columns. A lower cost approach is available from the potential theory for far-field interactions mentioned above. Recall that the far-field interaction can be written in terms of a linear combination of basis function defined on the proxy circle. For a Laplace problem, the basis functions can be defined as the potential due to a collection of point charges placed on the proxy circle. In practice, 75 points uniformly sampled on the proxy circle is sufficient. Let $\mathbf{A}_{b,proxy}$ denote the interaction between

points on Γ_b and the point charges on the proxy. One can obtain the skeleton points $skel(b)$ and interpolation matrix $\hat{\mathbf{P}}_b$ by applying ID to compress the rows of $\hat{\mathbf{A}}_{bc} = [\mathbf{A}_{b,near(b)} | \mathbf{A}_{b,proxy}]$. Once $skel(b)$ is determined, one only needs to evaluate $\mathbf{A}_{skel(b),c}$ the interaction between the skeleton points on Γ_b and points on Γ_c to have the low-rank approximation $\mathbf{A}_{bc} \approx \hat{\mathbf{P}}_b \mathbf{A}_{skel(b),c}$.

2.3 Hierarchically block-separable (HBS) solver for BIEs

The work presented in Chapter 3 and Chapter 4 is based on a fast direct solution technique, the *Hierarchically block-separable (HBS)* method [23, 30]. This section briefly reviews the method and prepares the reader for the two later chapters.

The data structure used for storing the matrices in the HBS solver is very similar to that in the well-known *Hierarchically semi-separable (HSS)* method [26]. The terminology is adopted mainly because it clarifies the matrix property used for the compression and inversion schemes.

2.3.1 Block-separable matrix

The data-sparse representation of the coefficient matrix in the HBS method starts with the following definition of a block-separable matrix [30]:

Definition 2.3 Assume \mathbf{A} is an $np \times np$ matrix of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \cdots & \mathbf{A}_{1,p} \\ \mathbf{A}_{2,1} & \mathbf{D}_2 & \mathbf{A}_{2,3} & \cdots & \mathbf{A}_{2,p} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{A}_{p,1} & \mathbf{A}_{p,2} & \mathbf{A}_{p,3} & \cdots & \mathbf{D}_p \end{bmatrix}, \quad (2.3.1)$$

where each of the sub-blocks has dimension $n \times n$. \mathbf{A} is *block-separable* with block-rank k if, for $\alpha, \beta = 1, 2, \dots, p$, there exist matrices \mathbf{U}_α , \mathbf{V}_β , and $\hat{\mathbf{A}}_{\alpha,\beta}$ such that

$$\mathbf{A}_{\alpha,\beta} = \mathbf{U}_\alpha \hat{\mathbf{A}}_{\alpha,\beta} \mathbf{V}_\beta^*. \quad (2.3.2)$$

The dimension of the factor matrices depends on k : \mathbf{U}_α and \mathbf{V}_β has dimension $n \times k$, and $\hat{\mathbf{A}}_{\alpha,\beta}$ has dimension $k \times k$.

Let \mathbf{A} be a block-separable matrix as defined above, the off-diagonal sub-block factorizations (2.3.2) can be combined together to form a factorization of \mathbf{A}

$$\mathbf{A} = \mathbf{U} \hat{\mathbf{A}} \mathbf{V}^* + \mathbf{D}, \quad (2.3.3)$$

where

$$\begin{aligned} \mathbf{U} &= \text{diag}(\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_p), \\ \mathbf{V} &= \text{diag}(\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_p), \\ \mathbf{D} &= \text{diag}(\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_p), \text{ and} \\ \mathbf{A} &= \begin{bmatrix} \mathbf{0} & \hat{\mathbf{A}}_{1,2} & \hat{\mathbf{A}}_{1,3} & \cdots & \hat{\mathbf{A}}_{1,p} \\ \hat{\mathbf{A}}_{2,1} & \mathbf{0} & \hat{\mathbf{A}}_{2,3} & \cdots & \hat{\mathbf{A}}_{2,p} \\ \vdots & \vdots & \vdots & & \vdots \\ \hat{\mathbf{A}}_{p,1} & \hat{\mathbf{A}}_{p,2} & \hat{\mathbf{A}}_{p,3} & \cdots & \mathbf{0} \end{bmatrix}. \end{aligned} \quad (2.3.4)$$

Figure 2.3.1 demonstrates the sparsity pattern of this decomposition for the case of $p = 4$.

$$\mathbf{A} = \mathbf{U} \hat{\mathbf{A}} \mathbf{V}^* + \mathbf{D}$$

Figure 2.3.1 : Illustration of a block-separable factorization as in (2.3.3) . Matrix \mathbf{A} is a block-separable matrix with $4 \times 4 = 16$ sub-blocks. \mathbf{U} , \mathbf{V} , and \mathbf{D} are all block-diagonal matrices. Matrix $\hat{\mathbf{A}}$ has zero blocks on the diagonal.

The factorization (2.3.3) allows matrix \mathbf{A} to be inverted via a variant of the Sherman-Morrison-Woodbury formula:

Lemma 2.1

Suppose that \mathbf{A} is an $np \times np$ invertible block-separable matrix that admits the decomposition in (2.3.3). Furthermore, assume that matrix \mathbf{D} , $(\mathbf{V}^* \mathbf{D}^{-1} \mathbf{U})$, and $(\hat{\mathbf{A}} + (\mathbf{V}^* \mathbf{D}^{-1} \mathbf{U})^{-1})$ are all invertible, then

$$\mathbf{A}^{-1} = \mathbf{E}(\hat{\mathbf{A}} + \tilde{\mathbf{D}})^{-1} \mathbf{F}^* + \mathbf{G}, \quad (2.3.5)$$

where

$$\tilde{\mathbf{D}} = (\mathbf{V}^* \mathbf{D}^{-1} \mathbf{U})^{-1},$$

$$\mathbf{E} = \mathbf{D}^{-1} \mathbf{U} \tilde{\mathbf{D}},$$

$$\mathbf{F} = (\tilde{\mathbf{D}} \mathbf{V}^* \mathbf{D}^{-1})^*, \text{ and}$$

$$\mathbf{G} = \mathbf{D}^{-1} - \mathbf{D}^{-1} \mathbf{U} \tilde{\mathbf{D}} \mathbf{V}^* \mathbf{D}^{-1}.$$

Figure 2.3.2 demonstrates the sparsity pattern of this factorization of \mathbf{A}^{-1} .

$$\mathbf{A}^{-1} = \mathbf{E} (\tilde{\mathbf{A}} + \hat{\mathbf{D}})^{-1} \mathbf{F}^* + \mathbf{G}$$

Figure 2.3.2 : Illustration of the inverse factorization of a block separable matrix \mathbf{A} (2.3.5) . Matrix \mathbf{A} is a block-separable matrix with $4 \times 4 = 16$ sub-blocks and can be factored as (2.3.3). \mathbf{E} , $\tilde{\mathbf{D}}$, \mathbf{F} , and \mathbf{G} are all block-diagonal matrices. Matrix $(\tilde{\mathbf{A}} + \hat{\mathbf{D}})^{-1}$ is the only dense matrix remained.

For a block-separable matrix \mathbf{A} , Lemma 2.1 reduces dense inversion of \mathbf{A} , which scales $O(n^3 p^3)$, to the inversion of a much smaller matrix $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$, which scales $O(k^3 p^3)$, plus multiplication with block diagonal matrices, which scales $O(pn^3)$. In the case where $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ is also block-separable, one can apply Lemma 2.1 to $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ as well and obtain a nested factorization of \mathbf{A}^{-1} with a even smaller dense matrix to be inverted directly. This motivates the definition of the *hierarchically block-separable* matrix in [30].

2.3.2 Hierarchically block-separable matrix

The block-separable matrix definition in the previous section can be coupled with a tree structure for the row index of the matrix to define a hierarchically block-separable matrix factorization. For simple demonstration, Figure 2.3.3 visualizes a three-level binary tree for a matrix with 400 rows in total, which corresponds to a 400 point discretization for the boundary Γ in a Laplace problem.

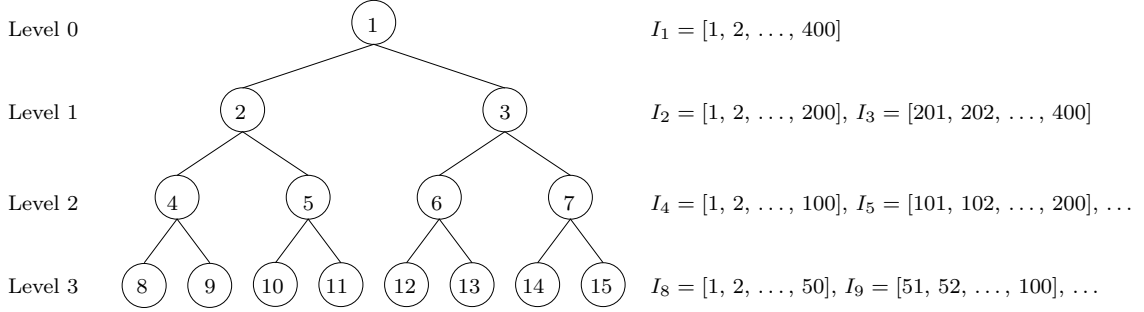


Figure 2.3.3 : Numbering of box nodes in a fully populated binary tree with $L = 3$ levels. The root is the original index vector $I = I_1 = [1, 2, \dots, 400]$ of the discretization points in the parameter space of Γ .

For a given binary tree structure for the row index, [30] defines a hierarchically block-separable matrix as follows.

Definition 2.4 Suppose \mathcal{T} is a given binary tree with L fully-populated levels and that for each leaf node τ , the index vector I_τ holds precisely n points. A matrix \mathbf{A} of dimension $N \times N$ is *hierarchically block-separable* with block rank k and with respect to \mathcal{T} if the following conditions hold:

- (1) (Leaf level) For any pair of distinct leaf nodes τ and τ' , let $\mathbf{A}_{\tau,\tau'} := \mathbf{A}(I_\tau, I_{\tau'})$, whose dimension is $n \times n$, then there must exist matrices \mathbf{U}_τ , $\mathbf{V}_{\tau'}$, and $\hat{\mathbf{A}}_{\tau,\tau'}$ such that

$$\mathbf{A}_{\tau,\tau'} = \mathbf{U}_\tau \hat{\mathbf{A}}_{\tau,\tau'} \mathbf{V}_{\tau'}^*, \quad (2.3.6)$$

where $\hat{\mathbf{A}}_{\tau,\tau'}$ is of dimension $k \times k$.

- (2) (Non-leaf level) On level $l = L - 1, L - 2, \dots, 1$, for any pair of distinct box nodes τ and τ' with children σ_1, σ_2 and σ'_1, σ'_2 respectively, define

$$\mathbf{A}_{\tau,\tau'} := \begin{bmatrix} \hat{\mathbf{A}}_{\sigma_1,\sigma'_1} & \hat{\mathbf{A}}_{\sigma_1,\sigma'_2} \\ \hat{\mathbf{A}}_{\sigma_2,\sigma'_1} & \hat{\mathbf{A}}_{\sigma_2,\sigma'_2} \end{bmatrix}, \quad (2.3.7)$$

where $\mathbf{A}_{\tau,\tau'}$ is of dimension $2k \times 2k$. Then there must exist matrices \mathbf{U}_τ , $\mathbf{V}_{\tau'}$, and $\hat{\mathbf{A}}_{\tau,\tau'}$ such that

$$\mathbf{A}_{\tau,\tau'} = \mathbf{U}_\tau \hat{\mathbf{A}}_{\tau,\tau'} \mathbf{V}_{\tau'}^*, \quad (2.3.8)$$

where dimension of $\hat{\mathbf{A}}_{\tau,\tau'}$ equals to $k \times k$.

For the three level tree structure shown in Figure 2.3.3, a hierarchically block-separable matrix \mathbf{A} can be factored as

$$\mathbf{A} = \bar{\mathbf{U}}^{(3)} (\bar{\mathbf{U}}^{(2)} (\bar{\mathbf{U}}^{(1)} \bar{\mathbf{B}}^{(0)} (\bar{\mathbf{V}}^{(1)})^* + \bar{\mathbf{B}}^{(1)}) (\bar{\mathbf{V}}^{(2)})^* + \bar{\mathbf{B}}^{(2)}) (\bar{\mathbf{V}}^{(3)})^* + \bar{\mathbf{D}}^{(3)}, \quad (2.3.9)$$

with sparsity pattern shown in Figure 2.3.4.

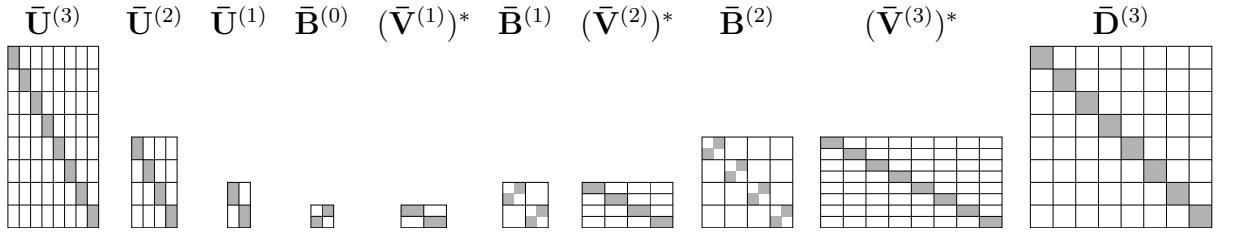


Figure 2.3.4 : Illustration of the block structure of the telescoping factors for a 3-level HBS representation. The telescoping factorization is given in (2.3.9).

Lemma 2.1 can then be applied in a nested fashion to invert the telescoping factorization (2.3.9).

2.3.3 HBS representation for the coefficient matrix

With the Nyström discretization, the rows and columns in the coefficient matrix corresponds to quadrature points on the boundary of the domain. By the rank-deficient nature of the off-diagonal blocks, an HBS representation for the coefficient matrix can

be built with a binary tree structure for the discretization points in parameterization space. However, directly constructing the factors via applying matrix factorization methods, such as ID, to the off-diagonal blocks is very expensive, since the block has either lots of rows or lots of columns. To further reduce the cost, [30] uses the separation of near-field and far-field idea covered in section 2.2 and replaces explicit sub-block evaluation corresponding to far-field interactions with proxy circle interactions. The resulting algorithm is linear with respect to the number of quadrature points on the boundary, and, once the forward HBS representation and the inversion are completed, solving one new right-hand-side is extremely cheap.

2.4 A fast direct solver for locally-perturbed geometry

This section reviews the author’s previous work on building a fast direct solver for BVPs defined on locally perturbed geometries in 2D [50]. A locally perturbed geometry is a geometry that is the same as the original geometry except in small (or local) portion of the geometry. Such problems arise in applications such as optimal shape design. In each iteration or optimization cycle the changes to the object shape often stay local to certain parts of the object.

The basic idea of the solver, as first proposed in [56], is to recast the BVP as a BIE and formulate an extended linear system that allows for the BIE on the new geometry to be expressed in terms of a linear system on the original geometry plus a correction to account for the local perturbation. The fast direct solver reviewed in this section follows this idea.

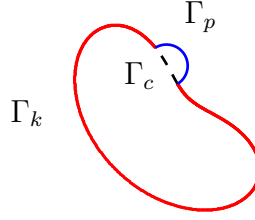


Figure 2.4.1 : A sample locally perturbed geometry where the original boundary is $\Gamma_o = \Gamma_k \cup \Gamma_c$, the portion of the boundary being removed is Γ_c , the portion of the original boundary remaining is Γ_k and the newly added boundary piece is Γ_p .

2.4.1 Locally perturbed geometry and extended linear system

Consider a boundary value problem on a geometry with a local perturbation as illustrated in Figure 2.4.1. Let Γ_o denote the boundary of the original geometry, Γ_k denote the portion of the boundary that is not changing and Γ_c denote the portion that is cut or removed. So $\Gamma_o = \Gamma_c \cup \Gamma_k$. Let Γ_p denote the new portion of the boundary. Then the new geometry has a boundary Γ_n defined by $\Gamma_n = \Gamma_k \cup \Gamma_p$.

The discretized linear systems (1.3.6) can be partitioned according to this notation. In other words, the original system can be expressed as

$$\mathbf{A}_{oo}\sigma_o = \begin{bmatrix} \mathbf{A}_{kk} & \mathbf{A}_{kc} \\ \mathbf{A}_{ck} & \mathbf{A}_{cc} \end{bmatrix} \begin{pmatrix} \sigma_k \\ \sigma_c \end{pmatrix} = \begin{pmatrix} \mathbf{f}_k \\ \mathbf{f}_c \end{pmatrix} = \mathbf{f}_o, \quad (2.4.1)$$

and the linear system for the perturbed geometry can be expressed as

$$\mathbf{A}_{nn}\sigma_n = \begin{bmatrix} \mathbf{A}_{kk} & \mathbf{A}_{kp} \\ \mathbf{A}_{pk} & \mathbf{A}_{pp} \end{bmatrix} \begin{pmatrix} \sigma_k \\ \sigma_p \end{pmatrix} = \begin{pmatrix} \mathbf{f}_k \\ \mathbf{f}_p \end{pmatrix} = \mathbf{f}_n \quad (2.4.2)$$

where σ_k denotes the vector whose entries are the approximate solution at the discretization points on Γ_k , σ_c denotes the vector whose entries are the approximate

solution at the discretization points on Γ_c , etc. Likewise \mathbf{A}_{kk} is the submatrix of the discretized integral equation corresponding to the interaction of Γ_k with itself, \mathbf{A}_{kc} is the submatrix of the discretized integral equation corresponding to the interaction of Γ_k with Γ_c , etc.

The discretized problem on Γ_n can be expressed as an extended linear system [56] by

$$\left(\underbrace{\begin{bmatrix} \mathbf{A}_{oo} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{pp} \end{bmatrix}}_{\mathbf{A}} + \underbrace{\begin{bmatrix} 0 & \begin{pmatrix} -\mathbf{A}_{kc} \\ -\mathbf{B}_{cc} \end{pmatrix} & \mathbf{A}_{op} \\ \mathbf{A}_{pk} & 0 & 0 \end{bmatrix}}_{\mathbf{Q}_{orig}} \right) \underbrace{\begin{pmatrix} \sigma_k \\ \sigma_c \\ \sigma_p \end{pmatrix}}_{\sigma_{ext}} = \underbrace{\begin{pmatrix} \mathbf{f}_k \\ \mathbf{0} \\ \mathbf{f}_p \end{pmatrix}}_{\mathbf{f}_{ext}} \quad (2.4.3)$$

where \mathbf{A}_{kc} denotes the submatrix of \mathbf{A}_{oo} corresponding to the interaction between Γ_k and Γ_c , \mathbf{A}_{op} denotes the discretization of the double layer integral operator on Γ_p evaluated on Γ_o , \mathbf{A}_{pk} denotes the discretization of the double layer integral operator on Γ_k evaluated on Γ_p , and \mathbf{B}_{cc} denotes the sub-matrix of \mathbf{A}_{oo} corresponding to the interaction of Γ_c with itself but the diagonal entries are set to zero. The matrix \mathbf{Q}_{orig} is called the *update matrix*. The extended system (2.4.3) is obtained by subtracting the contributions from Γ_c in \mathbf{A}_{oo} and adding the contributions from Γ_p . Upon solving (2.4.3), only σ_k and σ_p are used to evaluate the solution inside of Γ_n . Effectively σ_c is a dummy vector. Details of the derivation of (2.4.3) are provided in [56, 50].

The subscript notation “orig” stands for original, indicating that this update matrix formulation is the original formulation used in [50]. Chapter 4 defines a new formulation for the update matrix \mathbf{Q}_{new} and extended system.

2.4.2 A fast direct solver

When constructing the fast direct solver for the locally perturbed boundary value problem, there are advantages to writing the system in the form of (2.4.3). Since the matrix \mathbf{A}_{oo} is the system resulting from the discretization of the integral equation on the original geometry, we assume that a fast direct solver was already computed for it. Any fast direct solver such as the HBS solver can be used. Additionally, the update matrix \mathbf{Q}_{orig} is low rank. This allows for the inverse of the extended systems to be applied rapidly via a Sherman-Morrison-Woodbury formula

$$\begin{aligned}\sigma_{ext} &= \left(\tilde{\mathbf{A}} + \mathbf{Q}_{\text{orig}}\right)^{-1} \mathbf{g}_{ext} \\ &\approx \left(\tilde{\mathbf{A}} + \mathbf{LR}\right)^{-1} \mathbf{g}_{ext} \\ &\approx \tilde{\mathbf{A}}^{-1} \mathbf{g}_{ext} - \tilde{\mathbf{A}}^{-1} \mathbf{L} \left(\mathbf{I} + \mathbf{R} \tilde{\mathbf{A}}^{-1} \mathbf{L}\right)^{-1} \mathbf{R} \tilde{\mathbf{A}}^{-1} \mathbf{g}_{ext},\end{aligned}\tag{2.4.4}$$

where \mathbf{I} is an identity matrix, and \mathbf{LR} denotes the low rank factorization of the update matrix \mathbf{Q}_{orig} .

The low rank property of the update matrix \mathbf{Q}_{orig} can be observed by noting that the matrices \mathbf{A}_{kc} , \mathbf{A}_{pk} and \mathbf{A}_{op} are low rank. The low-rank approximation $\mathbf{Q}_{\text{orig}} \approx \mathbf{LR}$ can be obtained efficiently via techniques similar to that for compressing off-diagonal blocks in the HBS method. The details can be found in [50].

The solver would also work for the situation where the changes or perturbations are not for the boundary but the discretization of the boundary. One example of this shown in Figure 2.4.2 is solving the problem with a new discretization obtained by locally refining the original one. The Stokes solution scheme proposed in chapter 4 falls in this category and uses this algorithm for handling the refinements added to the boundary wall in the region approached by swimmers.

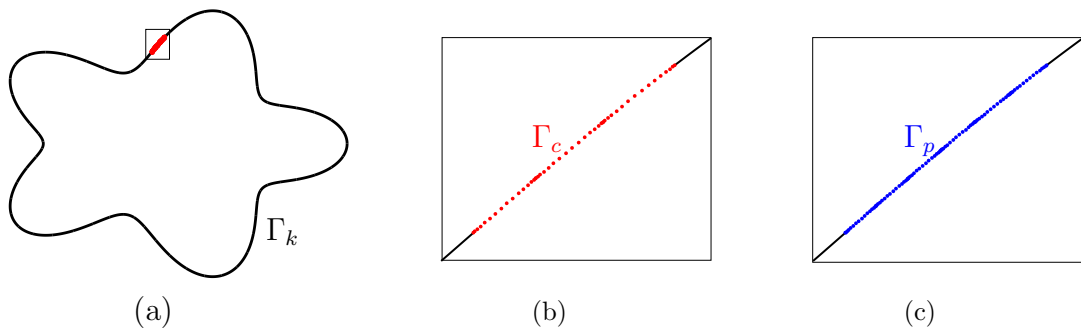


Figure 2.4.2 : (a) The star geometry with the portion of the boundary to be refined boxed. (b) The three Gaussian panels in the boxed region from the original discretization. (c) The six Gaussian panels that replaced the original three panels.

Chapter 3

A fast direct solver for quasi-periodic scattering in layered media

Periodic layered structures are important in designing optical and electromagnetic devices. Some specific examples are solar cells (thin-filmed photovoltaic cells [42, 43] and solar thermal power units [44]), dielectric gratings for high-powered laser [45, 46] and wideband [47] applications. Most of these applications require solving a scattering problem for a large number of incident angles. For example, in many engineering applications, a Bragg diagram created from the solution of 200 boundary value problems is desired. In optimal design setting, a scattering problem is nested inside of an optimization loop. When the geometry and material properties are close to the optimal choice, the changes in the scattering problem are often localized to a few layers. This chapter presents a fast direct solution technique for solving two dimensional wave scattering problems from periodic multilayered structures. When the interface geometries are complex, the dominant term in the computational cost of creating the direct solver scales $O(NI)$ where N is the number discretization points on each interface and I is the number of interfaces. The bulk of the precomputation can be re-used for any choice of incident wave. An added benefit of the presented solver is that building an updated solver for a new structure with a replaced interface or change in material property in a layer is inexpensive compared to building a new fast direct solver from scratch.

3.1 Introduction

This chapter considers the $I + 1$ layered scattering problem defined by

$$\begin{aligned}
 (\Delta + \omega_i^2)u_i(\mathbf{x}) &= 0 & \mathbf{x} \in \Omega_i \\
 u_1 - u_2 &= -u^{\text{inc}}(\mathbf{x}) & \mathbf{x} \in \Gamma_1 \\
 \frac{\partial u_1}{\partial \nu} - \frac{\partial u_2}{\partial \nu} &= -\frac{\partial u^{\text{inc}}}{\partial \nu} & \mathbf{x} \in \Gamma_1 \\
 u_i - u_{i+1} &= 0 & \mathbf{x} \in \Gamma_i, \quad 1 < i < I + 1 \\
 \frac{\partial u_i}{\partial \nu} - \frac{\partial u_{i+1}}{\partial \nu} &= 0 & \mathbf{x} \in \Gamma_i, \quad 1 < i < I + 1
 \end{aligned} \tag{3.1.1}$$

where u_i is the unknown solution in the region $\Omega_i \in \mathbb{R}^2$, the wave number in Ω_i is given by ω_i for $i = 1, \dots, I + 1$, and $\nu(\mathbf{x})$ is the normal vector at \mathbf{x} . The interface Γ_i for $i = 1, \dots, I$ between each layer is periodic with period d . The boundary conditions enforce continuity of the solution and its flux through the interfaces Γ_i . The incident wave u^{inc} is defined by $u^{\text{inc}}(\mathbf{x}) = e^{i\mathbf{k}\cdot\mathbf{x}}$ where the incident vector is $\mathbf{k} = (\omega_1 \cos \theta^{\text{inc}}, \omega_1 \sin \theta^{\text{inc}})$ and the incident angle is $-\pi < \theta^{\text{inc}} < 0$. Figure 3.1.1 illustrates a five layered periodic geometry. The incident wave u^{inc} is *quasi-periodic* up to a phase, i.e. $u^{\text{inc}}(x + d, y) = \alpha u^{\text{inc}}(x, y)$ for $(x, y) \in \mathbb{R}^2$, where α is the Bloch phase defined by

$$\alpha := e^{i\omega_1 d \cos \theta^{\text{inc}}}.$$

In the top and bottom layer, the solution satisfies a radiation condition that is characterized by the uniform convergence of the Rayleigh-Bloch expansions (see section 3.2 of [57].)

This chapter presents a fast direct solver for the multilayered media integral equa-

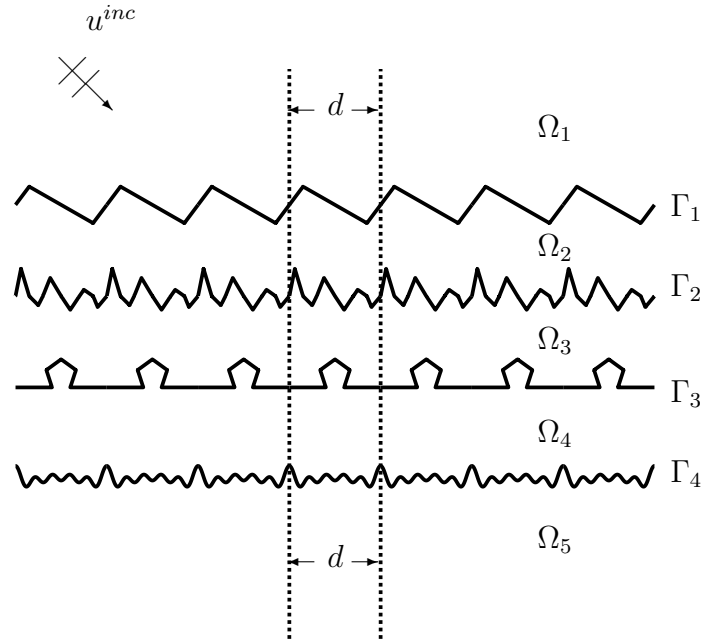


Figure 3.1.1 : A five layered periodic geometry. 7 periods are shown.

tion formulation presented in [48]. This integral formulation is robust even at the so-called Wood's anomalies. The computational cost of the proposed fast direct solver scales linearly with respect to the number of discretization points on the interfaces. When a layer is changed with new material properties and/or a new interface geometry, the cost of updating the direct solver scales linearly with respect to the number of discretization points affected by the modification. For changing an interface, updating the direct solver has a cost that scales linearly with respect to the number of discretization points on the new interface. For updating a wave number in Ω_i , the cost scales linearly with respect to the number of discretization points on the interfaces bounding Ω_i . This makes the solution technique a good option for optimal design and inverse scattering applications.

3.1.1 Related work

Direct discretization of (3.1.1) is possible via finite difference or finite element methods [58] but it faces two challenges, (i) meshing to the interfaces to maintain accuracy and (ii) enforcing the radiation condition. Meshing to the interfaces can be effectively handled using mesh generation software such as GMSH [59]. Techniques such as perfectly matched layers [60] can artificially enforce radiation conditions but can introduce artificial reflections and suffer from high condition numbers. Another challenging aspect of using finite element methods is that there is a loss in accuracy if the points per wavelength remains fixed (the so-called *pollution* effect) [61]. Another alternative method for directly discretizing (3.1.1) is the rigorous-coupled wave analysis (RCWA) or Fourier Modal Method. It is designed for multilayer gratings [62] and it depends on an iterative solve. While a Fourier factorization method [63, 64] can be used to accelerate convergence of an iterative solver, this solution approach is not ideal for problems with many right hand sides that arise in applications. RCWA also has difficulty solving problems with interfaces that cannot be defined as a graph of a function of the x-coordinate such as the “hedgehog” interfaces in Figure 3.4.4(b). Most often RCWA is used for geometries like cones and pillars. There is a concern that the method is too simplified to capture complex structures [65, 66, 67, 68].

When each layer is comprised of constant coefficient (not heterogeneous) medium, it is possible to recast (3.1.1) as a collection of boundary integral equations where the unknowns lie on the interfaces between layers. There has been much work towards the use of boundary integral equations for quasi-periodic scattering problems including [69, 70, 71, 72, 73]. Reviews of boundary integral equation techniques for scattering off a quasi-periodic array of obstacles are presented in [48, 74, 38]. The review in this paper focuses on techniques for layered media. A boundary integral technique utilizing

a fast direct solver for two layered media with periodic structures was presented in [75]. The integral formulation utilized the quasi-periodic Green’s function which is defined as an infinite series. For some choices of boundary data, this series does not converge even though the problem is well posed. An incident angle θ^{inc} that causes the quasi-periodic Green’s function not to converge is called a *Wood’s anomaly*. There have been many techniques suggested to avoid these anomalies (such as [74, 76]). A fast direct solver was constructed for quasi-periodic scattering off an infinite array of scatterers in [38] for the robust integral formulation presented in [74]. The work in this chapter is an extension of that work to multilayered media problems. It builds on the robust integral formulation in [48] though it is likely possible to build similar direct solvers for other formulations that are robust at Wood’s anomalies. The integral formulation in [48] makes use of the free space Helmholtz Green’s function, avoids the infinite sum and uses auxiliary unknowns to enforce periodicity. The radiation condition is satisfied by enforcing continuity of the integral equation solution with the Rayleigh-Bloch expansions.

Recently, [77] replaced the boundary integral formulation in this approach to a technique based on the method of fundamental solutions. This exchanges a second kind integral equation for a formulation that results in a system that is exponentially ill-conditioned.

3.1.2 High level view of the solution technique

Due to the problems associated with the quasi-periodic Green’s function and a desire to exploit the constant coefficient medium, the fast direct solver is built for the robust boundary integral formulation proposed in [48]. Each interface has a boundary integral equation that has “structural” similarities to a boundary integral equation

for scattering off a single closed curve. The structural similarity is that a block matrix in the discretized boundary integral equation is amenable to fast direct solution techniques such as HBS method (section 2.3). Recall from section 1.3 these fast direct solvers utilize the fact that the off-diagonal blocks of the discretized integral equation are low rank to create compressed representations of the matrix and its inverse.

The linear system resulting from the discretization of the integral formulation in [48] is rectangular where the principle sub-block is a block tridiagonal matrix. Each block in this tridiagonal matrix corresponds to a discretized boundary integral operator that (in the low frequency regime) is amenable to compression techniques such as those in fast direct solvers. Utilizing this and separating the matrices that depend on Bloch phase allows for the precomputation of the direct solver to be utilized for all choices of incident angle. The Bloch phase dependence of many of the other block matrices in the rectangular system can be separated out in a similar manner allowing them to be reused for multiple solves. Further acceleration is gained by exploiting the block diagonal or nearly block diagonal sparsity pattern of all the matrices. The combination of all these efforts dramatically reduces the cost of processing the many solves needed in applications.

The fast direct solver presented in this chapter is ideally suited for applications that require many solves per geometry, involve solving problems where there are changes in a subset of the layers (material properties and/or interface geometries), or a combination of many solves per geometry and changes in the geometry. Applications where the solver can be of benefit include optimal design of layered materials and inverse scattering problems where the goal is to recover the thickness and/or the material properties of intermediate layers. While the problems under consideration are acoustic scattering, the solution technique can be extended to *transverse electric*

(TE) and *transverse magnetic* (TM) wave problems.

The direct solver presented in this chapter is built for the robust boundary integral formulation proposed in [48] which enforces continuity of the solution and flux through interfaces. The integral formulation can be extended to problems where there are jumps in the solution and flux as long as these jumps are consistent with the quasi-periodicity conditions.

3.1.3 Outline

The chapter begins by reviewing the integral formulation from [48] in section 3.2. Next, section 3.3 presents the proposed fast direct solver. Numerical results in section 5.4 illustrate the performance of the direct solver. Section 5.5 summarizes and reviews the key features of the presented work.

3.2 Periodizing scheme

This section provides a review of the boundary integral formulation presented in [48]. The necessary integral operators are presented in 3.2.1. Then the full representation is presented in 3.2.2. Finally, the linear system resulting from enforcing continuity and quasi-periodicity of the solution is presented in section 3.2.3. The integral formulation proposed in [48] solves (3.1.1) in an infinite vertical unit strip of width d . Because the solution is known to be quasi-periodic, the solution outside of the unit strip can be found by scaling the solution by the appropriate Bloch phase factor. Let $x = L$ and $x = R$ denote the left and right bounds for the unit strip. The solution technique further partitions space by introducing artificial top and bottom walls to the unit strip at $y = y_U$ and $y = y_D$ respectively. Figure 3.2.1(a) illustrates this partitioning. The box bounded by these artificial boundaries is called the *unit cell*. Inside the unit

cell the solution is represented via an integral formulation. Above and below the unit cell, (i.e. for points in the unit strip where $y > y_U$ and $y < y_D$), the solution is given by Rayleigh-Bloch expansions. Specifically, for $\mathbf{x} = (x, y)$ in the unit strip where $y > y_U$, the solution is given by

$$u(x, y) = \sum_{n \in \mathbb{Z}} a_n^U e^{i\kappa_n x} e^{ik_n^U (y - y_U)} \quad (3.2.1)$$

and, for $\mathbf{x} = (x, y)$ in the unit strip where $y < y_D$, the solution is given by

$$u(x, y) = \sum_{n \in \mathbb{Z}} a_n^D e^{i\kappa_n x} e^{ik_n^D (-y + y_D)} \quad (3.2.2)$$

where $\kappa_n := \omega_1 \cos \theta^{\text{inc}} + \frac{2\pi n}{d}$, $k_n^U = \sqrt{\omega_1^2 - \kappa_n^2}$, $k_n^D = \sqrt{\omega_{I+1}^2 - \kappa_n^2}$ and the sets $\{a_n^U\}$ and $\{a_n^D\}$ are coefficients to be determined. The square root can be either a positive real or positive imaginary number.

3.2.1 Integral operators

This section presents the integral operators needed to represent the solution inside the unit cell.

Let Γ_i for $i = 1, \dots, I$ denote the interfaces inside the unit cell and Ω_i denote the regions between for each layer in the unit cell. Both are numbered from the top to the bottom. Figure 3.2.1(a) illustrates the numbering of the five layered geometry within the unit cell.

Let $G_\omega(\mathbf{x}, \mathbf{y}) = \frac{i}{4} H_0^{(1)}(\omega \|\mathbf{x} - \mathbf{y}\|)$ denote the two dimensional free space Green's function for the Helmholtz equation with wave number ω where H_0^1 is the Hankel function of zeroth order [78].

The standard Helmholtz single and double layer integral operators defined on a

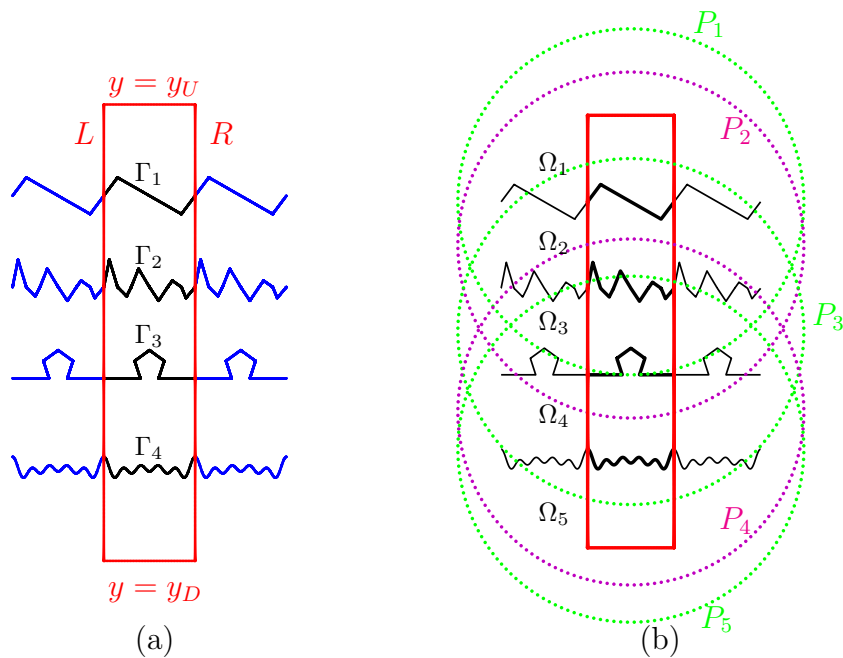


Figure 3.2.1 : This figure illustrates a five layered periodic geometry with artificial walls and proxy circles. Only three periods of the infinite periodic geometry are shown. The period contained within the unit cell is in black while the other two periods are in blue. Figure (a) illustrates the notation for the unit cell with left, right, upper, and lower boundary L , R , U , and D shown in red lines. Figure (b) illustrates the proxy circles P_i for each layer. The color of the proxy circles alternates between green and magenta.

curve W [79] are

$$(\mathcal{S}_W^\omega \rho)(\mathbf{x}) = \int_W G_\omega(\mathbf{x}, \mathbf{y}) \rho(\mathbf{y}) dl(\mathbf{y}) \quad \text{and} \quad (\mathcal{D}_W^\omega \rho)(\mathbf{x}) = \int_W \partial_{\nu_{\mathbf{y}}} G_\omega(\mathbf{x}, \mathbf{y}) \rho(\mathbf{y}) dl(\mathbf{y}),$$

respectively, where $\nu_{\mathbf{y}}$ denotes the normal vector at the point $\mathbf{y} \in W$.

For the periodizing scheme, integral operators involving the unit cell and its neighbors (left and right) are needed.

These operators, denoted with tilde, are defined as follows

$$\begin{aligned} (\tilde{\mathcal{S}}_W^\omega \rho)(\mathbf{x}) &= \sum_{l=-1}^1 \alpha^l \int_W G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}) \\ &= (\mathcal{S}_W^\omega \rho)(\mathbf{x}) + (\mathcal{S}_W^{\omega, pm} \rho)(\mathbf{x}) \end{aligned} \quad (3.2.3)$$

and

$$\begin{aligned} (\tilde{\mathcal{D}}_W^\omega \rho)(\mathbf{x}) &= \sum_{l=-1}^1 \alpha^l \int_W \partial_{\nu_{\mathbf{y}}} G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}) \\ &= (\mathcal{D}_W^\omega \rho)(\mathbf{x}) + (\mathcal{D}_W^{\omega, pm} \rho)(\mathbf{x}) \end{aligned} \quad (3.2.4)$$

where

$$(\mathcal{S}_W^{\omega, pm} \rho)(\mathbf{x}) = \sum_{l=-1, l \neq 0}^1 \alpha^l \int_W G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}) \quad (3.2.5)$$

and

$$(\mathcal{D}_W^{\omega, pm} \rho)(\mathbf{x}) = \sum_{l=-1, l \neq 0}^1 \alpha^l \int_W \partial_{\nu_{\mathbf{y}}} G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}). \quad (3.2.6)$$

The notation pm stands for plus-minus referring to the right and left neighboring copies of the W .

These integral operators are not sufficient to enforce quasi-periodicity. They are missing information from the infinite copies that are “far” from the unit cell. A proxy

basis is used to capture the missing information. For simplicity, consider a layer Ω_l . Let $\{\mathbf{y}_j\}_{j=1}^P$ denote a collection of uniformly distributed points on a circle P_l of radius $2d$ that is centered in Ω_l . The proxy circle needs to be large enough to shield the interface in the unit cell from its far-field copies, which are more than $\frac{3d}{2}$ away from the center of Γ_i in the horizontal direction. It is proved in [80] that larger proxy radius leads to higher order convergence rate with respect to the number of basis functions P . However, the radius cannot be arbitrarily large, as the magnitude of the coefficients grows exponentially with respect to the ratio between the proxy radius and $\frac{3d}{2}$. We set the radius of the proxy circle to be $R_{proxy} \in [\frac{3d}{2}, 2d]$ as in [48]. The elements of the *proxy basis* used to capture the far field information are defined by

$$\phi_j^{\omega_l} = \frac{\partial G_{\omega_l}}{\partial \mathbf{n}_j}(\mathbf{x}, \mathbf{y}_j) + i\omega_l G_{\omega_l}(\mathbf{x}, \mathbf{y}_j) \quad (3.2.7)$$

where \mathbf{n}_j is the normal vector at \mathbf{y}_j on P_l . This choice of basis results in smaller coefficients when compared to using just the single or double layer potential as a basis [48]. If the layer has a high aspect ratio (i.e. much taller than d), the proxy surface should be taken to be an ellipse; see page 8 of [48]. Figure 3.2.1(b) illustrates the proxy circles for a five layered geometry.

The boundary integral equations involve additional integral operators which we define in this section for simplicity of presentation. Specifically, an integral operator defined on an interface W will need to be evaluated at $\mathbf{x} \in V$ where V is an interface (the same or a vertical neighbor of W). For $\mathbf{x} \in V$ where V is an interface, let $(\tilde{S}_{V,W}^\omega \rho)$ denote the evaluation of (3.2.3) at \mathbf{x} , i.e.

$$(\tilde{S}_{V,W}^\omega \rho)(\mathbf{x}) = \sum_{l=-1}^1 \alpha^l \int_W G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}).$$

Likewise, let $(\tilde{D}_{V,W}^\omega \rho)$ denote the evaluation of (3.2.4) at $\mathbf{x} \in V$, i.e.

$$(\tilde{D}_{V,W}^\omega \rho)(\mathbf{x}) = \sum_{l=-1}^1 \alpha^l \int_W \partial_{\nu_{\mathbf{y}}} G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}).$$

The pm notation for the neighbor interactions follows in a similar fashion. For example, the operator $(\tilde{S}_{V,W}^\omega \rho)(\mathbf{x})$ can be written as the following sum

$$(\tilde{S}_{V,W}^\omega \rho)(\mathbf{x}) = (S_{V,W}^\omega \rho)(\mathbf{x}) + (S_{V,W}^{\omega, pm} \rho)(\mathbf{x}),$$

where

$$(S_{V,W}^{\omega, pm} \rho)(\mathbf{x}) = \sum_{l=-1, l \neq 0}^1 \alpha^l \int_W G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}).$$

In order to enforce continuity of the fluxes, the normal derivatives of these integral operators are required. For $\mathbf{x} \in V$ where V is an interface, let $(\tilde{D}_{W,V}^{*,\omega} \rho)$ denote the evaluation of the normal derivative of the single layer operator (3.2.3) at \mathbf{x} , i.e.

$$(\tilde{D}_{W,V}^{*,\omega} \rho)(\mathbf{x}) = \sum_{l=-1}^1 \alpha^l \int_W \partial_{\nu_{\mathbf{x}}} G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y})$$

where $\nu_{\mathbf{x}}$ is the normal vector at $\mathbf{x} \in V$. Similarly, let $(\tilde{T}_{W,V}^\omega \rho)$ denote the evaluation of the normal derivative of the double layer operator (3.2.4) at \mathbf{x} , i.e.

$$(\tilde{T}_{W,V}^\omega \rho)(\mathbf{x}) = \sum_{l=-1}^1 \alpha^l \int_W \partial_{\nu_{\mathbf{x}}} \partial_{\nu_{\mathbf{y}}} G_\omega(\mathbf{x}, \mathbf{y} + l\mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}).$$

3.2.2 Integral formulation

The periodizing scheme within the unit cell is based on a modified version of the combined field boundary integral formulation [81, 82]. Specifically, the solution in

the unit cell is expressed as

$$u_1(\mathbf{x}) = (\tilde{\mathcal{S}}_{\Gamma_1}^{\omega_1} \sigma_1)(\mathbf{x}) + (\tilde{\mathcal{D}}_{\Gamma_1}^{\omega_1} \tau_1)(\mathbf{x}) + \sum_{j=1}^P c_j^1 \phi_j^{\omega_1}(\mathbf{x}) \quad \text{for } \mathbf{x} \in \Omega_1, \quad (3.2.8)$$

$$u_{I+1}(\mathbf{x}) = (\tilde{\mathcal{S}}_{\Gamma_I}^{\omega_{I+1}} \sigma_I)(\mathbf{x}) + (\tilde{\mathcal{D}}_{\Gamma_I}^{\omega_{I+1}} \tau_I)(\mathbf{x}) + \sum_{j=1}^P c_j^{I+1} \phi_j^{\omega_{I+1}}(\mathbf{x}) \quad \text{for } \mathbf{x} \in \Omega_{I+1}, \quad \text{and} \quad (3.2.9)$$

$$u_i(\mathbf{x}) = (\tilde{\mathcal{S}}_{\Gamma_{i-1}}^{\omega_i} \sigma_{i-1})(\mathbf{x}) + (\tilde{\mathcal{D}}_{\Gamma_{i-1}}^{\omega_i} \tau_{i-1})(\mathbf{x}) + (\tilde{\mathcal{S}}_{\Gamma_i}^{\omega_i} \sigma_i)(\mathbf{x}) + (\tilde{\mathcal{D}}_{\Gamma_i}^{\omega_i} \tau_i)(\mathbf{x}) + \sum_{j=1}^P c_j^i \phi_j^{\omega_i}(\mathbf{x}) \quad (3.2.10)$$

for $\mathbf{x} \in \Omega_i$, $2 \leq i \leq I$ where σ_i and τ_i are unknown boundary charge distributions and $\{c_j^i\}_{j=1}^P$ are unknown constants, for $i = 1, \dots, I$.

Enforcing the transmission condition in equation (3.1.1) corresponding to continuity of the solution through the interfaces results in the following integral equations:

$$\begin{aligned} -\tau_1 + (\tilde{D}_{\Gamma_1, \Gamma_1}^{\omega_1} - \tilde{D}_{\Gamma_1, \Gamma_1}^{\omega_2}) \tau_1 + (\tilde{S}_{\Gamma_1, \Gamma_1}^{\omega_1} - \tilde{S}_{\Gamma_1, \Gamma_1}^{\omega_2}) \sigma_1 - \tilde{D}_{\Gamma_1, \Gamma_2}^{\omega_2} \tau_2 - \tilde{S}_{\Gamma_1, \Gamma_2}^{\omega_2} \sigma_2 \\ + \sum_{p=1}^P (c_p^1 \phi_p^{\omega_1} - c_p^2 \phi_p^{\omega_2})|_{\Gamma_1} = -u^{\text{inc}} \quad \text{on } \Gamma_1, \end{aligned} \quad (3.2.11)$$

$$\begin{aligned} -\tau_I + (\tilde{D}_{\Gamma_I, \Gamma_I}^{\omega_I} - \tilde{D}_{\Gamma_I, \Gamma_I}^{\omega_{I+1}}) \tau_I + (\tilde{S}_{\Gamma_I, \Gamma_I}^{\omega_I} - \tilde{S}_{\Gamma_I, \Gamma_I}^{\omega_{I+1}}) \sigma_I - \tilde{D}_{\Gamma_I, \Gamma_{I-1}}^{\omega_{I+1}} \tau_{I-1} - \tilde{S}_{\Gamma_I, \Gamma_{I-1}}^{\omega_{I+1}} \sigma_{I-1} \\ + \sum_{p=1}^P (c_p^I \phi_p^{\omega_I} - c_p^{I+1} \phi_p^{\omega_{I+1}})|_{\Gamma_I} = 0 \quad \text{on } \Gamma_I, \end{aligned} \quad (3.2.12)$$

and

$$\begin{aligned}
& -\tau_i + (\tilde{D}_{\Gamma_i, \Gamma_i}^{\omega_i} - \tilde{D}_{\Gamma_i, \Gamma_i}^{\omega_{i+1}})\tau_i + (\tilde{S}_{\Gamma_i, \Gamma_i}^{\omega_i} - \tilde{S}_{\Gamma_i, \Gamma_i}^{\omega_{i+1}})\sigma_i + \tilde{D}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i}\tau_{i-1} + \tilde{D}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}}\tau_{i+1} + \\
& \tilde{S}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i}\sigma_{i-1} + \tilde{S}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}}\sigma_{i+1} + \sum_{p=1}^P (c_p^i \phi_p^{\omega_i} - c_p^{i+1} \phi_p^{\omega_{i+1}})|_{\Gamma_i} = 0 \text{ on } \Gamma_i \text{ for } 1 < i < I
\end{aligned} \tag{3.2.13}$$

where $\tilde{S}_{\Gamma_{i+1}, \Gamma_i}^{\omega_i}$ denotes the periodized single layer integral operator (3.2.3) defined on Γ_i evaluated on Γ_{i+1} , etc.

Likewise, enforcing the transmission condition in equation (3.1.1) corresponding to continuity of the flux through the interfaces results in the following boundary integral equations:

$$\begin{aligned}
& -\sigma_1 + (\tilde{T}_{\Gamma_1, \Gamma_1}^{\omega_1} - \tilde{T}_{\Gamma_1, \Gamma_1}^{\omega_2})\tau_1 + (\tilde{D}_{\Gamma_1, \Gamma_1}^{*, \omega_1} - \tilde{D}_{\Gamma_1, \Gamma_1}^{*, \omega_2})\sigma_1 - \tilde{T}_{\Gamma_1, \Gamma_2}^{\omega_2}\tau_2 - \tilde{D}_{\Gamma_1, \Gamma_2}^{*, \omega_2}\sigma_2 \\
& + \sum_{p=1}^P \left(c_p^1 \frac{\partial \phi_p^{\omega_1}}{\partial \nu} - c_p^2 \frac{\partial \phi_p^{\omega_2}}{\partial \nu} \right) |_{\Gamma_1} = -\frac{\partial u^{\text{inc}}}{\partial \nu} \text{ on } \Gamma_1,
\end{aligned} \tag{3.2.14}$$

$$\begin{aligned}
& -\sigma_I + (\tilde{T}_{\Gamma_I, \Gamma_I}^{\omega_I} - \tilde{T}_{\Gamma_I, \Gamma_I}^{\omega_{I+1}})\tau_I + (\tilde{D}_{\Gamma_I, \Gamma_I}^{*, \omega_I} - \tilde{D}_{\Gamma_I, \Gamma_I}^{*, \omega_{I+1}})\sigma_I - \tilde{T}_{\Gamma_I, \Gamma_{I-1}}^{\omega_{I+1}}\tau_{I-1} - \tilde{D}_{\Gamma_I, \Gamma_{I-1}}^{*, \omega_{I+1}}\sigma_{I-1} \\
& + \sum_{p=1}^P \left(c_p^I \frac{\partial \phi_p^{\omega_I}}{\partial \nu} - c_p^{I+1} \frac{\partial \phi_p^{\omega_{I+1}}}{\partial \nu} \right) |_{\Gamma_I} = 0 \text{ on } \Gamma_I,
\end{aligned} \tag{3.2.15}$$

and

$$\begin{aligned}
& -\sigma_i + (\tilde{T}_{\Gamma_i, \Gamma_i}^{\omega_i} - \tilde{T}_{\Gamma_i, \Gamma_i}^{\omega_{i+1}})\tau_i + (\tilde{D}_{\Gamma_i, \Gamma_i}^{*, \omega_i} - \tilde{D}_{\Gamma_i, \Gamma_i}^{*, \omega_{i+1}})\sigma_i + \tilde{T}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i}\tau_{i-1} + \tilde{T}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}}\tau_{i+1} \\
& + \tilde{D}_{\Gamma_i, \Gamma_{i-1}}^{*, \omega_i}\sigma_{i-1} + \tilde{D}_{\Gamma_i, \Gamma_{i+1}}^{*, \omega_{i+1}}\sigma_{i+1} + \sum_{p=1}^P \left(c_p^i \frac{\partial \phi_p^{\omega_i}}{\partial \nu} - c_p^{i+1} \frac{\partial \phi_p^{\omega_{i+1}}}{\partial \nu} \right) \Big|_{\Gamma_i} = 0 \text{ on } \Gamma_i \text{ for } 1 < i < I.
\end{aligned} \tag{3.2.16}$$

3.2.3 The linear system

Once the representation of the solution has been determined and the boundary integral equations derived, the unknown densities, periodicity constants c_j^i for the proxy surfaces and the coefficients of the Rayleigh-Bloch expansion need to be approximated. This is done by approximating the boundary integral equations, enforcing the quasi-periodicity of the solution and its flux on the left and right walls, and enforcing the continuity of the solution through the top and bottom of the unit cell.

In this chapter, the boundary integral equations are discretized via a Nyström method but the fast direct solver can be applied to the linear system arising from other discretizations. Let N_l denote the number of discretization points on interface Γ_l . As in [48], the quasi-periodicity is enforced at points that lie on Gaussian panels between each interface on the left and right walls of the unit cell. Let M_w denote the number of points used to enforce periodicity in a layer. (For simplicity of presentation, we assume this number is the same for all the layers.) Lastly, the continuity of the integral representation and the Rayleigh-Bloch expansions is enforced at collection of M uniformly distributed points on the top and bottom of the unit cell. The Rayleigh-Bloch expansions are truncated at $\pm K$.

The rectangular linear system that arises from these choices has the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{Q} & \mathbf{0} \\ \mathbf{Z} & \mathbf{V} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \hat{\sigma} \\ \mathbf{c} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad (3.2.17)$$

where \mathbf{A} is a matrix of size $2N \times 2N$ with $N = \sum_{l=1}^I N_l$, \mathbf{B} is a matrix of size $2N \times P$ with $P = \sum_{l=1}^{I+1} P_l$, \mathbf{C} is a matrix of size $2(I+1)M_w \times 2N$, \mathbf{Q} is a matrix of size $2(I+1)M_w \times P$, \mathbf{Z} is a matrix of size $4M \times 2N$, \mathbf{V} is a matrix of size $4M \times P$, and \mathbf{W} is a matrix of size $4M \times 2(2K+1)$. The first row equation enforces the continuity of the scattered field and its flux through the interfaces. The second row equation enforces the quasi-periodicity of the solution and the flux. The last row equation enforces continuity of the integral representation and the Rayleigh-Bloch expansions.

When the interface geometries are complex, a large number of discretization points N are needed to achieve a desired accuracy. Because the number of discretization points on an interface N_i is significantly larger than M , K , and P in this scenario, the cost of inverting a matrix the size of \mathbf{A} dominates the cost of building a direct solver. For this reason, it is best to build the direct solver in a manner that allows for the bulk of the computational cost associated with matrices of the size $2N \times 2N$ to be reused. We choose to build a fast direct solver for (3.2.17) via the following block

solve:

$$\hat{\sigma} = -\mathbf{A}^{-1} \left([\mathbf{B} \ \mathbf{0}] \begin{bmatrix} \mathbf{c} \\ \mathbf{a} \end{bmatrix} + \mathbf{A}^{-1} \mathbf{f} \right) \quad (3.2.18)$$

$$\begin{bmatrix} \mathbf{c} \\ \mathbf{a} \end{bmatrix} = - \left(\begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{V} & \mathbf{W} \end{bmatrix} - \begin{bmatrix} \mathbf{C} \\ \mathbf{Z} \end{bmatrix} \mathbf{A}^{-1} [\mathbf{B} \ \mathbf{0}] \right)^\dagger \begin{bmatrix} \mathbf{C} \\ \mathbf{Z} \end{bmatrix} \mathbf{A}^{-1} \mathbf{f} \quad (3.2.19)$$

where \dagger denotes the Penrose pseudo-inverse.

Remark 3.2.1 A linear scaling direct solver can be built by processing the block solve in the same order as in [48]; i.e. solving for $[\mathbf{c} \ \mathbf{a}]^T$ first. The matrix that needs to be inverted in order to solve for $\hat{\sigma}$ is an approximation of the quasi-periodic Green's function and thus can be ill-conditioned when the incident angle is a Wood's anomaly.

Each of the matrices in (3.2.17) has a sparsity pattern that can be used to accelerate the block solve. The bulk of the acceleration comes from a fast direct solver for the matrix \mathbf{A} (see section 3.3.1).

The matrix \mathbf{A} is block tridiagonal. The diagonal blocks of \mathbf{A} denoted by \mathbf{A}_{ii} can be written as the sum of two matrices \mathbf{A}_{ii}^s and \mathbf{A}_{ii}^{pm} where \mathbf{A}_{ii}^s corresponds to the integral operator on Γ_i in the unit cell evaluated on Γ_i , i.e.

$$\mathbf{A}_{ii}^s = \begin{bmatrix} -I + D_{\Gamma_i, \Gamma_i}^{\omega_i} - D_{\Gamma_i, \Gamma_i}^{\omega_{i+1}} & S_{\Gamma_i, \Gamma_i}^{\omega_i} - S_{\Gamma_i, \Gamma_i}^{\omega_{i+1}} \\ T_{\Gamma_i, \Gamma_i}^{\omega_i} - T_{\Gamma_i, \Gamma_i}^{\omega_{i+1}} & I + D_{\Gamma_i, \Gamma_i}^{*, \omega_i} - D_{\Gamma_i, \Gamma_i}^{*, \omega_{i+1}} \end{bmatrix},$$

where I denotes the identity operator, and \mathbf{A}_{ii}^{pm} is the contributions from the left and

right neighboring copies, i.e.

$$\mathbf{A}_{ii}^{pm} = \begin{bmatrix} D_{\Gamma_i, \Gamma_i}^{\omega_i, pm} - D_{\Gamma_i, \Gamma_i}^{\omega_{i+1}, pm} & S_{\Gamma_i, \{\Gamma_i\}}^{\omega_i, pm} - S_{\Gamma_i, \Gamma_i}^{\omega_{i+1}, pm} \\ T_{\Gamma_i, \Gamma_i}^{\omega_i, pm} - T_{\Gamma_i, \Gamma_i}^{\omega_{i+1}, pm} & D_{\Gamma_i, \Gamma_i}^{*, \omega_i, pm} - D_{\Gamma_i, \Gamma_i}^{*, \omega_{i+1}, pm} \end{bmatrix},$$

for $i = 1, \dots, I$. The upper diagonal block $\mathbf{A}_{i, i+1}$ corresponds to the integral operators on Γ_{i+1} being evaluated on Γ_i , i.e.

$$mtx \mathbf{A}_{i, i+1} = \begin{bmatrix} -\tilde{D}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}} & -\tilde{S}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}} \\ -\tilde{T}_{\Gamma_i, \Gamma_{i+1}}^{\omega_{i+1}} & -\tilde{D}_{\Gamma_i, \Gamma_{i+1}}^{*, \omega_{i+1}} \end{bmatrix},$$

for $i = 1, \dots, I-1$. The lower diagonal blocks $\mathbf{A}_{i, i-1}$ correspond to the integral operators on Γ_{i-1} being evaluated on Γ_i , i.e.

$$\mathbf{A}_{i, i-1} = \begin{bmatrix} \tilde{D}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i} & \tilde{S}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i} \\ \tilde{T}_{\Gamma_i, \Gamma_{i-1}}^{\omega_i} & \tilde{D}_{\Gamma_i, \Gamma_{i-1}}^{*, \omega_i} \end{bmatrix},$$

for $i = 2, \dots, I$.

The matrix \mathbf{B} is upper block diagonal with block defined by

$$\mathbf{B}_{i, i} = \begin{bmatrix} \phi_1^{\omega_i} |_{\Gamma_i} & \cdots & \phi_P^{\omega_i} |_{\Gamma_i} \\ \frac{\partial \phi_1^{\omega_i}}{\partial \mathbf{n}} |_{\Gamma_i} & \cdots & \frac{\partial \phi_P^{\omega_i}}{\partial \mathbf{n}} |_{\Gamma_i} \end{bmatrix} \text{ and } \mathbf{B}_{i, i+1} = \begin{bmatrix} -\phi_1^{\omega_{i+1}} |_{\Gamma_i} & \cdots & -\phi_P^{\omega_{i+1}} |_{\Gamma_i} \\ -\frac{\partial \phi_1^{\omega_{i+1}}}{\partial \mathbf{n}} |_{\Gamma_i} & \cdots & -\frac{\partial \phi_P^{\omega_{i+1}}}{\partial \mathbf{n}} |_{\Gamma_i} \end{bmatrix} \quad (3.2.20)$$

for $i = 1, \dots, I$. The matrix \mathbf{C} is lower block diagonal with blocks defined by

$$\mathbf{C}_{i, i} = \begin{bmatrix} \alpha^{-2} D_{R_i + \mathbf{d}, \Gamma_i}^{\omega_i} - \alpha D_{L_i - \mathbf{d}, \Gamma_i}^{\omega_i} & \alpha^{-2} S_{R_i + \mathbf{d}, \Gamma_i}^{\omega_i} - \alpha S_{L_i - \mathbf{d}, \Gamma_i}^{\omega_i} \\ \alpha^{-2} T_{R_i + \mathbf{d}, \Gamma_i}^{\omega_i} - \alpha T_{L_i - \mathbf{d}, \Gamma_i}^{\omega_i} & \alpha^{-2} D_{R_i + \mathbf{d}, \Gamma_i}^{*, \omega_i} - \alpha D_{L_i - \mathbf{d}, \Gamma_i}^{*, \omega_i} \end{bmatrix} \text{ and } \quad (3.2.21)$$

$$\mathbf{C}_{i,i-1} = \begin{bmatrix} \alpha^{-2} D_{R_i+\mathbf{d},\Gamma_{i-1}}^{\omega_i} - \alpha D_{L_i-\mathbf{d},\Gamma_{i-1}}^{\omega_i} & \alpha^{-2} S_{R_i+\mathbf{d},\Gamma_{i-1}}^{\omega_i} - \alpha S_{L_i-\mathbf{d},\Gamma_{i-1}}^{\omega_i} \\ \alpha^{-2} T_{R_i+\mathbf{d},\Gamma_{i-1}}^{\omega_i} - \alpha T_{L_i-\mathbf{d},\Gamma_{i-1}}^{\omega_i} & \alpha^{-2} D_{R_i+\mathbf{d},\Gamma_{i-1}}^{*,\omega_i} - \alpha D_{L_i-\mathbf{d},\Gamma_{i-1}}^{*,\omega_i} \end{bmatrix} \quad (3.2.22)$$

for $i = 1, \dots, I$ and $i = 2, \dots, I + 1$, respectively. The matrix \mathbf{Q} is block diagonal with blocks given by

$$\mathbf{Q}_{ii} = \begin{bmatrix} \alpha^{-1} \phi_1^{\omega_i}|_{R_i} - \phi_1^{\omega_i}|_{L_i} & \cdots & \alpha^{-1} \phi_P^{\omega_i}|_{R_i} - \phi_P^{\omega_i}|_{L_i} \\ \alpha^{-1} \frac{\partial \phi_1^{\omega_i}}{\partial \mathbf{n}}|_{R_i} - \frac{\partial \phi_1^{\omega_i}}{\partial \mathbf{n}}|_{L_i} & \cdots & \alpha^{-1} \frac{\partial \phi_P^{\omega_i}}{\partial \mathbf{n}}|_{R_i} - \frac{\partial \phi_P^{\omega_i}}{\partial \mathbf{n}}|_{L_i} \end{bmatrix} \quad (3.2.23)$$

for $i = 1, \dots, I + 1$.

The matrices \mathbf{Z} , \mathbf{V} , and \mathbf{W} are sparse matrices of the form

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_U & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{Z}_D \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_U & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{V}_D \end{bmatrix}, \quad \text{and } \mathbf{W} = \begin{bmatrix} \mathbf{W}_U & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_D \end{bmatrix}$$

where

$$\mathbf{Z}_U = \begin{bmatrix} \tilde{D}_{U,\Gamma_1}^{\omega_1} & \tilde{S}_{U,\Gamma_1}^{\omega_1} \\ \tilde{T}_{U,\Gamma_1}^{\omega_1} & \tilde{D}_{U,\Gamma_1}^{*,\omega_1} \end{bmatrix}, \quad \mathbf{Z}_D = \begin{bmatrix} \tilde{D}_{D,\Gamma_I}^{\omega_{I+1}} & \tilde{S}_{D,\Gamma_I}^{\omega_{I+1}} \\ \tilde{T}_{U,\Gamma_I}^{\omega_{I+1}} & \tilde{D}_{U,\Gamma_I}^{*,\omega_{I+1}} \end{bmatrix}, \quad (3.2.24)$$

$$\mathbf{V}_U = \begin{bmatrix} \phi_1^{\omega_1}|_U & \cdots & \phi_P^{\omega_1}|_U \\ \frac{\phi_1^{\omega_1}}{\partial \nu}|_U & \cdots & \frac{\phi_P^{\omega_1}}{\partial \nu}|_U \end{bmatrix}, \quad \mathbf{V}_D = \begin{bmatrix} \phi_1^{\omega_{I+1}}|_D & \cdots & \phi_P^{\omega_{I+1}}|_D \\ \frac{\phi_1^{\omega_{I+1}}}{\partial \nu}|_D & \cdots & \frac{\phi_P^{\omega_{I+1}}}{\partial \nu}|_D \end{bmatrix}, \quad (3.2.25)$$

$$\mathbf{W}_U = \begin{bmatrix} -e^{i\kappa_{-K}x}|_U & \cdots & -e^{i\kappa_K x}|_U \\ -ik_{-K}^U e^{i\kappa_{-K}x}|_U & \cdots & -ik_K^U e^{i\kappa_K x}|_U \end{bmatrix}, \quad \text{and } \mathbf{W}_D = \begin{bmatrix} -e^{i\kappa_{-K}x}|_D & \cdots & -e^{i\kappa_K x}|_D \\ ik_{-K}^D e^{i\kappa_{-K}x}|_D & \cdots & ik_K^D e^{i\kappa_K x}|_D \end{bmatrix}. \quad (3.2.26)$$

The matrices \mathbf{W}_U and \mathbf{W}_D correspond to the evaluation of the terms in the

Rayleigh-Bloch expansions at points on the top and bottom of the unit cell where continuity of the solution is enforced.

3.3 The fast direct solver

While exploiting the sparsity of the matrices can accelerate the construction of a direct solver, the speed gains are not sufficient for applications when the interface geometries are complex. When the interface geometries are complex, the cost of building a direct solver for the rectangular system is dominated by the cost of inverting \mathbf{A} . The fast direct solver proposed in this section exploits not only the sparsity but also the data sparse nature of the matrix \mathbf{A} .

The foundation of the fast direct solver is a fast inversion technique for \mathbf{A} presented in section 3.3.1. The fast inversion of \mathbf{A} allows for $\hat{\sigma}$ to be computed for a cost that scales linearly with respect to N via equation (3.2.18). Constructing and applying an approximation of the pseudo-inverse of the Schur complement

$$\mathbf{S} = - \left(\begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{v} & \mathbf{W} \end{bmatrix} - \begin{bmatrix} \mathbf{C} \\ \mathbf{z} \end{bmatrix} \mathbf{A}^{-1} [\mathbf{B} \ \mathbf{0}] \right). \quad (3.3.1)$$

is needed to find \mathbf{c} and \mathbf{a} via (3.2.19). The approximate pseudo-inverse is created by first computing an ϵ_{Schur} -truncated singular value decomposition (SVD) and then applying the pseudo-inverse of this factorization.

Definition 3.3.1 Let $\mathbf{U}\mathbf{\Sigma}\mathbf{T}^*$ be the SVD of the Schur complement matrix \mathbf{S} of size $(2(I+1)M_w + 4M) \times (P + 2(2K+1))$ where $\mathbf{\Sigma}$ is a diagonal rectangular matrix with entries of the singular values of \mathbf{S} and matrices \mathbf{U} and \mathbf{T} are unitary matrices of size $(2(I+1)M_w + 4M) \times (2(I+1)M_w + 4M)$ and $(P + 2(2K+1)) \times (P + 2(2K+1))$,

respectively. Then the ϵ_{Schur} -truncated SVD is

$$\hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{T}}^*$$

where $\hat{\mathbf{\Sigma}}$ is a diagonal square matrix of size $l \times l$ where l is the number of singular values of \mathbf{S} that are larger than ϵ_{Schur} , $\hat{\mathbf{U}}$ is an $(2(I+1)M_w + 4M) \times l$ matrix and $\hat{\mathbf{T}}$ is an $(P + 2(2K + 1)) \times l$ matrix.

In practice, we found $\epsilon_{\text{Schur}} = 10^{-13}$ is a good choice when the desired accuracy for the solution is 10^{-10} .

Then \mathbf{c} and \mathbf{a} can be approximated by

$$\begin{bmatrix} \mathbf{c} \\ \mathbf{a} \end{bmatrix} \approx \hat{\mathbf{T}}\hat{\mathbf{\Sigma}}^{-1}\hat{\mathbf{U}}^* \begin{bmatrix} \mathbf{C} \\ \mathbf{Z} \end{bmatrix} \mathbf{A}^{-1}\mathbf{f}.$$

The most efficient way to find \mathbf{c} and \mathbf{a} is to apply the matrices from right to left in this equation meaning that the vectors are found via a collection of matrix vector multiplies.

Remark 3.3.1 The cost of constructing the truncated SVD for \mathbf{S} scales cubically with respect to the number of interfaces I , if done via dense linear algebra, but is constant with respect to the number of points per interface.

Combining the fact that many of the matrices (less scalar factors) in (3.2.17) can be re-used for multiple incident angles (see section 3.3.3) and the fast direct solver for \mathbf{A} results in a fast direct solver that is ideal for problems where many solves are required. An additional key feature of the fast direct solver is that the bulk of the precomputation can be re-used if an interface Γ_j or a wave speed ω_j is changed (see

Section 3.3.4).

3.3.1 Fast inversion of \mathbf{A}

The key to building the fast direct solver for the block system (3.2.17) is having a fast way of inverting \mathbf{A} . This technique is designed to make solves for different Bloch phases as efficient as possible.

The solver considers the matrix \mathbf{A} written as the sum of two matrices

$$\begin{aligned}
 \mathbf{A} = & \underbrace{\begin{bmatrix} \mathbf{A}_{11}^s & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^s & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{I-1,I-1}^s & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{II}^s \end{bmatrix}}_{\mathbf{A}_0} \\
 + & \underbrace{\begin{bmatrix} \mathbf{A}_{11}^{pm} & \mathbf{A}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{A}_{22}^{pm} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{I-1,I-2} & \mathbf{A}_{I-1,I-1}^{pm} & \mathbf{A}_{I-1,I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{I,I-1} & \mathbf{A}_{II}^{pm} \end{bmatrix}}_{\hat{\mathbf{A}}}
 \end{aligned} \tag{3.3.2}$$

where the block diagonal matrix \mathbf{A}_0 contains all the self-interaction matrices and the block tridiagonal matrix $\hat{\mathbf{A}}$ contains diagonal blocks corresponding to the interaction of an interface with its left and right neighbors and off-diagonal blocks corresponding to the interactions between the interfaces directly above and below each other. Since the submatrices in $\hat{\mathbf{A}}$ correspond to “far” interactions, they are numerically low rank.

Let \mathbf{LR} denote the low rank factorization of $\hat{\mathbf{A}}$ where \mathbf{L} and \mathbf{R}^T are $2N \times k_{\text{tot}}$ matrices and k_{tot} is the numerical rank of $\hat{\mathbf{A}}$. Section 3.3.2 presents a technique for constructing this factorization. Then \mathbf{A} can be approximated by

$$\mathbf{A} \approx \mathbf{A}_0 + \mathbf{LR}.$$

The advantage of this representation is that the factors \mathbf{L} and \mathbf{R} can be computed in a way that is independent of Bloch phase as presented in section 3.3.2. Additionally, the inverse can be formulated via a Woodbury formula [83]

$$\mathbf{A}^{-1} \approx (\mathbf{A}_0 + \mathbf{LR})^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{L}(\mathbf{I} + \mathbf{RA}_0^{-1}\mathbf{L})^{-1}\mathbf{RA}_0^{-1}. \quad (3.3.3)$$

Not only is the matrix \mathbf{A}_0 block diagonal but each of the diagonal blocks is amenable to a fast direct solver such as the HBS method in section 2.3. Thus an approximate inverse of \mathbf{A}_0 can be constructed and applied for a cost that scales linearly with respect to the number of discretization points on the interfaces. This computation is independent of Bloch phase. The condition number of $(\mathbf{I} + \mathbf{RA}_0^{-1}\mathbf{L})$ is bounded above by the product of the condition number of \mathbf{A}_0 and $(\mathbf{A}_0 + \mathbf{LR})$ [84]. Since both \mathbf{A} and \mathbf{A}_0 result from the discretization of second kind boundary integral equations, they are well-conditioned. Thus applying the Woodbury formula is numerically stable.

It is never necessary to construct the approximation of \mathbf{A}^{-1} . It is only necessary to have a fast algorithm for applying it to a vector $\mathbf{f} \in \mathbb{C}^{2N \times 1}$, i.e. a fast algorithm is needed for evaluating

$$\mathbf{A}^{-1}\mathbf{f} \approx \mathbf{A}_0^{-1}\mathbf{f} - \mathbf{A}_0^{-1}\mathbf{L}(\mathbf{I} + \mathbf{RA}_0^{-1}\mathbf{L})^{-1}\mathbf{RA}_0^{-1}\mathbf{f}. \quad (3.3.4)$$

The fast direct solver for \mathbf{A}_0 and the block structure of the matrices \mathbf{L} and \mathbf{R} allow for $\mathbf{A}_0^{-1}\mathbf{L}$ and $\mathbf{R}\mathbf{A}_0^{-1}\mathbf{f}$ to be evaluated for a cost that scales linearly with N . Thanks to the sparsity pattern of the matrices, the intermediate matrix $\mathbf{S}_2 = \mathbf{I} + \mathbf{R}\mathbf{A}_0^{-1}\mathbf{L}$ of size $k_{\text{tot}} \times k_{\text{tot}}$ that needs to be inverted is block tridiagonal. Appendix A.1 reports on the construction of \mathbf{S}_2 . Since k_{tot} is much smaller than N in practice, the inverse of \mathbf{S}_2 can be applied rapidly using a block variant of the Thomas algorithm. This computation needs to be done for each new Bloch phase since \mathbf{L} and \mathbf{R} are dependent on Bloch phase.

Remark 3.3.2 To achieve nearly optimal ranks in the construction of the fast direct solver, it is advantageous to reorder the matrices in \mathbf{A} according to the physical location of the unknowns. For example, if there are N_1 discretization points on Γ_1 , the unknowns are $\sigma_{1,1}, \dots, \sigma_{1,N_1}$ and $\tau_{1,1}, \dots, \tau_{1,N_1}$, etc. Then the matrices should be ordered so $\hat{\sigma}$ is as follows

$$\hat{\sigma}^T = [\sigma_{1,1}, \tau_{1,1}, \dots, \sigma_{1,N_1}, \tau_{1,N_1}, \dots, \sigma_{I,1}, \tau_{I,1}, \dots, \sigma_{I,N_I}, \tau_{I,N_I}].$$

3.3.2 Low rank factorization of $\hat{\mathbf{A}}$

The technique for creating the low rank factorizations of the blocks in $\hat{\mathbf{A}}$ is slightly different depending on whether or not the block is on the diagonal. This section begins by presenting the technique for creating low rank factorizations of the diagonal blocks. Then the technique for creating the low rank factorizations of the off-diagonal blocks is presented.

Recall the diagonal blocks of $\hat{\mathbf{A}}$ are \mathbf{A}_{ii}^{pm} corresponding to the discretized version

of

$$(\mathcal{S}_{\Gamma_i, \Gamma_i}^{\omega, pm} \rho)(\mathbf{x}) = \alpha \underbrace{\int_{\Gamma_i} G_\omega(\mathbf{x}, \mathbf{y} + \mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y})}_{\text{right copy: } (\mathcal{S}_{\Gamma_i, \Gamma_i}^{\omega, p} \rho)(\mathbf{x})} + \alpha^{-1} \underbrace{\int_{\Gamma_i} G_\omega(\mathbf{x}, \mathbf{y} - \mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y})}_{\text{left copy: } (\mathcal{S}_{\Gamma_i, \Gamma_i}^{\omega, m} \rho)(\mathbf{x})}.$$

The matrix \mathbf{A}_{ii}^{pm} can be written as the sum of two matrices that are independent of Bloch phase; $\mathbf{A}_{ii}^{pm} = \alpha \mathbf{A}_{ii}^p + \alpha^{-1} \mathbf{A}_{ii}^m$. Thus by creating low rank factorizations of \mathbf{A}_{ii}^p and \mathbf{A}_{ii}^m independently, the factorizations can be used for any Bloch phase α . Let $\mathbf{L}_i^p \mathbf{R}_i^p$ and $\mathbf{L}_i^m \mathbf{R}_i^m$ denote the low rank approximations of \mathbf{A}_{ii}^p and \mathbf{A}_{ii}^m respectively. These two approximations are combined to create a low rank approximation of \mathbf{A}_{ii}^{pm} as follows:

$$\mathbf{A}_{ii}^{pm} \approx \underbrace{[\mathbf{L}_i^p, \mathbf{L}_i^m]}_{\mathbf{L}_{ii}^{pm}} \underbrace{\begin{bmatrix} \alpha \mathbf{R}_i^p \\ \alpha^{-1} \mathbf{R}_i^m \end{bmatrix}}_{\mathbf{R}_{ii}^{pm}}$$

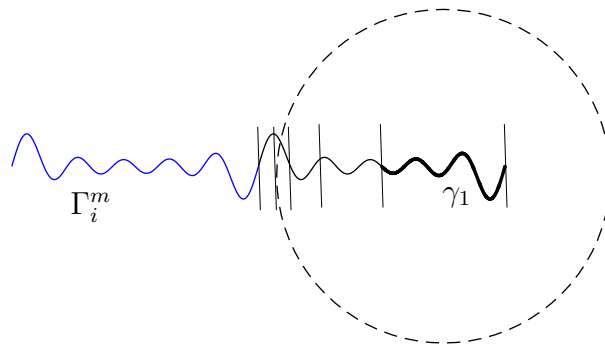
The technique used to create the low rank factorizations is similar to the one used in [85]. The new technique has an extra step to keep the rank k_{tot} small.

For brevity, this chapter only presents the technique for compressing the interaction with the left neighbor (i.e. computing the low rank factorization of \mathbf{A}_{ii}^m). The technique for compressing the interaction with the right neighbor follows directly.

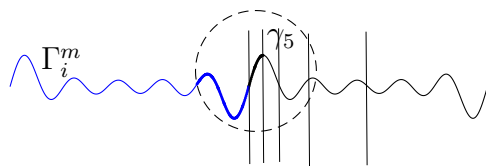
We choose to build the factorization via the interpolatory decomposition [86, 87] defined as in section 2.1.

Creating the low rank factorization of \mathbf{A}_{ii}^m by directly plugging it into the interpolatory decomposition has a computational cost of $O(N_i^2 k_i)$ where k_i is the numerical rank of \mathbf{A}_{ii}^m . This would result in a solution technique that has a computational cost that scales quadratically, *not linearly*, with respect to N . To achieve the linear computational complexity, we utilize potential theory.

Recall Γ_i denotes the part of the i^{th} interface in the unit cell. Let Γ_i^m denote the part of the i^{th} interface in the left neighboring cell. First Γ_i is partitioned into a collection of S segments γ_j via dyadic refinement where the segments get smaller as they approach Γ_i^m so that $\Gamma_i = \cup_{j=1}^S \gamma_j$. Figure 3.3.1 illustrates a partitioning when compressing the interaction of Γ_i with Γ_i^m . The refinement is stopped when the segment closest to Γ_i^m has less than n_{\max} points on it. Typically, $n_{\max} = 45$ is a good choice.



(a)



(b)

Figure 3.3.1 : Illustration of the dyadic refinement partitioning of Γ_i with 5 levels of refinement and geometries for compressing \mathbf{A}_{ii}^m . (a) Illustration of the proxy surface (dashed circle) used to compress neighbor interactions when γ_l is far from Γ_i^m . (c) Illustration of the proxy surface (dashed circle) and near points (bold blue curve on Γ_i^m) when γ_l is touching Γ_i^m .

For each segment γ_j not touching Γ_i^m , consider a circle concentric with the bounding box containing γ_j with a radius slightly less than the distance from the center of the bounding box to Γ_i^m . Figure 3.3.1(a) illustrates the proxy surface for γ_1 when there are 5 levels of dyadic refinement toward Γ_i^m . From potential theory, we know that any field generated by sources outside of this circle can be approximated to high accuracy by placing enough equivalent charges on the circle. In practice, it is enough to place a small number of *proxy points* evenly on the circle. Let n_{proxy} denote the number of proxy points on the circle. For problems where the direct solver scales linearly, n_{proxy} is small and chosen to be a constant independent of ω_i . For the experiments in this chapter, it is sufficient to set $n_{\text{proxy}} = 80$. Let n_j denote the number of points on γ_j . An interpolatory decomposition can be constructed for the matrix $\mathbf{A}^{\text{proxy}}$ capturing the interaction between γ_j and the proxy points. The result is an index vector J_j and an interpolation matrix \mathbf{P}_j of size $n_j \times k_j$ where k_j is the numerical rank of $\mathbf{A}^{\text{proxy}}$. For γ_S (the segment touching Γ_i^m), n_{proxy} proxy points are placed uniformly on a circle of radius 1.75 times larger than the radius of the smallest circle containing all the points on γ_S . All the points on Γ_i^m inside the circle are labeled *near points* and indexed I_{near} . Figure 3.3.1(b) illustrates the proxy circle and near points for γ_5 when there are 5 levels of dyadic refinement toward Γ_i^m . An interpolatory decomposition is then performed on $[\mathbf{A}_{ii}^m(\gamma_S, I_{\text{near}}) \mid \mathbf{A}^{\text{proxy}}]$. The result is an index vector J_S and an interpolation matrix \mathbf{P}_S of size $n_S \times k_S$.

The low rank factorization of the matrix \mathbf{A}_{ii}^m can be constructed with the result of this compression procedure. Let $J = [J_1(1 : k_1), \dots, J_S(1 : k_S)]$ denote an index vector consisting of the index vectors for each segment. Then \mathbf{L}_i^m is a block diagonal matrix with block entries \mathbf{P}_j for $J = 1, \dots, S$ and $\mathbf{R}_i^m = \mathbf{A}_{ii}^m(J, :)$. The points on Γ_i corresponding to the index vector J are called the *skeleton points*.

The rank of this factorization is far from optimal and will result in an excessively large constant prefactor in the application of the Woodbury formula (3.3.3). A re-compression step is necessary to resolve this problem. Let k_{orig} denote the rank of the original approximate factorization, i.e. the length of J . If k_{orig} is small enough, applying the interpolatory decomposition to $\mathbf{A}_{ii}^m(J, :)$ can be done efficiently resulting in an index vector J_{up} and an interpolation matrix \mathbf{P}_{up} of size $k_{\text{orig}} \times k_{\text{up}}$. Let $\mathbf{L}_{\text{up}} = \mathbf{P}_{\text{up}}$. Otherwise, the interpolatory decomposition can be applied to the submatrices corresponding to a lump of the segments at a time. For example, suppose S is even, then the segments can be bunched two at a time. The interpolatory decomposition can be applied to $\mathbf{A}_{ii}^m([J_j(1 : k_j), J_{j+1}(1 : k_{j+1})], :)$ for odd values of j . The resulting interpolation matrices are the block diagonal entries for the block diagonal matrix \mathbf{L}_{up} . The corresponding index vector J_{up} is formed in a similar manner to the vector J . Finally the low rank factorization of \mathbf{A}_{ii}^m can be formed by multiplying \mathbf{L}_i^m from before by \mathbf{L}_{up} and using the updated skeleton of $J(J_{\text{up}})$. In other words, $\mathbf{L}_i^m = \mathbf{L}_i^m \mathbf{L}_{\text{up}}$ and $\mathbf{R}_i^m = \mathbf{A}_{ii}^m(J(J_{\text{up}}), :)$.

The technique for constructing the low rank factorization of the off-diagonal blocks of $\hat{\mathbf{A}}$ is similar. Recall that each off-diagonal block \mathbf{A}_{ij} , for $i \neq j$, corresponds to the discretization of the following integral operator where $\mathbf{x} \in \Gamma_i$:

$$\begin{aligned} (\tilde{\mathcal{S}}_{\Gamma_i, \{\Gamma_j\}}^\omega \rho)(\mathbf{x}) &= \int_{\Gamma_j} G_\omega(\mathbf{x}, \mathbf{y}) \rho(\mathbf{y}) dl(\mathbf{y}) + \alpha \int_{\Gamma_j} G_\omega(\mathbf{x}, \mathbf{y} + \mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}) \\ &\quad + \alpha^{-1} \int_{\Gamma_j} G_\omega(\mathbf{x}, \mathbf{y} - \mathbf{d}) \rho(\mathbf{y}) dl(\mathbf{y}). \end{aligned}$$

It is natural to write \mathbf{A}_{ij} as the summation of three parts,

$$\mathbf{A}_{ij} = \mathbf{A}_{0,ij} + \alpha \mathbf{A}_{ij}^p + \alpha^{-1} \mathbf{A}_{ij}^m,$$

where $\mathbf{A}_{0,ij}$, \mathbf{A}_{ij}^p , and \mathbf{A}_{ij}^m are the discrete approximations of the corresponding integral operators.

While the actual matrix entries of \mathbf{A}_{ij} are dependent on α , the low rank factorization can be computed independent of α since the matrices need only be scaled by α . As with the diagonal blocks, building the factorization of \mathbf{A}_{ij} directly is computationally prohibitive. (The computational cost of the direct factorization is $O(N_i N_j k_{ij})$ where k_{ij} is the numerical rank of \mathbf{A}_{ij} .) Potential theory is again utilized to decrease the computational cost. Consider an ellipse horizontally large enough to enclose Γ_i and vertically shields Γ_i from its top and bottom neighbor interface. A collection of n_{proxy} equivalent charges are evenly distributed on the ellipse in parameter space. Figure 3.3.2 illustrates a proxy surface used for compressing $\mathbf{A}_{i,i+1}$. The interpolatory decomposition is applied to the matrix characterizing the interactions between the points on Γ_i and the proxy surface, $\mathbf{A}^{\text{proxy}}$. The index vector J_i and the $N_i \times k_{\text{proxy}}$ interpolation matrix $\mathbf{P}_{\text{orig},ij}$ are returned. Let $J_{\text{orig}} = J_i(1 : k_{\text{proxy}})$.

As with the diagonal block factorization, k_{proxy} is far from the optimal rank. To reduce the rank, we apply the interpolatory decomposition to $[\mathbf{A}_{0,ij} | \mathbf{A}_{ij}^p | \mathbf{A}_{ij}^m](J_{\text{orig}}, \cdot)$. A $k_{\text{proxy}} \times k_{\text{new}}$ interpolation matrix $\mathbf{P}_{\text{new},ij}$ and an index vector J_{new} are returned. Then the low rank factorization is complete. One factor can be used for all Bloch phases; $\mathbf{L}_{ij} = \mathbf{P}_{\text{orig},ij} \mathbf{P}_{\text{new},ij}$. The other factor is simply a matrix evaluation; $\mathbf{R}_{ij} = \mathbf{A}_{ij}(J_{ij}, \cdot)$ where $J_{ij} = J_{\text{orig}}(J_{\text{new}})$. It is important to note that the matrices $\mathbf{A}_{0,ij}(J_{ij}, \cdot)$, $\mathbf{A}_{ij}^p(J_{ij}, \cdot)$ and $\mathbf{A}_{ij}^m(J_{ij}, \cdot)$ are computed once as they are independent of Bloch phase. Thus \mathbf{R}_{ij} is formed simply by matrix addition for each new Bloch phase.

Remark 3.3.3 The rank of the factorizations of \mathbf{A}_{ij} will depend on the distance between the interfaces. If the interfaces are space filling, the interaction between the interfaces is not low rank.

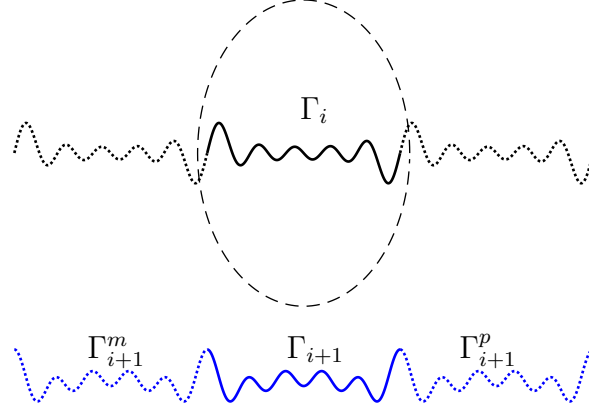


Figure 3.3.2 : Illustration of the proxy surface for compressing $\mathbf{A}_{i,i+1}$.

3.3.3 The Bloch phase and incident angle dependence

Beyond the matrix \mathbf{A} and exploiting the sparsity of the other matrices in (3.2.17), additional acceleration can be gained for problems where the solution is desired for multiple incident angles.

The matrix \mathbf{B} has block entries (3.2.20) that are independent of Bloch phase and thus only needs to be computed once. This is also the case for \mathbf{V} . The non-zero block matrices in \mathbf{C} in (3.2.21) and (3.2.22) are dependent on α but only as a constant multiple. Thus the submatrices of \mathbf{C} can be precomputed and used for all incident angles. The same statement is true for \mathbf{Q} and \mathbf{Z} .

The only matrix that has entries dependent on incident angle is \mathbf{W} . In fact, the matrix \mathbf{W} is only dependent on the Bloch phase α . Recall that the Bloch phase is defined as $\alpha = e^{id\omega_1 \cos \theta^{\text{inc}}}$. This means that for all incident angles that share a Bloch phase, there exist a representative angle $\hat{\theta}$ such that $\omega_1 \cos \theta^{\text{inc}} = \hat{\theta} + \frac{2\pi m}{d}$ for $m \in \mathbb{Z}$.

Since the entries of \mathbf{W} involve

$$e^{i\kappa_j x} = e^{i(\omega_1 \cos \theta^{\text{inc}} + \frac{2\pi j}{d})x}$$

for $j = -K, \dots, K$, for angles with a shared Bloch phase, the entries of \mathbf{W} are the same up to a shift in the index. For example, suppose that we know that 12 incident angles $\{\theta_1^{\text{inc}}, \dots, \theta_{12}^{\text{inc}}\}$ share a Bloch phase and $\omega_1 \cos \theta_j^{\text{inc}} = \hat{\theta} + \frac{2\pi(j-1)}{d}$ for $j = 1, \dots, 12$. The matrix \mathbf{W} is constructed so that it has entries with κ_j indexed from $-K$ to $K + 12$. This allows the singular value decomposition of \mathbf{S} to be used for all angles that share a Bloch phase which results in substantial savings. To evaluate the solution using the resulting coefficients for the Rayleigh-Bloch expansion above or below the unit cell, it is only necessary to use the terms that correspond to $-K, \dots, K$ for that incident angle.

3.3.4 Extensions

Many applications consider boundary value problem (3.1.1) for a collection of geometries where the variation is in a single interface or wave number in a layer. The proposed direct solver can efficiently update an existing fast direct solver for these localized changes in the geometry.

For example, if a user wants to replace Γ_i , only the matrices corresponding to that interface need to be recomputed. This includes: the parts of the fast direct solver for \mathbf{A} corresponding to that block row and column, the corresponding block columns of \mathbf{C} and the corresponding block rows of \mathbf{B} . If the replaced layer is either the top or the bottom, then sub-blocks in \mathbf{V} and \mathbf{Z} need to be updated as well. Independent of the interface changed, the cost of creating a new direct solver is linear with respect

to the number of discretization points on the new interface.

If a user wants to change the wave number ω_i in Ω_i where $1 < i < I$, there are two interfaces affected Γ_i and Γ_{i+1} . The corresponding block rows and columns of the fast direct solver of \mathbf{A} need to be recomputed. In addition to updating those matrices, the corresponding blocks in \mathbf{B} , \mathbf{C} and \mathbf{Q} need to be updated. If the wave number is changed in the top or bottom layer, then the corresponding blocks in \mathbf{Z} and \mathbf{V} need to be updated as well. The cost of the update is $O(N_i + N_{i+1})$, which is linear with respect to the number of discretization points on the affected interfaces.

3.4 Numerical examples

This section illustrates the performance of the fast direct solver for several geometries with up to 11 layers, though the solution technique can be applied to geometries of arbitrary number of layers. Section 3.4.1 demonstrates the scaling of the fast direct solver for 3-layer and 9-layer geometries where the wave number alternates between 10 and $10\sqrt{2}$ in the layers. The ability for the solution technique to efficiently solve (3.1.1) for hundreds of incident waves is demonstrated in section 3.4.2. Section 3.4.3 illustrates the performance of the solution technique when the problem has: an interface geometry that is changed or a wave number that is changed in one of the layers.

All the geometries considered in this section have period fixed of $d = 1$. The vertical separation between the neighbor interfaces is roughly 1 for the geometries considered in section 3.4.1 and roughly 1.5 for the rest of the experiments. The interfaces are discretized via the Nyström method with a 16-point composite Gaussian quadrature. The diagonal blocks require specialized quadrature to handle the weakly singular kernels. For the experiments presented in this section generalized Gaussian

[88] quadrature was utilized, but the fast direct solver is compatible with other specialized quadrature including Alpert [89], Helsing [90], Kapur–Rokhlin [91], and QBX [92]. The geometries under consideration involve both smooth interfaces and interfaces that have corners. In order to achieve high accuracy without over-discretizing, each corner is discretized with five levels of dyadic refinement. Additionally, the integral operators are discretized in \mathcal{L}^2 [93]. The artificial separation walls and proxy circles are discretized with parameter choices similar to those in [48]. Specifically, the left and right (vertical) artificial walls for each layer are discretized by $M_w = 120$ nodes Gauss-Legendre quadrature, the (horizontal) upper and lower walls of the unit cell are sampled at $M = 60$ equi-spaced nodes and $P = 160$ equi-spaced nodes are chosen on the proxy circle for each layer. For the wave numbers under consideration in these experiments, it is sufficient to truncate the Rayleigh-Bloch expansions at $K = 20$.

For all experiments, a HBS fast direct solver with tolerance $\epsilon = 10^{-12}$ was used to construct the approximation of \mathbf{A}_0^{-1} in (3.3.3). The tolerance for all of the low rank factorizations is set to 10^{-12} . The singular value decomposition of \mathbf{S} was truncated at $\epsilon_{\text{Schur}} = 10^{-13}$.

All experiments were run on a dual 2.3 GHz Intel Xeon Processor E5-2695 v3 desktop workstation with 256 GB of RAM. The code was implemented in MATLAB, apart from the interpolatory decomposition, which uses Fortran.

The computational cost of the direct solution technique is broken into four parts:

- Precomputation I: This consists of all computations for the *fast linear algebra* that are independent of Bloch phase. This includes the fast application of \mathbf{A}_0^{-1} , and the low rank factors \mathbf{L}_{ij} and \mathbf{R}_{ij} needed to make \mathbf{L} and \mathbf{R} as presented in section 3.3.1). The computational cost of this step is $O(N)$ where $N = \sum_{l=1}^I N_l$,

and N_l denotes the number of discretization points on interface l .

- Precomputation II: This consists of the remainder of the precomputation that is independent of Bloch phase as presented in section 3.3.3. The computational cost of this step is $O(N)$.
- Precomputation III: This consists of all the precomputation that can be used for incident angles that share a Bloch phase α , including scaling matrices by α , constructing the matrix \mathbf{W} as explained in section 3.3.3, constructing the fast apply of \mathbf{A}^{-1} , evaluating the Schur complement matrix \mathbf{S} (3.3.1), and computing the ϵ_{Schur} SVD of \mathbf{S} . The computational cost of this step is $O(N)$. For a fixed number of discretization points on an interface, the computational cost is $O(I^3)$.
- Solve: This consists of the application of the precomputed solver to the right hand side of (3.2.17) via (3.2.18) and (3.2.19). The computational cost of the solve is $O(N)$. For a fixed number of discretization points on an interface, the computational cost is $O(I^2)$.

The error is approximated via an flux error estimate as in [48] which measures conservation of energy. This has been demonstrated to agree with the relative error at any point in the domain.

3.4.1 Scaling experiment

This section illustrates the scaling of the fast direct solver for problems with 3- and 9-layers corresponding to two and eight interface geometries. The wave number in the layers remains fixed (alternating between 10 and $10\sqrt{2}$) while the number of discretization points per layer increases. The geometry consists of alternating two

interface geometries $\gamma_1 = (x_1(t), y_1(t))$ and $\gamma_2 = (x_2(t), y_2(t))$, which are defined as

$$\gamma_1 : \begin{cases} x_1(t) = t - 0.5 \\ y_1(t) = \frac{1}{60} \sum_{j=1}^{30} a_j \sin(2\pi jt) \end{cases} \quad \text{and} \quad \gamma_2 : \begin{cases} x_2(t) = t - 0.5 \\ y_2(t) = \frac{1}{60} \sum_{j=1}^{30} b_j \cos(2\pi jt) \end{cases} \quad (3.4.1)$$

for $t \in [0, 1]$, where $\{a_j\}_{j=1}^{30}$ and $\{b_j\}_{j=1}^{30}$ are random numbers in $[0, 1)$ sorted in descending order. Figure 3.4.1 illustrates the two interface geometries. In each experiment, γ_1 and γ_2 are discretized with the same number of points N_i . The run time in seconds and flux error estimates are reported in Table 3.1.

Each part of the solution technique scales linearly with respect to N_i . The factor of four increase in time for Precomputation I is expected since the cost scales linearly with the number of interfaces. Precomputation II should scale linearly with the number of layers and this is observed with a factor three increase in the timings for this portion of the solver. Precomputation III is expected to observe a factor nine increase in the computational cost as this step scales cubically with the number of layers. A factor of six is observed. This is likely because the problems under consideration are sub-asymptotic in the number of layers. The same statement is true for the solve step. As expected the precomputation dominates the cost of the solver for both experiments. Precomputation parts I, II and III account for approximately 90%, 3%, and 7% of the precomputation time, respectively. Thus the Bloch phase independent parts of the direct solver dominate the computational cost.

3.4.2 Sweep over multiple incident angles

Many applications require solving (3.1.1) for many incident angles (as discussed in section 3.1). In this setting, Precomputation I and II only need to be done once.

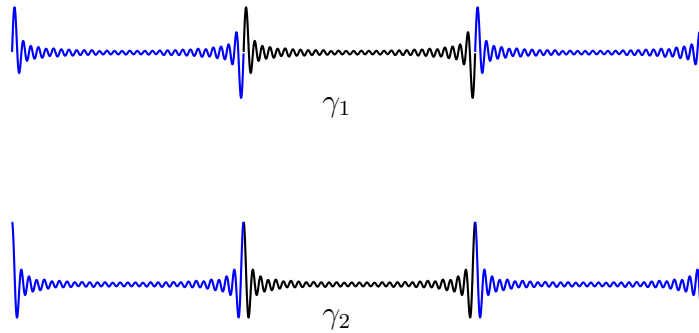


Figure 3.4.1 : Three periods of the interface geometries γ_1 and γ_2 as defined in equation (3.4.1).

Precomputation III can be utilized for all incident angles that share a Bloch phase α . This section demonstrates the efficiency of the fast direct solver for handling scattering problems involving many incident angles. Specifically, we consider the geometry in Figure 3.4.2 which has eleven layers. The interfaces consist of three different corner geometries repeated in order. Each of the interfaces contains 40 to 50 right-angle corners. With the five levels of dyadic refinement into each corner there are 10,000 to 15,000 discretization points per interface. Figure 3.4.3 provides more details about the corner geometries including how many discretization points were used on each geometry. The wave number in the layers alternates between 40 and $40\sqrt{2}$. Equation (3.1.1) was solved for 287 incident angles between $[-0.89\pi, -0.11\pi]$. This corresponds to 24 different Bloch phases.

The average flux error estimate over the 287 incident angles is $2.4e - 8$. Figure 3.4.2 illustrates the real part of the total field for the incident angle $\theta^{inc} = -0.845\pi$. The time for constructing and applying the fast direct solver for one incident angle

| | N_i | 1280 | 2560 | 5120 | 10240 | 20480 |
|-------------|---------|--------|--------|--------|---------|---------|
| Precomp I | 3-layer | 50 | 100 | 185 | 337 | 569 |
| | 9-layer | 201 | 407 | 768 | 1390 | 2360 |
| Precomp II | 3-layer | 1.3 | 2.4 | 4.5 | 8.2 | 15.8 |
| | 9-layer | 3.8 | 7.6 | 13.4 | 26.1 | 51.4 |
| Precomp III | 3-layer | 2.5 | 5.9 | 10.6 | 21.4 | 41.9 |
| | 9-layer | 14.9 | 31.5 | 61.5 | 117.1 | 231.2 |
| Solve | 3-layer | 0.1 | 0.2 | 0.8 | 1.6 | 3.4 |
| | 9-layer | 0.6 | 2.5 | 4.9 | 13.5 | 27.9 |
| Flux error | 3-layer | 4.2e-5 | 6.9e-6 | 2.3e-8 | 3.8e-10 | 4.5e-10 |
| | 9-layer | 2.1e-4 | 1.2e-5 | 1.5e-7 | 4.6e-11 | 4.6e-10 |

Table 3.1 : Time in seconds and flux error estimates for applying the direct solver to a 3- and 9-layer geometry where the interfaces alternate between γ_1 and γ_2 defined in (3.4.1). N_i denotes the number of discretization points for each boundary charge density on the interface. The wave number alternates between 10 and $10\sqrt{2}$.

is reported in the column labeled *Original Problem* in Table 3.3. Table 3.2 reports the time for applying the proposed solution technique to the 287 boundary value problems using the same Precomputation I and II for all solves and exploiting the shared Bloch phase accelerations.

Since the first two parts of the precomputation dominate the computational cost for this geometry, significant speed up over building the direct solver for each angle independently is observed. For this problem, the proposed solution technique is 100 times faster than building a fast direct solver from scratch for each incident angle. There is a 10 times speed up in the time for applying the solver for the multiple incident angles. This results from the fact that angles that share a Bloch phase are processed together.

| N_{total} | Precomp I | Precomp II | Precomp III | Solve |
|-------------|-----------|------------|-----------------------------------|------------------------------------|
| 121136 | 2369.3 | 32.3 | 4517.4 (188.2 per Bloch phase) | 482.2 (1.7 per incident angle) |

Table 3.2 : Time in seconds for solving 287 incident angles and 24 distinct Bloch phases on an 11-layer geometry shown in Figure 3.4.2. The incident angles are sampled from $[-0.89\pi, -0.11\pi]$.

3.4.3 Local change to the geometry

This section illustrates the performance of the direct solver when there is a change in one layer of the geometry for the boundary value problem. Either there is a change in an interface geometry or the wave number has been changed in a layer.

We consider an 11-layer geometry where the original set of interfaces are as illustrated in Figure 3.4.2. As in the last section, the wave number alternates between 40 and $40\sqrt{2}$ in the layers of the original geometry. The incident angle for these experiments is fixed at $\theta^{inc} = -\frac{\pi}{5}$. In the first experiment, the fourth interface in the original geometry is changed to the “hedgehog” interface. Figure 3.4.4 illustrates the original and new geometries. The hedgehog geometry consists of 17 sharp corners and cannot be written as the graph of a function defined on the x -axis. The number of discretization points needed on the new interface to maintain the same accuracy as the original problem is $N_4 = 14,496$. In the second experiment, the wave number for the second layer is changed from $40\sqrt{2}$ to 30 but the interfaces are kept fixed with the geometry illustrated in Figure 3.4.4(a). As presented in section 3.3.3, only a small number of the matrices in each step of the precomputation need to be recomputed.

Table 3.3 reports time in seconds for building a direct solver from scratch for the original problem, updating the solver when the fourth interface is replaced and updating the direct solver when there is a change in wave number in the second layer. The parts of Precomputation I and II that are needed in the updating scheme

are smaller than building them from scratch. Precomputation I is approximately twice as expensive for the problem with the changed wave number because it requires updating matrices for two interfaces while the changed interface problem only involves one interface. Precomputation III needs to be redone for the new problem. This is why it is nearly as expensive as it is for the original problem. There is slight cost savings because the ranks related to the replaced wave number are lower since the new wave number is smaller than the original. The cost savings for updating the solver is greater for the changed interface problem since more existing operators of the original solver can be re-used.

When an application requires many solves per new geometry and many new geometries (with local changes) need to be considered, the speed gains over building a solver from scratch for each new geometry will be significant.

| | Original problem | Replace interface Γ_4 | Change wave number $\omega_2 = 30$ |
|-------------|------------------|------------------------------|------------------------------------|
| N_{total} | 121,136 | 125,184 | 121,136 |
| Precomp I | 2369.3 | 237 | 433 |
| Precomp II | 32.3 | 11.7 | 3.5 |
| Precomp III | 174.1 | 41.7 | 109.2 |
| Solve | 18.8 | 15.7 | 13.6 |

Table 3.3 : Time in seconds for constructing and applying the fast direct solver to an 11-layer geometry (first column), a geometry that has the fourth interface changed (second column) and the wave number for the second layer changed from $40\sqrt{2}$ to 30 (third column). N_{total} is the number of discretization points on the interfaces in the unit cell.

3.5 Summary

This chapter presents a fast direct solution technique for multilayered media quasi-periodic scattering problem. For low frequency problems, the computational cost of

the direct solver scales linearly with respect to the number of discretization points. The bulk of the precomputation can be used for all solves independent of incident angle and Bloch phase α . For a problem where over 200 hundred incident angles are considered, the proposed fast direct solver is 100 times faster than building a direct solver from scratch for each incident angle.

An additional benefit of this solution technique is that modifications in the wave number of a layer or an interface geometry result in only local updates to the solver corresponding to that layer or interface. The cost of updating the precomputation part scales linearly with the number of points on that interface. For a problem with a changed interface, the constant associated with the linear scaling is very small for the precomputation (relative to building a new direct solver from scratch). In optimal design and inverse scattering applications where the geometry will be changed many times and for each geometry many solves are required, the fast direct solver will have significant savings.

Two dimensional geometries have to be complex in order to justify the need for the fast direct solver. For three dimensional problems, a fast direct solver will be necessary for most geometries of interest in applications. The extension to three dimensional problems is not trivial but the work presented in this chapter provides the foundations for that work.

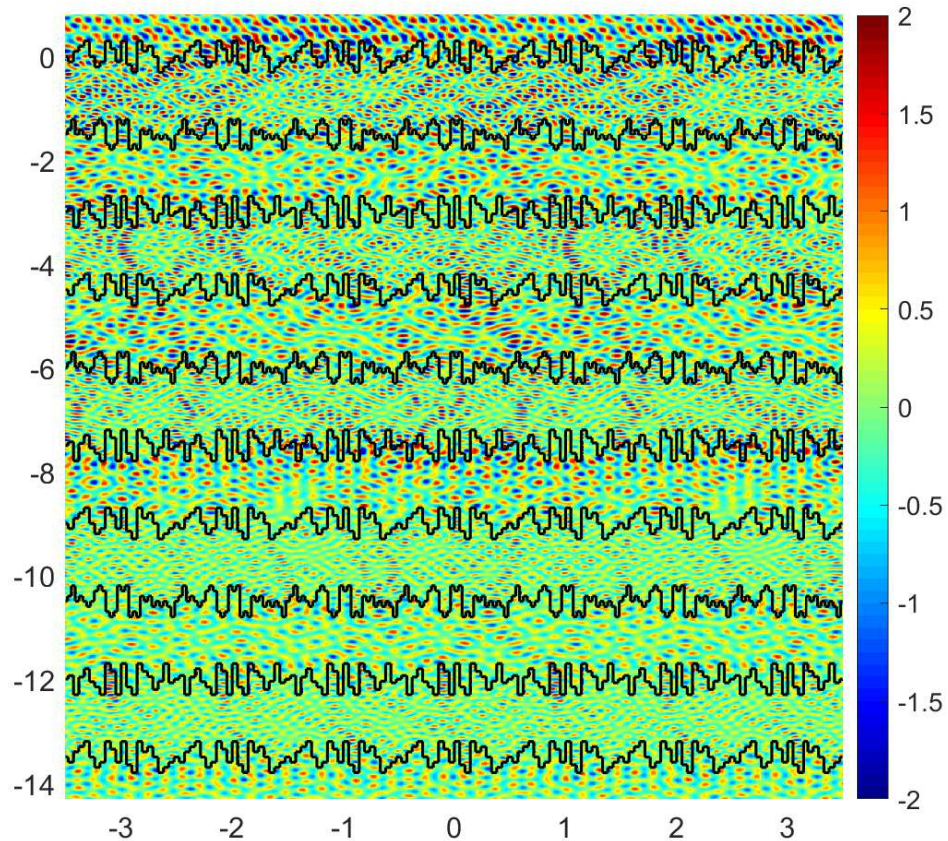


Figure 3.4.2 : Illustration of the real part of the total field of the solution to (3.1.1) for a geometry with 10 interfaces where the wave number alternates between 40 and $40\sqrt{2}$. The shown solution is for $\theta^{inc} = -0.845\pi$. The total number of discretization points was set to $N = 121, 136$, resulting in a flux error estimate of $2.3e - 8$. Seven periods in the geometry are shown.

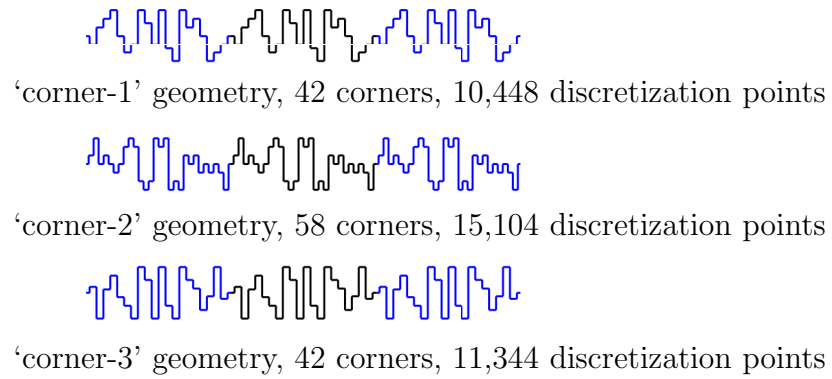


Figure 3.4.3 : The three different “corner” geometries in the 11-layer structure. Three periods are shown. See Figure 3.4.2 for the full structure.

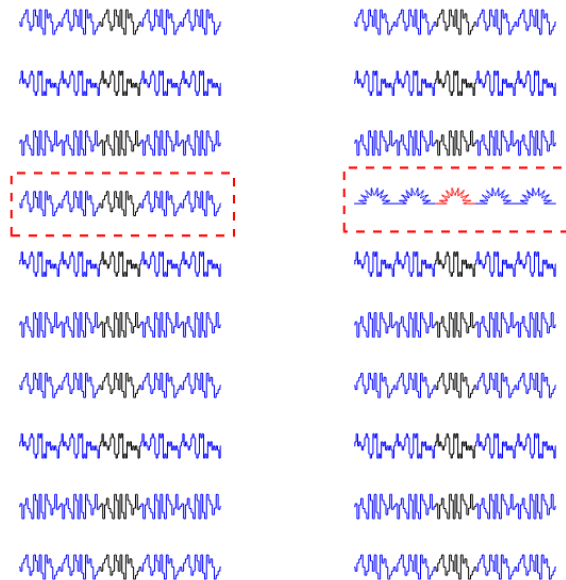


Figure 3.4.4 : Illustration of 5 periods of (a) the original 11-layer structure and (b) the new structure obtained from replacing the fourth interface with a different geometry. The modified interface is in red box.

Chapter 4

A fast direct solver for modeling Stokes flow in confined geometry

The numerical simulation of objects in viscous flow within confined geometries arises in many applications. Examples include modeling blood coagulation [94] (away from the heart), developing microfluidic devices [95, 96], and simulating bacteria [97] and other micro-swimmers [98]. As an alternative to traditional methods, the boundary integral formulation defines unknowns only on the surface of the swimmers/bacteria/blood cells and the confining wall geometry, reducing the dimensionality of the problem by one. For simplicity, the swimmers/bacteria/vesicles are referred to as *bodies* in this chapter. Other examples of bodies in fluid include cilia and flagella structures on the surface of cells or microfluidic devices, which are attached to the confining wall and not “suspending in” or “flowing through” the domain channel [1]. The location and/or shape of the bodies evolve in time, while the confining wall is static. Previous developments on BIE formulations [99, 100, 101, 102, 103, 104] and fast methods [105, 106, 107, 108, 109] have enabled practitioners to simulate large crowds of bodies and study the collective behavior for various applications.

The proposed work focus on simulating bodies in complicated confining geometries. Examples include simulation of bacteria motion and cilia beating dynamics in channel structures, cytoplasmic streaming for movements of nutrient in cells, and optimal design for microfluidic devices[110, 1, 97, 111, 112, 113]. Such problems are often multi-scale by nature. The wall geometry may be complex and requires lots of

discretization points to resolve when no bodies exist. For example, [1] simulates the flow due to cilia beating inside a geometry extracted from a generic microscopic image of the cross-section of a Fallopian tube shown in Figure 4.0.1. Microfluidic devices for transporting fluid or particles are often designed to have complex geometries as well [114, 49]. As bodies get close to the wall, the discretization of the wall often needs to be refined locally to capture the behavior in the density. As the bodies travel to new locations for each time step, the part of the boundary that is approached by bodies often change. Thereby, out-of-box fast direct solution techniques, such the HBS method, is too expensive to be helpful if applied directly, as the discretized linear system changes for each time step. The proposed fast direct solver is designed to be coupled with such “time-evolving” discretization refinement.

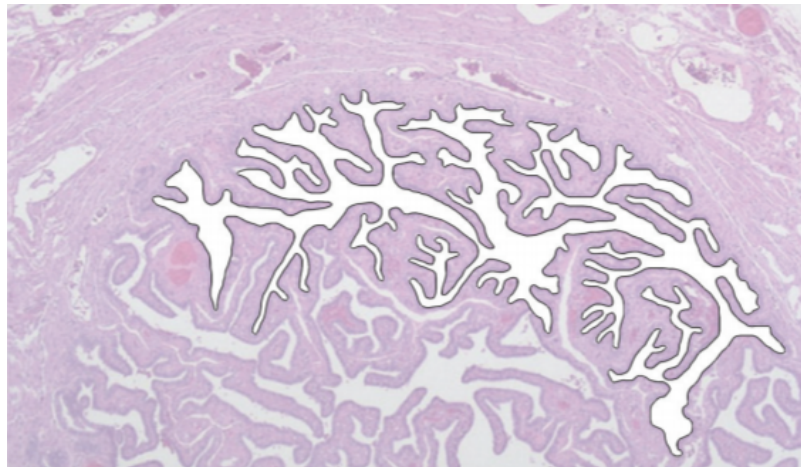


Figure 4.0.1 : Example of a complex confining wall geometry from [1]. The geometry is extracted from a generic microscopic image of the cross-section of a Fallopian tube.

The work is based on the author’s previous work on solving BIEs with locally perturbed geometry [50] reviewed in section 2.4. The proposed work includes an improved version of the solver in [50] and an extension of the algorithm to handle the

time-evolving discretization refinement for confined Stokes flow. The general idea is to precompute a fast direct solver for the original wall discretization independently of time and then update the original solver to handle the extra refinements introduced for later time steps.

This chapter is organized as follows. Section 4.1 gives the BIE formulation for modeling bodies in confined Stokes flow and an example time stepping scheme for evolving the bodies. The time stepping scheme decouples the problem into BIEs defined on the confining wall and a linear system for variables defined on the surface of the bodies. The decoupling allows separate quadratures and solution techniques to be used on the two parts to obtain better accuracy and efficiency. The geometry of the bodies are often very simple and can be resolved to high accuracy with small number of points. Thus, the major computational cost comes from solving the BIE defined on the confining wall. Section 4.2 presents on a fast solution technique for BIEs with local discretization refinement. This solver can be coupled with a local refinement strategy, which specifies the region to be refined and generates the new discretization, to efficiently obtain an accurate solution to the BIE on the confining wall. Section 5.4 illustrates the performance of the solver with numerical examples, and section 4.4 summarizes the chapter and gives future research directions.

4.1 Boundary integral formulation

This section first presents the BIE formulation for the steady-state problem and then the time-stepping scheme for simulating the bodies. The complete formulation depends on the kind of bodies studied. For example, deformable vesicles can be modeled by tracking the surface positions and surface tension while rigid-bodies can often be modeled with much less degrees of freedom, such as tangential and rotational

velocity of the centroid [103, 104]. Even for non-deformable bodies, different types often satisfy different governing equations. Although the body type may be different, the boundary integral formulations of the problems share similar format, especially in terms of how the effect of the confining wall is incorporated into the dynamics of the bodies. For illustrative purposes, section 4.1.1 derives a BIE formulation for a single vesicle in confined Stokes flow, and section 4.1.2 presents the semi-implicit time stepping scheme for evolving the vesicle from [85]. This specific example demonstrates how a time stepping scheme decouples the problem into two separate problems at each time step: one for the confining wall and one for the vesicle. Section 4.1.3 then briefly discusses the key parts of the numerical simulation and explains the difficulty associated with handling local refinement for the confining wall.

4.1.1 The steady-state problem

This section describes a BIE method for the steady-state problem for modeling single vesicle in confined Stokes flow. First, assume no vesicle exists and consider the pipe geometry shown in Figure 4.1.1. Let μ denote the fluid viscosity, the velocity field and pressure (\mathbf{u}, p) satisfies the BVP

$$\begin{aligned} -\mu\nabla^2\mathbf{u}(\mathbf{x}) + \nabla p(\mathbf{x}) &= \mathbf{0}, & \text{for } \mathbf{x} \in \Omega \\ \nabla \cdot \mathbf{u}(\mathbf{x}) &= 0, & \text{for } \mathbf{x} \in \Omega \\ \mathbf{u}(\mathbf{x}) &= \mathbf{g}(\mathbf{x}), & \text{for } \mathbf{x} \in \Gamma, \end{aligned} \tag{4.1.1}$$

where $\mathbf{u}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are vector-valued functions, and $p(\mathbf{x})$ is scalar-valued. Let $\mathbf{nu}(\mathbf{x})$ be the outward pointing normal vector at \mathbf{x} . The Dirichlet boundary data needs to

satisfy the consistency condition



Figure 4.1.1 : Model geometry for the Stokes BVP.

Remark 4.1.1 Domain Ω may be multiply connected, i.e. Ω may have “holes”. The normal vector on the boundary of the hole should be defined to point to the fluid domain.

The BIE formulation for (4.1.1) can be derived following the same procedure for the Laplace problem in section 1.2. The velocity field in Ω can be represented as a layer potential of the format

$$\mathbf{u}(\mathbf{x}) = (\mathcal{K}\tau)(\mathbf{x}), \quad (4.1.3)$$

where τ is some unknown vector-valued boundary density function. Here, the notation for kernel \mathcal{K} is intentionally left to be general. Two options are the double layer kernel

$\mathcal{K} = \mathcal{D}$ and the combined field kernel $\mathcal{K} = \mathcal{S} + \mathcal{D}$, where \mathcal{S} denotes the single layer kernel for Stokes. Section A.2 gives a list of kernel definitions for completeness.

The pressure is given by a similar format

$$p(\mathbf{x}) = (\mathcal{K}^P \tau)(\mathbf{x}), \quad (4.1.4)$$

where \mathcal{K}^P denotes the corresponding pressure kernel operator.

The kernel functions guarantee that the (\mathbf{u}, p) pair defined by (4.1.3) and (4.1.4) will automatically satisfy the first two equations in (4.1.1). The BIE

$$\lambda \tau(\mathbf{x}) + (\mathcal{K} \tau)(\mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad (4.1.5)$$

results from taking the limit of (4.1.3) as $\mathbf{x} \rightarrow \Gamma$ and enforcing the Dirichlet boundary condition. The term $\lambda \tau(\mathbf{x})$ comes from the jump condition of the kernel function, where λ is a scalar. For example, $\lambda = -\frac{1}{2}$ if the double layer or the combined field kernel is used.

Now, consider the flow due to the deformation of a single vesicle in free space. Let γ denote the boundary of the vesicle and assume it is parameterized by $\mathbf{x}(s)$ where s is the arc-length. The membrane exerts force $\mathbf{f} = -\kappa_B \mathbf{x}_{ssss} + (\sigma \mathbf{x}_s)_s$ on the fluid, where κ_B is the bending modulus, σ is the tension, and \mathbf{x}_{ssss} denotes the fourth derivative of \mathbf{x} with respect to s . Assuming the force is known, it can be shown that the representation

$$\mathbf{u}(\mathbf{x}) = (\mathcal{S}_\gamma \mathbf{f})(\mathbf{x}), \text{ and } p(\mathbf{x}) = (\mathcal{S}_\gamma^P \mathbf{f})(\mathbf{x}) \quad (4.1.6)$$

satisfies the required jump conditions on the membrane surface as well as the Stokes equation in the vesicle domain [85]. The subscript notation is introduced to differen-

tiate the solution representation in (4.1.3) and (4.1.4). For the rest of this chapter, (4.1.5) is written as

$$\lambda\tau(\mathbf{x}) + (\mathcal{K}_\Gamma\tau)(\mathbf{x}) = \mathbf{g}(\mathbf{x}).$$

The fluid flow generated by the vesicle in the confined domain can be written as $\mathbf{u}(\mathbf{x}) = \mathbf{u}_i(\mathbf{x}) + \mathbf{u}_r(\mathbf{x})$, where $\mathbf{u}_i(\mathbf{x})$ is the *imposed flow* due to the vesicle in free space and $\mathbf{u}_r(\mathbf{x})$ is the *response flow* due to the wall. This is similar to the idea of incident and scatter field in wave scattering problems. If the force \mathbf{f} is known, the imposed flow is given in (4.1.6), and the response flow can be evaluated via solving (4.1.5) with the right-hand-side set to be the negative imposed flow velocity. Namely,

$$\mathbf{u}_r(\mathbf{x}) = (\mathcal{K}_\Gamma\tau_{\mathbf{f}})(\mathbf{x}), \quad (4.1.7)$$

where $\tau_{\mathbf{f}}$ is the solution to

$$\lambda\tau_{\mathbf{f}}(\mathbf{x}) + (\mathcal{K}_\Gamma\tau_{\mathbf{f}})(\mathbf{x}) = -(\mathcal{S}_\gamma\mathbf{f})(\mathbf{x}). \quad (4.1.8)$$

The formulation can be easily extended to the multiple-vesicle case. The idea of the imposed and response flow can be extended to modeling other kinds of bodies. The major difference between the formulations of different kinds of bodies is how the imposed flow is modeled and evaluated. The BIE operator (left-hand-side operator in (4.1.8)) for evaluating the response flow will be the same, but the right-hand-side to be plugged into the BIE will be different.

4.1.2 Time-dependent problem formulation

This section presents the semi-implicit time-stepping scheme for evolving vesicles in confined Stokes flow from [85], which is a slightly modified version of the time stepping scheme in [115]. The dynamics of the vesicle is modeled by location and tension pair (\mathbf{x}, σ) satisfying

$$\begin{aligned}\dot{\mathbf{x}} &= (\mathcal{S}_\gamma \mathbf{f})(\mathbf{x}) + \mathbf{u}_r(\mathbf{x}) \\ 0 &= \mathbf{x}_s \cdot (\dot{\mathbf{x}})_s,\end{aligned}\tag{4.1.9}$$

where \mathbf{f} depends on (\mathbf{x}, σ) as $\mathbf{f} = -\kappa_B \mathbf{x}_{ssss} + (\sigma \mathbf{x}_s)_s$ and \mathbf{u}_r depends on (\mathbf{x}, σ) through its dependence on \mathbf{f} in (4.1.7) and (4.1.8). The first equation comes from setting the time derivative of the location of points on the vesicle boundary to be the fluid velocity evaluated on the boundary, and the second equation results from enforcing the surface divergence of the membrane velocity to be zero.

Numerically, the fourth order derivative with respect to arc-length s leads to stiffness. Explicit method, such as forward Euler, needs to have very small time step size and is thus too expensive for numerical simulations. Instead, a semi-implicit time stepping scheme is used to evolve (\mathbf{x}, σ) . Let Δt be the time step size, the method solves the following system for the unknown $\mathbf{v}^{k+1} := \frac{\mathbf{x}^{k+1} - \mathbf{x}^k}{\Delta t}$ and σ^{k+1}

$$\begin{aligned}\mathbf{v}^{k+1} - \mathcal{S}_\gamma[-\Delta t \kappa_B \mathbf{v}_{ssss} + (\sigma^{k+1} \mathbf{x}_s^k)_s] &= \mathcal{S}_\gamma[-\kappa_B \mathbf{x}_{ssss}^k] + \mathbf{u}_r^k \\ \mathbf{x}_s^k \cdot \mathbf{v}_s^{k+1} &= 0.\end{aligned}\tag{4.1.10}$$

Note that the response flow is treated explicitly: $\mathbf{u}_r^k = \mathcal{K}_\Gamma \tau_k$, where τ_k is the solution to

$$\lambda \tau_k + \mathcal{K}_\Gamma \tau_k = -\mathcal{S}_\gamma[-\kappa_B \mathbf{x}_{ssss}^k + (\sigma^k \mathbf{x}_s^k)_s].$$

Once \mathbf{v}^{k+1} and σ^{k+1} are solved from (4.1.10), the membrane positions can be updated as $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta t \mathbf{v}^{k+1}$.

The semi-implicit time-stepping scheme decouples the problem into a BIE defined on the confining wall geometry and a linear system for \mathbf{v}^{k+1} and σ^{k+1} , which evolve the membrane's location/shape in time. For simulating other kind of bodies, the same idea of writing the field velocity as $\mathbf{u} = \mathbf{u}_i + \mathbf{u}_r$ can be used. Then time-stepping schemes with an explicit treatment of the response flow will decouple the problem into a confining wall BIE and a linear system for updating the location of the bodies similar to (4.1.10). Information transfers between the two decoupled problems in the form of potential evaluations. Thus, an efficient solver for the BIE defined on the confining wall geometry will be useful for numerical simulations for various bodies.

4.1.3 Numerical simulation and local discretization refinements

In practice, the vesicle geometry is often simple and can be discretized with small number of discretization points. For example, only 64 points per vesicle are used in [115]. The major cost of one time step is solving the BIE defined on the confining wall and evaluating the potential of the form $\mathbf{u}_r|_\gamma = (\mathcal{K}_\Gamma \tau)|_\gamma$ (wall-to-vesicle) and $\mathbf{u}_i|_\Gamma = (\mathcal{S}_\gamma \mathbf{f})|_\Gamma$ (vesicle-to-wall). When the bodies get close to the wall, special near-field evaluations are necessary to evaluate the potentials accurately [116, 40, 117]. The “not-so-close” potential evaluations can be accelerated by FMM, leaving the BIE solve to be the major cost concern. Since the wall is static, a direct solver for the wall geometry is ideal for this problem. The forward representation and the inversion only need to be constructed once (independently of the time steps), and each time step only involves applying the inverse to a new right-hand-side. In [115], speed-ups are observed by using a fast direct solver for the wall geometry instead of an FMM

accelerated iterative solver for each time step.

When simulating bodies in complex confining wall geometries such as the simulating cilia motion in the Fallopian tube example in [1], the wall geometry itself requires lots of discretization points when no bodies exit. Difficulty arises when the bodies are close to the boundary: the wall may need to be refined locally to capture the density accurately. Roughly speaking, if the boundary wall is discretized in panel quadrature, then local refinement is often necessary if the distance between the bodies and the wall is smaller than the panel width of the wall discretization. For simulating the bodies' dynamics in time, the location of the bodies evolve and the region on the wall subject to local refinements changes. It is not possible to predict which region to refine a priori. Due to the complexity of the boundary wall, a priori uniform refinement is not practical. Time dependent local discretization refinement should be adopted.

If the wall discretization is locally refined in different regions for each time step, out-of-the-box fast direct solvers, such as HBS, are too expensive since a new inverse needs to be constructed for each time step. Section 4.2 presents a fast direct solver which avoids the new forward and inversion when handling local refinements in discretization. This solver can be coupled with a local refinement strategy, which determines the region to be refined and generates the new discretization, and a time stepping scheme to numerically simulate the bodies' dynamics in confined Stokes flow.

4.2 A fast direct solver for BIEs with locally refined discretization

The section presents a fast direct solver for BIEs with locally-refined discretization. It is an improved version of the fast direct solver in [50], which is reviewed in section

2.4. The algorithm in [50] is originally designed to handle BIEs defined on locally perturbed geometries. With the Nyström discretization, the row and columns of the coefficient matrix corresponds to discretization points on the boundary. Thus, the local-perturbation changes to the discretized linear system correspond to physically adding and deleting discretization points. This allows the method to be naturally extended to handle local discretization refinements, which can also be regarded as adding and deleting discretization points on the boundary. Unfortunately, the fast direct solver for the original extended system as in section 2.4 required inverting a matrix the size of the number of discretization points removed from the original geometry which is expensive if the removed portion is large. Another difficulty of the original extended system is that care is required when the technique is applied to systems discretized using quadrature for weakly singular kernels, e.g. the single layer Laplace or Stokes kernel. This section proposes an alternative extended system formulation which overcomes these two difficulties. Additionally, a fast direct solver for the new extended system can be constructed from the tools presented in [50] but is more efficient than the original fast direct solver.

4.2.1 An alternative extended system formulation

Assume that the original problem (2.4.1) is solved via a fast direct solver, i.e. an approximate inverse $\mathbf{A}_{oo}^{inv} \approx \mathbf{A}_{oo}^{-1}$ is available. This section presents an alternative extended system formulation, which is equivalent to the linear system for the locally refined discretization (2.4.2).

Recall from section 2.4, in the original extended system formulation, σ_c is not used to find the solution inside of Γ_n , thus we introduce the vector σ_c^{dum} fully knowing a priori that it will contain useless information. Then solving (2.4.2) is equivalent to

solving the following

$$\begin{bmatrix} \mathbf{A}_{kk} & \mathbf{0} & \mathbf{A}_{kp} \\ \mathbf{A}_{ck} & \mathbf{A}_{cc} & \mathbf{0} \\ \mathbf{A}_{pk} & \mathbf{0} & \mathbf{A}_{pp} \end{bmatrix} \begin{pmatrix} \boldsymbol{\sigma}_k \\ \boldsymbol{\sigma}_c^{dum} \\ \boldsymbol{\sigma}_p \end{pmatrix} = \begin{pmatrix} \mathbf{g}_k \\ \mathbf{0} \\ \mathbf{g}_p \end{pmatrix}. \quad (4.2.1)$$

The expanded form of (4.2.1) is

$$\left(\underbrace{\begin{bmatrix} \mathbf{A}_{oo} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{pp} \end{bmatrix}}_{\tilde{\mathbf{A}}} + \underbrace{\begin{bmatrix} \mathbf{0} & -\mathbf{A}_{kc} & \mathbf{A}_{kp} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{pk} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}_{new}} \right) \underbrace{\begin{pmatrix} \boldsymbol{\sigma}_k \\ \boldsymbol{\sigma}_c^{dum} \\ \boldsymbol{\sigma}_p \end{pmatrix}}_{\boldsymbol{\sigma}_{ext}} = \underbrace{\begin{pmatrix} \mathbf{g}_k \\ \mathbf{0} \\ \mathbf{g}_p \end{pmatrix}}_{\mathbf{g}_{ext}}. \quad (4.2.2)$$

Here \mathbf{Q}_{new} is the new update matrix. Notice that \mathbf{Q}_{new} has a zero row.

Since the three non-trivial subblocks in the update matrix \mathbf{Q}_{new} are all rank-deficient, \mathbf{Q}_{new} can be approximated by

$$\begin{array}{ccc} \mathbf{Q}_{new} & = & \mathbf{L} \quad \mathbf{R} \\ N_{ext} \times N_{ext} & & N_{ext} \times k \quad k \times N_{ext} \end{array} \quad (4.2.3)$$

where

$$\mathbf{L} = \begin{bmatrix} \mathbf{0} & -\mathbf{L}_{kc} & \mathbf{L}_{kp} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{pk} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{pk} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{kc} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{kp} \end{bmatrix},$$

and

$$\begin{aligned}
\mathbf{A}_{kc} &\approx \mathbf{L}_{kc} \mathbf{R}_{kc}, & \mathbf{A}_{kp} &\approx \mathbf{L}_{kp} \mathbf{R}_{kp}, \text{ and} \\
N_k \times N_c & \quad N_k \times k_{kc} \quad k_{kc} \times N_c & N_k \times N_p & \quad N_k \times k_{kp} \quad k_{kp} \times N_p \\
\mathbf{A}_{pk} &\approx \mathbf{L}_{pk} \mathbf{R}_{pk}. \\
N_p \times N_k & \quad N_p \times k_{pk} \quad k_{pk} \times N_k
\end{aligned} \tag{4.2.4}$$

Here $k = k_{pk} + k_{kc} + k_{kp}$ and $N_{ext} = N_k + N_c + N_p$.

4.2.2 Low-rank approximation for the update matrix

For simplicity of presentation, this section details the technique for compressing the subblock \mathbf{A}_{kp} in the update matrix \mathbf{Q}_{new} . The technique is used to compress the subblock \mathbf{A}_{kc} , and with minor modifications, it is applied to compress \mathbf{A}_{pk} as well. The algorithm starts with a proxy circle for Γ_p , the part of the boundary gets refined. Figure 4.2.1 (a) plots the proxy circle. For clarity, let P_p^{div} denote this proxy circle, and superscript notation “far” and “near” is used to denote the far- and near- field. The far-field interaction \mathbf{A}_{kp}^{far} and near-field interaction \mathbf{A}_{kp}^{near} are compressed separately and then combined together to form an ID for the subblock \mathbf{A}_{kp} .

For the far-field compression, the potential due to Γ_p evaluated on Γ_k^{far} can be approximated by a linear combination of basis functions defined on any proxy surface that shields Γ_p away from Γ_k^{far} . Let P_p^{bas} denote this “shielding” proxy circle for Γ_p . The proxy circle P_p^{bas} is typically choose to have the same center as P_p^{div} but smaller radius as shown in Figure 4.2.1 (b). A ID approximation for \mathbf{A}_{kp}^{far} can then be obtained by constructing an ID approximation for the interaction between Γ_k^{far} and P_p^{bas} . However, due to the large number of discretization points on Γ_k^{far} , directly building an ID can be very expensive. Thus, a dyadic partition to group points in

Γ_k^{far} together base on their distance to Γ_p is used. For simple illustration, assume the boundary curve Γ is parameterized as $\Gamma = \{\gamma(t) : t \in [t_a, t_c]\}$ with $t_c = t_a + 2\pi$. And $\Gamma_k^{far} = \{\gamma(t) : t \in [t_a, t_b]\}$. Figure 4.2.2 shows a dyadic partition in parameterization space of Γ_k^{far} . ID approximations are constructed first for blocks of rows corresponding to each box in the partition and then combined together to form the ID for \mathbf{A}_{kp}^{far} .

The near field part \mathbf{A}_{kp}^{near} is compressed similarly to the compression of an off-diagonal block shown in section 2.2, but with an underlying dyadic partition for the rows similar to the dyadic partition for Γ_k^{far} in compressing \mathbf{A}_{kp}^{far} . For each box b in the partition, a proxy circle is determined for $\Gamma_b \subsetneq \Gamma_k$. Two scenarios are possible: the first scenario is that the proxy circle does not enclose any part of Γ_p ; the other scenario is that the proxy circle enclose certain parts of Γ_p , which is denoted by $\Gamma_p^{near(b)}$. Let matrix \mathbf{A}_b^{proxy} be the interaction between Γ_b and the proxy circle, and let \mathbf{A}_b be the submatrix of \mathbf{A}_{kp}^{near} with rows corresponding to discretization points in b . For the first scenario, it is sufficient to build a row-based ID for \mathbf{A}_b^{proxy} to obtain a row skeleton index J_b and an interpolation matrix \mathbf{P}_b . Then \mathbf{A}_b can be approximated as $\mathbf{A}_b = \mathbf{P}_b \mathbf{A}_b(J_b, :)$. For the second scenario, a low-rank factorization for \mathbf{A}_b can be obtained by constructing a row-based ID for

$$\hat{\mathbf{A}}_b = [\mathbf{A}_b^{near} \mid \mathbf{A}_b^{proxy}],$$

where matrix \mathbf{A}_b^{near} is interaction between Γ_b and $\Gamma_p^{near(b)}$. The factorizations for each of the box b in the partition are then combined together to form an ID for \mathbf{A}_{kp}^{near} .

Remark 4.2.1 For both \mathbf{A}_{kp}^{far} and \mathbf{A}_{kp}^{near} , combining the individual box's ID together by simply concatenating the resulting low-rank factors together can lead to an approximation with very high total rank numbers. Thereby, an extra re-compression step

from [38] is used for merging the individual box compressions.

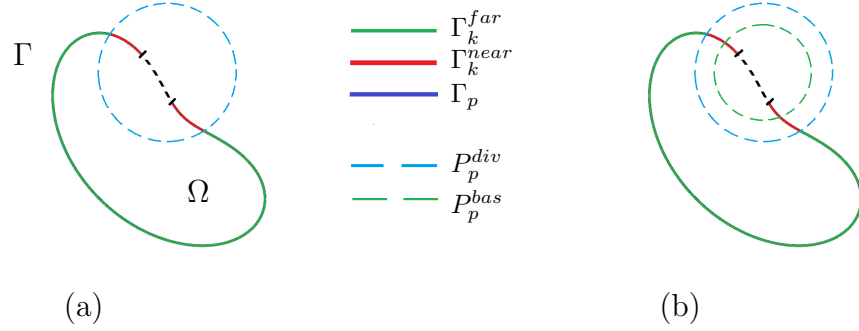


Figure 4.2.1 : (a) The proxy circle for Γ_p shown in dash blue line divides Γ_k into far (in green) and near (in red) with respect to Γ_p (b) The interaction between the far-filed part of Γ_k and Γ_p can be captured by the interaction between Γ_k^{far} and a smaller proxy circle for Γ_p shown in dash green.

The low-rank approximation algorithm applies to the compression of the subblock \mathbf{A}_{kc} as well. Since the removed points and added points discretize the same boundary curve segment $\Gamma_c = \Gamma_p$, the far-filed part of the low-rank approximation for \mathbf{A}_{kp} and \mathbf{A}_{kc} can be merged into one. The compression for \mathbf{A}_{pk} is similar to that for \mathbf{A}_{kp}^{near} .

4.2.3 Solution evaluation

Once a low-rank approximation for the update matrix \mathbf{Q}_{new} is constructed, the solution to the extended system can be evaluated via the Sherman-Morrison-Woodbury formula (2.4.4). Note that the new update matrix \mathbf{Q}_{new} does not contain any full-rank subblock and has lower rank numbers when compared to the original update matrix \mathbf{Q}_{orig} in section 2.4. Thus, the Sherman-Morrison-Woodbury formula can be applied more rapidly.

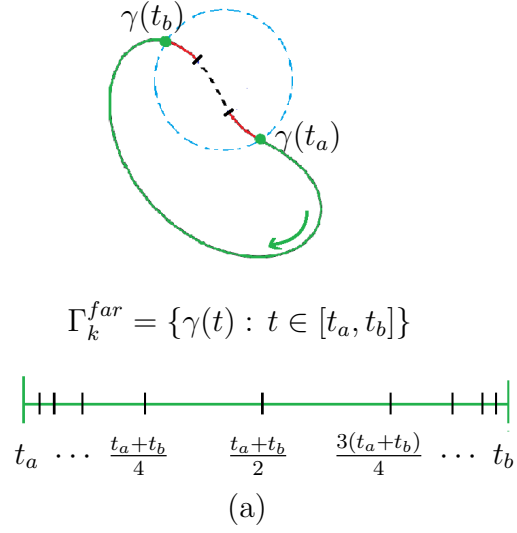


Figure 4.2.2 : Dyadic partition for Γ_k^{far} used in the compression of \mathbf{A}_{kp}^{far} .

Evaluating the solution σ_{ext} (2.4.4) requires matrix-matrix multiplication, matrix-vector multiplication, and solving a linear system of the form $(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L})\vec{x} = \vec{b}$. For efficiency, the sparsity pattern in the operators should be exploited and the matrix-matrix multiplication should be done block-wise. For example, let

$$\tilde{\mathbf{L}} = \begin{pmatrix} -\mathbf{L}_{kc} & \mathbf{L}_{kp} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

then $\tilde{\mathbf{A}}^{-1}\mathbf{L}$ should be evaluated as

$$\tilde{\mathbf{A}}^{-1}\mathbf{L} \approx \begin{bmatrix} \mathbf{A}_{oo}^{inv} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{pp}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{L}} \\ \mathbf{L}_{pk} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{oo}^{inv}\tilde{\mathbf{L}} \\ \mathbf{A}_{pp}^{-1}\mathbf{L}_{pk} & \mathbf{0} \end{bmatrix}$$

and $\mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}$ as

$$\mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L} \approx \begin{bmatrix} \mathbf{0} & \begin{pmatrix} \mathbf{R}_{pk} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{kc} \end{pmatrix} \mathbf{A}_{oo}^{inv} \tilde{\mathbf{L}} \\ \mathbf{R}_{kp} \mathbf{A}_{pp}^{-1} \mathbf{L}_{pk} & \mathbf{0} \end{bmatrix}.$$

The repeated terms in (2.4.4) should be only computed once. The self operator defined for the locally refined region \mathbf{A}_{pp} can be inverted densely if the region contains small number of points or via a separate HBS if it contains large number of points. For the targeted applications, the Sherman-Morrison-Woodbury operator (of size $k \times k$) is usually small enough for a dense inversion.

The cost of the presented solver contains two parts: precomputation, which is independent of the right-hand-side, and the solve, which depends on the right-hand-side. The precomputation includes low-rank approximation for subblocks in the update matrix \mathbf{Q}_{new} , evaluating terms such as $\tilde{\mathbf{A}}^{-1}\mathbf{L}$ and $\mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}$ in (2.4.4), and the evaluation and inversion of the Sherman-Morrison-Woodbury operator $(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L})$. The precomputation part dominates the total cost but only needs to be done once for problems with multiple right-hand-sides. Both precomputation and solve scales linearly with respect to N_k, N_c and N_p , provided that a separate HBS is constructed for Γ_p .

4.3 Numerical experiments

This section demonstrates the performance of the fast direct solver by an example. To validate the accuracy of the solution approximation, the solver is applied to two

test problems with known solution

$$\mathbf{u}_{ext}(\mathbf{x}) = \begin{bmatrix} -x_2^2 + x_2 \\ 0 \end{bmatrix}.$$

Then the solution approximation u_{app} is evaluated at a collection of interior target points $\{\mathbf{t}_i\}_{i=1}^{N_{trg}}$. The average relative error over the target locations given by

$$E = \frac{1}{N_{trg}} \sum_{i=1}^{N_{trg}} \frac{|\mathbf{u}_{app}(\mathbf{t}_i) - \mathbf{u}_{ext}(\mathbf{t}_i)|}{|\mathbf{u}_{ext}(\mathbf{t}_i)|} \quad (4.3.1)$$

is reported as a measure of accuracy.

The algorithm is implemented in Matlab, except that the random sampling based ID is in Fortran. All tests are run on a dual 2.3 GHz Intel Xeon Processor E5-2695 v3 desktop workstation with 256GB RAM.

The following values are reported:

- N_k , N_c , and N_p , number of discretization points kept the same, removed, and added for the refinement.
- $T_{new,p}$: time in seconds for the precomputation of the proposed solver.
- $T_{hbs,p}$: time in seconds for the precomputation of HBS from scratch.
- r_p : the ratio between $T_{hbs,p}$ and $T_{new,p}$.
- $T_{new,s}$: time in seconds for one right-hand-side solve of the proposed solver.
- $T_{hbs,s}$: time in seconds for one right-hand-side solve of HBS from scratch.
- r_s : the ratio between $T_{hbs,s}$ and $T_{new,s}$.

The ratios r_p and r_s are measures for the speed-up (or slow-down) by using the proposed solver versus building a new fast direct solver from scratch for the new geometry. If r_p is greater than 1, the precomputation of the proposed solver is faster than building a fast direct solver from scratch. If r_p is less than 1, the precomputation of the proposed solver is slower than building a fast direct solver from scratch, etc. For all tests, the tolerance for HBS compression and low-rank approximation is set to $\epsilon = 10^{-10}$.

4.3.1 Scaling test

Consider a Stokes boundary value problem defined on a locally-refined pipe shown in Figure 4.3.1. Assume that the pipe is originally discretized with N_o many points, and local refinements occurs in the boxed region in Figure 4.3.1. This results in a new discretization with $N_n = N_k + N_p$ many points. The errors for both the new solver and HBS from scratch are about 10^{-9} since the tolerance for compression is set to be $\epsilon = 10^{-10}$.

Table 4.1 summarizes the timings for the test. When the number of unknown on Γ_p exceeds 4000, i.e. $N_p > 2000$, an HBS solver is built for \mathbf{A}_{pp} . Table 4.1 shows that the cost of the precomputation and solve step both scales linearly with respect to N_k , N_c , and N_p for the new solver and HBS from scratch. The cost of precomputation for the new solver is about 5 times faster than that for building an HBS solver from scratch, while the solve step for the new solver is about 1.5 times slower than applying an HBS solver. Since the precomputation is much more expensive than one right-hand-side solve, it takes at least 600 right-hand-sides to justify building an HBS solver from scratch rather than using the new solver for the test problem.

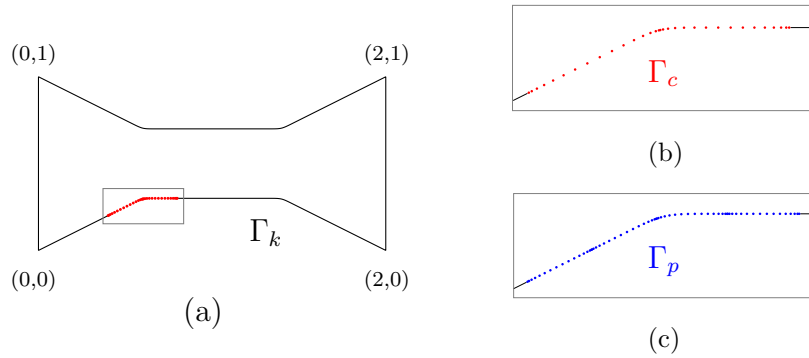


Figure 4.3.1 : (a) The pipe geometry with the portion of the boundary to be refined boxed. The coordinates of the four sharp corners of the pipe is also shown. (b) The two Gaussian panels in the boxed region from the original discretization. (c) The four Gaussian panels that replaced the original two panels.

| N_k, N_c, N_p | $T_{\text{new,p}}$ | $T_{\text{hbs,p}}$ | r_p | $T_{\text{new,s}}$ | $T_{\text{hbs,s}}$ | r_s |
|--------------------------|--------------------|--------------------|-------------|--------------------|--------------------|-------------|
| 3744, 128, 512 | 0.75 | 4.98 | 6.67 | 0.04 | 0.01 | 0.30 |
| 7392, 256, 1024 | 1.38 | 8.47 | 6.25 | 0.03 | 0.02 | 0.72 |
| 14816, 512, 2048 | 3.59 | 15.79 | 4.35 | 0.06 | 0.04 | 0.65 |
| 29728, 992, 3968 | 6.38 | 34.32 | 5.26 | 0.13 | 0.09 | 0.65 |
| 59456, 1984, 7936 | 13.90 | 75.17 | 5.56 | 0.26 | 0.18 | 0.67 |

Table 4.1 : Timing results for Stokes boundary value problem on the pipe with refined panel geometry. A separate HBS for Γ_p is constructed for test values with $N_p > 2000$, shown in bold font.

4.3.2 Complex geometry test

This section applies the solver to a more realistic problem. Consider the “bumpy pipe” geometry shown in Figure 4.3.2 (a), which is generated by applying the corner smoothing technique in [2] to a polygonal pipe. For the initial discretization more panels are placed near the corners of the bumps and the straight line segments are discretized with few panels to resolve the geometry. The error E for the initial discretization is about 10^{-9} when the tolerance of HBS compression is set to $\epsilon = 10^{-10}$.

Assume that at a future time step $k \geq 1$, vesicles or other type of bodies pass through the narrow part of the pipe and approaches the boundary in the region highlighted in red in Figure 4.3.2(a). Panels in the red region get refined as shown in Figure 4.3.2(c) in order to capture the density. Table 4.2 reports the the solver’s cost as the number of panels added to the red region increases. For the test cases where $N_p \leq 96N_c$, a two to five times speed up is observed. Thus, the solver is very efficient in handling local refinements even if the refinement adds large number of new panels when compared to the original discretization.

4.3.3 Other tests

In section A.3 in the Appendix, the new solver is applied to a collection of Laplace and Helmholtz BVPs. The results show that the new extended system formulation leads to better efficiency compared to the version in [50].

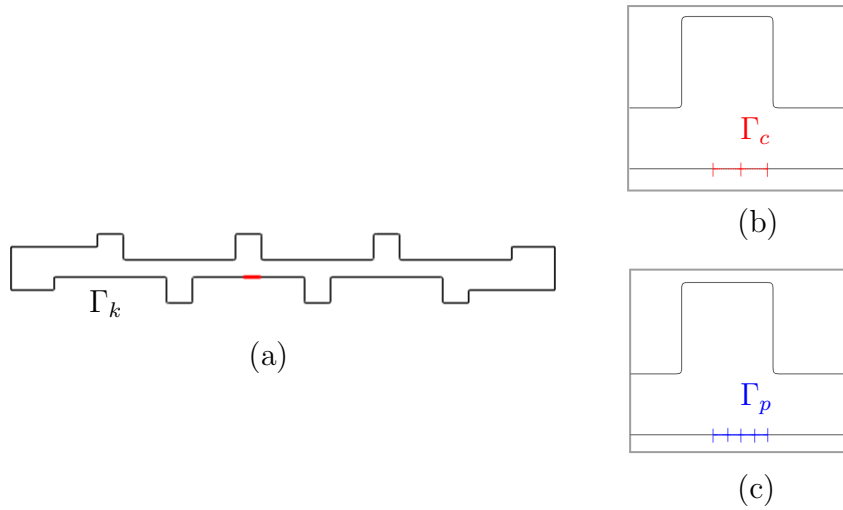


Figure 4.3.2 : (a) The bumpy pipe geometry with the portion of the boundary to be refined highlighted in red. (b) The two Gaussian panels to be removed from the original discretization. (c) The four Gaussian panels to be added to replace the original two panels.

| N_p | $T_{\text{new,p}}$ | $T_{\text{hbs,p}}$ | r_p | $T_{\text{new,s}}$ | $T_{\text{hbs,s}}$ | r_s |
|-------|--------------------|--------------------|-------|--------------------|--------------------|-------|
| 96 | 1.61 | 7.42 | 4.61 | 0.02 | 0.01 | 0.78 |
| 192 | 1.32 | 7.11 | 5.39 | 0.02 | 0.01 | 0.66 |
| 384 | 1.44 | 7.51 | 5.22 | 0.02 | 0.01 | 0.58 |
| 768 | 1.73 | 9.81 | 5.67 | 0.02 | 0.02 | 1.09 |
| 1536 | 2.83 | 11.16 | 3.84 | 0.02 | 0.03 | 1.23 |
| 3072 | 5.62 | 13.12 | 2.33 | 0.03 | 0.03 | 0.90 |
| 6144 | 22.42 | 20.15 | 0.90 | 0.05 | 0.05 | 0.81 |
| 12288 | 112.27 | 32.24 | 0.29 | 0.12 | 0.07 | 0.53 |

Table 4.2 : Timing results for applying the solver to the Stokes BVP on the bumpy pipe geometry. $Nk = 2528$, $Nc = 32$, and \mathbf{A}_{pp} is evaluated and inverted via dense linear algebra for all test cases.

4.4 Summary

This chapter presents a fast direct solution technique for BIEs on locally-refined discretization, which is an critical component of numerical simulation of bodies in confined Stokes flow. The solver is an improved version of the authors' previous work in [50]. The major novelty is the new extended system formulation, of which the update matrix no longer contains any full-rank subblock of size $N_c \times N_c$. Thereby, the solver scales linearly with respect to both N_k and N_c . If \mathbf{A}_{pp} is compressed and inverted via a separate HBS, the solver also scales linearly with respect to N_p . When the part of the boundary refined is relatively small compared to the total size of the problem, the precomputation of the solver is much smaller than that of constructing an HBS solver for the new discretization from scratch. Another advantage of the new solver is that it can be easily applied to discretizations utilizing quadrature for weakly singular kernels, which allows single layer or combined field solution representation to be used for formulating the BIE.

Extending the solution technique to fast simulation of bodies in confined Stokes flow is an on-going project.

Chapter 5

An adaptive discretization technique for BIEs on the plane

The discretization of BIEs is one of the major difficulties for numerical solution to BIEs. This thesis considers the Nyström method, which uses quadrature to approximate the integral in the integral equation. This chapter constructs a panel-based adaptive discretization algorithm which returns a Gaussian panel quadrature that resolves the boundary density to an accuracy level specified by the user.

For BIEs defined on complex boundary geometries, the boundary density often requires lots of panels to resolve. However, this may still be the case even when the boundary geometry is simple. The source of the complexity may be the boundary conditions or the underlying physics. Thus, adaptive discretization techniques that focus only on resolving the geometric features such as arc length and corners may not capture the density well enough to reach the user desired tolerance.

A standard adaptive discretization algorithm for resolving the boundary density is an iterative process of solving a sequence of boundary densities on finer and finer mesh. Each new mesh is obtained by refining the previous one locally in the region where the boundary density is under-resolved. The process will terminate when the boundary density is resolved to the prescribed tolerance. Since each refinement level involves one global solve for the boundary density, the computational cost of the standard approach is too high to be practical. The adaptive discretization scheme presented in this chapter replaces the global solves of the true density with local

updates of a new quantity that we call *the artificial density*. The new scheme is less expensive compared to the standard true density guided adaptive discretization but still maintains the nice properties for using the density rather than geometric quantities as the guiding function for the adaption.

This chapter is organized as follows. Section 5.1 introduces notation and motivates the technique. Section 5.2 describes a standard adaptive discretization algorithm which solves for the true boundary density on each intermediate mesh. Section 5.3 is dedicated to the new adaptive algorithm which uses the artificial density defined in Section 5.3.1. Section 5.4 applies both the standard algorithm and the new algorithm to a collection of boundary geometries to show performance. Finally, section 5.5 concludes the chapter.

5.1 Introduction

To introduce notation, consider the BIE

$$\lambda\sigma(\mathbf{x}) + \int_{\Gamma} K(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) dy = f(\mathbf{x}), \quad (5.1.1)$$

where λ denotes a scalar and $K(\mathbf{x}, \mathbf{y})$ denotes a kernel function. Γ is a simple closed curve, f is the given boundary data, and σ is the unknown boundary charge density. In this chapter, we assume the boundary curve Γ is parameterized by $\gamma(t)$ for $t \in [0, T]$ and $\{\omega_j, t_j\}_{j=1}^N$ gives a quadrature rule (weights and nodes) for integrating functions on $[0, T]$. The BIE (5.1.1) can then be approximated via the Nyström method

$$\lambda\sigma(\mathbf{x}) + \sum_{j=1}^N \omega_j K(\mathbf{x}, \gamma(t_j)) \sigma(\gamma(t_j)) |\gamma'(t_j)| = f(\mathbf{x}). \quad (5.1.2)$$

Seeking the solution to (5.1.2) at the quadrature nodes $\{\gamma(t_j)\}_{j=1}^N$ yields the linear system

$$\mathbf{A}\sigma = \left(\lambda\mathbf{I} + \hat{\mathbf{K}}\right)\sigma = \mathbf{f}, \quad (5.1.3)$$

where \mathbf{I} is the identity matrix, σ is the unknown boundary density evaluated at the quadrature nodes, $\hat{\mathbf{K}}$ is a matrix with entry $\hat{\mathbf{K}}_{ij} = \omega_j K(\gamma(t_i), \gamma(t_j)) |\gamma'(t_j)|$, and \mathbf{f} is the given boundary data evaluated at the quadrature nodes.

5.1.1 Existing adaptive discretization strategies in literature

To motivate the proposed algorithm, this section discusses two common types of adaptive discretization strategies for BIEs and their advantages and shortcomings. Generally speaking, there are two different approaches to adaptively determine a discretization based on the quantity used to guide the adaptive process:

- *The purely geometric approach* that adaptively refines the discretization until geometric properties such as arc length and curvature are resolved to a prescribed tolerance.
- *The boundary density based approach* that solves for the boundary density and adaptively refine the discretization until the boundary density is resolved to a prescribed tolerance.

The purely geometric approach is simple and numerically efficient, since the geometric properties are often easy to evaluate and compare on different meshes. One example of such algorithms is the adaptive discretization given in [49] for handling multi-connected geometries. However, for certain problems where the boundary geometry is simple but the boundary density is more complex due to the boundary data or underlying physics, resolving geometric properties to high accuracy does not

guarantee the solution to the problem reaches the user desired accuracy level. Let's look at two examples to illustrate the disadvantages of using purely geometric quantities to guide an adaptive discretization. Consider the Laplace-Dirichlet problem with the second kind BIE formulation on two different ellipse geometries shown in Figure 5.1.1 (a) and 5.1.2 (a) with different boundary conditions due to a point source charge placed at different locations. The boundary of each ellipse is discretized with 16th order Gaussian panel quadrature. The accuracy of the arc length, curvature, and boundary density are plotted against the number of panels in Figure 5.1.1 (c) and 5.1.2 (c). The panels are uniformly refined in these two examples for simplicity of demonstration. For the ellipse that is close to a circle in Figure 5.1.1(a), the arc length and curvature can be resolved with relatively small number of Gaussian panels. An exterior source charge is placed relatively close to the boundary of the ellipse. The boundary density plotted in Figure 5.1.1 (d) has a (negative) peak in the region close to the source charge, roughly at $t \approx \pi/2 \approx 1.57$ in parameterization space. More panels are needed to resolve the boundary density than are needed to resolve the arc length and curvature for a given tolerance because of this peak behavior. For the ellipse with high eccentricity in Figure 5.1.2(a), the arc length and curvature are more complicated than that for the first ellipse example and require more panels to resolve. An exterior source charge is placed very far away from the ellipse, and the boundary density is rather simple and can be resolved with less number of panels than the arc length or the curvature.

Consider another more realistic example in the context of acoustic scattering. The double-disc geometry illustrated in Figure 5.1.3 is hit with a plane wave with incident angle $\theta^{inc} = -\frac{\pi}{2}$. Figure 5.1.4 shows the boundary density (a, c, and e) and the real part of the scatter field (b, d, and f) solved for different wave number ω . While

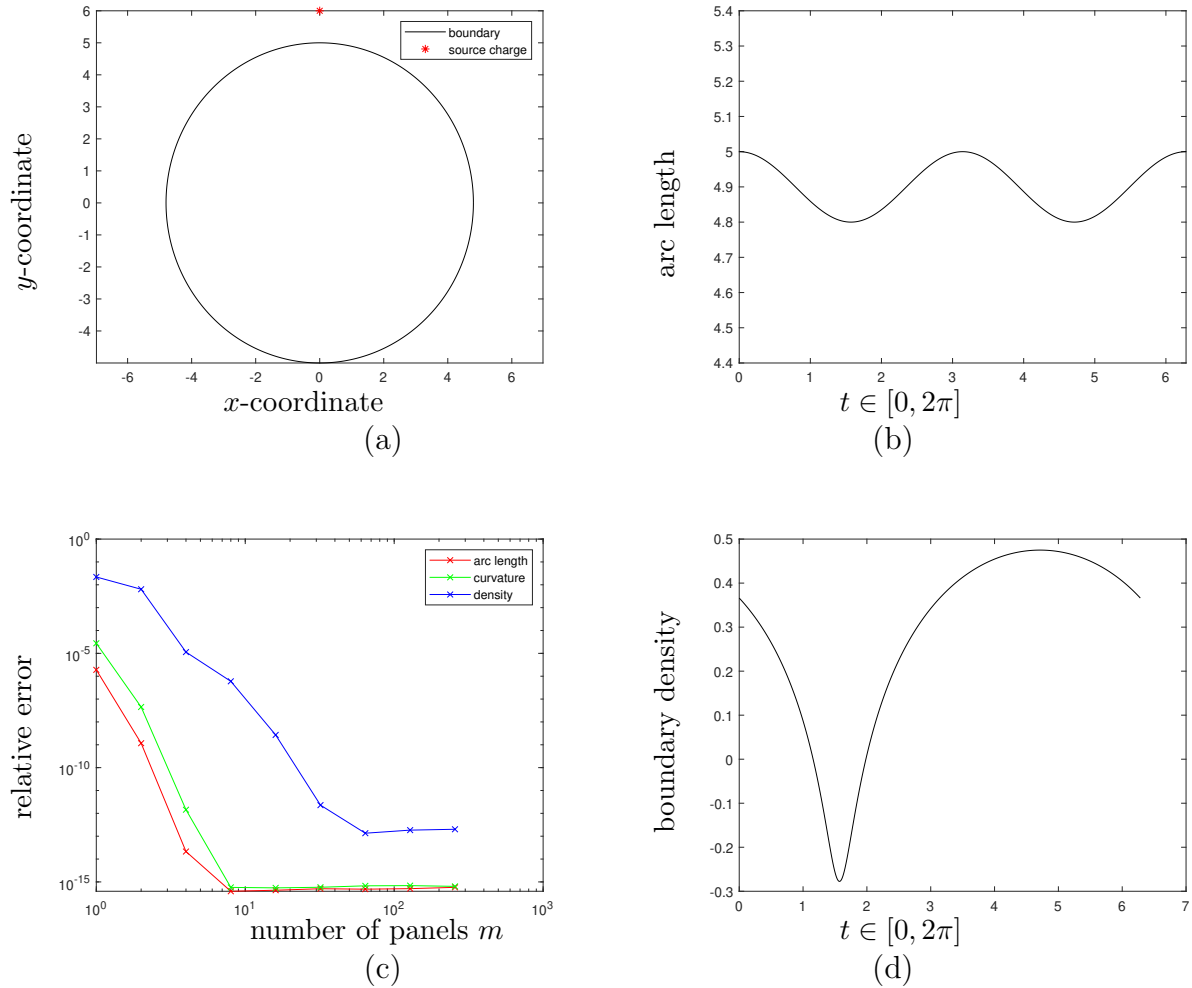


Figure 5.1.1 : An example where the geometric approach leads to under-discretization. The boundary geometry and the location of the exterior source are given in (a). (b) and (d) plot the arc length function and the boundary density on a reference grid with 300 Gaussian panels against parameterization variable t . The relative errors for the arc length, curvature, and boundary density for discretization with different number of panels are plotted on a log scale in (c). The errors of the density in (c) are evaluated by comparing the solved density to the density solved from the reference grid.

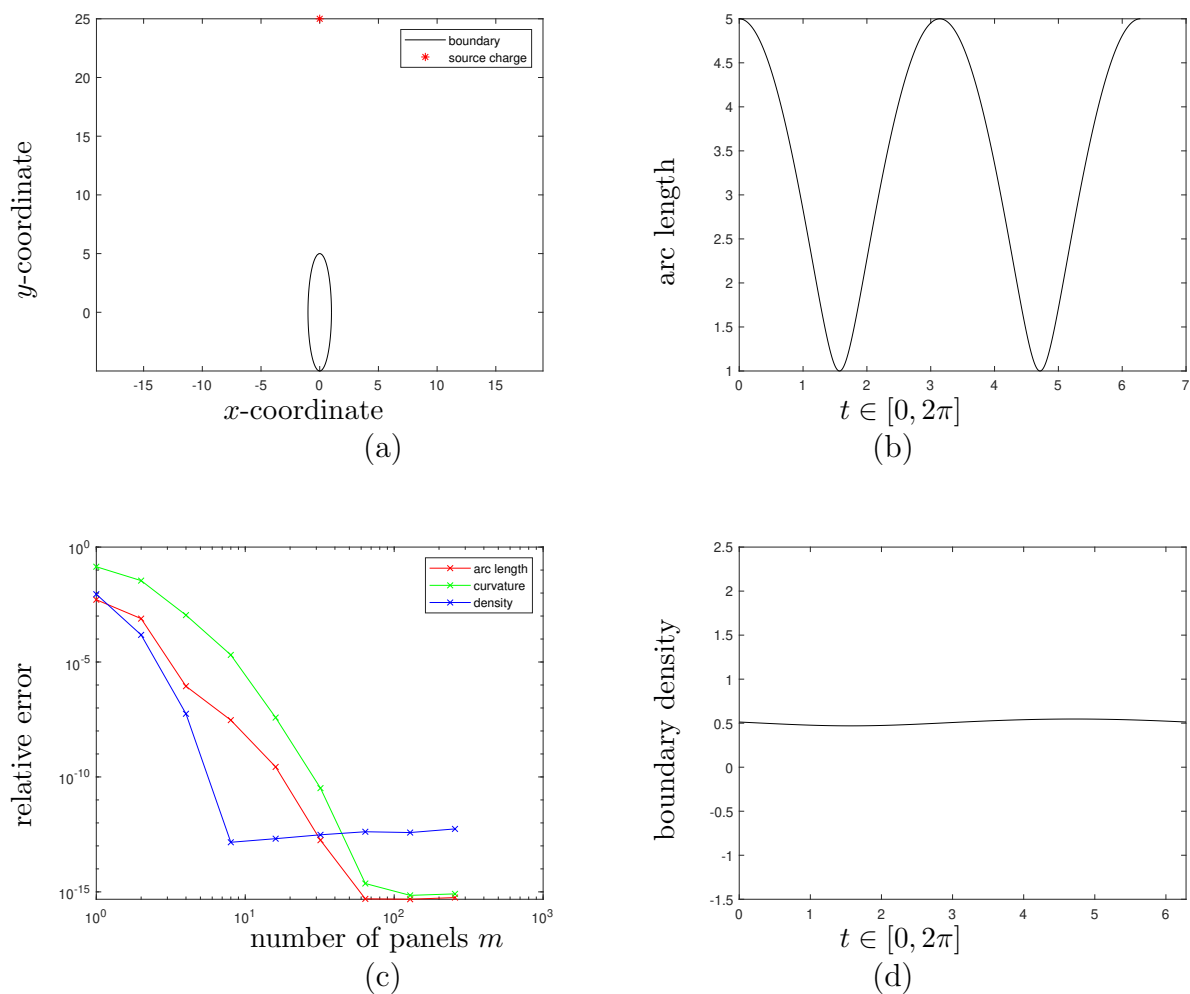


Figure 5.1.2 : An example where the geometric approach leads to over-discretization. The boundary geometry and the location of the exterior source are given in (a). (b) and (d) plot the arc length function and the boundary density on a reference grid with 300 Gaussian panels against parameterization variable t . The relative errors for the arc length, curvature, and boundary density for discretization with different number of panels are plotted on a log scale in (c). The errors of the density in (c) are evaluated by comparing the solved density to the density solved from the reference grid.

the boundary geometry is simple enough to be resolved with a few Gaussian panels, higher resolution is needed for capturing the density for larger wave numbers.

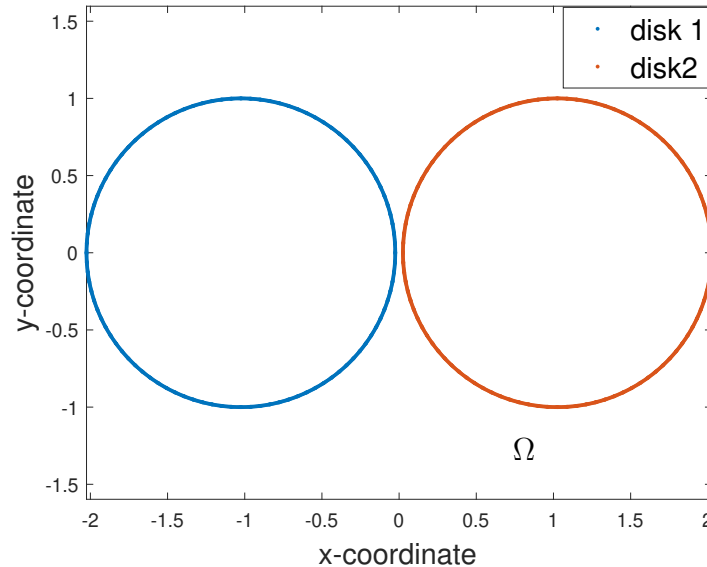
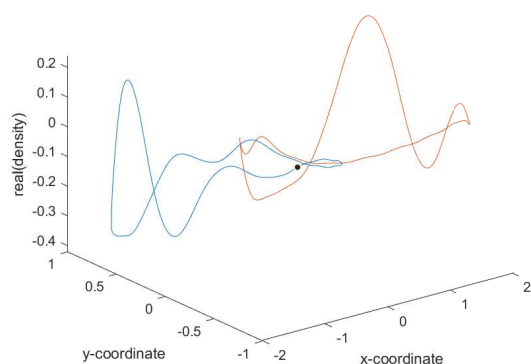
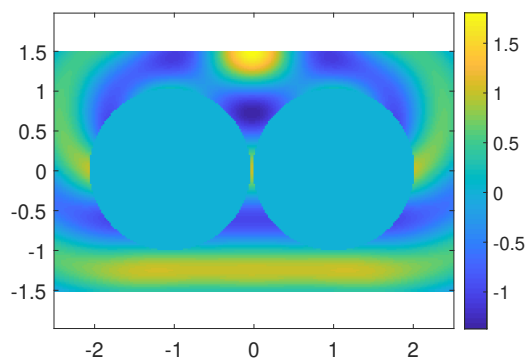


Figure 5.1.3 : Double disc geometry hit by plane wave with incident angle $\theta^{inc} = -\frac{\pi}{2}$. The discs have radius equal to 1, and the minimum distance between the two is chosen to be 0.05.

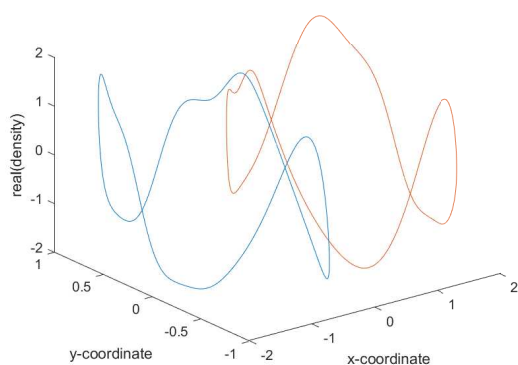
These simple demonstrations show that there is no guarantee on the accuracy of the density based upon how accurate the geometric property evaluations are. The density based approach, on the other hand, checks the density's resolution directly and ensures the user given tolerance is reached. [118] presents an adaptive algorithm which automatically resolves the density to the desired tolerance for an integral formulation of a stiff two-point boundary value problem. For each intermediate mesh, the algorithm looks at the coefficients of the local Chebyshev expansion of the density for each panel to determine which panels to be refined in the next level. The major drawback of the density based approach when applied to a BIE on the plane is that solving for the density on a collection of meshes could be costly.



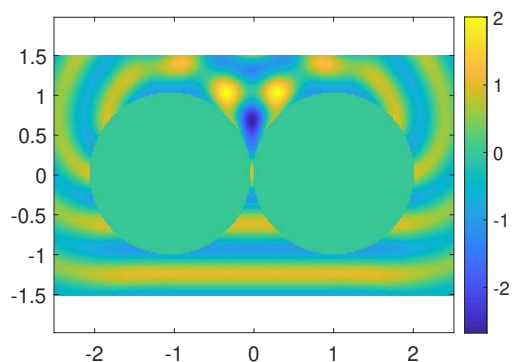
(a)



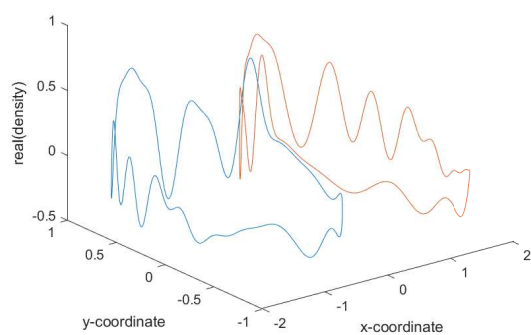
(b)



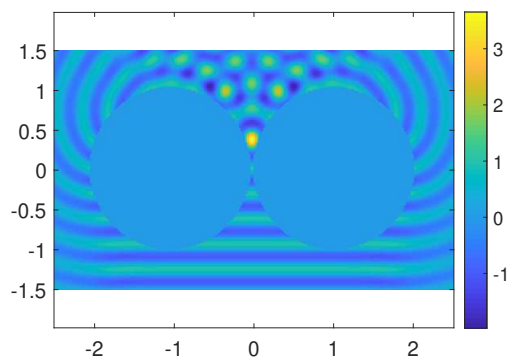
(c)



(d)



(e)



(f)

Figure 5.1.4 : The boundary density and real part of the scatter field for solving the Helmholtz BVP with the different wave numbers: (a-b) – $\omega = 5$, (c-d) – $\omega = 10$, (e-f) – $\omega = 20$. The boundary data is due to a plane wave with incident angle $\theta^{inc} = -\frac{\pi}{2}$. The boundary is discretized with 100 uniformly distributed 10th order Gaussian panels.

5.2 An adaptive scheme based on the true boundary density

This section introduces notation and presents a Gaussian panel based adaptive discretization technique which requires global solves for the boundary density for each intermediate mesh. The adaptive algorithm considered in this chapter is adaptive only in panel distributions, meaning that the number of panels m increases but the order of Gaussian quadrature on each panel n remains fixed. Given a user prescribed tolerance ϵ , the algorithm solves for the density $\sigma^{(0)}$ for the initial mesh $\mathcal{L}^{(0)}$ and calculates a convergence error. If the convergence error does not meet the tolerance ϵ , the algorithm picks out the under-resolved and over-resolved panels by looking at the panel-wise Legendre expansions of the density. The algorithm moves on to the next refinement level $k = 1$ by refining the under-resolved panels and merging the over-resolved panels. The whole process gets repeated until the tolerance ϵ is met for some refinement level k_{final} , then the corresponding mesh is returned as the final mesh, i.e. $\mathcal{L}_{\text{final}} = \mathcal{L}^{(k^*)}$.

For the rest of this chapter, let k denote the refinement level. For each $k \geq 0$, let $m^{(k)}$, $\mathcal{L}^{(k)}$, $\mathcal{L}_{\text{refine}}^{(k)}$ and $\mathcal{L}_{\text{merge}}^{(k)}$ denote the number of panels, the list of panels and the sublist of panels chosen to be refined or merged for the next refinement level.

5.2.1 Initialization

Since the true density is solved for each intermediate mesh, the algorithm can be initialized with simple discretization such as a single panel. However, this is not desirable, as the algorithm is very likely to refine almost all panels for the first several refinement levels, thus the total number of refinement levels required to reach the tolerance ϵ may be very large. Large number of refinement levels means lots of global solves for the true density and high numerical cost. One way to avoid this is to

add a initialization step and require the initial mesh to satisfy certain conditions. One option is to let the initial mesh be determined by applying adaptive Gaussian quadrature to integrate the arc length function to a tolerance $\epsilon_{init} \gg \epsilon$. In practice, we set ϵ_{init} to be 10^{-1} . The cost of this initialization step is minor, since the arc length function is very cheap to evaluate and integrate numerically. For complex geometries, the number of levels required to reach the tolerance ϵ in the main adaptive algorithm may be reduced greatly due to the initialization step.

5.2.2 Refining and coarsening condition

The adaptive scheme requires a refining condition and a coarsening condition to determine whether panels should be picked to be refined or merged in the next refinement level. We consider a panel-picking strategy which looks at the panel-wise Legendre expansions of the density and defines a refining and a coarsening condition based on the decay in the expansion coefficients for all panels in the current refinement level. This strategy is a generalization of the panel-picking strategy used in [118] to two-dimensional BIEs.

The idea behind this strategy is that if the density approximation converges, its local polynomial expansions on each panel should also converge. Thus, higher order coefficients of the expansions should be small in magnitude. For a two-point boundary value problem, it is shown in [118] that the converse of the idea is also true: if the local expansions of the density all decays enough, then the global solution converges. The proposed strategy generalizes the panel-picking method in [118] to handle BIEs on planar curves. In particular, Legendre expansions are used, since the BIE is discretized with Gaussian Legendre panels and the Legendre coefficients can be calculated from the point-wise density values easily.

Let panel $b \in \mathcal{L}^{(k)}$ and $\sigma_b^{(k)}$ be the density approximation restricted to b . Consider the $(n - 1)^{\text{th}}$ order local Legendre expansion of $\sigma_b^{(k)}$

$$\sigma_b^{(k)}(t) \approx \sum_{l=0}^{n-1} \alpha_{b,l}^{(k)} P_{b,l}^{(k)}(t)$$

where $P_{b,l}^{(k)}(t)$ denotes the l^{th} order Legendre polynomial defined on panel b , and $\alpha_{b,l}^{(k)}$ denotes the corresponding Legendre coefficient. The local Legendre coefficients $\alpha_b^{(k)} = [\alpha_{b,0}^{(k)}, \dots, \alpha_{b,n-1}^{(k)}]^T$ can be evaluated as

$$\alpha_b^{(k)} = M_n \sigma_b^{(k)}, \quad (5.2.1)$$

where the operator M_n is the linear map that takes function values at Legendre nodes to Legendre expansion coefficients. The formal definition of M_n is given in section A.4 of the Appendix. Note that $M_n \in \mathbb{R}^{n \times n}$ only depends on the order of the local quadrature/expansion and can be precomputed independent of the panel distribution.

The value

$$\phi_b^{(k)} := |\alpha_{b,n-2}^{(k)}| + |\alpha_{b,n-1}^{(k)} - \alpha_{b,n-3}^{(k)}| \quad (5.2.2)$$

is introduced to quantify the decay of the tail of the Legendre interpolant. A threshold value

$$\phi_{\text{threshold}}^{(k)} = \max_{b \in \mathcal{L}^{(k)}} \left\{ \frac{\phi_b^{(k)}}{2C} \right\}$$

is chosen based on a user specified value for the positive scalar C . Any panels in $\mathcal{L}^{(k)}$ satisfying the condition

$$\phi_b^{(k)} \geq \phi_{\text{threshold}}^{(k)}$$

is picked to be refined in the next level. Note that, for each level k , $\mathcal{L}_{\text{refine}}^{(k)}$ contains

at least one panel $b^* = \operatorname{argmax} \left\{ \frac{\phi_b^{(k)}}{2^C} \right\}$. A larger value of C results in more panels in $\mathcal{L}_{\text{refine}}^{(k)}$. In practice, we set $C = 4$. The value of the threshold function will monotonically decrease as k increases.

To avoid over-discretization, a coarsening condition is used to merge panels. A pair of panels b_1 and b_2 is called a sibling pair if they come from bisecting the same parent panel from a previous level. A sibling panel pair b_1 and b_2 is merged into one panel if

$$\phi_{b_1} + \phi_{b_2} < \frac{\phi_{\text{threshold}}^{(k)}}{2^n}, \text{ for } k \geq 1. \quad (5.2.3)$$

The coarsening condition for the initial mesh $\mathcal{L}^{(0)}$ is slightly more general: we do not assume the panels in the initial mesh form any “sibling” relationship and any pair of adjacent panels with equal panel size b_1 and b_2 are merged into one if the inequality in (5.2.3) is satisfied.

Algorithm 1 provides the pseudocode for the panel-picking algorithm. In practice, the coarsening condition is rarely satisfied for $k \geq 1$, i.e. $\mathcal{L}_{\text{merge}}^{(k)}$ is mostly empty for $k \geq 1$.

5.2.3 Convergence error and exit condition

Given a refinement level k and the corresponding panel list $\mathcal{L}^{(k)}$, a *convergence error* is defined to check if the density is already well-resolved by the current mesh. Let $\mathbf{A}^{(k)} \sigma^{(k)} = \mathbf{f}^{(k)}$ be the linear system of the density for the k^{th} level. In addition, define a *doubly-refined reference discretization* $\mathcal{L}^{(k, \text{double})}$ by refining every panels in $\mathcal{L}^{(k)}$ into two, and let $\mathbf{A}^{(k, \text{double})} \sigma^{(k, \text{double})} = \mathbf{f}^{(k, \text{double})}$ be the corresponding linear system. The

ALGORITHM 1 (Local Legendre expansion based panel-picking)

Given a scalar C , the panel list $\mathcal{L}^{(k)}$ for the k^{th} refinement level, and the corresponding density approximation $\sigma^{(k)}$, this algorithm determines $\mathcal{L}_{\text{refine}}^{(k)}$, the list of under-resolved panels in $\mathcal{L}^{(k)}$, and $\mathcal{L}_{\text{merge}}^{(k)}$, the list of over-resolved panels in $\mathcal{L}^{(k)}$.

Evaluate operator M_n in equation (5.2.1)

loop over panels b in $\mathcal{L}^{(k)}$,

 Compute Legendre coefficients $\alpha_b^{(k)} = M_n \sigma_b^{(k)}$.

 Evaluate $\phi_b^{(k)}$ as in (5.2.2).

end loop

Set $\phi_{\text{threshold}}^{(k)} = \max_{b \in \mathcal{L}^{(k)}} \left\{ \frac{\phi_b^{(k)}}{2^C} \right\}$.

Initialize $\mathcal{L}_{\text{refine}}^{(k)}$ and $\mathcal{L}_{\text{merge}}^{(k)}$ to be empty lists.

loop over panels b in $\mathcal{L}^{(k)}$,

if $\phi_b^{(k)} \geq \phi_{\text{threshold}}^{(k)}$,

 Add b to $\mathcal{L}_{\text{refine}}^{(k)}$.

end if

end loop

loop over sibling pairs b_1 and b_2 in $\mathcal{L}^{(k)}$,

% if $k = 0$, then loop over all pairs of consecutive panels same in size

if $\phi_{b_1}^{(k)} + \phi_{b_2}^{(k)} < \frac{\phi_{\text{threshold}}^{(k)}}{2^n}$,

 Add b_1 and b_2 to $\mathcal{L}_{\text{merge}}^{(k)}$.

end if

end loop

convergence error is defined as

$$e^{(k)} = \frac{\|\sigma^{(k)} - L^{(k)}\sigma^{(k,\text{double})}\|}{\|\sigma^{(k)} + L^{(k)}\sigma^{(k,\text{double})}\|}, \quad (5.2.4)$$

where $L^{(k)}$ is a block-diagonal interpolation operator. For each panel $b \in \mathcal{L}^{(k)}$, let b_1 and b_2 denote the two children panels of b in $\mathcal{L}^{(k,\text{double})}$. A local Lagrange interpolation operator from the two children panels to the parent panel, which interpolates the values $\sigma_{b_1}^{(k,\text{double})}$ and $\sigma_{b_2}^{(k,\text{double})}$ to quadrature points in b , constitutes one diagonal block of $L^{(k)}$. Note that there is no need to explicitly form the global interpolation operator $L^{(k)}$. Instead, the local interpolation operator can be pre-computed for the reference interval $[-1, 1]$ and applied to the corresponding sub-vectors of $\sigma^{(k,\text{double})}$ panel-wise.

The cost of evaluating $e^{(k)}$ is dominated by the two linear system solves for obtaining $\sigma^{(k)}$ and $\sigma^{(k,\text{double})}$. The adaptive process stops when the convergence error is smaller than the user prescribed tolerance. Namely, if $e^{(k^*)} \leq \epsilon$, then $\mathcal{L}^{(k^*)}$ is the final discretization and the algorithm terminates.

5.2.4 Full Algorithm and cost analysis

Algorithm 2 gives a pseudocode for the full algorithm. For each refinement level k , let $m^{(k)}$ denote the number of panels in $\mathcal{L}^{(k)}$. The cost of the algorithm per refinement level is dominated by the two linear system solves for evaluating $e^{(k)}$, which is $O([m^{(k)}n]^3)$ if done via dense linear algebra. As k increases, $m^{(k)}$ also increases. Thus the total cost of the algorithm is very high for problems that require many levels of refinement.

ALGORITHM 2 (Globally solved boundary density guided adaptive discretization)

Given a user prescribed tolerance ϵ , the max number of refinement level $MaxLevel$, and the initial discretization $\mathcal{L}^{(0)}$, this algorithm adaptively refines the discretization to resolve the boundary density to ϵ .

Initialize mesh $k = 0$.

Solve $\mathbf{A}^{(0)}\sigma^{(0)} = \mathbf{f}^{(0)}$.

Solve $\mathbf{A}^{(0,\text{double})}\sigma^{(0,\text{double})} = \mathbf{f}^{(0,\text{double})}$.

$$e^{(0)} = \frac{\|\sigma^{(0)} - L^{(0)}\sigma^{(0,\text{double})}\|}{\|\sigma^{(0)} + L^{(0)}\sigma^{(0,\text{double})}\|}.$$

if $e^{(0)} \leq \epsilon$

Return with $\mathcal{L}_{\text{final}} = \mathcal{L}^{(0)}$. Success.

else

Determine $\mathcal{L}_{\text{refine}}^{(0)}$ and $\mathcal{L}_{\text{merge}}^{(0)}$ by Algorithm 1.

Bisect panels in $\mathcal{L}_{\text{refine}}^{(0)}$ and merge pairs of panels in $\mathcal{L}_{\text{merge}}^{(0)}$ to get $\mathcal{L}^{(1)}$.

end if

while $k < MaxLevel$

$k = k + 1$.

Solve $\mathbf{A}^{(k)}\sigma^{(k)} = \mathbf{f}^{(k)}$.

Solve $\mathbf{A}^{(k,\text{double})}\sigma^{(k,\text{double})} = \mathbf{f}^{(k,\text{double})}$.

$$e^{(k)} = \frac{\|\sigma^{(k)} - L^{(k)}\sigma^{(k,\text{double})}\|}{\|\sigma^{(k)} + L^{(k)}\sigma^{(k,\text{double})}\|}.$$

if $e^{(k)} \leq \epsilon$

Return with $\mathcal{L}_{\text{final}} = \mathcal{L}^{(k)}$. Success.

else

Determine $\mathcal{L}_{\text{refine}}^{(k)}$ and $\mathcal{L}_{\text{merge}}^{(k)}$ by Algorithm 1.

Bisect panels in $\mathcal{L}_{\text{refine}}^{(k)}$ and merge pairs of panels in $\mathcal{L}_{\text{merge}}^{(k)}$ to get $\mathcal{L}^{(k+1)}$.

end if

end while loop

Return with $MaxLevel$ reached. Fail.

5.3 A new adaptive scheme

The density guided adaption in Algorithm 2 is numerically expensive for practical problems, since it requires a sequence of global density solves with growing number of unknowns as more and more panels get refined. This section proposes an adaptive discretization that replaces the global solves for the true density by local solves for an *artificial density* as defined in Section 5.3.1.

5.3.1 Locally updated artificial density

For the initial mesh, $\mathcal{L}^{(0)}$, the artificial density $\rho^{(0)}$ is simply defined to be equal to the globally solved boundary density, i.e. $\rho^{(0)} = \sigma^{(0)}$. Now we define the artificial density for $k \geq 1$ by giving the steps for updating $\rho^{(k-1)}$ to get $\rho^{(k)}$.

The artificial density will be updated in two rounds, and the updates for the first round are only for the entries that correspond to the panels chosen to be refined or merged. The first round sweeps through all panels in $\mathcal{L}_{\text{refine}}^{(k-1)} \cup \mathcal{L}_{\text{merge}}^{(k-1)}$. For each panel $b \in \mathcal{L}_{\text{refine}}^{(k-1)}$, let $C(b) = \{l \in \mathcal{L}^{(k-1)} : l \neq b\}$ denote the complement of b and let b_1 and b_2 denote the two children panels of b in $\mathcal{L}^{(k)}$. The artificial density restricted to b_1 and b_2 gets updated by solving the following $2n \times 2n$ local system

$$\left(\lambda \mathbf{I} + \begin{bmatrix} \hat{\mathbf{K}}_{b_1, b_1} & \hat{\mathbf{K}}_{b_1, b_2} \\ \hat{\mathbf{K}}_{b_2, b_1} & \hat{\mathbf{K}}_{b_2, b_2} \end{bmatrix} \right) \begin{bmatrix} \rho_{b_1}^{(k, \text{temp})} \\ \rho_{b_2}^{(k, \text{temp})} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{b_1} \\ \mathbf{f}_{b_2} \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{K}}_{b_1, C(b)} \\ \hat{\mathbf{K}}_{b_2, C(b)} \end{bmatrix} \rho_{C(b)}^{(k-1)}, \quad (5.3.1)$$

where matrix notation $\hat{\mathbf{K}}_{b_1, b_2}$ denotes the subblock of matrix $\hat{\mathbf{K}}$ in equation (5.1.3) with rows corresponding to points in b_1 and columns corresponding to points in b_2 , etc.

For each pair of panels b and c in $\mathcal{L}_{\text{merge}}^{(k-1)}$, let a be the panel formed by merging

b and c . The artificial density restricted to a gets updated by solving the following $n \times n$ local system

$$\left(\lambda \mathbf{I} + \hat{\mathbf{K}}_{a,a}\right) \rho_a^{(k,\text{temp})} = \mathbf{f}_a - \hat{\mathbf{K}}_{a,C(b \cup c)} \rho_{C(b \cup c)}^{(k-1)}, \quad (5.3.2)$$

where $C(b \cup c) = \{l \in \mathcal{L}^{(k-1)} : l \neq b \text{ and } l \neq c\}$ is the complement of $b \cup c$. The superscript notation “temp” means temporary, indicating this value for the artificial density is not final and will be updated in a second round.

Once all panels in $\mathcal{L}_{\text{refine}}^{(k-1)} \cup \mathcal{L}_{\text{merge}}^{(k-1)}$ are processed in the first round, the resulting artificial density $\rho^{(k,\text{temp})}$ is updated again to account for the fact that more than one panels get updated and that the updated artificial density on one panel affects the artificial density on a different panel. For each panel $p \in \mathcal{L}^{(k)}$, let $C(p) = \{l \in \mathcal{L}^{(k)} : l \neq p\}$ be the complement. The updated artificial density restricted to panel p is defined to be the solution of

$$\left(\lambda \mathbf{I} + \hat{\mathbf{K}}_{p,p}\right) \rho_p^{(k)} = \mathbf{f}_p - \hat{\mathbf{K}}_{p,C(p)} \rho_{C(p)}^{(k,\text{temp})}. \quad (5.3.3)$$

5.3.2 Exit condition

The collection of local solves for updating the artificial density replaces the global solves for the true boundary density $\mathbf{A}^{(k)} \sigma^{(k)} = \mathbf{f}^{(k)}$ for $k \geq 1$ in Algorithm 2. The next question is how to remove the global solves on the doubly refined reference mesh, i.e. $\mathbf{A}^{(k,\text{double})} \sigma^{(k,\text{double})} = \mathbf{f}^{(k,\text{double})}$ for $k \geq 1$, which is required for evaluating the convergence error defined by (5.2.4). This is done by introducing a new definition for $\rho^{(k,\text{double})}$ similar to the updating scheme for $\rho^{(k)}$ and a new exit condition. For the initial mesh, let $\rho^{(0,\text{double})} = \sigma^{(0,\text{double})}$ be obtained from a global solve. Then for

each refinement level $k \geq 1$, we can assume both $\rho^{(k-1)}$ and $\rho^{(k-1,\text{double})}$ are available. We then use the results from (5.3.1) and (5.3.2) to update the value of $\rho^{(k-1,\text{double})}$ to obtain $\rho^{(k,\text{double})}$. For each panel $p \in \mathcal{L}^{(k)}$, let p_1 and $p_2 \in \mathcal{L}^{(k,\text{double})}$ be the two children panels resulted from bisecting p . Let the restriction of $\rho^{(k,\text{double})}$ on p_1 and p_2 be updated by solving the following $2n \times 2n$ linear system

$$\left(\lambda \mathbf{I} + \begin{bmatrix} \hat{\mathbf{K}}_{p_1,p_1} & \hat{\mathbf{K}}_{p_1,p_2} \\ \hat{\mathbf{K}}_{p_2,p_1} & \hat{\mathbf{K}}_{p_2,p_2} \end{bmatrix} \right) \begin{bmatrix} \rho_{p_1}^{(k,\text{double})} \\ \rho_{p_2}^{(k,\text{double})} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{p_1} \\ \mathbf{f}_{p_2} \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{K}}_{p_1,C(p)} \\ \hat{\mathbf{K}}_{p_2,C(p)} \end{bmatrix} \rho_{C(p)}^{(k,\text{temp})}. \quad (5.3.4)$$

We introduce a new quantity $\eta^{(k)}$ which measure the relative improvement in the artificial density

$$\eta^{(k)} = \frac{\|\rho^{(k)} - L^{(k)}\rho^{(k,\text{double})}\|}{\|\rho^{(k)} + L^{(k)}\rho^{(k,\text{double})}\|}. \quad (5.3.5)$$

It is used in place of $e^{(k)}$ in the adaptive process for determining when the adaptive process terminates. The algorithm terminates with $\mathcal{L}^{(k^*)}$ as the final mesh if $\eta^{(k^*)} \leq \epsilon$.

5.3.3 Full algorithm and efficient implementation

The pseudocode for the new algorithm is given in Algorithm 3. Unlike the adaptive algorithm which solves for the true density $\sigma^{(k)}$ for each refinement level, the artificial density approach requires only one global solve for the initial mesh and updates locally for $k \geq 1$. Thus the initial mesh must be reasonable and a nontrivial initialization step, such as the arc length approach described in Section 5.2.1, is recommended.

Remark 5.3.1 Algorithm 3 only produces the mesh. The linear system for the final mesh $\mathcal{L}_{\text{final}}$ needs to be constructed and solved to obtain a solution to the BIE. For problems defined on complex geometries, fast solution algorithm, such as FMM accelerated iterative solvers, should be used, since the global systems might be too

large to solve densely.

Dense evaluation and inversion of the local systems in equation (5.3.1), (5.3.2), (5.3.3), and (5.3.4) for updating $\rho^{(k)}$ and $\rho^{(k,\text{double})}$ is efficient, since the matrices are small. Many of the local inverses can be reused for consecutive refinement levels, i.e. k and $k + 1$. This fact is exploited in the implementation. New matrix evaluation and inversion is only done when new panels get refined.

In the efficient implementation, we construct a collection of local inverses of size $n \times n$ for each panel in $\mathcal{L}^{(0)} \setminus \mathcal{L}_{\text{refine}}^{(0)}$ and local operators and inverses of size $2n \times 2n$ for each sibling pair of panels in $\mathcal{L}^{(0,\text{double})}$ as an extra step for the initial mesh $\mathcal{L}^{(0)}$. For each refinement level $k \geq 1$, the local inverses in equation (5.3.1) are available from the local inverses stored for $\mathcal{L}^{(k-1,\text{double})}$. The inverses of the local operator in equation (5.3.2) are available from the stored information for $\mathcal{L}^{(k-2)}$ if $k \geq 2$. The local inverses in (5.3.3) are available either from the size $n \times n$ local inverses for $\mathcal{L}^{(k-1)}$ and $\mathcal{L}^{(k-2)}$ or can be obtained by inverting submatrices from the $2n \times 2n$ self operators stored for $\mathcal{L}^{(k-1,\text{double})}$. New local operator construction and/or inversion is required only for the following three scenarios:

- (1) New operators of size $n \times n$ are constructed and inverted for panels in $\mathcal{L}_{\text{merge}}^{(0)}$ to solve (5.3.2).
- (2) Submatrices of size $n \times n$ are extracted and inverted when solving (5.3.3) for each $p \in \mathcal{L}^{(k)} \setminus \mathcal{L}^{(k-1)}$.
- (3) New operators of size $2n \times 2n$ are constructed and inverted for each panel in $\mathcal{L}^{(k)} \setminus \mathcal{L}^{(k-1)}$ for $k \geq 1$ on the doubly refined reference mesh.

Thus the cost of evaluating and inverting the local systems is $O([m^{(k)} - m^{(k-1)}]n^3)$ for

each level $k \geq 1$. The total amount of memory required for storing the local operators is $O(\bar{m}n^2)$, where $\bar{m} = \max_k m^{(k)}$.

The major remaining cost of the new algorithm is evaluating the right-hand-side potential of the form $\hat{\mathbf{K}}_{p,C(p)}$ in equation (5.3.1), (5.3.2), (5.3.3), and (5.3.4). When memory is not a major concern, it is worthwhile to store and update the right-hand-side operator $\hat{\mathbf{K}}_{p,C(p)}$ for each panel p in the current mesh $\mathcal{L}^{(k)}$. The operator is first used to solve equation (5.3.3) for the k^{th} level. Let $L_p^{(k)}$ be a local Lagrange interpolation operator mapping values at points in p to points in the two children panels p_1 and p_2 of p on the doubly refined reference mesh $\mathcal{L}^{k,\text{double}}$. Then $\hat{\mathbf{K}}_{p,C(p)}$ is reused to approximate the right-hand-side potential for equation (5.3.4)

$$\begin{bmatrix} \hat{\mathbf{K}}_{p_1,C(p)} \\ \hat{\mathbf{K}}_{p_2,C(p)} \end{bmatrix} \rho_{C(p)}^{(k,\text{temp})} \approx L_p^{(k)} \hat{\mathbf{K}}_{p,C(p)} \rho_{C(p)}^{(k,\text{temp})}.$$

The right-hand-side potential in (5.3.1) for the $(k+1)^{\text{th}}$ level can be approximated similarly by reusing $\hat{\mathbf{K}}_{p,C(p)}$ stored for the k^{th} level. The memory cost for storing the off-diagonal operators is $O(\bar{m}^2 n^2)$, and the computational complexity per refinement level for evaluating the right-hand-side potential is $O([m^{(k)}n]^2)$.

Remark 5.3.2 The right-hand-side potential evaluation can be accelerated by an FMM. The memory cost drops down to $O(\bar{m}n)$ and the computational cost drops down to $O(m^{(k)}n)$. This acceleration is not included in the current implementation.

The cost of the initial refinement level is $O([m^{(0)}n]^3)$. Let k^{final} be the last refinement level, the total cost per level of the new algorithm is $O([m^{(k)}n]^2 + [m^{(k)} - m^{(k-1)}]n^3)$ for intermediate level $0 < k < k^{\text{final}}$, which is much smaller than the cost of the true density guided algorithm in section 5.2.

ALGORITHM 3 (Locally updated artificial density guided adaptive discretization)

Given a user prescribed tolerance ϵ , the max number of refinement level $MaxLevel$, and the initial discretization $\mathcal{L}^{(0)}$, this algorithm adaptively refines the discretization and produces a final discretization which resolves the true boundary density to ϵ .

Initialize mesh $k = 0$.

Solve $\mathbf{A}^{(0)}\sigma^{(0)} = \mathbf{f}^{(0)}$.

Solve $\mathbf{A}^{(0,\text{double})}\sigma^{(0,\text{double})} = \mathbf{f}^{(0,\text{double})}$.

$$e^{(0)} = \frac{\|\sigma^{(0)} - L^{(0)}\sigma^{(0,\text{double})}\|}{\|\sigma^{(0)} + L^{(0)}\sigma^{(0,\text{double})}\|}.$$

if $e^{(0)} \leq \epsilon$

Return with $\mathcal{L}_{\text{final}} = \mathcal{L}^{(0)}$. Success.

else

Determine $\mathcal{L}_{\text{refine}}^{(0)}$ and $\mathcal{L}_{\text{merge}}^{(0)}$ by Algorithm 1.

Bisect panels in $\mathcal{L}_{\text{refine}}^{(0)}$ and merge pairs of panels in $\mathcal{L}_{\text{merge}}^{(0)}$ to get $\mathcal{L}^{(1)}$.

Set $\rho^{(0)} = \sigma^{(0)}$ and $\rho^{(0,\text{double})} = \sigma^{(0,\text{double})}$.

end if

while $k < MaxLevel$

$k = k + 1$.

loop over $b \in \mathcal{L}_{\text{refine}}^{(k-1)}$

Let b_1 and $b_2 \in \mathcal{L}^{(k)}$ be the two children panel for b .

Solve (5.3.1) to update $\rho_{b_1}^{(k,\text{temp})}$ and $\rho_{b_2}^{(k,\text{temp})}$.

end loop

loop over sibling pair b and c in $\mathcal{L}_{\text{merge}}^{(k-1)}$

Let a be the panel created by merging b and c .

Solve (5.3.2) to update $\rho_a^{(k,\text{temp})}$.

end loop

loop over $p \in \mathcal{L}^{(k)}$

Solve (5.3.3) for $\rho_p^{(k)}$.

Let p_1 and $p_2 \in \mathcal{L}^{(k,\text{double})}$ be the two children panel for p .

Solve (5.3.4) for $\rho_{p_1}^{(k,\text{double})}$ and $\rho_{p_2}^{(k,\text{double})}$.

end loop

$$\eta^{(k)} = \frac{\|\rho^{(k)} - L^{(k)}\rho^{(k,\text{double})}\|}{\|\rho^{(k)} + L^{(k)}\rho^{(k,\text{double})}\|}.$$

if $\eta^{(k)} \leq \epsilon$

Return with $\mathcal{L}_{\text{final}} = \mathcal{L}^{(k)}$.

else

Determine $\mathcal{L}_{\text{refine}}^{(k)}$ and $\mathcal{L}_{\text{merge}}^{(k)}$ by Algorithm 1.

Bisect panels in $\mathcal{L}_{\text{refine}}^{(k)}$ and merge pairs of panels in $\mathcal{L}_{\text{merge}}^{(k)}$ to get $\mathcal{L}^{(k+1)}$.

end if

end while loop

Return with $MaxLevel$ reached. Fail.

5.4 Numerical results

This section illustrates the performance of the artificial density guided adaptive discretization algorithm when applied to BIEs defined on planar curves. The current implementation for both the new algorithm and the true density algorithm is in MATLAB and uses dense linear algebra (MATLAB's built-in backslash) for global density solves. All experiments were run on a dual 2.3 GHz Intel Xeon Processor E5-2695 v3 desktop workstation with 256 GB of RAM.

The following values are defined and reported to help analyze the results of the algorithms:

- k_{final} : total number of refinement levels, i.e. refinement levels, required to reach $\mathcal{L}_{\text{final}}$.
- $m^{(k_{\text{final}})}$: number of panels in $\mathcal{L}_{\text{final}}$.
- T : total time in seconds used by the algorithms to generate the final mesh.
- E_{interior} : the relative error in solution to the BVP evaluated at the interior target locations. In the case of multiple interior target locations, the average is reported.
- $\eta^{(k_{\text{final}})}$: the artificial density convergence measure as defined in (5.3.5) for the final refinement level. This value is only reported for the artificial density approach.
- $e^{(k_{\text{final}})}$: the convergence error as defined in (5.2.4) for the final refinement level. For the new algorithm, this is not part of the algorithm and is calculated as an extra step once the algorithm terminates.

5.4.1 Comparison with the true density guided algorithm

To demonstrate the efficiency of the new algorithm, we first compare the algorithm with the true density approach by applying the two algorithms to an interior Laplace problem with Dirichlet boundary conditions.

Consider the butterfly geometry shown in Figure 5.4.1(a), which is created by smoothing a butterfly shaped polygon via the technique in [2]. We consider a test BVP with a known solution given by

$$\mathbf{u}_{\text{ext}}(\mathbf{x}) \sum_{i=1}^{N_{\text{src}}} G(\mathbf{x}, \mathbf{y}_i) q_i,$$

where $\{\mathbf{y}_i, q_i\}_{i=1}^{N_{\text{src}}}$ gives the location and charge value of a collection of exterior source charges.

A mesh with two equal size panels that integrates the arc length function to $\epsilon_{\text{init}} = 10^{-1}$ is chosen to be the initial mesh. Several different values for ϵ are considered to verify the correctness of the new algorithms and compare its efficiency against the true density approach.

The results are reported in Table 5.1. For all tested values of ϵ , the final mesh produced by the artificial density approach resolves the true density to more digits than the desired accuracy. For example, the $\epsilon = 10^{-6}$ row in Table 5.1(a) shows that the final mesh has $e^{(k_{\text{final}})} = 2.5 \times 10^{-9} \ll \epsilon = 10^{-6}$. This indicates the resulting mesh meets the user-prescribed tolerance but is not optimal. For all test cases, the T column shows that the computational cost of the new algorithm is much smaller than that of the true density approach. When high accuracy is desired, the new algorithm cuts down the cost of generating the mesh by a factor of 7.

As a reference, we also solve the same BVP by uniform panel distribution with

increasing number of panels. The idea here is to show that uniform panel distribution is not efficient in resolving the boundary density for complex geometries. The number of panels required for a uniform panel distribution to reach a comparable level of accuracy as the adaptive algorithms is much larger, and thus uniformly refining every existing panels to reach the desired tolerance is not practical for most problems. We report the number of panels m , the convergence error e as defined in (5.3.5), and the solution error at interior target locations E_{interior} in Table 5.2. The results show that the uniform mesh does a poor job in resolving the density when compared to the adaptive algorithms. With about 400 panels, the adaptive algorithms gives convergence error smaller than 1.0×10^{-12} and interior solution error smaller than 1.0×10^{-14} , while the uniform mesh results in a convergence error of 3.9×10^{-8} and interior solution error of 1.7×10^{-9} . The inefficiency of uniform panel distribution in resolving the boundary density will be even more significant if the geometry is more challenging, e.g. the Fallopian tube piece geometry in Figure 5.4.4.

5.4.2 Same geometry with different right-hand-side functions

Special care may be required to resolve the boundary density even though the boundary geometry is simple. One possible source of the complexity is the right-hand-side function. In this section, we consider two Laplace problems defined on the same fish geometry shown in Figure 5.4.2 (a) but with Dirichlet data generated by a collection of exterior source charges placed at different locations indicated by (i) and (ii) in Figure 5.4.2 (a). The right-hand-side function $f(\mathbf{x})$ corresponding to case (i) and (ii) is plotted in Figure 5.4.2 (b). The right-hand-side function $f(\mathbf{x})$ for case (ii) has a sharper peak near $t = \frac{\pi}{2}$ than that for case (i). Thus, we expect more panels are needed in the peak region for case (ii), which corresponds to the fish head on the

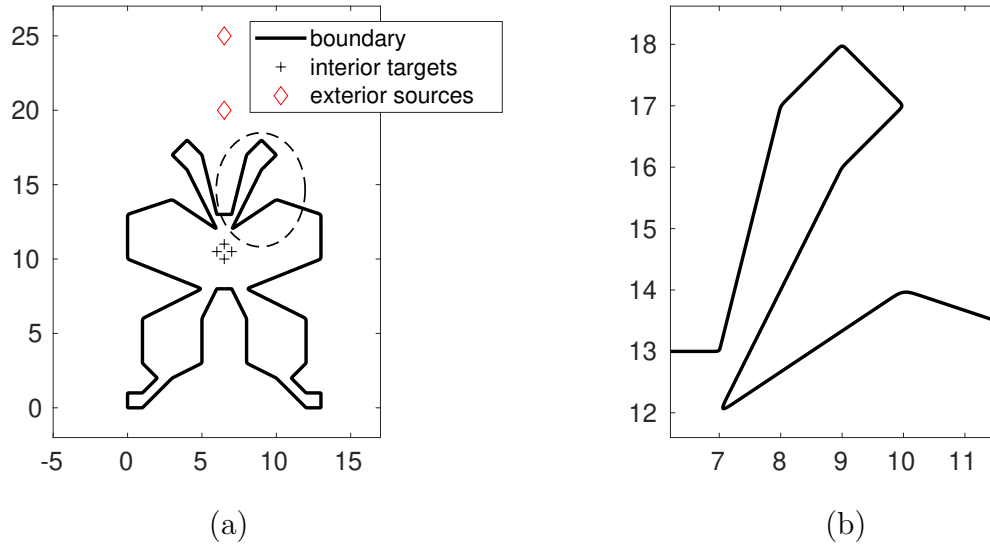


Figure 5.4.1 : (a) A butterfly geometry with interior target locations and exterior source locations. The geometry is generated by applying a corner smoothing scheme present in [2] to a butterfly shaped polygon. (b) zooms in the circled region in (a) to show that there is no sharp corners for this geometry.

| ϵ | k_{final} | $m^{(k_{\text{final}})}$ | $\eta^{(k_{\text{final}})}$ | $e^{(k_{\text{final}})}$ | E_{interior} | T |
|------------|--------------------|--------------------------|-----------------------------|--------------------------|-----------------------|-----|
| 1e-6 | 12 | 194 | 1.0e-7 | 2.5e-9 | 5.1e-14 | 1.9 |
| 1e-8 | 14 | 254 | 3.3e-10 | 2.5e-12 | 1.0e-14 | 2.4 |
| 1e-10 | 15 | 312 | 5.4e-11 | 2.4e-13 | 2.0e-15 | 3.3 |
| 1e-12 | 17 | 404 | 7.8e-13 | 4.8e-14 | 2.2e-15 | 4.6 |

(a) new algorithm

| ϵ | k_{final} | $m^{(k_{\text{final}})}$ | $e^{(k_{\text{final}})}$ | E_{interior} | T |
|------------|--------------------|--------------------------|--------------------------|-----------------------|------|
| 1e-6 | 13 | 172 | 3.0e-7 | 3.3e-9 | 7.0 |
| 1e-8 | 16 | 256 | 1.3e-9 | 2.5e-14 | 16.7 |
| 1e-10 | 18 | 368 | 5.0e-13 | 7.9e-15 | 35.7 |
| 1e-12 | 18 | 368 | 5.0e-13 | 7.9e-15 | 35.7 |

(b) true density guided algorithm

Table 5.1 : Results for applying (a) the artificial density algorithm and (b) the true density algorithm to a Laplace BVP defined on the butterfly geometry given in Figure 5.4.1. All tests start with a two-panel initial mesh which integrates the arc length function to one digit of accuracy.

| m | e | E_{interior} |
|------|---------|-----------------------|
| 100 | 6.2e-3 | 3.4e-4 |
| 200 | 8.0e-6 | 8.1e-7 |
| 400 | 3.9e-8 | 1.7e-9 |
| 800 | 3.1e-11 | 5.3e-14 |
| 1000 | 1.2e-12 | 2.6e-15 |

Table 5.2 : Results for solving a Laplace BVP defined on the butterfly geometry by uniform Gaussian panel quadrature.

boundary geometry. The tolerance is set to be $\epsilon = 10^{-8}$ and the initial mesh contains only two panels of equal size which integrates the arc length function to $\epsilon_{\text{init}} = 10^{-1}$. The final mesh for the two cases presented in Figure 5.4.3 matches our expectation: the final meshes are the same except that the mesh for case (ii) has more panels near the fish head.

5.4.3 Complex geometry

This section applies the new algorithm to a Laplace BVP defined on a more challenging geometry shown in Figure 5.4.4. This geometry is created by applying the corner smoothing scheme in [2] to a polygon formed by a collection of 621 data points sampled from the Fallopian tube cross section image (Figure 4.0.1) provided in [1]. We apply the new algorithm with two choices of desired tolerance: $\epsilon = 10^{-6}$ and $\epsilon = 10^{-8}$. The initial mesh contains 737 panels and integrates the arc length function to $\epsilon_{\text{init}} = 10^{-1}$. The time and error results are reported in Table 5.3. It takes the new algorithm 231.5 seconds to produce a mesh which resolves the true boundary density to $e^{(k_{\text{final}})} = 1.2 \times 10^{-10}$ when the desired tolerance is set to $\epsilon = 10^{-8}$.

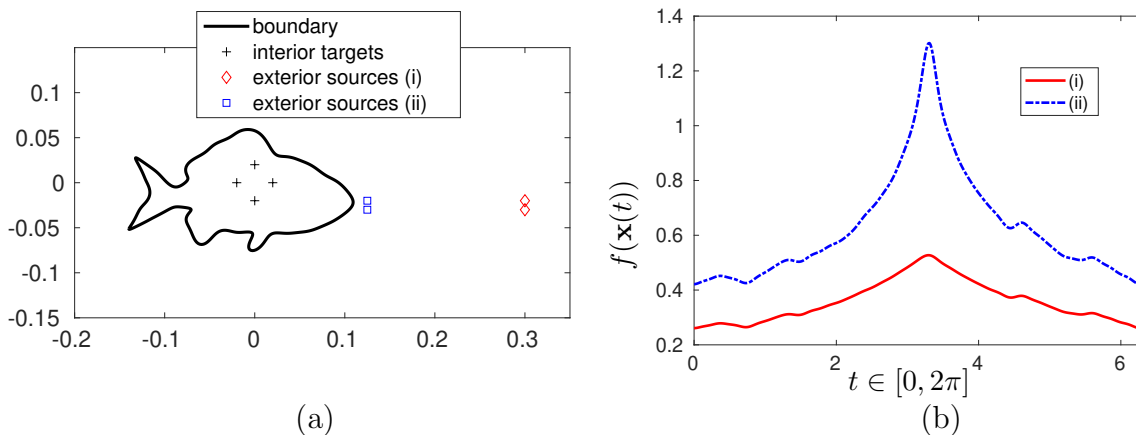


Figure 5.4.2 : (a) A fish geometry with interior target locations and two choices of exterior source locations (i) and (ii) for the right-hand-side function $f(\mathbf{x})$. (b) The right-hand-side function $f(\mathbf{x}(t))$ plotted against parameterization variable t for the two choices of exterior source locations (i) and (ii).

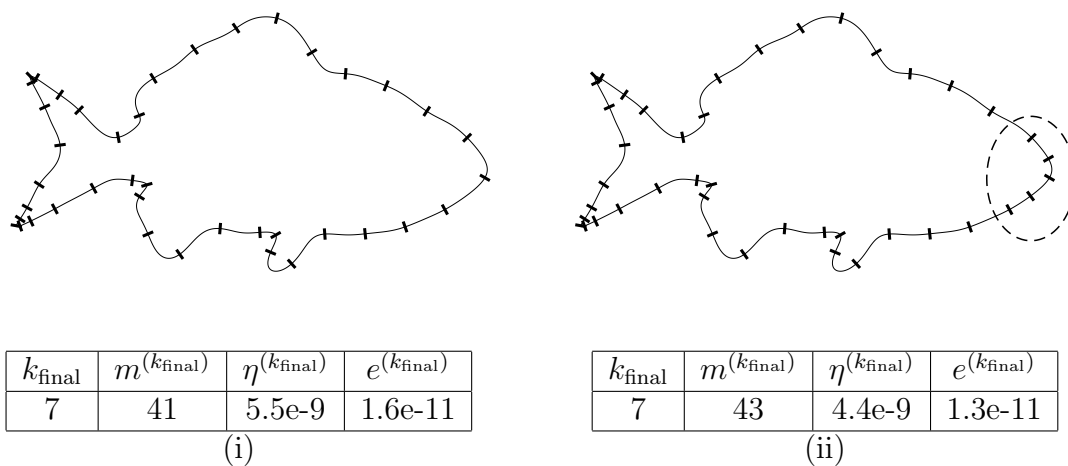


Figure 5.4.3 : The final mesh produced by the new algorithm for $f(\mathbf{x})$ generated by choices of exterior source locations (i) and (ii). More panels are placed at the fish head (circled region) for (ii). Desired accuracy is set to $\epsilon = 10^{-8}$, and the initial mesh integrates the arc length function to $\epsilon_{\text{init}} = 10^{-1}$.

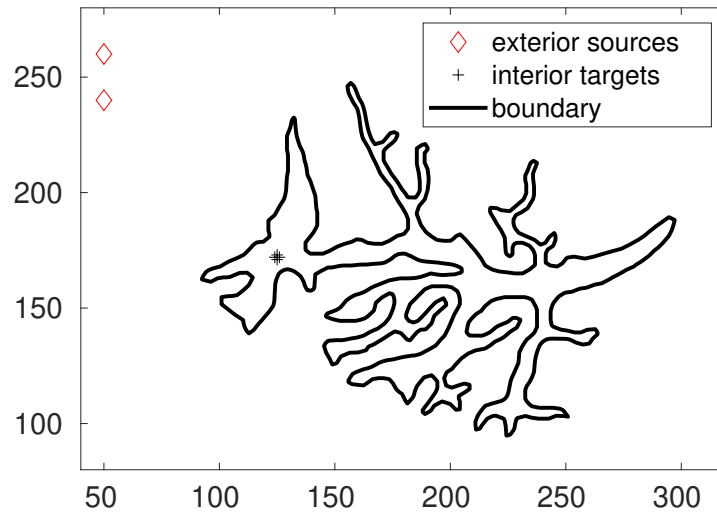


Figure 5.4.4 : The “Fallopian tube piece” geometry.

| ϵ | k_{final} | $m^{(k_{\text{final}})}$ | $\eta^{(k_{\text{final}})}$ | $e^{(k_{\text{final}})}$ | E_{interior} | T |
|------------|--------------------|--------------------------|-----------------------------|--------------------------|-----------------------|-------|
| 1e-6 | 15 | 2155 | 3.9e-7 | 5.2e-8 | 1.1e-9 | 150.4 |
| 1e-8 | 18 | 2701 | 7.9e-9 | 1.2e-10 | 4.9e-13 | 231.5 |

Table 5.3 : Results for applying the artificial density guided adaptive discretization algorithm to the “Fallopian tube piece” geometry.

5.5 Summary

This chapter presents a panel-based adaptive discretization technique for two-dimensional BIEs. The method introduces a new quantity, the artificial density, to guide the adaptive process. Small local linear systems are solved for each intermediate mesh to update the artificial density. Thus, the method is much less expensive than the standard true density guided adaptive discretization which requires global solves for the boundary density for each intermediate mesh.

Chapter 6

Conclusions

The goal of the proposed work is to build tools to expand the applicability of BIE formulations and fast direct solution techniques to a broader range of applications. The fast direct solver for quasi-periodic scattering in layered media and the fast direct solver for confined Stokes flow with locally refined discretization are driven by applications and tackle the bottleneck problems encountered by practitioners in the respective research fields. The last project addresses the difficulty associated with discretizing BIEs on complex boundary geometries and introduces an adaptive discretization scheme as a way to deal with this difficulty. The adaptive discretization scheme is not designed for a particular application and can be applied to lots of problems so that the resulting discretization achieves the user prescribed tolerance. This section highlights the work from each of the projects and concludes the thesis.

The first project considers the quasi-periodic scattering problem in layered media. The proposed solution technique uses an existing BIE formulation from [48] and solves a block tridiagonal linear system, of which the major block is inverted via a collection of fast direct solvers each for an individual layer interface. The resulting solver for the entire scatterer structure scales linearly with respect to the number of unknowns on the interfaces, and can be modified with a small cost if a few layers change in shape or material property. This piece of work is published in [119] and has peaked the interest of people in a variety of communities. Currently, Owen Miller at Yale is integrating this solver into his optimal design framework for developing composite materials. The

solver can easily handle interfaces with corners and high contrast media, both of these are challenging for existing numerical methods, making it difficult for practitioners to model composite materials.

The second project focuses on the numerical simulation of bodies in confined Stokes flow. The major bottleneck for a BIE formulation based simulation is solving the BIE on the confining wall. Existing out-of-box fast direct solvers are efficient for this problem only if the discretization of the confining wall does not change over time, while the proposed fast direct solver is designed to work with time-evolving discretization refinements. Preliminary experiments show that for one time step local refinement and one right-hand-side solve, the new solver can be 4-5 times faster than building an HBS solver for the new discretization from scratch. This will result in hundreds times speed-up for simulations with multiple time steps and/or multiple right-hand-sides. Part of the work shown for the project is published in [120]. An on-going project is to incorporate the solver into a time stepping scheme for modeling the dynamics of cilia structures in complex geometries. This is a collaboration with the authors of [1].

Finally, the last project develops a panel based adaptive discretization technique for two-dimensional BIEs. The proposed algorithm introduces the idea of artificial density to guide the adaptive procedure. This prevents the potential over-discretization or under-discretization when compared to adaptive discretization algorithms guided by purely geometric quantities such as arc length and curvature and avoids the expensive global solves when compared to the standard density guided adaptive discretization. Future work includes extending the algorithm to more complicated two-dimensional BIEs, such as the Helmholtz equations on complex geometries.

Bibliography

- [1] H. Guo, H. Zhu, and S. Veerapaneni, “Simulating cilia-driven mixing and transport in complex geometries,” *Physical Review Fluids*, vol. 5, p. 053103, May 2020.
- [2] C. L. Epstein and M. O’Neil, “Smoothed corners and scattered waves,” *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2665–A2698, 2016.
- [3] R. Kress, *Linear Integral Equations*, vol. 82. Springer-Verlag New York, 3 ed., 2014.
- [4] K. E. Atkinson, *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 1997.
- [5] D. Colton, *Partial Differential Equations: An Introduction*. Random House, 1988.
- [6] G. Hsial and W. L. Wendland, *Boundary Integral Equations*, vol. 164 of *0066-5452*. Springer-Verlag Berlin Heidelberg, 1 ed., 2008.
- [7] K. E. Atkinson, “The numerical solution of fredholm integral equations of the second kind with singular kernels,” *Numerische Mathematik*, vol. 19, no. 3, pp. 248–259, 1972.

- [8] W. Ang, *A Beginner's Course in Boundary Element Methods*. Universal Publishers, 2007.
- [9] J. Katsikadelis, *Boundary Elements: Theory and Applications*. Elsevier Science, 2002.
- [10] J. Katsikadelis, *The Boundary Element Method for Engineers and Scientists: Theory and Applications*. Elsevier Science, 2016.
- [11] S. Kapur and V. Rokhlin, "High-order corrected trapezoidal quadrature rules for singular functions," *SIAM Journal on Numerical Analysis*, vol. 34, pp. 1331 – 1356, 8 1997.
- [12] R. Duan and V. Rokhlin, "High-order quadratures for the solution of scattering problems in two dimensions," *Journal of Computational Physics*, vol. 228, no. 6, pp. 2152 – 2174, 2009.
- [13] J. Bremer, "On the nyström discretization of integral equations on planar curves with corners," *Applied and Computational Harmonic Analysis*, vol. 32, no. 1, pp. 45 – 64, 2012.
- [14] J. Bremer, V. Rokhlin, and I. Sammis, "Universal quadratures for boundary integral equations on two-dimensional domains with corners," *Journal of Computational Physics*, vol. 229, no. 22, pp. 8259 – 8280, 2010.
- [15] J. Helsing, "Corner singularities for elliptic problems: special basis functions versus "brute force"," *Communications in Numerical Methods in Engineering*, vol. 16, no. 1, pp. 37–46, 2000.

- [16] A. Gillman, S. Hao, and P.-G. Martinsson, “A simplified technique for the efficient and highly accurate discretization of boundary integral equations in 2d on domains with corners,” *Journal of Computational Physics*, vol. 256, pp. 214 – 219, 2014.
- [17] K. Serkh and V. Rokhlin, “On the solution of elliptic partial differential equations on regions with corners,” *Journal of Computational Physics*, vol. 305, pp. 150 – 171, 2016.
- [18] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *Journal of Computational Physics*, vol. 73, no. 2, pp. 325 – 348, 1987.
- [19] W. Hackbusch, “A sparse matrix arithmetic based on H-matrices. I. Introduction to \mathcal{H} -matrices,” *Computing*, vol. 62, no. 2, pp. 89–108, 1999.
- [20] W. Hackbusch and B. N. Khoromskij, “A sparse \mathcal{H} -matrix arithmetic : general complexity estimates,” *Journal of computational and applied mathematics*, vol. 125, no. 1-2, pp. 479–501, 2000.
- [21] W. Hackbusch and B. N. Khoromskij, “A sparse \mathcal{H} -matrix arithmetic. part ii. application to multi-dimensional problems,” *Computing*, vol. 64, no. 1, pp. 21–47, 2000.
- [22] S. Börm and W. Hackbusch, “A short overview of \mathcal{H}^2 -matrices,” *Proceedings in applied mathematics and mechanics*, vol. 2, no. 1, pp. 33–36, 2003.
- [23] P.-G. Martinsson and V. Rokhlin, “A fast direct solver for boundary integral equations in two dimensions,” *Journal of Computational Physics*, vol. 205, no. 1, pp. 1 – 23, 2005.

- [24] P.-G. Martinsson, “A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix,” *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1251–1274, 2011.
- [25] X. Liu, J. Xia, and M. V. de Hoop, “Parallel randomized and matrix-free direct solvers for large structured dense linear systems,” *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. S508–S538, 2016.
- [26] Z. Sheng, P. Dewilde, and S. Chandrasekaran, *Algorithms to Solve Hierarchically Semi-separable Systems*, pp. 255–294. Basel: Birkhäuser Basel, 2007.
- [27] J. Xia, S. Chandrasekaran, M. Gu, and X. Li, “Fast algorithms for hierarchically semiseparable matrices,” *Numerical Linear Algebra with Applications*, vol. 17, no. 6, pp. 953–976, 2010.
- [28] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, “Superfast multifrontal method for large structured linear systems of equations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1382–1411, 2009.
- [29] S. Chandrasekaran, M. Gu, and W. Lyons, “A fast adaptive solver for hierarchically semiseparable representations,” *Calcolo*, vol. 42, no. 3-4, pp. 171–185, 2005.
- [30] A. Gillman, P. M. Young, and P.-G. Martinsson, “A direct solver with $\mathcal{O}(n)$ complexity for integral equations on one-dimensional domains,” *Frontiers of Mathematics in China*, vol. 7, no. 2, pp. 217–247, 2012.
- [31] S. Ambikasaran and E. Darve, “An $o(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices,” *Journal of Scientific Computing*, vol. 57, no. 3, pp. 477–501, 2013.

- [32] B. Carpentieri, “Fast preconditioned krylov methods for boundary integral equations in electromagnetic scattering,” 2012.
- [33] K. Chen and P. J. Harris, “Efficient preconditioners for iterative solution of the boundary element equations for the three-dimensional helmholtz equation,” *Applied Numerical Mathematics*, vol. 36, no. 4, pp. 475 – 489, 2001.
- [34] B. Carpentieri, “Preconditioning for large-scale boundary integral equations in electromagnetics [open problems in cem],” *IEEE Antennas and Propagation Magazine*, vol. 56, pp. 338–345, Dec 2014.
- [35] S. C. Hawkins and K. Chen, “New wavelet preconditioner for solving boundary integral equations over nonsmooth boundaries,” *International Journal of Computer Mathematics*, vol. 81, no. 3, pp. 353–360, 2004.
- [36] X. Antoine, A. Bendali, and M. Darbas, “Analytic preconditioners for the boundary integral solution of the scattering of acoustic waves by open surfaces,” *Journal of Computational Acoustics*, vol. 13, no. 03, pp. 477–498, 2005.
- [37] S. Ambikasaran and E. Darve, “An $\mathcal{O}(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices,” *Journal of Scientific Computing*, vol. 57, pp. 477–501, December 2013.
- [38] A. Gillman and A. Barnett, “A fast direct solver for quasi-periodic scattering problems,” *Journal of Computational Physics*, vol. 248, pp. 309 – 322, 2013.
- [39] A. Barnett, B. Wu, and S. Veerapaneni, “Spectrally accurate quadratures for evaluation of layer potentials close to the boundary for the 2d stokes and laplace equations,” *SIAM Journal on Scientific Computing*, vol. 37, 10 2014.

- [40] J. Helsing and R. Ojala, “On the evaluation of layer potentials close to their sources,” *Journal of Computational Physics*, vol. 227, no. 5, pp. 2899 – 2921, 2008.
- [41] A. Klöckner, A. Barnett, L. Greengard, and M. O’Neil, “Quadrature by expansion: A new method for the evaluation of layer potentials,” *Journal of Computational Physics*, vol. 252, pp. 332 – 349, 2013.
- [42] H. A. Atwater and A. Polman, “Plasmonics for improved photovoltaic devices,” *Nature Materials*, vol. 9, pp. 205– 213, 2010.
- [43] M. D. Kelzenberg, S. W. Boettcher, J. A. Petykiewicz, D. B. Turner-Evans, M. C. Putnam, E. L. Warren, J. M. Spurgeon, R. M. Briggs, N. S. Lewis, and H. A. Atwater, “Enhanced absorption and carrier collection in si wire arrays for photovoltaic applications,” *Nature Materials*, vol. 9, pp. 239–244, 2010.
- [44] N. Sergeant, M. Agrawal, and P. Peumans, “High performance solar-selective absorbers using coated sub-wavelength gratings,” *Optics Express*, vol. 18, no. 6, pp. 5525–5540, 2010.
- [45] M. D. Perry, R. D. Boyd, J. A. Britten, D. Decker, B. W. Shore, C. Shannon, and E. Shults, “High-efficiency multilayer dielectric diffraction gratings,” *Optics Letters*, vol. 20, pp. 940–942, 1995.
- [46] C. Barty, M. Key, J. Britten, R. Beach, G. Beer, C. Brown, S. Bryan, J. Caird, T. Carlson, J. Crane, J. Dawson, A. Erlandson, D. Fittinghoff, M. Hermann, C. Hoaglan, A. Iyer, L. J. II, I. Jovanovic, A. Komashko, O. Landen, Z. Liao, W. Molander, S. Mitchell, E. Moses, N. Nielsen, H.-H. Nguyen, J. Nissen, S. Payne, D. Pennington, L. Risinger, M. Rushford, K. Skulina, M. Spaeth,

- B. Stuart, G. Tietbohl, and B. Wattellier, “An overview of llnl high-energy short-pulse technology for advanced radiography of laser fusion experiments,” *Nuclear Fusion*, vol. 44, no. 12, p. S266, 2004.
- [47] G. A. Kalinchenko and A. M. Lerer, “Wideband all-dielectric diffraction grating on chirped mirror,” *Journal of Lightwave Technology*, vol. 28, pp. 2743–2749, 2010.
- [48] M. Cho and A. Barnett, “Robust fast direct integral equation solver for quasi-periodic scattering problems with a large number of layers,” *Optics Express*, vol. 23, no. 2, pp. 1775–1799, 2015.
- [49] B. Wu, H. Zhu, A. Barnett, and S. Veerapaneni, “Solution of stokes flow in complex nonsmooth 2d geometries via a linear-scaling high-order adaptive integral equation scheme,” *Journal of Computational Physics*, vol. 410, p. 109361, 2020.
- [50] Y. Zhang and A. Gillman, “A fast direct solver for boundary value problems on locally perturbed geometries,” *Journal of Computational Physics*, vol. 356, pp. 356 – 371, 2018.
- [51] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, Fourth 2013.
- [52] T. F. Chan, “Rank revealing qr factorizations,” *Linear Algebra and its Applications*, vol. 88, pp. 67 – 82, 1987.
- [53] D. C. Sorensen and M. Embree, “A deim induced cur factorization,” *SIAM Journal on Scientific Computing*, vol. 38, no. 3, pp. A1454–A1482, 2016.

- [54] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, “On the compression of low rank matrices,” *SIAM Journal on Scientific Computing*, vol. 26, no. 4, pp. 1389–1404, 2005.
- [55] P.-G. Martinsson, V. Rokhlin, and M. Tygert, “A randomized algorithm for the decomposition of matrices,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47 – 68, 2011.
- [56] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, “Fast direct solvers for integral equations in complex three-dimensional domains,” *Acta Numer.*, vol. 18, pp. 243–275, 2009.
- [57] A.-S. Bonnet-Bendhia and F. Starling, “Guided waves by electromagnetic gratings and non-uniqueness examples for the diffraction problem,” *Mathematical Methods in the Applied Sciences*, vol. 17, no. 5, pp. 305–338, 1994.
- [58] T. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.
- [59] C. Geuzaine and J.-F. Remacle, “Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities,” *International Journal for Numerical Methods in Engineering*, vol. 79, pp. 1309 – 1331, 09 2009.
- [60] D. Komatitsch and J. Tromp, “A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation,” *Geophysical Journal International*, vol. 154, pp. 146 – 153, 07 2003.
- [61] I. M. Babuska and S. A. Sauter, “Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?,” *SIAM Journal of Numerical Analysis*, vol. 34, no. 6, pp. 2392–2423, 1997.

- [62] M. G. Moharam and T. G. Gaylord, "Rigorous coupled-wave analysis of planar-grating diffraction," *Journal of the Optical Society of America*, vol. 71, pp. 811–818, 1981.
- [63] L. Li, "Use of fourier series in the analysis of discontinuous periodic structures," *Journal of the Optical Society of America A*, vol. 13, pp. 1870–1876, 1996.
- [64] L. Li, "Formulation and comparison of two recursive matrix algorithms for modeling layered diffraction gratings," *Journal of the Optical Society of America A*, vol. 13, pp. 1024–1035, 1996.
- [65] K. Han and C.-H. Chang, "Numerical modeling of sub-wavelength anti-reflective structures for solar module applications," *Nanomaterials*, vol. 4, pp. 87–128, 01 2014.
- [66] H.-Y. Tsai, "Finite difference time domain analysis of three-dimensional sub-wavelength structured arrays," *Japanese Journal of Applied Physics*, vol. 47, p. 5007, 05 2008.
- [67] C.-J. Ting, C.-F. Chen, and C. Chou, "Antireflection subwavelength structures analyzed by using the finite difference time domain method," *Optik*, vol. 120, no. 16, pp. 814 – 817, 2009.
- [68] T.-H. Chou, K.-Y. Cheng, T.-L. Chang, C.-J. Ting, H.-C. Hsu, C.-J. Wu, J.-H. Tsai, and T.-Y. Huang, "Fabrication of antireflection structures on tco film for reflective liquid crystal display," *Microelectronic Engineering*, vol. 86, no. 4, pp. 628 – 631, 2009. MNE '08.
- [69] O. Bruno and M. Haslam, "Efficient high-order evaluation of scattering by periodic surfaces: Deep gratings, high frequencies, and glancing incidences,"

- Journal of the Optical Society of America. A, Optics, image science, and vision*, vol. 26, pp. 658–68, 04 2009.
- [70] K. Horoshenkov and S. Chandler-Wilde, “Efficient calculation of two-dimensional periodic and waveguide acoustic green’s functions,” *The Journal of the Acoustical Society of America*, vol. 111, pp. 1610–22, 05 2002.
- [71] T. Arens, “Scattering by biperiodic layered media: The integral equation approach,” *Habilitation thesis*, 2010.
- [72] T. Arens, S. Chandler-Wilde, and J. Desanto, “On integral equation and least squares methods for scattering by diffraction gratings,” *Communications in Computational Physics*, vol. 1, pp. 1010–1042, 12 2006.
- [73] M. Nicholas, “A higher order numerical method for 3-d doubly periodic electromagnetic scattering problems,” *Communications in Mathematical Sciences*, vol. 6, 09 2008.
- [74] A. Barnett and L. Greengard, “A new integral representation for quasi-periodic fields and its application to two-dimensional band structure calculations,” *Journal of Computational Physics*, vol. 229, pp. 6898–6914, 01 2010.
- [75] L. Greengard, K. Ho, and J.-Y. Lee, “A fast direct solver for scattering from periodic structures with multiple material interfaces in two dimensions,” *Journal of Computational Physics*, vol. 258, pp. 738–751., 2014.
- [76] O. P. Bruno and A. G. Fernandez-Lado, “Rapidly convergent quasi-periodic green functions for scattering by arrays of cylinders—including wood anomalies,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2199, 2017.

- [77] M. H. Cho, “Spectrally-accurate numerical method for acoustic scattering from doubly-periodic 3d multilayered media,” *Journal of Computational Physics*, vol. 393, pp. 46 – 58, 2019.
- [78] M. Abramowitz, *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*,. New York, NY, USA: Dover Publications, Inc., 1974.
- [79] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory*, vol. 93 of *Applied Mathematical Sciences*. Berlin: Springer-Verlag, second ed., 1998.
- [80] A. Barnett and T. Betcke, “Stability and convergence of the method of fundamental solutions for helmholtz problems on analytic domains,” *Journal of Computational Physics*, vol. 227, no. 14, pp. 7003 – 7026, 2008.
- [81] R. H. Torres and G. V. Welland, “The helmholtz equation and transmission problems with lipschitz interfaces,” *Indiana University Mathematics Journal*, vol. 42, no. 4, pp. 1457–1485, 1993.
- [82] V. Rokhlin, “Solution of acoustic scattering problems by means of second kind integral equations,” *Wave Motion*, vol. 5, pp. 257–272, 1983.
- [83] G. H. Golub and C. F. Van Loan, *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences, Baltimore, MD: Johns Hopkins University Press, third ed., 1996.
- [84] E. L. Yip, “A note on the stability of solving a rank-p modification of a linear system by the sherman–morrison–woodbury formula,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 2, pp. 507–513, 1986.

- [85] G. Marple, A. Barnett, A. Gillman, and S. Veerapaneni, “A fast algorithm for simulating multiphase flows through periodic geometries of arbitrary shape,” *SIAM Journal of Scientific Computing*, vol. 38, no. 5, pp. B740–B772, 2016.
- [86] M. Gu and S. C. Eisenstat, “Efficient algorithms for computing a strong rank-revealing qr factorization,” *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, 1996.
- [87] H. Cheng, Z. Gimbutas, P. Martinsson, and V. Rokhlin, “On the compression of low rank matrices,” *SIAM J. Scientific Computing*, vol. 26, pp. 1389–1404, 01 2005.
- [88] S. Hao, A. H. Barnett, P. G. Martinsson, and P. Young, “High-order accurate nystrom discretization of integral equations with weakly singular kernels on smooth curves in the plane,” *Advances in Computational Mathematics*, vol. 40, pp. 245 – 272, 2014.
- [89] B. Alpert, “Hybrid gauss-trapezoidal quadrature rules,” *SIAM Journal on Scientific Computing*, vol. 20, no. 5, pp. 1551–1584, 1999.
- [90] J. Helsing and R. Ojala, “Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning,” *Journal of Computational Physics*, vol. 227, no. 20, pp. 8820 – 8840, 2008.
- [91] J. Aguilar and Y. Chen, “High-order corrected trapezoidal quadrature rules for functions with a logarithmic singularity in 2-d,” *Computers & Mathematics with Applications*, vol. 44, no. 8, pp. 1031 – 1039, 2002.
- [92] A. Klöckner, A. Barnett, L. Greengard, and M. O’Neil, “Quadrature by expansion: A new method for the evaluation of layer potentials,” *Journal of*

Computational Physics, vol. 252, pp. 332 – 349, 2013.

- [93] J. Bremer, “On the nyström discretization of integral equations on planar curves with corners,” *Applied and Computational Harmonic Analysis*, vol. 32, 01 2012.
- [94] R. Schoeman, K. Rana, N. Danes, M. Lehmann, J. Paola, A. Fogelson, K. Leiderman, and K. Neeves, “A microfluidic model of hemostasis sensitive to platelet function and coagulation,” *Cellular and Molecular Bioengineering*, vol. 10, pp. 1–13, 10 2016.
- [95] A. A. Kayani, K. Khoshmanesh, S. A. Ward, A. Mitchell, and K. Kalantar-zadeh, “Optofluidics incorporating actively controlled micro- and nanoparticles,” *Biomicrofluidics*, vol. 6, no. 3, p. 031501, 2012.
- [96] T. Salafi and K. Kwek, “Advancements in microfluidics for nanoparticle separation,” *Lab Chip*, vol. 17, 11 2016.
- [97] H. Wioland, E. Lushi, and R. E. Goldstein, “Directed collective motion of bacteria under channel confinement,” *New Journal of Physics*, vol. 18, p. 075002, jul 2016.
- [98] J. Elgeti, R. G. Winkler, and G. Gompper, “Physics of microswimmers—single particle motion and collective behavior: a review,” *Reports on Progress in Physics*, vol. 78, p. 056601, apr 2015.
- [99] G. K. Youngren and A. Acrivos, “Stokes flow past a particle of arbitrary shape: a numerical method of solution,” *Journal of Fluid Mechanics*, vol. 69, no. 2, p. 377–403, 1975.

- [100] G. K. Youngren and A. Acrivos, “On the shape of a gas bubble in a viscous extensional flow,” *Journal of Fluid Mechanics*, vol. 76, no. 3, p. 433–442, 1976.
- [101] C. Sorgentone and A.-K. Tornberg, “A highly accurate boundary integral equation method for surfactant-laden drops in 3d,” *Journal of Computational Physics*, vol. 360, pp. 167 – 191, 2018.
- [102] A. Rahimian, S. Veerapaneni, D. Zorin, and G. Biros, “Boundary integral method for the flow of vesicles with viscosity contrast in three dimensions,” *Journal of Computational Physics*, vol. 298, pp. 766–786, 10 2015.
- [103] E. Corona, L. Greengard, M. Rachh, and S. Veerapaneni, “An integral equation formulation for rigid bodies in stokes flow in three dimensions,” *Journal of Computational Physics*, vol. 332, pp. 504 – 519, 2017.
- [104] M. Rachh and L. Greengard, “Integral equation methods for elastance and mobility problems in two dimensions,” *SIAM Journal on Numerical Analysis*, vol. 54, 07 2015.
- [105] A.-K. Tornberg and L. Greengard, “A fast multipole method for the three-dimensional stokes equations,” *Journal of Computational Physics*, vol. 227, no. 3, pp. 1613 – 1619, 2008.
- [106] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros, “Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures,” in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, Nov 2010.

- [107] A. Kumar and M. D. Graham, “Accelerated boundary integral method for multiphase flow in non-periodic geometries,” *Journal of Computational Physics*, vol. 231, no. 20, pp. 6682 – 6713, 2012.
- [108] S. K. Veerapaneni, A. Rahimian, G. Biros, and D. Zorin, “A fast algorithm for simulating vesicle flows in three dimensions,” *Journal of Computational Physics*, vol. 230, no. 14, pp. 5610 – 5634, 2011.
- [109] Y. Fu and R. G.J., “Fast solution method for threedimensional stokesian many-particle problems,” *Communications in Numerical Methods in Engineering*, vol. 16, pp. 145–149, 2000.
- [110] E. Lushi, H. Wioland, and R. E. Goldstein, “Fluid flows created by swimming bacteria drive self-organization in confined suspensions,” *Proceedings of the National Academy of Sciences*, vol. 111, no. 27, pp. 9733–9738, 2014.
- [111] R. Voituriez, J. F. Joanny, and J. Prost, “Spontaneous flow transition in active polar gels,” *Europhysics Letters*, vol. 70, pp. 404–410, may 2005.
- [112] F. G. Woodhouse and R. E. Goldstein, “Spontaneous circulation of confined active suspensions,” *Physical Review Letters*, vol. 109, p. 168105, Oct 2012.
- [113] S. E. Hulme, S. S. Shevkoplyas, J. Apfeld, W. Fontana, and G. M. Whitesides, “A microfabricated array of clamps for immobilizing and imaging *c. elegans*,” *Lab on a Chip*, vol. 7, p. 1515–1523, 2007.
- [114] S. Khaderi, C. Craus, J. Hussong, N. Schorr, J. Belardi, J. Westerweel, O. Prucker, J. Ruehe, J. Toonder, den, and P. Onck, “Magnetically-actuated artificial cilia for microfluidic propulsion,” *Lab on a Chip*, vol. 11, no. 12, pp. 2002–2010, 2011.

- [115] S. K. Veerapaneni, D. Gueyffier, D. Zorin, and G. Biros, “A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2d,” *Journal of Computational Physics*, vol. 228, no. 7, pp. 2334 – 2353, 2009.
- [116] A. Barnett, B. Wu, and S. Veerapaneni, “Spectrally accurate quadratures for evaluation of layer potentials close to the boundary for the 2d stokes and laplace equations,” *SIAM Journal on Scientific Computing*, vol. 37, 10 2014.
- [117] R. Ojala and A.-K. Tornberg, “An accurate integral equation method for simulating multi-phase stokes flow,” *Journal of Computational Physics*, vol. 298, pp. 145 – 160, 2015.
- [118] J.-Y. Lee and L. Greengard, “A fast adaptive numerical method for stiff two-point boundary value problems,” *SIAM Journal on Scientific Computing*, vol. 18, no. 2, pp. 403–429, 1997.
- [119] Y. Zhang and A. Gillman, “A fast direct solver for two dimensional quasi-periodic multilayered media scattering problems,” *BIT Numerical Mathematics*, pp. 1–31, 2019.
- [120] Y. Zhang and A. Gillman, “An alternative extended linear system for boundary value problems on locally perturbed geometries,” 2020.
- [121] P. Kolm and V. Rokhlin, “Numerical quadratures for singular and hypersingular integrals,” *Comput. Math. Appl.*, vol. 41, pp. 327–352, 2001.

Appendix A

Appendix

A.1 Efficient construction of \mathbf{S}_2 .

This section presents an efficient technique for constructing the tridiagonal matrix $\mathbf{S}_2 = \mathbf{I} + \mathbf{R}\mathbf{A}_0^{-1}\mathbf{L}$ in the fast direct solver for quasi-periodic scattering in layered media presented in Chapter 3.

For simplicity of presentation, let the blocks of \mathbf{S}_2 be denoted as follows

\mathbf{X}_i for $1 \leq i \leq I$ denotes the diagonal blocks,

\mathbf{Y}_i for $2 \leq i \leq I$ denotes the lower diagonal blocks, and

\mathbf{Z}_i for $1 \leq i \leq I - 1$ denotes the upper diagonal blocks.

The diagonal blocks are given by

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{I} + \mathbf{R}_{11}^{pm} \mathbf{A}_{0,11}^{-1} \mathbf{L}_{11}^{pm} & \mathbf{R}_{11}^{pm} \mathbf{A}_{0,11}^{-1} \mathbf{L}_{12} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

$$\mathbf{X}_I = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R}_{II}^{pm} \mathbf{A}_{0,II}^{-1} \mathbf{L}_{I,I-1} & \mathbf{I} + \mathbf{R}_{II}^{pm} \mathbf{A}_{0,II}^{-1} \mathbf{L}_{II}^{pm} \end{bmatrix},$$

and, for $2 \leq i \leq (I - 1)$,

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{ii}^{pm} \mathbf{A}_{0,ii}^{-1} \mathbf{L}_{i,i-1} & \mathbf{I} + \mathbf{R}_{ii}^{pm} \mathbf{A}_{0,ii}^{-1} \mathbf{L}_{ii}^{pm} & \mathbf{R}_{ii}^{pm} \mathbf{A}_{0,ii}^{-1} \mathbf{L}_{i,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

The lower diagonal blocks are given by

$$\mathbf{Y}_2 = \begin{bmatrix} \mathbf{R}_{21} \mathbf{A}_{0,11}^{-1} \mathbf{L}_{11}^{pm} & \mathbf{R}_{21} \mathbf{A}_{0,11}^{-1} \mathbf{L}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

and, for $3 \leq i \leq I$,

$$\mathbf{Y}_i = \begin{bmatrix} \mathbf{R}_{i,i-1} \mathbf{A}_{0,(i-1)(i-1)}^{-1} \mathbf{L}_{i-1,i-2} & \mathbf{R}_{i,i-1} \mathbf{A}_{0,(i-1)(i-1)}^{-1} \mathbf{L}_{(i-1)(i-1)}^{pm} & \mathbf{R}_{i,i-1} \mathbf{A}_{0,(i-1)(i-1)}^{-1} \mathbf{L}_{i-1,i} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Finally the upper diagonal blocks are defined by

$$\mathbf{Z}_i = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{i,i+1} \mathbf{A}_{0,(i+1)(i+1)}^{-1} \mathbf{L}_{i+1,i} & \mathbf{R}_{i,i+1} \mathbf{A}_{0,(i+1)(i+1)}^{-1} \mathbf{L}_{(i+1)(i+1)}^{pm} & \mathbf{R}_{i,i+1} \mathbf{A}_{0,(i+1)(i+1)}^{-1} \mathbf{L}_{i+1,i+2} \end{bmatrix}$$

for $1 \leq i \leq (I - 2)$, and

$$\mathbf{Z}_{I-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{I-1,I} \mathbf{A}_{0,II}^{-1} \mathbf{L}_{I,I-1} & \mathbf{R}_{I-1,I} \mathbf{A}_{0,II}^{-1} \mathbf{L}_{II}^{pm} \end{bmatrix}.$$

The matrix \mathbf{S}_2 can be inverted via a block variant of the Thomas algorithm.

Let the sum of the ranks of the low-rank approximations be defined as $N_i^{block} =$

$k_{pm,ii} + k_{i,i-1} + k_{i,i+1}$ for $2 \leq i \leq I-1$, $N_1^{block} = k_{pm,11} + k_{1,2}$ and $N_I^{block} = k_{pm,II} + k_{I,I-1}$. The diagonal block \mathbf{X}_i is of size $N_i^{block} \times N_i^{block}$. The upper diagonal block \mathbf{Z}_i has size $N_i^{block} \times N_{i+1}^{block}$. The lower diagonal block \mathbf{Y}_i has size $N_i^{block} \times N_{i-1}^{block}$.

For the tested geometries and wave numbers, N_i^{block} is only several hundreds and the diagonal blocks can be inverted rapidly via dense linear algebra. If all of the blocks are of similar size $N_i^{block} \approx N^{block}$, then the cost of inverting \mathbf{S}_2 via the block Thomas algorithm is $\mathcal{O}([N^{block}]^3 I)$, which is linear with respect to the number of interfaces.

A.2 Definitions for the Stokes boundary integral operators

This section defines the different Stokes potentials, which is used in the BIE formulation for confined Stokes flow in section 4.1 of chapter 4. Let τ be the boundary density on the boundary Γ , and, for simplicity, let $r = x - y$. \mathbf{I} is a 2×2 identity matrix.

- The single-layer velocity potential is

$$(\mathcal{S}\tau)(x) = \frac{1}{4\pi\mu} \int_{\Gamma} \left[\mathbf{I} \log \left(\frac{1}{\|r\|} \right) + \frac{r \otimes r}{\|r\|^2} \right] \tau(y) dl(y),$$

where \mathbf{I} is a 2×2 identity matrix.

- The double-layer velocity potential is

$$(\mathcal{D}\tau)(x) = \frac{1}{\pi} \int_{\Gamma} \left[\frac{(r \cdot \nu_y)(r \otimes r)}{\|r\|^4} \right] \tau(y) dl(y).$$

- The single-layer pressure potential is

$$(\mathcal{S}^P \tau)(x) = \frac{1}{2\pi} \int_{\Gamma} \frac{r \cdot \tau(y)}{\|r\|^2} dl(y).$$

- The double-layer pressure potential is

$$(\mathcal{D}^P \tau)(x) = \frac{\mu}{\pi} \int_{\Gamma} \left(-\frac{\nu_y \cdot \tau(y)}{\|r\|^2} + \frac{2(r \cdot \tau(y))(r \cdot \nu_y)}{\|r\|^4} \right) dl(y).$$

A.3 Numerical tests for Laplace and Helmholtz BVPs on locally perturbed geometries

This section reports the numerical test results from applying the fast direct solver in section 4.2 to solve a collection of problems. The integral equations are discretized via the Nyström method with a 16th order composite Gaussian quadrature. For all problems, the original geometry is discretized with enough points in order for the boundary value problem to be solved to 10 digits of accuracy. The HBS direct solver was used in all tests. The tolerance for HBS compression and low-rank approximation is set to $\epsilon = 10^{-10}$.

To illustrate the efficiency of the proposed technique, we compare the performance of the new solution technique with the fast solver developed for the original extended system in section 2.4 and building an HBS solver from scratch for the new geometry. We report the following:

- N_o : the number of discretization points on the original geometry;
- N_c : the number of discretization points cut from the original geometry;
- N_p : the number of discretization points added;

- $T_{\text{new},p}$: the time in seconds for the precomputation of the proposed solver;
- $T_{\text{orig},p}$: the time in seconds for the precomputation of the original fast solver;
- $T_{\text{hbs},p}$: the time in seconds for the precomputation of HBS from scratch for the new geometry;
- $r_p = \frac{T_{\text{hbs},p}}{T_{\text{new},p}}$;
- $T_{\text{new},s}$: the time in seconds for applying the proposed solver to one right-hand-side;
- $T_{\text{orig},s}$: the time in seconds for applying the original solver to one right-hand-side;
- $T_{\text{hbs},s}$: the time in seconds for applying the HBS inverse to one right-hand-side;
- $r_s = \frac{T_{\text{hbs},s}}{T_{\text{new},s}}$.

For Laplace problems, we consider a problem with known solution: the exact solution is defined as the potential due to N_{src} charges with location and charge value $\{(s_j, q_j)\}_{j=1}^{N_{\text{src}}}$ placed at the exterior of domain Ω ,

$$u_{\text{ext}}(x) = \sum_{j=1}^{N_{\text{src}}} q_j G(x, s_j).$$

To check the solution's accuracy, the approximate solution u_{app} evaluated at a collection of interior target points $\{t_i\}_{i=1}^{N_{\text{trg}}}$ is compared with exact solution u_{ext} , and the average relative error over the target locations

$$E = \frac{1}{N_{\text{trg}}} \sum_{i=1}^{N_{\text{trg}}} \frac{|u_{\text{app}}(t_i) - u_{\text{ext}}(t_i)|}{|u_{\text{ext}}(t_i)|} \quad (\text{A.3.1})$$

is reported as a measure of solution accuracy.

The tested geometries are simple enough that all tested discretization fully resolves the problem, and the error in the solution approximation E reaches $\epsilon = 10^{-10}$ for all test cases. .

A.3.1 A local change in the geometry

Consider the interior Laplace-Dirichlet BVP on the geometry illustrated in Figure A.3.1. The corners are smoothed via the scheme in [2]. A detailed description of this geometry is given in [50].

In the first experiment, the number of points cut remains fixed, $N_c = 16$, while the number of discretization points on Γ_k grows. In Figure A.3.1, this corresponds to the nose height d decreasing as N_k grows. The results are reported in Table A.2. All three solution techniques are linear with respect to N_o and the precomputation time for the new solution technique is about the same as the original extended system solver. It is roughly 3.5 times faster than building a new direct solver from scratch for the new geometry. The cost of applying the proposed solver is almost as fast applying the HBS approximate inverse.

In the next example, N_c grows by the same factor as N_k . The nose height d in Figure A.3.1 remains fixed. Table A.3 reports on the performance of all three solvers for this geometry. The proposed solution technique is the fastest for the precomputation step. It is much faster than the solver based on the original extended system formulation, especially for the case where N_c is large. A factor of roughly 2.9 speed up in the precomputation is observed. Again applying the proposed solver is slightly slower than applying the HBS approximate inverse.

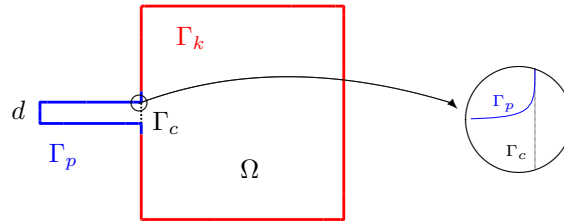


Figure A.3.1 : The square with nose geometry. A nose of height d is smoothly attached to the a square.

A.3.2 A Laplace problem with a locally refined discretization

Next the proposed solution technique is applied to the interior Laplace-Dirichlet BVP where the local perturbation is a refinement in a portion of the geometry. Figure A.3.2(a) illustrates the geometry under consideration. It is given by the following parameterization:

$$\mathbf{x}(t) = \begin{pmatrix} r(t) \cos(t) \\ r(t) \sin(t) \end{pmatrix}, \text{ with } r(t) = 1 + 0.3 \sin(30t) \text{ for } t \in [0, 2\pi].$$

The portion of the boundary being refined is highlighted in red. Figure A.3.2(b) is a zoomed in illustration of that region. Figure A.3.2(c) illustrates the local refinement. Three Gaussian panels ($N_c = 48$) are replaced with N_p discretization points ($N_p/16$ Gaussian panels). The number of discretization points on Γ_k remains fixed; $N_k = 6352$.

Table A.4 reports on the performance of all three solution techniques for this problem. The proposed solution technique is 13 to 21 times faster than building a new solver from scratch while applying the solver is less than a factor two slower than

applying the HBS approximate inverse.

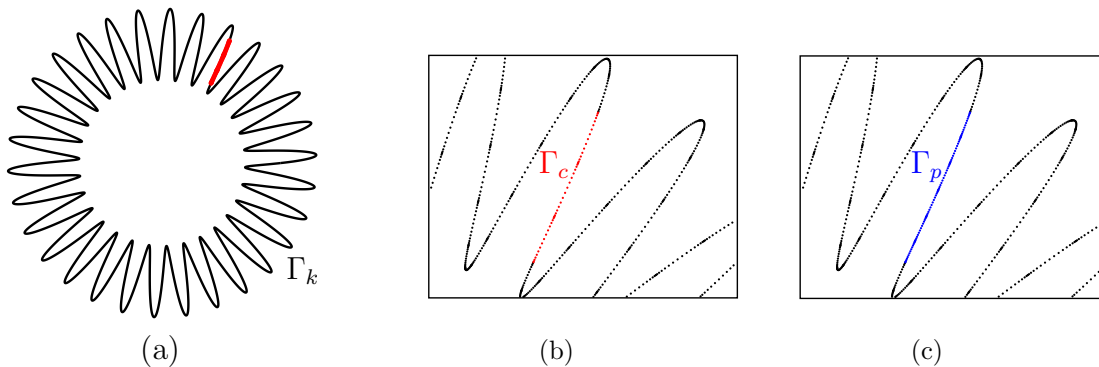


Figure A.3.2 : (a) The sunflower geometry with the portion of the boundary to be refined in red. (b) The three Gaussian panels in the boxed region from the original discretization. (c) Six Gaussian panels replacing the original three panels.

A.3.3 A Helmholtz problem with a locally refined discretization

Besides being faster than the solver for the original extended system, the proposed solver has the advantage that it can easily handle problems that are using specialized quadrature for weakly singular kernels. The issue that arises for the original extended system is that it would be cumbersome to evaluate the entries of the matrix \mathbf{A}_{op} corresponding to the interaction of Γ_c with Γ_p . This matrix does not arise in the new extended system.

To illustrate the efficiency of the solver for systems that involve specialized quadrature we consider the following exterior Dirichlet Helmholtz boundary value problem

$$\begin{aligned} -\Delta u(\mathbf{x}) + \omega^2 u &= 0 & \text{for } \mathbf{x} \in \Omega^c, \\ u(\mathbf{x}) &= g(\mathbf{x}) & \text{for } \mathbf{x} \in \Gamma \end{aligned} \tag{A.3.2}$$

with Sommerfeld radiation condition on the sunflower geometry illustrated in Figure A.3.2 where ω denotes the wave number. We chose to represent the solution with the following combined field

$$u(\mathbf{x}) = \int_{\Gamma} D_{\omega}(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) ds(\mathbf{y}) - i\omega \int_{\Gamma} S_{\omega}(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) ds(\mathbf{y}), \quad (\text{A.3.3})$$

where D_{ω} and S_{ω} denote the double and single layer Helmholtz kernel and $\sigma(\mathbf{x})$ is the unknown boundary charge distribution.

The integral equation that results from enforcing the Dirichlet boundary condition is

$$\frac{1}{2}\sigma(\mathbf{x}) + \int_{\Gamma} D_{\omega}(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) ds(\mathbf{y}) - i\omega \int_{\Gamma} S_{\omega}(\mathbf{x}, \mathbf{y})\sigma(\mathbf{y}) ds(\mathbf{y}) = g(\mathbf{x}). \quad (\text{A.3.4})$$

We discretize the operator via Nyström with a composite generalized Gaussian quadrature [121]. The wave number is set to $\omega = 20$ which corresponds to the geometry being approximately 8.3 wavelengths in size. Again, we consider the local refinement problem. Table A.1 reports on the performance of the proposed solution technique and building a fast direct solver from scratch. For this problem, the proposed solver is anywhere from 15 to 35 times faster than building the fast direct solver from scratch. This speed up is the result of the increased ranks associated with Helmholtz problems. Applying the proposed solver to a right-hand-side is roughly 1.5 times slower than applying the HBS solver.

| N_p | $\frac{N_p}{N_o}$ | $T_{\text{new},p}$ | $T_{\text{hbs},p}$ | r_p | $T_{\text{new},s}$ | $T_{\text{hbs},s}$ | r_s |
|-------|-------------------|--------------------|--------------------|-------|--------------------|--------------------|-------|
| 96 | 0.015 | 1.13e+00 | 3.97e+01 | 35.2 | 4.06e-02 | 2.86e-02 | 0.71 |
| 192 | 0.03 | 1.36e+00 | 4.08e+01 | 29.9 | 4.64e-02 | 2.93e-02 | 0.63 |
| 384 | 0.06 | 1.44e+00 | 4.08e+01 | 28.4 | 3.91e-02 | 2.54e-02 | 0.65 |
| 768 | 0.12 | 1.64e+00 | 4.17e+01 | 25.4 | 3.69e-02 | 2.70e-02 | 0.73 |
| 1536 | 0.24 | 2.64e+00 | 4.08e+01 | 15.4 | 4.39e-02 | 3.33e-02 | 0.76 |

Table A.1 : Times for applying the solution techniques to (A.3.2) on the geometry in Figure A.3.2 with local refinement.

| N_o | $T_{\text{orig},p}$ | $T_{\text{new},p}$ | $T_{\text{hbs},p}$ | r_p | $T_{\text{orig},s}$ | $T_{\text{new},s}$ | $T_{\text{hbs},s}$ | r_s |
|--------|---------------------|--------------------|--------------------|-------|---------------------|--------------------|--------------------|-------|
| 9232 | 3.69e-01 | 4.83e-01 | 1.57e+00 | 3.25 | 1.99e-02 | 1.12e-02 | 1.32e-02 | 1.18 |
| 18448 | 5.60e-01 | 6.50e-01 | 2.38e+00 | 3.66 | 2.76e-02 | 1.74e-02 | 1.46e-02 | 0.84 |
| 36880 | 1.11e+00 | 1.11e+00 | 3.79e+00 | 3.42 | 5.49e-02 | 4.00e-02 | 3.33e-02 | 0.83 |
| 73744 | 2.25e+00 | 1.84e+00 | 6.38e+00 | 3.47 | 9.79e-02 | 8.06e-02 | 7.04e-02 | 0.87 |
| 147472 | 3.87e+00 | 3.56e+00 | 1.18e+01 | 3.33 | 1.95e-01 | 1.71e-01 | 1.52e-01 | 0.89 |

Table A.2 : Times for applying the solution technique to an interior Laplace-Dirichlet BVP on the square with thinning nose geometry.

| N_o | N_c | $T_{\text{orig},p}$ | $T_{\text{new},p}$ | $T_{\text{hbs},p}$ | r_p | $T_{\text{orig},s}$ | $T_{\text{new},s}$ | $T_{\text{hbs},s}$ | r_s |
|--------|-------|---------------------|--------------------|--------------------|-------|---------------------|--------------------|--------------------|-------|
| 9344 | 128 | 5.01e-01 | 5.10e-01 | 1.28e+00 | 2.50 | 2.08e-02 | 1.02e-02 | 7.92e-03 | 0.77 |
| 18688 | 256 | 1.08e+00 | 9.25e-01 | 2.18e+00 | 2.36 | 3.44e-02 | 2.15e-02 | 1.59e-02 | 0.74 |
| 37376 | 512 | 2.67e+00 | 1.30e+00 | 3.49e+00 | 2.69 | 5.64e-02 | 3.97e-02 | 3.00e-02 | 0.76 |
| 74752 | 1024 | 7.76e+00 | 2.31e+00 | 6.63e+00 | 2.87 | 1.16e-01 | 8.67e-02 | 6.40e-02 | 0.74 |
| 149504 | 2048 | 2.48e+01 | 4.06e+00 | 1.19e+01 | 2.92 | 2.34e-01 | 1.71e-01 | 1.61e-01 | 0.94 |

Table A.3 : Times for applying the solution techniques to an interior Laplace-Dirichlet BVP on the square with fixed nose geometry.

| N_p | $\frac{N_p}{N_o}$ | $T_{\text{orig},p}$ | $T_{\text{new},p}$ | $T_{\text{hbs},p}$ | r_p | $T_{\text{orig},s}$ | $T_{\text{new},s}$ | $T_{\text{hbs},s}$ | r_s |
|-------|-------------------|---------------------|--------------------|--------------------|-------|---------------------|--------------------|--------------------|-------|
| 96 | 0.015 | 6.06e-01 | 5.03e-01 | 7.52e+00 | 14.9 | 1.10e-02 | 1.30e-02 | 1.32e-02 | 1.02 |
| 192 | 0.03 | 6.16e-01 | 3.62e-01 | 7.77e+00 | 21.4 | 1.17e-02 | 1.25e-02 | 9.30e-03 | 0.74 |
| 384 | 0.06 | 6.83e-01 | 3.90e-01 | 7.72e+00 | 19.8 | 1.36e-02 | 1.42e-02 | 9.13e-03 | 0.64 |
| 768 | 0.12 | 7.60e-01 | 4.11e-01 | 7.78e+00 | 18.9 | 2.01e-02 | 1.20e-02 | 9.06e-03 | 0.76 |
| 1536 | 0.24 | 1.01e+00 | 6.09e-01 | 8.03e+00 | 13.2 | 4.72e-02 | 1.66e-02 | 1.00e-02 | 0.60 |

Table A.4 : Times for applying the solution techniques to an interior Laplace-Dirichlet BVP on the sunflower geometry in Figure A.3.2 with local refinement.

A.4 Definition of matrix M_n

This section defines the matrix M_n , which maps function values at Gauss-Legendre quadrature points to the coefficients for Legendre expansion. This is used in the local expansion based panel-picking strategy in section 5.2.2 of chapter 5. For simplicity of illustration, the matrix is defined for the interval $[-1, 1]$. The definition can be extended to arbitrary interval $[a, b]$ in a straightforward manner. Consider $g : [-1, 1] \mapsto \mathbb{R}$. Let $P_j(x)$ for $j = 0, 1, \dots, (n - 1)$ denote the n th order Legendre polynomial defined on $(-1, 1)$ and $g(x) = \sum_{j=0}^{\infty} \alpha_j P_j(x)$ be the Legendre expansion for $g(x)$ on $(-1, 1)$. By linearity and orthogonality of Legendre polynomials,

$$\int_{-1}^1 g(x) P_l(x) dx = \sum_{j=0}^{\infty} \alpha_j \int_{-1}^1 P_l(x) P_j(x) dx = \alpha_l \frac{2}{2l+1}$$

and thus

$$\alpha_l = \frac{2l+1}{2} \int_{-1}^1 g(x) P_l(x) dx \tag{A.4.1}$$

for $l = 0, 1, \dots$.

Recall the recurrence relation for Legendre polynomials

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_l(x) = \frac{2l-1}{l} x P_{l-1}(x) - \frac{l-1}{l} P_{l-2}(x), \text{ for } l \geq 2.$$

Plugging the recurrence relation into (A.4.1) yields

$$\begin{aligned}
\alpha_0 &= \frac{1}{2} \int_{-1}^1 g(x) dx \\
\alpha_1 &= \frac{3}{2} \int_{-1}^1 g(x)x dx \\
\alpha_l &= \frac{2l+1}{2} \int_{-1}^1 g(x) \left(\frac{2l-1}{l} x P_{l-1}(x) - \frac{l-1}{l} P_{l-2}(x) \right) dx, \text{ for } l \geq 2.
\end{aligned} \tag{A.4.2}$$

Using the n th order Gauss-Legendre quadrature to approximate the integrals in (A.4.2) gives

$$\begin{aligned}
\alpha_0 &\approx \frac{1}{2} \sum_{j=1}^n g(t_j) w_j \\
\alpha_1 &\approx \frac{3}{2} \sum_{j=1}^n t_j g(t_j) w_j \\
\alpha_l &\approx \frac{2l+1}{2} \sum_{j=1}^n \left(\frac{2l-1}{l} t_j P_{l-1}(t_j) - \frac{l-1}{l} P_{l-2}(t_j) \right) g(t_j) w_j, \text{ for } l \geq 2.
\end{aligned}$$

Define the matrix M_n entry-wise as

$$\begin{aligned}
(M_n)_{1,j} &= \frac{1}{2} w_j, \text{ for } j = 1, \dots, n \\
(M_n)_{2,j} &= \frac{3}{2} t_j w_j, \text{ for } j = 1, \dots, n \\
(M_n)_{l,j} &= \frac{2l+1}{2} \left(\frac{2l-1}{l} t_j P_{l-1}(t_j) - \frac{l-1}{l} P_{l-2}(t_j) \right) w_j, \text{ for } l \geq 2 \text{ and } j = 1, \dots, n
\end{aligned}$$

and let $\mathbf{g} = [g(t_1), \dots, g(t_n)]^T$. Then the first n coefficients $\alpha = [\alpha_0, \dots, \alpha_{n-1}]$ can be numerically approximated as

$$\alpha \approx M_n \mathbf{g}.$$