A Spectral Finite Element Method on Non-Conforming Meshes: Domain Decomposition for High Frequency Scattering Problems

Ryan Galagusz

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

December 2018

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Contents

List of Figures

List of Tables

Abstract

Computational electromagnetics—the solution of Maxwell's equations using computers—is a key component of the modern design cycle for a wide variety of electrical engineering devices. These include, but are not limited to, antennas, microwave devices, photonic crystals, optical waveguides, and electric machines. This wide range of devices demonstrates the predictive power of the theory of electromagnetism and the need to accurately analyze Maxwell's equations in situations for which classical mathematical techniques are ineffective.

This thesis describes a high accuracy finite element method suitable for solving Poisson and Helmholtz problems, which arise from Maxwell's equations. High accuracy finite element methods are particularly useful for high frequency electromagnetic scattering problems. This is because experimental and theoretical results regarding dispersion errors for finite element methods applied to the Helmholtz problem indicate that an effective approach to control dispersion is to increase the polynomial degree of the finite element model as a function of element size and frequency. Increasing the polynomial degree where solutions are smooth leads to high accuracy. However, there are difficulties associated with the solution of the resulting linear systems when the polynomial degree increases. This tends to limit the extent to which high degree polynomial modeling is adopted in practice.

To circumvent these difficulties, this thesis develops, from first principles, a high accuracy one-dimensional finite element method that exploits Legendre polynomial expansions and the associated fast Legendre transform. The method is extended to higher dimensions, and implemented and tested in two dimensions, by developing a systematic approach to enforce inter-element continuity. This approach allows for both arbitrary refinement of local polynomial degree and non-conforming mesh refinement.

The method proposed in this thesis is capable of computing solutions to a user specified tolerance—potentially as stringent as machine precision—efficiently. All element-wise computations are performed with near linear computational complexity, which allows for the use of high polynomial degree to achieve high accuracy. The developed method is efficient because it consists of a domain decomposition method that fully exploit these fast element-wise computations. As long as the coupling between domains in the decomposition increases in such a way as to control dispersion errors, the method can be applied to compute high accuracy solutions while only solving systems that are much smaller than the total number of unknowns. The thesis demonstrates this behavior on several electromagnetic problems, including beam steering by lenses and photonic crystal waveguides, and radar cross section computation for dielectric, perfect electric conductor, and electromagnetic cloak scatterers.

Résumé

L'électromagnétisme numérique—la solution des équations de Maxwell par ordinateur—est un élément clé du cycle de conception moderne d'une grande variété de dispositifs communs en génie électrique. Ces dispositifs comprennent les antennes, les dispositifs à micro-ondes, les cristaux photoniques, les guides d'ondes optiques et les machines électriques. Cette large gamme de dispositifs démontre le pouvoir prédictif de la théorie de l'électromagnétisme et la nécessité d'analyser avec précision les équations de Maxwell quand les techniques mathématiques classiques ne sont pas efficaces.

Cette thèse décrit une méthode des éléments finis de haute précision pour résoudre numériquement les équations de Poisson et de Helmholtz qui proviennent des équations de Maxwell. La méthode des éléments finis de haute précision est particulièrement utile pour résoudre numériquement les problèmes de diffusion électromagnétique à haute fréquence. Plusieurs résultats expérimentaux et théoriques concernant les erreurs de dispersion de la méthode des éléments finis pour l'équation de Helmholtz indiquent qu'une approche efficace pour contrôler la dispersion consiste à augmenter le degré du modèle polynomial des éléments finis en fonction de la taille des éléments et de la fréquence du problème. L'augmentation du degré polynomial mène à haute précision là où la solution est lisse. Cependant, lorsque le degré polynomial augmente, le temps de calcul pour résoudre le système linéaire provenant de la méthode des éléments finis devient considérable. Cela limite la mesure dans laquelle la modélisation polynomiale à haut degré est adoptée en pratique.

Pour contourner ces difficultés, cette thèse développe, à partir de principes de base, une méthode des éléments finis unidimensionnel de haute précision qui exploite les polynômes de Legendre et la transformée de Legendre rapide. La méthode est étendue à dimensions supérieures, et réalisée et testée en deux dimensions, en développant une approche systématique pour assurer la continuité entre les éléments. Cette approche permet à la fois un raffinement arbitraire du degré polynomial local et un raffinement du maillage non conforme.

La méthode proposée dans cette thèse est capable de calculer des solutions à une tolérance spécifiée par l'utilisateur—possiblement aussi stricte que la précision double—efficacement. Les calculs pour chaque élément sont effectués avec une complexité de calcul quasi linéaire, ce qui permet d'utiliser un degré polynomial élevé pour obtenir une méthode à haute précision. La méthode développée est efficace car elle consiste d'une méthode de décomposition de domaine qui exploite les calculs rapides pour chaque élément. Tandis que le couplage entre les domaines dans la décomposition augmente de manière à contrôler les erreurs de dispersion, la méthode peut calculer des solutions de haute précision en résolvant que des systèmes

beaucoup plus petits que le nombre total d'inconnues. Cette thèse démontre ce comportement avec plusieurs problèmes électromagnétiques, comprenant l'analyse par simulation des ondes dirigées par lentille ou par guides d'ondes à cristal photonique, et le calcul de la section équivalente radar des diffuseurs diélectriques et métalliques, et des capes d'invisibilité.

Acknowledgements

I would like to thank my supervisor Professor Steve McFee for the advice that he has provided throughout my graduate studies at McGill University. I am grateful for his patience, understanding, and encouragement. His questions and comments during our conversations have, on many occasions, completely changed my perspective on a problem and opened my mind to ideas I may not have considered otherwise.

I would also like to extend my grattitude towards the members of my supervisory committee, Professors Dennis Giannacopoulos and Roni Khazaka. They, along with my supervisor, showed me the ropes of computational electrical engineering. I didn't know then, but I know now that teaching is research.

In addition, I would like to thank my peers in the Computational Electromagnetics Lab for stimulating discussions over the years. In particular, I would like to thank Adrian whose enthusiasm for all things engineering science and mathematics is infectious. His curiosity and mathematical rigor are exemplary, and I am thankful for our continued friendship.

Finally, I would like to express my appreciation to my family. Thank you for your unwavering love and support. When I was younger, my parents and grandparents stressed the importance of education. I don't think they, nor I, thought that that meant pursuing doctoral studies, but somehow, that's what I've done. Thank you for instilling in me the desire to learn.

This research was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG).

Chapter 1

Introduction

In 2013, the United Nations Educational, Scientific, and Cultural Organization (UNESCO) proclaimed 2015 the International Year of Light and Light-based Technologies (IYL 2015) [1]. Their resolution cites notable anniversaries of several key scientific theories in the history of light, including the 150th anniversary of Maxwell's theory of electromagnetism. This theory informs all aspects of the modern electrical engineer's work. While the study of classical electromagnetism is performed through the analysis of Maxwell's equations, set forth in [2], the modern form of the equations was described in 1885 by Oliver Heaviside [3], an electrical engineer. Heaviside used the reformulation to study a wide variety of topics, including transmission line theory and the skin effect and developed the operational calculus (a precursor to the Laplace transform) to facilitate this analysis. His contributions to electrical engineering continue to be taught in the undergraduate curricula of major universities through the instruction of electric circuits and electromagnetic fields and waves.

The finite element method applied to Maxwell's equations is—to some extent—a natural continuation of Heaviside's pioneering work on the solution of Maxwell's equations. Heaviside undertook a systematic study of the transmission line (a one-dimensional reduction of Maxwell's equations). In light of Heaviside's solution of the one-dimensional equations, it is natural to ask which solutions to the two- and three-dimensional cases can be computed. In advanced electromagnetics courses, the first tool taught is the classical method of solution using eigenfunction expansions [4, 5, 6, 7, 8, 9]. Classical eigenfunction expansions make use of special geometric properties of the problem which allow for the separation of partial differential equations (PDEs) into a set of related ordinary differential equations (ODEs) whose series solutions are known. In the last 50 years, the finite element method has been used to solve Maxwell's equations in situations where the one-dimensional simplifications, or classical eigenfunction expansions are not applicable [10, 11, 12, 13]. The method, put simply, breaks the region of interest over which Maxwell's equations are to be solved into a

number of disjoint subdomains (called elements). Over each element, the solution is approximated by a linear combination of simple functions (often polynomial, called basis functions) whose coefficients we seek. By ensuring that the local simple functions "agree" across element boundaries, and that the local linear combinations "satisfy" the PDE, we determine an approximate solution to Maxwell's equations[1].

The primary objective of this thesis is to develop a high accuracy finite element method which uses high polynomial degree basis functions on elements for the solution of Poisson and Helmholtz equations, which arise from Maxwell's equations. A high accuracy solver is particularly useful for high frequency problems because experimental and theoretical results regarding dispersion errors (sometimes called pollution errors) for finite element methods applied to the Helmholtz equation suggest that an effective approach to control pollution is to increase the polynomial degree (sometimes called order) as a function of element size and frequency [14, 15, 16, 17, 18, 19]. While significant efforts have been made to avoid polynomials in attempts to eliminate pollution errors, experimental evidence suggests that high-order finite element methods are competitive with these alternative approaches [20]. However, there are difficulties associated with the solution of the resulting linear systems when the polynomial degree increases [21, 22] which tend to limit the extent to which high-order methods are adopted in practice.

The method proposed in this thesis, which will be developed in one dimension, then subsequently extended to higher dimensions (using two dimensions to illustrate the approach), is capable of computing solutions to a user specified tolerance (as stringent as machine precision) efficiently. All element-wise computations can be performed with $\mathcal{O}(N)$ or $\mathcal{O}(N(\log_2 N)^2)$ complexity, where N is the number of unknowns per element. This allows for the use of high polynomial degree to achieve high accuracy. The proposed method is efficient because it consists of a domain decomposition algorithm which continues to exploit these efficient decoupled element-wise computations at the cost of introducing a sparse coupling matrix. In the domain decomposition method, the action of the inverse of the coupling matrix applied to a vector is required. Since the dimension of the coupling matrix is a frequency-dependent multiple of the number of interfaces between elements, the coupling problem can become costly, but several examples in the thesis demonstrate that its cost is small when the frequency of the problem is small and when the number of fine geometric features of the problem is small. The coupling problem is always much smaller than the total number of unknowns in the finite element problem. In constructing this high accuracy finite element method, the main contributions of this thesis are:

1. The development of a one-dimensional finite element method that exploits Legendre

[1]The sense in which "agree" and "satisfy" are to be understood will be made precise later.

polynomial expansions to represent spatially varying parameters and forcing functions which uses the fast Legendre transform to efficiently compute these expansions.

2. The development of a systematic approach to allow arbitrary element refinement and polynomial degree mismatch between elements in higher dimensions needed to extend the one-dimensional approach of Contribution 1.

3. The development of a domain decomposition method specially tailored for the types of discretizations arising from Contribution 2.

4. Explanations and demonstrations of how the domain decomposition method in Contribution 3 is suitable for both low and high frequency electromagnetic problems.

1.1 Historical Overview

The finite element method employed in the analysis of Maxwell's equations is not new and, as one might expect, significant progress has been made towards a fast and accurate solver. In fact, the history of the finite element method can be traced back some 300 years beginning with the work of several great mathematicians. For an excellent review of the history of the finite element method, see [23], which covers the first treatment of variational problems in 1696 up until the first modern electrical computing applications of the finite element method in 1956. Here, we highlight some key contributions in this history, as outlined in that review.

We begin with the brachistochrone problem of Johann Bernoulli in 1696, which asks, given two points A and B in the plane, to find the curve yielding the fastest travel time of an infinitesimal point starting from A and traveling on the curve under the effect of gravity to B. While many had provided solutions to the brachistochrone problem by 1744, Euler solved the problem and a large class of similar variational problems by formulating a more general minimization problem, and then showing that the minimizing solution solved a corresponding ODE. Interestingly, Euler's proof relied on breaking the interval into a set of small segments, and approximating the curve by linear polynomials on each segment. Some view this approach as a precursor to the finite element method, although many of the ingredients that comprise a finite element method today are missing. In fact, that proof has since been forgotten in favor of an elegant alternative presented by Lagrange in 1755 which relied on perturbing the solution by a small arbitrary variation. This is the modern approach used to derive from a variational equation the corresponding underlying ODE. It was not until 1857 that Riemann showed that the solution of Laplace's equation subject to Dirichlet boundary conditions was equivalent to the solution minimizing a given energy functional.

This is Dirichlet's principle, and was made rigorous between 1900 and 1904 by Hilbert. Ten years later, Courant (Hilbert's student) improved upon, and simplified Hilbert's treatment.

The idea that such variational formulations could be solved approximately by substituting a linear combination of basis functions for the solution, reducing the infinite-dimensional problem of minimizing the functional over a space of functions into a finite-dimensional minimization problem in the unknown coefficients of the expansion was first proposed in 1908 by Ritz. There, Ritz began with two PDEs—the biharmonic equation as well as Laplace's equation—and wrote down the equivalent problem in variational form using Dirichlet's principle. He then introduced a linear combination of basis functions that led to a finite-dimensional quadratic function in terms of the coefficients whose minimizer solved a linear system of equations. His choice of basis functions was made so that integrals could be evaluated by human computer, resulting in a diagonally dominant linear system that could be solved (again, by human computer) using an iterative method!

The adoption of these methods in Western Europe was slow, but there was immediate endorsement of these ideas in Russia, with Timoshenko (1913), Bubnov (1914), and Galerkin (1915) all making use of and illustrating the effectiveness of the method for structural engineering problems relating to rods and plates for shipbuilding. Bubnov noticed that it was not necessary to appeal to the variational formulation, then recover the linear system via minimization after the fact. He observed that one could obtain the same linear system directly by substituting the linear expansion for the solution in the PDE and then integrating against a given basis function to obtain each equation separately. Today, this method is known as Galerkin's method as he was the first to observe that such an approach could also be applied to equations that do not admit a variational formulation.

The basis functions employed by Ritz and Galerkin had been either eigenfunctions or orthogonal functions to ease the computation of integrals in their respective approaches (recall that neither had access to modern electrical computers). It was not until 1943 that Courant proposed using basis functions that exist only on local subsets of the total domain of interest. Courant suggested low order polynomials on disjoint triangular subdomains to solve the plane torsion problem in what is considered by most mathematicians to be the first paper on the finite element method. To most engineers, however, many details remained to be specified. In fact, it was not until 1956 that Jon Turner's structural dynamics group at Boeing described the first modern electrical computing application of the method.

1.2 Relation to Prior Work

This historical perspective leads us to review the finite element method as applied to electrical engineering problems. To my knowledge, the first such occasion dates to 1969 where the finite element method was employed by Silvester to analyze homogeneous waveguide problems [24]. It is interesting to note the lag between the first application of the finite element method to electromagnetics and its first application in structural mechanics. This adoption was likely slow for two reasons: one, discretization of Helmholtz's equation (time-harmonic problems) leads to indefinite matrices so that standard solvers for symmetric, positive definite systems which arise when dealing with Poisson's equation (static problems) such as the Cholesky factorization or the Conjugate Gradient method should no longer be used [25], and two, that the engineering community had not yet found a satisfactory way to impose tangential continuity of fields between elements rather than full continuity when solving the vector wave equation [12]. This thesis focuses on contributions to the first of these two difficulties, but in this section, past efforts towards solving both problems are highlighted.

Regarding tangential continuity, Nedelec's seminal 1980 work on mixed finite elements in $\mathbb{R}^3$ has since been recognized as fundamental [26]. Nedelec proposed on the tetrahedron two families of finite element basis functions which were polynomial, and that allow direct enforcement of tangential or normal continuity of vector fields, rather than standard continuity across element boundaries[2]. To do this, Nedelec used the formalism of Ciarlet [27], which requires the specification of three properties to define an element: one, the canonical region of the element (denoted K), two, the space of functions that can be represented on K (denoted P_K), and three, a set of linear functionals (referred to as degrees of freedom). By specifying that K be the tetrahedron, P_K be one of two special spaces of polynomials, and that the degrees of freedom be special moments along edges, faces, and volumes of K, Nedelec was able to show that the resulting polynomial basis functions could be used to easily enforce tangential or normal continuity between adjacent elements.

Unfortunately, for the electrical engineer, such a specification does not yield explicit representations of the basis functions which makes including them in a general finite element code somewhat difficult. That being said, the basis can be computed by first picking an arbitrary basis for P_K and applying the linear functionals to this expansion, requiring that the basis be interpolatory in the specified degrees of freedom. This yields a matrix whose inverse is the basis transformation matrix from the arbitrary basis to the Nedelec basis. Note that while the basis interpolates the degrees of freedom, they are degrees of freedom

[2] In the parlance of modern finite element analysis, these elements are conforming in the spaces H (curl) and H (div) respectively (contrary to the standard H^1 conforming elements).

associated with integrals along edges, faces, and volumes of K and do not interpolate a vector function in the sense of a classical nodal basis [12, 28]. It is for this reason that the name "edge element" has been associated with Nedelec's tangential conforming elements, which specify degrees of freedom of integrals on edges first, then faces, and finally volumes. One of the important properties of the Nedelec families is that they satisfy a commuting discrete de Rham diagram [29]. This property is important in proving the convergence of finite element solutions for Maxwell's equations.

While Nedelec's specification is valid for arbitrary orders of approximation, the resulting basis functions are not hierarchical, meaning that if one increases or decreases the degree of polynomial representation on a given element, one must do so for all elements. To this end, Demkowicz et al. proposed a set of degrees of freedom that preserve the discrete de Rham diagram, and that allow for arbitrary polynomial degrees on neighboring elements [30]. However, like Nedelec's paper, no explicit description of the basis functions is given. That being said, there is extensive literature on a variety of basis functions that have explicit representation, and that are hierarchical. To highlight but a few, we mention the works [12, 31, 32, 33, 34, 35]. The works by Webb and Jin are geared towards an electrical engineering audience while the remaining sources are aimed at applied mathematicians.

None of the aforementioned references to hierarchical basis functions for use in the finite element method applied to the vector wave equation address issues of optimal conditioning and sparsity of the resulting local element matrices. This is likely related to the fact that for arbitrary material properties varying spatially over an element, one obtains full local element matrices and so a discussion on these topics is often ignored. However, in the computational fluid dynamics community, Sherwin and Karniadakis describe the existence of basis functions for scalar elements that lead to well-conditioned, sparse matrices when material parameters are constants over the element [36, 37]. Their methods rely on the theory of orthogonal polynomials on the triangle and tetrahedron which can be found through the Duffy transformation that takes the square or cube to the triangle or tetrahedron [38, 39]. Their methods are a natural extension of the study of spectral methods in one dimension relying on the use of Legendre polynomials, which are orthogonal on the interval $(-1, 1)$ [40, 41, 42, 43]. It was not until recently that such ideas have started to be posed in the context of vector finite element methods and it remains to be seen if such ideas can be made useful when arbitrary material properties varying spatially over an element are introduced [44, 45, 46, 47].

All basis functions in this thesis will be related to one-dimensional Legendre polynomials. These functions have been known for some time in one dimension within the finite element community as the Babuška-Szabó basis [48] written as integrals of Legendre polynomials,

or within the spectral method community as the Shen basis [49] written as particular linear combinations of Legendre polynomials. They are the same as the one-dimensional polynomials in [37] written in terms of Jacobi polynomials. All three representations are identical. These polynomials are hierarchical, and lead to sparse and well-conditioned matrices in the presence of constant coefficient problems. This thesis makes use of recent advances in fast polynomial transforms [50, 51] combined with classical results regarding integrals of products of three Legendre polynomials [52] to extend these sparsity properties to variable coefficient problems.

To extend the method to higher dimensions, tensor products of these integrated Legendre polynomials are used. By doing so, local element problems can be solved iteratively by using variants of the constant coefficient fast direct solver for spectral Legendre-Galerkin methods [49] as a preconditioner when combined with the fast eigenvalue and eigenvector routines of [53, 54, 55, 56, 57] which leverage one-dimensional fast multipole methods [58, 59, 60, 61]. To avoid large numbers of iterations in situations where elements differ in shape from the square, this thesis describes a mesh generation technique which constructs a quadtree mesh [62] away from interfaces and boundaries, with only a few layers of elements fitted to the interfaces and boundaries. The mesh generation approach belongs to the family of superposition methods [63]. Creating high order boundary fitted meshes uses a variety of mesh generation techniques, including implicit level set function projections [64], volume fraction determination and pillow layer insertion [65], mesh smoothing [66], and mesh optimization [67, 68].

In the quadtree mesh, arbitrary levels of refinement lead to non-conforming meshes. That is, a single element may have multiple neighbors along a given edge. To enforce continuity of the solution along such edges, appropriate constraint equations using relationships between Legendre polynomials and Legendre polynomials under affine transformations are formulated. These relationships can be expressed recursively either by the method presented in [69], or by a similar method described in this thesis. When dealing with high polynomial degree, an alternative algorithm is needed, which is developed as part of this research.

While low frequency problems can be effectively solved using Krylov subspace methods with preconditioners suitable for nearby static problems [70] (such as standard domain decomposition methods [71] or multigrid methods [72]), high frequency problems remain difficult to precondition effectively due to the indefinite and/or complex-symmetric nature of their associated discretized systems [73]. Efforts to formulate improved preconditioners for discretizations of the Helmholtz equation include extensions of domain decomposition methods, variants on multigrid methods, shifted-Laplacian preconditioners, and combinations of these methods, e.g. multigrid applied to a shifted-Laplacian preconditioner (see [74]

for an overview of such methods). In addition, more recently, a type of domain decomposition method called sweeping preconditioner has been proposed for second-order finite difference discretizations of the Helmholtz equation which can be applied with near linear computational complexity [75, 76].

The iterative method proposed in this thesis is closely related to domain decomposition methods of finite element tearing and interconnecting (FETI) type [77]. In particular, ideas from dual-primal FETI (FETI-DP) [78, 79, 80] and their extension to the Helmholtz problem (FETI-DPH) [81] are used. Recent experimental evidence [82] suggests that FETI-DPH can be effective for high frequency Helmholtz problems when compared to other domain decomposition methods if a suitably chosen coarse space of plane waves is used to augment the primal constraints.

This thesis makes two primary modifications to FETI-DPH. First, rather than augment the coarse space with carefully chosen plane waves, a finite element method where continuity between element subdomains is imposed via a hierarchy of weak constraints is formulated. The method is globally conforming (unlike a mortar method [83]), but the flexibility of the weak constraints allows construction of a non-conforming coarse space used in a corresponding FETI-DP domain decomposition method. The thesis shows experimentally that, by choosing the size of the coarse space based on a dispersion error criterion [14], the number of iterations in the domain decomposition method depends only weakly on the frequency. Second, Robin boundary conditions [84, 85] between subdomains are used to eliminate non-physical resonant frequencies which arise in the FETI-DPH method. This robustness can come at a computational cost, increasing the number of iterations required for convergence.

The method is applicable to interior problems with Dirichlet or Robin boundary conditions, and the thesis shows that exterior Helmholtz problems can be treated by introducing perfectly matched layers (PML) [86, 87]. With suitable parameter choices, PML do not adversely affect convergence rates for the iterative method. The weak continuity constraints between elements are sufficiently general so as to allow irregular mesh refinement and polynomial mismatch between adjacent elements, making mesh and element degree refinement possible.

1.3 Thesis Outline

The writing of this thesis shares an important view described in the resolution adopted by the United Nations General Assembly containing the decision on IYL 2015. Broadly speaking, IYL 2015 aims to "promote improved public and political understanding of the central role of light in the modern world" [1]. In keeping with this idea, this thesis describes

the finite element method with nothing more than a strong foundation of calculus and linear algebra so as to keep as broad an audience as possible. Admittedly, possessing background knowledge in convex and numerical optimization [88, 89, 90] and numerical linear algebra [25, 91, 92, 93] is helpful (one might argue that a strong foundation in calculus and linear algebra *subsumes* these topics). This goes counter to the standard approach used in most applied mathematics treatments where modern functional analysis is a prerequisite [27, 29, 33, 48, 94, 95, 96]. Rather, the treatment is an extension of the engineering literature on the subject [12, 28, 97, 98, 99].

The first part of the thesis studies the one-dimensional formulation of the finite element method applied to a prototypical differential equation characteristic of transmission lines and one-dimensional electromagnetic problems. To this end, Chapter 2 outlines the relationships between classical numerical differentiation, numerical integration, and polynomial interpolation in one dimension. An emphasis is placed on writing these rules using vector notation to simplify their treatment, as well as connecting these formulae to classical orthogonal polynomials [100]. Both of these ideas simplify construction of the fast and accurate finite element solver in one dimension. Chapter 3 guides the reader through the theory of the classical finite element method. The treatment anticipates modifications needed to construct a fast adaptive solver. Chapter 4 describes the necessary modifications required to obtain the one-dimensional fast and accurate solver, and includes examples demonstrating its efficiency and accuracy. The one-dimensional formulation is designed to parallel the later development in higher dimensions. In fact, Chapter 5 extends the method to wave equations in two dimensions (one spatial and one temporal) using the concept of space-time and includes additional wave propagation examples. This chapter introduces certain ideas necessary to extending the one-dimensional method to higher spatial dimensions without having to treat complicated geometries.

The second part of the thesis generalizes the one-dimensional finite element solver to higher spatial dimensions, using two dimensions as a prototypical example. Chapter 6 mirrors Chapter 3, describing the Ritz and Galerkin methods with a focus on constraint equations. Comments on how to generalize from one-dimensional elements to higher dimensional elements are made. Chapter 7 uses the one-dimensional basis functions from Chapter 4 to solve PDEs on the square, and the key differences in one and higher dimensions are highlighted. Chapter 8 describes how to generalize from the square to planar four sided curvilinear domains with a focus on accurate representation of curvilinear geometry. Chapter 9 first describes an approach used to construct non-conforming meshes for more complicated domains, then describes an adaptive finite element method suitable for these types of meshes, focusing on how to impose continuity constraints between elements. The chapter concludes

with the method applied to electrostatic problems and electromagnetic scattering problems. Developments in the previous three chapters are relied upon extensively throughout. Finally, Chapter 10 shows how to solve the resulting system of equations from Chapter 9 using iterative methods. In particular, Chapter 10 describes a domain decomposition algorithm applicable to both Poisson and Helmholtz problems. Key differences between Poisson and Helmholtz problems are highlighted, and connections to previous domain decomposition algorithms are discussed. Conclusions are summarized in Chapter 11, which also includes a description of future work.

For those well acquainted with the finite element method and numerical methods, Chapter 2, Chapter 3, and the first part of Chapter 6 will be primarily review. These chapters are intended to provide an appropriate foundation for the work that follows. Therefore, readers possessing these prerequisites may proceed directly to Chapter 4 for the one-dimensional development, Chapter 5 for the space-time development, or Chapters 7 through 10 for the higher-dimensional development, should they feel so inclined.

As a final note, each chapter begins with a brief introduction outlining the content to follow. Thereafter, the remaining body of each chapter is written in an expository style using the editorial "we," which is used to represent "the author and the reader." This style is common in mathematical writing. As a consequence, the "we" used in this thesis does not indicate multiple co-authors speaking in unison. All contributions made in this thesis are solely those of the author. Professor Steve McFee, acting as supervisor, performed an editorial role in the writing of this thesis, providing comments, suggestions, and possible revisions as drafts of each chapter were written.

Chapter 2

Preliminaries

Before treating the finite element method in one dimension, this chapter describes connections between numerical integration (numerical quadrature), numerical differentiation (finite difference formulae), and polynomial interpolation in one dimension. The three topics are covered in a wide variety of introductory numerical analysis and numerical method textbooks [101, 102, 103, 104, 105, 106, 107]. Why then devote a chapter to these topics? To demonstrate that high accuracy numerical integration and differentiation of a function depends on how closely a corresponding polynomial interpolant approximates that given function. This will later extend to the finite element method where, on each element, it is imperative that basis functions possess good approximation properties.

In addition, this chapter emphasizes the importance of linear algebra notation in the treatment of numerical integration, differentiation, and interpolation. This is not typical of many numerical method textbooks and, while this may not seem important, greatly simplifies developments when manipulating basis functions in the finite element method. Of particular interest is the use of vectors of orthogonal polynomials to represent Lagrange interpolating polynomials, the derivation of differentiation matrices for those vectors of polynomials, and overloaded notation to symbolize entrywise integration for vectors and matrices. These ideas will be recurrent throughout the thesis.

2.1 Interpolation-based Numerical Integration

Many textbooks begin their description of one-dimensional numerical integration schemes with a discussion of the so-called Newton-Cotes quadrature rules. These rules compute an approximation to

$$I[f] = \int_a^b f(x)\,dx \tag{2.1}$$

where $f : \mathbb{R} \to \mathbb{R}$ is a scalar function of one variable x, and $I[f]$ is the integral of f over the interval $x \in (a, b)$ (when the context is clear, we simply write I instead of $I[f]$). Using the numerically stable invertible map [108]

$$x = \frac{(b+a) + (b-a)\,u}{2}, \tag{2.2}$$

$$u = \frac{(x-a) - (b-x)}{b-a}, \tag{2.3}$$

the integral I becomes

$$I = \frac{b-a}{2} \int_{-1}^{+1} f\left(\frac{(b+a) + (b-a)\,u}{2}\right) du. \tag{2.4}$$

Since this map is always defined (as the case $a = b$ yields a trivial integral of zero and is of no practical value), Newton-Cotes quadrature rules are typically defined on the canonical interval $u \in (-1, +1)$, and are then modified to suit the appropriate interval of integration when needed. Rather than use u, we develop the theory using $x \in (-1, +1)$ in keeping with standard notation[1]. Since the integral of a function can be thought of as a generalized summation (think Riemann sums), it is natural to look for quadrature rules of the form

$$I \approx \sum_{i=0}^{n} w_i f(x_i), \tag{2.5}$$

where the w_i are a set of coefficients called weights, and the x_i are a set of points often called nodes or abscissae.

To obtain such a quadrature rule, the key idea is to first approximate the function f with a polynomial p. Since the integrals of monomials on the interval $(-1, +1)$ are given by

$$I[x^n] = \int_{-1}^{+1} x^n\, dx \tag{2.6}$$

$$= \left.\frac{x^{n+1}}{n+1}\right|_{-1}^{+1} \tag{2.7}$$

$$= \frac{1 - (-1)^{n+1}}{n+1} \tag{2.8}$$

$$= \begin{cases} \dfrac{2}{n+1} & \mod(n, 2) = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{2.9}$$

[1]The distinction will only become important when dealing with several intervals in later chapters.

by the linearity of integration, it follows that the integral of any polynomial over that same interval is known. This reduces the numerical integration problem (find a suitable quadrature rule to closely approximate the integral of f) to that of the polynomial interpolation problem (find an appropriate polynomial to closely approximate the function f). If we are interested in a quadrature rule of the form (2.5), then the simplest approach is to approximate f as a linear combination of interpolatory Lagrange polynomials over the interval $(-1, +1)$. That is, we take

$$p(x) = \sum_{i=0}^{n} f(x_i)\, l_i(x) \tag{2.10}$$

where l_i is the Lagrange polynomial of degree n which is 1 at the node x_i, and zero at all other nodes x_j where $j \neq i$. In one dimension, the Lagrange polynomials can be written explicitly as

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{2.11}$$

Using this expression for p as a surrogate for f, we find that

$$\begin{aligned}
I &= \int_{-1}^{+1} f(x)\, dx && (2.12)\\
&\approx \int_{-1}^{+1} p(x)\, dx && (2.13)\\
&\approx \int_{-1}^{+1} \left[\sum_{i=0}^{n} f(x_i)\, l_i(x) \right] dx && (2.14)\\
&\approx \sum_{i=0}^{n} \underbrace{\left[\int_{-1}^{+1} l_i(x)\, dx \right]}_{w_i} f(x_i) && (2.15)\\
&\approx \sum_{i=0}^{n} w_i f(x_i), && (2.16)
\end{aligned}$$

which is precisely the form of the quadrature rule that was anticipated in (2.5). A Newton-Cotes quadrature rule chooses an *equidistant* arrangement of nodes over the interval $(-1, +1)$ [106]. That is,

$$x_i = -1 + \frac{2}{n} i \tag{2.17}$$

for $i = 0, 1, ..., n$. Note that the interpolation nodes need not be inside the interval $(-1, +1)$ and, indeed, that is not the case for the equally spaced nodes which include the endpoints of the interval. Once the distribution of nodes is set, we then proceed to compute the weights, which amounts to integrating the Lagrange polynomials over the interval. Integrating the Lagrange polynomials of degree n appears to be quite difficult for large n. That being said,

one can determine an algorithm to do just that by appealing to linear algebra.

We begin by noting that the Lagrange polynomial of degree n can be written as

$$l_i(x) = \sum_{k=0}^{n} \alpha_{i,k} x^k, \tag{2.18}$$

where $x^0 = 1$. That is, we express the degree n Lagrange polynomial as a linear combination of $n+1$ monomials, which form a linearly independent set of functions on the interval $(-1, +1)$ (of course, numerically, this set is almost dependent for large n due to finite precision). Let

$$\bar{l}(x) = \begin{bmatrix} l_0(x) \\ l_1(x) \\ \vdots \\ l_n(x) \end{bmatrix}, \quad \bar{x}(x) = \begin{bmatrix} 1 \\ x \\ \vdots \\ x^n \end{bmatrix}, \quad \underline{A} = \begin{bmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n} \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n,0} & \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix}, \tag{2.19}$$

then $\bar{l}(x) = \underline{A}\bar{x}(x)$. To determine the entries of the matrix $\underline{A}$, we use the fact that the Lagrange polynomials are interpolatory. Thus,

$$\bar{l}(x_i) = \underline{A}\bar{x}(x_i) \tag{2.20}$$

$$\bar{e}_{i+1} = \underline{A}\bar{x}(x_i), \tag{2.21}$$

where $\bar{e}_i$ is the unit vector with a 1 in the ith coordinate, and zeros elsewhere. This is true for all x_i with $i = 0, 1, ..., n$, so that taking these $n+1$ expressions and writing them together yields

$$\underbrace{\begin{bmatrix} \bar{e}_1 & \bar{e}_2 & \cdots & \bar{e}_{n+1} \end{bmatrix}}_{\underline{I}_{n+1}} = \underline{A} \underbrace{\begin{bmatrix} \bar{x}(x_0) & \bar{x}(x_1) & \cdots & \bar{x}(x_n) \end{bmatrix}}_{\underline{V}^T}, \tag{2.22}$$

where $\underline{V}$ is the Vandermonde matrix and $\underline{I}_{n+1}$ is the $(n+1) \times (n+1)$ identity matrix. Since the monomials are a set of linearly independent functions, so long as the nodes x_i are distinct, the columns of $\underline{V}^T$ will form a set of linearly independent vectors, and thus $\underline{A} = \underline{V}^{-T}$.

We now rewrite the integral using this matrix notation. In particular, let

$$\bar{f} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}, \quad \bar{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}, \tag{2.23}$$

and overload the integration symbol to mean integration entrywise for vectors, then

$$I = \int_{-1}^{+1} f(x)\, dx \tag{2.24}$$

$$\approx \int_{-1}^{+1} \bar{f}^T \bar{l}(x)\, dx \tag{2.25}$$

$$\approx \int_{-1}^{+1} \bar{f}^T \underline{V}^{-T} \bar{x}(x)\, dx \tag{2.26}$$

$$\approx \underbrace{\bar{f}^T \underline{V}^{-T} \int_{-1}^{+1} \bar{x}(x)\, dx}_{\bar{w}}. \tag{2.27}$$

Note that the weights in (2.16) are exactly the same as those found in (2.27), but, given the nodes x_i, we can directly compute said weights as the Vandermonde matrix is determined, and the vector $\int_{-1}^{+1} \bar{x}(x)\, dx$ can be computed exactly through the use of (2.9). Thus, we solve

$$\underline{V}^T \bar{w} = \int_{-1}^{+1} \bar{x}(x)\, dx \tag{2.28}$$

using LU factorization to determine the weight vector $\bar{w}$.

Table 2.1 gives the values of the weights for increasing n. These were computed using the MATLAB code given in Algorithm 2.1, which is included to emphasize the simplicity of the approach. Note that for $n = 8$ and $n = 10$, we observe negative weights. If we continue to increase n, we find that the weights grow in magnitude. In particular, for $n = 30$, we have weights ranging from 10^{-2} to 10^4. This behavior is also observed in the condition number of $\underline{V}^T$, which MATLAB estimates to be 5.6424×10^{13}. The weights continue to grow as n increases. To understand why this is, we note that polynomial interpolation in equally spaced nodes is ill-conditioned, and this ill-conditioning is reflected in the Vandermonde matrix [109].

To contrast with these classical Newton-Cotes quadrature rules, let us consider redistributing the nodes. As an example, consider the nodes given by

$$x_i = \cos\left(\frac{\pi}{n} i\right), \tag{2.29}$$

for $i = 0, 1, ..., n$. Rules of this type are called Clenshaw-Curtis quadrature rules and the nodes are Chebyshev nodes [110]. In this case, for $n = 30$, we have weights which are all positive, between 0 and 0.105. Here, the condition number is 1.1773×10^{11}, two orders of magnitude smaller than the corresponding equally spaced case. Figure 2.1 gives the weights as a function of their position for this particular Clenshaw-Curtis quadrature rule. If we

Algorithm 2.1 Newton-Cotes weights for equally spaced nodes on the interval $(-1, +1)$.

```
% Equally spaced nodes
i = 0:n;
x_i = -1 + (2/n)*i;

% Construct the transpose of the Vandermonde matrix
VT = zeros(n+1,n+1);
for k = 1:n+1
    VT(k,:) = x_i.^(k-1);
end

% Compute the integrals of monomials on the interval (-1,+1)
I_monomial = (2./(i.'+1)).*(mod(i,2) == 0);

% Compute the weights
w = VT\I_monomial;
```

Table 2.1: Newton-Cotes weights for equally spaced nodes on the interval $(-1, +1)$.

n	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1	1.0000	1.0000									
2	0.3333	1.3333	0.3333								
3	0.2500	0.7500	0.7500	0.2500							
4	0.1556	0.7111	0.2667	0.7111	0.1556						
5	0.1319	0.5208	0.3472	0.3472	0.5208	0.1319					
6	0.0976	0.5143	0.0643	0.6476	0.0643	0.5143	0.0976				
7	0.0869	0.4140	0.1531	0.3459	0.3459	0.1531	0.4140	0.0869			
8	0.0698	0.4154	-0.0655	0.7405	-0.3203	0.7405	-0.0655	0.4154	0.0698		
9	0.0638	0.3514	0.0241	0.4318	0.1290	0.1290	0.4318	0.0241	0.3514	0.0638	
10	0.0537	0.3551	-0.1621	0.9099	-0.8703	1.4275	-0.8703	0.9099	-0.1621	0.3551	0.0537

continue to increase n, even for this well-behaved set of nodes, we run into numerical round-off errors for $n = 40$ and larger. This suggests a different approach is needed.

Suppose that instead of expressing the polynomial approximation p to f as a linear combination of monomials, we instead choose to write

$$l_i(x) = \sum_{k=0}^{n} \tilde{\alpha}_{i,k} p_k(x), \tag{2.30}$$

where p_k is the kth orthonormal polynomial on the interval $(-1, +1)$ (with respect to the

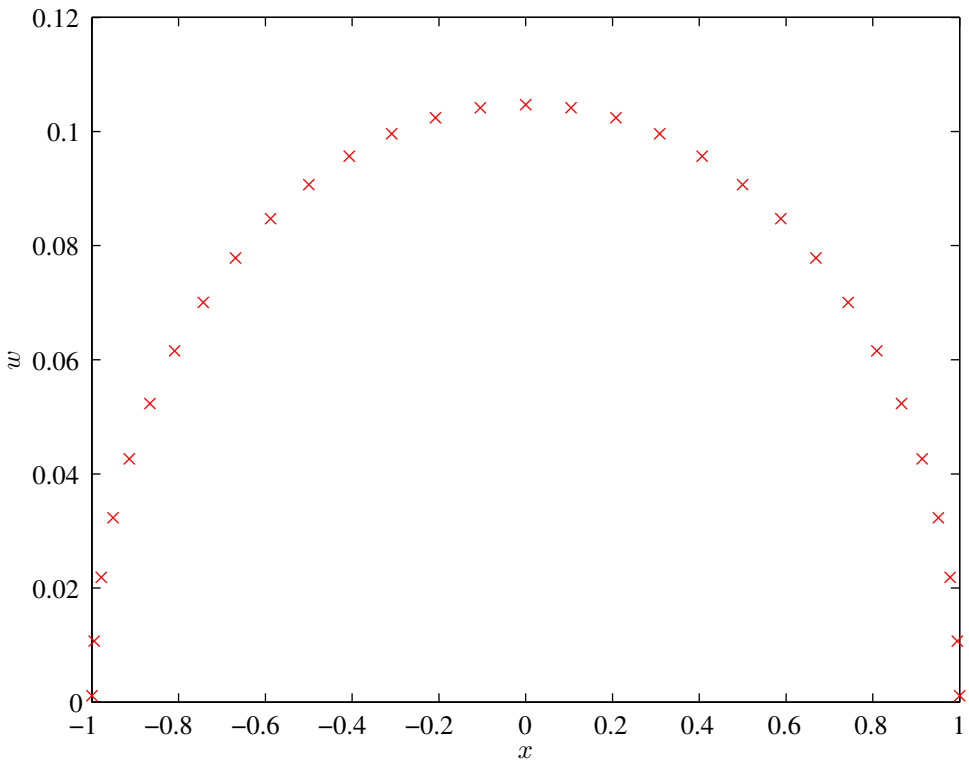

Figure 2.1: Clenshaw-Curtis weights as a function of the node positions $x_i = \cos\left(\frac{\pi}{n}i\right)$ for $i = 0, 1, ..., n$.

unit weight function) [100]. Then, repeating the same process as before, we let

$$\bar{p}(x) = \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_n(x) \end{bmatrix}, \quad \underline{\tilde{A}} = \begin{bmatrix} \tilde{\alpha}_{0,0} & \tilde{\alpha}_{0,1} & \cdots & \tilde{\alpha}_{0,n} \\ \tilde{\alpha}_{1,0} & \tilde{\alpha}_{1,1} & \cdots & \tilde{\alpha}_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\alpha}_{n,0} & \tilde{\alpha}_{n,1} & \cdots & \tilde{\alpha}_{n,n} \end{bmatrix}, \tag{2.31}$$

and observe that $\bar{l}(x) = \underline{\tilde{A}}\bar{p}(x)$. By evaluating this expression at the interpolation nodes of the Lagrange polynomials and collecting the columns of the $n+1$ vector equations, we obtain

$$\underbrace{\begin{bmatrix} \bar{e}_1 & \bar{e}_2 & \cdots & \bar{e}_{n+1} \end{bmatrix}}_{\underline{I}_{n+1}} = \underline{\tilde{A}} \underbrace{\begin{bmatrix} \bar{p}(x_0) & \bar{p}(x_1) & \cdots & \bar{p}(x_n) \end{bmatrix}}_{\underline{\tilde{V}}^T}. \tag{2.32}$$

Since the orthonormal polynomials up to degree n are an independent set of functions on the interval $(-1, +1)$, we have that $\underline{\tilde{V}}$, the generalized Vandermonde matrix, has a set of linearly independent rows as long as the interpolation nodes are distinct. Thus $\underline{\tilde{A}} = \underline{\tilde{V}}^{-T}$,

and we can write the quadrature rule as

$$I \approx \bar{f}^T \underbrace{\tilde{\underline{V}}^{-T} \int_{-1}^{+1} \bar{p}\,(x)\,dx}_{\bar{w}}. \tag{2.33}$$

The difficulty with this formulation resides in computing the integrals $\int_{-1}^{+1} \bar{p}\,(x)\,dx$.

To circumvent this apparent difficulty, we attack the problem of integrating the orthonormal polynomials on the interval by appealing to some of their fundamental properties. Since explicit formulae for orthonormal polynomials on the interval are known, it is possible to perform this integration directly. On the interval $(-1,+1)$ with unit weight function, the classical orthonormal polynomials are the Legendre polynomials given by the three-term recurrence relation

$$p_0\,(x) = \frac{1}{\sqrt{2}}, \tag{2.34}$$

$$p_1\,(x) = \sqrt{\frac{3}{2}}x, \tag{2.35}$$

$$p_{k+1}\,(x) = \frac{\sqrt{2k+3}}{k+1}\left[\sqrt{2k+1}xp_k\,(x) - \frac{k}{\sqrt{2k-1}}p_{k-1}\,(x)\right], \quad k = 1,2,... \tag{2.36}$$

To determine their integrals on the interval $(-1,+1)$, we will need three properties. First, $p_k\,(-x) = p_k\,(x)$ when k is even (that is, p_{2l} are even functions with l a positive integer). Second, $p_k\,(-x) = -p_k\,(x)$ when k is odd (so p_{2l+1} are odd functions). Finally,

$$\int_{-1}^{+1} p_i\,(x)\,p_j\,(x)\,dx = \delta_{i,j}, \tag{2.37}$$

where $\delta_{i,j}$ is the Kronecker delta function (for these properties, see [100] under Jacobi polynomials with $\alpha = \beta = 0$). Thus,

$$\int_{-1}^{+1} p_{2l+1}\,(x)\,dx = 0 \tag{2.38}$$

for all $l \geq 0$ by the second property. In addition, consider

$$\int_{-1}^{+1} p_{2l}\,(x)\,p_{2k}\,(x)\,dx = \delta_{2l,2k}. \tag{2.39}$$

If $k = 0$, then $p_{2k}(x) = p_0(x) = 1/\sqrt{2}$ and we obtain

$$\frac{1}{\sqrt{2}} \int_{-1}^{+1} p_{2l}(x)\, dx = \delta_{2l,0} \tag{2.40}$$

so that if $l = 0$, then $\int_{-1}^{+1} p_0(x)\, dx = \sqrt{2}$, and $l \neq 0$ yields a zero result. Thus, if we return to our problem, we obtain

$$\overline{w} = \underline{\tilde{V}}^{-T} \int_{-1}^{+1} \bar{p}(x)\, dx \tag{2.41}$$

$$= \underline{\tilde{V}}^{-T} \sqrt{2}\, \overline{e}_1 \tag{2.42}$$

where $\overline{e}_1$ is the unit vector with its first entry one, and all other entries zero. We use the recurrence relation for the orthonormal polynomials to fill the generalized Vandermonde matrix $\underline{\tilde{V}}$. On a computer, computing the weights reduces to solving

$$\underline{\tilde{V}}^T \overline{w} = \sqrt{2}\, \overline{e}_1 \tag{2.43}$$

which can be done using LU factorization.

To illustrate the difference between the standard Vandermonde approach and this generalized Vandermonde construction, consider the Clenshaw-Curtis quadrature rule with $n = 30$ (note that the equally spaced nodes remain unusable, even when using orthonormal polynomials in place of monomials). Earlier we said that the condition number of $\underline{V}^T$ was 1.1773×10^{11}. Using this orthonormal construction, we obtain a condition number for $\underline{\tilde{V}}^T$ of 7.2859. In fact, with this reduction in the condition number, no significant numerical round-off errors for $n = 40$ and above are observed. It should be noted that while this difference is considerable, for the Clenshaw-Curtis quadrature weights there exists an algorithm which is far more efficient than the one presented here [111]. The method exploits the structure of the nodes and utilizes the Fast Fourier Transform (FFT) to compute the weights in $\mathcal{O}(n \log_2 n)$ floating point operations (FLOPs), whereas naive LU factorization of the generalized Vandermonde matrix uses $\mathcal{O}(n^3)$ FLOPs[2]. Nevertheless, in getting to the generalized Vandermonde matrix with Chebyshev nodes, we have seen quite clearly that the equally spaced nodes and monomial basis are not suitable for arbitrary order polynomial approximation and computation. In addition, linear algebra has allowed us to concisely represent Lagrange interpolating polynomials and orthonormal Legendre polynomials, as well as to describe how to compute weights for a corresponding numerical integration scheme.

[2]There are specialized inversion algorithms for generalized Vandermonde systems, but these still require $\mathcal{O}(n^2)$ FLOPs [112].

2.2 Interpolation-based Numerical Differentiation

A large number of numerical analysis texts begin their discussion of finite difference approximations to derivatives through the Taylor series. In this section, our goal is to illustrate how constructing derivative approximations through Lagrange interpolation is directly analogous to the Taylor series approach, and that significantly more accurate derivatives can be obtained if we leave the standard equally spaced forward, backward, and centered differences in favor of more exotic ones. Again, an emphasis is placed on formulating the numerical differentiation rules using linear algebra.

To begin, let us try to approximate the kth derivative of a function f, evaluated at point x_0 using a linear combination of m function evaluations

$$f^{(k)}(x_0) \approx \sum_{j=0}^{m} \alpha_j f(x_j). \tag{2.44}$$

This is analogous to our approach for approximating integrals on the interval $(-1, +1)$, although here we expect some coefficients α_j to be negative, as the very definition of the derivative involves the difference (recall the conventional limit definition of the derivative). Consider the Taylor expansion

$$f(x_j) = f(x_0) + f^{(1)}(x_0)(x_j - x_0) + \frac{1}{2} f^{(2)}(x_0)(x_j - x_0)^2 + ... \tag{2.45}$$

centered at x_0. If we let $\tilde{x}_j = x_j - x_0$, then we can write

$$\alpha_j f(x_j) = \alpha_j \sum_{i=0}^{\infty} \frac{1}{i!} f^{(i)}(x_0) \tilde{x}_j^i. \tag{2.46}$$

Summing over j, we get

$$\sum_{j=0}^{m} \alpha_j f(x_j) = \sum_{j=0}^{m} \left[\alpha_j \sum_{i=0}^{\infty} \frac{1}{i!} f^{(i)}(x_0) \tilde{x}_j^i \right], \tag{2.47}$$

which can be rewritten by interchanging the order of summation on the right hand side, giving

$$\sum_{j=0}^{m} \alpha_j f(x_j) = \sum_{i=0}^{\infty} \left[\frac{1}{i!} f^{(i)}(x_0) \sum_{j=0}^{m} \alpha_j \tilde{x}_j^i \right]. \tag{2.48}$$

If we set

$$\frac{1}{k!}\sum_{j=0}^{m}\alpha_j\tilde{x}_j^k = 1 \tag{2.49}$$

$$\sum_{j=0}^{m}\alpha_j\tilde{x}_j^k = k! \tag{2.50}$$

and

$$\sum_{j=0}^{m}\alpha_j\tilde{x}_j^i = 0, \quad i = 0, 1, ..., k-1, \tag{2.51}$$

we get

$$\sum_{j=0}^{m}\alpha_j f\left(x_j\right) = f^{(k)}\left(x_0\right) + \sum_{i=k+1}^{\infty}\left[\frac{1}{i!}f^{(i)}\left(x_0\right)\sum_{j=0}^{m}\alpha_j\tilde{x}_j^i\right] \tag{2.52}$$

$$\sum_{j=0}^{m}\alpha_j f\left(x_j\right) \approx f^{(k)}\left(x_0\right) \tag{2.53}$$

which is precisely the linear combination we were searching for to approximate the derivative. If $m = k$ and the nodes x_j have been specified (and are all distinct), then (2.50) and (2.51) yield the linear system

$$\underline{V}^T\bar{\alpha} = k!\bar{e}_{k+1} \tag{2.54}$$

where $\underline{V} \in \mathbb{R}^{(k+1)\times(k+1)}$ and $\bar{\alpha} \in \mathbb{R}^{k+1}$ with entries

$$\left(\underline{V}\right)_{ij} = \tilde{x}_{i-1}^{j-1}, \quad \left(\bar{\alpha}\right)_i = \alpha_{i-1}. \tag{2.55}$$

Here, $\underline{V}$ is the Vandermonde matrix, and $\bar{\alpha}$ can be obtained by LU factorization.

To fix ideas, let us consider the forward differences. That is, let $x_j = j$. Note that usually, the forward difference is written with $x_j = jh$ where h is some small distance. We omit the factor of h because it will simply scale $\bar{\alpha}$ by h^{-k}. To see why, we observe that

$$\sum_{j=0}^{m}\alpha_j\tilde{x}_j^k = k! \tag{2.56}$$

$$\sum_{j=0}^{m}\alpha_j\left(jh\right)^k = k! \tag{2.57}$$

$$h^k\sum_{j=0}^{m}\alpha_j j^k = k! \tag{2.58}$$

$$\sum_{j=0}^{m}\alpha_j j^k = h^{-k}k!. \tag{2.59}$$

Similarly, for $i = 0, 1, ..., k-1$,

$$\sum_{j=0}^{m} \alpha_j \tilde{x}_j^i = 0 \tag{2.60}$$

$$\sum_{j=0}^{m} \alpha_j (jh)^i = 0 \tag{2.61}$$

$$h^i \sum_{j=0}^{m} \alpha_j j^i = 0 \tag{2.62}$$

$$\sum_{j=0}^{m} \alpha_j j^i = 0 \tag{2.63}$$

so that $\underline{V}^T \bar{\alpha} = k! \bar{e}_{k+1}$ becomes $\underline{V}^T \bar{\alpha} = h^{-k} k! \bar{e}_{k+1}$ which can be rewritten as

$$\underline{V}^T (h^k \bar{\alpha}) = k! \bar{e}_{k+1}. \tag{2.64}$$

It should be noted that in this setting, we can determine the truncation error directly as we have omitted the infinite sum in (2.52). For the forward difference, we obtain

$$T = \sum_{i=k+1}^{\infty} \left[\frac{1}{i!} f^{(i)}(x_0) \sum_{j=0}^{m} \alpha_j \tilde{x}_j^i \right] \tag{2.65}$$

$$= \frac{1}{(k+1)!} f^{(k+1)}(x_0) \sum_{j=0}^{m} \alpha_j (jh)^{k+1} + \sum_{i=k+2}^{\infty} \left[\frac{1}{i!} f^{(i)}(x_0) \sum_{j=0}^{m} \alpha_j \tilde{x}_j^i \right] \tag{2.66}$$

$$= \frac{1}{(k+1)!} f^{(k+1)}(x_0) \underbrace{h^{k+1} \sum_{j=0}^{m} \alpha_j j^{k+1}}_{\mathcal{O}(h)} + \underbrace{\sum_{i=k+2}^{\infty} \left[\frac{1}{i!} f^{(i)}(x_0) \sum_{j=0}^{m} \alpha_j \tilde{x}_j^i \right]}_{\mathcal{O}(h^2)} \tag{2.67}$$

with the first term being order h since each coefficient α_j contains a factor h^{-k}. This also gives us a systematic way to construct more accurate rules by setting the order h term to zero, and so on. Of course, if we add one such equation, in order to keep $\underline{V}$ square, we also must add an additional node x_{k+1}. If we let $k_{\text{eff}} + 1$ denote the number of equations to retain from our infinite sum, then $\underline{V} \in \mathbb{R}^{(m+1)\times(k_{\text{eff}}+1)}$ with $m = k_{\text{eff}}$ yields the coefficients for the kth derivative with accuracy $\mathcal{O}(h^{k_{\text{eff}}-k+1})$.

Similarly, for backward differences, we take $x_j = -j$ and find that the coefficients are precisely the same as those for the forward difference when k is even. When k is odd, we

change sign for all coefficients. To see why, note that

$$\sum_{j=0}^{m} \alpha_j \tilde{x}_j^k = k! \tag{2.68}$$

$$\sum_{j=0}^{m} \alpha_j (-j)^k = k! \tag{2.69}$$

$$(-1)^k \sum_{j=0}^{m} \alpha_j j^k = k! \tag{2.70}$$

$$\sum_{j=0}^{m} \alpha_j j^k = (-1)^k k! \tag{2.71}$$

thus only the right hand side changes sign, and the Vandermonde matrix remains unchanged (the other equations are unchanged as their right hand sides are zero).

Finally, for centered differences, we take $x_j = j$ but change the indexing so that j ranges from $-l$ to l (where $m = 2l$ and $m \geq k$). We also take $k_{\text{eff}} = m + (1 - \text{mod}\,(k, 2))$. Note that this means that $\underline{V}$ will not be square when k is even. We do this to emphasize the fact that for k even, the right hand side of

$$\underline{V}^T \bar{\alpha} = k! \bar{e}_{k+1} \tag{2.72}$$

remains in the range of the extended $\underline{V}^T$ due to the symmetry of the nodes. This is important because it explains why the truncation error for centered differences is always an even power of h. That is, the truncation error is given by $\mathcal{O}(h^{k_{\text{eff}}-k+1})$ where

$$k_{\text{eff}} - k + 1 = 2l + 1 - \text{mod}\,(k, 2) - k + 1 \tag{2.73}$$

$$= \underbrace{2\,(l+1)}_{\text{even}} - \underbrace{(k + \text{mod}\,(k, 2))}_{\text{even}}. \tag{2.74}$$

To solve for the coefficients $\bar{\alpha}$, we can apply QR factorization to the overdetermined system.

Now that we have confirmed how to derive arbitrarily high order finite difference formulae via the Taylor series and linear algebra, we consider polynomial interpolation to derive these same finite difference coefficients. First, recall that Lagrange polynomials can be written as

$$\bar{l}\,(x) = \underline{V}^{-T} \bar{x}\,(x)\,. \tag{2.75}$$

By the linearity of differentiation, we have that

$$\frac{d}{dx}\bar{l}\left(x\right) = \frac{d}{dx}\left[\underline{V}^{-T}\bar{x}\left(x\right)\right] \tag{2.76}$$

$$= \underline{V}^{-T}\frac{d}{dx}\bar{x}\left(x\right) \tag{2.77}$$

where the differentiation operator $\frac{d}{dx}$ is to be applied entrywise. Since $\bar{x}\left(x\right)$ is the vector containing the monomials, we can write an explicit expression for the component-wise differentiation by observing

$$\frac{d}{dx}\bar{x}\left(x\right) = \frac{d}{dx}\begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^n \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2x \\ \vdots \\ nx^{n-1} \end{bmatrix}. \tag{2.78}$$

Rewriting this operator as a matrix yields

$$\frac{d}{dx} = \underline{D} = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 2 & \ddots & & \\ & & \ddots & 0 & \\ & & & n & 0 \end{bmatrix} \tag{2.79}$$

so that $\frac{d}{dx}\bar{x}\left(x\right) = \underline{D}\bar{x}\left(x\right)$. We note that $\underline{D}$ is a nilpotent matrix (that is, $\underline{D}^{n+1} = 0$). This agrees conceptually with the fact that the monomial x^n vanishes after $n+1$ derivatives. If we use the Lagrange polynomials to approximate the function f, we obtain

$$f\left(x\right) \approx \bar{f}^T\bar{l}\left(x\right) \tag{2.80}$$

$$f\left(x\right) \approx \bar{f}^T\underline{V}^{-T}\bar{x}\left(x\right) \tag{2.81}$$

$$\frac{d}{dx}f\left(x\right) \approx \bar{f}^T\underline{V}^{-T}\underline{D}\bar{x}\left(x\right) \tag{2.82}$$

$$f^{(1)}\left(x_0\right) \approx \bar{f}^T\underbrace{\underline{V}^{-T}\underline{D}\bar{x}\left(x_0\right)}_{\bar{\alpha}} \tag{2.83}$$

and so the finite difference coefficients for the first derivative of f evaluated at x_0 are given by the vector $\bar{\alpha}$ that satisfies

$$\underline{V}^T\bar{\alpha} = \underline{D}\bar{x}\left(x_0\right). \tag{2.84}$$

To obtain the forward difference coefficients, we let $x_j = j$ which determines $\underline{V}$ and which reduces $\bar{x}(x_0) = \bar{x}(0) = \bar{e}_1$. The product $\underline{D}\bar{e}_1 = \bar{e}_2$, and so $\underline{V}^T\bar{\alpha} = \bar{e}_2$. This mirrors the result set forth by the Taylor series. In fact, we can apply this idea to higher derivatives. We note that $\underline{D}^k\bar{e}_1 = k!\bar{e}_{k+1}$ and so

$$f^{(k)}(0) = \bar{f}^T \underbrace{\underline{V}^{-T}\underline{D}^k\bar{e}_1}_{\bar{\alpha}} \tag{2.85}$$

with

$$\underline{V}^T\bar{\alpha} = \underline{D}^k\bar{e}_1 \tag{2.86}$$

$$\underline{V}^T\bar{\alpha} = k!\bar{e}_{k+1}. \tag{2.87}$$

This is precisely what we found for the Taylor series. An error analysis of equally spaced finite difference formulae beginning from polynomial interpolation error analysis can be performed, although there are subtleties that make this less straightforward [107]. Such analysis results in the same dependence on powers of the spacing h as described earlier. Table 2.2 gives the finite difference coefficients as computed by the method of Lagrange interpolation for the forward differences while Table 2.3 gives the finite difference coefficients for the centered differences.

As with numerical integration via Lagrange interpolation, we note that inversion of the Vandermonde matrix becomes impossible numerically for large n. This renders representing derivatives of functions to arbitrary accuracy quite difficult since, in general, we would like to take n large to achieve such a result. Just as in the case of integration, we consider orthonormal polynomials to remedy the problem. Instead of expressing the interpolatory Lagrange polynomials in terms of the monomials, we use the Legendre polynomials. Recall that $\bar{l}(x) = \tilde{\underline{V}}^{-T}\bar{p}(x)$ where $\bar{l}(x)$ is the vector containing the Lagrange polynomials, $\bar{p}(x)$ is the vector containing the orthonormal Legendre polynomials, and $\tilde{\underline{V}}$ is the generalized Vandermonde matrix (see (2.32) and its associated paragraph). Our goal here is to write

$$f^{(k)}(x_0) = \bar{f}^T \underbrace{\tilde{\underline{V}}^{-T}\tilde{\underline{D}}^k\bar{p}(x_0)}_{\tilde{\alpha}} \tag{2.88}$$

just as we did for the monomial expansion of the Lagrange polynomials. The difficulty lies in determining the structure of the differentiation matrix $\tilde{\underline{D}}$ for the Legendre polynomials.

Table 2.2: Forward finite difference coefficients.

k	n	α_0	α_1	α_2	α_3	α_4	α_5
	1	-1.0000	1.0000				
	2	-1.5000	2.0000	-0.5000			
1	3	-1.8333	3.0000	-1.5000	0.3333		
	4	-2.0833	4.0000	-3.0000	1.3333	-0.2500	
	5	-2.2833	5.0000	-5.0000	3.3333	-1.2500	0.2000

k	n	α_0	α_1	α_2	α_3	α_4	α_5	α_6
	1	1.0000	-2.0000	1.0000				
	2	2.0000	-5.0000	4.0000	-1.0000			
2	3	2.9167	-8.6667	9.5000	-4.6667	0.9167		
	4	3.7500	-12.8333	17.8333	-13.0000	5.0833	-0.8333	
	5	4.5111	-17.4000	29.2500	-28.2222	16.5000	-5.4000	0.7611

k	n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7
	1	-1.0000	3.0000	-3.0000	1.0000				
	2	-2.5000	9.0000	-12.0000	7.0000	-1.5000			
3	3	-4.2500	17.7500	-29.5000	24.5000	-10.2500	1.7500		
	4	-6.1250	29.0000	-57.6250	62.0000	-38.3750	13.0000	-1.8750	
	5	-8.0583	42.5333	-98.2250	129.6667	-106.0417	53.6000	-15.4083	1.9333

k	n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
	1	1.0000	-4.0000	6.0000	-4.0000	1.0000				
	2	3.0000	-14.0000	26.0000	-24.0000	11.0000	-2.0000			
4	3	5.8333	-31.0000	68.5000	-80.6667	53.5000	-19.0000	2.8333		
	4	9.3333	-55.5000	142.0000	-203.1667	176.0000	-92.5000	27.3333	-3.5000	
	5	13.3625	-87.7333	254.8167	-428.8000	458.0417	-318.1333	140.1500	-35.7333	4.0292

To do so, we use one additional property of the Legendre polynomials [40]:

$$\frac{d}{dx}\left[\frac{1}{\sqrt{2\left(n+1\right)+1}}p_{n+1}\left(x\right)-\frac{1}{\sqrt{2\left(n-1\right)+1}}p_{n-1}\left(x\right)\right]=\sqrt{2n+1}p_n\left(x\right). \tag{2.89}$$

The expression for the derivative of p_{n+1} in terms of lower degree p_j is found by rearranging this equation such that

$$\frac{1}{\sqrt{2\left(n+1\right)+1}}\frac{d}{dx}p_{n+1}=\sqrt{2n+1}p_n+\frac{1}{\sqrt{2\left(n-1\right)+1}}\frac{d}{dx}p_{n-1} \tag{2.90}$$

Table 2.3: Centered finite difference coefficients.

n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
2	-0.5000	0.0000	0.5000								
4	0.0833	-0.6667	0.0000	0.6667	-0.0833						
6	-0.0167	0.1500	-0.7500	0.0000	0.7500	-0.1500	0.0167				
8	0.0036	-0.0381	0.2000	-0.8000	0.0000	0.8000	-0.2000	0.0381	-0.0036		
10	-0.0008	0.0099	-0.0595	0.2381	-0.8333	-0.0000	0.8333	-0.2381	0.0595	-0.0099	0.0008

n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
2	1.0000	-2.0000	1.0000								
4	-0.0833	1.3333	-2.5000	1.3333	-0.0833						
6	0.0111	-0.1500	1.5000	-2.7222	1.5000	-0.1500	0.0111				
8	-0.0018	0.0254	-0.2000	1.6000	-2.8472	1.6000	-0.2000	0.0254	-0.0018		
10	0.0003	-0.0050	0.0397	-0.2381	1.6667	-2.9272	1.6667	-0.2381	0.0397	-0.0050	0.0003

n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
4	-0.5000	1.0000	0.0000	-1.0000	0.5000						
6	0.1250	-1.0000	1.6250	0.0000	-1.6250	1.0000	-0.1250				
8	-0.0292	0.3000	-1.4083	2.0333	-0.0000	-2.0333	1.4083	-0.3000	0.0292		
10	0.0068	-0.0834	0.4830	-1.7337	2.3181	0.0000	-2.3181	1.7337	-0.4830	0.0834	-0.0068

n	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
4	1.0000	-4.0000	6.0000	-4.0000	1.0000						
6	-0.1667	2.0000	-6.5000	9.3333	-6.5000	2.0000	-0.1667				
8	0.0292	-0.4000	2.8167	-8.1333	11.3750	-8.1333	2.8167	-0.4000	0.0292		
10	-0.0054	0.0834	-0.6440	3.4675	-9.2722	12.7417	-9.2722	3.4675	-0.6440	0.0834	-0.0054

which holds for any $n \geq 1$. We can then write this for the case $n-2$ to find

$$\frac{1}{\sqrt{2\left(n-1\right)+1}}\frac{d}{dx}p_{n-1} = \sqrt{2\left(n-2\right)+1}p_{n-2} + \frac{1}{\sqrt{2\left(n-3\right)+1}}\frac{d}{dx}p_{n-3} \tag{2.91}$$

and substitute this into the last term in (2.90) to obtain

$$\frac{1}{\sqrt{2\left(n+1\right)+1}}\frac{d}{dx}p_{n+1} = \sqrt{2n+1}p_n + \sqrt{2\left(n-2\right)+1}p_{n-2} + \frac{1}{\sqrt{2\left(n-3\right)+1}}\frac{d}{dx}p_{n-3}. \tag{2.92}$$

This process is repeated until no derivative terms remain on the right hand side. We finally multiply through by $\sqrt{2\left(n+1\right)+1}$ to obtain

$$\frac{d}{dx}p_{n+1} = \sqrt{2(n+1)+1}\left[\sqrt{2n+1}p_n + \sqrt{2\left(n-2\right)+1}p_{n-2} + ...\right]. \tag{2.93}$$

From this expression, we can compute the differentiation matrix entries. For example, consider the case with $n = 4$. Then

$$\frac{d}{dx}\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} 0 & & & & \\ \sqrt{3} & 0 & & & \\ 0 & \sqrt{5}\sqrt{3} & 0 & & \\ \sqrt{7} & 0 & \sqrt{7}\sqrt{5} & 0 & \\ 0 & \sqrt{9}\sqrt{3} & 0 & \sqrt{9}\sqrt{7} & 0 \end{bmatrix}\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \tag{2.94}$$

and we can observe that, in general, $\underline{\tilde{D}} = \text{diag}(\bar{h})\underline{H}\text{diag}(\bar{h})$ where

$$h_i = \sqrt{2(i-1)+1}, \quad H_{ij} = \begin{cases} \text{mod}\,(i+j,2) & j \leq i \\ 0 & \text{otherwise.} \end{cases} \tag{2.95}$$

Just as in the monomial basis case, $\underline{\tilde{D}}^{n+1} = 0$ and so $\underline{\tilde{D}}$ is a nilpotent matrix.

We can repeat the same exercise as with the monomial basis and compute the finite difference coefficients with this new basis. This is not particularly enlightening (we get the same coefficients). However, if we change the locations of the interpolation nodes, from equally spaced to, for example, the Chebyshev nodes, we obtain a situation where the generalized Vandermonde matrix condition number is small, and its factorization is accurate. This allows us to take derivatives of functions to high accuracy by increasing n. This is not always possible through the Taylor series approach. In effect, changing the basis for our Lagrange polynomials corresponds to discarding the Taylor series altogether and, instead, replacing it with an expansion in Legendre polynomials. We will exploit this approach when we discuss possible improvements to the classical finite element method in Chapter 4. Before that, we describe a classical finite element method.

Chapter 3

One-Dimensional Finite Element Methods

This chapter describes the classical finite element method in one dimension. The goal is to illustrate the classical method in such a way that anticipates the high accuracy domain decomposition approach put forth later in the thesis. First, the presentation covers the variational formulation and the Ritz method used to obtain the classical method. Unlike classical expositions, continuity and boundary conditions are explicitly imposed via additional constraint equations rather than enforced *a priori*. This gives rise to an associated constrained optimization problem whose solution solves a saddle point system. Second, the Galerkin method is used to obtain the same system and gives insight into the meaning of the Lagrange multipliers appearing in the optimization problem. When basis functions are chosen to interpolate the solution, the constraint equations are sparse, and a sparse basis for the null space of the constraint matrix exists. This basis is what allows the classical finite element method to sidestep the discussion of constraint equations. The constraint equations, which do not usually appear in a standard finite element method, will be crucial when describing domain decomposition methods.

3.1 The Variational Formulation

To keep our analysis general, we seek the unique solution to the boundary value problem (BVP)

$$-\frac{d}{dx}\left(\alpha\left(x\right)\frac{d\phi}{dx}\right)+\beta\left(x\right)\phi\left(x\right)=f\left(x\right),\qquad x\in\left(a,b\right), \tag{3.1}$$

subject to boundary conditions

$$\phi(a) = p, \tag{3.2}$$

$$\left[\alpha \frac{d\phi}{dx} + \gamma\phi\right]_{x=b} = q, \tag{3.3}$$

where we have imposed a Dirichlet boundary condition at $x = a$ and a Robin boundary condition at $x = b$ so as to cover the three most common boundary condition types (Neumann boundary conditions are a subset of Robin boundary conditions with $\gamma = 0$). Many other possible combinations of these three types of boundary conditions are possible, but we omit them for the sake of brevity. This BVP can be made into a Laplace, Poisson, or Helmholtz problem, each of which are of particular interest to the electrical engineer. Notice that ϕ satisfies the BVP in the open interval $\Omega = (a, b)$ but that it is assigned boundary values at the endpoints a and b. In this thesis, boundary statements should be interpreted in a limiting sense. That is, when we say, for example, that a Dirichlet boundary condition is imposed at $x = a$ by writing $\phi(a) = p$, we mean, more precisely, that the $\lim_{x\to a^+} \phi(x) = p$ where a^+ denotes that x approaches a from the right (a similar notation a^- would mean that x approaches a from the left). We use the less precise notation for economy when there is no ambiguity. Also, note that there is an implicit additional constraint that ϕ must satisfy, which is that ϕ must be continuous in (a, b). This can be seen if we recast the BVP as a variational problem

$$\delta F(\phi) = 0, \tag{3.4}$$

$$\phi(a) = p, \tag{3.5}$$

$$\phi \in C^0(\Omega), \tag{3.6}$$

where δF is the first variation of the functional

$$F(\phi) = \frac{1}{2}\int_a^b \alpha\left(\frac{d\phi}{dx}\right)^2 + \beta\phi^2 - 2\phi f\, dx + \left[\frac{1}{2}\gamma\phi^2 - q\phi\right]_{x=b}, \tag{3.7}$$

and $C^0(\Omega)$ is the space of all continuous functions on domain Ω. The ϕ satisfying (3.4)-(3.6) is called a weak solution of (3.1)-(3.3). To see why these two formulations are related, we take the first variation of F, given by

$$\delta F(\phi) = \lim_{\epsilon\to 0} \frac{F(\phi + \epsilon\delta\phi) - F(\phi)}{\epsilon}, \tag{3.8}$$

where ϵ is a small parameter and $\delta\phi \in C^0(\Omega)$ is a function which satisfies a homogeneous Dirichlet boundary condition at $x = a$. We are effectively assuming that ϕ is a function which produces a local extremum for the functional F. In that case, setting the first variation to zero is the same as finding a stationary solution ϕ to the functional F. For the sake of argument, if ϕ minimizes F, then $F(\phi) < F(\phi + \epsilon\delta\phi)$ for all $\epsilon > 0$ (where $\epsilon\delta\phi$ is interpreted as a small perturbation or variation of ϕ). If we treat F as a function of one parameter ϵ, then a necessary condition for ϕ to minimize F is for the derivative of F with respect to ϵ to be zero when ϵ approaches zero. This is directly analogous to taking the gradient of a multidimensional function and setting it to zero to find critical points. In principle, we must also check that ϕ does indeed minimize F, and that we have not found a saddle point, or local maximum. We can do this by computing the second variation and verifying that it is positive.

Before performing this check, we begin by computing the first variation

$$\begin{aligned}\frac{F(\phi+\epsilon\delta\phi)-F(\phi)}{\epsilon} = \frac{1}{\epsilon}\Bigg[&\frac{1}{2}\int_a^b \alpha\left(\frac{d\phi}{dx}+\epsilon\frac{d}{dx}\delta\phi\right)^2 + \beta(\phi+\epsilon\delta\phi)^2 - 2f(\phi+\epsilon\delta\phi)\,dx \\ &+\left[\frac{1}{2}\gamma(\phi+\epsilon\delta\phi)^2 - q(\phi+\epsilon\delta\phi)\right]_{x=b} \\ &-\frac{1}{2}\int_a^b \alpha\left(\frac{d\phi}{dx}\right)^2 + \beta\phi^2 dx - 2f\phi\,dx - \left[\frac{1}{2}\gamma\phi^2 - q\phi\right]_{x=b}\Bigg]. \quad (3.9)\end{aligned}$$

Squaring and canceling terms, we obtain

$$\begin{aligned}\frac{F(\phi+\epsilon\delta\phi)-F(\phi)}{\epsilon} = \frac{1}{2}\int_a^b \alpha\left(2\frac{d\phi}{dx}\frac{d}{dx}\delta\phi + \epsilon\left(\frac{d}{dx}\delta\phi\right)^2\right) + \beta\left(2\phi\delta\phi+\epsilon\delta\phi^2\right) - 2f\delta\phi\,dx \\ +\left[\frac{1}{2}\gamma\left(2\phi\delta\phi+\epsilon\delta\phi^2\right) - q\delta\phi\right]_{x=b}, \quad (3.10)\end{aligned}$$

which, in the limit as $\epsilon \to 0$, gives

$$\delta F(\phi) = \int_a^b \alpha\frac{d\phi}{dx}\frac{d}{dx}\delta\phi + (\beta\phi - f)\,\delta\phi\,dx + \left[(\gamma\phi - q)\,\delta\phi\right]_{x=b}. \quad (3.11)$$

Now, we apply integration by parts to the first term to transfer the derivative from $\delta\phi$ to $\alpha\frac{d\phi}{dx}$. We will assume that $\alpha(x)$ has a single discontinuity[1] at some $a < x_d < b$. That is,

[1]The case where there are finitely many discontinuities inside Ω will follow from this simpler treatment.

letting

$$u = \alpha \frac{d\phi}{dx}, \tag{3.12}$$

$$du = \frac{d}{dx}\left(\alpha \frac{d\phi}{dx}\right) dx, \tag{3.13}$$

$$v = \delta\phi, \tag{3.14}$$

$$dv = \frac{d}{dx}\delta\phi \, dx, \tag{3.15}$$

we obtain

$$\int_a^b u \, dv = \left[uv\Big|_a^{x_d^-} - \int_a^{x_d^-} v \, du \right] + \left[uv\Big|_{x_d^+}^b - \int_{x_d^+}^b v \, du \right], \tag{3.16}$$

where we have split the integration at the point of discontinuity x_d. If we had finitely many discontinuities of α in the interval (a, b), we would need to break up the integral into several integrals accordingly. This gives

$$\begin{aligned} \int_a^b \alpha \frac{d\phi}{dx}\frac{d}{dx}\delta\phi \, dx &= \alpha \frac{d\phi}{dx}\delta\phi\Big|_a^{x_d^-} + \alpha \frac{d\phi}{dx}\delta\phi\Big|_{x_d^+}^b - \int_a^b \delta\phi \frac{d}{dx}\left(\alpha \frac{d\phi}{dx}\right) dx \\ &= \alpha \frac{d\phi}{dx}\delta\phi\Big|_a^b - \int_a^b \frac{d}{dx}\left(\alpha \frac{d\phi}{dx}\right)\delta\phi \, dx + \alpha \frac{d\phi}{dx}\delta\phi\Big|_{x_d^+}^{x_d^-} \end{aligned} \tag{3.17}$$

where the integral on the right hand side is understood to be computed over each subinterval. Note that since we will be imposing the Dirichlet boundary condition $\phi(a) = p$, we must have $\delta\phi(a) = 0$ (as our perturbation $\phi + \epsilon\delta\phi$ must continue to satisfy the original boundary condition), giving

$$\delta F(\phi) = \int_a^b \left[-\frac{d}{dx}\left(\alpha \frac{d\phi}{dx}\right) + \beta\phi - f \right] \delta\phi \, dx + \alpha \frac{d\phi}{dx}\delta\phi\Big|_{x_d^+}^{x_d^-} + \left[\left(\alpha \frac{d\phi}{dx} + \gamma\phi - q \right) \delta\phi \right]_{x=b}. \tag{3.18}$$

Since $\delta\phi$ is continuous, this allows us to write

$$\delta F(\phi) = \int_a^b \left[-\frac{d}{dx}\left(\alpha \frac{d\phi}{dx}\right) + \beta\phi - f \right] \delta\phi \, dx + \alpha \frac{d\phi}{dx}\Big|_{x_d^+}^{x_d^-} \delta\phi(x_d) + \left[\left(\alpha \frac{d\phi}{dx} + \gamma\phi - q \right) \delta\phi \right]_{x=b}, \tag{3.19}$$

and since $\delta\phi$ is an arbitrary continuous function with homogeneous Dirichlet boundary con-

dition[2], by the fundamental lemma of the calculus of variations [113], setting $\delta F(\phi) = 0$ means that we must have

$$-\frac{d}{dx}\left(\alpha\frac{d\phi}{dx}\right) + \beta\phi - f = 0, \tag{3.20}$$

$$\alpha\frac{d\phi}{dx}\bigg|_{x=x_d^-} - \alpha\frac{d\phi}{dx}\bigg|_{x=x_d^+} = 0, \tag{3.21}$$

$$\left[\alpha\frac{d\phi}{dx} + \gamma\phi - q\right]_{x=b} = 0. \tag{3.22}$$

Equations (3.20) and (3.22) are precisely (3.1) and (3.3), respectively. We see that by taking the first variation of F, setting it to zero and imposing the Dirichlet boundary condition, along with requiring continuity of ϕ, we solve the original BVP satisfying the boundary condition of the third kind. In addition, we note that $\alpha\frac{d\phi}{dx}$ remains continuous at discontinuities of α naturally through the formulation.

To ensure that we are in fact minimizing the functional F, we take the second variation of F given by

$$\delta^2 F(\phi) = \lim_{\epsilon\to 0}\frac{\delta F(\phi + \epsilon\delta\phi) - \delta F(\phi)}{\epsilon}. \tag{3.23}$$

Using a process similar to computing the first variation (we are taking the first variation of the first variation), we obtain

$$\delta^2 F(\phi) = \int_a^b \alpha\left(\frac{d}{dx}\delta\phi\right)^2 + \beta(\delta\phi)^2\,dx + \left[\gamma(\delta\phi)^2\right]_{x=b}. \tag{3.24}$$

For ϕ to be the function which minimizes F, we require $\delta^2 F(\phi) > 0$ (in direct analogy with the second derivative test from any standard calculus course). Clearly, the problem parameters α, β, and γ have a strong influence over whether the stationary point corresponds to a local minimum. As a simple case, if $\alpha > 0$, $\beta > 0$, and $\gamma > 0$, then $\delta^2 F(\phi) > 0$ regardless of the perturbation $\delta\phi$, and we have in fact found a local minimum ϕ for the functional F.

3.2 The Ritz Method

Now that we have shown the equivalence of the BVP formulation with its variational formulation, we proceed with a discussion of how one can use the variational form to compute approximate solutions to the BVP. We begin by approximating ϕ by a linear combination of basis functions $\phi(x) = \sum_{i=1}^{M}\phi_i N_i(x)$, where ϕ_i are the coefficients to determine and N_i

[2]Sometimes referred to as an admissibile variation.

are the basis functions. This reduces the problem of finding a stationary point of F (an infinite-dimensional problem) into one of finding the coefficients ϕ_i which render stationary F for the given set of basis functions N_i (a finite-dimensional problem). Of course, this requires one to choose the set of basis functions, and there is no completely general, *a priori* optimal approach to make this choice. For now, we leave the choice unspecified[3].

Let us now solve the Ritz problem. First, note that

$$\phi = \sum_{i=1}^{M} \phi_i N_i \tag{3.25}$$

$$= \bar{\phi}^T \bar{N} \tag{3.26}$$

$$= \bar{N}^T \bar{\phi}, \tag{3.27}$$

where $\bar{\phi} = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_M \end{bmatrix}^T$ and $\bar{N}(x) = \begin{bmatrix} N_1(x) & N_2(x) & \cdots & N_M(x) \end{bmatrix}^T$ are vectors containing the coefficients ϕ_i to be determined, and the basis functions $N_i(x)$, respectively. Using this vector notation, we substitute the linear combination of basis functions into the functional F and obtain

$$F\left(\bar{\phi}\right) = \frac{1}{2}\int_a^b \alpha\left(\frac{d}{dx}\bar{N}^T\bar{\phi}\right)^2 + \beta\left(\bar{\phi}^T\bar{N}\right)^2 - 2\bar{\phi}^T\bar{N}f\,dx + \left[\frac{1}{2}\gamma\left(\bar{\phi}^T\bar{N}\right)^2 - q\bar{\phi}^T\bar{N}\right]_{x=b} \tag{3.28}$$

$$= \frac{1}{2}\int_a^b \alpha\left(\bar{\phi}^T\frac{d}{dx}\bar{N}\left[\frac{d}{dx}\bar{N}\right]^T\bar{\phi}\right) + \beta\left(\bar{\phi}^T\bar{N}\bar{N}^T\bar{\phi}\right)dx + \frac{1}{2}\gamma\left[\bar{\phi}^T\bar{N}\bar{N}^T\bar{\phi}\right]_{x=b} - \int_a^b \bar{\phi}^T\bar{N}f\,dx - \left[q\bar{\phi}^T\bar{N}\right]_{x=b}, \tag{3.29}$$

and since the vector $\bar{\phi}$ is independent of x and integration is a linear operator, we can write

$$F\left(\bar{\phi}\right) = \frac{1}{2}\bar{\phi}^T\underbrace{\left\{\int_a^b \left[\frac{d}{dx}\bar{N}\right]\alpha\left[\frac{d}{dx}\bar{N}\right]^T dx + \int_a^b \bar{N}\beta\bar{N}^T dx + \left[\bar{N}\gamma\bar{N}^T\right]_{x=b}\right\}}_{\underline{A}}\bar{\phi} - \bar{\phi}^T\underbrace{\left\{\int_a^b \bar{N}f\,dx - \left[q\bar{N}\right]_{x=b}\right\}}_{\bar{b}}, \tag{3.30}$$

where $\underline{A} \in \mathbb{C}^{M\times M}$ and $\bar{b} \in \mathbb{C}^{M\times 1}$. Here, the integration symbol is overloaded to mean integration entrywise. Remark that $\left[\frac{d}{dx}\bar{N}\right]\alpha\left[\frac{d}{dx}\bar{N}\right]^T = \alpha\left[\frac{d}{dx}\bar{N}\right]\left[\frac{d}{dx}\bar{N}\right]^T$ since α is a scalar

[3]Often, this choice is made such that the interval (a,b) is partitioned $a = x_0 < x_1 < \ldots < x_N = b$ into N subintervals with only a subset of the basis functions N_i having support over a given subinterval (x_{j-1}, x_j). This is in fact where the notion of element arises. When such a choice is made, we refer to the approach as the finite element method, otherwise, we call this the Ritz method.

function. We choose the less conventional representation $\left[\frac{d}{dx}\bar{N}\right]\alpha\left[\frac{d}{dx}\bar{N}\right]^T$ to emphasize that in two and three dimensions, when α is replaced by a matrix $\underline{\alpha}$ we may not exploit this commutativity unless $\underline{\alpha}$ possesses special structure. We also use the less conventional $\bar{N}f$ instead of $f\bar{N}$ anticipating further development. Note that in substituting ϕ, we have used both vector representations, which are equivalent, to reveal the quadratic structure of the functional in terms of $\bar{\phi}$, which can be written succinctly as

$$F(\bar{\phi}) = \frac{1}{2}\bar{\phi}^T\underline{A}\bar{\phi} - \bar{\phi}^T\bar{b}. \tag{3.31}$$

We seek the minimum of this functional, subject to the constraints that $\phi(a) = p$ and that ϕ is continuous. It is crucial that we require continuity of ϕ, otherwise the expression for $\underline{A}$ is incorrect. In particular, we would be missing terms of the form $\left[\phi\alpha\frac{d}{dx}\phi\right]_{x_d^+}^{x_d^-}$ wherever a discontinuity in α exists. To ensure that continuity is imposed, we enforce these constraints as a linear equality on $\bar{\phi}$ given by

$$\underline{C}\bar{\phi} = \bar{d}, \tag{3.32}$$

where the precise entries of $\underline{C} \in \mathbb{R}^{N_c \times M}$ and $\bar{d} \in \mathbb{C}^{N_c \times 1}$ depend on the basis functions chosen in $\bar{N}$ and N_c is the number of constraint equations[4].

Thus, to solve the Ritz problem, we must solve the optimization problem

$$\begin{aligned} \underset{\bar{\phi}}{\text{minimize}} \quad & \frac{1}{2}\bar{\phi}^T\underline{A}\bar{\phi} - \bar{\phi}^T\bar{b} \\ \text{subject to} \quad & \underline{C}\bar{\phi} = \bar{d}. \end{aligned} \tag{3.33}$$

We note that if $\underline{A}$ is symmetric non-negative definite, the optimization problem is convex, and we can solve the problem using the Karush-Kuhn-Tucker (KKT) conditions [88]. That is, we first form the Lagrangian

$$\mathcal{L}(\bar{\phi}, \bar{\nu}) = \frac{1}{2}\bar{\phi}^T\underline{A}\bar{\phi} - \bar{\phi}^T\bar{b} + \bar{\nu}^T(\underline{C}\bar{\phi} - \bar{d}), \tag{3.34}$$

where $\bar{\nu}$ is a vector of Lagrange multipliers, and take the gradient with respect to $\bar{\phi}$ and set it to zero to obtain the stationarity requirement

$$\nabla_{\bar{\phi}}\mathcal{L}(\bar{\phi}, \bar{\nu}) = \underline{A}\bar{\phi} - \bar{b} + \underline{C}^T\bar{\nu} = 0. \tag{3.35}$$

[4]In the finite element method, N_c corresponds to the number of Dirichlet boundary conditions (one in our BVP) plus the number of nodes in the partition, excluding the endpoints $(N-1)$ at which continuity must be imposed. Later, we will give an example to illustrate the construction of these matrices and vectors when using polynomial basis functions with disjoint support on subintervals (x_{j-1}, x_j).

Since the optimization problem does not have any inequality constraints, we need only satisfy this stationarity requirement subject to primal feasibility, which requires that $\underline{C}\bar{\phi} = \bar{d}$ (this is the same as taking the gradient of the Lagrangian with respect to $\bar{\nu}$ and setting it to zero). Thus, the minimum is achieved when both equations

$$\underline{A}\bar{\phi} + \underline{C}^T\bar{\nu} = \bar{b}, \tag{3.36}$$

$$\underline{C}\bar{\phi} = \bar{d}, \tag{3.37}$$

are satisfied. That is, the solution of the saddle point system

$$\underbrace{\begin{bmatrix} \underline{A} & \underline{C}^T \\ \underline{C} & 0 \end{bmatrix}}_{\underline{S}} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \bar{d} \end{bmatrix} \tag{3.38}$$

minimizes the functional $F(\bar{\phi})$ subject to the Dirichlet boundary condition $\phi(a) = p$ and the requirement that ϕ is continuous, both of which are encoded in the matrix $\underline{C}$ and vector $\bar{d}$. Note that if one were able to solve the saddle point system exactly, this would not mean that the original BVP would be solved exactly, but that the coefficient vector $\bar{\phi}$ would be chosen optimally for the given choice of basis functions in $\bar{N}$ (in the sense of minimizing the functional over the function space spanned by the basis functions chosen).

3.3 The Galerkin Method

Before we discuss conditions under which $\underline{S}$ is invertible, we return to the original BVP to discuss an alternative formulation yielding the same saddle point system. We do so to highlight the Lagrange multipliers $\bar{\nu}$ and to underscore their physical meaning. Let us take a general basis function N_i and weigh the original BVP by it, integrating over the domain Ω to obtain

$$-\int_a^b N_i \frac{d}{dx}\left(\alpha\frac{d}{dx}\phi\right) + N_i\beta\phi\,dx = \int_a^b N_i f\,dx. \tag{3.39}$$

Let us also assume that the basis function N_i possesses a discontinuity at $x = x_d$. Then, we cannot directly apply integration by parts to transfer the derivative from $\alpha\frac{d}{dx}\phi$ to N_i as the derivative of N_i does not exist at x_d. To avoid this, we break the integral into two disjoint

segments and then apply integration by parts to both:

$$-\int_a^{x_d^-} N_i \frac{d}{dx}\left(\alpha\frac{d}{dx}\phi\right)dx - \int_{x_d^+}^b N_i \frac{d}{dx}\left(\alpha\frac{d}{dx}\phi\right)dx + \int_a^b N_i\beta\phi\,dx = \int_a^b N_i f\,dx \tag{3.40}$$

$$\int_a^b \frac{d}{dx}N_i\alpha\frac{d}{dx}\phi\,dx - \left[N_i\alpha\frac{d}{dx}\phi\right]_a^{x_d^-} - \left[N_i\alpha\frac{d}{dx}\phi\right]_{x_d^+}^b + \int_a^b N_i\beta\phi\,dx = \int_a^b N_i f\,dx \tag{3.41}$$

$$\int_a^b \frac{d}{dx}N_i\alpha\frac{d}{dx}\phi + N_i\beta\phi\,dx - \left[N_i\alpha\frac{d}{dx}\phi\right]_a^b - \left[N_i\alpha\frac{d}{dx}\phi\right]_{x_d^+}^{x_d^-} = \int_a^b N_i f\,dx. \tag{3.42}$$

Next, we make use of the Robin boundary condition at $x = b$, as well as the fact that $\alpha\frac{d}{dx}\phi$ is continuous at $x = x_d$ to find that

$$\int_a^b \frac{d}{dx}N_i\alpha\frac{d}{dx}\phi + N_i\beta\phi\,dx + \left[N_i\gamma\phi\right]_{x=b} + \left[N_i\alpha\frac{d}{dx}\phi\right]_{x=a} - \left[N_i\right]_{x_d^+}^{x_d^-}\left[\alpha\frac{d}{dx}\phi\right]_{x=x_d} = \int_a^b N_i f\,dx - \left[N_i q\right]_{x=b}. \tag{3.43}$$

Now, let us define $\nu_1 = \left[\alpha\frac{d}{dx}\phi\right]_{x=a}$, $\nu_2 = \left[\alpha\frac{d}{dx}\phi\right]_{x=x_d}$, and make the approximation that $\phi = \bar{N}^T\bar{\phi}$. Substituting these three expressions yields

$$\underbrace{\int_a^b \frac{d}{dx}N_i\alpha\frac{d}{dx}\bar{N}^T\bar{\phi} + N_i\beta\bar{N}^T\bar{\phi}\,dx + \left[N_i\gamma\bar{N}^T\bar{\phi}\right]_{x=b}}_{\text{terms associated with } \underline{A}\bar{\phi}} + \underbrace{\left[N_i\right]_{x=a}\nu_1 - \left[N_i\right]_{x_d^+}^{x_d^-}\nu_2}_{\text{terms associated with } \underline{C}^T\bar{\nu}} = \underbrace{\int_a^b N_i f\,dx - \left[N_i q\right]_{x=b}}_{\text{terms associated with } \bar{b}} \tag{3.44}$$

which, we confirm later, corresponds to the ith equation in $\underline{A}\bar{\phi} + \underline{C}^T\bar{\nu} = \bar{b}$ with the appropriately defined constraint matrix $\underline{C}$. This means that the Lagrange multipliers in the saddle point system have specific meaning: they approximate the term $\alpha\frac{d}{dx}\phi$ wherever one terminates the region Ω by Dirichlet data and/or wherever one admits a discontinuity in basis functions. In the physics parlance, the Lagrange multipliers are associated with the flux described by our mathematical model (the BVP). For example, if we have α represent permittivity, $\beta = 0$, and f represent charge density, then the BVP has the physical interpretation of Gauss' law of electrostatics, and the flux term is precisely the electric flux density.

Before we move on, let us be clear about one thing. In deriving this physical meaning,

we used the fact that the flux is continuous even if α and/or N_i are discontinuous at $x = x_d$. This is true for the exact solution ϕ. When we solve the saddle point system, the flux of our computed solution can be discontinuous (there is nothing that enforces this continuity explicitly). However, we know from our variational study that continuity of flux is weakly enforced; this was a consequence found by taking the first variation of F and setting it to zero (see (3.21)). Roughly speaking, this means that as we better approximate ϕ, we get closer to a solution whose flux is continuous.

3.4 Solving the Saddle Point System

We now have two different useful approaches to determine the saddle point system (3.38). The first approach illustrates how one can view solving the BVP as an optimization problem, the second gives physical meaning to the Lagrange multipliers that arise in the first[5]. Regardless of approach, the final step in computing the approximation $\phi = \bar{N}^T \bar{\phi}$ is to solve (3.38). Here, we consider one important case where $\underline{S}$ is invertible and leave others to an excellent paper on the subject [114]. The special case is one where $\underline{A}$ is symmetric positive semidefinite. In this case, if $\underline{C}$ has full rank, and $\text{null}\,(\underline{A}) \cap \text{null}\,(\underline{C}) = \{0\}$, then $\underline{S}$ is invertible. To see why, let

$$\bar{u} = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} \tag{3.45}$$

be such that $\underline{S}\bar{u} = 0$. We will show that under these three conditions on $\underline{A}$, $\underline{C}$, and the intersection of their null spaces, $\bar{u}$ will be identically zero, meaning that the null space of $\underline{S}$ is the set containing only the zero vector, and that $\underline{S}$ is invertible. We begin by expanding $\underline{S}\bar{u} = 0$ to see that

$$\underline{A}\bar{x} + \underline{C}^T \bar{y} = 0, \tag{3.46}$$

$$\underline{C}\bar{x} = 0. \tag{3.47}$$

Multiplying the first equation by $\bar{x}^T$ gives

$$\bar{x}^T \underline{A}\bar{x} + \bar{x}^T \underline{C}^T \bar{y} = 0 \tag{3.48}$$

$$\bar{x}^T \underline{A}\bar{x} = -\left(\underline{C}\bar{x}\right)^T \bar{y} \tag{3.49}$$

$$= 0, \tag{3.50}$$

[5]The Galerkin method is more general than the Ritz method in that it can be applied to BVPs that do not have an associated functional. In such a case, the BVP may not exhibit the symmetry that we obtain in $\underline{A}$ after applying integration by parts.

where the last line comes from (3.47). Now, we know that $\underline{A}$ is symmetric positive semidefinite, which, together with $\bar{x}^T \underline{A} \bar{x} = 0$, implies that $\underline{A}\bar{x} = 0$. To see why this last statement is true, consider the function

$$p(t) = (\bar{x} + t\bar{y})^T \underline{A} (\bar{x} + t\bar{y}), \qquad t \in \mathbb{R}, \tag{3.51}$$

which is a quadratic function of the parameter t. We note that $p(t) \geq 0$ for all t since $\underline{A}$ is symmetric positive semidefinite. In addition, we note that $p(0) = \bar{x}^T \underline{A} \bar{x} = 0$. We now expand $p(t)$ to obtain

$$p(t) = \underbrace{\bar{x}^T \underline{A} \bar{x}}_{=0} + t\bar{x}^T \underline{A}\bar{y} + t\bar{y}^T \underline{A}\bar{x} + t^2 \bar{y}^T \bar{y} \tag{3.52}$$

$$= 2t\bar{y}^T \underline{A}\bar{x} + t^2 \bar{y}^T \bar{y} \tag{3.53}$$

$$= t\left(2\bar{y}^T \underline{A}\bar{x} + t\bar{y}^T \bar{y}\right). \tag{3.54}$$

We note that as $p(t)$ is quadratic in t, it has two zeros. However, since $p(t) \geq 0$ for all t and $p(0) = 0$, we also know that those two zeros must be repeated. Thus, when $t = 0$, we must have

$$\left[2\bar{y}^T \underline{A}\bar{x} + t\bar{y}^T \bar{y}\right]_{t=0} = 0 \tag{3.55}$$

$$\bar{y}^T \underline{A}\bar{x} = 0. \tag{3.56}$$

Now, since this last equation is true for all $\bar{y}$, we must have that $\underline{A}\bar{x} = 0$, as required. Thus, for our vector $\bar{u}$, we have that $\bar{x}$ satisfies both $\underline{A}\bar{x} = 0$ and $\underline{C}\bar{x} = 0$, which means that $\bar{x} \in \text{null}(\underline{A}) \cap \text{null}(\underline{C})$. But note that one of our assumptions was that $\text{null}(\underline{A}) \cap \text{null}(\underline{C}) = \{0\}$, and so $\bar{x} = 0$.

Now, we return to

$$\underbrace{\underline{A}\bar{x}}_{=0} + \underline{C}^T \bar{y} = 0 \tag{3.57}$$

to see that $\underline{C}^T \bar{y} = 0$. However, we know that $\underline{C}$ has full rank, and so $\bar{y} = 0$ as the columns of $\underline{C}^T$ are linearly independent. Finally, we see that $\bar{u} = 0$, and that $\underline{S}$ is invertible. Thus, as long as $\underline{C}$ has full rank (that is, we do not have constraints which are redundant, or which are implicitly satisfied given a set of other constraints) matrix $\underline{A}$ is symmetric positive semidefinite (this depends on the parameters α, β, and γ and is directly analogous to our discussion on whether the second variation of the original functional was positive), and the intersection of their null spaces is the zero vector, then the Ritz or Galerkin solution to the BVP is unique. Note that if $\underline{A}$ is symmetric positive definite, then $\text{null}(\underline{A}) = \{0\}$ and

$\text{null}\,(\underline{A}) \cap \text{null}\,(\underline{C}) = \{0\}$ so that, in this case, we only require $\underline{C}$ to have full rank. Also, if $\underline{A}$ is a general matrix, then we really only need to satisfy the condition $\text{null}\,(\underline{A}_S) \cap \text{null}\,(\underline{C}) = \{0\}$ where $\underline{A}_S = \frac{1}{2}(\underline{A} + \underline{A}^T)$ is the symmetric part of $\underline{A}$ [114].

Let us assume that our saddle point system satisfies these three conditions. Then a common approach to solving the saddle point system is to use a null space method. That is, first, we find a particular solution $\bar{\phi}_z$ satisfying the equality constraints

$$\underline{C}\bar{\phi}_z = \bar{d}. \tag{3.58}$$

This can be done by satisfying the Dirichlet boundary conditions first and working through the continuity conditions afterward. Second, we find a basis for the null space of $\underline{C}$ such that $\underline{C}\underline{Z} = 0$ with $\underline{Z} \in \mathbb{R}^{M \times (M - N_c)}$. If we let

$$\bar{\phi} = \underline{Z}\bar{\phi}_p + \bar{\phi}_z, \tag{3.59}$$

where $\bar{\phi}_p \in \mathbb{C}^{(M-N_c)\times 1}$, then

$$\underline{C}\bar{\phi} = \underbrace{\underline{C}\underline{Z}}_{=0}\bar{\phi}_p + \underbrace{\underline{C}\bar{\phi}_z}_{=\bar{d}} \tag{3.60}$$

$$= \bar{d}. \tag{3.61}$$

Thus, this choice of $\bar{\phi}$ satisfies the equality constraints. In addition, if we substitute this expression into

$$\underline{A}\bar{\phi} + \underline{C}^T\bar{\nu} = \bar{b} \tag{3.62}$$

$$\underline{A}(\underline{Z}\bar{\phi}_p + \bar{\phi}_z) + \underline{C}^T\bar{\nu} = \bar{b} \tag{3.63}$$

and multiply through by $\underline{Z}^T$, we obtain

$$\underline{Z}^T\underline{A}(\underline{Z}\bar{\phi}_p + \bar{\phi}_z) + \underbrace{\underline{Z}^T\underline{C}^T}_{=0}\bar{\nu} = \underline{Z}^T\bar{b} \tag{3.64}$$

$$\underline{Z}^T\underline{A}\underline{Z}\bar{\phi}_p = \underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z), \tag{3.65}$$

which can be solved for $\bar{\phi}_p$ using any method suitable for solving symmetric systems of equations. Once we have computed $\bar{\phi}_p$, then the solution to our Ritz or Galerkin problem is given by $\bar{\phi} = \underline{Z}\bar{\phi}_p + \bar{\phi}_z$.

The key step which remains to be addressed is how one computes $\underline{Z}$. One common way to compute a basis for the null space of a general rectangular matrix is to first compute its

singular value decomposition (SVD). In our case, since $\underline{C} \in \mathbb{R}^{N_c \times M}$ with $M > N_c$ and with $\text{rank}\,(\underline{C}) = N_c$, the SVD of $\underline{C}$ is given by

$$\underline{C} = \underline{U} \underbrace{\begin{bmatrix} \underline{\Sigma}_1 & 0 \end{bmatrix}}_{\underline{\Sigma}} \underbrace{\begin{bmatrix} \underline{V}_1^T \\ \underline{V}_2^T \end{bmatrix}}_{\underline{V}^T}, \tag{3.66}$$

where $\underline{U} \in \mathbb{R}^{N_c \times N_c}$ and is orthogonal, $\underline{\Sigma} \in \mathbb{R}^{N_c \times M}$, $\underline{V} \in \mathbb{R}^{M \times M}$ and is orthogonal, $\underline{\Sigma}_1 \in \mathbb{R}^{N_c \times N_c}$ and is diagonal and invertible, $\underline{V}_1 \in \mathbb{R}^{M \times N_c}$, and $\underline{V}_2 \in \mathbb{R}^{M \times (M-N_c)}$ [25]. Now, if we seek a basis for the null space of $\underline{C}$, we must look for vectors $\bar{x} \in \mathbb{R}^M$ which satisfy

$$\underline{C}\bar{x} = 0 \tag{3.67}$$
$$\underline{U}\underline{\Sigma}_1\underline{V}_1^T\bar{x} = 0 \tag{3.68}$$
$$\underline{V}_1^T\bar{x} = (\underline{\Sigma}_1^{-1}\underline{U}^T)0 \tag{3.69}$$
$$\underline{V}_1^T\bar{x} = 0. \tag{3.70}$$

Since $\underline{V}$ is orthogonal, $\underline{V}^T\underline{V} = \underline{I}$, and thus

$$\begin{bmatrix} \underline{V}_1^T \\ \underline{V}_2^T \end{bmatrix} \begin{bmatrix} \underline{V}_1 & \underline{V}_2 \end{bmatrix} = \begin{bmatrix} \underline{I} & 0 \\ 0 & \underline{I} \end{bmatrix} \tag{3.71}$$
$$\begin{bmatrix} \underline{V}_1^T\underline{V}_1 & \underline{V}_1^T\underline{V}_2 \\ \underline{V}_2^T\underline{V}_1 & \underline{V}_2^T\underline{V}_2 \end{bmatrix} = \begin{bmatrix} \underline{I} & 0 \\ 0 & \underline{I} \end{bmatrix}, \tag{3.72}$$

which gives that $\underline{V}_1^T\underline{V}_2 = 0$. Therefore, if we take an arbitrary vector $\bar{z} \in \mathbb{R}^{(M-N_c)\times 1}$, we can write $\bar{x} = \underline{V}_2\bar{z}$. That is, all vectors $\bar{x}$ in the null space of matrix $\underline{C}$ can be written as a linear combination of the columns of $\underline{V}_2$, which means that $\underline{V}_2$ is a basis for the null space of $\underline{C}$, and we can take $\underline{Z} = \underline{V}_2$.

Unfortunately, this completely general and systematic approach is rarely used because, even if $\underline{C}$ is a sparse matrix (which is often the case for the constraint matrix), the resulting orthogonal matrix V from the SVD need not be sparse (in general it is full). This is typically considered a drawback of the SVD approach because if the number of basis functions which share the same support on some subinterval of (a, b) is small compared to the total number of basis functions M, then the matrix $\underline{A}$ will be sparse, and this sparsity will be lost when multiplying by full $\underline{Z}$. However, if both $\underline{Z}$ and $\underline{A}$ are sparse, then the product $\underline{Z}^T\underline{A}\underline{Z}$ will also be sparse, and we can use a solver which takes advantage of this fact (for example, a sparsity-aware direct factorization, or an iterative algorithm exploiting sparse matrix-vector products). In fact, the finite element method is a Ritz or Galerkin method where the choice

of basis functions is made precisely in this fashion, leading to a sparse $\underline{A}$, and where it is straightforward to compute a sparse basis for the null space $\underline{Z}$. In the following, we discuss the construction of one such basis and explain the ubiquity of nodal finite elements through observations made on the resulting constraint matrices.

3.5 The Classical Finite Element Method

As alluded to previously, we begin by subdividing the domain $\Omega = (a, b)$ into N disjoint subintervals (x_{j-1}, x_j) using the partition $a = x_0 < x_1 < x_2 < ... < x_N = b$. Over each subinterval, we construct an interpolatory polynomial basis of degree L. Rather than treat each basis separately, we focus on constructing a single basis on the canonical interval $u \in (-1, 1)$. We use this interval because there is an invertible, affine map from the canonical interval to any other subinterval (x_{j-1}, x_j) using the transformations

$$x = \frac{(x_j + x_{j-1}) + (x_j - x_{j-1})\, u}{2}, \tag{3.73}$$

$$u = \frac{(x - x_{j-1}) - (x_j - x)}{x_j - x_{j-1}}. \tag{3.74}$$

Both of these transformations are numerically stable to compute [108]. In addition, since the map is affine, polynomials on the canonical interval are mapped to polynomials on the subinterval (x_{j-1}, x_j), preserving their approximation properties. Most standard texts on the subject proceed to use Lagrange interpolation to define the interpolatory polynomial basis of degree L on the canonical interval. The basis functions then take the form

$$l_i(u) = \prod_{\substack{j=0 \\ i \neq j}}^{L} \frac{(u - u_j)}{(u_i - u_j)} \tag{3.75}$$

where u_j denotes an interpolation node typically lying somewhere inside or on the boundary of $(-1, 1)$ (although this is not strictly necessary). For the interpolatory polynomials to be well defined, we require that the u_j be distinct. A standard choice in the finite element literature is for the nodes to be equally spaced with $u_j = -1 + \frac{2}{L} j$ and $j = 0, 1, 2, ..., L$ [12, 28].

There are several problems associated with this choice of basis function. First, interpolation using equally spaced nodes is ill-conditioned, as discussed in Chapter 2. To circumvent this fact, many practitioners of the classical finite element method take L small (say $L \leq 4$). Since our goal is to describe a fast and highly accurate solver, we use basis functions that are

well-conditioned for large L, in which case interpolation using Chebyshev nodes, for example, is preferred [109]. Second, the use of (3.75) in computation is inefficient and the barycentric form of the Lagrange interpolant is superior, particularly for large L [115].

Let us now discuss the constraint matrix $\underline{C}$ giving rise to $\underline{Z}$. We will show that the advantage of using interpolatory polynomials for our basis functions on disjoint subintervals arises when considering the sparsity of the null space matrix $\underline{Z}$ and the operator matrix $\underline{A}$. Consider the partition $a = x_0 < x_1 < x_2 < \ldots < x_N = b$ of the interval (a, b). If we map the subinterval (x_{j-1}, x_j) to $(-1, 1)$, then we can use our Lagrange interpolation construction to build basis functions for a degree L polynomial basis with support on (x_{j-1}, x_j). Thus, the global basis functions will be

$$N_i(x) = \begin{cases} l_k\left(\dfrac{(x - x_{j-1}) - (x_j - x)}{x_j - x_{j-1}}\right) & x \in (x_{j-1}, x_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.76}$$

with $i = (L+1)(j-1) + (k+1)$, $k = 0, 1, 2, ...L$, and $j = 1, 2, 3, ..., N$, for a total of $M = N(L+1)$ basis functions. This definition of global basis functions may be unsettling as certain basis functions are discontinuous at the points x_j and, furthermore, a solution ϕ constructed from a linear combination of such locally defined functions leaves the solution at the nodes x_j unspecified. We address such issues by imposing inter-element continuity. Note that, for a given basis function on subinterval (x_{j-1}, x_j), regardless of whether we use equidistant nodes or the Chebyshev nodes, we will always have one interpolation node at x_{j-1} and another at x_j (the remaining $L - 1$ nodes will be somewhere between these two). Similarly, for the subinterval (x_j, x_{j+1}), we will have interpolation nodes at x_j and at x_{j+1}, and so on. Because we have defined the basis functions locally, and because we have interpolatory basis functions, when we construct our constraint matrix $\underline{C}$ to impose continuity conditions, we recover a sparse matrix. This sparsity arises because equating $\phi(x_j^-) = \phi(x_j^+)$ reduces the equation

$$\sum_{i=1}^{M} \phi_i N_i(x_j^-) = \sum_{i=1}^{M} \phi_i N_i(x_j^+) \tag{3.77}$$

to one where only basis functions on neighboring elements to the point x_j are active:

$$\sum_{i=0}^{L} \phi_i^- N_i^-(x_j^-) = \sum_{i=0}^{L} \phi_i^+ N_i^+(x_j^+), \tag{3.78}$$

where positive superscripts indicate basis functions N_i and coefficients ϕ_i belonging to the

subinterval (x_j, x_{j+1}) and negative superscripts indicate the same quantities but for the subinterval (x_{j-1}, x_j). Since the basis functions are interpolatory, only $N_{L+1}^-(x_j^-) = 1$, whereas all other basis functions evaluated at x_j^- are zero, and $N_1^+(x_j^+) = 1$, whereas all other basis functions evaluated at x_j^+ are zero. This means that continuity can be enforced as

$$\phi_{L+1}^- = \phi_1^+ \tag{3.79}$$

$$\phi_{L+1}^- - \phi_1^+ = 0. \tag{3.80}$$

Thus, each continuity constraint will have a total of two nonzero entries in a row of $N(L+1)$ entries. This is sparse. In addition, since we have imposed inter-element continuity in a limiting sense, the right and left sided limits at each x_j will agree, and we can assign each common limiting value to ϕ at each point x_j. In doing so, we find an approximation to ϕ that is continuous.

For example, to keep things simple, consider the case with $L = 2$. Then, ignoring Dirichlet boundary conditions, we can represent continuity constraints using

$$\underline{C}_c = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 & 0 & & & & & & & & & \\ & & & 0 & 0 & 1 & -1 & 0 & 0 & & & & & & \\ & & & & & & 0 & 0 & 1 & -1 & 0 & 0 & & & \\ & & & & & & & & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & & & & & & & & & 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix}, \tag{3.81}$$

where each row corresponds to enforcing a continuity condition between two adjacent subintervals (x_{j-1}, x_j) and (x_j, x_{j+1}) (note that in our example, there will be $N-1$ such constraints). Thus, our continuity conditions can be written as

$$\underline{C}_c \bar{\phi} = 0, \tag{3.82}$$

which is in the form required for our saddle point system. We can add the Dirichlet boundary condition $\phi(a) = p$ by appending a row to $\underline{C}_c$. Again, since our basis is interpolatory, this can be done by enforcing $\phi_1 = p$. In other words, we can write

$$\underline{C} = \begin{bmatrix} \underline{C}_c \\ \bar{e}_1^T \end{bmatrix}, \tag{3.83}$$

where $\bar{e}_1^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$, and we can define $\bar{d} = p\bar{e}_N$, where $\bar{e}_N^T = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$ with

N entries to get

$$\underline{C}\bar{\phi} = \bar{d}. \tag{3.84}$$

We now look for a basis for the null space of this particular $\underline{C}$. We see that for the first $N-1$ columns we can take $|\underline{C}_c|^T$ where the absolute value is to be applied entrywise. Since we require $\underline{C}$ to have full rank, and $\underline{C} \in \mathbb{R}^{N \times N(L+1)}$, then by the rank-nullity theorem, we know that the dimension of the null space is NL. Since $\underline{C}_c$ has $N-1$ rows, we have already found $N-1$ columns of a basis for the null space. The $N(L-1)+1$ remaining basis vectors can be chosen as the unit vectors with a nonzero entry in any row corresponding to the column number containing all zeros in $\underline{C}$. For our simple example, we obtain

$$\underline{Z} = \begin{bmatrix}
0 & & & & & & & & & & \\
0 & & & & 1 & & & & & & \\
1 & & & & & & & & & & \\
1 & & & & & & & & & & \\
0 & 0 & & & & 1 & & & & & \\
 & 1 & & & & & & & & & \\
 & 1 & & & & & & & & & \\
 & 0 & 0 & & & & 1 & & & & \\
 & & 1 & \ddots & & & & & & & \\
 & & 1 & \ddots & & & & & & & \\
 & & 0 & \ddots & 0 & & & 1 & & & \\
 & & & \ddots & 1 & & & & & & \\
 & & & & 1 & & & & \ddots & & \\
 & & & & 0 & & & & & 1 & \\
 & & & & 0 & & & & & & 1
\end{bmatrix}, \tag{3.85}$$

which is a sparse basis for the null space (one can verify this fact by computing $\underline{C}\underline{Z}$ and observing that the result is identically zero). While this process may seem tedious, if we rearrange the columns of $\underline{Z}$, we obtain a very natural and simple way to construct this basis. Consider the vector

$$\bar{\phi}_{\text{global}} = \begin{bmatrix} \phi_{g,1} \\ \phi_{g,2} \\ \vdots \\ \phi_{g,NL+1} \end{bmatrix}, \tag{3.86}$$

where $\bar{\phi}_{\text{global}}$ lists the value of ϕ at each interpolation node from left to right, making sure not to double count at the locations x_j for $j = 1, ..., N-1$. Rearranging $\underline{Z}$ such that

$$\underline{Z} = \begin{bmatrix} 0 & 0 & & & & & & & & & \\ 1 & 0 & & & & & & & & & \\ & 1 & & & & & & & & & \\ & 1 & & & & & & & & & \\ & 0 & 1 & 0 & & & & & & & \\ & & & 1 & & & & & & & \\ & & & 1 & & & & & & & \\ & & & 0 & 1 & 0 & & & & & \\ & & & & & 1 & & & & & \\ & & & & & 1 & & & & & \\ & & & & & 0 & 1 & \ddots & & & \\ & & & & & & & \ddots & 1 & 0 & \\ & & & & & & & & & 1 & \\ & & & & & & & & & 1 & \\ & & & & & & & & & 0 & 1 \\ & & & & & & & & & 0 & 0 & 1 \end{bmatrix}, \tag{3.87}$$

we see that $\bar{\phi} = \begin{bmatrix} \bar{e}_1 & \underline{Z} \end{bmatrix} \bar{\phi}_{\text{global}}$. Thus, $\underline{Z}$ is simply a global to local node identification matrix with columns removed corresponding to nodes set by Dirichlet boundary conditions. Perhaps the most important aspect of this observation is that there is no need to explicitly solve for a sparse basis for the null space. This means that we can efficiently solve the finite element problem without forming $\underline{Z}$ as long as we know how global nodes are related to local nodes. This is usually done using what is called a connectivity array. The key fact is that the matrix $\underline{Z}^T \underline{A} \underline{Z}$ can be directly constructed from $\underline{A}$ and the connectivity array without performing matrix products. This process is called global assembly. For one possible description of global assembly, see [12]. The same fact applies to the construction of $\underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z)$. In particular, for our choice of interpolatory basis functions, $\bar{\phi}_z = p\bar{e}_1$. Once we have solved for $\bar{\phi}_p$, it is straightforward to compute $\bar{\phi} = \underline{Z}\bar{\phi}_p + \bar{\phi}_z$ without performing the matrix-vector product $\underline{Z}\bar{\phi}_p$ (again, the matrix-vector product is encoded in the connectivity array).

Another important observation is that solving

$$\underline{Z}^T \underline{A} \underline{Z} \bar{\phi}_p = \underline{Z}^T (\bar{b} - \underline{A}\bar{\phi}_z) \tag{3.88}$$

encodes the elimination of nodes related to Dirichlet data. In several texts, including [12], the recommended approach is to replace the row corresponding to a node enforcing a Dirichlet boundary condition by a zero row with a 1 on the main diagonal. This operation undoes the symmetry of $\underline{A}$. Thus, to restore symmetry (which is desirable), the column corresponding to the node is brought to the right hand side of the equations and both the row and column are removed from the modified $\underline{A}$. This is precisely what the reduced system of equations for $\bar{\phi}_p$ accomplishes. Since $\underline{Z}$ lacks the unit vectors corresponding to nodes where Dirichlet conditions are imposed, the multiplication of $\underline{Z}^T\underline{A}\underline{Z}$ removes those rows and columns. In addition, the data in the removed columns are moved to the right hand side via the term $-\underline{Z}^T\underline{A}\bar{\phi}_z$. In fact, this tells us how $\bar{\phi}_{\text{global}}$ and $\bar{\phi}_p$ are related: $\bar{\phi}_p$ is $\bar{\phi}_{\text{global}}$ with Dirichlet nodes removed.

Finally, we discuss the construction of $\underline{A}$ and its sparsity. Using our interpolatory basis functions, we note that

$$\underline{A} = \begin{bmatrix} \underline{A}_1 & & & \\ & \underline{A}_2 & & \\ & & \ddots & \\ & & & \underline{A}_N \end{bmatrix}, \tag{3.89}$$

where $\underline{A}_k \in \mathbb{C}^{(L+1)\times(L+1)}$. The matrix $\underline{A}$ has this block diagonal structure because its entries are given by

$$\underline{A} = \int_a^b \left[\frac{d}{dx}\bar{N}\right] \alpha \left[\frac{d}{dx}\bar{N}\right]^T dx + \int_a^b \bar{N}\beta\bar{N}^T dx + \left[\bar{N}\gamma\bar{N}^T\right]_{x=b}. \tag{3.90}$$

It is clear that if a basis function in $\bar{N}$ or $\frac{d}{dx}\bar{N}$ does not share common support with another, that entry of $\underline{A}$ will be zero (integrating against the zero function returns zero). Of course, this block structure only arises because we have grouped basis functions with shared support over the same subinterval (this is standard practice). This result can be viewed as an approximate orthogonality; by defining basis functions on disjoint subintervals, we ensure that the majority of inner products in $\underline{A}$ evaluate to zero, guaranteeing sparsity. We note that if we use either equally spaced or Chebyshev nodal interpolation for our local basis functions, then each block matrix $\underline{A}_k$ is full, regardless of how $\alpha(x)$ and $\beta(x)$ vary over the element. This is not a problem when the local degree on each element L is small with respect to the total number of unknowns $N(L+1)$. When α and β are constants over each element, then analytical expressions for $\underline{A}_k$ can be used. Otherwise, it is standard to apply a Gaussian quadrature rule to evaluate the integrals [28].

To conclude, we note that in the classical finite element method, the $\underline{A}_k$ matrices are

referred to as local element matrices, which are then assembled into a global finite element matrix via the multiplication $\underline{Z}^T \underline{A} \underline{Z}$ (again, this process is usually hidden by use of the connectivity array). In our example, the global system is a banded matrix with bandwidth $L+1$ due to the sequential numbering of elements from left to right.

3.6 Adaptive Finite Element Considerations

In the preceding section, we assumed that the polynomial degree L for each interpolatory basis was uniform. Thus, to produce more accurate solutions to the finite element problem, we have two choices: we may either further subdivide the interval (a, b) by using more elements, or we may increase the polynomial degree L over all elements. We call element refinement h-adaptivity and increasing local element degree p-adaptivity. p-adaptivity is well suited to situations where the solution ϕ to the original BVP is smooth over the element. This is intuitive as polynomials are infinitely differentiable; it makes little sense to expect to represent a function with only a small number of derivatives over the subinterval (x_j, x_{j+1}) by one comprised of a finite sum of smooth functions (although in the limit this is possible, see [109] for example). When the solution ϕ is not smooth, then h-adaptivity is preferred. Any finite element code that allows for both local h-adaptivity and local p-adaptivity is deemed an hp-adaptive code. These codes differ in the way that elements are chosen for h-adaption or p-adaption.

How then can our description of the finite element method be modified to allow for hp-adaption? First, each element must be assigned a local polynomial degree L_k. To allow for arbitrary L_k, it becomes imperative that we use the Chebyshev nodes[6] rather than the equally spaced nodes since interpolation in equally spaced nodes need not converge as L_k grows [109]. If we make this choice, then the block diagonal matrix $\underline{A}$ will now have blocks of size $(L_k+1) \times (L_k+1)$. One drawback associated with the nodal finite element basis chosen is that these blocks must be recomputed if the local degree is changed. In addition, suppose that our adaption algorithm concludes that p-adaption is preferred for all elements, then the local element matrices become large compared to the total size of $\underline{A}$, and the sparsity of the finite element method is lost. In the following chapter, we discuss a choice of basis which circumvents these drawbacks.

[6] In principle, any set of nodes that cluster near the endpoints of the canonical interval with the same asymptotic node density [116], and that includes the endpoints (for sparse inter-element continuity) is sufficient, although the Chebyshev set described here is the simplest I know of.

Chapter 4

A Fast Adaptive Finite Element Method

This chapter combines ideas from Chapters 2 and 3 to describe a fast, adaptive finite element method. In particular, the chapter uses the strong approximation properties for smooth functions that orthonormal Legendre polynomials and polynomial interpolation at Chebyshev nodes have (as described in Chapter 2) to modify the classical finite element method (as described in Chapter 3). The presentation is directed in such a way as to naturally arrive at the basis functions of Babuška-Szabó [48] and Shen [49] and to illustrate the desirable properties these functions possess, including sparsity and conditioning of the associated local element matrices. These results hold under simplified assumptions on material parameters. The chapter goes on to show how to extend these results to spatially varying parameters using Legendre expansions. Efficient computation of these expansions is described using the algorithm of [51]. The chapter concludes with a description of the finite element method itself, and its application in an *hp*-adaptive framework.

4.1 Sparsity of the Operator Matrix

We begin by making two simplifying assumptions. First, let α and β be scalar constants. We will return to the case where α and β vary arbitrarily in space later; this requires some additional insight. Second, assume $\Omega = (-1, 1)$ and use a single element to span the interval. Again, we will return to the case with several elements later, but the generalization is relatively straightforward. With these assumptions, the BVP of interest becomes

$$-\frac{d}{dx}\left(\alpha\frac{d\phi}{dx}\right)+\beta\phi\left(x\right)=f\left(x\right),\qquad x\in\left(-1,1\right), \tag{4.1}$$

$$\phi\left(-1\right)=p, \tag{4.2}$$

$$\left[\alpha\frac{d\phi}{dx}+\gamma\phi\right]_{x=1}=q, \tag{4.3}$$

and the corresponding functional is

$$F\left(\phi\right)=\frac{1}{2}\left\{\alpha\int_{-1}^{1}\left(\frac{d\phi}{dx}\right)^{2}dx+\beta\int_{-1}^{1}\phi^{2}dx+\gamma\left[\phi(1)\right]^{2}\right\}-\left\{\int_{-1}^{1}f\phi\,dx+q\phi(1)\right\}. \tag{4.4}$$

Recall (see (3.31), for example) that by approximating ϕ by a linear combination of basis functions, we obtain the quadratic form

$$F=\frac{1}{2}\bar{\phi}^{T}\underline{A}\bar{\phi}-\bar{\phi}^{T}\bar{b} \tag{4.5}$$

with

$$\underline{A}=\alpha\int_{-1}^{1}\left[\frac{d}{dx}\bar{N}\right]\left[\frac{d}{dx}\bar{N}\right]^{T}dx+\beta\int_{-1}^{1}\bar{N}\bar{N}^{T}dx+\gamma\left[\bar{N}\bar{N}^{T}\right]_{x=1}. \tag{4.6}$$

Let us treat each term in (4.6) separately, commenting on the sparsity of the resulting matrices.

If we approximate ϕ using interpolatory Lagrange polynomials, then $\bar{N}\left(x\right)=\bar{l}\left(x\right)$ where $\bar{l}$ is the vector containing Lagrange polynomials. Recall (see (2.32) and its associated paragraph) that we can change basis to the orthonormal Legendre polynomials using $\bar{l}\left(x\right)=\underline{\tilde{V}}^{-T}\bar{p}\left(x\right)$ where $\underline{\tilde{V}}$ is the generalized Vandermonde matrix and $\bar{p}$ is the vector containing the orthonormal Legendre polynomials. For the first term in (4.6), we obtain

$$\alpha\int_{-1}^{1}\left[\frac{d}{dx}\bar{N}\right]\left[\frac{d}{dx}\bar{N}\right]^{T}dx=\alpha\int_{-1}^{1}\left[\frac{d}{dx}\bar{l}\right]\left[\frac{d}{dx}\bar{l}\right]^{T}dx \tag{4.7}$$

$$=\alpha\int_{-1}^{1}\left[\frac{d}{dx}\underline{\tilde{V}}^{-T}\bar{p}\right]\left[\frac{d}{dx}\underline{\tilde{V}}^{-T}\bar{p}\right]^{T}dx. \tag{4.8}$$

Next, we use the linearity of differentiation to interchange $\underline{\tilde{V}}^{-T}$ and $\frac{d}{dx}$ and replace differentiation of $\bar{p}$ with multiplication by the appropriate differentiation matrix $\underline{\tilde{D}}$ for orthonormal

Legendre polynomials (defined in (2.95)). This gives

$$\alpha \int_{-1}^{1} \left[\frac{d}{dx}\bar{N}\right]\left[\frac{d}{dx}\bar{N}\right]^T dx = \alpha \int_{-1}^{1} \left[\underline{\tilde{V}}^{-T}\frac{d}{dx}\bar{p}\right]\left[\underline{\tilde{V}}^{-T}\frac{d}{dx}\bar{p}\right]^T dx \tag{4.9}$$

$$= \alpha \int_{-1}^{1} \left[\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\bar{p}\right]\left[\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\bar{p}\right]^T dx. \tag{4.10}$$

Finally, we apply the matrix transposition rule and use the linearity of integration to integrate only the outer product of $\bar{p}$; this procedure yields

$$\alpha \int_{-1}^{1} \left[\frac{d}{dx}\bar{N}\right]\left[\frac{d}{dx}\bar{N}\right]^T dx = \alpha \int_{-1}^{1} \left[\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\bar{p}\right]\left[\bar{p}^T\underline{\tilde{D}}^T\underline{\tilde{V}}^{-1}\right] dx \tag{4.11}$$

$$= \alpha\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\underbrace{\int_{-1}^{1}\bar{p}\bar{p}^T dx}_{\underline{I}_{L+1}}\underline{\tilde{D}}^T\underline{\tilde{V}}^{-1} \tag{4.12}$$

$$= \alpha\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\underline{\tilde{D}}^T\underline{\tilde{V}}^{-1} \tag{4.13}$$

where the last step comes from the orthogonality of the Legendre polynomials over the interval $(-1, 1)$. Similarly, we can repeat the same procedure on the second term in (4.6) to obtain

$$\beta \int_{-1}^{1} \bar{N}\bar{N}^T dx = \beta \int_{-1}^{1} \bar{l}\,\bar{l}^T dx \tag{4.14}$$

$$= \beta \int_{-1}^{1} \left[\underline{\tilde{V}}^{-T}\bar{p}\right]\left[\underline{\tilde{V}}^{-T}\bar{p}\right]^T dx \tag{4.15}$$

$$= \beta \int_{-1}^{1} \left[\underline{\tilde{V}}^{-T}\bar{p}\right]\left[\bar{p}^T\underline{\tilde{V}}^{-1}\right] dx \tag{4.16}$$

$$= \beta\underline{\tilde{V}}^{-T}\underbrace{\int_{-1}^{1}\bar{p}\bar{p}^T dx}_{\underline{I}_{L+1}}\underline{\tilde{V}}^{-1} \tag{4.17}$$

$$= \beta\underline{\tilde{V}}^{-T}\underline{\tilde{V}}^{-1}. \tag{4.18}$$

Finally, the third term in (4.6) is straightforward, as long as an interpolation node exists at $x = 1$. In both the equally spaced and Chebyshev node cases, such a node exists, and we conclude that all but one entry in $\bar{N}$ will be zero, the exception being one. To keep things simple, let us say that the ordering of Lagrange polynomials in $\bar{N}$ is such that $\bar{N}(1) = \bar{e}_{L+1}$ for the chosen basis of polynomial degree L. Putting these three computations together yields

$$\underline{A} = \alpha\underline{\tilde{V}}^{-T}\underline{\tilde{D}}\underline{\tilde{D}}^T\underline{\tilde{V}}^{-1} + \beta\underline{\tilde{V}}^{-T}\underline{\tilde{V}}^{-1} + \gamma\bar{e}_{L+1}\bar{e}_{L+1}^T. \tag{4.19}$$

Thus, if we have α and β constant, then computing these matrices does not require integration. In fact, we can clearly see that the quality of the finite element solution depends not only on the polynomial order, but also where the interpolation nodes lie. This is because $\underline{A}$ depends on the inversion of the generalized Vandermonde matrix which we have shown, in Chapter 2, to be poorly conditioned for large L on equally spaced nodes. If we take the Chebyshev nodes, this conditioning problem is significantly reduced, and we are free to take L large. That being said, the matrix $\underline{A}$ is full since $\underline{\tilde{V}}$ and its inverse are both full matrices and we have lost one of the defining properties of a classical finite element method, the sparsity of $\underline{A}$. This means that, using such a basis, we can construct a highly accurate solver, but it will not be fast since solving the saddle point system will cost $\mathcal{O}\left(L^3\right)$ FLOPs using LU factorization.

How then can we rid ourselves of the generalized Vandermonde matrix appearing in $\underline{A}$? Consider no longer using an interpolatory basis. The previous discussion points us to one potentially desirable basis, the orthonormal Legendre polynomials. That is, take $\bar{N} = \bar{p}$ and perform the same computations to obtain

$$\underline{A} = \alpha \underline{\tilde{D}}\,\underline{\tilde{D}}^T + \beta \underline{I}_{L+1} + \gamma \bar{p}\left(1\right)\bar{p}\left(1\right)^T. \tag{4.20}$$

In doing so, we have successfully removed the Vandermonde matrix, but we are presented with two new problems. First, the product $\underline{\tilde{D}}\,\underline{\tilde{D}}^T$ is roughly half full. In fact, the sparsity has a checkerboard structure with nonzeros if $i > 1$ and $j > 1$ and $\mathrm{mod}\left(i + j, 2\right) = 0$. In addition, and more importantly, the outer product $\bar{p}\left(1\right)\bar{p}\left(1\right)^T$ is full. This is also a concern as the vector $\bar{p}\left(-1\right)$ arises when imposing the Dirichlet boundary condition and, should we generalize to multiple elements, both $\bar{p}\left(1\right)$ and $\bar{p}\left(-1\right)$ arise when imposing inter-element continuity conditions. In such a situation, it becomes unclear how to choose a sparse basis for the null space for the constraint matrix.

To circumvent the first problem, let us attempt to eliminate the matrix $\underline{\tilde{D}}\,\underline{\tilde{D}}^T$ by making one further modification to our orthonormal Legendre basis. We saw that by taking $\bar{N} = \bar{p}$, the *second* term in $\underline{A}$ simplified to the identity matrix. Thus one natural question which arises is whether a similar approach—choosing $\bar{N}$ such that the *first* term in $\underline{A}$ simplifies to something close to the identity matrix—can be used (hoping, of course, that this transformation does not adversely affect the remaining terms in $\underline{A}$). To find such a transformation, we remark that the reason the second term evaluated to the identity matrix was because of the orthonormality of the Legendre polynomials on the interval $(-1, 1)$. Thus, if we want the first term to evaluate to identity, we must look for functions whose derivatives are orthonormal on the same interval. That is, we must choose functions that are the indefinite

integrals (or antiderivatives) of the orthonormal Legendre polynomials.

To express such functions, we return to the property of orthonormal Legendre polynomials that we used to derive the differentiation matrix $\underline{\tilde{D}}$ in Section 2.2. The property is

$$\sqrt{2n+1}p_n = \frac{d}{dx}\left[-\frac{1}{\sqrt{2\left(n-1\right)+1}}p_{n-1} + \frac{1}{\sqrt{2\left(n+1\right)+1}}p_{n+1}\right]. \tag{4.21}$$

Integrating both sides and simplifying radicals yields

$$\int p_n\,dx = -\frac{1}{\sqrt{2n+1}\sqrt{2\left(n-1\right)+1}}p_{n-1} + \frac{1}{\sqrt{2n+1}\sqrt{2\left(n+1\right)+1}}p_{n+1} \tag{4.22}$$

$$= -\frac{1}{\sqrt{4n^2-1}}p_{n-1} + \frac{1}{\sqrt{4\left(n+1\right)^2-1}}p_{n+1} \tag{4.23}$$

which we can use to build an integration matrix. For example, consider the case with n ranging from 0 to 3 which gives

$$\begin{bmatrix} \int p_0\,dx \\ \int p_1\,dx \\ \int p_2\,dx \\ \int p_3\,dx \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \frac{1}{\sqrt{3}} & & & \\ -\frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}}\frac{1}{\sqrt{5}} & & \\ & -\frac{1}{\sqrt{3}}\frac{1}{\sqrt{5}} & 0 & \frac{1}{\sqrt{5}}\frac{1}{\sqrt{7}} & \\ & & -\frac{1}{\sqrt{5}}\frac{1}{\sqrt{7}} & 0 & \frac{1}{\sqrt{7}}\frac{1}{\sqrt{9}} \end{bmatrix}}_{\underline{\tilde{S}}_{\text{inc}}} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}. \tag{4.24}$$

Unfortunately, we cannot choose $\bar{N} = \underline{\tilde{S}}_{\text{inc}}\bar{p}$ for our finite element method. This is because $\underline{\tilde{S}}_{\text{inc}}\bar{p}$ does not span a complete space of polynomials; that is, it does not contain the constant function. To see why, try to find a linear combination of rows of $\underline{\tilde{S}}_{\text{inc}}$ which results in $c\bar{e}_1^T$ where c is any constant. This would lead to cp_0 being representable in the set of integrals of orthonormal Legendre polynomials, and since the function p_0 is the constant function, then $\underline{\tilde{S}}_{\text{inc}}\bar{p}$ would contain the constant function. To do this in our example, we must take row 2 as it is the only row with an entry in the first column. We then are forced to scale row 4 appropriately and add it to row 2 to cancel the entry in column 3 of row 2, but this leaves a nonzero entry in column 5. The same is true for any number of additional rows and columns when n is an arbitrary integer, and we confirm that it is impossible to represent a constant function with the integrals of orthonormal Legendre polynomials. To circumvent this problem, we add the constant function to our set of basis functions so that

$$\bar{N} = \begin{bmatrix} p_0 \\ \int \bar{p}\,dx \end{bmatrix}, \quad \underline{\tilde{S}} = \begin{bmatrix} \bar{e}_1^T \\ \underline{\tilde{S}}_{\text{inc}} \end{bmatrix}, \tag{4.25}$$

and $\bar{N} = \underline{\tilde{S}}\bar{p}$. This additional row renders $\underline{\tilde{S}}$ a square matrix. In addition, since $\underline{\tilde{S}}$ is lower triangular with nonzero diagonal entries, its determinant is nonzero. This means that $\underline{\tilde{S}}$ is invertible which confirms that the space of polynomials spanned by the orthonormal Legendre basis and this modified basis is the same. For arbitrary polynomial degree L, we can write the integration matrix $\underline{\tilde{S}}$ by defining the vector $\bar{g} \in \mathbb{R}^{L\times 1}$ with entries

$$g_i = \begin{cases} 1 & i = 1 \\ \dfrac{1}{\sqrt{4\left(i-1\right)^2 - 1}} & \text{otherwise} \end{cases} \tag{4.26}$$

and the shift matrix

$$\underline{S}_{DL} = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & \ddots & & \\ & & \ddots & 0 & \\ & & & 1 & 0 \end{bmatrix} \tag{4.27}$$

which shifts entries in a matrix down a row when multiplying from the left (adding zeros to the first row), and shifts entries left one column when multiplying from the right (adding zeros to the last column). With this vector and shift matrix, the integration matrix is given by

$$\underline{\tilde{S}} = \text{diag}\left(\bar{g}\right) - \underline{S}_{DL}\text{diag}\left(\bar{g}\right)\underline{S}_{DL}. \tag{4.28}$$

We now return to the matrix $\underline{A}$ which, with this change of basis, is given by

$$\underline{A} = \alpha\underline{\tilde{S}}\underline{\tilde{D}}\underline{\tilde{D}}^T\underline{\tilde{S}}^T + \beta\underline{\tilde{S}}\underline{\tilde{S}}^T + \gamma\left[\bar{N}\bar{N}^T\right]_{x=1} \tag{4.29}$$

where we have replaced $\underline{\tilde{V}}^{-T}$ with $\underline{\tilde{S}}$ in (4.19). We now have three terms to investigate. We begin with the first by computing the product $\underline{\tilde{S}}\underline{\tilde{D}}$. Remember, the intention was that, in choosing $\bar{N}$ to be the vector of integrals of Legendre polynomials (supplemented by the constant function), we would obtain sparsity in the product $\underline{\tilde{S}}\underline{\tilde{D}}$. We now demonstrate that this is true. Recall (from Chapter 2) that $\underline{\tilde{D}} = \text{diag}(\bar{h})\underline{H}\text{diag}(\bar{h})$ where

$$h_i = \sqrt{2(i-1)+1}, \quad H_{ij} = \begin{cases} \text{mod}\left(i+j,2\right) & j \leq i \\ 0 & \text{otherwise.} \end{cases} \tag{4.30}$$

Then

$$\underline{\tilde{S}}\underline{\tilde{D}} = \left[\text{diag}\,(\bar{g}) - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\right]\left[\text{diag}(\bar{h})\underline{H}\text{diag}(\bar{h})\right] \tag{4.31}$$

$$= \left[\text{diag}\,(\bar{g})\,\text{diag}(\bar{h}) - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\text{diag}(\bar{h})\right]\underline{H}\text{diag}(\bar{h}) \tag{4.32}$$

where we have postmultiplied by $\text{diag}(\bar{h})$ from the right. The term

$$\underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\text{diag}(\bar{h}) = \text{diag}\,(\bar{g})\,\text{diag}(\bar{h})\underline{S}_{DL}\underline{S}_{DL}. \tag{4.33}$$

This commutativity of matrix multiplication is not true in general but is due to the specific entries of $\bar{g}$ and $\bar{h}$ (the identity is false for arbitrary $\bar{g}$ and $\bar{h}$). To see why this equality holds, $\underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}$ is $\text{diag}\,(\bar{g})$ shifted down and to the left with zeros in the first row and last column. This is followed by multiplication from the right by $\text{diag}(\bar{h})$ which scales the columns of $\underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}$ by $\bar{h}$. Compare this to multiplying $\bar{g}$ and $\bar{h}$ entrywise along the diagonal and then shifting the result left twice; the resulting matrices are identical[1]. Thus

$$\underline{\tilde{S}}\underline{\tilde{D}} = \text{diag}\,(\bar{g})\,\text{diag}(\bar{h})\left[\underline{I} - \underline{S}_{DL}^2\right]\underline{H}\text{diag}(\bar{h}) \tag{4.34}$$

$$= \text{diag}\,(\bar{g})\,\text{diag}(\bar{h})\left[\underline{H} - \underline{S}_{DL}^2\underline{H}\right]\text{diag}(\bar{h}) \tag{4.35}$$

where we have used this special commuting property to factor the diagonal scaling matrices from the left. Examining the matrix in square brackets, we use the structure of $\underline{H}$ which is strictly lower triangular with alternating subdiagonals of ones and zeros to conclude that

$$\underline{H} - \underline{S}_{DL}^2\underline{H} = \underline{S}_{DL}. \tag{4.36}$$

This is true because $\underline{S}_{DL}^2\underline{H}$ is 2 shifts downward of the alternating subdiagonal matrix $\underline{H}$. Subtraction of this product from $\underline{H}$ cancels all nonzero subdiagonals with the exception of the first. This gives

$$\underline{\tilde{S}}\underline{\tilde{D}} = \text{diag}\,(\bar{g})\,\text{diag}(\bar{h})\underline{S}_{DL}\text{diag}(\bar{h}) \tag{4.37}$$

which, after scaling of rows by $\text{diag}\,(\bar{g})\,\text{diag}(\bar{h})$ and scaling of columns by $\text{diag}(\bar{h})$ yields

$$\underline{\tilde{S}}\underline{\tilde{D}} = \underline{S}_{DL}, \tag{4.38}$$

again, because of the specific entries in $\bar{g}$ and $\bar{h}$. Having shown this simplification, we return

[1]The simplest way to verify this is by direct computation. Take the case $L = 5$ for simplicity as the general case can be seen immediately from there.

to the first term of $\underline{A}$ to see that

$$\alpha\underline{\tilde{S}}\underline{\tilde{D}}\underline{\tilde{D}}^T\underline{\tilde{S}}^T = \alpha\underline{\tilde{S}}\underline{\tilde{D}}[\underline{\tilde{S}}\underline{\tilde{D}}]^T \tag{4.39}$$

$$= \alpha\underline{S}_{DL}\underline{S}_{DL}^T \tag{4.40}$$

which is indeed quite simple since $\underline{S}_{DL}\underline{S}_{DL}^T$ is the identity matrix with the first entry along the main diagonal set to zero.

Next, we examine the structure of the second term which involves the multiplication $\underline{\tilde{S}}\underline{\tilde{S}}^T$. We know that

$$\underline{\tilde{S}}\underline{\tilde{S}}^T = \left[\text{diag}\,(\bar{g}) - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\right]\left[\text{diag}\,(\bar{g}) - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\right]^T \tag{4.41}$$

$$= \left[\text{diag}\,(\bar{g}) - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\right]\left[\text{diag}\,(\bar{g}) - \underline{S}_{DL}^T\text{diag}\,(\bar{g})\,\underline{S}_{DL}^T\right] \tag{4.42}$$

which—after multiplication—yields four terms

$$\underline{\tilde{S}}\underline{\tilde{S}}^T = \text{diag}\,(\bar{g})\,\text{diag}\,(\bar{g}) + \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\underline{S}_{DL}^T\text{diag}\,(\bar{g})\,\underline{S}_{DL}^T$$
$$- \text{diag}\,(\bar{g})\,\underline{S}_{DL}^T\text{diag}\,(\bar{g})\,\underline{S}_{DL}^T - \underline{S}_{DL}\text{diag}\,(\bar{g})\,\underline{S}_{DL}\text{diag}\,(\bar{g})\,. \tag{4.43}$$

The first two terms contribute only to the main diagonal, the third term contributes to the second superdiagonal, and the fourth term contributes to the second subdiagonal. The result is a pentadiagonal matrix with three nonzero diagonals. Thus, in choosing the integrals of Legendre polynomials as our basis functions we have—in a sense—balanced sparsity in both the first and second terms of $\underline{A}$. This is an improvement in sparsity over interpolatory Lagrange polynomials (full matrices for both terms) and orthonormal Legendre polynomials (full matrix for the first term, diagonal for the second).

However, before we can be sure that $\underline{A}$ is sparse, we need to consider the third term arising from the Robin boundary condition. To do so, we need to know what these modified basis functions evaluate to at the endpoints of the domain $(-1, 1)$. We consider properties of the orthonormal Legendre polynomials [117]. In particular,

$$p_n\,(1) = \sqrt{\frac{2n+1}{2}}. \tag{4.44}$$

Using (4.23), we see that, for $n > 0$,

$$\left[\int p_n\,dx\right]_{x=1} = -\frac{1}{\sqrt{4n^2-1}}p_{n-1}\,(1) + \frac{1}{\sqrt{4\,(n+1)^2-1}}p_{n+1}\,(1)\,. \tag{4.45}$$

Property (4.44) gives

$$\left[\int p_n\,dx\right]_{x=1} = -\frac{1}{\sqrt{4n^2-1}}\sqrt{\frac{2\,(n-1)+1}{2}} + \frac{1}{\sqrt{4\,(n+1)^2-1}}\sqrt{\frac{2\,(n+1)+1}{2}} \tag{4.46}$$

$$= -\frac{1}{\sqrt{2}\sqrt{2n+1}} + \frac{1}{\sqrt{2}\sqrt{2n+1}} \tag{4.47}$$

$$= 0 \tag{4.48}$$

whereas, for $n = 0$,

$$\left[\int p_0\,dx\right]_{x=1} = \frac{1}{\sqrt{4\,(1)^2-1}}p_1\,(1) \tag{4.49}$$

$$= \frac{1}{\sqrt{3}}\sqrt{\frac{2\,(1)+1}{2}} \tag{4.50}$$

$$= \frac{1}{\sqrt{2}}. \tag{4.51}$$

Finally, $p_0 = 1/\sqrt{2}$ so that

$$\bar{N}\,(1) = \frac{1}{\sqrt{2}}\,(\bar{e}_1 + \bar{e}_2)\,, \tag{4.52}$$

and

$$\gamma\left[\bar{N}\bar{N}^T\right]_{x=1} = \gamma\frac{1}{2}\,(\bar{e}_1 + \bar{e}_2)\,(\bar{e}_1 + \bar{e}_2)^T\,. \tag{4.53}$$

This is also sparse; in fact, only the leading 2-by-2 block is nonzero. Thus, for our BVP, all three terms in $\underline{A}$ are sparse, and $\underline{A}$ is given by

$$\underline{A} = \alpha\underline{S}_{DL}\underline{S}_{DL}^T + \beta\underline{\tilde{S}}\,\underline{\tilde{S}}^T + \gamma\frac{1}{2}\,(\bar{e}_1 + \bar{e}_2)\,(\bar{e}_1 + \bar{e}_2)^T\,. \tag{4.54}$$

In addition, we can let the polynomial degree of the basis L be arbitrarily large while maintaining this sparsity. To summarize, Figure 4.1 contrasts the sparsity patterns for the three choices of basis functions described in this section. Clearly, the third row of Figure 4.1, which corresponds to integrals of Legendre polynomials, balances sparsity in all three components of matrix $\underline{A}$ better than either the interpolatory Lagrange polynomial or Legendre polynomial representations (rows one and two respectively).

For completeness, we also investigate the case where the Robin boundary condition is imposed at $x = -1$ to make sure that this case is also sparse. We note that

$$p_n\,(-x) = (-1)^n\,p_n\,(x) \tag{4.55}$$

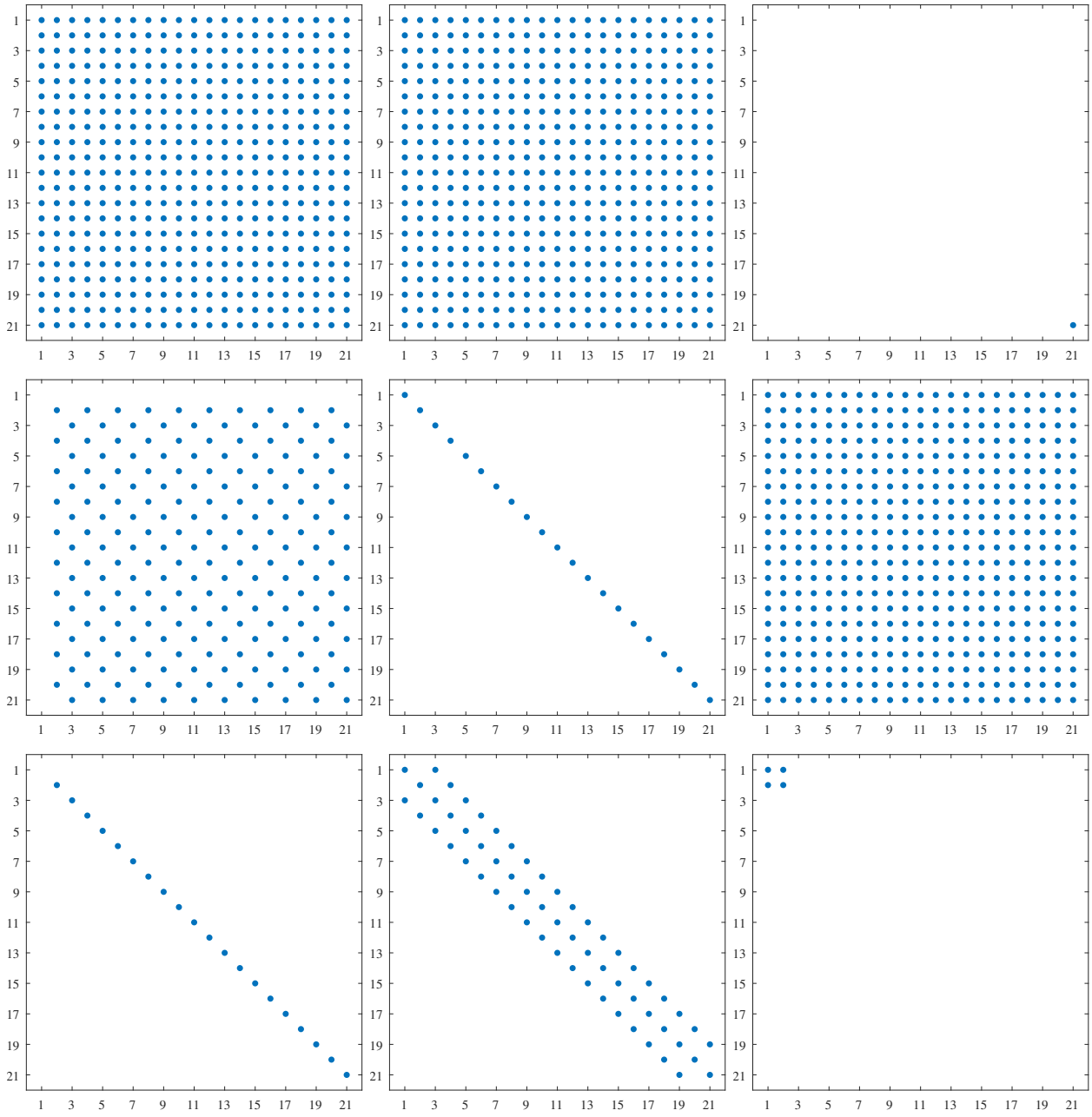

Figure 4.1: Comparison of sparsity patterns for different terms in $\underline{A}$. Row 1 corresponds to interpolatory Lagrange polynomials, row 2 corresponds to Legendre polynomials, and row 3 corresponds to integrated Legendre polynomials. Each column represents the sparsity pattern for a single term in $\underline{A}$. Column 1 corresponds to the α term, column 2 corresponds to the β term, and column 3 corresponds to the γ term. Degree $L = 20$ polynomials are used in all cases.

so that

$$p_n(-1) = (-1)^n p_n(1) \tag{4.56}$$

$$= (-1)^n \sqrt{\frac{2n+1}{2}}. \tag{4.57}$$

Then, using (4.23), we observe that

$$\left[\int p_n \, dx\right]_{x=-1} = \begin{cases} 0 & n > 0 \\ -\dfrac{1}{\sqrt{2}} & n = 0. \end{cases} \tag{4.58}$$

We now can write

$$\bar{N}(-1) = \frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2), \tag{4.59}$$

confirming that an identical sparsity pattern is also obtained in this case. These two cases for the Robin boundary condition are also important when considering Dirichlet boundary conditions because either $\bar{N}(1)$ or $\bar{N}(-1)$ may arise in a constraint equation. In our example BVP, we have $\phi(-1) = p$ which leads to the constraint

$$\bar{N}(-1)^T \bar{\phi} = p \tag{4.60}$$

$$\frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \bar{\phi} = p \tag{4.61}$$

and the constraint matrix

$$\underline{C} = \frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \tag{4.62}$$

which is also sparse (the constraint vector is $\bar{d} = p$). Combining this constraint matrix with $\underline{A}$, we note that, with this basis, solving the saddle point system directly or by the associated null space method is an $\mathcal{O}(L)$ process since we can exploit the pentadiagonal structure of $\underline{A}$. This is a fast method.

In addition to speed, it is also important to consider whether solving such a system is numerically stable. To characterize this stability, we compute the condition number of matrix $\underline{A}$ for various choices of basis and polynomial degree. As an example, Figure 4.2 illustrates the condition number measured in the 2-norm for a given fixed set of parameters $\alpha = 1$, $\beta = -1$, and $\gamma = 0$. Notice how the monomials and interpolatory Lagrange polynomials in terms of equidistant nodes lead to extremely ill-conditioned matrices for polynomial degrees larger than 5. Interpolatory Lagrange polynomials in terms of Chebyshev nodes as well as Legendre polynomials are better behaved but grow increasingly ill-conditioned as the degree

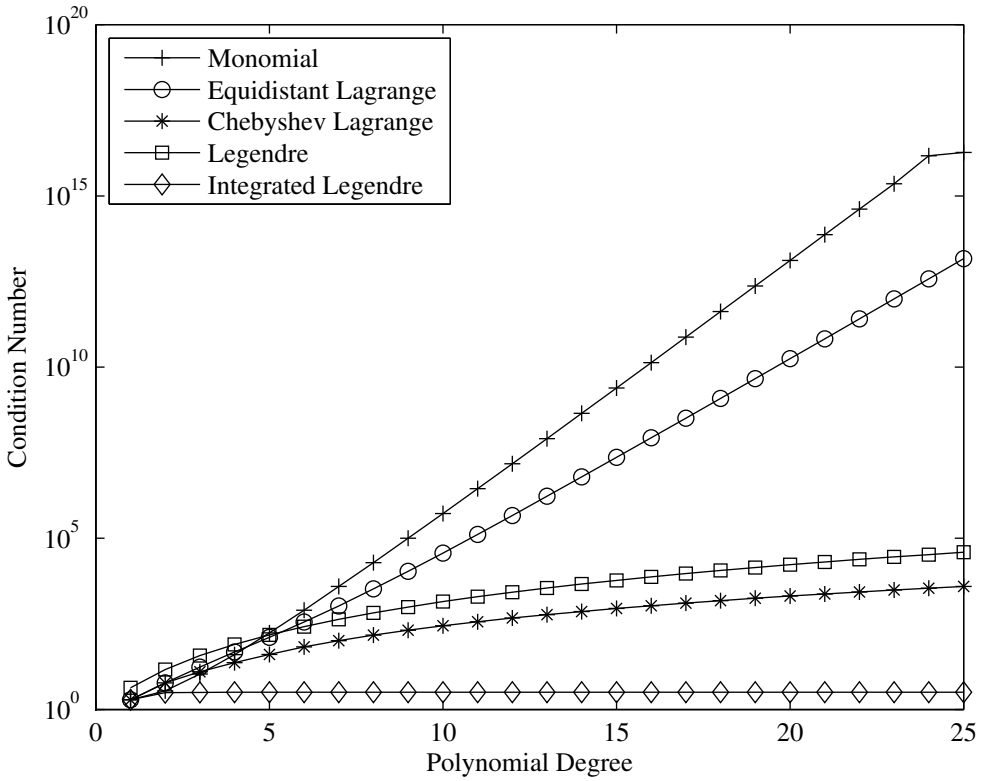

Figure 4.2: Condition number $\kappa_2(\underline{A})$ with parameters $\alpha = 1$, $\beta = -1$, and $\gamma = 0$ fixed, as a function of increasing polynomial degree L. Comparison of the condition number for different basis functions for lower polynomial degrees. Basis functions are monomials, interpolatory Lagrange polynomials with equidistant nodes and Chebyshev nodes, Legendre polynomials, and integrated Legendre polynomials.

is increased. Integrated Legendre polynomials are well-conditioned, even for high polynomial degree. This is because the eigenvalues of the second term in $\underline{A}$ decay to zero, and the first term is effectively the identity matrix. The matrix behaves like a compact perturbation of the identity matrix with eigenvalues clustering near unity. Figure 4.3 shows the trend in Figure 4.2 extended to higher polynomial degrees. It should be pointed out that in general the conditioning of $\underline{A}$ depends on the choice of parameters α and β. For example, increasing β has the effect of increasing the condition number for the integrated Legendre basis but does not change the fact that this condition number is effectively independent of the polynomial degree.

Finally, we take this opportunity to give a third representation of the integrated Legendre polynomials (the first is their definition as integrals of Legendre polynomials and the second is their description as linear combinations of Legendre polynomials given by $\bar{N} = \underline{\tilde{S}}\bar{p}$). These

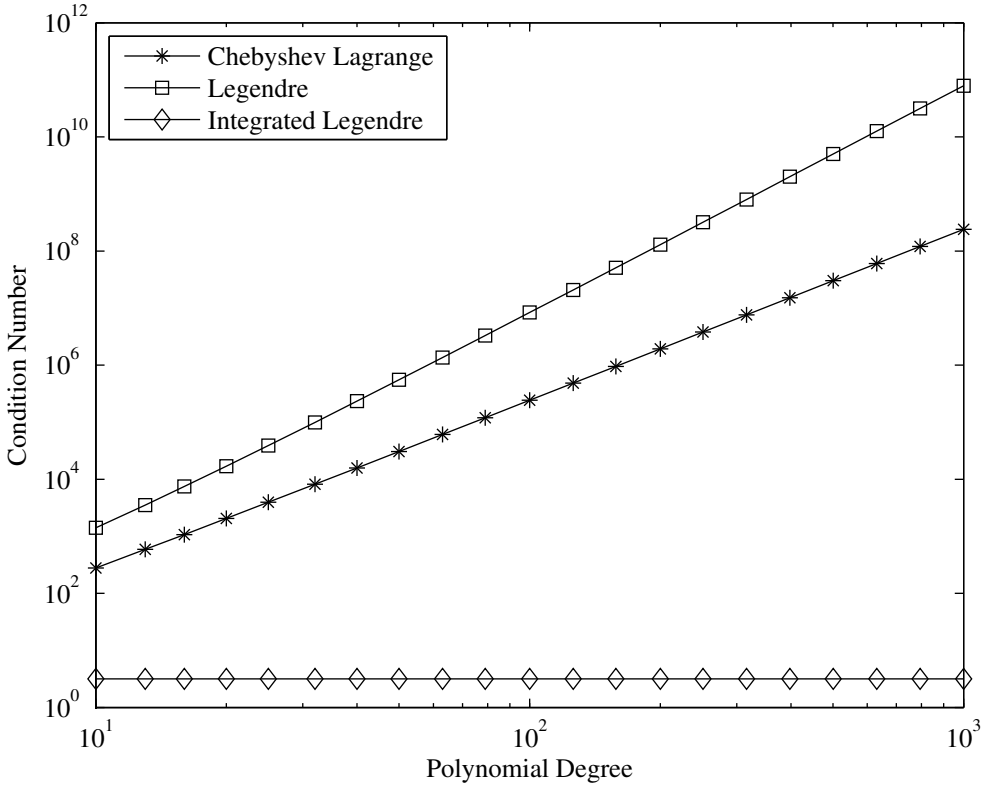

Figure 4.3: Condition number $\kappa_2(\underline{A})$ with parameters $\alpha = 1$, $\beta = -1$, and $\gamma = 0$ fixed, as a function of increasing polynomial degree L. Comparison of the condition number for different basis functions for higher polynomial degrees. Monomials and interpolatory Lagrange polynomials with equidistant nodes are not shown due to their extremely poor conditioning.

polynomials can also be defined as

$$N_0(x) = \frac{1}{\sqrt{2}}, \tag{4.63}$$

$$N_1(x) = \frac{x}{\sqrt{2}}, \tag{4.64}$$

$$N_k(x) = -\frac{(1-x)(1+x)}{\sqrt{k(k-1)}} p_{k-2}^{(1,1)}(x), \qquad k = 2, 3, 4, ... \tag{4.65}$$

where $p_k^{(\alpha,\beta)}(x)$ is an orthonormal Jacobi polynomial. For $k \geq 2$, the polynomials are eigenfunctions satisfying

$$\frac{d^2}{dx^2} N_k + \frac{\lambda_k}{(1-x)(1+x)} N_k = 0 \tag{4.66}$$

with eigenvalues $\lambda_k = k(k-1)$ subject to homogeneous Dirichlet boundary conditions at $x = \pm 1$. Depending on the literature, any of these three representations can appear; they are identical. However, often the first two polynomials $N_0(x)$ and $N_1(x)$ are replaced by

interpolatory polynomials

$$l_0(x) = \frac{1+x}{2}, \qquad l_1(x) = \frac{1-x}{2}. \tag{4.67}$$

This simplifies certain boundary expressions at the cost of reducing the sparsity of $\underline{A}$. In practice, we transform between the two representations using

$$\begin{bmatrix} N_0 \\ N_1 \end{bmatrix} = \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{\underline{Q}} \begin{bmatrix} l_0 \\ l_1 \end{bmatrix}. \tag{4.68}$$

Note that $\underline{Q}$ is orthogonal and symmetric and thus involutory, which means that $\underline{Q}^{-1} = \underline{Q}$. Therefore, if $\tilde{N}(x)$ is the vector containing the integrated Legendre polynomials with first two polynomials interpolatory, then

$$\tilde{N}(x) = \underbrace{\begin{bmatrix} \underline{Q} & 0 \\ 0 & \underline{I}_{L-1} \end{bmatrix}}_{\underline{B}} \bar{N}(x) \tag{4.69}$$

and also $\bar{N}(x) = \underline{B}\tilde{N}(x)$. Babuška and Szabó describe $\tilde{N}(x)$ using integrals of Legendre polynomials [48], Shen describes $\tilde{N}(x)$ using linear combinations of Legendre polynomials [49], and Sherwin and Karniadakis describe $\tilde{N}(x)$ using Jacobi polynomials [37]. These references use numerical integration to assemble $\underline{A}$ when computing with spatially varying α and β or to assemble the forcing vector when computing with spatially varying forcing function f. By doing so, the sparsity of the operator matrix $\underline{A}$ no longer holds. Section 4.2 describes how to represent the forcing function f as a linear combination of Legendre polynomials and emphasizes how to efficiently compute this representation. Then, in Section 4.3, such Legendre expansions are used to extend the method described thus far to cases with variable α and β in a way that continues to preserve sparsity wherever possible.

4.2 Computing the Forcing Term

In this section, we consider how to compute the vector $\bar{b}$ given by

$$\bar{b} = \int_{-1}^{1} \bar{N} f \, dx - \left[q\bar{N} \right]_{x=1}. \tag{4.70}$$

The second term is straightforward as we can use the results obtained in applying the Robin boundary condition directly to show that

$$\left[q\bar{N}\right]_{x=1} = q\frac{1}{\sqrt{2}}\left(\bar{e}_1 + \bar{e}_2\right). \tag{4.71}$$

The first term requires some comment. First, recall that f is an arbitrary function. This means that, in a general code, evaluating the first term must be done numerically. For this reason, most finite element codes use a Gaussian quadrature rule to evaluate each entry in the vector [12]. If the quadrature rule uses L_{quad} nodes, then the cost of computing the first term is $\mathcal{O}\left(L_{\text{quad}}L\right)$. If $L_{\text{quad}} \approx L$, which is typical, then the computational bottleneck in our fast solver is precisely the evaluation of this term. To circumvent this apparent problem, let us again use the framework set up in Chapter 2.

Suppose we approximate f as a sum of orthonormal Legendre polynomials such that

$$f \approx \bar{p}^T \bar{f} \tag{4.72}$$

where $\bar{f}$ is a vector containing the coefficients in this expansion. Using this in our expression for the first term gives

$$\int_{-1}^{1} \bar{N} f \, dx = \int_{-1}^{1} \underline{\tilde{S}} \bar{p} f \, dx \tag{4.73}$$

$$\approx \int_{-1}^{1} \underline{\tilde{S}} \bar{p} \bar{p}^T \bar{f} \, dx \tag{4.74}$$

$$\approx \underline{\tilde{S}} \underbrace{\int_{-1}^{1} \bar{p}\bar{p}^T dx}_{\underline{I}_{L+1}} \bar{f} \tag{4.75}$$

$$\approx \underline{\tilde{S}} \bar{f} \tag{4.76}$$

and since $\underline{\tilde{S}} = \text{diag}\left(\bar{g}\right) - \underline{S}_{DL}\text{diag}\left(\bar{g}\right)\underline{S}_{DL}$, applying $\underline{\tilde{S}}$ to the vector $\bar{f}$ is an $\mathcal{O}\left(L\right)$ computation involving scaling and shifting. The work has been transferred into computing the coefficients in $\bar{f}$. In fact, we have not accomplished much, since any continuous function on $(-1, 1)$ can be written as an infinite sum of Legendre polynomials p_k multiplied by some coefficients f_k given by

$$f = \sum_{i=0}^{\infty} f_i p_i. \tag{4.77}$$

The coefficients are found by multiplying by p_k and integrating:

$$\int_{-1}^{1} p_k f \, dx = \sum_{i=0}^{\infty} f_i \int_{-1}^{1} p_k p_i \, dx \tag{4.78}$$

$$\int_{-1}^{1} p_k f \, dx = f_k. \tag{4.79}$$

In vector notation, $\bar{f} = \int_{-1}^{1} \bar{p} f \, dx$, and we have a problem very similar to the one we first encountered in (4.73). That being said, there are a variety of algorithms that can be used to compute $\bar{f}$.

First, let us approximate f by interpolation. Then,

$$\bar{f} = \int_{-1}^{1} \bar{p} f \, dx \tag{4.80}$$

$$\approx \int_{-1}^{1} \bar{p} \left[\bar{p}^T \underline{\tilde{V}}^{-1} \bar{f}_s \right] dx \tag{4.81}$$

$$\approx \underbrace{\int_{-1}^{1} \bar{p} \bar{p}^T dx}_{\underline{I}_{L+1}} \underline{\tilde{V}}^{-1} \bar{f}_s \tag{4.82}$$

$$\approx \underline{\tilde{V}}^{-1} \bar{f}_s \tag{4.83}$$

where $\underline{\tilde{V}}$ is the generalized Vandermonde matrix evaluated at a set of interpolation nodes and $\bar{f}_s$ is the vector containing the values of the function f at the nodes. While inversion of the dense matrix $\underline{\tilde{V}}$ is naively a $\mathcal{O}(L^3)$ process, there exist specialized algorithms which exploit the structure of the generalized Vandermonde matrix to compute $\bar{f}$ in $\mathcal{O}(L^2)$ operations [112].

Alternatively, we can apply numerical integration to write

$$\bar{f} \approx \sum_{j=0}^{M} w_j \bar{p}(x_j) f(x_j) \tag{4.84}$$

$$\approx \underbrace{\begin{bmatrix} \bar{p}(x_0) & \bar{p}(x_1) & \dots & \bar{p}(x_M) \end{bmatrix}}_{\underline{\tilde{V}}^T} \mathrm{diag}(\bar{w}) \bar{f}_s \tag{4.85}$$

so that $\bar{f} = \underline{\tilde{V}}^T \mathrm{diag}(\bar{w}) \bar{f}_s$ where $\bar{w}$ is the vector of weights w_j and x_j are the nodes for the integration rule. Clearly, multiplication by $\mathrm{diag}(\bar{w})$ is $\mathcal{O}(M)$ FLOPs whereas multiplication by $\underline{\tilde{V}}^T$ is $\mathcal{O}(ML)$ FLOPs. Of particular interest is the case where $M = L$ and x_j are the roots of p_{L+1}. In this case, the integration rule is the standard Gauss-Legendre rule, and $\underline{\tilde{V}}^{-1} = \underline{\tilde{V}}^T \mathrm{diag}(\bar{w})$ so that the two methods are in fact identical. For our purposes, we use

$M = 2L$ and x_j the Chebyshev nodes so as to avoid computing the roots of p_{L+1}[2], however, in this case the two methods are distinct. When L exceeds 256, we use a fast method to compute the Legendre coefficients [51]. The method first computes Chebyshev coefficients in $\mathcal{O}(L \log L)$ FLOPs, then converts the Chebyshev coefficients to Legendre coefficients in $\mathcal{O}(L(\log L)^2)$ FLOPs. Alternative algorithms exist, such as [119, 120, 121], but they have not been considered in this work. For a detailed description of the fast method, see Appendix A.

Putting these results together, we have a fast solver for the constant coefficient case of our BVP on a single element with saddle point system

$$\begin{bmatrix} \alpha \underline{S}_{DL} \underline{S}_{DL}^T + \beta \tilde{\underline{S}} \tilde{\underline{S}}^T + \gamma \frac{1}{2} (\bar{e}_1 + \bar{e}_2)(\bar{e}_1 + \bar{e}_2)^T & \frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2) \\ \frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \tilde{\underline{S}} \bar{f} - q \frac{1}{\sqrt{2}} (\bar{e}_1 + \bar{e}_2) \\ p \end{bmatrix}. \quad (4.86)$$

All entries on the left are known explicitly, and only $\bar{f}$ on the right needs to be computed. To do so, we begin by computing $\bar{f}$ for $L = 8$ and continue to increase L (either by doubling or by incrementing by 1) until at least the last two coefficients in $\bar{f}$ have decayed below a user specified tolerance (say $100\epsilon_{\text{machine}} \approx 2 \times 10^{-14}$ in double precision taken relative to the magnitude of f). We require two coefficients to be zero consecutively because the Legendre polynomials are even and odd functions, and if the function f is even or odd, it is possible for all even or odd coefficients to be zero (to the relative tolerance) but their counterparts nonzero. Once this criterion is met, we can construct the saddle point matrix for the L that was sufficient in representing f to the relative tolerance as long as this new $L \geq 1$ (effectively throwing away coefficients that were zero to the relative tolerance). We then solve the system using a direct sparse method for the pentadiagonal system. We can either do this directly on the saddle point system (4.86), or we can use the null space method by taking $\underline{Z} = \begin{bmatrix} \bar{e}_1 + \bar{e}_2 & \bar{e}_3 & \bar{e}_4 & \dots & \bar{e}_{L+1} \end{bmatrix}$ and $\bar{\phi}_z = \sqrt{2} p \bar{e}_1$. Note that $\underline{Z}$ is a sparse basis for the null space; that is, we have not lost this property by changing to the basis of integrals of Legendre polynomials[3].

Once we have computed $\bar{\phi}$, we can verify how well our computed solution approximates

[2]It should be pointed out that there exists a fast $\mathcal{O}(L)$ algorithm for the computation of the Gauss-Legendre nodes and weights when $L > 100$, although the issue of multiplying by $\tilde{V}^T$ with less than $\mathcal{O}(L^2)$ FLOPs is not obvious [118].

[3]The null space method is not very useful here because it reduces the size of the linear system by only one equation and one unknown. It will become more important when we move to multiple elements.

the true solution to the BVP by changing to the Legendre basis. We have that

$$\phi \approx \bar{\phi}^T \bar{N} \tag{4.87}$$

$$\approx \underbrace{\bar{\phi}^T \underline{\tilde{S}}}_{\bar{\phi}_{\text{Leg}}^T} \bar{p} \tag{4.88}$$

so that $\bar{\phi}_{\text{Leg}} = \underline{\tilde{S}}^T \bar{\phi}$ are coefficients for the computed solution in the Legendre basis. Multiplication by $\underline{\tilde{S}}^T$ costs $\mathcal{O}(L)$ FLOPs so the conversion is inexpensive. In the Legendre basis, we know that the coefficients decay based on the smoothness of the underlying function [109] so we can expect such a decay in the coefficients of our solution. If the coefficients have not decayed to the user defined tolerance, we increase L and solve the system again (by either doubling or incrementing L by 1). Note that the expensive part of computing $\bar{f}$ need not be recomputed; we pad $\bar{f}$ with zeros as we have already resolved the function f to the required tolerance, but have not resolved ϕ to a similar tolerance. Note that the structure of the saddle point system does not change with increasing L, we need only add rows and columns to $\underline{A}$ and $\underline{C}$, and they remain sparse.

4.3 Spatially Varying Parameters

The sparsity results obtained in Section 4.1 are valid only when α and β are constant over the interval $(-1, 1)$. Let us now allow both to vary spatially so that

$$\underline{A} = \int_{-1}^{1} \left[\frac{d}{dx}\bar{N}\right] \alpha(x) \left[\frac{d}{dx}\bar{N}\right]^T dx + \int_{-1}^{1} \bar{N}\beta(x)\bar{N}^T dx + \gamma \left[\bar{N}\bar{N}^T\right]_{x=1}. \tag{4.89}$$

Here, there can be no general sparsity results as the sparsity of each integral depends on the arbitrary functions α and β. Nevertheless, to describe sparsity, we borrow the idea used in computing the forcing term $\bar{b}$ in Section 4.2. That is, we approximate α and β by two separate Legendre expansions

$$\alpha(x) \approx \sum_{k=0}^{K_\alpha} \alpha_k p_k, \quad \beta(x) \approx \sum_{k=0}^{K_\beta} \beta_k p_k, \tag{4.90}$$

where K_α and K_β are chosen large enough to represent α and β to the user specified tolerance in the same way that f was approximated, although here we allow for the possibility that K_α or K_β are zero (this recovers the constant case we started with). Let us examine how this change in the first term of $\underline{A}$ affects its sparsity.

We begin by using the approach characteristic of Section 4.1 of replacing $\bar{N}$ by $\underline{\tilde{S}}\bar{p}$ and

differentiation by $\underline{\tilde{D}}$ to obtain

$$\int_{-1}^{1}\left[\frac{d}{dx}\bar{N}\right]\alpha\left(x\right)\left[\frac{d}{dx}\bar{N}\right]^{T}dx=\int_{-1}^{1}\left[\underline{\tilde{S}}\underline{\tilde{D}}\bar{p}\right]\alpha\left(x\right)\left[\underline{\tilde{S}}\underline{\tilde{D}}\bar{p}\right]^{T}dx \tag{4.91}$$

$$=\int_{-1}^{1}\left[\underline{S}_{DL}\bar{p}\right]\alpha\left(x\right)\left[\underline{S}_{DL}\bar{p}\right]^{T}dx \tag{4.92}$$

$$=\underline{S}_{DL}\int_{-1}^{1}\bar{p}\,\alpha\left(x\right)\bar{p}^{T}dx\,\underline{S}_{DL}^{T}. \tag{4.93}$$

We now substitute the Legendre expansion for α to find

$$\int_{-1}^{1}\left[\frac{d}{dx}\bar{N}\right]\alpha\left(x\right)\left[\frac{d}{dx}\bar{N}\right]^{T}dx=\underline{S}_{DL}\int_{-1}^{1}\bar{p}\left[\sum_{k=0}^{K_{\alpha}}\alpha_{k}p_{k}\right]\bar{p}^{T}dx\,\underline{S}_{DL}^{T} \tag{4.94}$$

$$=\underline{S}_{DL}\left[\sum_{k=0}^{K_{\alpha}}\alpha_{k}\int_{-1}^{1}p_{k}\bar{p}\bar{p}^{T}dx\right]\underline{S}_{DL}^{T} \tag{4.95}$$

$$=\underline{S}_{DL}\left[\sum_{k=0}^{K_{\alpha}}\alpha_{k}\underline{T}_{k}\right]\underline{S}_{DL}^{T} \tag{4.96}$$

where each matrix

$$\underline{T}_{k}=\int_{-1}^{1}p_{k}\bar{p}\bar{p}^{T}dx \tag{4.97}$$

is a frontal slice of the order three tensor[4] $\underline{\underline{T}}\in\mathbb{R}^{(L+1)\times(L+1)\times(K_{\alpha}+1)}$ whose entries are given by

$$\left(\underline{\underline{T}}\right)_{ijk}=\int_{-1}^{1}p_{i-1}p_{j-1}p_{k-1}dx. \tag{4.98}$$

Repeating similar operations, we can also find an expression for the second term in $\underline{A}$, which is

$$\int_{-1}^{1}\bar{N}\beta\left(x\right)\bar{N}^{T}dx=\underline{\tilde{S}}\left[\sum_{k=0}^{K_{\beta}}\beta_{k}\underline{T}_{k}\right]\underline{\tilde{S}}^{T}. \tag{4.99}$$

Thus, to understand the sparsity of either term in $\underline{A}$, we must understand the structure of the matrices $\underline{T}_{k}$.

To do this, we need to be able to compute the integral of triple products of Legendre polynomials. This property of the Legendre polynomials is, to my knowledge, not listed in standard texts on orthogonal polynomials. However, it is common to find the integral of triple products of Legendre polynomials by reading sources describing the $3j$ symbol found in quantum mechanics [52, 123]. This work uses the classical formula [124] given by Adams

[4] See [122] for standard notation and nomenclature regarding tensors.

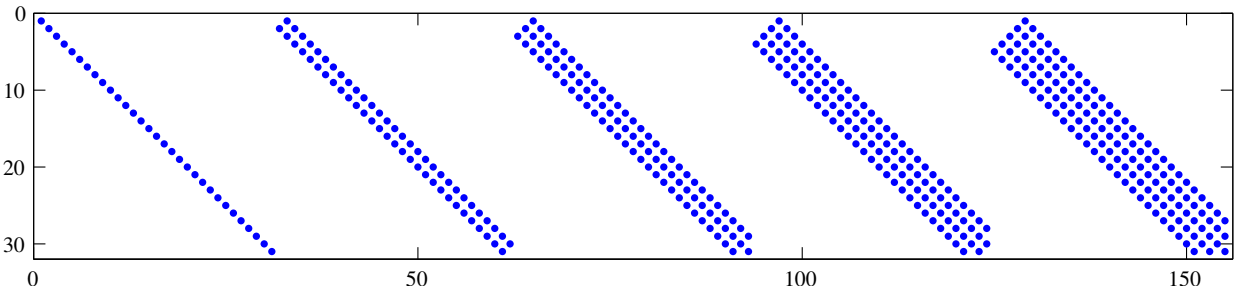

Figure 4.4: Sparsity pattern for the tensor $\underline{\underline{T}}$ with dimensions $L = 30$ and $K_\alpha = 4$ (or $K_\beta = 4$). The slices $\underline{T}_k$ with $k = 0, 1, ..., 4$ are arranged from left to right. The total number of entries in $\underline{\underline{T}}$ is $5 \times 31^2 = 4,805$ and the total number of nonzeros is 432.

to compute the integrals[5]. Here we restate Adams' formula, but modified for orthonormal Legendre polynomials which we use to populate $\underline{\underline{T}}$:

$$\int_{-1}^{1} p_i p_j p_k dx = \begin{cases} C_{ijk} \dfrac{A(s-i)\,A(s-j)\,A(s-k)}{A(s)} & i+j+k \text{ even}, i+j \geq k, |i-j| \leq k, \\ 0 & \text{otherwise}, \end{cases} \tag{4.100}$$

where

$$C_{ijk} = \frac{2}{2s+1}\sqrt{\frac{2i+1}{2}}\sqrt{\frac{2j+1}{2}}\sqrt{\frac{2k+1}{2}}, \tag{4.101}$$

$$A(m) = 1 \cdot \frac{3}{2} \cdot \frac{5}{3} \cdot \ldots \cdot \left(2 - \frac{1}{m}\right), \tag{4.102}$$

$A(0) = 1$, and $s = (i+j+k)/2$. The sparsity of the tensor $\underline{\underline{T}}$ is visualized in Figure 4.4 for a particular example.

Let us comment on the sparsity of $\underline{\underline{T}}$ in more depth. First, the condition that $i+j+k$ be an even integer means that for each slice $\underline{T}_k$, we will observe a checkerboard pattern of nonzeros which alternates with k. Second, the condition $i+j \geq k$ means that for each slice, there is an anti-diagonal before and including which all anti-diagonal entries are zero (the kth anti-diagonal counting from the top left corner of each slice). Third, the condition $|i-j| \leq k$ implies that each slice has a bandwidth k. However, for a general function α (the β case is completely analogous, simply replace coefficients α_k with β_k and K_α with

[5] Adams derived the formula by first computing an expression for the product of two Legendre polynomials in terms of a sum of weighted Legendre polynomials. He accomplished this by direct computation of $p_1 p_n$, $p_2 p_n$, $p_3 p_n$, and $p_4 p_n$, after which a pattern emerges for the general case $p_m p_n$ where $m < n$. He then used induction to prove that the general pattern is indeed correct. Finally, as a corollary, the integral of triple products of Legendre polynomials is explicitly obtained since the integral of $p_m p_n$ multiplied by another p_l yields a sum of integrals involving products of two Legendre polynomials, most of which go to zero by orthogonality.

K_β), we obtain a matrix of the form $\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k$ in our expression for $\underline{A}$, thus we are really interested in the sparsity of this sum. Since the checkerboard pattern alternates with k and the bandwidth grows with k for each slice $\underline{T}_k$, the sum $\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k$ with arbitrary nonzero coefficients α_k is banded with bandwidth K_α (the band itself is full). This is desirable if K_α is small compared to the degree of polynomial basis L ($K_\alpha \ll L$) as we preserve the banded nature of the matrix $\underline{A}$, and undesirable if $K_\alpha \approx L$, in which case $\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k$ is roughly full and we lose the sparsity of $\underline{A}$.

For consistency, we verify that this generalization to arbitrary spatially varying α agrees with our results in Section 4.1. In the special case where α is constant, $K_\alpha = 0$ since p_0—the constant orthonormal Legendre polynomial—is sufficient to represent α and

$$\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k = \alpha_0 \underline{T}_0 \tag{4.103}$$

$$= \alpha_0 \frac{\sqrt{2}}{2} \underline{I}_{L+1} \tag{4.104}$$

using our expression for the entries of $\underline{\underline{T}}$. Since

$$\alpha_0 = \int_{-1}^{1} \alpha p_0 \, dx \tag{4.105}$$

$$= \alpha \int_{-1}^{1} p_0 \, dx \tag{4.106}$$

$$= \alpha\sqrt{2}, \tag{4.107}$$

we find that

$$\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k = (\alpha\sqrt{2}) \frac{\sqrt{2}}{2} \underline{I}_{L+1} \tag{4.108}$$

$$= \alpha \underline{I}_{L+1}, \tag{4.109}$$

which agrees with our results assuming α is constant. That is,

$$\int_{-1}^{1} \left[\frac{d}{dx}\bar{N}\right] \alpha(x) \left[\frac{d}{dx}\bar{N}\right]^T dx = \underline{S}_{DL} \left[\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k\right] \underline{S}_{DL}^T \tag{4.110}$$

$$= \underline{S}_{DL} \left[\alpha \underline{I}_{L+1}\right] \underline{S}_{DL}^T \tag{4.111}$$

$$= \alpha \underline{S}_{DL} \underline{S}_{DL}^T, \tag{4.112}$$

as before.

We noted earlier that when K_α or K_β are large with respect to the degree L of the polynomial basis then the banded nature of the matrix $\underline{A}$ is lost. In fact, we can say how sparse $\underline{A}$ will be by looking at the sparsity of the first two terms in (4.89). Recall that the first term is given by

$$\int_{-1}^{1} \left[\frac{d}{dx}\bar{N}\right] \alpha\left(x\right) \left[\frac{d}{dx}\bar{N}\right]^T dx = \underline{S}_{DL} \left[\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k\right] \underline{S}_{DL}^T. \tag{4.113}$$

Since the sum in brackets has bandwidth K_α and multiplication from the left by $\underline{S}_{DL}$ shifts rows down by one and multiplication from the right by $\underline{S}_{DL}^T$ shifts columns right by one, the bandwidth K_α is preserved.

Also, recall that the second term is given by

$$\int_{-1}^{1} \bar{N}\beta\left(x\right)\bar{N}^T dx = \tilde{\underline{S}} \underbrace{\left[\sum_{k=0}^{K_\beta} \beta_k \underline{T}_k\right]}_{\underline{\beta}} \tilde{\underline{S}}^T \tag{4.114}$$

which we rewrite using shift matrices as

$$\tilde{\underline{S}}\underline{\beta}\tilde{\underline{S}}^T = \left[\text{diag}\left(\bar{g}\right) - \underline{S}_{DL}\text{diag}\left(\bar{g}\right)\underline{S}_{DL}\right]\underline{\beta}\left[\text{diag}\left(\bar{g}\right) - \underline{S}_{DL}^T\text{diag}\left(\bar{g}\right)\underline{S}_{DL}^T\right]. \tag{4.115}$$

Multiplying yields four terms

$$\begin{aligned}\tilde{\underline{S}}\underline{\beta}\tilde{\underline{S}}^T = {} & \text{diag}\left(\bar{g}\right)\underline{\beta}\text{diag}\left(\bar{g}\right) + \underline{S}_{DL}\text{diag}\left(\bar{g}\right)\underline{S}_{DL}\underline{\beta}\underline{S}_{DL}^T\text{diag}\left(\bar{g}\right)\underline{S}_{DL}^T \\ & - \text{diag}\left(\bar{g}\right)\underline{\beta}\underline{S}_{DL}^T\text{diag}\left(\bar{g}\right)\underline{S}_{DL}^T - \underline{S}_{DL}\text{diag}\left(\bar{g}\right)\underline{S}_{DL}\underline{\beta}\text{diag}\left(\bar{g}\right).\end{aligned} \tag{4.116}$$

The first term has the same bandwidth K_β as $\underline{\beta}$ since we only scale by diagonal matrices. The second term also has the same bandwidth by the same argument involving shift matrices that was applied for the α case. The third term is $\underline{\beta}$ shifted to the right, scaled twice, then shifted to the right again. Similarly, the last term is $\underline{\beta}$ shifted down, scaled twice, and shifted down again. The third term and fourth term taken together thus increase the bandwidth of $\int_{-1}^{1} \bar{N}\beta\left(x\right)\bar{N}^T dx$ by 2, meaning its bandwidth is $K_\beta + 2$. Again, when $K_\alpha = K_\beta = 0$, these results agree with the constant α and β results leading to a diagonal term $\alpha\underline{S}_{DL}\underline{S}_{DL}^T$ and a pentadiagonal term $\beta\tilde{\underline{S}}\tilde{\underline{S}}^T$. When we consider the sparsity of $\underline{A}$ as a whole, we have a banded matrix with bandwidth $K_{\underline{A}} = \max\left(K_\alpha,\, K_\beta + 2\right)$. Of course, if $K_{\underline{A}}$ is equal to or larger than the degree L then the matrix $\underline{A}$ is full.

Unfortunately, for arbitrary α and β it may not be possible to have $K_{\underline{A}}$ small. This will depend on the number of Legendre coefficients required to represent α and β on the interval

$(-1,1)$ to the user specified tolerance. One possible approach to avoid large $K_{\underline{A}}$ would be to limit the number of terms K_α and K_β used to represent α and β in the integrals. This would lead to an inaccurate $\underline{A}$, and inaccurate solution ϕ, however, the solution of the associated saddle point system will be faster as the bandwidth will be smaller. In this approach, we trade accuracy for speed. To avoid this tradeoff, we propose a finite element method using the integrated Legendre polynomial basis functions on subdomains which preserves a user specified bandwidth while maintaining high accuracy.

4.4 A Sparsity-Aware Finite Element Method

Before we describe how to construct a fast and accurate solver using the methods discussed in this chapter, let us describe a finite element method for the generalized domain $\Omega = (a,b)$. Until now, we have taken our problem domain to be $\Omega = (-1,1)$ to simplify our treatment. We now make the transition to the more general problem

$$-\frac{d}{dx}\left(\alpha(x)\frac{d\phi}{dx}\right) + \beta(x)\phi(x) = f(x), \qquad x \in (a,b), \tag{4.117}$$

$$\phi(a) = p, \tag{4.118}$$

$$\left[\alpha\frac{d\phi}{dx} + \gamma\phi\right]_{x=b} = q. \tag{4.119}$$

The simplified treatment on the canonical domain $(-1,1)$ will be used for this more general problem. In fact, as we did in the classical finite element method described in Chapter 3, we will define basis functions on disjoint subintervals of (a,b) which are shifted and scaled versions of those found on the canonical domain $(-1,1)$. To do so, we partition Ω such that $a = x_0 < x_1 < x_2 < \ldots < x_N = b$. If we map the interval (x_{j-1}, x_j) to $(-1,1)$, then we can use our integrals of Legendre polynomial construction to build basis functions for a degree L_j polynomial basis with support on (x_{j-1}, x_j). Thus, the global basis functions will be

$$\bar{N}_j(x) = \begin{cases} \tilde{\underline{S}}\bar{p}\left(\dfrac{(x - x_{j-1}) - (x_j - x)}{x_j - x_{j-1}}\right) & x \in (x_{j-1}, x_j) \\ 0 & \text{otherwise} \end{cases} \tag{4.120}$$

with $\bar{N} = \begin{bmatrix} \bar{N}_1^T & \bar{N}_2^T & \ldots & \bar{N}_N^T \end{bmatrix}^T$ for a total of $M = \sum_{j=1}^{N}(L_j + 1)$ basis functions. That is, we allow each subinterval to have a polynomial degree L_j basis independent of all other subintervals (making sure that $L_j \geq 1$). By choosing this basis, our approximate solution will be given by $\phi = \bar{N}^T\bar{\phi}$ where $\bar{\phi} = \begin{bmatrix} \bar{\phi}_1^T & \bar{\phi}_2^T & \ldots & \bar{\phi}_N^T \end{bmatrix}^T$.

We now verify the structure of the operator matrix $\underline{A}$. Recall from our discussion of classical finite element methods that by defining basis functions on disjoint subintervals, we recover a block diagonal matrix

$$\underline{A} = \begin{bmatrix} \underline{A}_1 & & & \\ & \underline{A}_2 & & \\ & & \ddots & \\ & & & \underline{A}_N \end{bmatrix}. \tag{4.121}$$

To build each block $\underline{A}_j$, we must compute

$$\underline{A}_j = \int_{x_{j-1}}^{x_j} \left[\frac{d}{dx}\bar{N}_j\right] \alpha(x) \left[\frac{d}{dx}\bar{N}_j\right]^T dx + \int_{x_{j-1}}^{x_j} \bar{N}_j \beta(x) \bar{N}_j^T dx \tag{4.122}$$

and will need to add the term

$$\gamma \left[\bar{N}_j \bar{N}_j^T\right]_{x=b} \tag{4.123}$$

when $j = N$. To evaluate the entries of this matrix, we change variables to the canonical domain using

$$x = \underbrace{\frac{x_j - x_{j-1}}{2}}_{\Xi_j} u + \underbrace{\frac{x_j + x_{j-1}}{2}}_{\xi_j} \tag{4.124}$$

where $u \in (-1, 1)$, $dx = |\Xi_j| \, du$, and, by the chain rule, $\frac{d}{dx}\bar{N}_j = \Xi_j^{-1} \frac{d}{du}\bar{N}_j$. Then

$$\underline{A}_j = \int_{-1}^{1} \left[\Xi_j^{-1} \frac{d}{du}\bar{N}_j\right] \alpha(\Xi_j u + \xi_j) \left[\Xi_j^{-1} \frac{d}{du}\bar{N}_j\right]^T |\Xi_j| \, du + \int_{-1}^{1} \bar{N}_j \beta(\Xi_j u + \xi_j) \bar{N}_j^T |\Xi_j| \, du \tag{4.125}$$

$$= |\Xi_j| \, \Xi_j^{-2} \int_{-1}^{1} \left[\frac{d}{du}\bar{N}_j\right] \alpha(\Xi_j u + \xi_j) \left[\frac{d}{du}\bar{N}_j\right]^T du + |\Xi_j| \int_{-1}^{1} \bar{N}_j \beta(\Xi_j u + \xi_j) \bar{N}_j^T du, \tag{4.126}$$

and $\alpha(\Xi_j u + \xi_j)$ and $\beta(\Xi_j u + \xi_j)$ are approximated by a linear combination of Legendre polynomials on the canonical domain. Since $\bar{N}_j = \tilde{\underline{S}}\bar{p}(u)$ on the interval $(-1, 1)$, we can reuse our previous sparsity results from Section 4.3 directly, yielding

$$\underline{A}_j = |\Xi_j| \, \Xi_j^{-2} \underline{S}_{DL} \left[\sum_{k=0}^{K_{\alpha_j}} \alpha_k^{(j)} \underline{T}_k\right] \underline{S}_{DL}^T + |\Xi_j| \, \tilde{\underline{S}} \left[\sum_{k=0}^{K_{\beta_j}} \beta_k^{(j)} \underline{T}_k\right] \tilde{\underline{S}}^T \tag{4.127}$$

where K_{α_j} is the number of Legendre coefficients required to approximate α on the interval

$(x_{j-1,}x_j)$ and $\alpha_k^{(j)}$ are the corresponding coefficients. Similarly, for approximating β on the same interval we have K_{β_j} terms with coefficients $\beta_k^{(j)}$. For the case $j = N$, we add

$$\gamma \left[\bar{N}_j \bar{N}_j^T\right]_{x=b} = \gamma \left[\bar{N}_N \bar{N}_N^T\right]_{u=1} \tag{4.128}$$

$$= \gamma \frac{1}{2} \left(\bar{e}_1 + \bar{e}_2\right) \left(\bar{e}_1 + \bar{e}_2\right)^T \tag{4.129}$$

to $\underline{A}_N$.

For the forcing vector, we have $\bar{b} = \left[\begin{array}{cccc} \bar{b}_1^T & \bar{b}_2^T & \ldots & \bar{b}_N^T \end{array}\right]^T$ with

$$\bar{b}_j = \int_{x_{j-1}}^{x_j} \bar{N}_j f \, dx \tag{4.130}$$

and, for the case $j = N$, we subtract

$$\left[q\bar{N}_j\right]_{x=b}. \tag{4.131}$$

Thus, using the same transformation to the canonical domain, we obtain

$$\bar{b}_j = \int_{-1}^{1} \bar{N}_j f \left(\Xi_j u + \xi_j\right) \left|\Xi_j\right| du \tag{4.132}$$

$$= \left|\Xi_j\right| \int_{-1}^{1} \bar{N}_j f \left(\Xi_j u + \xi_j\right) du \tag{4.133}$$

where we approximate $f\left(\Xi_j u + \xi_j\right)$ on the interval (x_{j-1}, x_j) using a Legendre expansion whose coefficients we place in a vector $\bar{f}_j$ so that

$$\bar{b}_j = \left|\Xi_j\right| \underline{\tilde{S}} \bar{f}_j. \tag{4.134}$$

We subtract

$$\left[q\bar{N}_j\right]_{x=b} = q \left[\bar{N}_N\right]_{u=1} \tag{4.135}$$

$$= q \frac{1}{\sqrt{2}} \left(\bar{e}_1 + \bar{e}_2\right) \tag{4.136}$$

from $\bar{b}_j$ when $j = N$.

Finally, we must ensure that inter-element continuity is enforced, as well as the Dirichlet boundary condition. We do this via constraint equations. First, for the Dirichlet boundary

condition, we obtain

$$\phi\left(a\right) = p \tag{4.137}$$

$$\bar{N}\left(a\right)^T \bar{\phi} = p \tag{4.138}$$

$$\bar{N}_1\left(a\right)^T \bar{\phi}_1 = p \tag{4.139}$$

since all other basis functions besides those in $\bar{N}_1$ are zero at $x = a$ due to their disjoint nature. Since $\bar{N}_1\left(a\right) = \underline{\tilde{S}}\bar{p}\left(-1\right)$, we obtain

$$\frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right)^T \bar{\phi}_1 = p \tag{4.140}$$

$$\left[\begin{array}{cccc} \frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right)^T & 0 & \ldots & 0 \end{array}\right] \bar{\phi} = p \tag{4.141}$$

for our constraint equation imposing the Dirichlet boundary condition.

To impose inter-element continuity, let us consider the case of imposing continuity at x_j. Then

$$\phi(x_j^-) = \phi(x_j^+) \tag{4.142}$$

$$\bar{N}(x_j^-)^T\bar{\phi} = \bar{N}(x_j^+)^T\bar{\phi} \tag{4.143}$$

$$\bar{N}_j(x_j^-)^T\bar{\phi}_j = \bar{N}_{j+1}(x_j^+)^T\bar{\phi}_{j+1} \tag{4.144}$$

since only the intervals (x_{j-1}, x_j) and (x_j, x_{j+1}) have the limit points x_j^- and x_j^+ respectively. We know that $\bar{N}_j(x_j^-) = \underline{\tilde{S}}\bar{p}\left(1\right)$ and $\bar{N}_{j+1}(x_j^+) = \underline{\tilde{S}}\bar{p}\left(-1\right)$ so that

$$\bar{N}_j(x_j^-)^T\bar{\phi}_j - \bar{N}_{j+1}(x_j^+)^T\bar{\phi}_{j+1} = 0 \tag{4.145}$$

$$\frac{1}{\sqrt{2}}\left(\bar{e}_1 + \bar{e}_2\right)^T \bar{\phi}_j - \frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right)^T \bar{\phi}_{j+1} = 0 \tag{4.146}$$

$$\left[\begin{array}{ccccccc} 0 & \ldots & 0 & \frac{1}{\sqrt{2}}\left(\bar{e}_1 + \bar{e}_2\right)^T & -\frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right)^T & 0 & \ldots \quad 0 \end{array}\right] \bar{\phi} = 0 \tag{4.147}$$

which gives us one of the inter-element continuity conditions. There are a total of $N - 1$ such equations, each shifted blockwise for continuity at x_j with $j = 1, 2, ..., N - 1$. Note that the lengths of $\bar{e}_1$ and $\bar{e}_2$ depend on the polynomial degree of element j. As an example, consider the case where all $L_j = 2$. Then the constraint matrix $\underline{C}$ and constraint right hand

side vector $\bar{d}$ are given by

$$\underline{C} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & & & & & & & & & & & & & \\ 1 & & 1 & 0 & -1 & 1 & 0 & & & & & & & & & \\ & & & & 1 & 1 & 0 & -1 & 1 & 0 & & & & & & \\ & & & & & & & & \ddots & \ddots & \ddots & & & & & \\ & & & & & & & & & 1 & 1 & 0 & -1 & 1 & 0 & \\ & & & & & & & & & & & & 1 & 1 & 0 & -1 & 1 & 0 \end{bmatrix}, \quad \bar{d} = \begin{bmatrix} p \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{4.148}$$

where the first row corresponds to the Dirichlet boundary condition, and the remaining rows are inter-element continuity conditions. To construct a sparse basis for the null space of $\underline{C}$, denoted by $\underline{Z}$, we remark that, by the rank-nullity theorem, the number of columns in $\underline{C}$ (which is $\sum_{j=1}^{N}(L_j+1)$) must be equal to the rank of $\underline{C}$ (which is N) plus the nullity of $\underline{C}$. This means that

$$\begin{aligned} \text{null}\,(\underline{C}) &= \sum_{j=1}^{N}(L_j+1) - \text{rank}\,(\underline{C}) && (4.149)\\ &= \sum_{j=1}^{N} L_j + N - N && (4.150)\\ &= \sum_{j=1}^{N} L_j. && (4.151) \end{aligned}$$

Observe that there are

$$\sum_{j=1}^{N}[(L_j+1)-2] = \sum_{j=1}^{N} L_j - N \tag{4.152}$$

columns of $\underline{C}$ that are zero vectors, meaning that we can take unit vectors with a 1 in the row equal to the column index of each zero column in $\underline{C}$ as vectors in our basis for $\underline{Z}$. This leaves us to find an additional N columns for $\underline{Z}$ that form a linearly independent set of vectors when appended to the unit vectors we have already found. $N-1$ such vectors can be found by taking the $N-1$ last rows of $\underline{C}$ with the negative sign in the second pair of nonzeros interchanged. The final vector corresponds to the vector with ones only in the antepenultimate and penultimate rows, with zeros elsewhere. Again, for example, when all

$L_j = 2$, the resulting sparse basis for the null space is

$$\underline{Z} = \begin{bmatrix}
0 & & & & & 1 & & & & & \\
0 & & & & & 1 & & & & & \\
1 & & & & & 0 & & & & & \\
0 & 0 & & & & 1 & 1 & & & & \\
0 & 0 & & & & -1 & 1 & & & & \\
0 & 1 & & & & 0 & 0 & & & & \\
 & 0 & \ddots & & & & 1 & 1 & & & \\
 & 0 & \ddots & & & & -1 & 1 & \ddots & & \\
 & 0 & \ddots & 0 & & & 0 & 0 & \ddots & 1 & \\
 & & \ddots & 0 & & & & 1 & \ddots & 1 & \\
 & & & 1 & & & & -1 & \ddots & 0 & \\
 & & & 0 & 0 & & & 0 & \ddots & 1 & 1 \\
 & & & 0 & 0 & & & & \ddots & -1 & 1 \\
 & & & 0 & 1 & & & & & 0 & 0
\end{bmatrix}. \tag{4.153}$$

We can rearrange the three types of columns in $\underline{Z}$ to obtain

$$\underline{Z} = \begin{bmatrix}
1 & 0 & & & & & & & & \\
1 & 0 & & & & & & & & \\
0 & 1 & & & & & & & & \\
1 & 0 & 1 & 0 & & & & & & \\
-1 & 0 & 1 & 0 & & & & & & \\
0 & 0 & 0 & 1 & & & & & & \\
 & & 1 & 0 & 1 & & & & & \\
 & & -1 & 0 & 1 & \ddots & & & & \\
 & & 0 & 0 & 0 & \ddots & 1 & 0 & & \\
 & & & & 1 & \ddots & 1 & 0 & & \\
 & & & & -1 & \ddots & 0 & 1 & & \\
 & & & & 0 & \ddots & 1 & 0 & 1 & 0 \\
 & & & & & & -1 & 0 & 1 & 0 \\
 & & & & & & 0 & 0 & 0 & 1
\end{bmatrix}. \tag{4.154}$$

If the L_j are arbitrary, the only changes to $\underline{C}$ are to add additional zero columns, meaning that $\underline{Z}$ needs to contain additional unit vectors (and the vectors with four nonzero entries need to have their entries shifted). One can verify that this construction always yields $\underline{C}\,\underline{Z} = 0$.

Now that we can use the integrated Legendre polynomials on disjoint intervals to solve the finite element problem, we describe the last step towards a fast and accurate finite element solver: a sparsity-aware adaptive step. The main issue with our method up until now has been that the sparsity of each block $\underline{A}_j$ depends on the number of terms in the Legendre expansions associated with representing α and β. To circumvent this issue, we begin with a coarse partition of $\Omega = (a, b)$ and a fixed bandwidth limit for $\underline{A}$ given by $K_{\text{tol}} \geq 2$ (recall that 2 is the lower limit since $K_{\underline{A}_j} = \max(K_{\alpha_j}, K_{\beta_j} + 2)$). This maximum bandwidth indicates that K_{α_j} can be no larger than K_{tol} and that K_{β_j} can be no larger than $K_{\text{tol}} - 2$. We then compute the Legendre coefficients in the expansions to approximate α and β on each subinterval. If the coefficients have not decayed to the user specified tolerance, rather than truncate the Legendre expansion and compute with a potentially insufficient approximation to $\underline{A}_j$, we instead subdivide the subinterval and repeat this process until the sparsity requirement on each subinterval is met[6]. This process can be viewed as an initial h-adaptive step for our finite element solver. Note that this process terminates when α and β are continuous. For this reason, it is imperative to create a starting partition which takes into account discontinuities in α and β (so that element boundaries coincide with discontinuities in α and β).

Once the bandwidth tolerance has been met, the sparsity of $\underline{A}$ is guaranteed to be K_{tol}, regardless of the polynomial degree on each element. This means we are now able to adopt an approach similar to the one described in Section 4.2. That is, on each subinterval, we compute the Legendre coefficients for the forcing function f to the user specified tolerance. This gives us a starting polynomial degree L_j on each subinterval. We then build the $\underline{A}_j$, $\bar{b}_j$, $\underline{C}$, and $\bar{d}$ matrices and vectors and solve the associated saddle point system, either directly using a sparse direct solver, or using the null space method with our sparse basis for the null space $\underline{Z}$. We then change basis to the Legendre coefficients on each element and verify if they have decayed to the user specified tolerance. We do this by computing element-wise error indicators which are, in this thesis, the sum of the magnitude squared of the final two Legendre coefficients for each element.

Unlike Section 4.2, now that we allow multiple elements, we have a choice between p-refinement and h-refinement for each element, and we may even choose not to refine certain elements. The hp-adaptive scheme that we will use to make these decisions is closely related

[6] I bisect the interval but there is room for potential improvement here which I have not explored.

to the one described in [125]. The difference is only in the way that the error indicators are computed. In [125], they compute the more complicated, but provably reliable and efficient indicator based on the ideas of [126]. Their element-wise indicator is a weighted integral of the residual of (4.117) over the element interval (x_{j-1}, x_j) divided by $\alpha L_j(L_j+1)$. Strictly speaking, their indicator is only appropriate for constant α. In practice, we have used this indicator as well as the simpler one based on magnitudes of Legendre coefficients and have found both to perform similarly. The refinement step procedes as follows. Once the error indicators η_j are computed for each element j, we determine the largest error indicator $\eta_{\max}$ and refine all elements whose indicator is larger than $\zeta\eta_{\max}$ where $\zeta < 1$ is a user specified threshold. In our examples, we choose $\zeta = 3/4$. For each of those elements chosen for refinement, the decision to perform h- or p-refinement is based on estimating the rate of decay of its Legendre coefficients computed thus far.

From approximation theory [127], if ϕ is analytic throughout the interior of the Bernstein ellipse (an ellipse centered at the origin with major axis of length a along the real axis and minor axis of length b along the imaginary axis), then ϕ can be written as a Legendre series that converges absolutely and uniformly on any closed set inside the ellipse. Moreover, the coefficients ϕ_n in the series satisfy

$$\limsup_{n\to\infty} |\phi_n|^{1/n} = \frac{1}{r} \tag{4.155}$$

where $r = a + b$. When r is large (that is, when ϕ is analytic in a large ellipse), this statement shows that the Legendre coefficients decay rapidly. The smallest possible ellipse covering real interval $(-1, 1)$ has $r = 1$ since $a = 1$ and $b = 0$. In cases where r is close to one, the magnitudes of the coefficients in the series decay slowly. This means that we should favor p-refinement when r is large and h-refinement when r is small. If we set $\theta = 1/r$, then θ near zero suggests using p-refinement and θ near one suggests using h-refinement. Of course, the statement regarding decay of coefficients assumes that we have knowledge of all coefficients. Instead, we compute a least squares fit to determine θ using only those coefficients $\bar{\phi}_{\text{Leg}}$ that have been computed thus far. From (4.155), we have

$$|\phi_{\text{Leg},n}|^{1/n} \approx \theta \tag{4.156}$$

$$\frac{1}{n}\log|\phi_{\text{Leg},n}| \approx \log\theta \tag{4.157}$$

$$\log|\phi_{\text{Leg},n}| \approx n\underbrace{\log\theta}_{a_1}. \tag{4.158}$$

We then estimate a_1 by producing a linear least squares fit $\log|\phi_{\text{Leg},n}| \approx na_1 + a_2$ using

$$\underline{W} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ \vdots & \vdots \\ L_j & 1 \end{bmatrix}, \qquad \bar{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \qquad \bar{y} = \log|\bar{\phi}_{\text{Leg}}|,$$

where $\log|\cdot|$ is meant to be applied entrywise to $\bar{\phi}_{\text{Leg}}$. Then $\underline{W}\bar{a} \approx \bar{y}$ and we solve this over-determined system using QR factorization. Once a_1 is estimated, we compute $\theta = e^{a_1}$ and choose p-refinement if θ is less than a user specified threshold $0 < \theta_t < 1$. Otherwise, h-refinement is performed. In our examples, we choose $\theta_t = 3/4$. This process is applied independently to all elements that have been marked for refinement.

Elements undergoing p-refinement have their element degree L_j doubled, while elements undergoing h-refinement are bisected. We then update the matrices and vectors $\underline{A}_j$, $\bar{b}_j$, $\underline{C}$, and $\bar{d}$, and compute a new solution. Once the Legendre expansions for all elements in the mesh have decayed below the user specified tolerance, we terminate the algorithm. In other words, the algorithm terminates once all element-wise error indicators are sufficiently small.

4.5 Numerical Results

To verify that the adaptive algorithm works as intended, we use the example

$$-\epsilon\frac{d^2}{dx^2}\phi + \phi = 1, \quad x \in (0,1), \tag{4.159}$$

$$\phi(0) = 0, \tag{4.160}$$

$$\phi(1) = 0, \tag{4.161}$$

taken from [128] where ϵ is a parameter[7]. In terms of our general framework defined in Section 4.1, this corresponds to $\alpha = \epsilon$, $\beta = 1$, $f = 1$, and $p_1 = p_2 = 0$ for the two distinct Dirichlet boundary conditions. There are no Robin boundary conditions and thus we need not define γ or q. This fundamental BVP can be solved analytically to find that

$$\phi_{\text{exact}}(x) = \frac{e^{-1/\sqrt{\epsilon}} - 1}{e^{1/\sqrt{\epsilon}} - e^{-1/\sqrt{\epsilon}}} e^{x/\sqrt{\epsilon}} + \frac{1 - e^{1/\sqrt{\epsilon}}}{e^{1/\sqrt{\epsilon}} - e^{-1/\sqrt{\epsilon}}} e^{-x/\sqrt{\epsilon}} + 1 \tag{4.162}$$

[7] This BVP is a simplification of the screened Poisson equation which arises when considering electric field screening effects.

which is of particular interest when $\epsilon \ll 1$ since ϕ exhibits boundary layers which can be difficult to resolve with lower accuracy finite element codes. We take $\epsilon = 10^{-5}$ in our tests. Our starting discretization consists of three equal sized elements spanning the interval $(0, 1)$. The maximum bandwidth is set by $K_{\text{tol}} = 2$ with the user specified accuracy tolerance set to 10^{-12}. All computations are done in double precision using sparse matrices. The saddle point system, whose typical sparsity pattern is shown in Figure 4.5, is solved using a sparse direct solver. Note that the first step in the algorithm is to compute the Legendre coefficients in the expansions for α and β. Since both are constant, we expect the algorithm to identify K_{α_j} and K_{β_j} as zero on all starting elements. As such, we expect the initialization step of h-adaption to leave our starting partition unchanged. See the last plot in Figure 4.6 for the solution ϕ computed to the desired tolerance. Circles indicate where elements begin and end. Since the starting mesh consists of three elements, we can see from this figure that h-adapation was performed twice near both boundary layers, and that otherwise p-adaption was applied to resolve the behavior of the solution.

To quantitatively judge the accuracy of our algorithm, we consider three measures of error. The first is measured in the operator norm given by

$$\|\mathcal{E}\|_E^2 = \int_0^1 \epsilon \left(\frac{d\mathcal{E}}{dx}\right)^2 + \mathcal{E}^2 \, dx, \tag{4.163}$$

where E is meant to remind us that this norm is associated with the energy related to the functional. We take $\mathcal{E} = \phi - \phi_{\text{exact}}$ so that our error measure is $\|\phi - \phi_{\text{exact}}\|_E$. The second error measure is computed in the standard L^2 norm given by

$$\|\mathcal{E}\|_2^2 = \int_0^1 \mathcal{E}^2 \, dx \tag{4.164}$$

and our error measure is $\|\phi - \phi_{\text{exact}}\|_2$. Finally, our third error measure is obtained by computing the difference in the value of the functional evaluated at the exact solution from that evaluated at our computed solution. Recall that the functional is given by

$$F(\phi) = \frac{1}{2} \int_0^1 \epsilon \left(\frac{d\phi}{dx}\right)^2 + \phi^2 dx - \int_0^1 \phi \, dx \tag{4.165}$$

so that our error measure is $|F(\phi) - F(\phi_{\text{exact}})|$. Note that in order to compute the first two error measures, we can restrict them to element subintervals, and sum each element contribution to the norm squared. Once all contributions have been added, we take the square root to obtain the norm. This is possible due to the integral definition of each norm which can be broken up into integrals on each subinterval. The first three plots in Figure 4.6

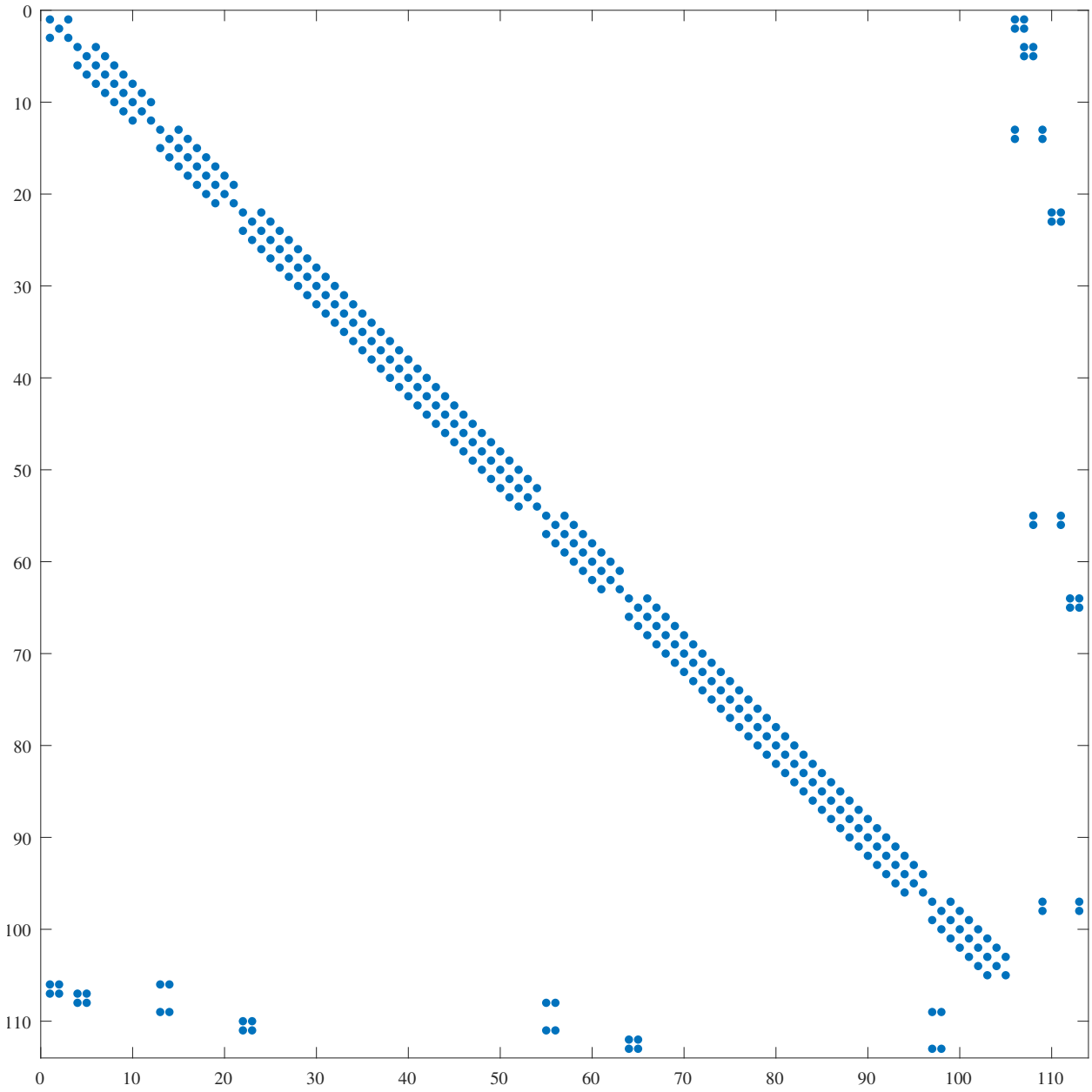

Figure 4.5: Sparsity pattern for the saddle point system which computes ϕ to 14 digits of accuracy for the boundary layer problem. Note that the bandwidth of the operator matrix $\underline{A}$ is restricted to $K_{\text{tol}} = 2$, as required. The constraint matrix $\underline{C}$ is not structured as in (4.148) because the ordering of the elements is not increasing from left to right on the interval. This is a consequence of the mesh generation algorithm used, which was coded in anticipation of two- and three-dimensional problems. We reserve comment on the specific algorithms used for mesh generation for the sake of brevity. Note that the number of entries in the saddle point matrix is $113^2 = 12,769$ whereas there are only 343 nonzero entries.

Table 4.1: Error for uniform h-refinement applied to the boundary layer problem.

Element degree	Error measured in the operator norm	Degrees of freedom
1	3.1×10^{-2}	192
2	2.4×10^{-2}	144
4	2.1×10^{-2}	120
8	5.9×10^{-4}	216

illustrate the error measures, each as a function of the number of degrees of freedom used at each step in the adaptive algorithm. We note that the error measures all decrease at each adaptive step since we only ever add degrees of freedom in our algorithm, retaining the full modeling flexibility of all previous discretizations.

For comparison purposes, we solved the same problem using a fixed polynomial degree for all elements in the mesh, and performed uniform h-refinements on all elements in the mesh at each h-adaption step until the total number of degrees of freedom exceeded 105 (the final number required by the fully adaptive algorithm). This type of uniform h-refinement is characteristic of classical finite element methods. We report the operator norm error for various choices of polynomial degree in Table 4.1. Notice that none of the uniform refinement methods approach the small operator norm error (approximately 10^{-14}) of the fully adaptive method, even when their numbers of degrees of freedom exceed the number required for the fully adaptive method to converge.

As a second example, we consider a unit radius sphere of uniform charge density ρ_0, which is centered on the origin and embedded in free space. The objective is to determine the electrostatic potential ϕ for this system. We solve Gauss' law

$$\nabla \cdot \bar{E} = \begin{cases} \dfrac{\rho_0}{\epsilon_0} & R \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.166}$$

where $\bar{E} = -\nabla\phi$ is the electric field intensity, ϵ_0 is the permittivity of free space, and R is the radial distance from the origin in spherical coordinates. Since the geometry of our problem is spherically symmetric, we know that the field and potential also vary only as functions of R so that $\phi(R)$ and $\bar{E} = E_R(R)\,\bar{a}_R$ where $\bar{a}_R$ is the radially outward pointing unit vector in spherical coordinates. Using the integral form of Gauss' law, we find that

$$\bar{E} = \begin{cases} \dfrac{\rho_0}{\epsilon_0}\dfrac{R}{3}\bar{a}_R & R \leq 1 \\ \dfrac{\rho_0}{\epsilon_0}\dfrac{1}{3R^2}\bar{a}_R & \text{otherwise} \end{cases} \tag{4.167}$$

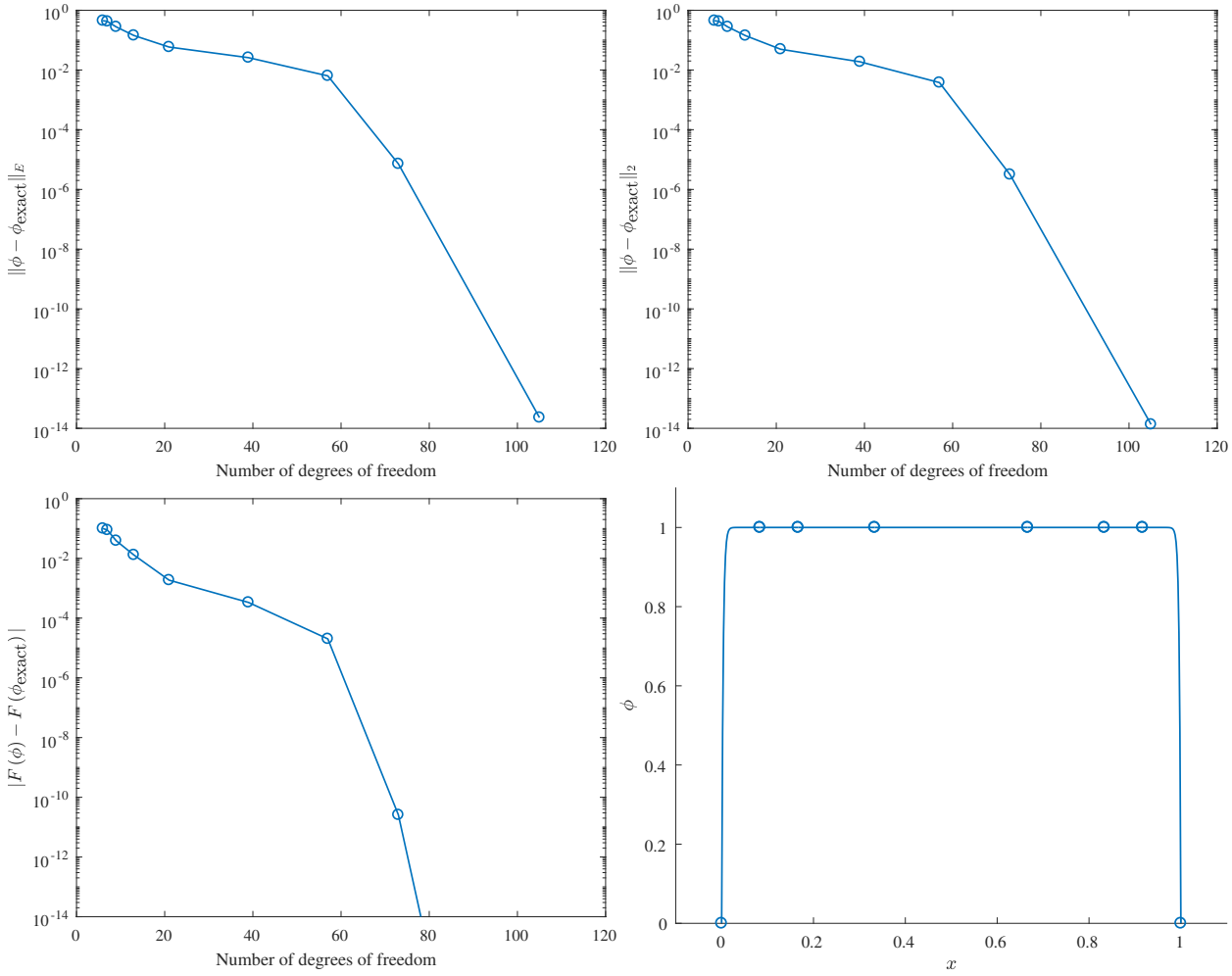

Figure 4.6: Comparison of various measures of error in the computed solution as a function of the total number of degrees of freedom, as well as a plot of the final computed solution for the boundary layer problem. In order (reading left to right, from top to bottom), we have the error in the operator norm, the error in the L^2 norm, the absolute difference in the value of the functional, and a plot of the computed solution using the adaptive finite element code for the final number of degrees of freedom required to have all Legendre expansions decay below the user specified tolerance. The error plots were obtained by computing the error measures at the conclusion of each adaptive step of the finite element method.

and that

$$\phi = \begin{cases} \frac{\rho_0}{\epsilon_0}\left(\frac{1}{2} - \frac{R^2}{6}\right) & R \leq 1 \\ \frac{\rho_0}{\epsilon_0}\frac{1}{3R} & \text{otherwise} \end{cases} \tag{4.168}$$

when a reference potential of zero as R approaches infinity is chosen.

We can use the finite element method to compute this same solution over a finite interval $R \in (0, R_{\text{far}})$ where R_{far} is some positive real number larger than 1. Using the gradient and divergence in spherical coordinates, and the fact that ϕ depends only on R, Gauss' law in diffierential form simplifies to

$$-\frac{d}{dR}\left(R^2\frac{d\phi}{dR}\right) = \begin{cases} \frac{\rho_0}{\epsilon_0}R^2 & R \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.169}$$

This fits into our general framework with $\alpha = R^2$, $\beta = 0$,

$$f = \begin{cases} \frac{\rho_0}{\epsilon_0}R^2 & R \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4.170}$$

and $x = R$. We complete the specification of the BVP by describing boundary conditions on ϕ at the two endpoints of the interval. First, through a symmetry argument at the origin, we know that a homogeneous Neumann condition must hold. We can show that such a condition holds for the solution (4.168) by computing the normal flux $-\alpha\frac{\partial\phi}{\partial R}$ at $R = 0$. This gives

$$\left[-\alpha\frac{\partial\phi}{\partial R}\right]_{R=0} = \left[-R^2\frac{\rho_0}{\epsilon_0}\left(-\frac{R}{3}\right)\right]_{R=0} \tag{4.171}$$

$$= \left[\frac{\rho_0}{\epsilon_0}\frac{R^3}{3}\right]_{R=0} \tag{4.172}$$

$$= 0. \tag{4.173}$$

Similarly, computing the normal flux $\alpha\frac{\partial\phi}{\partial R}$ at $R = R_{\text{far}}$ yields

$$\left[\alpha\frac{\partial\phi}{\partial R}\right]_{R=R_{\text{far}}} = \left[R^2\frac{\rho_0}{\epsilon_0}\left(-\frac{1}{3R^2}\right)\right]_{R=R_{\text{far}}} \tag{4.174}$$

$$= \left[-R\frac{\rho_0}{\epsilon_0}\left(\frac{1}{3R}\right)\right]_{R=R_{\text{far}}}. \tag{4.175}$$

Table 4.2: Error for uniform h-refinement applied to the sphere of charge problem.

Element degree	Error measured in the operator norm	Degrees of freedom
1	6.9×10^{-3}	176
2	5.2×10^{-4}	120
4	9.4×10^{-6}	100
8	2.3×10^{-8}	90

Rather than cancel the factors of R, we have written the last statement such that we have $-R\phi$ on the right hand side i.e. (4.175) is $-R\phi$ for $R_{\text{far}} > 1$ and ϕ given by (4.168). Then

$$\alpha \frac{\partial \phi}{\partial R} + R\phi(R) = 0$$

holds at $R = R_{\text{far}}$, and we can use a homogeneous Robin boundary condition with $\gamma = R_{\text{far}}$ and $q = 0$ to complete the description of the BVP. Alternatively, we could have used an inhomogeneous Neumann boundary condition with $\gamma = 0$ and $q = -\rho_0/(3\epsilon_0)$.

Using our adaptive algorithm, we compute ϕ with $R_{\text{far}} = 5$. We choose $\rho_0 = \epsilon_0$ so as to avoid computations with the permittivity of free space. This can be viewed as a scaling of R, ϕ, and ρ_0 which makes quantities unitless. We use a starting partition with five elements each of length 1 with all other parameters set as in the previous boundary layer example. Including the point $R = 1$ as a vertex in the mesh is important, otherwise Legendre expansions of the forcing function f will require many terms to adequately model the discontinuity. Figure 4.7 shows the three error measures computed at each iteration of the adaptive algorithm, as well as the final computed solution ϕ, along with the post-processed electric field component E_R (obtained by taking the negative derivative of ϕ with respect to R). The figure does not show the exact solution as computed via the integral form of Gauss' law because the finite element solution is accurate to machine precision and indistinguishable from the exact solution.

Again, we compare the adaptive algorithm to a uniform h-refinement algorithm which we terminate once the number of degrees of freedom have exceeded 89 (the number required to compute the adaptive solution to machine precision). Table 4.2 shows the error measure in the operator norm for four different element degrees. Here, uniform h-refinement performs better than it did for the boundary layer problem, but is still far from the level of accuracy produced by the adaptive algorithm.

Finally, Figure 4.8 shows the sparsity pattern for the saddle point system used to compute the final adaptive solution. Note that the bandwidth of each block is 2 since α can be exactly represented by a degree 2 Legendre expansion. There are five elements, and correspondingly five blocks visible in the operator portion of the saddle point system. By examining the

sparsity patterns of the blocks, we can deduce which block corresponds to which element. The first block corresponds to the leftmost element because the exact solution is a quadratic polynomial over this element, and the solver has identified that a degree 4 polynomial is sufficient to represent the solution here[8]. The second block corresponds to the rightmost element because in its top corner, we see the contribution of the Robin boundary term. The remaining three elements appear from right to left in the mesh since the polynomial degree required to resolve the $1/R$ behavior of the solution is larger closer to the origin than away from it.

[8]This is minimal since the solver does not know that the degree three and four coefficients in the expansion are zero to machine precision until it computes them. In practice, one could truncate the expansion after making such an observation, but I have not done so in my implementation.

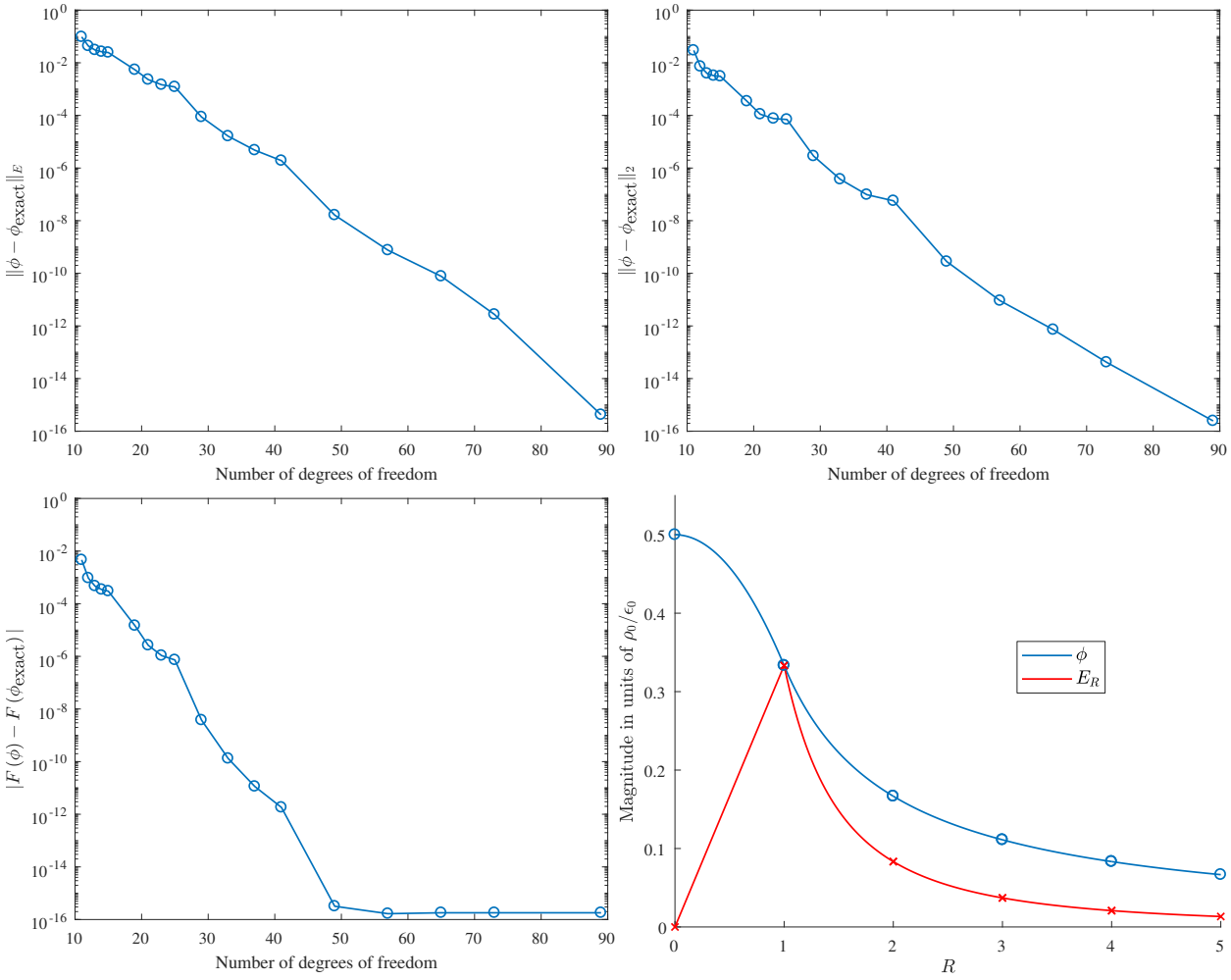

Figure 4.7: Comparison of various measures of error in the computed solution as a function of the total number of degrees of freedom, as well as a plot of the final computed solution for the uniform sphere of charge problem. In order (reading left to right, from top to bottom), we have the error in the operator norm, the error in the L_2 norm, the absolute difference in the value of the functional, and a plot of the computed solution using the adaptive finite element code. In the solution figure, the blue curve with circles depicts the electrostatic potential ϕ while the red curve with crosses depicts the radial component of the electric field intensity E_R. Circles and crosses are placed at endpoints of each element.

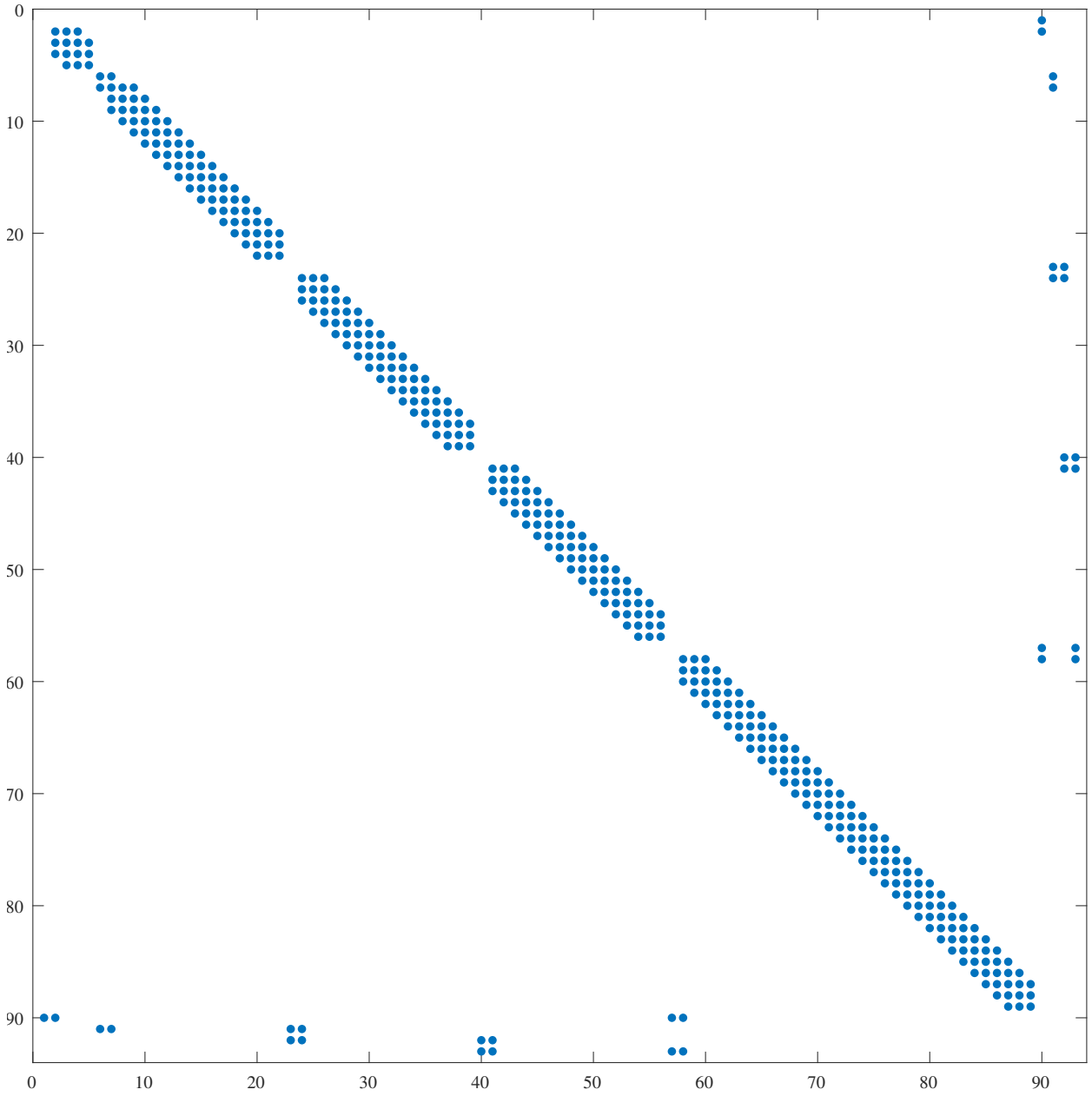

Figure 4.8: Sparsity pattern for the saddle point system used to compute the electrostatic potential for the uniform sphere of charge example. Note that the number of entries in the saddle point matrix is $93^2 = 8,649$ whereas there are only 425 nonzero entries.

Chapter 5

A Space-Time Finite Element Method

Before generalizing the finite element method from Chapter 4 to multiple spatial dimensions, this chapter demonstrates how the techniques introduced throughout Chapter 4 can be applied to time-domain problems. In particular, this chapter considers the wave problem

$$-\frac{\partial}{\partial x}\left(\alpha\left(x\right)\frac{\partial\phi}{\partial x}\right)+\frac{\partial^2\phi}{\partial t^2}=0, \qquad (x,t)\in(a,b)\times(0,T)\,, \tag{5.1}$$

$$\phi\left(a,t\right)=0, \tag{5.2}$$

$$\phi\left(b,t\right)=0, \tag{5.3}$$

$$\phi\left(x,0\right)=\phi_0\left(x\right), \tag{5.4}$$

$$\frac{\partial}{\partial t}\phi\left(x,0\right)=\phi_0'\left(x\right), \tag{5.5}$$

where ϕ is a function of both space and time. Homogeneous Dirichlet boundary conditions in space are taken for simplicity, but this is not necessary. Typically, these problems are approached via semi-discretization [129, 130]. That is, typically, one first discretizes ϕ in space using the techniques described in Chapter 4. This is possible because, ignoring the time derivatives, the PDE (5.1) reduces to the BVP (4.117) with $\beta=1$ and $f=0$. Thus, if ϕ is replaced by a linear combination of basis functions depending only on space, with coefficients depending only on time, a second order system of ODEs for the unknown coefficient vector results. This system is solved using an appropriate time stepping scheme [131, 132].

In what follows, a different approach is taken, discretizing in both space and time using a tensor product of the integrated Legendre basis functions. Since time is often thought of as being orthogonal to space, this is quite natural. The concept of a space-time finite element method to solve second order wave equations is not new [133, 134]. However, this chapter shows how the approach in one spatial dimension can be extended to a space-time framework and demonstrates that the construction remains efficient and accurate. This has

the added benefit of introducing some of the mathematical techniques required for extension to multiple spatial dimensions without the complications of discussing mesh generation and curvilinear elements which will have to be introduced later.

Standard time domain finite element schemes for Maxwell's equations are globally second order methods [12], which means that to compute a space-time solution accurate to a stringent tolerance, the size of time steps must be extremely small. This issue is avoided altogether by permitting basis functions in time of arbitrary order. One potentially unsettling characteristic of such a method is that the space-time solution is obtained by solving a single linear system. As a result, the inherently causal nature of a time stepping scheme (at each time step, the solution depends only on the solution from previous time steps) is lost. This is not an issue if the solution is computed to high accuracy. Another potentially surprising attribute of the method is that boundary conditions are imposed on only three of the four edges of the space-time domain. In fact, two conditions are imposed on a single edge (those corresponding to initial conditions) and the edge corresponding to the final time is left unconstrained. This is different from the boundary treatment of the more typical extension of the BVP (4.117) to a PDE in two spatial dimensions which will be treated in later chapters.

5.1 Space-Time Galerkin Methods

As in the one-dimensional case, let us begin on a single element. In this case, the single domain is a rectangular patch of space-time given by $(a, b) \times (0, T)$. We will need to map this domain to the canonical domain $(-1, 1) \times (-1, 1)$ to reuse results for integrated Legendre polynomials. To do this, we define the affine transformations

$$x = \underbrace{\frac{b-a}{2}}_{\Xi} u + \underbrace{\frac{b+a}{2}}_{\xi} \tag{5.6}$$

and

$$t = \frac{T}{2}\tau + \frac{T}{2} \tag{5.7}$$

where u and τ are space and time variables in the canonical domain. We also expand our solution using a linear combination of basis functions

$$\phi = \bar{\phi}^T \operatorname{vec}\left(\bar{N}(u)\,\bar{N}(\tau)^T\right) \tag{5.8}$$

where the vectorization operator $\text{vec}(\cdot)$ takes a matrix and stacks its columns (from left to right) into a single column vector. Since the outer product $\underline{N}(x,t) = \bar{N}(x)\,\bar{N}(t)^T$ contains all possible combinations of products of basis functions in space and time, so too does the vector $\bar{N}(x,t) = \text{vec}\left(\bar{N}(x)\,\bar{N}(t)^T\right)$. We emphasize that in writing $\bar{N}(x,t)$, we really mean to refer to the vector $\text{vec}\left(\bar{N}(u)\,\bar{N}(\tau)^T\right)$ which can be written in terms of x and t by inverting the maps (5.6) and (5.7). This is the same as how basis functions were defined in Chapter 4. Note that

$$\text{vec}\left(\bar{N}(x)\,\bar{N}(t)^T\right) = \begin{bmatrix} N_0(t)\,\bar{N}(x) \\ N_1(t)\,\bar{N}(x) \\ \vdots \\ N_{L_t}(t)\,\bar{N}(x) \end{bmatrix} \tag{5.9}$$

where L_t is the total degree of polynomials in time. If the total degree of polynomials in space is L_x, then there are $(L_x+1)\times(L_t+1)$ total basis functions.

To manipulate the $\text{vec}(\cdot)$ operator, we introduce the Kronecker product and certain of its properties [25]. The Kronecker product of matrices $\underline{A} \in \mathbb{R}^{m\times n}$ and $\underline{B}$ is given by

$$\underline{A}\otimes\underline{B} = \begin{bmatrix} a_{11}\underline{B} & a_{12}\underline{B} & \dots & a_{1n}\underline{B} \\ a_{21}\underline{B} & a_{22}\underline{B} & \dots & a_{2n}\underline{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\underline{B} & a_{m2}\underline{B} & \dots & a_{mn}\underline{B} \end{bmatrix} \tag{5.10}$$

and satisfies two key properties that we will make extensive use of:

$$(\underline{A}\otimes\underline{B})^T = \underline{A}^T\otimes\underline{B}^T, \tag{5.11}$$

$$(\underline{A}\otimes\underline{B})(\underline{C}\otimes\underline{D}) = (\underline{A}\underline{C})\otimes(\underline{B}\underline{D}), \tag{5.12}$$

where the second property holds only if the matrix products $\underline{A}\underline{C}$ and $\underline{B}\underline{D}$ are valid (the matrices must have appropriate dimensions). The third property we will need relates the $\text{vec}(\cdot)$ operator to the Kronecker product and matrix-vector multiplication via

$$\text{vec}(\underline{A}\underline{B}\underline{C}) = \left(\underline{C}^T\otimes\underline{A}\right)\text{vec}(\underline{B}). \tag{5.13}$$

We now apply these properties to understand how differentiation in time and space affect

basis functions. In particular, we can write

$$\frac{\partial}{\partial x}\phi = \frac{\partial}{\partial x}\left[\bar{\phi}^T \mathrm{vec}\left(\bar{N}\left(u\right)\bar{N}\left(\tau\right)^T\right)\right] \tag{5.14}$$

$$= \Xi^{-1}\frac{\partial}{\partial u}\left[\bar{\phi}^T \mathrm{vec}\left(\tilde{\underline{S}}\bar{p}\left(u\right)\bar{N}\left(\tau\right)^T\right)\right] \tag{5.15}$$

where we have used the chain rule to transform derivatives depending on x to ones depending on u. We then use the linearity of differentiation to obtain

$$\frac{\partial}{\partial x}\phi = \Xi^{-1}\left[\bar{\phi}^T \mathrm{vec}\left(\tilde{\underline{S}}\frac{\partial}{\partial u}\bar{p}\left(u\right)\bar{N}\left(\tau\right)^T\right)\right] \tag{5.16}$$

$$= \Xi^{-1}\bar{\phi}^T \mathrm{vec}\left(\tilde{\underline{S}}\tilde{\underline{D}}\bar{p}\left(u\right)\bar{N}\left(\tau\right)^T\right) \tag{5.17}$$

$$= \Xi^{-1}\bar{\phi}^T \mathrm{vec}\left(\underline{S}_{DL}\bar{p}\left(u\right)\bar{N}\left(\tau\right)^T\right) \tag{5.18}$$

where the second step replaces the differentiation operator with its corresponding differentiation matrix and the third step simplifies the product of the integration and differentiation matrices. We are now in a position to apply property (5.13) to find that

$$\frac{\partial}{\partial x}\phi = \Xi^{-1}\bar{\phi}^T\left(\bar{N}\left(\tau\right)\otimes\underline{S}_{DL}\right)\mathrm{vec}\left(\bar{p}\left(u\right)\right) \tag{5.19}$$

$$= \Xi^{-1}\bar{\phi}^T\left(\bar{N}\left(\tau\right)\otimes\underline{S}_{DL}\right)\bar{p}\left(u\right) \tag{5.20}$$

with the final result obtained using the fact that the $\mathrm{vec}\left(\cdot\right)$ operator maps vectors to vectors, as expected. Similarly, applying the same types of operations, we find that

$$\frac{\partial}{\partial t}\phi = \frac{2}{T}\bar{\phi}^T\left(\underline{S}_{DL}\otimes\bar{N}\left(u\right)\right)\bar{p}\left(\tau\right). \tag{5.21}$$

We are now prepared to use Galerkin's method. We multiply by the test functions $\bar{N}\left(x,t\right)$ and integrate (5.1) in space and time to obtain

$$\int_a^b\int_0^T \bar{N}\left(x,t\right)\left[-\frac{\partial}{\partial x}\left(\alpha\left(x\right)\frac{\partial\phi}{\partial x}\right)+\frac{\partial^2\phi}{\partial t^2}\right]dt\,dx = 0 \tag{5.22}$$

$$\int_a^b\int_0^T \bar{N}\left(x,t\right)\left[-\frac{\partial}{\partial x}\left(\alpha\left(x\right)\frac{\partial\phi}{\partial x}\right)\right]dt\,dx + \int_a^b\int_0^T \bar{N}\left(x,t\right)\frac{\partial^2\phi}{\partial t^2}\,dt\,dx = 0. \tag{5.23}$$

In one dimension, we apply integration by parts to transfer differentiation in space to the test functions. We do the same here, but for the first term, we integrate by parts in space,

and for the second term, we integrate by parts in time. This gives

$$\int_a^b \int_0^T \frac{\partial}{\partial x} \bar{N}(x,t)\, \alpha(x) \frac{\partial \phi}{\partial x}\, dt\, dx - \int_0^T \left[\bar{N}(x,t)\, \alpha(x) \frac{\partial \phi}{\partial x} \right]_{x=a}^{b} dt \\ - \int_a^b \int_0^T \frac{\partial}{\partial t} \bar{N}(x,t) \frac{\partial \phi}{\partial t}\, dt\, dx + \int_a^b \left[\bar{N}(x,t) \frac{\partial \phi}{\partial t} \right]_{t=0}^{T} dx = 0 \quad (5.24)$$

which we treat term by term. For the first term, let us first introduce the expression (5.20) and recall that α can be approximated by a sum of Legendre polynomials in x. This gives

$$\int_a^b \int_0^T \frac{\partial}{\partial x} \bar{N}(x,t)\, \alpha(x) \frac{\partial \phi}{\partial x}\, dt\, dx = \\ |\Xi|\, \Xi^{-2} \frac{T}{2} \int_{-1}^{1} \int_{-1}^{1} \left(\bar{N}(\tau) \otimes \underline{S}_{DL} \right) \bar{p}(u) \left[\sum_{k=0}^{K_\alpha} \alpha_k p_k(u) \right] \bar{p}(u)^T \left(\bar{N}(\tau) \otimes \underline{S}_{DL} \right)^T \bar{\phi}\, d\tau\, du \quad (5.25)$$

where we have transformed to $u\tau$-space and used the transpose of (5.20) for the derivative of ϕ. Integrating with respect to u and using the tensor of triple products of Legendre polynomials gives

$$\int_a^b \int_0^T \frac{\partial}{\partial x} \bar{N}(x,t)\, \alpha(x) \frac{\partial \phi}{\partial x}\, dt\, dx = \\ |\Xi|\, \Xi^{-2} \frac{T}{2} \int_{-1}^{1} \left(\bar{N}(\tau) \otimes \underline{S}_{DL} \right) \underbrace{\left[\sum_{k=0}^{K_\alpha} \alpha_k \underline{T}_k \right]}_{\underline{\alpha}} \left(\bar{N}(\tau)^T \otimes \underline{S}_{DL}^T \right) \bar{\phi}\, d\tau. \quad (5.26)$$

To integrate with respect to τ we require one additional trivial, but important observation:

$$1 \otimes \underline{\alpha} = \underline{\alpha}. \quad (5.27)$$

With this, and property (5.12), we get

$$\int_a^b \int_0^T \frac{\partial \bar{N}}{\partial x} \alpha(x) \frac{\partial \phi}{\partial x}\, dt\, dx = |\Xi|\, \Xi^{-2} \frac{T}{2} \int_{-1}^{1} \left(\bar{N}(\tau)\, \bar{N}(\tau)^T \otimes \underline{S}_{DL} \underline{\alpha} \underline{S}_{DL}^T \right) \bar{\phi}\, d\tau \quad (5.28)$$

$$= |\Xi|\, \Xi^{-2} \frac{T}{2} \int_{-1}^{1} \left(\underline{\tilde{S}} \bar{p}(\tau)\, \bar{p}(\tau)^T\, \underline{\tilde{S}}^T \otimes \underline{S}_{DL} \underline{\alpha} \underline{S}_{DL}^T \right) \bar{\phi}\, d\tau \quad (5.29)$$

$$= |\Xi|\, \Xi^{-2} \frac{T}{2} \left(\underline{\tilde{S}} \underline{\tilde{S}}^T \otimes \underline{S}_{DL} \underline{\alpha} \underline{S}_{DL}^T \right) \bar{\phi}. \quad (5.30)$$

The structure of the matrix $\underline{\tilde{S}} \underline{\tilde{S}}^T \otimes \underline{S}_{DL} \underline{\alpha} \underline{S}_{DL}^T$ is worth further comment. In particular, the product $\underline{\tilde{S}} \underline{\tilde{S}}^T$ is what was found in the case of constant β for the ϕ term in the one-dimensional

BVP, whereas the $\underline{S}_{DL}\underline{\alpha}\underline{S}_{DL}^T$ term corresponds to the $-\frac{\partial}{\partial x}\left(\alpha\left(x\right)\frac{\partial\phi}{\partial x}\right)$ term. Since the first matrix has bandwidth 2 and the second term has bandwidth K_α, if K_α is small, then the Kronecker product of the two matrices will also be sparse. In fact, since $\underline{\tilde{S}}\underline{\tilde{S}}^T$ is pentadiagonal, by the definition of the Kronecker product, the matrix $\underline{\tilde{S}}\underline{\tilde{S}}^T \otimes \underline{S}_{DL}\underline{\alpha}\underline{S}_{DL}^T$ can be thought of as a block pentadiagonal matrix with each nonzero block possessing bandwidth K_α.

We now consider the third term in (5.24) which has similar structure. We obtain

$$\int_a^b \int_0^T \frac{\partial}{\partial t}\bar{N}\left(x,t\right)\frac{\partial\phi}{\partial t}\,dt\,dx = \\ |\Xi|\,\frac{2}{T}\int_{-1}^{1}\int_{-1}^{1}\left(\underline{S}_{DL}\otimes\bar{N}\left(u\right)\right)\bar{p}\left(\tau\right)\bar{p}\left(\tau\right)^T\left(\underline{S}_{DL}\otimes\bar{N}\left(u\right)\right)^T\bar{\phi}\,dt\,du \quad (5.31)$$

where we have used (5.21). Integrating in time, using (5.12), and then integrating in space yields

$$\int_a^b \int_0^T \frac{\partial}{\partial t}\bar{N}\left(x,t\right)\frac{\partial\phi}{\partial t}\,dt\,dx = |\Xi|\,\frac{2}{T}\int_{-1}^{1}\left(\underline{S}_{DL}\otimes\bar{N}\left(u\right)\right)\left(\underline{S}_{DL}^T\otimes\bar{N}\left(u\right)^T\right)\bar{\phi}\,du \quad (5.32)$$

$$= |\Xi|\,\frac{2}{T}\int_{-1}^{1}\left(\underline{S}_{DL}\underline{S}_{DL}^T\otimes\bar{N}\left(u\right)\bar{N}\left(u\right)^T\right)\bar{\phi}\,du \quad (5.33)$$

$$= |\Xi|\,\frac{2}{T}\int_{-1}^{1}\left(\underline{S}_{DL}\underline{S}_{DL}^T\otimes\underline{\tilde{S}}\bar{p}\left(u\right)\bar{p}\left(u\right)^T\underline{\tilde{S}}^T\right)\bar{\phi}\,du \quad (5.34)$$

$$= |\Xi|\,\frac{2}{T}\left(\underline{S}_{DL}\underline{S}_{DL}^T\otimes\underline{\tilde{S}}\underline{\tilde{S}}^T\right)\bar{\phi}. \quad (5.35)$$

Again, the matrix $\underline{S}_{DL}\underline{S}_{DL}^T \otimes \underline{\tilde{S}}\underline{\tilde{S}}^T$ is sparse. In fact, by the definition of the Kronecker product, the matrix is block diagonal with each block pentadiagonal. Thus the matrix itself is pentadiagonal.

5.2 Boundary Constraints and Lagrange Multipliers

In this section, we impose constraints on the solution of the PDE and show how they are related to the boundary terms from the weak form. First, consider the term

$$\int_0^T\left[\bar{N}\left(x,t\right)\alpha\left(x\right)\frac{\partial\phi}{\partial x}\right]_{x=a}^{b}dt = \\ \int_0^T\bar{N}\left(b,t\right)\alpha\left(b\right)\frac{\partial\phi}{\partial x}\left(b,t\right)\,dt - \int_0^T\bar{N}\left(a,t\right)\alpha\left(a\right)\frac{\partial\phi}{\partial x}\left(a,t\right)\,dt \quad (5.36)$$

as well as the boundary conditions $\phi(a,t) = 0$ and $\phi(b,t) = 0$. Let us first impose the boundary condition at $x = b$ using the linear combination of basis functions in space and time. That is,

$$\phi(b,t) = \text{vec}\left(\bar{N}(b)\,\bar{N}(t)^T\right)^T \bar{\phi} \tag{5.37}$$

$$= \text{vec}\left(\bar{N}(b)\,\bar{N}(t)^T \underline{I}_{L_t+1}\right)^T \bar{\phi} \tag{5.38}$$

$$= \left[\left(\underline{I}_{L_t+1} \otimes \bar{N}(b)\right) \text{vec}\left(\bar{N}(t)^T\right)\right]^T \bar{\phi} \tag{5.39}$$

$$= \bar{N}(t)^T \left(\underline{I}_{L_t+1} \otimes \bar{N}(b)^T\right) \bar{\phi} \tag{5.40}$$

where we have used properties (5.13) and (5.11) of the vectorization operator and the Kronecker product. Note that $\bar{N}(b) = \frac{1}{\sqrt{2}}(\bar{e}_1 + \bar{e}_2)$ since $x = b$ maps to $u = 1$, so to impose the boundary condition, we require that

$$\bar{N}(t)^T \left(\underline{I}_{L_t+1} \otimes \frac{1}{\sqrt{2}}(\bar{e}_1 + \bar{e}_2)^T\right) \bar{\phi} = 0. \tag{5.41}$$

Since this must hold for all time t, and the vector $\bar{N}(t)$ contains a linearly independent set of functions, this is equivalent to requiring

$$\underbrace{\left(\underline{I}_{L_t+1} \otimes \frac{1}{\sqrt{2}}(\bar{e}_1 + \bar{e}_2)^T\right)}_{\underline{C}_1} \bar{\phi} = 0 \tag{5.42}$$

which constitutes our first set of constraint equations.

Returning to the boundary term in our Galerkin method, we see that $\bar{N}(b,t)$ arises. From our derivation of the Dirichlet boundary condition, we know that

$$\bar{N}(b,t) = \underline{C}_1^T \bar{N}(t) \tag{5.43}$$

which means that

$$\int_0^T \bar{N}(b,t)\,\alpha(b)\,\frac{\partial \phi}{\partial x}(b,t)\,dt = \int_0^T \underline{C}_1^T \bar{N}(t)\,\alpha(b)\,\frac{\partial \phi}{\partial x}(b,t)\,dt \tag{5.44}$$

$$= \underline{C}_1^T \underbrace{\int_0^T \bar{N}(t)\,\alpha(b)\,\frac{\partial \phi}{\partial x}(b,t)\,dt}_{\bar{\nu}_1} \tag{5.45}$$

where we have defined our first set of Lagrange multipliers $\bar{\nu}_1$. Their physical meaning is less clear than those found in the one-dimensional BVP, but we can still interpret their

significance. Each Lagrange multiplier corresponds to the inner product in time of a temporal basis function with the flux $\alpha\frac{\partial\phi}{\partial x}$ at $x = b$.

Similarly, we now impose the boundary condition $\phi(a,t) = 0$. We replace $\bar{N}(b)$ with $\bar{N}(a) = \frac{1}{\sqrt{2}}(\bar{e}_1 - \bar{e}_2)$ since $x = a$ maps to $u = -1$ to get a second set of constraints

$$\underbrace{\left(\underline{I}_{L_t+1} \otimes \frac{1}{\sqrt{2}}(\bar{e}_1 - \bar{e}_2)^T\right)}_{\underline{C}_2}\bar{\phi} = 0. \tag{5.46}$$

From these constraints, we see that $\bar{N}(a,t) = \underline{C}_2^T\bar{N}(t)$ and conclude that the boundary term

$$-\int_0^T \bar{N}(a,t)\,\alpha(a)\,\frac{\partial\phi}{\partial x}(a,t)\;dt = -\int_0^T \underline{C}_2^T\bar{N}(t)\,\alpha(a)\,\frac{\partial\phi}{\partial x}(a,t)\;dt \tag{5.47}$$

$$= \underline{C}_2^T\underbrace{\left[-\int_0^T \bar{N}(t)\,\alpha(a)\,\frac{\partial\phi}{\partial x}(a,t)\;dt\right]}_{\bar{\nu}_2}. \tag{5.48}$$

This means that the second set of Lagrange multipliers correspond to the negative inner product in time of a temporal basis function with the flux at $x = a$.

The final term to address in the weak form is

$$\int_a^b \left[\bar{N}(x,t)\,\frac{\partial\phi}{\partial t}\right]_{t=0}^{T} dx = \int_a^b \bar{N}(x,T)\,\frac{\partial\phi}{\partial t}(x,T)\;dx - \int_a^b \bar{N}(x,0)\,\frac{\partial\phi}{\partial t}(x,0)\;dx \tag{5.49}$$

and the remaining two boundary conditions to formulate constraints for are the initial conditions $\phi(x,0) = \phi_0(x)$ and $\frac{\partial}{\partial t}\phi(x,0) = \phi_0'(x)$. Let us derive the associated constraint equations for each. First, consider

$$\phi(x,0) = \mathrm{vec}\left(\bar{N}(x)\,\bar{N}(0)^T\right)^T\bar{\phi} \tag{5.50}$$

$$= \mathrm{vec}\left(\underline{I}_{L_x+1}\bar{N}(x)\,\bar{N}(0)^T\right)^T\bar{\phi} \tag{5.51}$$

$$= \left[\left(\bar{N}(0)\otimes\underline{I}_{L_x+1}\right)\bar{N}(x)\right]^T\bar{\phi} \tag{5.52}$$

$$= \bar{N}(x)^T\left(\bar{N}(0)^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi}. \tag{5.53}$$

Since $t = 0$ maps to $\tau = -1$, we have $\bar{N}(0) = \frac{1}{\sqrt{2}}(\bar{e}_1 - \bar{e}_2)$ and

$$\phi(x,0) = \bar{N}(x)^T\left(\frac{1}{\sqrt{2}}(\bar{e}_1 - \bar{e}_2)^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi}. \tag{5.54}$$

Remark that here $\bar{e}_1$ and $\bar{e}_2$ are vectors of length $L_t + 1$. This differs from the lengths of

$\bar{e}_1$ and $\bar{e}_2$ in constraint matrices $\underline{C}_1$ and $\underline{C}_2$ (which were vectors of length $L_x + 1$). We now specify how to set $\phi(x, 0)$ equal to the initial condition $\phi_0(x)$. We compute the first $L_x + 1$ Legendre coefficients for the function ϕ_0 on the interval (a, b) so that

$$\phi_0(x) \approx \bar{p}(x)^T \bar{\phi}_0 \tag{5.55}$$

and set this equal to (5.54). This gives

$$\bar{N}(x)^T \left(\frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \otimes \underline{I}_{L_x+1} \right) \bar{\phi} = \bar{p}(x)^T \bar{\phi}_0 \tag{5.56}$$

$$\bar{p}(x)^T \underline{\tilde{S}}^T \left(\frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \otimes \underline{I}_{L_x+1} \right) \bar{\phi} = \bar{p}(x)^T \bar{\phi}_0 \tag{5.57}$$

and, since this must hold for all x and $\bar{p}(x)$ contains a linearly independent set of functions, this is equivalent to

$$\underline{\tilde{S}}^T \left(\frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \otimes \underline{I}_{L_x+1} \right) \bar{\phi} = \bar{\phi}_0 \tag{5.58}$$

$$\underbrace{\left(\frac{1}{\sqrt{2}} (\bar{e}_1 - \bar{e}_2)^T \otimes \underline{I}_{L_x+1} \right)}_{\underline{C}_3} \bar{\phi} = \underbrace{\underline{\tilde{S}}^{-T} \bar{\phi}_0}_{\bar{d}_3} \tag{5.59}$$

which corresponds to a third set of constraint equations. The inversion of $\underline{\tilde{S}}^T$ is performed using backward substitution since $\underline{\tilde{S}}^T$ is upper triangular. In fact, since $\underline{\tilde{S}}$ only has two nonzero diagonals, this inversion requires $\mathcal{O}(L_x)$ operations which is negligible in comparison to computing the Legendre coefficients $\bar{\phi}_0$ (which we compute using the fast algorithm described in Appendix A). Also, note that to obtain accurate solutions to the wave equation, it is imperative to choose L_x large enough such that the Legendre coefficients of the initial condition $\phi_0(x)$ sufficiently decay.

We are now prepared to address the boundary term

$$-\int_a^b \bar{N}(x, 0) \frac{\partial \phi}{\partial t}(x, 0)\, dx. \tag{5.60}$$

We recognize from the derivation of the third set of constraints that $\bar{N}(x, 0) = \underline{C}_3^T \bar{N}(x)$

which gives

$$-\int_a^b \bar{N}(x,0)\frac{\partial\phi}{\partial t}(x,0)\,dx = -\int_a^b \underline{C}_3^T\bar{N}(x)\frac{\partial\phi}{\partial t}(x,0)\,dx \tag{5.61}$$

$$= \underline{C}_3^T\underbrace{\left[-\int_a^b \bar{N}(x)\frac{\partial\phi}{\partial t}(x,0)\,dx\right]}_{\bar{\nu}_3} \tag{5.62}$$

and leads to the interpretation that each Lagrange multiplier corresponds to the negative inner product in space of a spatial basis function with the initial condition for the time derivative.

To deal with the second initial condition, we follow the same procedure as before, but now must take into account the time derivative of ϕ. This amounts to considering

$$\frac{\partial}{\partial t}\phi(x,0) = \frac{2}{T}\frac{\partial}{\partial\tau}\phi(x,-1) \tag{5.63}$$

$$= \frac{2}{T}\frac{\partial}{\partial\tau}\left[\text{vec}\left(\bar{N}(x)\bar{N}(\tau)^T\right)^T\bar{\phi}\right]_{\tau=-1} \tag{5.64}$$

$$= \frac{2}{T}\left[\text{vec}\left(\bar{N}(x)\frac{\partial}{\partial\tau}\bar{N}(\tau)^T\right)^T\bar{\phi}\right]_{\tau=-1} \tag{5.65}$$

$$= \frac{2}{T}\text{vec}\left(\bar{N}(x)\left(\underline{S}_{DL}\bar{p}(-1)\right)^T\right)^T\bar{\phi}. \tag{5.66}$$

We now introduce the identity matrix and apply the vectorization and Kronecker product properties (5.13) and (5.11) to obtain

$$\frac{\partial}{\partial t}\phi(x,0) = \frac{2}{T}\text{vec}\left(\underline{I}_{L_x+1}\bar{N}(x)\left(\underline{S}_{DL}\bar{p}(-1)\right)^T\right)^T\bar{\phi} \tag{5.67}$$

$$= \frac{2}{T}\left[\left(\underline{S}_{DL}\bar{p}(-1)\otimes\underline{I}_{L_x+1}\right)\bar{N}(x)\right]^T\bar{\phi} \tag{5.68}$$

$$= \frac{2}{T}\bar{N}(x)^T\left(\bar{p}(-1)^T\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi} \tag{5.69}$$

which we can set equal to the Legendre expansion of $\phi_0'(x)\approx\bar{p}(x)^T\bar{\phi}_0'$. This gives

$$\frac{2}{T}\bar{N}(x)^T\left(\bar{p}(-1)^T\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi} = \bar{p}(x)^T\bar{\phi}_0' \tag{5.70}$$

$$\frac{2}{T}\bar{p}(x)^T\underline{\tilde{S}}^T\left(\bar{p}(-1)^T\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi} = \bar{p}(x)^T\bar{\phi}_0' \tag{5.71}$$

which reduces to

$$\frac{2}{T}\tilde{\underline{S}}^T\left(\bar{p}\,(-1)^T\,\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi}=\bar{\phi}_0' \tag{5.72}$$

$$\underbrace{\left(\bar{p}\,(-1)^T\,\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)}_{\underline{C}_4}\bar{\phi}=\underbrace{\frac{T}{2}\tilde{\underline{S}}^{-T}\bar{\phi}_0'}_{\bar{d}_4} \tag{5.73}$$

using the orthonormality of the Legendre polynomials $\bar{p}\,(x)$. Unfortunately, there is no boundary term in the weak form that makes use of $\frac{\partial}{\partial t}\bar{N}\,(x,0)=\underline{C}_4^T\bar{N}\,(x)$. To preserve symmetry of constraint matrices with Lagrange multiplier matrices, we introduce the zero vector to the Galerkin equation by adding $\underline{C}_4^T\bar{\nu}_4$ where $\bar{\nu}_4$ is a zero vector of Lagrange multipliers. To understand why $\bar{\nu}_4$ can be no other vector, we note that since the rank of $\underline{C}_4$ is L_x+1 and $\text{rank}\,(\underline{C}_4)=\text{rank}(\underline{C}_4^T)$, we can use the rank-nullity theorem to confirm that $\text{null}(\underline{C}_4^T)=0$, meaning that the only vector $\bar{\nu}_4$ that gives $\underline{C}_4^T\bar{\nu}_4=0$ is the zero vector. This means that we can check if the approximate solution to the wave equation is correct by verifying whether $\bar{\nu}_4\approx 0$.

Now that we have treated all of the constraints, we are left with a single boundary term unaddressed in the Galerkin method, given by

$$\int_a^b \bar{N}\,(x,T)\,\frac{\partial\phi}{\partial t}\,(x,T)\;dx. \tag{5.74}$$

To discretize this term, we use (5.53) and (5.69), replacing $\bar{N}\,(0)$ with $\bar{N}\,(T)$ and $\bar{p}\,(-1)$ with $\bar{p}\,(1)$. This gives

$$\bar{N}\,(x,T)=\left(\bar{N}\,(T)\otimes\underline{I}_{L_x+1}\right)\bar{N}\,(x) \tag{5.75}$$

$$=\left(\frac{1}{\sqrt{2}}\,(\bar{e}_1+\bar{e}_2)\otimes\underline{I}_{L_x+1}\right)\bar{N}\,(x) \tag{5.76}$$

and

$$\frac{\partial}{\partial t}\phi\,(x,T)=\frac{2}{T}\bar{N}\,(x)^T\left(\bar{p}\,(1)^T\,\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi}. \tag{5.77}$$

Substituting these expressions into the boundary term gives

$$\int_a^b \bar{N}\,(x,T)\,\frac{\partial\phi}{\partial t}\,(x,T)\;dx=$$
$$|\Xi|\,\frac{2}{T}\int_{-1}^{1}\left(\frac{1}{\sqrt{2}}\,(\bar{e}_1+\bar{e}_2)\otimes\underline{I}_{L_x+1}\right)\bar{N}\,(u)\,\bar{N}\,(u)^T\left(\bar{p}\,(1)^T\,\underline{S}_{DL}^T\otimes\underline{I}_{L_x+1}\right)\bar{\phi}\,du \tag{5.78}$$

which, after integration, results in

$$\int_a^b \bar{N}(x,T)\,\frac{\partial\phi}{\partial t}(x,T)\,dx = |\Xi|\,\frac{2}{T}\left(\frac{1}{\sqrt{2}}(\bar{e}_1+\bar{e}_2)\otimes \underline{I}_{L_x+1}\right)\tilde{\underline{S}}\tilde{\underline{S}}^T\left(\bar{p}(1)^T\,\underline{S}_{DL}^T\otimes \underline{I}_{L_x+1}\right)\bar{\phi}. \tag{5.79}$$

This term is sparse, but not symmetric. To see why, we use the same approach as in (5.27) to multiply $\tilde{\underline{S}}\tilde{\underline{S}}^T$ into the other Kronecker product terms using the multiplication property (5.12) to see that

$$\int_a^b \bar{N}(x,T)\,\frac{\partial\phi}{\partial t}(x,T)\,dx = |\Xi|\,\frac{2}{T}\left(\frac{1}{\sqrt{2}}(\bar{e}_1+\bar{e}_2)\left(\underline{S}_{DL}\bar{p}(1)\right)^T\otimes \tilde{\underline{S}}\tilde{\underline{S}}^T\right)\bar{\phi}. \tag{5.80}$$

The matrix $\frac{1}{\sqrt{2}}(\bar{e}_1+\bar{e}_2)\left(\underline{S}_{DL}\bar{p}(1)\right)^T$ is an outer product matrix with nonzeros only in the first and second row. Thus, by the definition of the Kronecker product, the Kronecker product of the outer product matrix with $\tilde{\underline{S}}\tilde{\underline{S}}^T$ has nonzero blocks only in the first two block rows, and each block has the pentadiagonal sparsity pattern of $\tilde{\underline{S}}\tilde{\underline{S}}^T$.

To complete the formulation, we combine (5.30), (5.35), and (5.80) and define the operator matrix

$$\begin{aligned}\underline{A} = |\Xi|\,\Xi^{-2}\frac{T}{2}\left(\tilde{\underline{S}}\tilde{\underline{S}}^T\otimes \underline{S}_{DL}\underline{\alpha}\underline{S}_{DL}^T\right) - |\Xi|\,\frac{2}{T}\left(\underline{S}_{DL}\underline{S}_{DL}^T\otimes \tilde{\underline{S}}\tilde{\underline{S}}^T\right)\\ + |\Xi|\,\frac{2}{T}\left(\frac{1}{\sqrt{2}}(\bar{e}_1+\bar{e}_2)\left(\underline{S}_{DL}\bar{p}(1)\right)^T\otimes \tilde{\underline{S}}\tilde{\underline{S}}^T\right).\end{aligned} \tag{5.81}$$

We also construct the constraint matrix, constraint right hand side vector, and Lagrange multiplier vector

$$\underline{C} = \begin{bmatrix}\underline{C}_1\\ \underline{C}_2\\ \underline{C}_3\\ \underline{C}_4\end{bmatrix}, \qquad \bar{d} = \begin{bmatrix}0\\ 0\\ \bar{d}_3\\ \bar{d}_4\end{bmatrix}, \qquad \bar{\nu} = \begin{bmatrix}\bar{\nu}_1\\ \bar{\nu}_2\\ \bar{\nu}_3\\ \bar{\nu}_4\end{bmatrix}, \tag{5.82}$$

so that we obtain the saddle point system

$$\begin{bmatrix}\underline{A} & \underline{C}^T\\ \underline{C} & 0\end{bmatrix}\begin{bmatrix}\bar{\phi}\\ \bar{\nu}\end{bmatrix} = \begin{bmatrix}0\\ \bar{d}\end{bmatrix}. \tag{5.83}$$

There are two subtleties hidden by this saddle point notation. The first is that although this saddle point system has the same block structure as those found in earlier chapters, the matrix $\underline{A}$ in this case is not symmetric due to the free boundary term at time $t = T$. The second, and more important point is that the matrix $\underline{C}$ does not have full rank and, as a

consequence, the saddle point system is singular. Taken separately, each constraint matrix $\underline{C}_k$ with $k = 1, 2, 3, 4$, has full rank but, when combined, there are four redundant equations. To understand why, consider the fact that constraint matrices $\underline{C}_1$ and $\underline{C}_2$ force ϕ to be zero at $x = b$ and $x = a$ respectively. Constraint matrix $\underline{C}_3$ forces ϕ to match $\phi_0(x)$ for $a \leq x \leq b$ and $t = 0$. Suppose now that $\phi_0(a) \neq 0$ or $\phi_0(b) \neq 0$. Then constraints $\underline{C}_1$ and $\underline{C}_2$ can be in direct conflict with $\underline{C}_3$: we have specified the solution ϕ at the points $(a, 0)$ and $(b, 0)$ twice. The correct approach is thus to remove the constraints either in $\underline{C}_1$ and $\underline{C}_2$ or $\underline{C}_3$ at those points. Since only the first two basis functions in the time dimension and the first two in the space dimension have any impact on the solution at the points $(a, 0)$ and $(b, 0)$, it is enough to eliminate the first two equations in both $\underline{C}_1$ and $\underline{C}_2$ to restore full rank to $\underline{C}$. Note that this is not the only choice that restores the full rank property to $\underline{C}$, but it is the simplest when generalizing to multiple elements in space. Once this issue is clarified, we can solve for the space-time solution ϕ by inverting the saddle point system. We can also certify that our solution is accurate by measuring $\|\bar{\nu}_4\|_2$ and verifying that the coefficients in $\bar{\phi}$ have decayed to the user specified tolerance in both space and time.

5.3 Space-Time Reflection of a Gaussian Pulse

To test that this formulation is correct, we apply the method to a classical wave propagation problem in one spatial dimension. Consider Maxwell's equations under the assumptions that the electric and magnetic field intensities reduce to

$$\bar{E} = \begin{bmatrix} 0 \\ E_y(x,t) \\ 0 \end{bmatrix}, \qquad \bar{H} = \begin{bmatrix} 0 \\ 0 \\ H_z(x,t) \end{bmatrix}. \tag{5.84}$$

Then the Faraday and Ampère laws in the absence of current and charge densities, given by

$$\nabla \times \bar{E} = -\mu \frac{\partial}{\partial t} \bar{H}, \tag{5.85}$$

$$\nabla \times \bar{H} = \epsilon \frac{\partial}{\partial t} \bar{E}, \tag{5.86}$$

can be reduced to the one-dimensional wave equation

$$-\frac{\partial}{\partial x}\left(\frac{1}{\epsilon}\frac{\partial}{\partial x} H_z\right) + \mu \frac{\partial^2}{\partial t^2} H_z = 0. \tag{5.87}$$

This equation can be transformed to a dimensionless one with the change of variables

$$\hat{t} = \frac{1}{L\sqrt{\epsilon_0 \mu_0}} t, \qquad \hat{x} = \frac{1}{L} x, \qquad \hat{H}_z = \frac{1}{H_{\text{norm}}} H_z, \qquad \hat{E}_y = \frac{1}{H_{\text{norm}}} \sqrt{\frac{\epsilon_0}{\mu_0}} E_y, \tag{5.88}$$

where L is some characteristic length (often associated with wavelength) and H_{norm} is a normalizing magnetic field strength. Performing this change of variables and dropping the circumflex notation, we obtain

$$-\frac{\partial}{\partial x}\left(\frac{1}{\epsilon_r}\frac{\partial}{\partial x} H_z\right) + \mu_r \frac{\partial^2}{\partial t^2} H_z = 0 \tag{5.89}$$

and, if we let $\mu_r = 1$, we recover the wave equation (5.1) where $\alpha(x) = 1/[\epsilon_r(x)]$ and $\phi = H_z$. For our test case, we choose $\epsilon_r = 1$ with initial conditions

$$H_0(x) = \exp\left[-\frac{1}{2}\left(\frac{x - x_0}{\Delta x}\right)^2\right], \tag{5.90}$$

$$\frac{\partial}{\partial t} H_0(x) = -\frac{1}{(\Delta x)^2}(x - x_0) \exp\left[-\frac{1}{2}\left(\frac{x - x_0}{\Delta x}\right)^2\right], \tag{5.91}$$

with $x_0 = 0$ and $\Delta x = 1/8$. We also set $a = -1$ and $b = 1$ which ensures that H_0 has decayed to machine precision at the boundaries. Since the boundary conditions at a and b require that H_z be zero for all time, we expect the Gaussian pulse to propagate in the negative x direction, and perfectly reflect at each boundary. We choose $T = 4$ so that the pulse reflects at both endpoints and returns to its original position (the wave has unit speed in the unitless system). To illustrate the sparsity of the saddle point system, we set $L_x = L_t = 10$ (see Figure 5.1) but we choose $L_x = L_t = 100$ when computing an accurate solution to the wave problem. The saddle point system is solved using a sparse direct solver. The computed solution is accurate to machine precision (this can be verified since the exact solution to this problem is known). In addition, the Lagrange multipliers $\bar{\nu}_4$ are zero to within machine precision.

Figure 5.2 illustrates the space-time distribution of the magnetic field intensity of the Gaussian pulse. The interpretation of space-time distributions may seem to be ambiguous to the uninitiated. A point (x, t) on the plot has a corresponding value $H_z(x, t)$ represented in color by red for positive one and blue for negative one (with intermediate values taken from a color gradient between red and blue with white as zero). If one takes a horizontal slice of the space-time distribution, say at time $t = t_\star$, then the slice gives $H_z(x, t_\star)$ which is the profile of the spatially one-dimensional wave (in this case, a Gaussian pulse) which one typically sees from a time-stepping visualization. Alternatively, one can take a vertical slice,

say at $x = x_\star$, which corresponds to tracking the value of H_z at a fixed point in space over the full duration of time, that is, $H_z(x_\star, t)$. For the particular example depicted in Figure 5.2, one can think of the space-time plot as corresponding to a more detailed version of a reflection diagram one might typically find for transmission-line circuits in electromagnetic fields and waves books (see [135], for example).

5.4 Space-Time Simulation of a Fiber Bragg Grating

As a second example, consider the case where ϵ_r is now a function of space. In particular, let us choose $a = 0$ and $b = 6$ up to a final time $T = 4$ with a relative permittivity given by

$$\epsilon_r(x) = \left\{ \frac{1}{2} \exp\left[-\frac{1}{2} \left(\frac{x - x_{0,r}}{\Delta x_r} \right)^2 \right] \left[\cos\left(2\pi\omega_r \left(x - x_{0,r} \right) \right) + 1 \right] + 1 \right\}^2 \tag{5.92}$$

where $\Delta x_r = 3/8$, $\omega_r = 10$, and $x_{0,r} = 5/2$. This relative permittivity can be thought of as corresponding to a Gaussian apodized fiber Bragg grating [136]. Let us consider two problems. First, we take the initial conditions

$$H_0(x) = \exp\left[-\frac{1}{2} \left(\frac{x - x_0}{\Delta x} \right)^2 \right], \tag{5.93}$$

$$\frac{\partial}{\partial t} H_0(x) = -\frac{1}{(\Delta x)^2} (x - x_0) \exp\left[-\frac{1}{2} \left(\frac{x - x_0}{\Delta x} \right)^2 \right], \tag{5.94}$$

with $\Delta x = 1/8$ and $x_0 = 5$ which corresponds to a Gaussian pulse traveling in the negative x direction. Second, we take the initial conditions

$$H_0(x) = \cos\left(2\pi\omega \left(x - x_0 \right) \right) \exp\left[-\frac{1}{2} \left(\frac{x - x_0}{\Delta x} \right)^2 \right], \tag{5.95}$$

$$\begin{aligned} \frac{\partial}{\partial t} H_0(x) = &-2\pi\omega \sin\left(2\pi\omega \left(x - x_0 \right) \right) \exp\left[-\frac{1}{2} \left(\frac{x - x_0}{\Delta x} \right)^2 \right] \\ &- \frac{1}{(\Delta x)^2} (x - x_0) \cos\left(2\pi\omega \left(x - x_0 \right) \right) \exp\left[-\frac{1}{2} \left(\frac{x - x_0}{\Delta x} \right)^2 \right], \end{aligned} \tag{5.96}$$

with the same Δx and x_0 as in the first set of initial conditions and with frequency $\omega = 3.8197$. This corresponds to a modulated Gaussian pulse traveling in the negative x direction. Figure 5.3 shows the reciprocal of the relative permittivity $\alpha(x) = 1/[\epsilon_r(x)]$ along with the second initial condition. Note that $\alpha(x)$ is a smooth function and can be approximated to machine precision by a Legendre expansion. Unfortunately, in this case K_α—the number of required terms to represent α—is large (roughly 500). Thus, if we solve this problem using a single

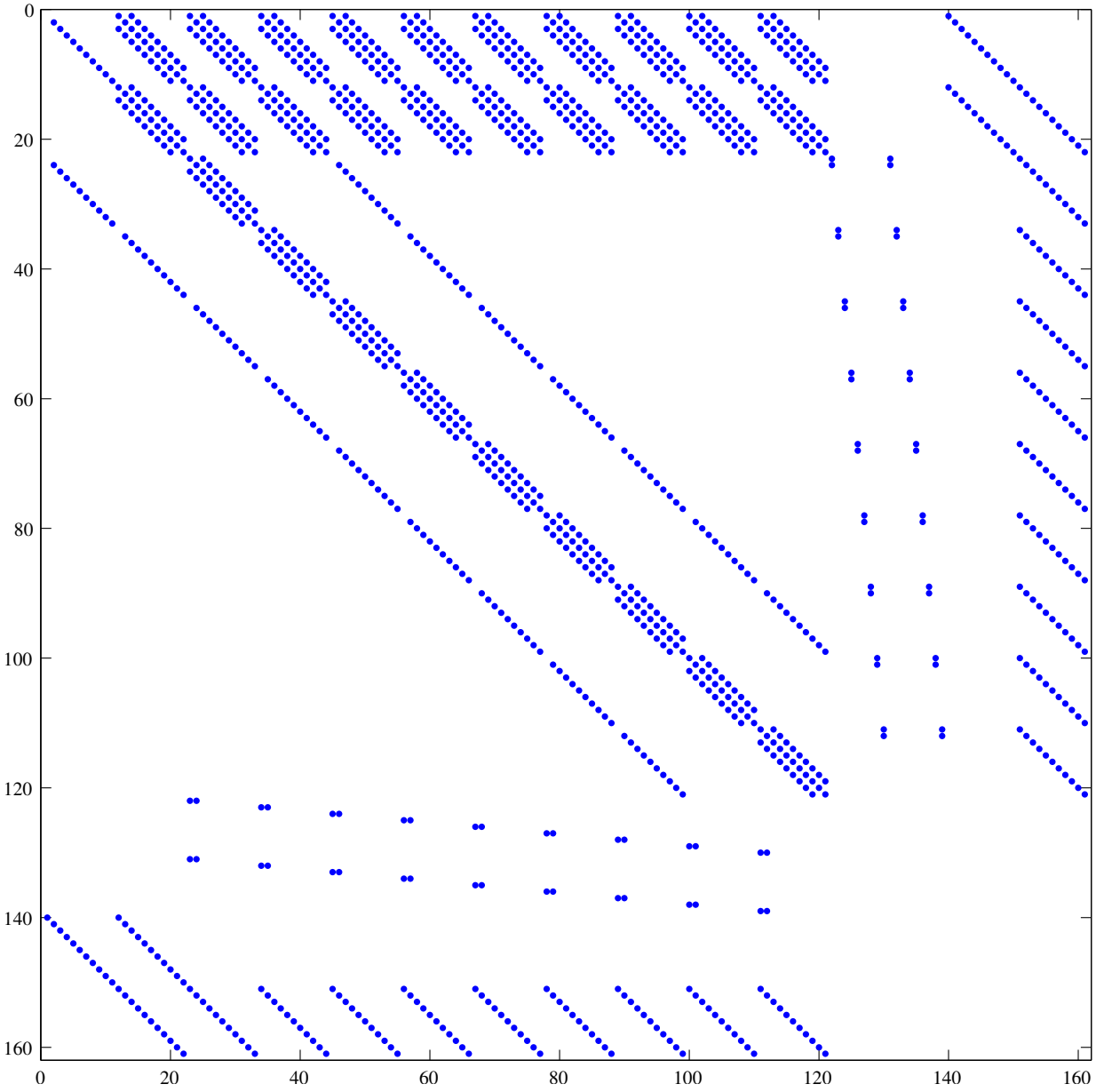

Figure 5.1: Sparsity pattern for the single element space-time formulation corresponding to the saddle point system (5.83) with $L_x = L_t = 10$. We show this reduced-size case to emphasize key features of the sparsity pattern that become difficult to see when $L_x = L_t = 100$. The band of nonzeros along the top of the plot corresponds to the nonsymmetric term in $\underline{A}$. When $L_x = L_t = 100$, there are $(10,601)^2$ total entries in the matrix (roughly 110 million) whereas there are only $130,497$ nonzeros.

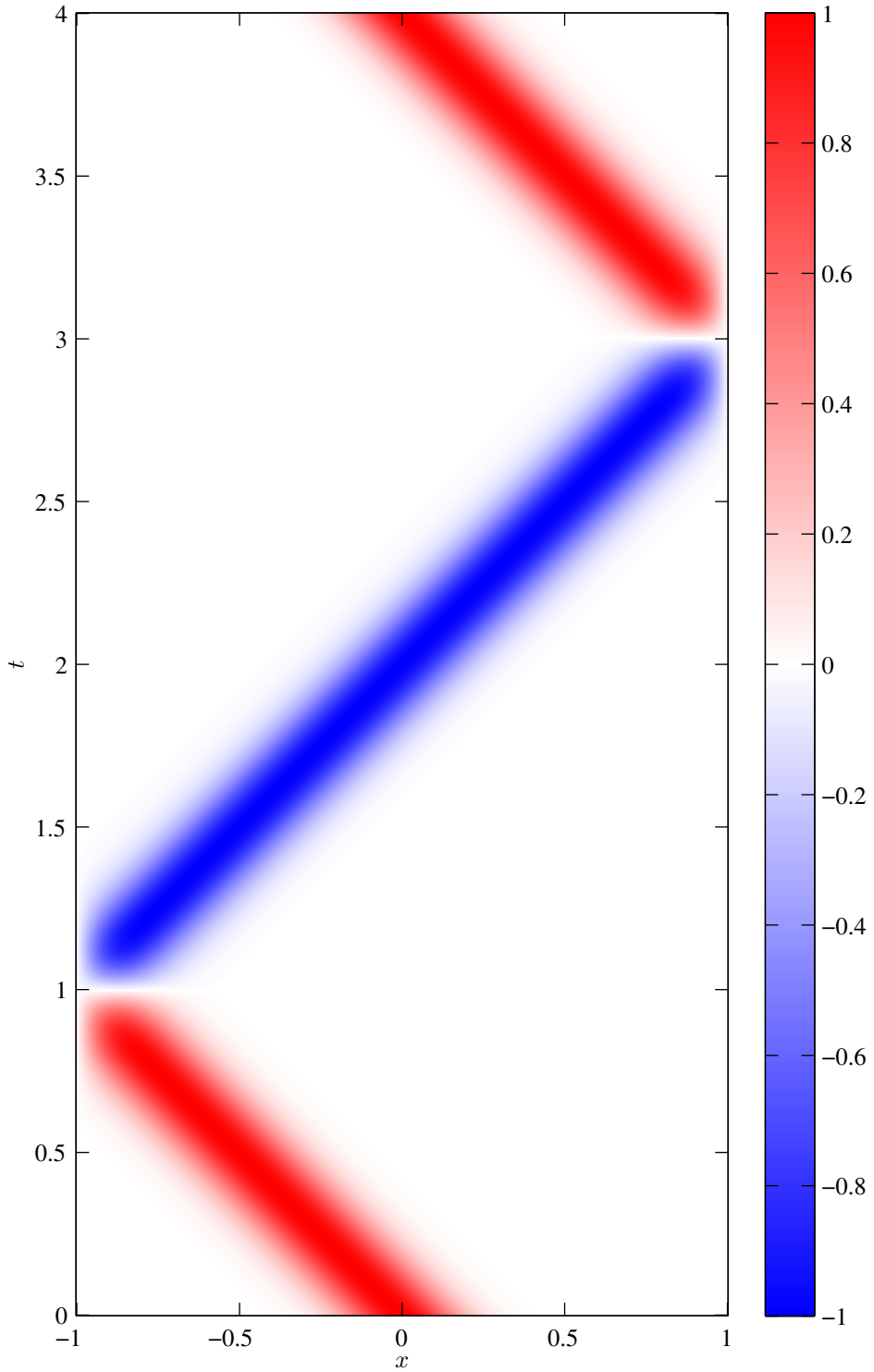

Figure 5.2: Space-time solution of the one-dimensional wave equation with perfect reflecting boundary conditions. A Gaussian pulse travels to the left and reflects at the boundary $x = -1$ at time $t = 1$, travels to the right and reflects at the boundary $x = 1$ at time $t = 3$, then finally travels to the left and returns to its initial state at time $t = 4$.

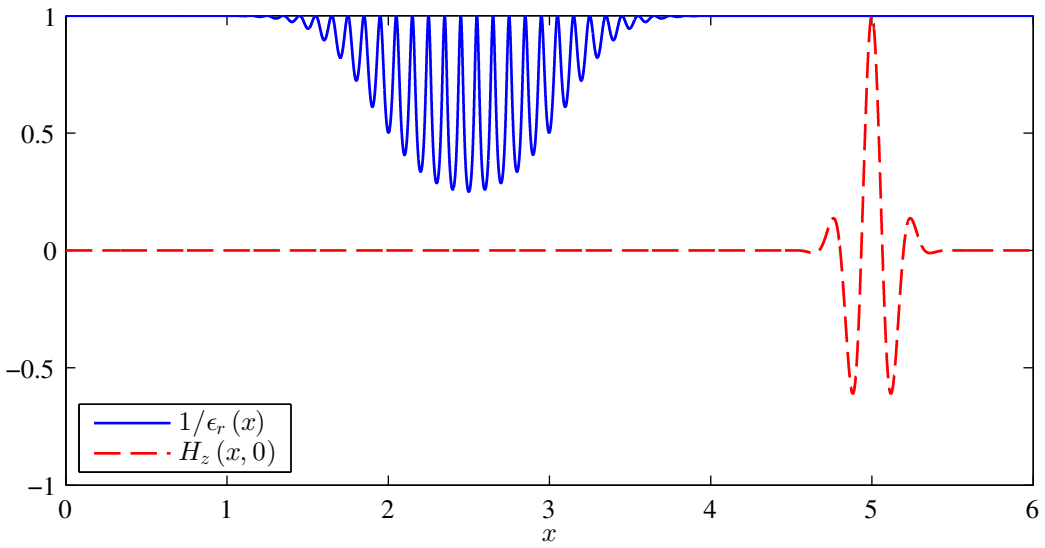

Figure 5.3: Plot of $\alpha\,(x) = 1/[\epsilon_r(x)]$ corresponding to the Gaussian apodized fiber Bragg grating as well as the modulated Gaussian wave packet used as initial condition $H_z\,(x,0)$.

element, the sparsity of the saddle point system degrades since the first term in (5.81) is block pentadiagonal with each block having bandwidth K_α (assuming K_α is smaller than the degree L_x, otherwise the block is full).

To avoid compromising sparsity, we introduce several elements in space. Each element is permitted to have different spatial degree L_x but all elements share the same temporal degree L_t. Only minor modifications of the single element approach are required to include multiple elements. In assembling the saddle point system, we replace $\underline{A}$ with a block diagonal matrix containing each element's corresponding local $\underline{A}$ matrix, as in Section 4.4. In addition, we impose the initial conditions on each element and the homogeneous Dirichlet boundary conditions on the two boundary elements that share points at $x = a$ and $x = b$. Finally, if two elements share a boundary, say at point x_j where x_j^- is the right boundary of element j and x_j^+ is the left boundary of element $j+1$ (assuming elements are ordered by increasing number from left to right), then we impose continuity by requiring

$$\phi(x_j^-, t) = \phi(x_j^+, t) \tag{5.97}$$

which, from our previous discussion of imposing Dirichlet boundary conditions in the single

element case, reduces to

$$\underline{C}_1 \bar{\phi}_j = \underline{C}_2 \bar{\phi}_{j+1} \tag{5.98}$$

$$\begin{bmatrix} \underline{C}_1 & -\underline{C}_2 \end{bmatrix} \begin{bmatrix} \bar{\phi}_j \\ \bar{\phi}_{j+1} \end{bmatrix} = 0 \tag{5.99}$$

where $\bar{\phi}_j$ is the block of unknowns corresponding to those basis functions defined on the jth element. When there are more than two elements, we pad the constraint matrix with blocks of zeros corresponding to unknowns for other elements so that a typical continuity constraint between two neighboring elements is given by

$$\begin{bmatrix} 0 & \cdots & 0 & \underline{C}_1 & -\underline{C}_2 & 0 & \cdots & 0 \end{bmatrix} \bar{\phi} = 0. \tag{5.100}$$

As with the single element case, we remove the first two rows in this inter-element continuity constraint to ensure that the complete constraint matrix $\underline{C}$ has full rank. We then perform h-adaption so as to preserve a specified bandwidth K_α in space. In our example, we set $K_\alpha = 10$, which results in a non-uniform partition of the interval $(0, 6)$ into 109 elements. We then set $L_x = 10$ for all elements, except those that require more degrees of freedom to represent the initial conditions to machine precision. The time degree is set such that $L_t = 150$ for all elements.

Figure 5.4 shows the typical sparsity of a space-time finite element saddle point system where we have reduced the number of elements to six so that the structure in the matrix can be observed. Figure 5.5 illustrates the computed space-time solutions (obtained using a sparse direct solver) for the first and second sets of initial conditions. Note that the fine structure of the scattered solution for the modulated Gaussian pulse for the second set of initial conditions would require high order polynomial basis functions for a single element solution which is expensive when K_α must be large (on the order of 500).

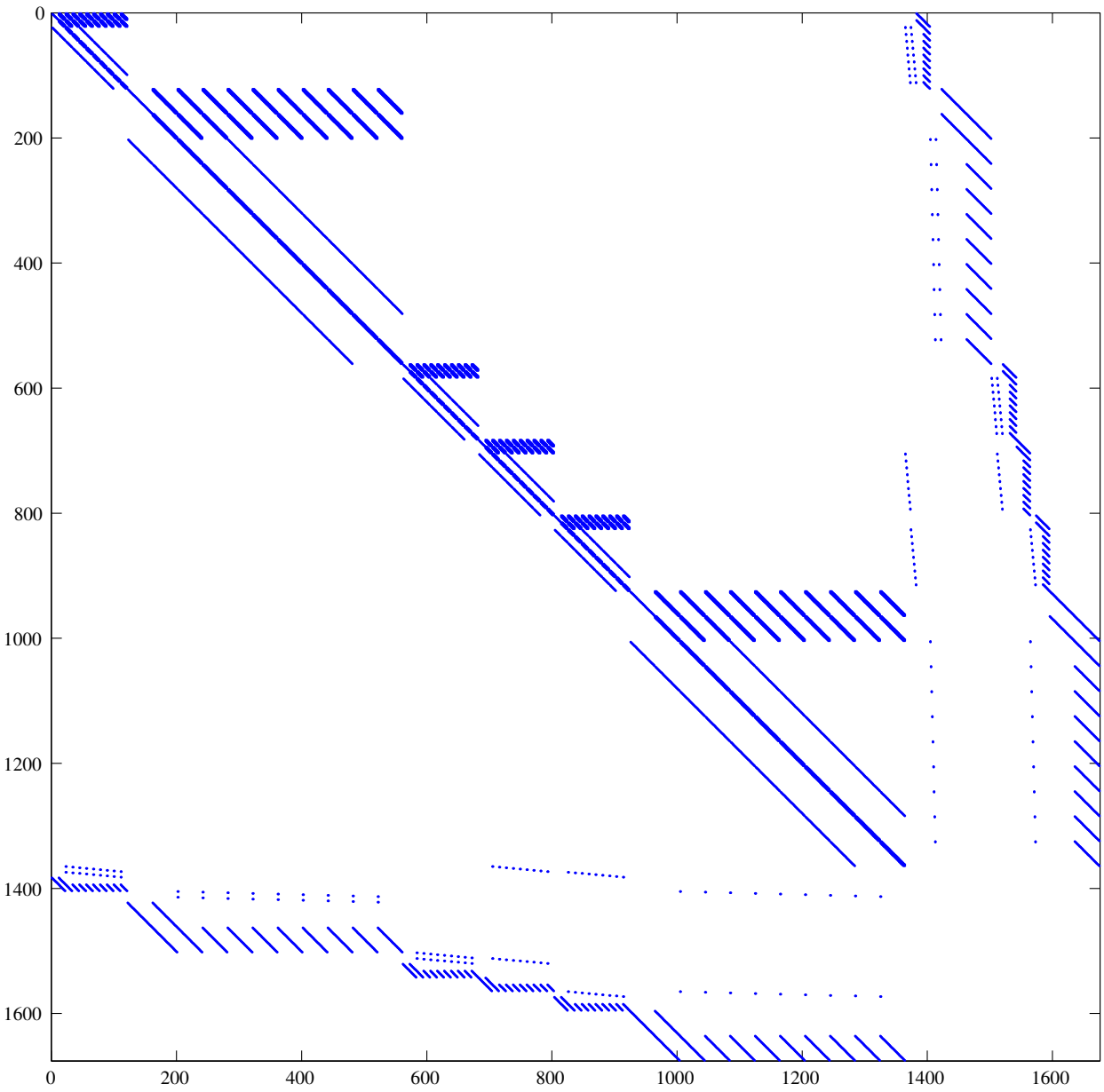

Figure 5.4: Sparsity pattern for a typical six-element space-time formulation corresponding to the saddle point system (5.83) with $L_t = 10$. We show this reduced-size case to emphasize key features of the sparsity pattern that become difficult to see when $L_t = 150$. Note how $\underline{A}$ is now built from six local versions of (5.81) along the main block diagonal. In addition, the constraint matrix now imposes two homogeneous Dirichlet boundary conditions, five inter-element continuity conditions, and six sets of initial conditions.

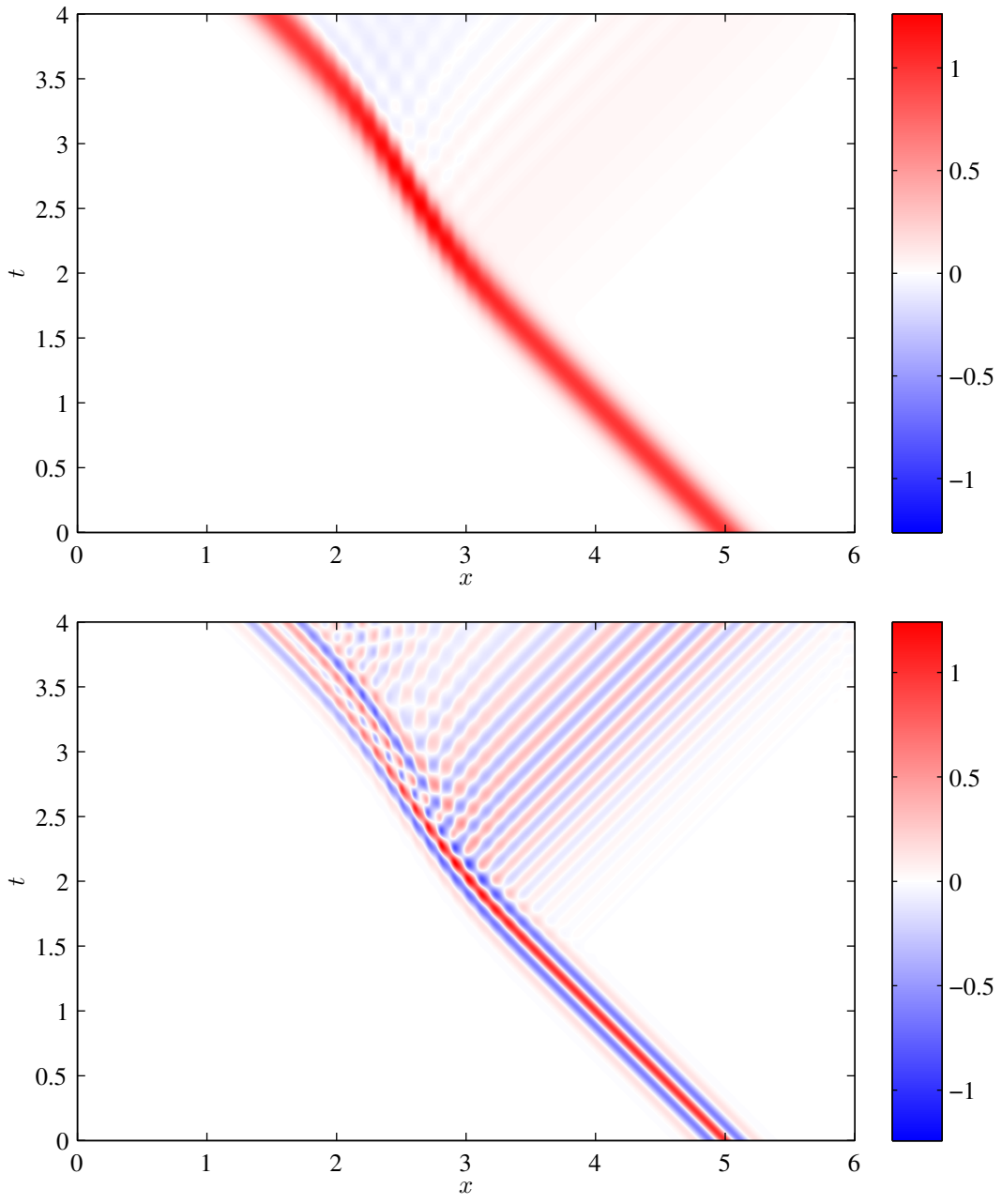

Figure 5.5: Space-time distribution of $H_z(x,t)$ for the fiber Bragg grating problem. The first set of initial conditions yields the top plot while the second set of initial conditions yields the bottom plot. Note that the magnitude of the reflected wave in the bottom plot is an order of magnitude larger than the reflected wave in the top plot (the exact difference cannot be observed by eye). This agrees with the typical behavior of fiber Bragg gratings which can be designed to reflect specific frequency bands.

Chapter 6

Extension to Higher Spatial Dimensions

The remainder of the thesis extends the finite element method described in Chapter 4 for the prototypical one-dimensional BVP (3.1)-(3.3) to an analogous method for the PDE

$$-\nabla\cdot(\underline{\alpha}\nabla\phi)+\beta\phi=f \qquad \text{in } \Omega, \tag{6.1}$$

subject to boundary conditions

$$\phi=p \qquad \text{on } \Gamma_D, \tag{6.2}$$

$$\bar{n}^T(\underline{\alpha}\nabla\phi)+\gamma\phi=q \qquad \text{on } \Gamma_R, \tag{6.3}$$

where: Ω is a specified d-dimensional domain; ϕ is a scalar function of $\bar{x}\in\Omega\subset\mathbb{R}^d$; $\underline{\alpha}\in\mathbb{C}^{d\times d}$ is a symmetric matrix dependent upon spatial variables $\bar{x}$; β, f, p, γ, and q are complex scalar functions of $\bar{x}$; and $\bar{n}\in\mathbb{R}^d$ is the outward pointing unit normal to Ω. The two boundary components Γ_D and Γ_R are disjoint; that is, $\Gamma_D\cap\Gamma_R=\emptyset$. Furthermore, the boundary of Ω, denoted $\partial\Omega$, is given by the union of the two boundary components Γ_D and Γ_R (which may themselves be composed of disconnected components). Each component is a codimension-1 open set (curves in $\mathbb{R}^2$, surfaces in $\mathbb{R}^3$, etc.). This means that when $\partial\Omega$ is described as the union of Γ_D and Γ_R, the understanding is that $\partial\Omega\backslash(\Gamma_D\cup\Gamma_G)$ may be a nonempty set of codimension-2 (points in $\mathbb{R}^2$, curves in $\mathbb{R}^3$, etc.). That is, $\partial\Omega$ is equivalent to $\Gamma_D\cup\Gamma_G$ modulo a set of codimension-2. This union property combined with the disjoint nature of the boundary components Γ_D and Γ_R will be referred to as a disjoint union in the remainder of the thesis.

This chapter, for the most part, leaves $d>1$ unspecified but uses $d=2$ and occasionally

$d = 3$ as examples. Note that $d = 1$ reduces to the BVP already treated in Chapter 4. To mirror the development in one dimension, this chapter describes how to solve the forward problem: find ϕ given $\underline{\alpha}$, β, f, p, γ, and q as well as a geometric description of Ω, Γ_D, and Γ_R. Equation (6.1) subsumes several interesting PDEs for the electrical engineer. For example, when ϕ is the electrostatic potential, $\underline{\alpha} = \underline{\epsilon}$ is the permittivity tensor, $\beta = 0$, and $f = \rho$ is the volume charge density, then (6.1) corresponds to the differential form of Gauss' law. Solving such an equation (called Poisson's equation) gives insight into the electrostatic behavior of potentially complicated devices. While this is the prototypical example, even time-harmonic wave problems with $d = 2$ fit into this framework. Equation (6.2) corresponds to a Dirichlet boundary condition whereas (6.3) corresponds to a Robin boundary condition (Neumann boundary conditions can be set if $\gamma = 0$). Not all practical problems contain both types of boundary conditions, but both are included for generality.

Higher-dimensional problems admit different approaches towards a finite element method, just as one-dimensional problems did. This chapter considers both the Ritz and Galerkin approaches and highlight their similarities. The development mirrors the one-dimensional development presented in Chapter 3. As in Chapter 3, the material is classical, but presented with an emphasis on constraints which will be useful in subsequent chapters. In addition, comments regarding choices of geometry of the element in higher dimensions which are not present in one dimension are made. These comments motivate the choice to use quadrilateral elements instead of triangular ones for the examples presented in this thesis.

6.1 The Variational Formulation

To use the Ritz approach, we need to write an equivalent variational formulation for the PDE. In particular, we can pose (6.1)-(6.3) as a variational problem

$$\delta F(\phi) = 0, \tag{6.4}$$

$$\phi(\bar{x}) = p(\bar{x}) \qquad \text{for } \bar{x} \in \Gamma_D, \tag{6.5}$$

$$\phi \in H^1(\Omega), \tag{6.6}$$

where

$$F(\phi) = \frac{1}{2}\int_\Omega (\nabla\phi)^T (\underline{\alpha}\nabla\phi) + \beta\phi^2 - 2f\phi \, d\Omega + \frac{1}{2}\int_{\Gamma_R} \gamma\phi^2 - 2q\phi \, d\Omega \tag{6.7}$$

is the functional whose first variation we set to zero and $H^1(\Omega)$ is the space of functions whose gradient is square integrable (for practical purposes we can think of this as the set of functions that are continuous and piecewise smooth). To see that this problem is equivalent

to (6.1)-(6.3), we take the first variation

$$\delta F = \lim_{\epsilon \to 0} \frac{F\left(\phi + \epsilon\, \delta\phi\right) - F\left(\phi\right)}{\epsilon} \tag{6.8}$$

of the functional F where $\delta\phi$ is an arbitrary function in $H^1\left(\Omega\right)$ satisfying $\delta\phi\left(\bar{x}\right) = 0$ for all $\bar{x} \in \Gamma_D$, and ϵ is a small positive real number. Evaluating the functional at $\phi + \epsilon\, \delta\phi$ yields

$$\begin{aligned} F\left(\phi + \epsilon\, \delta\phi\right) = \frac{1}{2} \int_\Omega \left(\nabla\phi + \epsilon\nabla\delta\phi\right)^T \left(\underline{\alpha}\nabla\phi + \epsilon\underline{\alpha}\nabla\delta\phi\right) + \beta\left(\phi + \epsilon\, \delta\phi\right)^2 - 2f\left(\phi + \epsilon\, \delta\phi\right)\, d\Omega \\ + \frac{1}{2} \int_{\Gamma_R} \gamma\left(\phi + \epsilon\, \delta\phi\right)^2 - 2q\left(\phi + \epsilon\, \delta\phi\right)\, d\Omega \end{aligned} \tag{6.9}$$

which, when exploiting the fact that $\underline{\alpha} = \underline{\alpha}^T$, gives

$$\begin{aligned} F\left(\phi + \epsilon\, \delta\phi\right) = \frac{1}{2} \int_\Omega \left(\nabla\phi\right)^T \left(\underline{\alpha}\nabla\phi\right) + \epsilon\left[2\left(\nabla\phi\right)^T \left(\underline{\alpha}\nabla\delta\phi\right)\right] + \epsilon^2\left\{\left(\nabla\delta\phi\right)^T \left(\underline{\alpha}\nabla\delta\phi\right)\right\} \\ + \beta\phi^2 + \epsilon\left[2\beta\phi\, \delta\phi\right] + \epsilon^2\{\beta\, \delta\phi^2\} - 2f\phi - \epsilon\left[2f\, \delta\phi\right]\, d\Omega \\ + \frac{1}{2} \int_{\Gamma_R} \gamma\phi^2 + \epsilon\left[2\gamma\phi\, \delta\phi\right] + \epsilon^2\{\gamma\, \delta\phi^2\} - 2q\phi - \epsilon\left[2q\, \delta\phi\right]\, d\Omega. \end{aligned} \tag{6.10}$$

Notice that all terms multiplied by ϵ are in square brackets, whereas all terms multiplied by ϵ^2 are in curly brackets. All other remaining terms exactly cancel with those in $F\left(\phi\right)$ when taking the difference $F\left(\phi + \epsilon\delta\phi\right) - F\left(\phi\right)$. Dividing this difference by ϵ and evaluating the limit yields

$$\delta F = \frac{1}{2} \int_\Omega 2\left(\nabla\phi\right)^T \underline{\alpha}\nabla\delta\phi + 2\beta\phi\, \delta\phi - 2f\, \delta\phi\, d\Omega + \frac{1}{2} \int_{\Gamma_R} 2\gamma\phi\, \delta\phi - 2q\, \delta\phi\, d\Omega \tag{6.11}$$

$$= \int_\Omega \left(\nabla\phi\right)^T \underline{\alpha}\nabla\delta\phi + \beta\phi\, \delta\phi - f\, \delta\phi\, d\Omega + \int_{\Gamma_R} \gamma\phi\, \delta\phi - q\, \delta\phi\, d\Omega \tag{6.12}$$

which consists of only those terms that were in square brackets (those in curly brackets were all multiplied by a factor of ϵ which went to zero in the limit).

Only the first term in (6.12) does not include a factor of $\delta\phi$, but instead its gradient. We use integration by parts (really a d-dimensional analogue) by combining the divergence theorem and a simple vector identity due to the product rule to isolate $\delta\phi$. In particular, we need the divergence theorem applied to a differentiable vector function $\bar{A}$ given by

$$\int_\Omega \nabla \cdot \bar{A}\, d\Omega = \oint_{\partial\Omega} \bar{n}^T \bar{A}\, d\Omega \tag{6.13}$$

as well as the identity

$$\nabla \cdot (b\bar{A}) = (\nabla b)^T \bar{A} + b\nabla \cdot \bar{A} \tag{6.14}$$

$$= \bar{A}^T \nabla b + (\nabla \cdot \bar{A}) b \tag{6.15}$$

for some arbitrary differentiable scalar function b. Taking $\bar{A} = \underline{\alpha}\nabla\phi$ and $b = \delta\phi$, the product rule becomes

$$\nabla \cdot (\delta\phi \underline{\alpha} \nabla\phi) = (\underline{\alpha}\nabla\phi)^T \nabla\delta\phi + \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \tag{6.16}$$

$$= (\nabla\phi)^T \underline{\alpha}\nabla\delta\phi + \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \tag{6.17}$$

from which we identify the term

$$(\nabla\phi)^T \underline{\alpha}\nabla\delta\phi = \nabla \cdot (\delta\phi \underline{\alpha}\nabla\phi) - \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi. \tag{6.18}$$

Taking the volume integral and applying the divergence theorem to the first term on the right hand side yields

$$\int_\Omega (\nabla\phi)^T \underline{\alpha}\nabla\delta\phi \, d\Omega = \int_\Omega \nabla \cdot (\delta\phi \underline{\alpha}\nabla\phi) - \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega \tag{6.19}$$

$$= \oint_{\partial\Omega} \bar{n}^T (\delta\phi \underline{\alpha}\nabla\phi)\, d\Omega - \int_\Omega \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega \tag{6.20}$$

$$= \oint_{\partial\Omega} \bar{n}^T (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega - \int_\Omega \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega. \tag{6.21}$$

Since the boundary of Ω is assumed to be composed of two disjoint surfaces Γ_D and Γ_R, we can split the surface integral accordingly to obtain

$$\int_\Omega (\nabla\phi)^T \underline{\alpha}\nabla\delta\phi \, d\Omega = \underbrace{\int_{\Gamma_D} \bar{n}^T (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega}_{0} + \int_{\Gamma_R} \bar{n}^T (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega - \int_\Omega \nabla \cdot (\underline{\alpha}\nabla\phi)\, \delta\phi \, d\Omega \tag{6.22}$$

with the first term equal to zero because $\delta\phi = 0$ everywhere on Γ_D. Finally, substituting this expression into (6.12) gives

$$\delta F = \int_\Omega \left[-\nabla \cdot (\underline{\alpha}\nabla\phi) + \beta\phi - f\right] \delta\phi \, d\Omega + \int_{\Gamma_R} \left[\bar{n}^T (\underline{\alpha}\nabla\phi) + \gamma\phi - q\right] \delta\phi \, d\Omega \tag{6.23}$$

and, since our variational formulation states that the first variation vanishes and that this

equation must hold for all possible $\delta\phi$, we recover the two PDE conditions

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta\phi - f = 0 \qquad \text{in } \Omega, \tag{6.24}$$

$$\bar{n}^T (\underline{\alpha} \nabla \phi) + \gamma\phi - q = 0 \qquad \text{on } \Gamma_R. \tag{6.25}$$

In a situation where $\underline{\alpha}$ changes discontinuously, we need to be careful about where we apply the divergence theorem. In particular, the divergence theorem assumes that the vector field in question is continuously differentiable inside the region Ω. When this does not hold, we need to break the integral into two subregions, one on each side of the discontinuity, and apply the divergence theorem on each side separately. This yields, in a simple case with adjacent components (ignoring all other boundary contributions which would be handled in the way already described),

$$\begin{aligned}\int_\Omega (\nabla\phi)^T \underline{\alpha} \nabla\delta\phi \, d\Omega = \int_{\Gamma_+} \bar{n}_+^T (\underline{\alpha}_+ \nabla\phi) \, \delta\phi \, d\Omega + \int_{\Gamma_-} \bar{n}_-^T (\underline{\alpha}_- \nabla\phi) \, \delta\phi \, d\Omega \\ - \int_{\Omega_+} \nabla \cdot (\underline{\alpha}_+ \nabla\phi) \, \delta\phi \, d\Omega - \int_{\Omega_-} \nabla \cdot (\underline{\alpha}_- \nabla\phi) \, \delta\phi \, d\Omega\end{aligned} \tag{6.26}$$

where Ω_+ and Ω_- are the two regions inside which $\underline{\alpha}_+$ and $\underline{\alpha}_-$ are smooth enough and $\Gamma_+ = \Gamma_-$ is the shared boundary along which $\underline{\alpha}$ is discontinuous. Note that ϕ and $\delta\phi$ must be continuous along this boundary in order to belong to $H^1(\Omega)$ and that the unit normal appearing upon use of the divergence theorem always points away from either region. This means that at the boundary $\Gamma_+ = \Gamma_-$ and $\bar{n}_+ = -\bar{n}_-$. This allows us to combine the boundary integrals to obtain

$$\int_\Omega (\nabla\phi)^T \underline{\alpha} \nabla\delta\phi \, d\Omega = \int_{\Gamma_+} \left[\bar{n}_+^T (\underline{\alpha}_+ \nabla\phi) - \bar{n}_+^T (\underline{\alpha}_- \nabla\phi)\right] \delta\phi \, d\Omega - \int_\Omega \nabla \cdot (\underline{\alpha} \nabla\phi) \, \delta\phi \, d\Omega \tag{6.27}$$

where the last integral on the right hand side needs to be understood in a generalized sense. That is, the divergence only exists in a piecewise defined way. In this weak sense, we get the following three conditions when setting the first variation to zero:

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta\phi - f = 0 \qquad \text{in } \Omega, \tag{6.28}$$

$$\bar{n}^T (\underline{\alpha} \nabla \phi) + \gamma\phi - q = 0 \qquad \text{on } \Gamma_R, \tag{6.29}$$

$$\bar{n}_+^T (\underline{\alpha}_+ \nabla\phi) - \bar{n}_+^T (\underline{\alpha}_- \nabla\phi) = 0 \qquad \text{on } \Gamma_+ = \Gamma_-. \tag{6.30}$$

Thus, we satisfy the PDE in the region Ω in a weak sense, and also have continuity of the flux density $\underline{\alpha}\nabla\phi$ in the normal direction across boundaries of discontinuous $\underline{\alpha}$.

6.2 The Ritz Method

If we wish to use this variational framework to find an approximation to the solution of the PDE, we assume ϕ can be expressed as a linear combination of "simple" functions which we collect in the vector $\bar{N}(\bar{x})$. The precise nature of the functions determines the type of method we use to solve the PDE (choosing functions—typically polynomials—that are nonzero only on small subdomains whose disjoint union spans Ω yields a finite element method). Without specifying the exact choice of functions, we have

$$\phi(\bar{x}) = \bar{\phi}^T \bar{N}(\bar{x}) \tag{6.31}$$

$$= \bar{N}(\bar{x})^T \bar{\phi} \tag{6.32}$$

where $\bar{\phi} \in \mathbb{C}^M$ is a vector of coefficients which we seek. Substituting this expression into the functional (6.7) yields

$$F(\phi) = \frac{1}{2}\int_\Omega \left(\underline{J}_{\bar{N}}(\bar{x})^T \bar{\phi}\right)^T \left(\underline{\alpha}\underline{J}_{\bar{N}}(\bar{x})^T \bar{\phi}\right) + \bar{\phi}^T \bar{N}(\bar{x})\,\beta \bar{N}(\bar{x})^T \bar{\phi} - 2\bar{\phi}^T \bar{N}(\bar{x})\, f\, d\Omega$$
$$+ \frac{1}{2}\int_{\Gamma_R} \bar{\phi}^T \bar{N}(\bar{x})\,\gamma \bar{N}(\bar{x})^T \bar{\phi} - 2\bar{\phi}^T \bar{N}(\bar{x})\, q\, d\Omega \tag{6.33}$$

where $\underline{J}_{\bar{N}}(\bar{x}) \in \mathbb{R}^{M\times d}$ is the Jacobian matrix with entries

$$\left[\underline{J}_{\bar{N}}(\bar{x})\right]_{ij} = \frac{\partial N_i}{\partial x_j}(\bar{x}), \tag{6.34}$$

N_i is the ith function in the vector $\bar{N}(\bar{x})$, and x_j is the jth spatial variable in d dimensions. Collecting quadratic and linear terms in $\bar{\phi}$ yields

$$F(\bar{\phi}) = \frac{1}{2}\bar{\phi}^T \underbrace{\left[\int_\Omega \underline{J}_{\bar{N}}(\bar{x})\,\underline{\alpha}\underline{J}_{\bar{N}}(\bar{x})^T\, d\Omega + \int_\Omega \bar{N}(\bar{x})\,\beta\bar{N}(\bar{x})^T\, d\Omega + \int_{\Gamma_R} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\, d\Omega\right]}_{\underline{A}} \bar{\phi}$$
$$- \bar{\phi}^T \underbrace{\left[\int_\Omega \bar{N}(\bar{x})\, f\, d\Omega + \int_{\Gamma_R} \bar{N}(\bar{x})\, q\, d\Omega\right]}_{\bar{b}}. \tag{6.35}$$

That is, once the solution is represented as a linear combination of functions in $\bar{N}(\bar{x})$, the functional becomes a quadratic function of the coefficients $\bar{\phi}$ in the combination. As in the one-dimensional case, if the functions are not continuous over the whole domain Ω, we need to explicitly enforce continuity. In addition, we also need to enforce Dirichlet boundary

conditions. As we will see, this can be accomplished using a constraint matrix $\underline{C}$ such that

$$\underline{C}\bar{\phi} = \bar{d}. \tag{6.36}$$

Note that by introducing Lagrange multipliers $\bar{\nu}$, we can find stationary points of the discretized functional subject to the continuity and Dirichlet constraints by solving the saddle point system

$$\begin{bmatrix} \underline{A} & \underline{C}^T \\ \underline{C} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \bar{d} \end{bmatrix}. \tag{6.37}$$

It is important to understand that this saddle point system and the one presented in (3.38) differ by the contents of their submatrices and subvectors but that the process to derive them is identical (we start form the Lagrangian and compute its gradient with respect to $\bar{\phi}$ and $\bar{\nu}$).

6.3 The Galerkin Method

As in the one-dimensional case, there is an analogous interpretation of the saddle point system that can be obtained through Galerkin's method of weighted residuals. In particular, we can derive meaning for the Lagrange multipliers and obtain, at least from an abstract point of view, the specific form of the constraint matrix $\underline{C}$. To do so, we begin by integrating the product of (6.1) with a test function ψ to obtain

$$\int_\Omega \psi \left[-\nabla \cdot (\underline{\alpha}\nabla\phi) + \beta\phi\right] d\Omega = \int_\Omega \psi f \, d\Omega \tag{6.38}$$

$$-\int_\Omega \psi \nabla \cdot (\underline{\alpha}\nabla\phi)\, d\Omega + \int_\Omega \psi\beta\phi \, d\Omega = \int_\Omega \psi f \, d\Omega. \tag{6.39}$$

Applying (6.18) with ψ in place of $\delta\phi$ yields

$$-\psi\nabla \cdot (\underline{\alpha}\nabla\phi) = (\nabla\phi)^T \underline{\alpha}\nabla\psi - \nabla \cdot (\psi\underline{\alpha}\nabla\phi) \tag{6.40}$$

so that the weighted equation becomes

$$\int_\Omega (\nabla\phi)^T \underline{\alpha}\nabla\psi \, d\Omega - \int_\Omega \nabla \cdot (\psi\underline{\alpha}\nabla\phi)\, d\Omega + \int_\Omega \psi\beta\phi \, d\Omega = \int_\Omega \psi f \, d\Omega. \tag{6.41}$$

Using the divergence theorem on the second integral gives

$$\int_\Omega (\nabla\phi)^T \underline{\alpha}\nabla\psi \, d\Omega - \oint_{\partial\Omega} \psi\bar{n}^T (\underline{\alpha}\nabla\phi)\, d\Omega + \int_\Omega \psi\beta\phi \, d\Omega = \int_\Omega \psi f \, d\Omega \tag{6.42}$$

and, using the Robin boundary condition (6.3), we obtain

$$\int_\Omega (\nabla\phi)^T \underline{\alpha}\nabla\psi \, d\Omega - \int_{\Gamma_D} \psi \bar{n}^T (\underline{\alpha}\nabla\phi) \, d\Omega - \int_{\Gamma_R} \psi \left[q - \gamma\phi\right] d\Omega + \int_\Omega \psi\beta\phi \, d\Omega = \int_\Omega \psi f \, d\Omega \quad (6.43)$$

which we rewrite as

$$\int_\Omega (\nabla\psi)^T \underline{\alpha}\nabla\phi \, d\Omega + \int_\Omega \psi\beta\phi \, d\Omega + \int_{\Gamma_R} \psi\gamma\phi \, d\Omega - \int_{\Gamma_D} \psi \bar{n}^T (\underline{\alpha}\nabla\phi) \, d\Omega = \int_\Omega \psi f \, d\Omega + \int_{\Gamma_R} \psi q \, d\Omega. \quad (6.44)$$

By making the substitution

$$\phi = \bar{N}(\bar{x})^T \bar{\phi} \quad (6.45)$$

in (6.44) and repeating the equation M times with $\psi = N_i(\bar{x})$ for $i = 1, 2, ..., M$, we obtain the equations

$$\underbrace{\left[\int_\Omega \underline{J}_{\bar{N}}(\bar{x}) \, \underline{\alpha} \underline{J}_{\bar{N}}(\bar{x})^T \, d\Omega + \int_\Omega \bar{N}(\bar{x}) \, \beta \bar{N}(\bar{x})^T \, d\Omega + \int_{\Gamma_R} \bar{N}(\bar{x}) \, \gamma \bar{N}(\bar{x})^T \, d\Omega\right]}_{\underline{A}} \bar{\phi}$$

$$- \int_{\Gamma_D} \bar{N}(\bar{x}) \, \bar{n}^T (\underline{\alpha}\nabla\phi) \, d\Omega = \underbrace{\left[\int_\Omega \bar{N}(\bar{x}) \, f \, d\Omega + \int_{\Gamma_R} \bar{N}(\bar{x}) \, q \, d\Omega\right]}_{\bar{b}}. \quad (6.46)$$

To determine the nature of the remaining boundary integral term, suppose that we expand the normal flux density as

$$-\bar{n}^T (\underline{\alpha}\nabla\phi) = \bar{N}_B(\bar{x})^T \bar{\nu} \quad (6.47)$$

using its own set of basis functions $\bar{N}_B(\bar{x})$. Then the weighted equations become

$$\underline{A}\bar{\phi} - \int_{\Gamma_D} \bar{N}(\bar{x}) \, \bar{n}^T (\underline{\alpha}\nabla\phi) \, d\Omega = \bar{b} \quad (6.48)$$

$$\underline{A}\bar{\phi} + \underbrace{\int_{\Gamma_D} \bar{N}(\bar{x}) \, \bar{N}_B(\bar{x})^T \, d\Omega}_{\underline{C}^T} \bar{\nu} = \bar{b}. \quad (6.49)$$

Thus, the Lagrange multipliers $\bar{\nu}$ are precisely the coefficients in a linear combination of basis functions $\bar{N}_B$ used to expand the normal flux density $-\bar{n}^T (\underline{\alpha}\nabla\phi)$ at the Dirichlet boundary. Under this interpretation, we obtain the equation $\underline{A}\bar{\phi} + \underline{C}^T\bar{\nu} = \bar{b}$ which arises as the first block row of the saddle point system (6.37).

In fact, this gives us a natural way to interpret the Dirichlet boundary condition which must be explicitly enforced. For symmetry, we enforce the Dirichlet boundary condition

(6.2) in the weak sense

$$\int_{\Gamma_D} \bar{N}_B(\bar{x})\,[\phi - p]\,d\Omega = 0 \tag{6.50}$$

$$\int_{\Gamma_D} \bar{N}_B(\bar{x})\,\phi\,d\Omega = \int_{\Gamma_D} \bar{N}_B(\bar{x})\,p\,d\Omega. \tag{6.51}$$

Making the substitution $\phi = \bar{N}(\bar{x})^T\,\bar{\phi}$ yields

$$\underbrace{\int_{\Gamma_D} \bar{N}_B(\bar{x})\,\bar{N}(\bar{x})^T\,d\Omega}_{\underline{C}}\,\bar{\phi} = \underbrace{\int_{\Gamma_D} \bar{N}_B(\bar{x})\,p\,d\Omega}_{\bar{d}} \tag{6.52}$$

which was precisely the form of the constraint equation $\underline{C}\bar{\phi} = \bar{d}$ in the second block row of (6.37).

In fact, the same saddle point form holds true for situations where $\underline{\alpha}$ is discontinuous. In such cases, we gain pairs of terms

$$\int_{\Gamma_+} \psi\left[\bar{n}_+^T(\underline{\alpha}_+\nabla\phi) - \bar{n}_+^T(\underline{\alpha}_-\nabla\phi)\right]\,d\Omega \tag{6.53}$$

in the Galerkin weighted residual which give rise to additional columns in $\underline{C}^T$. We then enforce continuity of ϕ explicitly along those boundaries resulting in additional rows of constraint equations

$$\int_{\Gamma_+} \bar{N}_+(\bar{x})\,[\phi_+ - \phi_-]\,d\Omega = 0 \tag{6.54}$$

in the matrix $\underline{C}$. As we will see in later chapters, this particular weak form of continuity enforcement can become quite powerful when considering nonconforming finite element methods.

6.4 The Canonical Element in Higher Dimensions

Recall that to solve the prototypical BVP (3.1)-(3.3) using a finite element method, we partitioned the one-dimensional domain $\Omega = (a, b)$ into a disjoint union of subintervals (which we called elements). This was done so that ϕ could be approximated on each element using a polynomial basis. In higher dimensions, the choice of element becomes less clear. In particular, there is no unique generalization of the canonical interval $(-1, 1)$. For example, one could think of the reference domain as a one-dimensional ball of radius 1, in which case,

the generalization to higher dimensions is the domain

$$B_p(0,1) = \left\{ \bar{x} \in \mathbb{R}^d \,:\, \|\bar{x}\|_p < 1 \right\} \tag{6.55}$$

where $1 \leq p \leq \infty$. When $d = 1$, the set $\|\bar{x}\|_p < 1$ is $-1 < x_1 < 1$ for all p. In higher dimensions, the choice of p changes the geometry of $B_p(0,1)$ substantially. In fact, only $p \to \infty$ (a hypercube) or the intersection of $B_\infty(0,1)$ and $\bar{e}^T\bar{x} < 2-d$ (where $\bar{e}$ is the vector of all ones giving rise to a simplex) find widespread use (actually we can rephrase everything that applies for $p \to \infty$ in terms of $p = 1$ because these norms are dual). Other cases of p do not work because, in general, tiling $\mathbb{R}^d$ without overlap is not possible (think of tiling $\mathbb{R}^2$ using only disks corresponding to $p = 2$); only the 1-norm and infinity-norm unit balls have planar faces. To make matters worse, the choice of element need not be restricted to hypercubes or simplices. For example, recently, there has been a trend toward polyhedral elements [137, 138, 139]. Faced with this myriad of possible element choices, one needs to carefully consider their pros and cons.

First, in dimensions $d > 1$ there is the difficulty of partitioning the domain Ω that must be considered. In one dimension, partitioning the domain $\Omega = (a,b)$ into disjoint subdomains is straightforward; simply split the interval into disjoint subintervals which are all affine transformations of the canonical interval $(-1,1)$. In higher dimensions (say $d = 2,3$), simplices have historically been favored as there exist robust algorithms for producing unstructured triangulations of d-dimensional domains [140]. Each resulting simplex in the triangulation can then be interpreted as the image of an affine map from a canonical simplex (for example, the intersection of $B_\infty(0,1)$ and $\bar{e}^T\bar{x} < 2-d$, although there is no universally agreed upon canonical simplex). Hypercube meshes have lagged behind in this respect but there are several approaches that find use (advancing front methods, grid superposition methods, hypercubes by means of simplex combination, etc.) [62][1].

On the other hand, since we are interested in extending the framework of Chapter 4, in extending ideas to higher dimensions, we must also consider which types of elements preserve the properties observed in one dimension. The key properties exploited for one-dimensional problems are the ability to analytically compute the integral of triple products of Legendre polynomials, as well as to produce Legendre expansions of smooth functions to high accuracy. Thus, to extend such an approach to the hypercube or simplex, we need to consider orthogonal polynomials on these domains. To gauge whether the hypercube or the simplex may be better suited to such a treatment, we consider comparing orthogonal

[1] I am not aware of automatic mesh generation for arbitrary polyhedral elements beyond using Voronoi diagrams.

polynomials in $d = 2$ dimensions on the square

$$H_2 = \left\{ \bar{x} \in \mathbb{R}^{2\times 1} \,:\, \|\bar{x}\|_\infty < 1 \right\} \tag{6.56}$$

$$= (-1,1)^2 \tag{6.57}$$

and the triangle

$$S_2 = \left\{ \bar{x} \in \mathbb{R}^{2\times 1} \,:\, \|\bar{x}\|_\infty < 1,\, \bar{e}^T \bar{x} < 0 \right\} \tag{6.58}$$

$$= (-1,1)^2 \cap \{ x_1, x_2 \in \mathbb{R} \,:\, x_1 + x_2 < 0 \}\,. \tag{6.59}$$

On the square H_2, the polynomials

$$p_{ij}\left(\bar{x}\right) = p_i\left(x_1\right) p_j\left(x_2\right) \tag{6.60}$$

are orthonormal where p with a single subscript is the usual orthonormal Legendre polynomial on the interval $(-1,1)$. That is, the two-dimensional polynomials p_{ij} and $p_{i'j'}$ satisfy the orthogonality property

$$\int_{H_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) d\Omega = \delta_{ii'}\delta_{jj'} \tag{6.61}$$

which we obtain through separation of variables:

$$\int_{H_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) d\Omega = \int_{-1}^{1}\int_{-1}^{1} p_i\left(x_1\right) p_j\left(x_2\right) p_{i'}\left(x_1\right) p_{j'}\left(x_2\right) dx_1 dx_2 \tag{6.62}$$

$$= \underbrace{\int_{-1}^{1} p_i\left(x_1\right) p_{i'}\left(x_1\right) dx_1}_{\delta_{ii'}} \underbrace{\int_{-1}^{1} p_j\left(x_2\right) p_{j'}\left(x_2\right) dx_2}_{\delta_{jj'}}\,. \tag{6.63}$$

In contrast, the orthonormal polynomials on the triangle S_2 are given by

$$p_{ij}\left(\bar{x}\right) = \sqrt{2} p_i\left(2\frac{1+x_1}{1-x_2} - 1\right) p_j^{(2i+1,0)}\left(x_2\right)\left(1-x_2\right)^i \tag{6.64}$$

where $p_j^{(2i+1,0)}\left(x_2\right)$ is the jth orthonormal Jacobi polynomial on the interval $(-1,1)$ with respect to weight function $(1-x_2)^{2i+1}$ (see [38, 141, 142, 143], for example). To see that these polynomials are orthogonal on S_2, we compute

$$\int_{S_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) d\Omega \tag{6.65}$$

by performing the change of variables

$$y_1 = 2\frac{1+x_1}{1-x_2} - 1, \tag{6.66}$$

$$y_2 = x_2, \tag{6.67}$$

whose Jacobian is given by

$$\underline{J}_{\bar{x}} = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{bmatrix}. \tag{6.68}$$

This transformation takes the right triangle S_2 to the square H_2. Rewriting the change of variables for x_1 and x_2 in terms of y_1 and y_2 yields

$$x_1 = \frac{(y_1+1)(1-y_2)}{2} - 1, \tag{6.69}$$

$$x_2 = y_2, \tag{6.70}$$

which gives the corresponding Jacobian

$$\underline{J}_{\bar{x}} = \begin{bmatrix} \frac{1-y_2}{2} & -\frac{y_1+1}{2} \\ 0 & 1 \end{bmatrix} \tag{6.71}$$

whose determinant is

$$\det\left(\underline{J}_{\bar{x}}\right) = \frac{1-y_2}{2}. \tag{6.72}$$

The orthogonality test becomes

$$\int_{S_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) d\Omega = \int_{H_2} p_{ij}\left(\bar{y}\right) p_{i'j'}\left(\bar{y}\right) \det\left(\underline{J}_{\bar{x}}\right) d\Omega, \tag{6.73}$$

or, more explicitly,

$$\int_{-1}^{1}\int_{-1}^{1} \sqrt{2}p_i\left(y_1\right) p_j^{(2i+1,0)}\left(y_2\right)\left(1-y_2\right)^i \sqrt{2}p_{i'}\left(y_1\right) p_{j'}^{(2i'+1,0)}\left(y_2\right)\left(1-y_2\right)^{i'} \frac{1-y_2}{2} dy_1 dy_2 =$$
$$\underbrace{\int_{-1}^{1} p_i\left(y_1\right) p_{i'}\left(y_1\right) dy_1}_{\delta_{ii'}} \int_{-1}^{1} p_j^{(2i+1,0)}\left(y_2\right) p_{j'}^{(2i'+1,0)}\left(y_2\right)\left(1-y_2\right)^{i+i'+1} dy_2 \tag{6.74}$$

which demonstrates that under this change of variables, polynomials (6.64) become separable. When $i \neq i'$, the integral is zero, regardless of the value of the integral with respect to y_2.

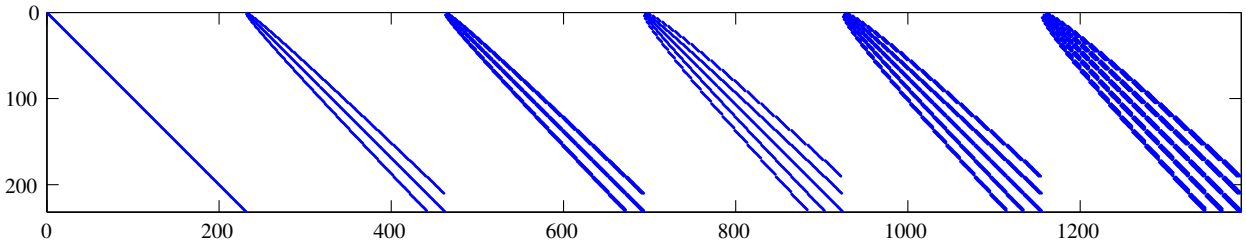

Figure 6.1: The first six frontal slices of the triple product integral tensor for the simplex corresponding to a degree 2 polynomial for the material coefficients and a degree twenty polynomial for basis functions on the triangle.

Assuming $i = i'$, we obtain

$$\int_{S_2} p_{ij}\left(\bar{x}\right) p_{ij'}\left(\bar{x}\right) d\Omega = \underbrace{\int_{-1}^{1} p_j^{(2i+1,0)}\left(y_2\right) p_{j'}^{(2i+1,0)}\left(y_2\right)\left(1-y_2\right)^{2i+1} dy_2}_{\delta_{jj'}} \tag{6.75}$$

and thus

$$\int_{S_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) d\Omega = \delta_{ii'}\delta_{jj'}. \tag{6.76}$$

Recall however that the sparsity preserving schemes developed in one dimension in Chapter 4 relied on properties of triple products of orthonormal polynomials. Under the same change of variables, the triple product on the triangle S_2 yields

$$\begin{aligned}\int_{S_2} p_{ij}\left(\bar{x}\right) p_{i'j'}\left(\bar{x}\right) p_{i''j''}\left(\bar{x}\right) d\Omega = \sqrt{2}\int_{-1}^{1} p_i\left(y_1\right) p_{i'}\left(y_1\right) p_{i''}\left(y_1\right) dy_1 \\ \cdot \int_{-1}^{1} p_j^{(2i+1,0)}\left(y_2\right) p_{j'}^{(2i'+1,0)}\left(y_2\right) p_{j''}^{(2i''+1,0)}\left(y_2\right)\left(1-y_2\right)^{i+i'+i''+1} dy_2\end{aligned} \tag{6.77}$$

with the integral with respect to y_1 giving the triple product of Legendre polynomials as in the one-dimensional case. However, the second integral is a great deal more complicated (see [144] for exact expressions[2]). See Figure 6.1 for an example of the sparsity of the triple product integral tensor for the triangle (analogous to the tensor computed for one-dimensional Legendre polynomials) and Figure 6.2 for the sum of the frontal slices of those tensors emulating a variable coefficient problem. The entries in the tensor were computed via Gauss-Legendre quadrature and set to zero whenever a resulting integral's absolute value was less than an absolute tolerance of 10^{-12}.

The difficulty in evaluating the triple products becomes even more pronounced when considering the orthogonal polynomials on the d-dimensional simplex S_d [39]. There, using

[2]My own direct implementation of the formulae contained in that reference is not numerically stable.

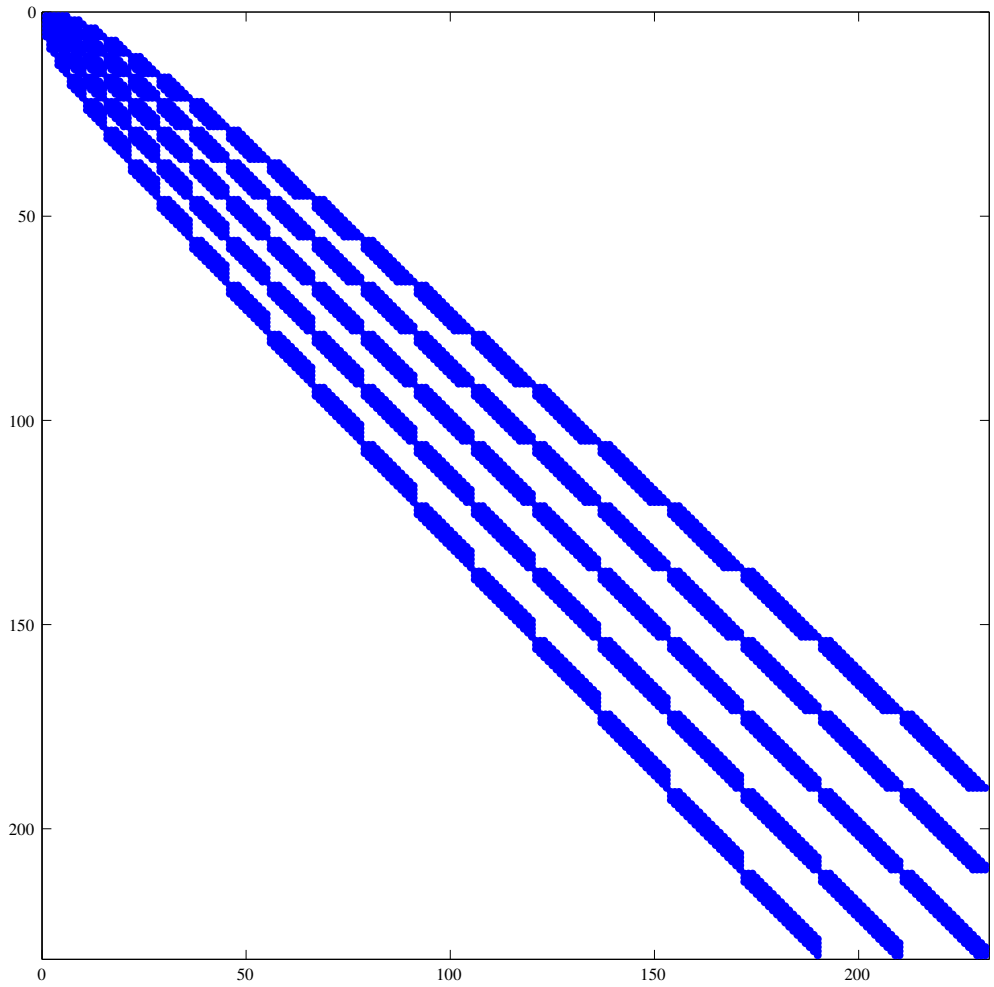

Figure 6.2: Sum of the first six frontal slices from Figure 6.1.

our notation, the polynomials are given by

$$p_{\bar{\alpha}}\left(\bar{z}\right) = N\left(\bar{\alpha}, d\right) \prod_{j=1}^{d} \left(\frac{2 - j - \sum_{i=1}^{j} z_j}{3 - j - \sum_{i=1}^{j-1} z_j} \right)^{\sum_{i=j+1}^{d} \alpha_i} p_{\alpha_j}^{\left(2\sum_{i=j+1}^{d} \alpha_i + d - j, 0\right)} \left(2 \frac{z_j + 1}{3 - j - \sum_{i=1}^{j-1} z_j} - 1 \right) \tag{6.78}$$

where N is a normalization constant, $\bar{\alpha}$ is a vector of indices indexing the multivariate polynomials, and empty summations are taken to be zero. To recover the two-dimensional case, substitute $z_j \to x_{d+1-j}$ and $\alpha_1 \to j$, $\alpha_2 \to i$. If we take $d = 3$ for example (see [37, 145]), then we need to perform integrals of the form

$$\int_{-1}^{1} p_i\left(y_1\right) p_{i'}\left(y_1\right) p_{i''}\left(y_1\right) dy_1, \tag{6.79}$$

$$\int_{-1}^{1} p_j^{(2i+1,0)}(y_2)\, p_{j'}^{(2i'+1,0)}(y_2)\, p_{j''}^{(2i''+1,0)}(y_2)\,(1-y_2)^{i+i'+i''+1}\, dy_2, \tag{6.80}$$

$$\int_{-1}^{1} p_k^{(2i+2j+2,0)}(y_3)\, p_{k'}^{(2i'+2j'+2,0)}(y_3)\, p_{k''}^{(2i''+2j''+2,0)}(y_3)\,(1-y_3)^{j+k+j'+k'+j''+k''+2}\, dy_3, \tag{6.81}$$

in order to populate the triple product tensor. These integrals can be computed using Gauss-Legendre quadrature, but this can compromise the efficiency of our method. In addition, the integrals increase in number and difficulty for dimensions four and higher.

In contrast, the orthonormal polynomials on hypercubes are

$$p_{\bar{\alpha}}(\bar{x}) = \prod_{j=1}^{d} p_{\alpha_j}(x_j), \tag{6.82}$$

giving rise to triple product integrals

$$\int_{-1}^{1} p_i(x_j)\, p_{i'}(x_j)\, p_{i''}(x_j)\, dx_j, \qquad j = 1, ..., d. \tag{6.83}$$

Thus, computing the triple product integrals of orthonormal polynomials for a d-dimensional hypercube only requires the computation of a single set of one-dimensional triple product integrals due to the simple separable form of the orthonormal polynomials (they are the tensor product of one-dimensional polynomials). This means that the triple product tensor can be assembled using the entries of the one-dimensional triple product tensor $\underline{\underline{T}}$ from Section 4.3. It is primarily for this simplicity that we choose to focus on the hypercube for our extension of the finite element method described in Chapter 4. In the upcoming chapters, we use the two-dimensional case ($d = 2$) as an example to show how this extension can be made possible.

Chapter 7

A Single Square Element

Chapter 6 avoided specifying the dimension d of the problem, the geometry of the domain Ω, the types of basis functions $\bar{N}(\bar{x})$ used to approximate ϕ, and the types of weight functions $\bar{N}_B(\bar{x})$ used to impose boundary constraints. This chapter specializes to the $d = 2$ domain $\Omega = (-1, 1)^2$. In later chapters, this domain will be used as a canonical domain for special curvilinear domains, as well as for subdomains in a finite element method. This chapter focuses on the two-dimensional case to highlight key differences between solving $d = 1$ and $d > 1$ PDEs. Because of the separability of orthonormal polynomials on the hypercube, the techniques outlined in this chapter apply with only minor modification in higher dimensions (one can think of the $(d+1)$-dimensional hypercube as the tensor product of a d-dimensional hypercube and a 1-dimensional hypercube). The two-dimensional case is chosen to concretely describe all involved computations in the simplest way possible for $d > 1$ applications.

7.1 Problem Specification and Basis Functions

In this chapter, we look to solve

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi = f, \qquad \bar{x} \in (-1, 1)^2, \tag{7.1}$$

subject to boundary conditions

$$\phi = p \qquad \text{on } \Gamma_D, \tag{7.2}$$

$$\bar{n}^T (\underline{\alpha} \nabla \phi) + \gamma \phi = q \qquad \text{on } \Gamma_R, \tag{7.3}$$

where $\bar{n}$ is the outward pointing unit normal to the square. For now, we reserve choosing which portions of the boundary $\partial\Omega$ belong to the Dirichlet boundary component Γ_D and to

the Robin boundary component Γ_R. Note that $\partial\Omega$ can be described as the disjoint union of boundary components Γ_1, Γ_2, Γ_3, and Γ_4, which are the sets

$$\Gamma_1 = \left\{\bar{x} \in \mathbb{R}^2 : x_1 = -1,\, x_2 \in (-1,1)\right\}, \tag{7.4}$$

$$\Gamma_2 = \left\{\bar{x} \in \mathbb{R}^2 : x_1 = +1,\, x_2 \in (-1,1)\right\}, \tag{7.5}$$

$$\Gamma_3 = \left\{\bar{x} \in \mathbb{R}^2 : x_1 \in (-1,1),\, x_2 = -1\right\}, \tag{7.6}$$

$$\Gamma_4 = \left\{\bar{x} \in \mathbb{R}^2 : x_1 \in (-1,1),\, x_2 = +1\right\}. \tag{7.7}$$

Boundary component Γ_1 corresponds to the left edge, Γ_2 to the right edge, Γ_3 to the bottom edge, and Γ_4 to the top edge of Ω. We assume that Γ_D and Γ_R satisfy $\partial\Omega = \Gamma_D \cup \Gamma_R$, $\Gamma_D \cap \Gamma_R = \emptyset$, and that they are each comprised of a union of a subset of Γ_1, Γ_2, Γ_3, and Γ_4. This last assumption eliminates the possibility of a single edge sharing more than one type of boundary condition.

Applying either the Ritz or the Galerkin approach, we expand the unknown function ϕ using a tensor product of integrated Legendre polynomials. This means that the vector of basis functions can be written as

$$\bar{N}(\bar{x}) = \bar{N}(x_2) \otimes \bar{N}(x_1) \tag{7.8}$$

where $\bar{N}(x_j) = \underline{\tilde{S}}\bar{p}(x_j)$ for $j = 1, 2$. Note that we could have just as easily interchanged the order of $\bar{N}(x_1)$ and $\bar{N}(x_2)$ in the tensor product; the choice is arbitrary although we will see later that it is not without consequence. As in Chapter 5, in the coming sections we make extensive use of the three properties of the Kronecker product given by (5.11), (5.12), and (5.13). Due to (5.12), we can write the basis functions as

$$\bar{N}(\bar{x}) = \underline{\tilde{S}}\bar{p}(x_2) \otimes \underline{\tilde{S}}\bar{p}(x_1) \tag{7.9}$$

$$= \left(\underline{\tilde{S}} \otimes \underline{\tilde{S}}\right) \underbrace{\left(\bar{p}(x_2) \otimes \bar{p}(x_1)\right)}_{\bar{p}(\bar{x})}. \tag{7.10}$$

Note that in the rest of this chapter, matrices or vectors that appear on the left of the Kronecker product symbol $\otimes$ are associated with operations regarding the variable x_2 whereas matrices and vectors on the right are related to operations regarding the variable x_1. If we allow for the possibility of different polynomial degrees L_{x_j} in the x_j directions then matrices have dimensions $(L_{x_j}+1) \times (L_{x_j}+1)$ and vectors have dimensions $(L_{x_j}+1) \times 1$. That is, even though we do not often specify the size of matrices, matrices appearing to the left of the $\otimes$ symbol may be a different size than those on the right. In all cases, the represented

matrix products are always conforming and well-defined.

Using such a basis, we can write ϕ as a linear combination of basis functions:

$$\phi(\bar{x}) = \bar{\phi}^T \left(\bar{N}(x_2) \otimes \bar{N}(x_1) \right) \tag{7.11}$$

$$= \left(\bar{N}(x_2)^T \otimes \bar{N}(x_1)^T \right) \bar{\phi}. \tag{7.12}$$

One useful representation of ϕ arises if we define the matrix $\underline{\phi} \in \mathbb{C}^{(L_{x_1}+1)\times(L_{x_2}+1)}$ (in higher dimensions this matrix is actually a d-dimensional tensor) such that $\text{vec}(\underline{\phi}) = \bar{\phi}$. Then, by property (5.13),

$$\phi(\bar{x}) = \left(\bar{N}(x_2)^T \otimes \bar{N}(x_1)^T \right) \text{vec}(\underline{\phi}) \tag{7.13}$$

$$= \text{vec}\left(\bar{N}(x_1)^T \underline{\phi} \bar{N}(x_2) \right) \tag{7.14}$$

$$= \bar{N}(x_1)^T \underline{\phi} \bar{N}(x_2) \tag{7.15}$$

where we can remove the vectorization operator in the final equation since the result is a scalar. This representation is particularly useful once we have solved for $\bar{\phi}$ (and by proxy $\underline{\phi}$) when evaluating the solution at arbitrary points $\bar{x}$ (we simply evaluate this "quadratic form"). In addition, examining the decay of entries in the matrix $\underline{\phi}$ can reveal how accurate a given solution is (doing so is also possible with the vector $\bar{\phi}$ but requires careful indexing).

7.2 Legendre Expansions for Spatially Varying Coefficients

In addition to these expressions for the potential ϕ, we also need to represent the functions $\underline{\alpha}$, β, f, γ, q, and p in terms of polynomials. This allows us to explicitly calculate the matrices $\underline{A}$ and $\underline{C}$, as well as the vectors $\bar{b}$ and $\bar{d}$ for the saddle point system. In Chapter 5, we accomplished this by computing a Legendre expansion for each given function. For functions γ, p, and q which need only be defined on boundaries, we continue to use such expansions. That is, we construct expansions of type

$$\gamma(x_j) = \sum_{k=0}^{K_\gamma} \gamma_k p_k(x_j), \tag{7.16}$$

$$q(x_j) = \sum_{k=0}^{K_q} q_k p_k(x_j), \tag{7.17}$$

for a given edge requiring a Robin boundary condition, or

$$p\left(x_j\right) = \sum_{k=0}^{K_p} \hat{p}_k p_k\left(x_j\right) \tag{7.18}$$

for a given edge requiring a Dirichlet boundary condition. We compute the expansion coefficients using the methods of Section 4.3 and Appendix A. Note that since there are four boundary edges, we need up to four sets of coefficients (the particular sets depend on the problem specification).

This leaves the task of representing $\underline{\alpha}$, β, and f, which are all functions of two variables, rather than a single variable. For the square domain, a natural approach towards representing these functions (which extends the one-dimensional approach) is to seek expansions of the form

$$f\left(\bar{x}\right) = \sum_{i,j} \hat{f}_{ij} p_i\left(x_1\right) p_j\left(x_2\right) \tag{7.19}$$

$$= \bar{p}\left(x_1\right)^T \underline{\hat{F}} \bar{p}\left(x_2\right). \tag{7.20}$$

In this thesis, we go one step further and require that $\underline{\hat{F}}$ be expressed in the form

$$\underline{\hat{F}} = \underline{U}\underline{\Sigma}\underline{V}^T \tag{7.21}$$

where $\underline{\Sigma} \in \mathbb{C}^{K\times K}$ and is a diagonal matrix, $\underline{U} \in \mathbb{C}^{(K_{x_1}+1)\times K}$, and $\underline{V} \in \mathbb{C}^{(K_{x_2}+1)\times K}$. While this looks like a rank K singular value decomposition of $\underline{\hat{F}}$, we do not require the columns of $\underline{U}$ and $\underline{V}$ to be orthonormal, nor do we require the entries on the diagonal of $\underline{\Sigma}$ to decay. Why the requirement (7.21) is necessary will only become clear once we discuss computing the entries of $\underline{A}$ or $\bar{b}$ involving terms associated with $\nabla\phi^T\underline{\alpha}\nabla\phi$, $\phi\beta\phi$, or ϕf. For now, this requirement allows us to write

$$\underline{\hat{F}} = \underbrace{\begin{bmatrix} \bar{u}_1 & \bar{u}_2 & \cdots & \bar{u}_K \end{bmatrix}}_{\underline{U}} \underbrace{\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_K \end{bmatrix}}_{\underline{\Sigma}} \underbrace{\begin{bmatrix} \bar{v}_1^T \\ \bar{v}_2^T \\ \vdots \\ \bar{v}_K^T \end{bmatrix}}_{\underline{V}^T} \tag{7.22}$$

$$= \sum_{k=1}^{K} \sigma_k \bar{u}_k \bar{v}_k^T. \tag{7.23}$$

Combining (7.20) and (7.23) yields

$$f(\bar{x}) = \bar{p}(x_1)^T \left[\sum_{k=1}^{K} \sigma_k \bar{u}_k \bar{v}_k^T\right] \bar{p}(x_2) \tag{7.24}$$

$$= \sum_{k=1}^{K} \sigma_k \underbrace{\bar{p}(x_1)^T \bar{u}_k}_{f_{k,x_1}(x_1)} \underbrace{\bar{v}_k^T \bar{p}(x_2)}_{f_{k,x_2}(x_2)}. \tag{7.25}$$

In words, we have written the two-dimensional function $f(\bar{x})$ as the sum of K separable functions $f_{k,x_1}(x_1) f_{k,x_2}(x_2)$ and refer to such a function as a rank K function. Occasionally, it is useful to write

$$\text{vec}[f(\bar{x})] = \text{vec}\left[\bar{p}(x_1)^T \underline{\hat{F}} \bar{p}(x_2)\right] \tag{7.26}$$

$$f(\bar{x}) = \text{vec}\left[\bar{p}(x_1)^T \left(\underline{U}\underline{\Sigma}\underline{V}^T\right) \bar{p}(x_2)\right] \tag{7.27}$$

$$= \left(\bar{p}(x_2)^T \otimes \bar{p}(x_1)^T\right) \text{vec}\left(\underline{U}\underline{\Sigma}\underline{V}^T\right) \tag{7.28}$$

where we have used property (5.13).

We use two distinct methods to compute the factors $\underline{U}$, $\underline{\Sigma}$, and $\underline{V}$. The first approach, which we call direct, is useful when K_{x_1} and K_{x_2} are small, or when K is known to be comparable in size with $K_{x_1}$and K_{x_2}. The idea is to first compute $\underline{\hat{F}}$ via numerical integration, then to compute the factors $\underline{U}$, $\underline{\Sigma}$, and $\underline{V}$ via singular value decomposition of $\underline{\hat{F}}$. To determine the entries of $\underline{\hat{F}}$, consider multiplying (7.19) by $p_{i'}(x_1) p_{j'}(x_2)$ and integrating over Ω. The result is

$$\int_{-1}^{1}\int_{-1}^{1} p_{i'}(x_1) p_{j'}(x_2) f(\bar{x}) dx_1 dx_2 = \sum_{i,j} \hat{f}_{ij} \underbrace{\int_{-1}^{1}\int_{-1}^{1} p_i(x_1) p_{i'}(x_1) p_j(x_2) p_{j'}(x_2) dx_1 dx_2}_{\delta_{ii'}\delta_{jj'}} \tag{7.29}$$

$$\int_{-1}^{1}\int_{-1}^{1} p_{i'}(x_1) p_{j'}(x_2) f(\bar{x}) dx_1 dx_2 = \hat{f}_{i'j'}, \tag{7.30}$$

thus the entries of $\underline{\hat{F}}$ can be calculated by evaluating two-dimensional integrals. Since f is an arbitrary function, we apply numerical integration. We use two one-dimensional quadrature rules of the form

$$\int_{-1}^{1} g(z) dz \approx \sum_{m=0}^{K_m} w_m g(z_m) = \bar{w}^T \bar{g} \tag{7.31}$$

where $\bar{w}$ is the vector of known quadrature weights and $\bar{g}$ is the vector whose entries are g

evaluated at the quadrature nodes z_m. Using such a rule for each integral in (7.30) yields

$$\hat{f}_{ij} = \int_{-1}^{1}\int_{-1}^{1} p_i(x_1)\, p_j(x_2)\, f(\bar{x})\, dx_1 dx_2 \tag{7.32}$$

$$= \int_{-1}^{1} p_j(x_2) \sum_{m=0}^{K_{x_1}} w_m p_i(x_{1,m})\, f(x_{1,m}, x_2)\, dx_2 \tag{7.33}$$

$$= \sum_{n=0}^{K_{x_2}} w_n p_j(x_{2,n}) \sum_{m=0}^{K_{x_1}} w_m p_i(x_{1,m})\, f(x_{1,m}, x_{2,n}) \tag{7.34}$$

$$= \sum_{m=0}^{K_{x_1}} \sum_{n=0}^{K_{x_2}} p_i(x_{1,m})\, w_m f(x_{1,m}, x_{2,n})\, w_n p_j(x_{2,n})\,. \tag{7.35}$$

We have arranged the final equation in this symmetric fashion to emphasize that there is a vectorized form that allows us to compute all entries of $\hat{\underline{F}}$ simultaneously. After careful consideration, one can check that

$$\hat{\underline{F}} = \underline{P}_{x_1} \mathrm{diag}(\bar{w})\, \underline{F} \mathrm{diag}(\bar{w})\, \underline{P}_{x_2}^T \tag{7.36}$$

where the entries of $\underline{F}$ are given by $(\underline{F})_{mn} = f(x_{1,m}, x_{2,n})$, and

$$\underline{P}_{x_j} = \begin{bmatrix} \bar{p}(x_{j,0}) & \bar{p}(x_{j,1}) & \cdots & \bar{p}(x_{j,K_{x_j}}) \end{bmatrix}, \qquad j = 1, 2. \tag{7.37}$$

Alternatively, since $\mathrm{diag}(\bar{w})\, \underline{F} \mathrm{diag}(\bar{w}) = \underline{F} \circ \bar{w}\bar{w}^T$ (where $\circ$ denotes the Hadamard product), we can compute

$$\hat{\underline{F}} = \underline{P}_{x_1} (\underline{F} \circ \bar{w}\bar{w}^T) \underline{P}_{x_2}^T \tag{7.38}$$

instead. We can choose either the Gauss-Legendre or Clenshaw-Curtis quadrature nodes and weights to perform the numerical integration. It is difficult to know how many quadrature points $K_{x_j} + 1$ to select *a priori*. *A posteriori*, we can verify whether the coefficients in $\hat{\underline{F}}$ have decayed sufficiently. To do so, we verify whether the last two rows and last two columns of $\hat{\underline{F}}$ have decayed below a user specified tolerance (say $100\epsilon_{\text{machine}}$). In practice, if the coefficients have not decayed sufficiently, we can double K_{x_j} and compute $\hat{\underline{F}}$ again[1].

Alternatively, we can use a method which is more efficient when K is small when com-

[1] If one is seriously concerned about efficiency, the Clenshaw-Curtis quadrature nodes with K_{x_j} and $2K_{x_j}$ are nested. Thus if the error tolerance is not met using a matrix $\underline{P}_{x_j}$ with $K_{x_j}+1$ columns, one can double the number of quadrature points and reuse the $K_{x_j}+1$ columns of $\underline{P}_{x_j}$ to construct a new $\underline{P}_{x_j}$ with $2K_{x_j}+1$ columns.

pared to K_{x_1} and K_{x_2} based closely on the method described in [50]. The method constructs

$$f(\bar{x}) = \sum_{k=1}^{K} \sigma_k f_{k,x_1}(x_1) f_{k,x_2}(x_2) \tag{7.39}$$

where

$$f_{k,x_1}(x_1) = \bar{p}(x_1)^T \bar{u}_k, \tag{7.40}$$

$$f_{k,x_2}(x_2) = \bar{v}_k^T \bar{p}(x_2), \tag{7.41}$$

are each one-dimensional Legendre expansions. Our approach differs from the one in [50] only in that we require Legendre expansions rather than Chebyshev expansions to facilitate integration later. To make conversion between samples of the functions $f_{k,x_j}(x_j)$ and their coefficients in a basis of orthogonal polynomials, the authors of [50] use Chebyshev polynomials $\bar{T}(x_j)$ in place of orthonormal Legendre polynomials $\bar{p}(x_j)$. In addition, to exploit matrix computations, the authors of [50] require that all vectors $\bar{u}_k$ be of the same length so as to form a matrix $\underline{U}$ (and similarly for vectors $\bar{v}_k$ and matrix $\underline{V}$). We do the same. The method requires $\mathcal{O}(K^2(K_{x_1} + K_{x_2}) + K^3)$ operations for a fixed user specified tolerance (the more stringent the tolerance, the larger the constant hidden by the $\mathcal{O}$ notation). Thus, if the function can be represented by a small number of separable functions, the method will be much faster asymptotically as the degrees K_{x_1} and K_{x_2} grow, than the direct method which requires computing the SVD of a $(K_{x_1}+1) \times (K_{x_2}+1)$ matrix (which would require $\mathcal{O}(K_{x_1}K_{x_2}(K_{x_1} + K_{x_2}) + K_{x_2}^3)$ operations [25]).

While we leave most details of the implementation of the algorithm to [50], we take this opportunity to describe what we feel are its most crucial aspects. The algorithm operates in two stages. The first stage is designed to determine the number of separable functions required to represent f. In this stage, the function $f(\bar{x})$ is sampled on a tensor product grid of Chebyshev nodes

$$x_i = -\cos\left(\frac{\pi}{2^{l+2}} i\right) \tag{7.42}$$

where $i = 0, 1, 2..., 2^{l+2}$, so that there are $(2^{l+2}+1)^2$ points in the grid. The additional factor of 2 in the exponent is used to avoid undersampling the function. In practice, l is an integer which we take to be between 6 and 8. We think of the function sampled on this grid as a matrix $\underline{F}$ on which we perform Gaussian elimination with complete pivoting. Starting with

iteration $k = 1$ and $\underline{F}_0 = \underline{F}$, we find the pivot indices and value

$$(i_k, j_k) = \arg\max_{i,j} |(\underline{F}_k)_{ij}|, \tag{7.43}$$

$$f_k = (\underline{F}_k)_{i_k, j_k}, \tag{7.44}$$

then we perform one step of Gaussian elimination

$$\underline{F}_k = \underline{F}_{k-1} - \frac{1}{f_k} \bar{f}_{j_k} \bar{f}_{i_k}^T \tag{7.45}$$

where $\bar{f}_{j_k}$ is the j_kth column of $\underline{F}_k$ and $\bar{f}_{i_k}^T$ is the i_kth row of $\underline{F}_k$. We repeat this process for $k \leq 2^l + 1$ or until $\|\underline{F}_k\|_F < \epsilon_{\text{tol}}$, that is, until the Frobenius norm of $\underline{F}_k$ is less than a user specified tolerance ϵ_{tol}. If we have not met the tolerance, we restart the process on a grid that is twice as fine in both directions. We use $k \leq 2^l + 1$ so that in a worst case,

$$k = 2^l + 1 \tag{7.46}$$

$$\log_2 (k - 1) = l \tag{7.47}$$

and the size of the sampled matrix $\underline{F}$ is

$$2^{l+2} + 1 = 2^{\log_2(k-1)+2} + 1 \tag{7.48}$$

$$= 2^{\log_2(k-1)} 2^2 + 1 \tag{7.49}$$

$$= 4(k - 1) + 1 \tag{7.50}$$

$$= \mathcal{O}(k). \tag{7.51}$$

This means that we perform at most k steps of Gaussian elimination on a matrix with $\mathcal{O}(k)$ rows and columns, yielding a total cost of $\mathcal{O}(k^3)$ operations in the first stage. Let us call this maximal k capital K, which will coincide with K as in (7.23).

The second stage aims to accurately resolve the K separable functions using Chebyshev expansions. In this second stage, the algorithm operates on a K-skeleton of the matrix $\underline{F}$ (that is, we only store and manipulate the K rows and K columns that were used as pivot rows and columns during Gaussian elimination). We create the K-skeleton with double the number of points in both the x_1 and x_2 directions and perform the K steps of Gaussian elimination on the K-skeleton using the same pivots as determined in the first phase of the algorithm. This is possible since the Chebyshev nodes are nested when we double their number and so the pivot locations are still present in the finer sampling. Since the nodes in each direction are the Chebyshev nodes, we can use an FFT to compute Chebyshev

coefficients for each row and column (see Section A.1 of Appendix A for details on how this is performed). We then check if the final two entries in all of the row expansions have fallen below the user specified tolerance. If yes, we conclude that the x_1-dependent components of the separable functions are sufficiently well resolved to represent f to user satisfaction and we stop refining the grid in the x_1 direction. We perform a similar check for the columns. If either the rows or the columns are not sufficiently refined, we double the number of Chebyshev points in the appropriate direction and repeat the process. If degree K_{x_1} polynomials are needed to resolve the rows and degree K_{x_2} polynomials are needed to resolve the columns, then the function f has to be sampled at a maximum of $K\left(K_{x_1}+K_{x_2}\right)$ locations. Performing K steps of Gaussian elimination on the K-skeleton requires $\mathcal{O}\left(K^2\left(K_{x_1}+K_{x_2}\right)\right)$ operations.

This two stage process creates a representation of f of the form

$$f\left(\bar{x}\right)=\sum_{k=1}^{K}\frac{1}{f_k}\left(\bar{T}\left(x_1\right)^T\tilde{u}_k\right)\left(\tilde{v}_k^T\bar{T}\left(x_2\right)\right) \tag{7.52}$$

where $\bar{T}\left(x_j\right)$ is a vector of Chebyshev polynomials. Finally, we apply the method described in Section 4.2 and Appendix A to convert the Chebyshev coefficients $\tilde{u}_k$ and $\tilde{v}_k$ to orthonormal Legendre coefficients $\bar{u}_k$ and $\bar{v}_k$. Since the Chebyshev to Legendre transform for a degree n expansion can be performed in $\mathcal{O}(n\left(\log_2 n\right)^2)$ operations, this conversion can be performed in $\mathcal{O}\left(K(K_{x_1}\left(\log_2 K_{x_1}\right)^2+K_{x_2}\left(\log_2 K_{x_2}\right)^2)\right)$ operations. This does not compromise the efficiency of the method. As a result, after relabeling, we obtain the expansion

$$f\left(\bar{x}\right)=\sum_{k=1}^{K}\sigma_k\left(\bar{p}\left(x_1\right)^T\bar{u}_k\right)\left(\bar{v}_k^T\bar{p}\left(x_2\right)\right) \tag{7.53}$$

where we have set $\sigma_k=1/f_k$.

7.3 Assembling the Operator Matrix

We now have the requisite Legendre expansions to address assembly of the matrices and vectors $\underline{A}$, $\underline{C}$, and $\bar{b}$, $\bar{d}$. We do so by going term by term through (6.46) and (6.52), starting with the operator matrix $\underline{A}$. Let us begin with

$$\int_\Omega \underline{J}_{\bar{N}}\left(\bar{x}\right)\underline{\alpha}\underline{J}_{\bar{N}}\left(\bar{x}\right)^T d\Omega. \tag{7.54}$$

This is the most complicated term in $\underline{A}$ but can be decomposed into a set of simpler terms if we carry out the matrix product. To see why, it is easier to take a step back and consider

$$\int_{\Omega} \nabla\psi^T \underline{\alpha} \nabla\phi \, d\Omega \tag{7.55}$$

which gave rise to (7.54). Expanding the integrand gives

$$\nabla\psi^T \underline{\alpha} \nabla\phi = \begin{bmatrix} \dfrac{\partial\psi}{\partial x_1} & \dfrac{\partial\psi}{\partial x_2} \end{bmatrix} \begin{bmatrix} \alpha_{11}(\bar{x}) & \alpha_{12}(\bar{x}) \\ \alpha_{21}(\bar{x}) & \alpha_{22}(\bar{x}) \end{bmatrix} \begin{bmatrix} \dfrac{\partial\phi}{\partial x_1} \\ \dfrac{\partial\phi}{\partial x_2} \end{bmatrix} \tag{7.56}$$

$$= \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial\psi}{\partial x_i} \alpha_{ij}(\bar{x}) \frac{\partial\phi}{\partial x_j} \tag{7.57}$$

or, written explicitly,

$$\nabla\psi^T \underline{\alpha} \nabla\phi = \frac{\partial\psi}{\partial x_1} \alpha_{11}(\bar{x}) \frac{\partial\phi}{\partial x_1} + \frac{\partial\psi}{\partial x_1} \alpha_{12}(\bar{x}) \frac{\partial\phi}{\partial x_2} + \frac{\partial\psi}{\partial x_2} \alpha_{21}(\bar{x}) \frac{\partial\phi}{\partial x_1} + \frac{\partial\psi}{\partial x_2} \alpha_{22}(\bar{x}) \frac{\partial\phi}{\partial x_2}. \tag{7.58}$$

This form demonstrates how to compute the integral (7.54), which we write as

$$\begin{aligned} \int_{\Omega} \underline{J}_{\bar{N}}(\bar{x}) \, \underline{\alpha} \underline{J}_{\bar{N}}(\bar{x})^T \, d\Omega = & \int_{\Omega} \left[\frac{\partial}{\partial x_1} \bar{N}(\bar{x}) \right] \alpha_{11}(\bar{x}) \left[\frac{\partial}{\partial x_1} \bar{N}(\bar{x}) \right]^T d\Omega \\ & + 2\,\mathrm{sym} \left(\int_{\Omega} \left[\frac{\partial}{\partial x_1} \bar{N}(\bar{x}) \right] \alpha_{12}(\bar{x}) \left[\frac{\partial}{\partial x_2} \bar{N}(\bar{x}) \right]^T d\Omega \right) \\ & + \int_{\Omega} \left[\frac{\partial}{\partial x_2} \bar{N}(\bar{x}) \right] \alpha_{22}(\bar{x}) \left[\frac{\partial}{\partial x_2} \bar{N}(\bar{x}) \right]^T d\Omega. \end{aligned} \tag{7.59}$$

Note that in this last expression, we have made use of the fact that $\underline{\alpha} = \underline{\alpha}^T$ so that $\alpha_{12} = \alpha_{21}$ and that for any matrix $\underline{A}$,

$$\mathrm{sym}(\underline{A}) = \frac{1}{2}(\underline{A} + \underline{A}^T). \tag{7.60}$$

As in the one-dimensional case, we aim to evaluate these integrals using triple products of Legendre polynomials. To do that, we need to express the derivatives of basis functions in terms of Legendre polynomials (this is similar to the approach outlined in Chapter 5). As an example, consider taking the partial derivative of the basis functions with respect to x_1. Then

$$\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right) = \frac{\partial}{\partial x_1}\left[\left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right)\right] \tag{7.61}$$

$$= \left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\right)\left(\bar{p}\left(x_2\right)\otimes\frac{\partial}{\partial x_1}\bar{p}\left(x_1\right)\right). \tag{7.62}$$

Using the one-dimensional result

$$\frac{\partial}{\partial x_j}\bar{p}\left(x_j\right) = \underline{\tilde{D}}\bar{p}\left(x_j\right), \qquad j = 1, 2, \tag{7.63}$$

and $\underline{\tilde{S}}\underline{\tilde{D}} = \underline{S}_{DL}$, we obtain

$$\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right) = \left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\right)\left(\bar{p}\left(x_2\right)\otimes\underline{\tilde{D}}\bar{p}\left(x_1\right)\right) \tag{7.64}$$

$$= \left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\right)\left(\underline{I}\otimes\underline{\tilde{D}}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right) \tag{7.65}$$

$$= \left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\underline{\tilde{D}}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right) \tag{7.66}$$

$$= \left(\underline{\tilde{S}}\otimes\underline{S}_{DL}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right) \tag{7.67}$$

where we have used property (5.12). A similar procedure yields

$$\frac{\partial}{\partial x_2}\bar{N}\left(\bar{x}\right) = \left(\underline{S}_{DL}\otimes\underline{\tilde{S}}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right). \tag{7.68}$$

Since

$$\bar{N}\left(\bar{x}\right) = \left(\underline{\tilde{S}}\otimes\underline{\tilde{S}}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right), \tag{7.69}$$

we note that all three of $\bar{N}\left(\bar{x}\right)$, $\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right)$, and $\frac{\partial}{\partial x_2}\bar{N}\left(\bar{x}\right)$ have the same generic form

$$\left(\underline{F}\otimes\underline{E}\right)\left(\bar{p}\left(x_2\right)\otimes\bar{p}\left(x_1\right)\right) \tag{7.70}$$

where $\underline{E}$ and $\underline{F}$ can be either $\underline{\tilde{S}}$ or $\underline{S}_{DL}$. In this generic form, all four integrals

$$\int_\Omega \left[\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right)\right]\alpha_{11}\left(\bar{x}\right)\left[\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right)\right]^T d\Omega, \tag{7.71}$$

$$\int_\Omega \left[\frac{\partial}{\partial x_1}\bar{N}\left(\bar{x}\right)\right]\alpha_{12}\left(\bar{x}\right)\left[\frac{\partial}{\partial x_2}\bar{N}\left(\bar{x}\right)\right]^T d\Omega, \tag{7.72}$$

$$\int_\Omega \left[\frac{\partial}{\partial x_2}\bar{N}\left(\bar{x}\right)\right]\alpha_{22}\left(\bar{x}\right)\left[\frac{\partial}{\partial x_2}\bar{N}\left(\bar{x}\right)\right]^T d\Omega, \tag{7.73}$$

and

$$\int_{\Omega} \bar{N}\left(\bar{x}\right) \beta\left(\bar{x}\right) \bar{N}\left(\bar{x}\right)^{T} d\Omega, \tag{7.74}$$

have the form

$$\int_{\Omega} \left(\underline{F} \otimes \underline{E}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) s\left(\bar{x}\right) \left[\left(\underline{H} \otimes \underline{G}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right)\right]^{T} d\Omega \tag{7.75}$$

where s is some specified scalar function and $\underline{E}$, $\underline{F}$, $\underline{G}$, and $\underline{H}$ are some known matrices.

To evaluate this generic integral, we find the separable Legendre representation of s given by

$$s\left(\bar{x}\right) = \sum_{k=1}^{K} \sigma_k \left(\bar{p}\left(x_1\right)^{T} \bar{u}_k\right) \left(\bar{v}_k^{T} \bar{p}\left(x_2\right)\right). \tag{7.76}$$

Since s is a scalar function, its vectorization is simply itself. In addition, using the linearity of the vectorization operator yields

$$\mathrm{vec}\left[s\left(\bar{x}\right)\right] = \mathrm{vec}\left[\sum_{k=1}^{K} \sigma_k \left(\bar{p}\left(x_1\right)^{T} \bar{u}_k\right) \left(\bar{v}_k^{T} \bar{p}\left(x_2\right)\right)\right] \tag{7.77}$$

$$s\left(\bar{x}\right) = \sum_{k=1}^{K} \sigma_k \mathrm{vec}\left[\left(\bar{p}\left(x_1\right)^{T} \bar{u}_k\right) \left(\bar{v}_k^{T} \bar{p}\left(x_2\right)\right)\right]. \tag{7.78}$$

If we treat the argument of each vectorization operation as the product of three matrices $\bar{p}\left(x_1\right)^{T}$, $\bar{u}_k \bar{v}_k^{T}$, and $\bar{p}\left(x_2\right)$, we obtain

$$s\left(\bar{x}\right) = \sum_{k=1}^{K} \sigma_k \mathrm{vec}\left[\bar{p}\left(x_1\right)^{T} \left(\bar{u}_k \bar{v}_k^{T}\right) \bar{p}\left(x_2\right)\right] \tag{7.79}$$

$$= \sum_{k=1}^{K} \sigma_k \left(\bar{p}\left(x_2\right)^{T} \otimes \bar{p}\left(x_1\right)^{T}\right) \mathrm{vec}\left(\bar{u}_k \bar{v}_k^{T}\right) \tag{7.80}$$

$$= \sum_{k=1}^{K} \sigma_k \left(\bar{p}\left(x_2\right)^{T} \otimes \bar{p}\left(x_1\right)^{T}\right) \left(\bar{v}_k \otimes \bar{u}_k\right). \tag{7.81}$$

Thus, due to the separable representation of s, we can write s as the sum of K terms each in a Kronecker product form. Substituting this form into (7.75) gives

$$\int_{\Omega} \left(\underline{F} \otimes \underline{E}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) \left[\sum_{k=1}^{K} \sigma_k \left(\bar{p}\left(x_2\right)^{T} \otimes \bar{p}\left(x_1\right)^{T}\right) \left(\bar{v}_k \otimes \bar{u}_k\right)\right] \cdot \left(\bar{p}\left(x_2\right)^{T} \otimes \bar{p}\left(x_1\right)^{T}\right) \left(\underline{H}^{T} \otimes \underline{G}^{T}\right) d\Omega. \tag{7.82}$$

Taking the sum outside the integral yields

$$\sum_{k=1}^{K} \sigma_k \int_{\Omega} \left(\underline{F} \otimes \underline{E}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) \left(\bar{p}\left(x_2\right)^T \otimes \bar{p}\left(x_1\right)^T\right) \left(\bar{v}_k \otimes \bar{u}_k\right) \cdot \left(\bar{p}\left(x_2\right)^T \otimes \bar{p}\left(x_1\right)^T\right) \left(\underline{H}^T \otimes \underline{G}^T\right) d\Omega \quad (7.83)$$

which, using (5.12), gives

$$\sum_{k=1}^{K} \sigma_k \int_{\Omega} \left(\underline{F} \bar{p}\left(x_2\right) \left[\bar{p}\left(x_2\right)^T \bar{v}_k\right] \bar{p}\left(x_2\right)^T \underline{H}^T \otimes \underline{E} \bar{p}\left(x_1\right) \left[\bar{p}\left(x_1\right)^T \bar{u}_k\right] \bar{p}\left(x_1\right)^T \underline{G}^T\right) d\Omega. \quad (7.84)$$

Note that the terms in square brackets can be written in summation form as

$$\bar{p}\left(x_2\right)^T \bar{v}_k = \sum_{i=0}^{K_{x_2}} v_{ik} p_i\left(x_2\right), \quad (7.85)$$

$$\bar{p}\left(x_1\right)^T \bar{u}_k = \sum_{i=0}^{K_{x_1}} u_{ik} p_i\left(x_1\right), \quad (7.86)$$

so that

$$\int_{-1}^{1} \bar{p}\left(x_2\right) \left[\bar{p}\left(x_2\right)^T \bar{v}_k\right] \bar{p}\left(x_2\right)^T dx_2 = \int_{-1}^{1} \bar{p}\left(x_2\right) \left[\sum_{i=0}^{K_{x_2}} v_{ik} p_i\left(x_2\right)\right] \bar{p}\left(x_2\right)^T dx_2 \quad (7.87)$$

$$= \sum_{i=0}^{K_{x_2}} v_{ik} \int_{-1}^{1} p_i\left(x_2\right) \bar{p}\left(x_2\right) \bar{p}\left(x_2\right)^T dx_2 \quad (7.88)$$

$$= \sum_{i=0}^{K_{x_2}} v_{ik} \underline{T}_i \quad (7.89)$$

and similarly

$$\int_{-1}^{1} \bar{p}\left(x_1\right) \left[\bar{p}\left(x_1\right)^T \bar{u}_k\right] \bar{p}\left(x_1\right)^T dx_1 = \sum_{i=0}^{K_{x_1}} u_{ik} \underline{T}_i \quad (7.90)$$

where $\underline{T}_i$ is the ith frontal slice of the tensor $\underline{\underline{T}}$ containing integrals of triple products of Legendre polynomials (recall Section 4.3). As a result,

$$\int_{\Omega} \left(\underline{F} \otimes \underline{E}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) s\left(\bar{x}\right) \left[\left(\underline{H} \otimes \underline{G}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right)\right]^T d\Omega = \sum_{k=1}^{K} \sigma_k \left(\underline{F} \left[\sum_{i=0}^{K_{x_2}} v_{ik} \underline{T}_i\right] \underline{H}^T \otimes \underline{E} \left[\sum_{i=0}^{K_{x_1}} u_{ik} \underline{T}_i\right] \underline{G}^T\right) \quad (7.91)$$

and we can write the explicit entries of each of (7.71)-(7.74) once we have computed the

separable forms of α_{11}, α_{12}, α_{22}, and β, respectively[2]. The relevant matrices are

$$\int_\Omega \left[\frac{\partial}{\partial x_1}\bar{N}(\bar{x})\right] \alpha_{11}(\bar{x}) \left[\frac{\partial}{\partial x_1}\bar{N}(\bar{x})\right]^T d\Omega = \sum_{k=1}^{K^{(\alpha_{11})}} \sigma_k^{(\alpha_{11})} \left(\underline{\tilde{S}} \left[\sum_{i=0}^{K_{x_2}^{(\alpha_{11})}} v_{ik}^{(\alpha_{11})} \underline{T}_i \right] \underline{\tilde{S}}^T \otimes \underline{S}_{DL} \left[\sum_{i=0}^{K_{x_1}^{(\alpha_{11})}} u_{ik}^{(\alpha_{11})} \underline{T}_i \right] \underline{S}_{DL}^T \right), \quad (7.92)$$

$$\int_\Omega \left[\frac{\partial}{\partial x_1}\bar{N}(\bar{x})\right] \alpha_{12}(\bar{x}) \left[\frac{\partial}{\partial x_2}\bar{N}(\bar{x})\right]^T d\Omega = \sum_{k=1}^{K^{(\alpha_{12})}} \sigma_k^{(\alpha_{12})} \left(\underline{\tilde{S}} \left[\sum_{i=0}^{K_{x_2}^{(\alpha_{12})}} v_{ik}^{(\alpha_{12})} \underline{T}_i \right] \underline{S}_{DL}^T \otimes \underline{S}_{DL} \left[\sum_{i=0}^{K_{x_1}^{(\alpha_{12})}} u_{ik}^{(\alpha_{12})} \underline{T}_i \right] \underline{\tilde{S}}^T \right), \quad (7.93)$$

$$\int_\Omega \left[\frac{\partial}{\partial x_2}\bar{N}(\bar{x})\right] \alpha_{22}(\bar{x}) \left[\frac{\partial}{\partial x_2}\bar{N}(\bar{x})\right]^T d\Omega = \sum_{k=1}^{K^{(\alpha_{22})}} \sigma_k^{(\alpha_{22})} \left(\underline{S}_{DL} \left[\sum_{i=0}^{K_{x_2}^{(\alpha_{22})}} v_{ik}^{(\alpha_{22})} \underline{T}_i \right] \underline{S}_{DL}^T \otimes \underline{\tilde{S}} \left[\sum_{i=0}^{K_{x_1}^{(\alpha_{22})}} u_{ik}^{(\alpha_{22})} \underline{T}_i \right] \underline{\tilde{S}}^T \right), \quad (7.94)$$

and

$$\int_\Omega \bar{N}(\bar{x})\,\beta(\bar{x})\,\bar{N}(\bar{x})^T\,d\Omega = \sum_{k=1}^{K^{(\beta)}} \sigma_k^{(\beta)} \left(\underline{\tilde{S}} \left[\sum_{i=0}^{K_{x_2}^{(\beta)}} v_{ik}^{(\beta)} \underline{T}_i \right] \underline{\tilde{S}}^T \otimes \underline{\tilde{S}} \left[\sum_{i=0}^{K_{x_1}^{(\beta)}} u_{ik}^{(\beta)} \underline{T}_i \right] \underline{\tilde{S}}^T \right). \quad (7.95)$$

Before moving on, we comment on the sparsity patterns of these matrices. Notice that in each Kronecker product term, we have one of four types of expressions, two of which have appeared in Section 4.3. In particular, we have already seen that matrices of the form

$$\underline{S}_{DL} \left[\sum_{i=0}^{K_\alpha} \alpha_i \underline{T}_i \right] \underline{S}_{DL}^T, \qquad \underline{\tilde{S}} \left[\sum_{i=0}^{K_\beta} \beta_i \underline{T}_i \right] \underline{\tilde{S}}^T, \quad (7.96)$$

have bandwidths K_α and $K_\beta + 2$ respectively. Terms of the form

$$\underline{\tilde{S}} \left[\sum_{i=0}^{K_\gamma} \gamma_i \underline{T}_i \right] \underline{S}_{DL}^T, \qquad \underline{S}_{DL} \left[\sum_{i=0}^{K_\delta} \delta_i \underline{T}_i \right] \underline{\tilde{S}}^T, \quad (7.97)$$

[2]If we allow each vector $\bar{u}_k$ and $\bar{v}_k$ to be of different lengths, then we can avoid adding unnecessary slices of the tensor. In practice, I do not do this, judging the simplicity of representation to outweigh any small computational gains.

have bandwidths $K_\gamma + 1$ and $K_\delta + 1$ respectively. To understand how these sparsity patterns affect the Kronecker product, let us refer to the bandwidth in the x_2 related matrices (which appear first in the Kronecker product) as BW_{x_2} and the bandwidth of x_1 related matrices as BW_{x_1}. By definition of the Kronecker product, a matrix of size $(L_{x_1}+1)^2 \times (L_{x_2}+1)^2$ with bandwidth $\mathrm{BW}_{\mathrm{total}} = (L_{x_1}+1)\,\mathrm{BW}_{x_2} + \mathrm{BW}_{x_1}$ results when taking the Kronecker product of a matrix of size $(L_{x_2}+1) \times (L_{x_2}+1)$ with bandwidth BW_{x_2}, together with a matrix of size $(L_{x_1}+1) \times (L_{x_1}+1)$ with bandwidth BW_{x_1}. This is because the corresponding Kronecker product possesses $L_{x_2}+1$ block rows and columns, with a block bandwidth of BW_{x_2}. Each nonzero block is populated by a $(L_{x_1}+1) \times (L_{x_1}+1)$ sized matrix of bandwidth BW_{x_1}. Combining these two observations yields the bandwidth of the whole matrix. For the four matrices (7.71)-(7.74), the bandwidths are $(L_{x_1}+1)\,(K_{x_2}^{(\alpha_{11})}+2)+K_{x_1}^{(\alpha_{11})}$, $(L_{x_1}+1)\,(K_{x_2}^{(\alpha_{12})}+1)+K_{x_1}^{(\alpha_{12})}+1$, $(L_{x_1}+1)\,K_{x_2}^{(\alpha_{22})}+K_{x_1}^{(\alpha_{22})}+2$, and $(L_{x_1}+1)\,(K_{x_2}^{(\beta)}+2)+K_{x_1}^{(\beta)}+2$, respectively.

Since the operator matrix $\underline{A}$ is a sum of banded matrices, some of which possess these four specified bandwidths, the total operator matrix is itself a banded matrix with bandwidth given by the maximal bandwidth of these four matrices (ignoring Robin boundary conditions). Of course, comparing these bandwidths is only possible once the associated Legendre expansions for a given problem are computed. However, assuming a problem with constant α_{11}, α_{12}, α_{22}, and β, they reduce to $2\,(L_{x_1}+1)$, $L_{x_1}+2$, 2, and $2\,(L_{x_1}+1)+2$, with the term associated with β dominating (as it did in the one-dimensional case).

Notice that the convention of using basis functions $\bar{N}\,(\bar{x}) = \bar{N}\,(x_2) \otimes \bar{N}\,(x_1)$ rather than $\bar{N}\,(\bar{x}) = \bar{N}\,(x_1) \otimes \bar{N}\,(x_2)$ plays a role in the sparsity pattern of these matrices. If we switch the order of basis functions, we modify the bandwidth of the resulting Kronecker product matrices, yielding an alternative bandwidth $\mathrm{BW}_{\mathrm{alt}} = (L_{x_2}+1)\,\mathrm{BW}_{x_1} + \mathrm{BW}_{x_2}$ which may be smaller than $\mathrm{BW}_{\mathrm{total}}$ for certain spatially varying coefficients and certain choices of polynomial degree. Finally, we note that when the Legendre expansions require high degree polynomials to represent α_{11}, α_{12}, α_{22}, or β, the bandwidths of these matrices grow, possibly resulting in full matrices in a worst case scenario.

To complete the operator matrix $\underline{A}$, we also need to consider any contributions from Robin boundary conditions. Thus we must take into account terms of the form

$$\int_{\Gamma_R} \bar{N}\,(\bar{x})\,\gamma \bar{N}\,(\bar{x})^T\,d\Omega \tag{7.98}$$

where Γ_R is a subset of the four edges Γ_1, Γ_2, Γ_3, or Γ_4 of the square domain. Since we have not specified any edges in particular, and since we ultimately would like to produce a method capable of handling any combination of these edges, we show the form of this matrix assuming $\Gamma_R = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$. Since this union is disjoint (there is no overlap between

edges), the integral becomes

$$\int_{\Gamma_1} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega + \int_{\Gamma_2} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega + \int_{\Gamma_3} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega + \int_{\Gamma_4} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega. \tag{7.99}$$

It is convenient to treat the first and second integrals simultaneously and the third and fourth in a similar fashion. To see why, note that by definition of Γ_1 and Γ_2, $x_1 = \mp 1$, respectively, and likewise, for Γ_3 and Γ_4, $x_2 = \mp 1$. Rather than write ∓ 1, we use $(-1)^i$ with $i = 1, 2, 3, 4$, instead.

Evaluating the basis functions along edges Γ_1 and Γ_2 gives

$$\bar{N}\left((-1)^i, x_2\right) = \bar{N}\left(x_2\right) \otimes \bar{N}\left((-1)^i\right) \tag{7.100}$$

$$= \underline{\tilde{S}}\bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right). \tag{7.101}$$

If we compute one-dimensional Legendre expansions of γ along both edges, then for $i = 1, 2$, we obtain

$$\gamma\left((-1)^i, x_2\right) = \sum_{k=0}^{K_\gamma^{(i)}} \gamma_k^{(i)} p_k\left(x_2\right) \tag{7.102}$$

where the required degree $K_\gamma^{(i)}$ and the coefficients $\gamma_k^{(i)}$ may differ on either edge. For $i = 1, 2$, we get

$$\int_{\Gamma_i} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega = \int_{-1}^{1} \left(\underline{\tilde{S}}\bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right) \left[\sum_{k=0}^{K_\gamma^{(i)}} \gamma_k^{(i)} p_k\left(x_2\right)\right] \cdot \left(\underline{\tilde{S}}\bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right)^T dx_2 \tag{7.103}$$

which, when applying the transpose and multiplying the scalar expression in square brackets into the rightmost term, gives

$$\int_{\Gamma_i} \bar{N}\left(\bar{x}\right)\gamma\bar{N}\left(\bar{x}\right)^T d\Omega = \int_{-1}^{1} \left(\underline{\tilde{S}}\bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right) \cdot \left(\left[\sum_{k=0}^{K_\gamma^{(i)}} \gamma_k^{(i)} p_k\left(x_2\right)\right] \bar{p}\left(x_2\right)^T \underline{\tilde{S}}^T \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T\right) dx_2. \tag{7.104}$$

Finally, multiplying the Kronecker product matrices and computing the integral yields

$$\int_{\Gamma_i} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(i)}}\gamma_k^{(i)}\underline{T}_k\right]\underline{\tilde{S}}^T \otimes \frac{1}{2}\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T\right). \tag{7.105}$$

An analogous treatment for boundary components Γ_3 and Γ_4 requires Legendre expansions for $i = 3, 4$, such that

$$\gamma\left(x_1,(-1)^i\right) = \sum_{k=0}^{K_\gamma^{(i)}}\gamma_k^{(i)}p_k(x_1). \tag{7.106}$$

The boundary integrals become

$$\int_{\Gamma_i} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\frac{1}{2}\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T \otimes \underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(i)}}\gamma_k^{(i)}\underline{T}_k\right]\underline{\tilde{S}}^T\right). \tag{7.107}$$

To summarize,

$$\int_{\Gamma_1} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(1)}}\gamma_k^{(1)}\underline{T}_k\right]\underline{\tilde{S}}^T \otimes \frac{1}{2}\left(\bar{e}_1-\bar{e}_2\right)\left(\bar{e}_1-\bar{e}_2\right)^T\right), \tag{7.108}$$

$$\int_{\Gamma_2} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(2)}}\gamma_k^{(2)}\underline{T}_k\right]\underline{\tilde{S}}^T \otimes \frac{1}{2}\left(\bar{e}_1+\bar{e}_2\right)\left(\bar{e}_1+\bar{e}_2\right)^T\right), \tag{7.109}$$

$$\int_{\Gamma_3} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\frac{1}{2}\left(\bar{e}_1-\bar{e}_2\right)\left(\bar{e}_1-\bar{e}_2\right)^T \otimes \underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(3)}}\gamma_k^{(3)}\underline{T}_k\right]\underline{\tilde{S}}^T\right), \tag{7.110}$$

$$\int_{\Gamma_4} \bar{N}(\bar{x})\,\gamma\bar{N}(\bar{x})^T\,d\Omega = \left(\frac{1}{2}\left(\bar{e}_1+\bar{e}_2\right)\left(\bar{e}_1+\bar{e}_2\right)^T \otimes \underline{\tilde{S}}\left[\sum_{k=0}^{K_\gamma^{(4)}}\gamma_k^{(4)}\underline{T}_k\right]\underline{\tilde{S}}^T\right). \tag{7.111}$$

These Robin boundary matrices are considerably more sparse than those associated with (7.71)-(7.74) due to terms of the form $\underline{R}_i = \frac{1}{2}\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T$. A similar analysis for bandwidth applies, although it is less illuminating because matrices $\underline{R}_i$ only possess four nonzeros, which are located in the first two rows and columns, thus they are banded matrices, but not banded matrices whose bandwidths are full.

7.4 Computing the Forcing Term

Next, we focus on computing terms associated with the forcing vector $\bar{b}$. The first type of contribution comes from the the function f in the form of the integral

$$\int_{\Omega} \bar{N}\left(\bar{x}\right) f \, d\Omega. \tag{7.112}$$

Again, we compute a separable two-dimensional Legendre expansion of f given by

$$f\left(\bar{x}\right) = \sum_{k=0}^{K^{(f)}} \sigma_k^{(f)} \left(\bar{p}\left(x_2\right)^T \otimes \bar{p}\left(x_1\right)^T\right) \left(\bar{v}_k^{(f)} \otimes \bar{u}_k^{(f)}\right) \tag{7.113}$$

$$= \left(\bar{p}\left(x_2\right)^T \otimes \bar{p}\left(x_1\right)^T\right) \mathrm{vec}\left(\underline{U}_{(f)} \underline{\Sigma}_{(f)} \underline{V}_{(f)}^T\right) \tag{7.114}$$

where the form of the second expression was derived earlier in (7.28). In addition, we use the definition of the basis functions

$$\bar{N}\left(\bar{x}\right) = \left(\underline{\tilde{S}} \otimes \underline{\tilde{S}}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) \tag{7.115}$$

to find that

$$\int_{\Omega} \bar{N}\left(\bar{x}\right) f \, d\Omega = \int_{\Omega} \left(\underline{\tilde{S}} \otimes \underline{\tilde{S}}\right) \left(\bar{p}\left(x_2\right) \otimes \bar{p}\left(x_1\right)\right) \left(\bar{p}\left(x_2\right)^T \otimes \bar{p}\left(x_1\right)^T\right) \mathrm{vec}\left(\underline{U}_{(f)} \underline{\Sigma}_{(f)} \underline{V}_{(f)}^T\right) d\Omega \tag{7.116}$$

$$= \left(\underline{\tilde{S}} \otimes \underline{\tilde{S}}\right) \mathrm{vec}\left(\underline{U}_{(f)} \underline{\Sigma}_{(f)} \underline{V}_{(f)}^T\right). \tag{7.117}$$

Using the relationship (5.13) between vectorization and the Kronecker product, we observe that this last expression corresponds to

$$\int_{\Omega} \bar{N}\left(\bar{x}\right) f \, d\Omega = \mathrm{vec}\left(\underline{\tilde{S}} \underline{U}_{(f)} \underline{\Sigma}_{(f)} \underline{V}_{(f)}^T \underline{\tilde{S}}^T\right) \tag{7.118}$$

$$= \mathrm{vec}\left[\left(\underline{\tilde{S}} \underline{U}_{(f)}\right) \underline{\Sigma}_{(f)} \left(\underline{\tilde{S}} \underline{V}_{(f)}\right)^T\right]. \tag{7.119}$$

These expressions for the forcing vector are only correct when the expansion for f shares the same degrees K_{x_1} and K_{x_2} as the basis functions used to represent ϕ (L_{x_1} and L_{x_2}). If $K_{x_1} \neq L_{x_1}$ or $K_{x_2} \neq L_{x_2}$, we can pad the matrices $\underline{U}_{(f)}$ and $\underline{V}_{(f)}$ with zero rows or remove rows to match the degrees (always padding or removing at the last rows of either matrix). This padding or removal must be performed to ensure that matrix products conform.

The second type of contribution to the forcing vector $\bar{b}$ comes from Robin boundary

conditions. In particular, we must evaluate integrals of the form

$$\int_{\Gamma_R} \bar{N}\left(\bar{x}\right) q \, d\Omega. \tag{7.120}$$

As in the operator case, recall (7.99), we consider contributions to this integral from all edges of the boundary of the square. Thus, we assume $\Gamma_R = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$ so that

$$\int_{\Gamma_R} \bar{N}\left(\bar{x}\right) q \, d\Omega = \int_{\Gamma_1} \bar{N}\left(\bar{x}\right) q \, d\Omega + \int_{\Gamma_2} \bar{N}\left(\bar{x}\right) q \, d\Omega + \int_{\Gamma_3} \bar{N}\left(\bar{x}\right) q \, d\Omega + \int_{\Gamma_4} \bar{N}\left(\bar{x}\right) q \, d\Omega. \tag{7.121}$$

Like the operator terms due to Robin boundary conditions, it is convenient to group these integrals in pairs according to boundaries Γ_1 and Γ_2, as well as Γ_3 and Γ_4. We begin with the first pair where $x_1 = (-1)^i$ for $i = 1, 2$. Then

$$\bar{N}\left(\left(-1\right)^i, x_2\right) = \bar{N}\left(x_2\right) \otimes \bar{N}\left(\left(-1\right)^i\right) \tag{7.122}$$

$$= \underline{\tilde{S}} \bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}} \left(\bar{e}_1 + \left(-1\right)^i \bar{e}_2\right). \tag{7.123}$$

If we compute one-dimensional Legendre expansions of q along both edges, then for $i = 1, 2$, we obtain

$$q\left(\left(-1\right)^i, x_2\right) = \bar{p}\left(x_2\right)^T \bar{q}_{(i)} \tag{7.124}$$

where the $\bar{q}_{(i)}$ are vectors of Legendre coefficients. We assume that the degree in each expansion agrees with the degree K_{x_2} of the basis functions $\bar{N}\left(\bar{x}\right)$. If that is not the case, we pad or truncate the vectors $\bar{q}_{(i)}$ so that all future matrix-vector manipulations are conforming. For $i = 1, 2$, we have

$$\int_{\Gamma_i} \bar{N}\left(\bar{x}\right) q \, d\Omega = \int_{-1}^{1} \left(\underline{\tilde{S}} \bar{p}\left(x_2\right) \otimes \frac{1}{\sqrt{2}} \left(\bar{e}_1 + \left(-1\right)^i \bar{e}_2\right)\right) \left[\bar{p}\left(x_2\right)^T \bar{q}_{(i)}\right] dx_2. \tag{7.125}$$

Since the expansion in square brackets is a scalar, we can multiply it into the Kronecker product to obtain

$$\int_{\Gamma_i} \bar{N}\left(\bar{x}\right) q \, d\Omega = \int_{-1}^{1} \left(\underline{\tilde{S}} \bar{p}\left(x_2\right) \bar{p}\left(x_2\right)^T \bar{q}_{(i)} \otimes \frac{1}{\sqrt{2}} \left(\bar{e}_1 + \left(-1\right)^i \bar{e}_2\right)\right) dx_2 \tag{7.126}$$

$$= \left(\underline{\tilde{S}} \bar{q}_{(i)} \otimes \frac{1}{\sqrt{2}} \left(\bar{e}_1 + \left(-1\right)^i \bar{e}_2\right)\right) \tag{7.127}$$

where we have made use of the fact that the integral of the outer product of orthonormal Legendre polynomials yields the identity matrix.

Similarly, by using

$$\bar{N}\left(x_1,(-1)^i\right) = \bar{N}\left((-1)^i\right) \otimes \bar{N}\left(x_1\right) \tag{7.128}$$

$$= \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right) \otimes \underline{\tilde{S}}\bar{p}\left(x_1\right) \tag{7.129}$$

with $i = 3, 4$, and expanding q on those edges as

$$q\left(x_1,(-1)^i\right) = \bar{p}\left(x_1\right)^T \bar{q}_{(i)}, \tag{7.130}$$

we obtain

$$\int_{\Gamma_i} \bar{N}\left(\bar{x}\right) q \, d\Omega = \left(\frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right) \otimes \underline{\tilde{S}}\bar{q}_{(i)}\right). \tag{7.131}$$

To summarize, Robin boundary conditions give rise to forcing vectors

$$\int_{\Gamma_1} \bar{N}\left(\bar{x}\right) q \, d\Omega = \left(\underline{\tilde{S}}\bar{q}_{(1)} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right)\right), \tag{7.132}$$

$$\int_{\Gamma_2} \bar{N}\left(\bar{x}\right) q \, d\Omega = \left(\underline{\tilde{S}}\bar{q}_{(2)} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + \bar{e}_2\right)\right), \tag{7.133}$$

$$\int_{\Gamma_3} \bar{N}\left(\bar{x}\right) q \, d\Omega = \left(\frac{1}{\sqrt{2}}\left(\bar{e}_1 - \bar{e}_2\right) \otimes \underline{\tilde{S}}\bar{q}_{(3)}\right), \tag{7.134}$$

$$\int_{\Gamma_4} \bar{N}\left(\bar{x}\right) q \, d\Omega = \left(\frac{1}{\sqrt{2}}\left(\bar{e}_1 + \bar{e}_2\right) \otimes \underline{\tilde{S}}\bar{q}_{(4)}\right). \tag{7.135}$$

7.5 Enforcing Dirichlet Boundary Conditions

As described in Section 6.3, Dirichlet boundary conditions give rise to auxiliary equations

$$\underbrace{\int_{\Gamma_D} \bar{N}_B\left(\bar{x}\right) \bar{N}\left(\bar{x}\right)^T d\Omega}_{\underline{C}} \bar{\phi} = \underbrace{\int_{\Gamma_D} \bar{N}_B\left(\bar{x}\right) p \, d\Omega}_{\bar{d}}. \tag{7.136}$$

To be more precise, in this section, we show how to choose the weight functions $\bar{N}_B\left(\bar{x}\right)$ along each edge so as to yield sparse constraint matrices. We enforce Dirichlet boundary conditions in a weak sense by requiring the Legendre moments

$$\int_{\Gamma_i} \bar{p}\left(x_2\right)\left[\phi\left((-1)^i, x_2\right) - p\left(x_2\right)\right] dx_2 = 0, \qquad i = 1, 2, \tag{7.137}$$

$$\int_{\Gamma_i} \bar{p}\left(x_1\right)\left[\phi\left(x_1,(-1)^i\right) - p\left(x_1\right)\right] dx_1 = 0, \qquad i = 3, 4, \tag{7.138}$$

to vanish along edges Γ_i. Of course, it is only necessary to impose such conditions on the edges that belong to Γ_D, although in this section we develop conditions for all four edges.

Let us begin with edges Γ_1 and Γ_2. To do this, we need expressions for $\phi\left((-1)^i, x_2\right)$ and $p(x_2)$. For ϕ we use

$$\bar{N}\left((-1)^i, x_2\right) = \bar{N}(x_2) \otimes \bar{N}\left((-1)^i\right) \tag{7.139}$$

$$= \underline{\tilde{S}}\bar{p}(x_2) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right) \tag{7.140}$$

to describe its basis functions, but where we isolate the $\bar{p}(x_2)$ term. To accomplish this, we use property (5.13) of the vectorization operator to obtain

$$\bar{N}\left((-1)^i, x_2\right) = \text{vec}\left(\frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\left[\underline{\tilde{S}}\bar{p}(x_2)\right]^T\right) \tag{7.141}$$

$$= \text{vec}\left(\frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\left[\underline{\tilde{S}}\bar{p}(x_2)\right]^T \underline{I}\right) \tag{7.142}$$

$$= \left(\underline{I} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right) \text{vec}\left(\left[\underline{\tilde{S}}\bar{p}(x_2)\right]^T\right) \tag{7.143}$$

$$= \left(\underline{I} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right) \underline{\tilde{S}}\bar{p}(x_2). \tag{7.144}$$

We also evaluate the function $p(x_2)$ along either edge using one-dimensional Legendre expansions

$$p(x_2) = \bar{p}(x_2)^T \bar{p}_{(i)}. \tag{7.145}$$

Next, we compute the Legendre moment equations

$$\int_{\Gamma_i} \bar{p}(x_2)\, \phi\left((-1)^i, x_2\right) dx_2 = \int_{\Gamma_i} \bar{p}(x_2)\, p(x_2)\, dx_2 \tag{7.146}$$

$$\int_{\Gamma_i} \bar{p}(x_2)\, \bar{N}\left((-1)^i, x_2\right)^T \bar{\phi}\, dx_2 = \int_{\Gamma_i} \bar{p}(x_2)\, \bar{p}(x_2)^T \bar{p}_{(i)} dx_2 \tag{7.147}$$

$$\int_{\Gamma_i} \bar{p}(x_2)\, \bar{p}(x_2)^T\, \underline{\tilde{S}}^T \left(\underline{I} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T\right) dx_2\, \bar{\phi} = \int_{\Gamma_i} \bar{p}(x_2)\, \bar{p}(x_2)^T\, dx_2\, \bar{p}_{(i)} \tag{7.148}$$

$$\underline{\tilde{S}}^T \left(\underline{I} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T\right) \bar{\phi} = \bar{p}_{(i)} \tag{7.149}$$

where we have used the orthogonality of the Legendre polynomials along edges Γ_1 and Γ_2.

We make a minor modification by premultiplying both sides of (7.149) by $\sqrt{2}\underline{\tilde{S}}^{-T}$ to obtain

$$\underbrace{\left(\underline{I}\otimes\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T\right)}_{\underline{C}_i}\bar{\phi}=\underbrace{\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(i)}}_{\bar{d}_i}. \tag{7.150}$$

Note that the term $\underline{\tilde{S}}^{-T}\bar{p}_{(i)}$ has a specific meaning. In particular, since $\bar{N}(x_2)=\underline{\tilde{S}}\bar{p}(x_2)$, this means that $\underline{\tilde{S}}^{-T}\bar{p}_{(i)}$ corresponds to a vector of coefficients in the basis $\bar{N}(x_2)$ for the function $p(x_2)$. That is, $\underline{\tilde{S}}^{-T}$ converts from coefficients in a Legendre expansion to coefficients in an integrated Legendre expansion. We multiply by the factor $\sqrt{2}$ to make the constraint matrices $\underline{C}_i$ rational (in fact they only contain zeros, ones, or negative ones). Thus the constraint equations

$$\underline{C}_i\bar{\phi}=\bar{d}_i \tag{7.151}$$

correspond to matching the coefficients of basis functions representing ϕ along a given edge to modes of the Dirichlet function p. A similar analysis for edges Γ_i with $i=3,4$, transforms the moment constraints

$$\int_{\Gamma_i}\bar{p}(x_1)\left[\phi\left(x_1,(-1)^i\right)-p(x_1)\right]dx_1=0 \tag{7.152}$$

into the linear equations

$$\underbrace{\left(\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T\otimes\underline{I}\right)}_{\underline{C}_i}\bar{\phi}=\underbrace{\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(i)}}_{\bar{d}_i} \tag{7.153}$$

where we had to evaluate the two Legendre expansions

$$p(x_1)=\bar{p}(x_1)^T\bar{p}_{(i)}. \tag{7.154}$$

To summarize, the constraint equations for the four Dirichlet boundary conditions are

$$\left(\underline{I}\otimes(\bar{e}_1-\bar{e}_2)^T\right)\bar{\phi}=\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(1)}, \tag{7.155}$$

$$\left(\underline{I}\otimes(\bar{e}_1+\bar{e}_2)^T\right)\bar{\phi}=\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(2)}, \tag{7.156}$$

$$\left((\bar{e}_1-\bar{e}_2)^T\otimes\underline{I}\right)\bar{\phi}=\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(3)}, \tag{7.157}$$

$$\left((\bar{e}_1+\bar{e}_2)^T\otimes\underline{I}\right)\bar{\phi}=\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}_{(4)}. \tag{7.158}$$

Notice that the constraint equations are extremely sparse, owing to the fact that the integrated Legendre polynomials along a given edge are for the most part zero. In fact, only

$2\left(L_{x_2}+1\right)$ functions are nonzero on each of Γ_1 and Γ_2. Similarly, only $2\left(L_{x_1}+1\right)$ functions are nonzero on each of Γ_3 and Γ_4.

Recall that in using the Galerkin method, we obtained (6.46) which contains a term of the form

$$-\int_{\Gamma_D} \bar{N}\left(\bar{x}\right) \bar{n}^T \left(\underline{\alpha}\nabla\phi\right) d\Omega. \tag{7.159}$$

Suppose we concentrate on edges Γ_i, $i=1,2$, for the moment. If we expand

$$-\bar{n_i}^T \left(\underline{\alpha}\nabla\phi\right) = \sqrt{2}\bar{p}\left(x_2\right)^T \underline{\tilde{S}}^{-1}\bar{\nu}_{(i)} \tag{7.160}$$

where $\bar{\nu}_{(i)}$ are two vectors of unknown coefficients to be determined, then

$$-\int_{\Gamma_i} \bar{N}\left(x_1,(-1)^i\right) \bar{n}^T \left(\underline{\alpha}\nabla\phi\right) dx_2 = \int_{\Gamma_i} \bar{N}\left(x_1,(-1)^i\right) \bar{p}\left(x_2\right)^T dx_2 \sqrt{2}\underline{\tilde{S}}^{-1}\bar{\nu}_{(i)}. \tag{7.161}$$

Notice that after this substitution, the remaining integral is the transpose of (7.147), thus, by virtue of the scaling $\sqrt{2}\underline{\tilde{S}}^{-1}$, (7.159) is discretized as $\underline{C}_i^T\bar{\nu}_{(i)}$. The same type of observation holds for edges Γ_i when $i=3,4$. Let us define the constraint matrix and vector as

$$\underline{C} = \begin{bmatrix} \underline{C}_1 \\ \underline{C}_2 \\ \underline{C}_3 \\ \underline{C}_4 \end{bmatrix}, \qquad \bar{d} = \begin{bmatrix} \bar{d}_1 \\ \bar{d}_2 \\ \bar{d}_3 \\ \bar{d}_4 \end{bmatrix}, \tag{7.162}$$

or, alternatively, $\underline{C}$ and $\bar{d}$ with certain block rows removed in cases where Γ_D does not include all four edges (for example, if Γ_D does not include Γ_3, we omit blocks $\underline{C}_3$ and $\bar{d}_3$ from $\underline{C}$ and $\bar{d}$). Then, with the interpretation of the normal flux density at the Dirichlet boundary expressed as Legendre expansions with vector

$$\bar{\nu} = \begin{bmatrix} \bar{\nu}_1 \\ \bar{\nu}_2 \\ \bar{\nu}_3 \\ \bar{\nu}_4 \end{bmatrix}, \tag{7.163}$$

(again allowing for the same block rows to be absent as in $\underline{C}$ and $\bar{d}$) we obtain the saddle point system

$$\begin{bmatrix} \underline{A} & \underline{C}^T \\ \underline{C} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \bar{d} \end{bmatrix} \tag{7.164}$$

where $\underline{A}$ is assembled using matrices from Section 7.3 and $\bar{b}$ is constructed as in Section 7.4.

Before we move on to some examples, there is a crucial detail we have glossed over. We must ask if the saddle point system is solvable. Recall, from Section 3.4, that the saddle point matrix is invertible if: $\underline{A}$ is symmetric positive semidefinite; $\underline{C}$ has full rank; and $\text{null}(\underline{A}) \cap \text{null}(\underline{C}) = \{0\}$. It turns out that for certain combinations of matrices $\underline{C}_i$, the constraint matrix $\underline{C}$ obtained by concatenation of rows is not full rank (similar behavior was observed in Chapter 5). To see why, consider imposing Dirichlet boundary conditions on edge Γ_1, followed by edge Γ_3 (the order is not important). Imposing a Dirichlet boundary condition on edge Γ_1 restricts ϕ along that edge to match the Legendre expansion used to approximate p along the edge. Since ϕ restricted to the edge is comprised of functions that are integrated Legendre polynomials, two of the functions are nonzero at a given vertex on the edge[3]. Similarly, we also restrict the solution to agree with the Legendre expansion of p along edge Γ_3. The two edges meet at vertex $(-1,-1)$. By imposing the constraints separately, we have twice specified the value of ϕ at the common vertex. Note that this would be acceptable if p was continuous at the vertex since the system of equations would be consistent, however, this causes the constraint matrix to contain a redundant equation. To correct this, we can eliminate the first equation from the constraint matrix $\underline{C}_3$ which restores full row rank[4].

Fortunately, there is a simple way of correcting this rank deficiency. One approach to doing so is to begin with an empty constraint matrix $\underline{C}$ and vector $\bar{d}$ and to update them by considering each edge Γ_i for $i = 1, 2, 3, 4$, sequentially while keeping and updating a list of vertices $(-1,-1)$, $(1,-1)$, $(-1,1)$, and $(1,1)$ and the edges connected to them. That is, for the square, we have a list of the form shown in Table 7.1. To impose Dirichlet constraints, we sequentially check edges to see if a Dirichlet boundary condition should be imposed. If yes, we go to the vertex-to-edge incidence list and mark that edge Γ_i in all locations it appears. We then add the constraints $\underline{C}_i$ and $\bar{d}_i$ by appending their rows to $\underline{C}$ and $\bar{d}$. For every list associated to a node that becomes fully marked, we have a redundant equation. If only one equation is redundant, we remove the first equation from the constraint matrix and vector $\underline{C}_i$ and $\bar{d}_i$ that we would otherwise append to the already present constraint matrix and vector. If we happen to empty two lists simultaneously, we remove the first two equations. This procedure ensures that the final constraint matrix $\underline{C}$ has full rank.

[3]Since Γ_i are open sets, they do not actually include the vertices. Thus, function values at vertices should be interpreted through a limiting process.

[4]There are a variety of cases where we lose full rank. If we impose Dirichlet constraints on all edges, we have four redundant equations. If we impose three Dirichlet constraints, we have two redundant equations. If we impose two Dirichlet constraints, we may have either one or zero redundant equations (for no redundant equations, consider imposing Dirichlet constraints on edges Γ_1 and Γ_2 for example).

Table 7.1: Vertex-to-edge incidence list for the square.

Vertex	Edges
$(-1,-1)$	Γ_1, Γ_3
$(+1,-1)$	Γ_2, Γ_3
$(-1,+1)$	Γ_1, Γ_4
$(+1,+1)$	Γ_2, Γ_4

One may wonder why this approach is presented to correct the rank when there are only four types of cases giving rise to redundant equations. The answer is that the same type of redundant equations arise when we have more than a single quadrilateral element. In that case, the list must include all vertices in the mesh, along with all edges incident to the vertices. In addition, we can also lose full rank when imposing continuity constraints between elements (we will see later that, like in Chapters 4 and 5, continuity constraints are very similar to Dirichlet constraints). That being said, the algorithm to correct for possible rank deficiency remains the same when elements come from a conforming mesh and requires only minor modification for the type of nonconforming meshes that we will use in later chapters, which is why it is described here.

7.6 Examples

To demonstrate the applicability of the proposed scheme, we begin by reproducing the results of three examples from [116]. We solve these problems to emphasize that the method we have described is a spectral method of comparable accuracy to those that employ Chebyshev polynomials. These examples impose Dirichlet boundary conditions on all four edges Γ_1, Γ_2, Γ_3, and Γ_4 and have constant coefficients $\underline{\alpha}$, β. As a result, they do not exercise the full range of possible problems that our framework supports. Unfortunately, they do not have known exact solutions against which we can compare quantitatively (but qualitative agreement is observed). In order to make a quantitative assessment of the method, we reproduce a test problem from [146] which contains a variable coefficient β. In addition, we use our own example to test imposition of Robin boundary conditions. These last two examples do have known exact solutions so that we can compute errors to show the accuracy of the method. In all examples, the user is required to specify the functions α_{11}, α_{12}, α_{22}, β, and f, as well as the type of boundary condition (Dirichlet or Robin) on each edge Γ_i, and the corresponding one-dimensional boundary functions $p_{(i)}$ (in the case of Dirichlet boundaries) or $q_{(i)}$ and $\gamma_{(i)}$ (in the case of Robin boundaries). A user specified tolerance ϵ_{tol} is also required. All Legendre expansions are computed to this tolerance. The resulting saddle point systems for each example can be solved using a sparse direct solver [147] or with methods described in

Appendix B.

For the first example, we solve Poisson's equation

$$\frac{\partial^2 \phi}{\partial x_1^2} + \frac{\partial^2 \phi}{\partial x_2^2} = 10 \sin\left(8 x_1 \left(x_2 - 1\right)\right) \tag{7.165}$$

subject to zero Dirichlet boundary conditions. This corresponds to taking $\underline{\alpha} = -\underline{I}$, $\beta = 0$, $f = 10 \sin\left(8 x_1 \left(x_2 - 1\right)\right)$ and $p = 0$ (since this is a purely Dirichlet problem, we do not need to specify γ or q). Note that since $\underline{\alpha}$ and β are constant, Legendre expansions for α_{11} and α_{22} have $K^{(\alpha_{11})} = K^{(\alpha_{22})} = 1$ terms with $K_{x_1}^{(\alpha_{11})} = K_{x_2}^{(\alpha_{11})} = K_{x_1}^{(\alpha_{22})} = K_{x_2}^{(\alpha_{22})} = 0$ degree expansions in directions x_1 and x_2. Similarly, since α_{12} and β are zero, their expansions have $K^{(\alpha_{12})} = K^{(\beta)} = 1$ terms which are just zero. The forcing function f is not separable, but to a tolerance of $\epsilon_{\text{tol}} = 10^{-12}$ has rank $K^{(f)} = 17$, with degree $K_{x_1}^{(f)} = 41$ and $K_{x_2}^{(f)} = 28$ polynomial expansions required in each direction. All of these Legendre expansions are computed automatically once the functions and tolerance are specified.

Figure 7.1 illustrates an approximate solution with degree $L_{x_1} = L_{x_2} = 41$ obtained by solving the appropriate saddle point system assembled as described in Sections 7.3, 7.4, and 7.5, along with the decay of coefficients used to represent the solution (plotted by computing the entrywise logarithm of absolute values $\log_{10}\left|\underline{\phi}\right|$). Note that the two first rows and columns of the coefficient matrix are zero (indicated by -16 on the log plot instead of $-\infty$) since we have imposed zero Dirichlet boundary conditions. In addition, every odd numbered row is also zero to machine precision. This reflects the fact that there is even symmetry of the solution in the x_1 direction. The sparsity of the saddle point system is illustrated in Figure 7.2 for a reduced polynomial degree so as to clearly show its structure. Each 11-by-11 block in Figure 7.2 corresponds to a 42-by-42 block with the same type of sparsity pattern. We show the smaller degree version to clearly illustrate the sparsity of these submatrices. The sparsity pattern of the saddle point system is tied to the operator matrix

$$\underline{A} = -\left(\underline{\tilde{S}}\,\underline{\tilde{S}}^T \otimes \underline{S}_{DL}\underline{S}_{DL}^T\right) - \left(\underline{S}_{DL}\underline{S}_{DL}^T \otimes \underline{\tilde{S}}\,\underline{\tilde{S}}^T\right) \tag{7.166}$$

for the $(1,1)$ block and the constraint matrix

$$\underline{C} = \begin{bmatrix} \left(\underline{I} \otimes \left(\bar{e}_1 - \bar{e}_2\right)^T\right) \\ \left(\underline{I} \otimes \left(\bar{e}_1 + \bar{e}_2\right)^T\right) \\ \left(\left(\bar{e}_1 - \bar{e}_2\right)^T \otimes \underline{I}\right) \\ \left(\left(\bar{e}_1 + \bar{e}_2\right)^T \otimes \underline{I}\right) \end{bmatrix} \tag{7.167}$$

for the $(2,1)$ block (actually the first two rows of the third and fourth block of $\underline{C}$ have been

removed to guarantee full rank, as described in Section 7.5). The transpose of $\underline{C}$ appears in the $(1,2)$ block.

For the second example, we solve Helmholtz's equation

$$\frac{\partial^2\phi}{\partial x_1^2}+\frac{\partial^2\phi}{\partial x_2^2}+k^2\phi=\exp\left\{-10\left[\left(x_1-\frac{1}{2}\right)^2+\left(x_2-1\right)^2\right]\right\} \tag{7.168}$$

subject to zero Dirichlet boundary conditions where wavenumber $k=9$. Using our notation, $\underline{\alpha}=-\underline{I}$, $\beta=k^2$, $f=\exp\{-10[(x_1-1/2)^2+(x_2-1)^2]\}$, and $p=0$. Here, the expansions for $\underline{\alpha}$ and β are straightforward to compute. Note that f is not a separable function of x_1 and x_2 but on the domain $(-1,1)^2$ it has rank 1. Using the same tolerance as for our first example, we obtain $K^{(f)}=1$ with $K_{x_1}^{(f)}=38$ and $K_{x_2}^{(f)}=35$.

Figure 7.3 illustrates a degree $L_{x_1}=L_{x_2}=38$ approximate solution along with the decay of coefficients used to represent the solution. Note that we still have the first two rows and columns in the coefficient matrix equal to zero to impose the zero Dirichlet boundary condition, but now that the solution has no symmetries, the remainder of the coefficients are nonzero. Figure 7.4 demonstrates a sample sparsity pattern of the saddle point system used to compute the solution in Figure 7.3. There is only a minor modification to the sparsity introduced by adding the term $k^2\phi$ to the operator. This makes the operator matrix $\underline{A}$ have three terms so that

$$\underline{A}=-\left(\underline{\tilde{S}}\,\underline{\tilde{S}}^T\otimes\underline{S}_{DL}\underline{S}_{DL}^T\right)-\left(\underline{S}_{DL}\underline{S}_{DL}^T\otimes\underline{\tilde{S}}\,\underline{\tilde{S}}^T\right)+k^2\left(\underline{\tilde{S}}\,\underline{\tilde{S}}^T\otimes\underline{\tilde{S}}\,\underline{\tilde{S}}^T\right). \tag{7.169}$$

This explains why the three bands in the sparsity pattern observed for the Poisson equation are now each pentadiagonal instead of diagonal, pentadiagonal, and diagonal respectively.

For the third example, we solve Laplace's equation

$$\frac{\partial^2\phi}{\partial x_1^2}+\frac{\partial^2\phi}{\partial x_2^2}=0 \tag{7.170}$$

subject to the inhomogeneous Dirichlet boundary conditions

$$\phi\left(-1,x_2\right)=0, \tag{7.171}$$

$$\phi\left(1,x_2\right)=\frac{1}{5}\sin\left(3\pi x_2\right), \tag{7.172}$$

$$\phi\left(x_1,-1\right)=0, \tag{7.173}$$

$$\phi\left(x_1,1\right)=\begin{cases}\left[\sin\left(\pi x\right)\right]^4 & x_1\leq 0\\ 0 & \text{otherwise.}\end{cases} \tag{7.174}$$

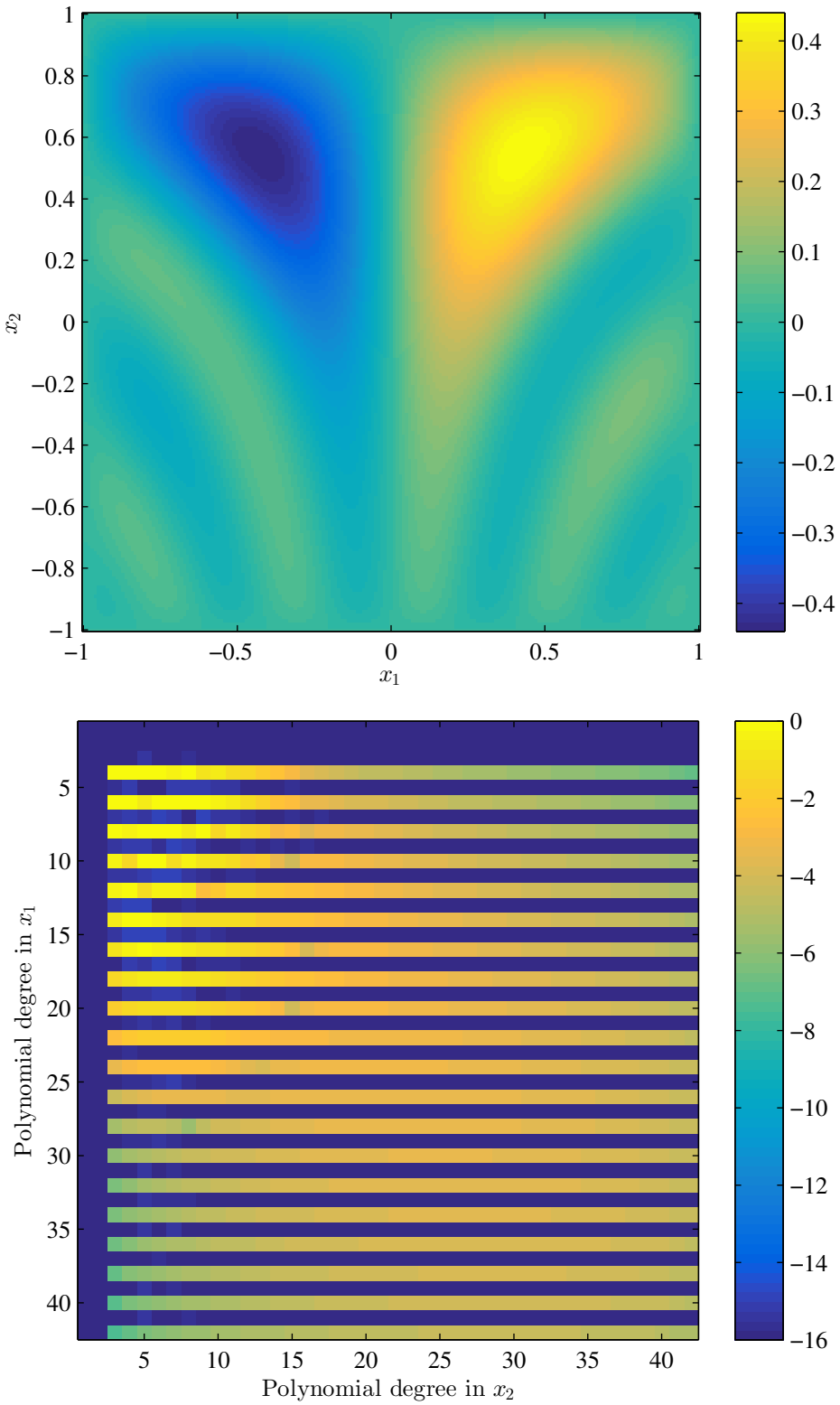

Figure 7.1: (top) Solution to Poisson's equation (7.165) subject to zero Dirichlet boundary conditions. (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

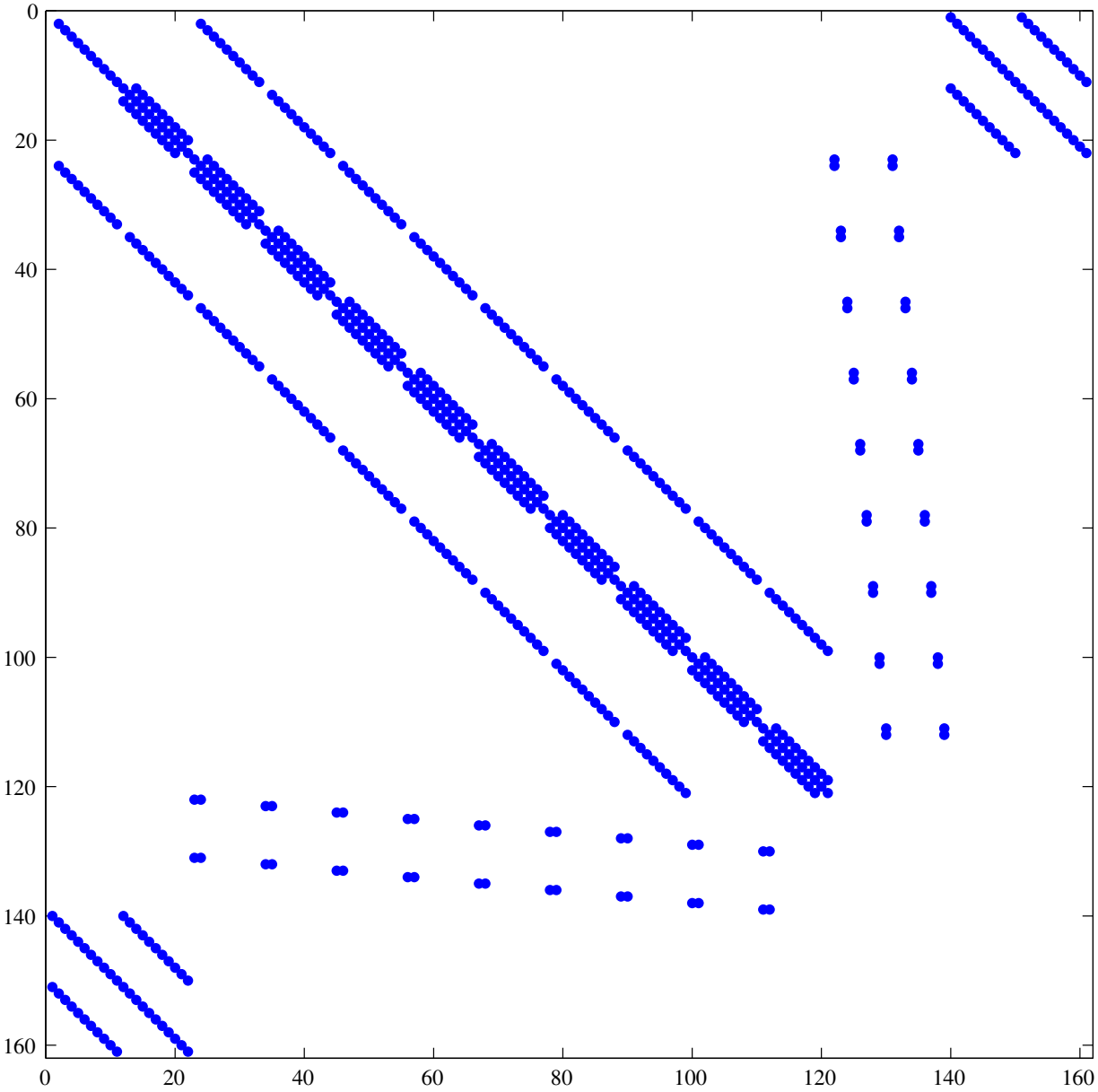

Figure 7.2: Example sparsity pattern of the saddle point system used to compute an approximate solution to (7.165) for a degree $L_{x_1} = L_{x_2} = 10$ solution (the solution in Figure 7.1 is degree 41).

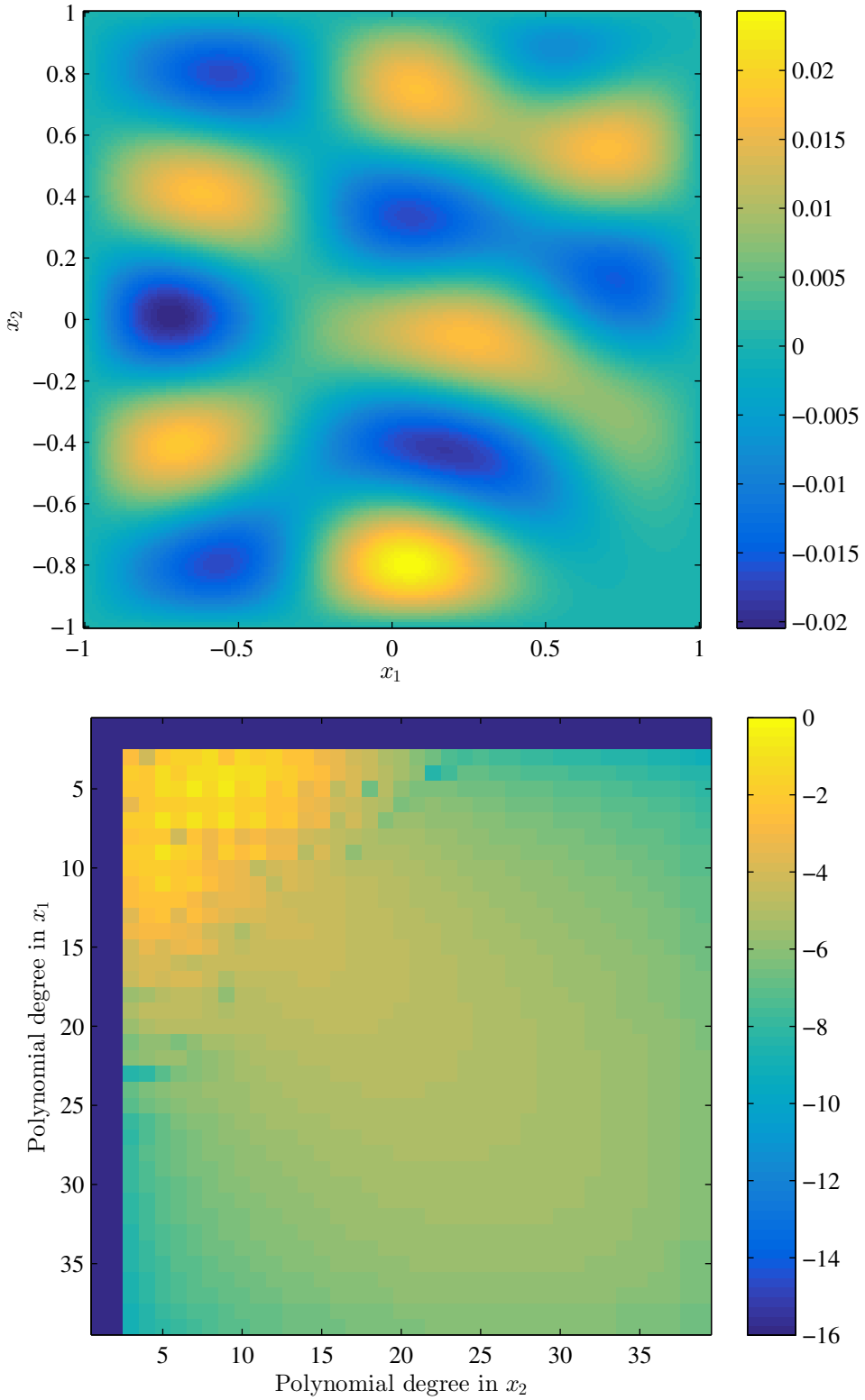

Figure 7.3: (top) Solution to Helmholtz's equation (7.168) subject to zero Dirichlet boundary conditions. (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

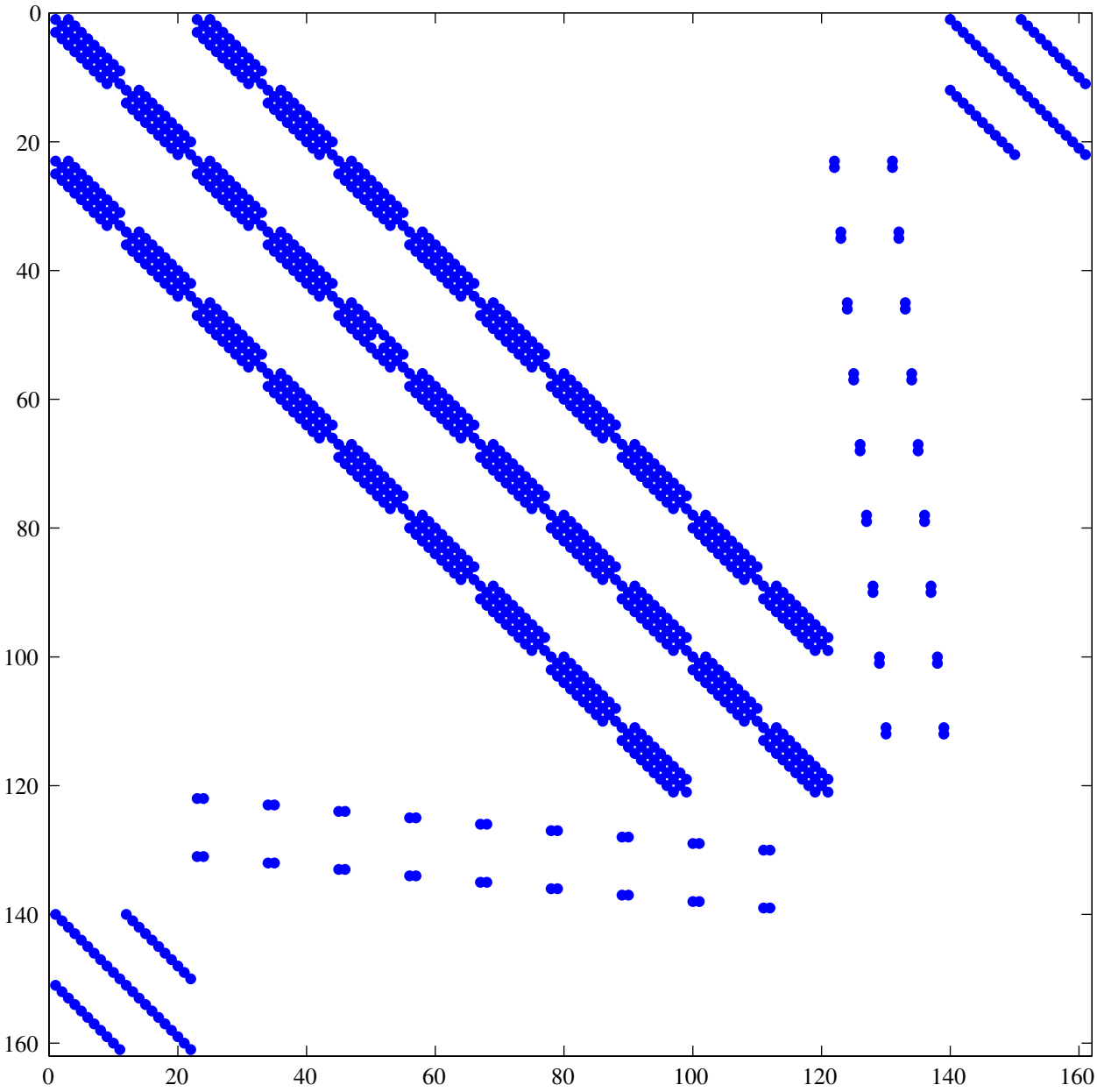

Figure 7.4: Example sparsity pattern of the saddle point system used to compute an approximate solution to (7.168) for a degree $L_{x_1} = L_{x_2} = 10$ solution (the solution in Figure 7.3 is degree 38).

Here, it takes $K_p^{(1)} = 0$, $K_p^{(2)} = 27$, $K_p^{(3)} = 0$, and $K_p^{(4)} = 509$ degree Legendre expansions to meet the $\epsilon_{\text{tol}} = 10^{-12}$ requirement for the Dirichlet boundary conditions. The condition on Γ_4 is particularly difficult because the function along that edge is not smooth (its fourth derivative is not continuous).

Figure 7.5 illustrates an approximate solution to this problem along with the decay of the coefficients used to represent the solution. We do not provide a sparsity diagram for this example since its pattern is identical to the one found in Figure 7.2 (although here we have degree $L_{x_1} = L_{x_2} = 509$). Note that our solution exhibits the maximum principle associated with Laplace's equation [148] although it is not explicitly enforced; its maximum and minimum are both achieved on the boundary.

For the fourth example, constructed to test spatially varying PDE coefficients, we solve the variable coefficient Helmholtz problem

$$\frac{\partial^2 \phi}{\partial x_1^2} + \frac{\partial^2 \phi}{\partial x_2^2} + \beta \phi = f \tag{7.175}$$

where $\underline{\alpha} = -\underline{I}$,

$$\beta\left(\bar{x}\right) = \left[x_1^2 + \left(x_2 + 1\right)^2\right] \sin\left[x_1 \left(x_2 + 1\right)^2\right], \tag{7.176}$$

$$f\left(\bar{x}\right) = \left[x_1^2 + \left(x_2 + 1\right)^2\right] \cos\left[x_1 \left(x_2 + 1\right)\right] \sin\left\{\cos\left[x_1 \left(x_2 + 1\right)\right]\right\}, \tag{7.177}$$

with inhomogeneous Dirichlet boundary conditions

$$p\left(\bar{x}\right) = \cos\left\{\cos\left[x_1 \left(x_2 + 1\right)\right]\right\} \tag{7.178}$$

on all boundary edges. Note that the solution to this equation is

$$\phi_{\text{exact}}\left(\bar{x}\right) = \cos\left\{\cos\left[x_1 \left(x_2 + 1\right)\right]\right\}. \tag{7.179}$$

In fact, this solution has been obtained via the method of manufactured solutions [149]. That is, upon selecting a sufficiently complicated expression for β and a desired solution ϕ_{exact}, we substitute the specification into the PDE and compute f so that the PDE is satisfied by construction. Finally, we select boundary conditions that the solution ϕ_{exact} satisfies exactly. The choice of β in this case was made such that it is not separable. The choice of ϕ_{exact} was made to excite several polynomial modes in the approximate solution. For this example, we use a tolerance of $\epsilon_{\text{tol}} = 10^{-14}$ (to match the accuracy in [146]) and find that we need $K^{(\beta)} = 64$ with $K_{x_1}^{(\beta)} = 20$ and $K_{x_2}^{(\beta)} = 17$, as well as $K^{(f)} = 64$ with $K_{x_1}^{(f)} = 32$ and $K_{x_2}^{(f)} = 22$. Note that we have actually limited K to 64 in our Legendre expansion

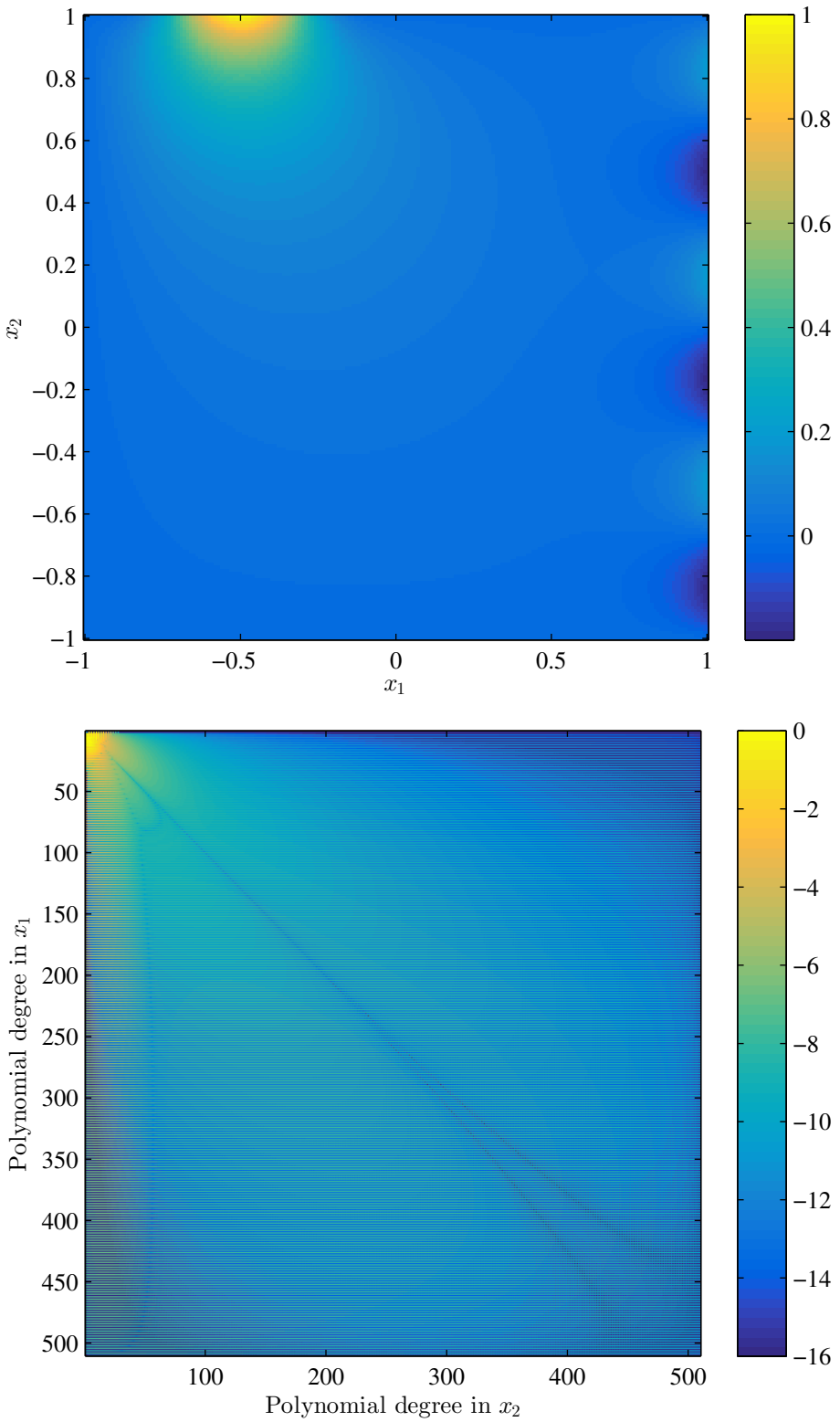

Figure 7.5: (top) Solution to Laplace's equation (7.170) subject to inhomogeneous Dirichlet boundary conditions. (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

implementation which explains why β and f are both expressed as 64 separable terms each. This does not adversely effect the accuracy of our computed solution.

Figure 7.6 illustrates the computed solution and the decay of coefficients in the degree $L_{x_1} = L_{x_2} = 32$ polynomial used to represent the solution. Figure 7.7 illustrates the pointwise error $|\phi - \phi_{\text{exact}}|$ between the computed and exact solutions (sampled with 201 uniformly spaced points in each coordinate), and the sparsity pattern of the saddle point system used to perform all calculations. Note that, in contrast to the previous examples presented in this chapter, the coefficients decay to somewhere between 10^{-13} and 10^{-14} in the highest order modes. This means that there is little to gain by increasing the degree of polynomial basis when computing a solution in double precision. This is borne out by computing the pointwise error $|\phi - \phi_{\text{exact}}|$ which is accurate to 14 digits throughout the domain (in many locations as good as 15 or 16 digits). Note that the sparsity of the saddle point matrix is severely compromised due to how large $K_{x_1}^{(\beta)}$ and $K_{x_2}^{(\beta)}$ are. This is because

$$\begin{aligned}\underline{A} = &-\left(\tilde{\underline{S}}\tilde{\underline{S}}^T \otimes \underline{S}_{DL}\underline{S}_{DL}^T\right) - \left(\underline{S}_{DL}\underline{S}_{DL}^T \otimes \tilde{\underline{S}}\tilde{\underline{S}}^T\right) \\ &+ \sum_{k=1}^{K^{(\beta)}} \sigma_k^{(\beta)} \left(\tilde{\underline{S}}\left[\sum_{i=0}^{K_{x_2}^{(\beta)}} v_{ik}^{(\beta)}\underline{T}_i\right]\tilde{\underline{S}}^T \otimes \tilde{\underline{S}}\left[\sum_{i=0}^{K_{x_1}^{(\beta)}} u_{ik}^{(\beta)}\underline{T}_i\right]\tilde{\underline{S}}^T\right)\end{aligned} \tag{7.180}$$

and the bandwidth of each term

$$\tilde{\underline{S}}\left[\sum_{i=0}^{K_{x_2}^{(\beta)}} v_{ik}^{(\beta)}\underline{T}_i\right]\tilde{\underline{S}}^T, \qquad \tilde{\underline{S}}\left[\sum_{i=0}^{K_{x_1}^{(\beta)}} u_{ik}^{(\beta)}\underline{T}_i\right]\tilde{\underline{S}}^T, \tag{7.181}$$

depends, as we have seen in one dimension, on the number of terms in the sum.

For the final example, we construct our own manufactured solution in order to test Robin boundary conditions, as well as variable $\underline{\alpha}$. We solve

$$-\nabla \cdot (\underline{\alpha}\nabla\phi) + \beta\phi = f \tag{7.182}$$

with

$$\underline{\alpha} = \begin{bmatrix} x_1 & x_1x_2 \\ x_1x_2 & x_2^3 \end{bmatrix}, \qquad \beta = k^2, \tag{7.183}$$

where wavenumber $k = 10$. We take the exact solution to be

$$\phi_{\text{exact}}(\bar{x}) = \exp\left[-(\bar{x} - \bar{c})^T(\bar{x} - \bar{c})\right], \qquad \bar{c} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{4} \end{bmatrix}, \tag{7.184}$$

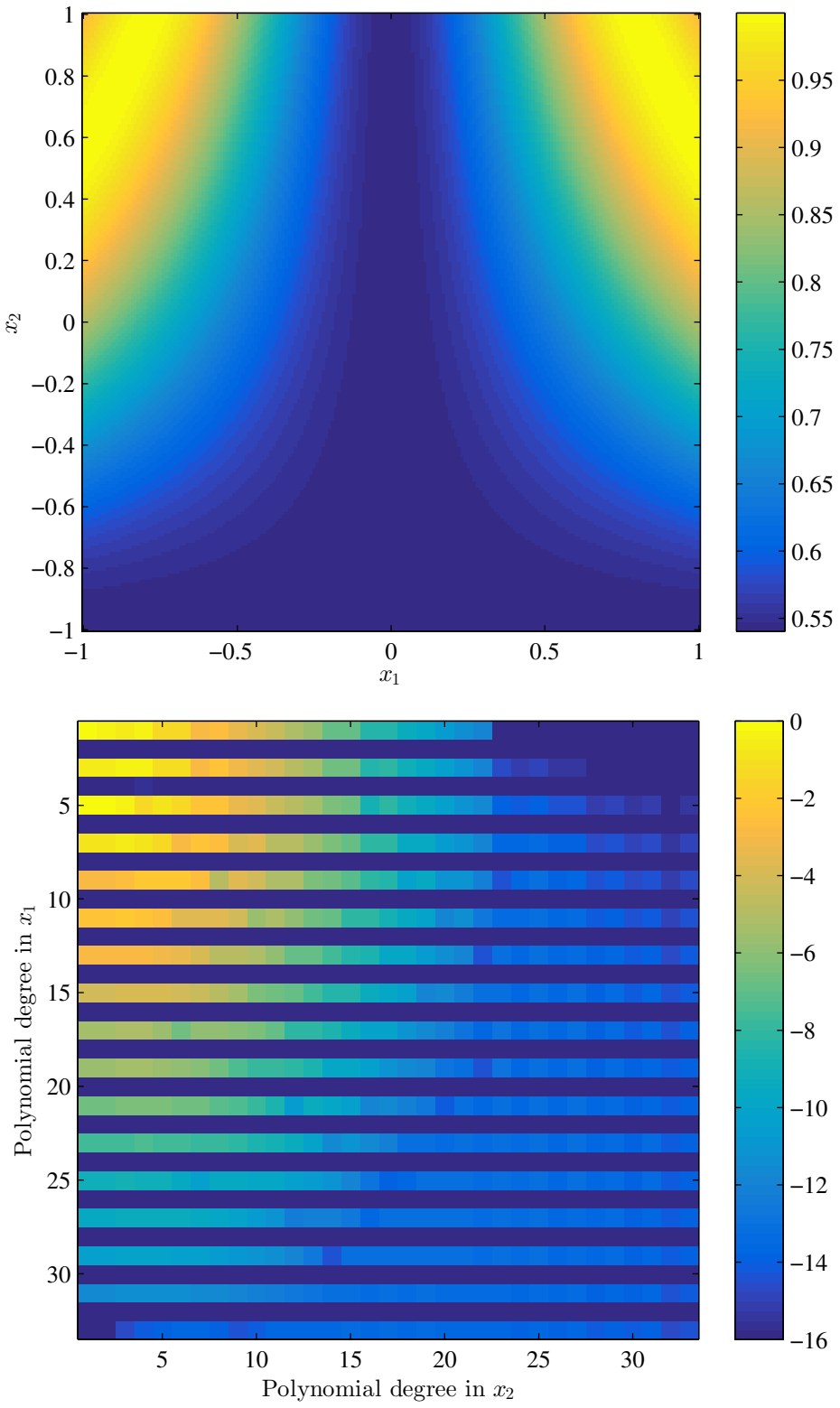

Figure 7.6: (top) Computed solution of the variable coefficient problem (7.175). (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

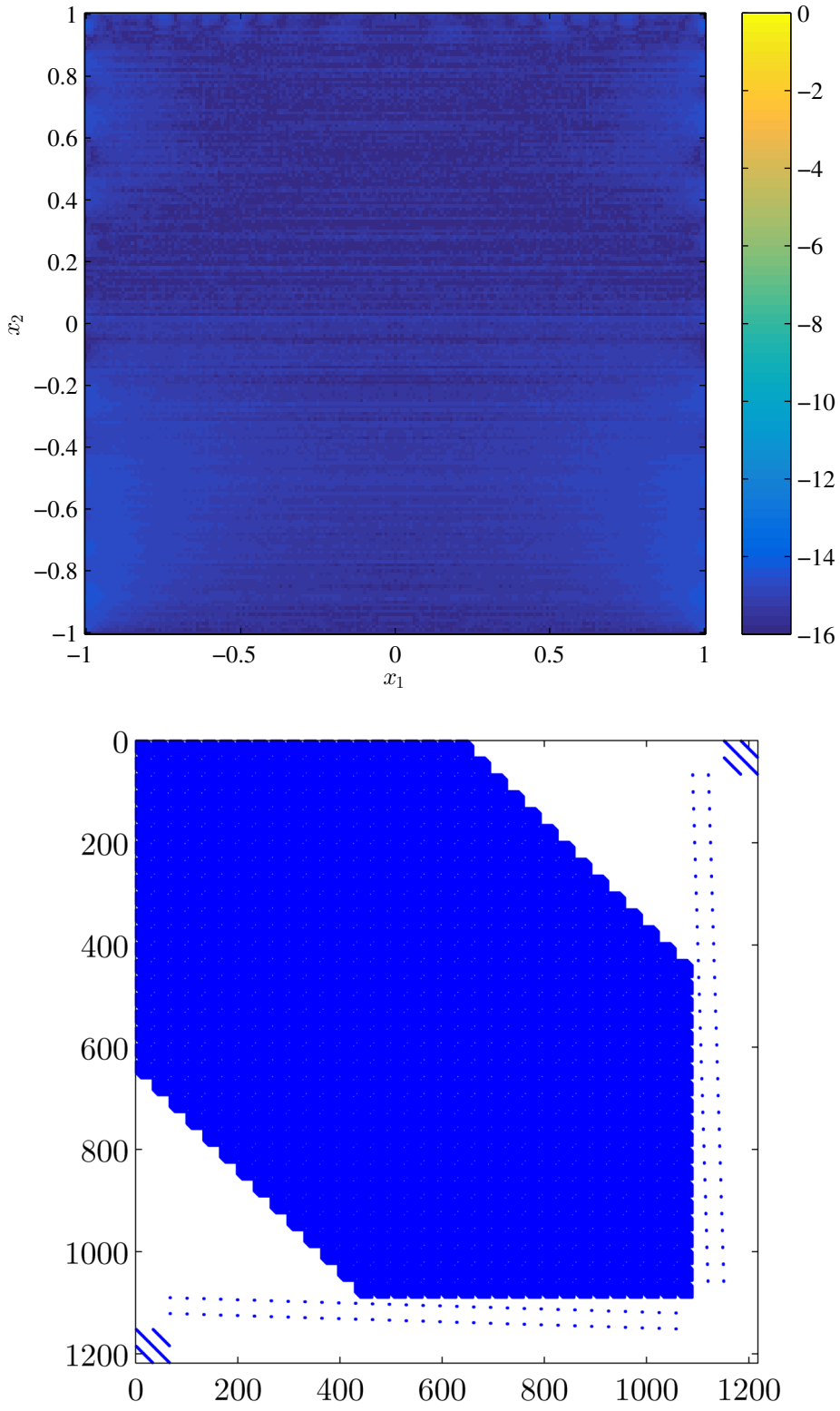

Figure 7.7: (top) Pointwise error $\log_{10}|\phi - \phi_{\text{exact}}|$ between the computed solution in Figure 7.6 and the exact solution. (bottom) Sparsity pattern of the saddle point system used to compute the solution.

so that

$$f = -\nabla \cdot (\underline{\alpha} \nabla \phi_{\text{exact}}) + \beta \phi_{\text{exact}} \tag{7.185}$$

and impose Dirichlet boundary conditions on edges Γ_1 and Γ_4 requiring $\phi = \phi_{\text{exact}}$. In addition, we impose Robin boundary conditions on edges Γ_2 and Γ_3 requiring

$$q_{(i)} = \bar{n}_i^T (\underline{\alpha} \nabla \phi_{\text{exact}}) + \gamma_{(i)} \phi_{\text{exact}} \tag{7.186}$$

where we specify $\gamma_{(2)} = \sin(x_2)$ and $\gamma_{(3)} = \exp(-x_1)$. Here, the Legendre expansions computed to a tolerance $\epsilon_{\text{tol}} = 10^{-14}$ require

$$\begin{aligned}
K^{(\alpha_{11})} &= 1, & K_{x_1}^{(\alpha_{11})} &= 1, & K_{x_2}^{(\alpha_{11})} &= 0, \\
K^{(\alpha_{12})} &= 1, & K_{x_1}^{(\alpha_{12})} &= 1, & K_{x_2}^{(\alpha_{12})} &= 1, \\
K^{(\alpha_{22})} &= 1, & K_{x_1}^{(\alpha_{22})} &= 0, & K_{x_2}^{(\alpha_{22})} &= 3, \\
K^{(\beta)} &= 1, & K_{x_1}^{(\beta)} &= 0, & K_{x_2}^{(\beta)} &= 0, \\
K^{(f)} &= 3, & K_{x_1}^{(f)} &= 23, & K_{x_2}^{(f)} &= 24, \\
& & K_p^{(1)} &= 19, & K_p^{(4)} &= 20, \\
& & K_q^{(2)} &= 23, & K_q^{(3)} &= 22, \\
& & K_\gamma^{(2)} &= 13, & K_\gamma^{(3)} &= 13.
\end{aligned}$$

Recall that all of these expansions are computed automatically once the user has specified the associated functions so, while their number may seem intimidating, very little work on the part of the user is required: they must simply supply a tolerance, as well as functions that take as input a vector $\bar{x}$ and return associated scalar values.

Figure 7.8 illustrates the computed solution ϕ and the associated decay of coefficients $\underline{\phi}$. Figure 7.9 illustrates the pointwise error $|\phi - \phi_{\text{exact}}|$ (sampled with 201 uniformly spaced points in each coordinate), and the sparsity pattern of the associated saddle point system. Note that while the solution may appear to the eye to vary slowly, the error plot shows that it oscillates in a complicated way about the exact solution. The eye does not pick up on this variation since the computed solution is accurate to roughly 13 digits over the domain (in many places it is more accurate). A degree $L_{x_1} = L_{x_2} = 24$ polynomial is sufficient to achieve this level of accuracy. The sparsity of the operator can again be analyzed. The first two-by-two dense blocks arise from the Robin boundary condition on edge Γ_3 whereas the sprinkling of nonzero entries off the main diagonal band arise due to the Robin boundary condition on edge Γ_2. The seven diagonal bands arise mostly from terms involving $\underline{\alpha}$ although the term

involving β does contribute. The operator matrix is given by

$$\begin{aligned}\underline{A} = \sigma_1^{(\alpha_{11})} &\left(\underline{\tilde{S}} \left[v_{01}^{(\alpha_{11})} \underline{T}_0 \right] \underline{\tilde{S}}^T \otimes \underline{S}_{DL} \left[\sum_{i=0}^{1} u_{i1}^{(\alpha_{11})} \underline{T}_i \right] \underline{S}_{DL}^T \right) \\ &+ \sigma_1^{(\alpha_{12})} \left(\underline{\tilde{S}} \left[\sum_{i=0}^{1} v_{i1}^{(\alpha_{12})} \underline{T}_i \right] \underline{S}_{DL}^T \otimes \underline{S}_{DL} \left[\sum_{i=0}^{1} u_{i1}^{(\alpha_{12})} \underline{T}_i \right] \underline{\tilde{S}}^T \right) \\ &+ \sigma_1^{(\alpha_{22})} \left(\underline{S}_{DL} \left[\sum_{i=0}^{3} v_{i1}^{(\alpha_{22})} \underline{T}_i \right] \underline{S}_{DL}^T \otimes \underline{\tilde{S}} \left[u_{01}^{(\alpha_{22})} \underline{T}_0 \right] \underline{\tilde{S}}^T \right) \\ &+ k^2 \left(\underline{\tilde{S}} \underline{\tilde{S}}^T \otimes \underline{\tilde{S}} \underline{\tilde{S}}^T \right) \\ &+ \left(\underline{\tilde{S}} \left[\sum_{k=0}^{13} \gamma_k^{(2)} \underline{T}_k \right] \underline{\tilde{S}}^T \otimes \frac{1}{2} \left(\bar{e}_1 + \bar{e}_2 \right) \left(\bar{e}_1 + \bar{e}_2 \right)^T \right) \\ &+ \left(\frac{1}{2} \left(\bar{e}_1 - \bar{e}_2 \right) \left(\bar{e}_1 - \bar{e}_2 \right)^T \otimes \underline{\tilde{S}} \left[\sum_{k=0}^{13} \gamma_k^{(3)} \underline{T}_k \right] \underline{\tilde{S}}^T \right) \end{aligned} \quad (7.187)$$

which we write out in detail here to illustrate how complicated such matrices can be. The process of assembling this matrix is automated.

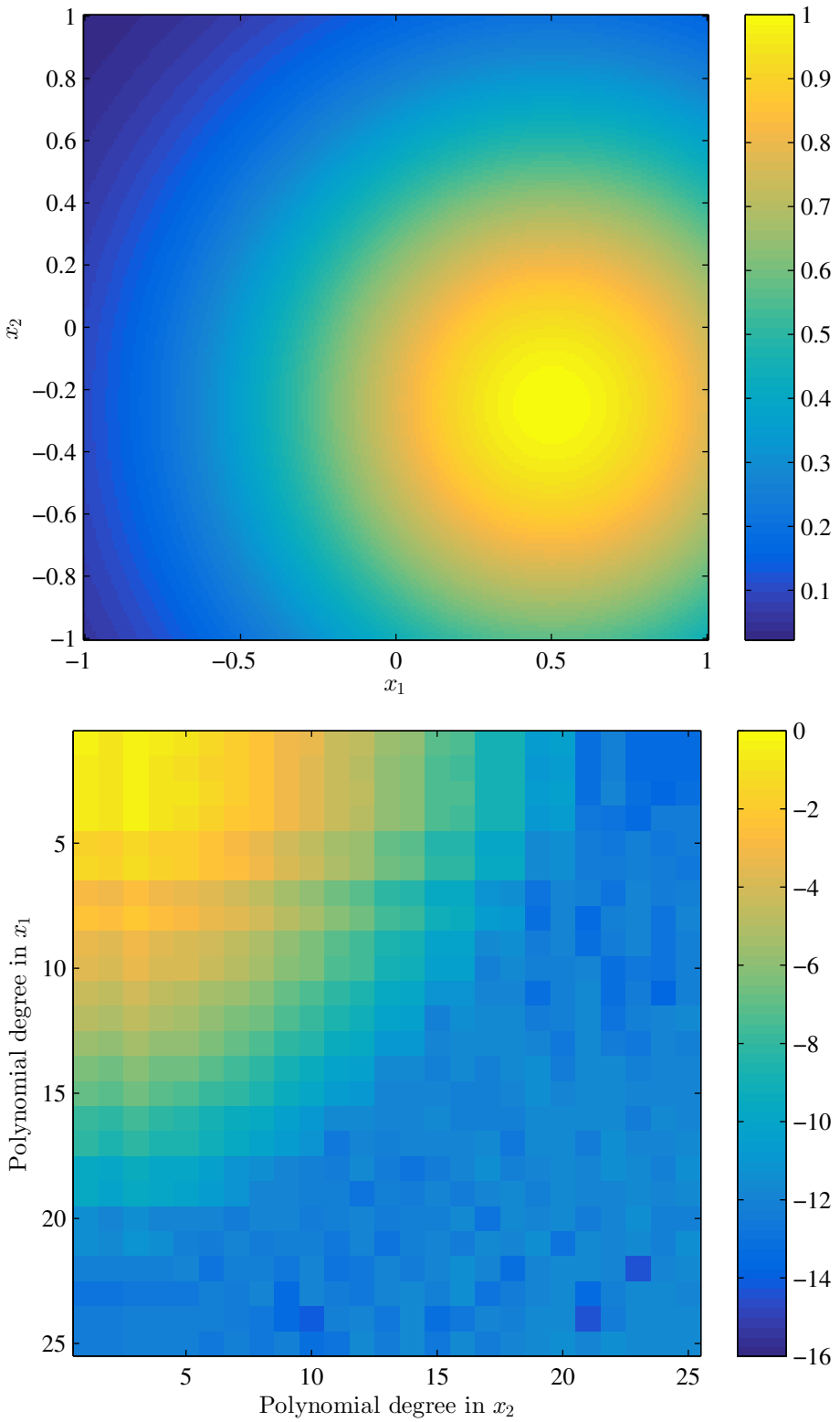

Figure 7.8: (top) Computed solution of the variable coefficient problem (7.182) with Robin boundary conditions. (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

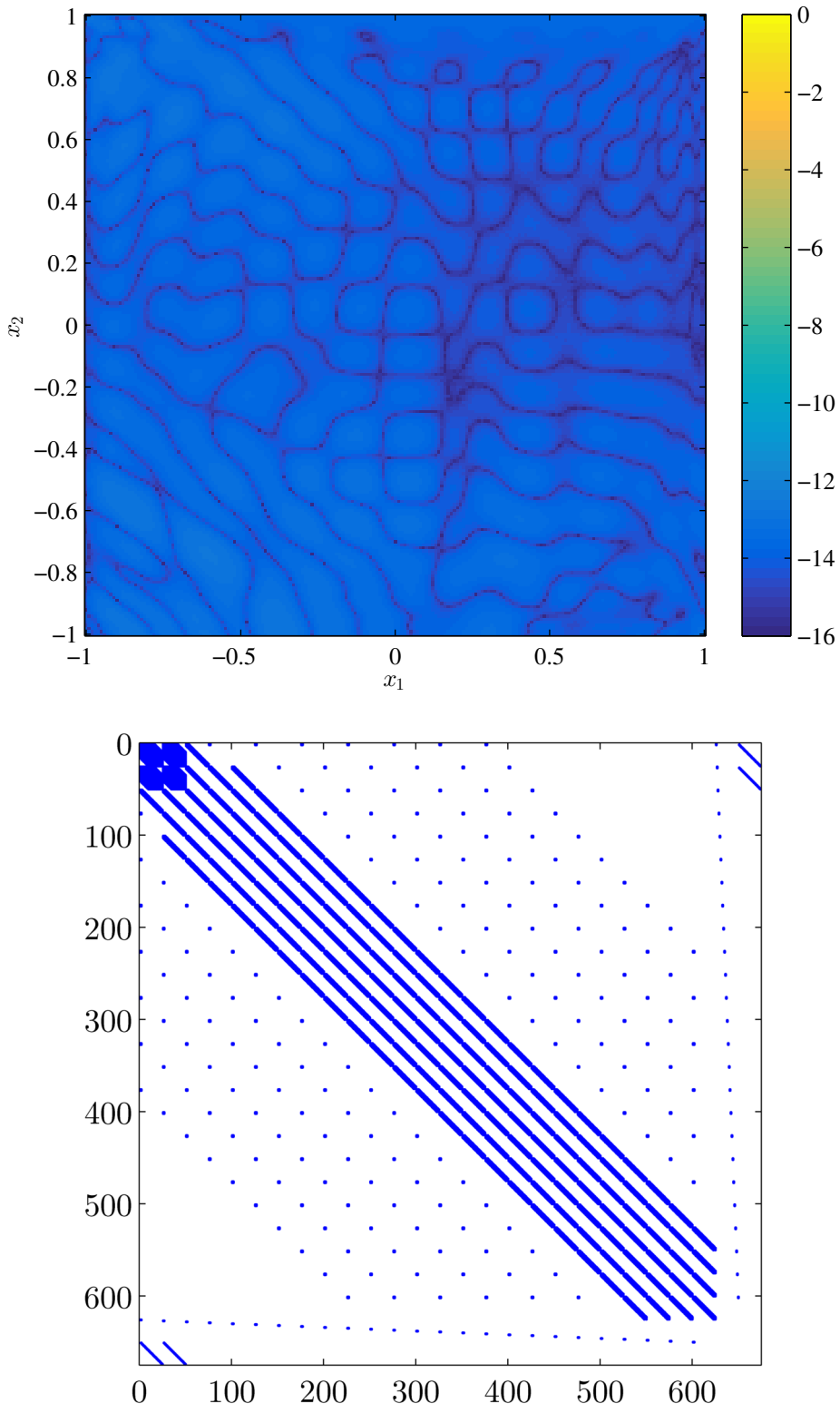

Figure 7.9: (top) Pointwise error $\log_{10}|\phi - \phi_{\text{exact}}|$ between the computed solution in Figure 7.8 and the exact solution. (bottom) Sparsity pattern of the saddle point system used to compute the solution.

Chapter 8

A Single Curvilinear Quadrilateral Element

Chapter 7 demonstrated that the solution of a wide variety of PDEs can be computed reliably to high accuracy using a single element on the domain $(-1,1)^2$. This chapter extends the ideas of Chapter 7 by considering the solution of PDEs on domains $\Omega \subset \mathbb{R}^2$ which arise as specific planar polynomial mappings $\bar{x} : (-1,1)^2 \to \Omega$ from the canonical domain $(-1,1)^2$. The notation $\bar{u} \in (-1,1)^2$ is used to denote spatial variables in the canonical domain and $\bar{x}(\bar{u}) \in \Omega$ to denote spatial variables in the curvilinear domain. If $\bar{x}$ is the identity map, then $\bar{x}(\bar{u}) = \bar{u}$ and the methods of the previous chapter are recovered. This chapter considers polynomial maps of a specific form. These maps originate from computational geometry contexts related to surface modeling [150], and have been used to approximate functions [151] and mesh certain geometries [152]. The approach goes by several names (Coons patches, for example) but the name transfinite interpolation which appears in [152] is used in this thesis. The computational geometry text [153] provides a clear, and practical overview of the matter.

The subsequent section describes how to compute the transfinite interpolation map given a description of the boundary of Ω by either parametric curves or implicit functions. The chapter goes on to show that the modifications required to extend the techniques from the previous chapter to curvilinear domains described by transfinite interpolation are straightforward; one simply replaces parameters $\underline{\alpha}$, β, f, p, γ, and q by effective parameters that take into account the map $\bar{x}(\bar{u})$.

8.1 Transfinite Interpolation

Before describing transfinite interpolation maps, we consider conventional isoparametric maps as in [12] (one of many finite element texts that uses such a map to model curved geometry) and argue why such maps may be poorly suited for our purpose. The name isoparametric means to use the same basis functions to represent the solution ϕ as we do to represent the components in the map $\bar{x}$. This makes the most sense when the basis functions representing ϕ are interpolatory since, in that case, we can place interpolation nodes on the curved boundary of the geometry we wish to approximate with the polynomial map. Because we use a modal basis which is not interpolatory to represent ϕ, we consider an "isoparametric" map[1] given by

$$\bar{x}\left(\bar{u}\right) = \underline{X}\bar{l}\left(\bar{u}\right) \tag{8.1}$$

where

$$\underline{X} = \left[\begin{array}{cccc} \bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_{(N_{u_1}+1)\times(N_{u_2}+1)} \end{array} \right] \tag{8.2}$$

is a matrix whose columns are coordinates of interpolation nodes and

$$\bar{l}\left(\bar{u}\right) = \bar{l}\left(u_2\right) \otimes \bar{l}\left(u_1\right) \tag{8.3}$$

are tensor products of one-dimensional Lagrange interpolating polynomials with degree N_{u_1} polynomials in the u_1 coordinate and degree N_{u_2} polynomials in the u_2 coordinate. The difficulty with such a formulation is that although it may be straightforward to determine where to interpolate along the four curvilinear boundaries of the domain Ω (although even this may not be obvious), it leaves the additional task of having to specify internal interpolation nodes inside Ω. To avoid dealing with the interior of Ω, we use transfinite interpolation.

The name transfinite interpolation arises because if the boundary of Ω is specified by four parametric functions each describing one curvilinear edge, then the map $\bar{x}$ reproduces the edges exactly. Thus the trans- prefix for transfinite means that we go beyond interpolating at a finite number of points. To see how such a map is produced, suppose that there is some function $\bar{G}\left(\bar{u}\right)$ that satisfies $\bar{G} : \left(-1,1\right)^2 \to \Omega$. Now, define the function

$$\left[P_1\bar{G}\right]\left(\bar{u}\right) = \frac{1-u_1}{2}\bar{G}\left(-1,u_2\right) + \frac{1+u_1}{2}\bar{G}\left(1,u_2\right). \tag{8.4}$$

Note that the functions $(1-u_1)/2$ and $(1+u_1)/2$ are the degree one interpolatory Lagrange polynomials, thus they are equal to 1 and 0 respectively when $u_1 = -1$, and 0 and 1

[1] We put isoparametric in quotation marks here to stress that we are not exactly matching the basis functions used to represent the geometry with the basis functions to represent the solution ϕ. That being said, the basis functions we are using to describe the geometry are the standard isoparametric basis functions.

respectively when $u_1 = 1$. As a consequence, the function $P_1\bar{G}$ is exactly equal to $\bar{G}(-1, u_2)$ when $u_1 = -1$ and exactly equal to $\bar{G}(1, u_2)$ when $u_1 = 1$. When $-1 < u_1 < 1$, $P_1\bar{G}$ smoothly blends the two boundary functions together. We can define a second function

$$[P_2\bar{G}](\bar{u}) = \frac{1-u_2}{2}\bar{G}(u_1, -1) + \frac{1+u_2}{2}\bar{G}(u_1, 1) \tag{8.5}$$

which performs a similar blending operation, but applied to the edges corresponding to $u_2 = -1$ and $u_2 = 1$.

To construct a function that matches $\bar{G}$ on all four edges, we follow a two step process. If we take the difference between $\bar{G}$ and $P_1\bar{G}$, we obtain an error function

$$\bar{E}_1 = \bar{G} - P_1\bar{G} \tag{8.6}$$

which, by construction, is zero on the edges corresponding to $u_1 = -1$ and $u_1 = 1$. Note however that the behavior on the edges corresponding to $u_2 = -1$ and $u_2 = 1$ may still be nonzero. We can then apply the same approach to zero the edges corresponding to $u_2 = -1$ and $u_2 = 1$ using the function $P_2\bar{E}_1$. This gives

$$\begin{aligned}
\bar{E}_2 &= \bar{E}_1 - P_2\bar{E}_1 && (8.7)\\
&= (\bar{G} - P_1\bar{G}) - P_2(\bar{G} - P_1\bar{G}) && (8.8)\\
&= \bar{G} - P_1\bar{G} - P_2\bar{G} + P_2(P_1\bar{G}). && (8.9)
\end{aligned}$$

Note that the function $\bar{E}_2$ is now zero on all four edges of the domain, although, on the interior, it may still be nonzero. Thus on the boundary of $(-1, 1)^2$ and, consequently, on the boundary of Ω, we have

$$\begin{aligned}
0 &= \bar{G} - P_1\bar{G} - P_2\bar{G} + P_2(P_1\bar{G}) && (8.10)\\
\bar{G} &= P_1\bar{G} + P_2\bar{G} - P_2(P_1\bar{G}). && (8.11)
\end{aligned}$$

Since

$$\begin{aligned}
P_1\bar{G} + P_2\bar{G} - P_2(P_1\bar{G}) = {} & \frac{1-u_1}{2}\bar{G}(-1, u_2) + \frac{1+u_1}{2}\bar{G}(1, u_2)\\
& + \frac{1-u_2}{2}\bar{G}(u_1, -1) + \frac{1+u_2}{2}\bar{G}(u_1, 1)\\
& - \frac{1-u_1}{2}\frac{1-u_2}{2}\bar{G}(-1, -1) - \frac{1+u_1}{2}\frac{1-u_2}{2}\bar{G}(1, -1)\\
& - \frac{1-u_1}{2}\frac{1+u_2}{2}\bar{G}(-1, 1) - \frac{1+u_1}{2}\frac{1+u_2}{2}\bar{G}(1, 1), \qquad (8.12)
\end{aligned}$$

we see that this interpolant only requires that we specify $\bar{G}$ on the boundary. Thus, if we have four parametric representations $\bar{x}_1(-1, u_2)$, $\bar{x}_2(1, u_2)$, $\bar{x}_3(u_1, -1)$, and $\bar{x}_4(u_1, 1)$, of the four curvilinear edges of Ω, we can define a transfinite map

$$\begin{aligned}\bar{x}(\bar{u}) = &\frac{1-u_1}{2}\bar{x}_1(-1, u_2) + \frac{1+u_1}{2}\bar{x}_2(1, u_2) \\ &+ \frac{1-u_2}{2}\bar{x}_3(u_1, -1) + \frac{1+u_2}{2}\bar{x}_4(u_1, 1) \\ &- \frac{(1-u_1)(1-u_2)}{4}\bar{x}_1(-1, -1) - \frac{(1+u_1)(1-u_2)}{4}\bar{x}_2(1, -1) \\ &- \frac{(1-u_1)(1+u_2)}{4}\bar{x}_1(-1, 1) - \frac{(1+u_1)(1+u_2)}{4}\bar{x}_2(1, 1) \end{aligned} \tag{8.13}$$

where the choice of which edges to use in evaluation at the vertices of the domain is not unique (we assume that pairs of adjacent curvilinear edges meet at the their shared vertices so that this ambiguity is removed). This approach can be extended to higher dimensions as needed.

8.2 Polynomial Representation for Explicit and Implicit Boundaries

We choose to represent each parametric curve as a polynomial

$$\bar{x}_1(-1, u_2) = \underline{X}_1 \bar{l}(u_2), \tag{8.14}$$

$$\bar{x}_2(1, u_2) = \underline{X}_2 \bar{l}(u_2), \tag{8.15}$$

$$\bar{x}_3(u_1, -1) = \underline{X}_3 \bar{l}(u_1), \tag{8.16}$$

$$\bar{x}_4(u_1, 1) = \underline{X}_4 \bar{l}(u_1), \tag{8.17}$$

where each matrix $\underline{X}_i$ contains $N_i + 1$ interpolation nodes $\bar{x}_i$ for a degree N_i interpolating polynomial basis. Comparing this representation of the domain with the isoparametric representation discussed earlier shows that interpolation is performed on codimension-1 objects rather than in the full space. In two dimensions, the transfinite interpolation with polynomial edges requires one-dimensional interpolation along curves representing the boundary $\partial\Omega$ whereas isoparametric interpolation requires two-dimensional interpolation inside the domain Ω. As in Chapter 2, we can always convert these Lagrange polynomials to Legendre polynomials through the identity $\bar{l}(u_j) = \underline{\tilde{V}}^{-T}\bar{p}(u_j)$ should the need arise. In fact, the final representation for the curves that we work with uses Legendre polynomials. In such a

setting, the four boundary curves take the form

$$\bar{x}_1(-1, u_2) = \underline{\tilde{X}}_1 \bar{p}(u_2), \tag{8.18}$$

$$\bar{x}_2(1, u_2) = \underline{\tilde{X}}_2 \bar{p}(u_2), \tag{8.19}$$

$$\bar{x}_3(u_1, -1) = \underline{\tilde{X}}_3 \bar{p}(u_1), \tag{8.20}$$

$$\bar{x}_4(u_1, 1) = \underline{\tilde{X}}_4 \bar{p}(u_1). \tag{8.21}$$

If the boundary curves are specified explicitly by parametric curves there is a straightforward way of determining an accurate polynomial representation. We repeat the same process for each boundary curve. We start by sampling the given parametric curve $\bar{x}_i(t)$ at Chebyshev points

$$t_k = -\cos\left(\frac{\pi}{N_i}k\right) \tag{8.22}$$

for $k = 0, 1, ..., N_i$, where t is either u_1 or u_2, depending on the boundary edge. We then convert these samples into two sets of Chebyshev coefficients (one for each component of the curve) using the FFT (as described in Section A.1 of Appendix A) and verify whether the coefficients have decayed below a user specified tolerance ϵ_{tol} (note that the decay must be met for both components of the curve). If we have not met the tolerance, we increase the number of samples and repeat the process until the tolerance has been met, or until a maximum polynomial degree is attained (this can be user specified). We finally convert from Chebyshev coefficients to Legendre coefficients using the techniques described in Section 4.2 and Appendix A. This yields the appropriate matrix $\underline{\tilde{X}}_i$ that captures the behavior of the boundary edge.

If the boundary components are specified implicitly by level set functions (which will be the case for the methods we investigate in later chapters), then we use a different algorithm to compute the interpolation nodes. Given a starting node $\bar{x}_{\text{start}}$ and ending node $\bar{x}_{\text{end}}$ that both lie on the implicit curve (how they may be specified is described in the next chapter), we construct the line segment connecting the two points

$$\bar{y}(t) = \frac{1-t}{2}\bar{x}_{\text{start}} + \frac{1+t}{2}\bar{x}_{\text{end}} \tag{8.23}$$

with $-1 \leq t \leq 1$. Generally, this line segment does not lie entirely in the boundary edge defined by the level set function $\Phi(\bar{x})$ whose zero contour

$$\Gamma = \left\{\bar{x} \in \mathbb{R}^2 \,:\, \Phi(\bar{x}) = 0\right\} \tag{8.24}$$

defines the curvilinear boundary edge we wish to represent with polynomials. On that line

segment, we define a set of Chebyshev nodes by sampling $\bar{y}(t)$ at the points (8.22). We then project the sampled points $\bar{y}_k = \bar{y}(t_k)$ onto the set Γ using the initial iterate $\bar{x}_k^{(0)} = \bar{y}_k$ and iteration

$$\bar{x}_k^{(i+1)} = \bar{x}_k^{(i)} - \frac{\Phi(\bar{x}_k^{(i)})}{\|\nabla\Phi(\bar{x}_k^{(i)})\|_2^2}\nabla\Phi(\bar{x}_k^{(i)}). \tag{8.25}$$

Note that the subscript k ranges over all nodes but that this iteration does not couple the nodes (and thus the iteration can be performed in parallel for all k). We iterate until a maximum number of iterations is achieved or until all points satisfy $|\Phi(\bar{x}_k^{(i+1)})| < \epsilon_{\text{tol}}$ where ϵ_{tol} is a user defined tolerance[2]. If such a condition is met, all of the projected nodes are deemed to lie sufficiently close to the boundary. Finally, we construct the interpolating polynomial curve $\underline{X}_j\bar{l}(u_j)$ using these nodes (the nodes form the columns of $\underline{X}_j$). Unfortunately, in performing this iteration, the nodes have lost their Chebyshev spacing so there is no fast transform available. Therefore, either we sample this polynomial at Chebyshev nodes and perform the transform to Legendre coefficients to check for sufficient decay, or we oversample the polynomial curve (say by a multiplicative factor of 10) and check that all sample points meet the tolerance $|\Phi(\bar{x})| < \epsilon_{\text{tol}}$. If this tolerance is met, we compute the matrix $\tilde{\underline{X}}_j$ of Legendre coefficients used to represent the boundary edge.

A few comments on the implicit boundary approach are needed. First, this outlined scheme is far from robust, particularly in cases where the boundary between points $\bar{x}_{\text{start}}$ and $\bar{x}_{\text{end}}$ cannot be written explicitly as a function in coordinates given by the line segment $\bar{y}$. That is, we need an implicit function theorem relative to line segment $\bar{y}$ to hold for the boundary Γ [155]. In practice, we construct cases where this is always true so that the method is useful. In addition, there is no guarantee that the projection of the nodes from the line segment to the boundary are distributed in such a way that the resulting interpolatory polynomial does not oscillate inappropriately. In practice, because the nodes along the line segment are clustered near the endpoints, for simple curved boundaries, the projected points also cluster near the endpoints and the interpolation does not oscillate inappropriately.

Further comment is also warranted regarding the projection iteration (8.25). In particular, the iteration comes from a first order approximation to the solution of the optimization

[2]The projection scheme (8.25) is exact and only requires one iteration when the level set function Φ is a smooth signed distance function. A signed distance function is a level set function satisfying $\|\nabla\Phi\|_2 = 1$ almost everywhere. In such a case the update equation is

$$\bar{x}^{(1)} = \bar{x}^{(0)} - \Phi(\bar{x}^{(0)})\nabla\Phi(\bar{x}^{(0)}). \tag{8.26}$$

Note that, for signed distance functions, $\Phi(\bar{x}^{(0)})$ returns the signed distance to the zero level set (its sign depends on whether $\bar{x}^{(0)}$ satisfies $\Phi(\bar{x}^{(0)}) < 0$ or $\Phi(\bar{x}^{(0)}) > 0$) and $\nabla\Phi(\bar{x}^{(0)})$ is directed normal to the boundary. These two properties explain why only one iteration is required to find the closest point to $\bar{x}^{(0)}$ on the boundary for such level set functions. For more detail, see [154].

problem

$$\begin{aligned} \min_{\overline{\Delta x}} \quad & \frac{1}{2}\|\overline{\Delta x}\|_2^2 \\ \text{s.t.} \quad & \Phi(\bar{x}^{(i)} + \overline{\Delta x}) = 0 \end{aligned} \tag{8.27}$$

where we have dropped the subscript k since each node can be treated independently. Such an optimization problem seeks the vector $\overline{\Delta x}$ which, when added to the known vector $\bar{x}^{(i)}$, causes $\bar{x}^{(i)} + \overline{\Delta x}$ to lie on the zero level contour of Φ and which also minimizes the distance from $\bar{x}^{(i)}$ to the contour. The method is first order because we replace the equality constraint with a first order Taylor approximation. That is, we expand

$$\Phi(\bar{x}^{(i)} + \overline{\Delta x}) = \Phi(\bar{x}^{(i)}) + \nabla\Phi(\bar{x}^{(i)})^T\overline{\Delta x} + \cdots \tag{8.28}$$

and ignore the quadratic and higher order terms. To solve the optimization problem, we write the Lagrangian of (8.27) as

$$L = \frac{1}{2}\|\overline{\Delta x}\|_2^2 + \nu\Phi(\bar{x}^{(i)} + \overline{\Delta x}) \tag{8.29}$$

with Lagrange multiplier ν and use the Taylor series approximation to obtain

$$L \approx \frac{1}{2}\|\overline{\Delta x}\|_2^2 + \nu\left[\Phi(\bar{x}^{(i)}) + \nabla\Phi(\bar{x}^{(i)})^T\overline{\Delta x}\right]. \tag{8.30}$$

To minimize such a function, we require that the KKT conditions (noted in Section 3.2) be satisfied. That is, the gradient with respect to $\overline{\Delta x}$ of the Lagrangian must be zero and equality constraints must be satisfied. The first condition yields

$$\nabla_{\overline{\Delta x}} L = \overline{\Delta x} + \nu\nabla\Phi(\bar{x}^{(i)}) = 0 \tag{8.31}$$

so that

$$\overline{\Delta x} = -\nu\nabla\Phi(\bar{x}^{(i)}). \tag{8.32}$$

The second condition stipulates that

$$\Phi(\bar{x}^{(i)}) + \nabla\Phi(\bar{x}^{(i)})^T\overline{\Delta x} = 0. \tag{8.33}$$

Substituting (8.32) into the equality constraint yields

$$\Phi(\bar{x}^{(i)}) + \nabla\Phi(\bar{x}^{(i)})^T\left[-\nu\nabla\Phi(\bar{x}^{(i)})\right] = 0 \tag{8.34}$$

which we solve for the Lagrange multiplier

$$\nu = \frac{\Phi(\bar{x}^{(i)})}{\|\nabla\Phi(\bar{x}^{(i)})\|_2^2}. \tag{8.35}$$

Combining this expression with (8.32) gives the required update

$$\bar{x}^{(i+1)} = \bar{x}^{(i)} + \overline{\Delta x} \tag{8.36}$$

$$= \bar{x}^{(i)} - \frac{\Phi(\bar{x}^{(i)})}{\|\nabla\Phi(\bar{x}^{(i)})\|_2^2}\nabla\Phi(\bar{x}^{(i)}). \tag{8.37}$$

In practice, we do not always have access to the gradient of the implicit function Φ. As a result, we compute the gradient using first order finite differences where the jth component of the gradient is approximated as

$$\frac{\partial\Phi}{\partial x_j} \approx \frac{\Phi(\bar{x}^{(i)} + \epsilon_{\text{machine}}^{1/2}\bar{e}_j) - \Phi(\bar{x}^{(i)})}{\epsilon_{\text{machine}}^{1/2}}. \tag{8.38}$$

For a second order update scheme which is more involved, see [64]. Note that although this scheme is first order, it is first order in computing the closest point on the boundary to the starting point $\bar{y}_k$. Since we are really only interested in projecting the point $\bar{y}_k$ to the boundary for the purposes of later interpolating, as long as the point is on the boundary to high precision, we do not need the point to be closest to $\bar{y}_k$. As a result, this first order scheme is sufficient and does not typically degrade the accuracy of subsequent computed PDE solutions.

8.3 Solving PDEs on Curvilinear Quadrilaterals

Now that we have seen how to construct a map $\bar{x} : (-1,1)^2 \to \Omega$, we consider the question of how to solve the PDE

$$-\nabla\cdot(\underline{\alpha}\nabla\phi) + \beta\phi = f \qquad \text{in } \Omega, \tag{8.39}$$

subject to boundary conditions

$$\phi = p \qquad \text{on } \Gamma_D, \tag{8.40}$$

$$\bar{n}^T(\underline{\alpha}\nabla\phi) + \gamma\phi = q \qquad \text{on } \Gamma_R. \tag{8.41}$$

The key is to perform all operations on the canonical domain $(-1,1)^2$ rather than on the more complicated curvilinear domain Ω. Doing so modifies the various quantities $\underline{\alpha}$, β, f, p,

γ, and q.

Recall that, while applying Galerkin's method, we found the weighted residual equation (6.44) which we repeat here for ease of reference:

$$\int_\Omega \nabla\psi^T \underline{\alpha}\nabla\phi \, d\Omega + \int_\Omega \psi\beta\phi \, d\Omega + \int_{\Gamma_R} \psi\gamma\phi \, d\Omega - \int_{\Gamma_D} \psi\bar{n}^T \left(\underline{\alpha}\nabla\phi\right) d\Omega = \int_\Omega \psi f \, d\Omega + \int_{\Gamma_R} \psi q \, d\Omega. \tag{8.42}$$

In addition, we also enforced Dirichlet boundary conditions via the integral constraint

$$\int_{\Gamma_D} \psi_B \left(\phi - p\right) d\Omega = 0. \tag{8.43}$$

In the following section, we consider how to transform each of these integrals back to the canonical domain via the map $\bar{x}\left(\bar{u}\right)$. We find that we can write each integral on the canonical domain in precisely the same form as in Chapter 7 but with new effective coefficients $\underline{\alpha}_{\text{eff}}$, β_{eff}, f_{eff}, p_{eff}, γ_{eff}, and q_{eff}, that depend on the original coefficients, along with factors that depend on the map $\bar{x}\left(\bar{u}\right)$. As a result, constructing the associated matrices and vectors to compute approximate solutions to the PDE on Ω is identical to constructing them for the canonical domain $\left(-1,1\right)^2$, differing only in the Legendre expansions required to accurately represent the variable coefficients.

Before we treat each term in the weighted residual formulation, we begin by specifying how the gradient of a function transforms under the map $\bar{x}\left(\bar{u}\right)$. We will call the components of the map x_1 and x_2 since they correspond to coordinates in the domain Ω. Using polynomial interpolation to represent the curvilinear boundaries of domain Ω, as in Section 8.2, we have the transfinite interpolation map

$$\begin{aligned}\bar{x}\left(\bar{u}\right) = \frac{1-u_1}{2}\underline{\tilde{X}}_1\bar{p}\left(u_2\right) + \frac{1+u_1}{2}\underline{\tilde{X}}_2\bar{p}\left(u_2\right) \\ + \frac{1-u_2}{2}\underline{\tilde{X}}_3\bar{p}\left(u_1\right) + \frac{1+u_2}{2}\underline{\tilde{X}}_4\bar{p}\left(u_1\right) \\ - \frac{\left(1-u_1\right)\left(1-u_2\right)}{4}\underline{\tilde{X}}_1\bar{p}\left(-1\right) - \frac{\left(1+u_1\right)\left(1-u_2\right)}{4}\underline{\tilde{X}}_2\bar{p}\left(-1\right) \\ - \frac{\left(1-u_1\right)\left(1+u_2\right)}{4}\underline{\tilde{X}}_1\bar{p}\left(1\right) - \frac{\left(1+u_1\right)\left(1+u_2\right)}{4}\underline{\tilde{X}}_2\bar{p}\left(1\right).\end{aligned} \tag{8.44}$$

Using the chain rule we obtain

$$\frac{\partial\phi}{\partial u_1} = \frac{\partial\phi}{\partial x_1}\frac{\partial x_1}{\partial u_1} + \frac{\partial\phi}{\partial x_2}\frac{\partial x_2}{\partial u_1}, \tag{8.45}$$

$$\frac{\partial\phi}{\partial u_2} = \frac{\partial\phi}{\partial x_1}\frac{\partial x_1}{\partial u_2} + \frac{\partial\phi}{\partial x_2}\frac{\partial x_2}{\partial u_2}. \tag{8.46}$$

Rewriting this expression with matrices and vectors yields

$$\underbrace{\begin{bmatrix} \dfrac{\partial \phi}{\partial u_1} \\ \dfrac{\partial \phi}{\partial u_2} \end{bmatrix}}_{\nabla_{\bar{u}}\phi} = \underbrace{\begin{bmatrix} \dfrac{\partial x_1}{\partial u_1} & \dfrac{\partial x_2}{\partial u_1} \\ \dfrac{\partial x_1}{\partial u_2} & \dfrac{\partial x_2}{\partial u_2} \end{bmatrix}}_{\underline{J}_{\bar{x}}^T} \underbrace{\begin{bmatrix} \dfrac{\partial \phi}{\partial x_1} \\ \dfrac{\partial \phi}{\partial x_2} \end{bmatrix}}_{\nabla_{\bar{x}}\phi} \tag{8.47}$$

where $\nabla_{\bar{u}}\phi$ is the gradient of ϕ with respect to the $\bar{u}$ variables, $\nabla_{\bar{x}}\phi$ is the gradient of ϕ with respect to the $\bar{x}$ variables, and $\underline{J}_{\bar{x}}$ is the Jacobian matrix for the transformation $\bar{x} : (-1,1)^2 \to \Omega$. Since the gradients in the integrands of (8.42) are taken with respect to variables $\bar{x}$ in Ω, we multiply by the inverse transpose of the Jacobian to obtain

$$\nabla_{\bar{x}}\phi = \underline{J}_{\bar{x}}^{-T} \nabla_{\bar{u}}\phi. \tag{8.48}$$

As a consequence, the integral

$$\int_\Omega \nabla_{\bar{x}}\psi^T \underline{\alpha} \nabla_{\bar{x}}\phi \, d\Omega \tag{8.49}$$

becomes

$$\int_\Omega \left[\underline{J}_{\bar{x}}^{-T}\nabla_{\bar{u}}\psi\right]^T \underline{\alpha} \left[\underline{J}_{\bar{x}}^{-T}\nabla_{\bar{u}}\phi\right] d\Omega = \int_\Omega \nabla_{\bar{u}}\psi^T \left[\underline{J}_{\bar{x}}^{-1}\underline{\alpha}\underline{J}_{\bar{x}}^{-T}\right] \nabla_{\bar{u}}\phi \, d\Omega. \tag{8.50}$$

Next, we change variables to the canonical domain and find that

$$\int_{(-1,1)^2} \nabla_{\bar{u}}\psi^T \left[\underline{J}_{\bar{x}}^{-1}\underline{\alpha}\underline{J}_{\bar{x}}^{-T}\right] \nabla_{\bar{u}}\phi \, |\det(\underline{J}_{\bar{x}})| \, d\Omega. \tag{8.51}$$

Since the absolute value of the determinant is a scalar function of $\bar{u}$, we can group it with the matrix in square brackets such that

$$\int_\Omega \nabla_{\bar{x}}\psi^T \underline{\alpha} \nabla_{\bar{x}}\phi \, d\Omega = \int_{(-1,1)^2} \nabla_{\bar{u}}\psi(\bar{u})^T \underbrace{\left[|\det(\underline{J}_{\bar{x}})| \, \underline{J}_{\bar{x}}^{-1}\underline{\alpha}(\bar{x}(\bar{u}))\,\underline{J}_{\bar{x}}^{-T}\right]}_{\underline{\alpha}_{\text{eff}}} \nabla_{\bar{u}}\phi(\bar{u}) \, d\Omega. \tag{8.52}$$

In other words, we can compute this integral on the canonical domain by replacing $\underline{\alpha}$ by a new effective coefficient matrix $\underline{\alpha}_{\text{eff}}$ as long as we choose to approximate ϕ by a linear combination of polynomial basis functions $\bar{N}(u_2) \otimes \bar{N}(u_1)$ in the canonical domain and substitute those same basis functions for ψ. Note that even though we approximate the solution ϕ using polynomials in the canonical domain, the transformation $\bar{x}(\bar{u})$ generally transforms these functions in ways that can be quite complicated. For example, they may not be polynomial in domain Ω.

The entries of the new $\underline{\alpha}_{\text{eff}}$ can be approximated by Legendre expansions, as before.

Since $\underline{\alpha}$ is symmetric, the new effective matrix is also symmetric. The explicit entries of the effective coefficient matrix can be obtained from the entries of the inverse Jacobian. Since $\underline{J}_{\bar{x}}$ is a two-by-two matrix, its inverse is

$$\underline{J}_{\bar{x}}^{-1} = \frac{1}{\det\left(\underline{J}_{\bar{x}}\right)} \begin{bmatrix} \dfrac{\partial x_2}{\partial u_2} & -\dfrac{\partial x_1}{\partial u_2} \\ -\dfrac{\partial x_2}{\partial u_1} & \dfrac{\partial x_1}{\partial u_1} \end{bmatrix} \tag{8.53}$$

and

$$\det\left(\underline{J}_{\bar{x}}\right) = \frac{\partial x_1}{\partial u_1}\frac{\partial x_2}{\partial u_2} - \frac{\partial x_1}{\partial u_2}\frac{\partial x_2}{\partial u_1}. \tag{8.54}$$

Note that these derivatives are all computable from the transfinite map (8.44). For reference, the derivatives are

$$\begin{aligned} \frac{\partial \bar{x}}{\partial u_1} = &-\frac{1}{2}\underline{\tilde{X}}_1\bar{p}\left(u_2\right) + \frac{1}{2}\underline{\tilde{X}}_2\bar{p}\left(u_2\right) + \frac{1-u_2}{2}\underline{\tilde{X}}_3\underline{\tilde{D}}\bar{p}\left(u_1\right) + \frac{1+u_2}{2}\underline{\tilde{X}}_4\underline{\tilde{D}}\bar{p}\left(u_1\right) \\ &+ \frac{1-u_2}{4}\underline{\tilde{X}}_1\bar{p}\left(-1\right) - \frac{1-u_2}{4}\underline{\tilde{X}}_2\bar{p}\left(-1\right) + \frac{1+u_2}{4}\underline{\tilde{X}}_1\bar{p}\left(1\right) - \frac{1+u_2}{4}\underline{\tilde{X}}_2\bar{p}\left(1\right) \end{aligned} \tag{8.55}$$

and

$$\begin{aligned} \frac{\partial \bar{x}}{\partial u_2} = &\frac{1-u_1}{2}\underline{\tilde{X}}_1\underline{\tilde{D}}\bar{p}\left(u_2\right) + \frac{1+u_1}{2}\underline{\tilde{X}}_2\underline{\tilde{D}}\bar{p}\left(u_2\right) - \frac{1}{2}\underline{\tilde{X}}_3\bar{p}\left(u_1\right) + \frac{1}{2}\underline{\tilde{X}}_4\bar{p}\left(u_1\right) \\ &+ \frac{1-u_1}{4}\underline{\tilde{X}}_1\bar{p}\left(-1\right) + \frac{1+u_1}{4}\underline{\tilde{X}}_2\bar{p}\left(-1\right) - \frac{1-u_1}{4}\underline{\tilde{X}}_1\bar{p}\left(1\right) - \frac{1+u_1}{4}\underline{\tilde{X}}_2\bar{p}\left(1\right) \end{aligned} \tag{8.56}$$

where $\underline{\tilde{D}}$ is the Legendre polynomial differentiation matrix from Chapter 2 with entries given by (2.95). Note that derivatives of the transfinite map are columns of the Jacobian matrix

$$\underline{J}_{\bar{x}} = \begin{bmatrix} \dfrac{\partial \bar{x}}{\partial u_1} & \dfrac{\partial \bar{x}}{\partial u_2} \end{bmatrix} \tag{8.57}$$

and contain all of the partial derivatives needed to specify the inverse Jacobian and its determinant. Computing the entries of the effective parameter matrix yields

$$\underline{\alpha}_{\text{eff}} = \left|\det\left(\underline{J}_{\bar{x}}\right)\right| \underline{J}_{\bar{x}}^{-1}\underline{\alpha}\underline{J}_{\bar{x}}^{-T} \tag{8.58}$$

$$= \frac{\left|\det\left(\underline{J}_{\bar{x}}\right)\right|}{\left[\det\left(\underline{J}_{\bar{x}}\right)\right]^2} \begin{bmatrix} \dfrac{\partial x_2}{\partial u_2} & -\dfrac{\partial x_1}{\partial u_2} \\ -\dfrac{\partial x_2}{\partial u_1} & \dfrac{\partial x_1}{\partial u_1} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix} \begin{bmatrix} \dfrac{\partial x_2}{\partial u_2} & -\dfrac{\partial x_2}{\partial u_1} \\ -\dfrac{\partial x_1}{\partial u_2} & \dfrac{\partial x_1}{\partial u_1} \end{bmatrix} \tag{8.59}$$

$$= \frac{\text{sign}\left[\det\left(\underline{J}_{\bar{x}}\right)\right]}{\det\left(\underline{J}_{\bar{x}}\right)} \begin{bmatrix} \alpha_{11,\text{eff}} & \alpha_{12,\text{eff}} \\ \alpha_{12,\text{eff}} & \alpha_{22,\text{eff}} \end{bmatrix} \tag{8.60}$$

where

$$\alpha_{11,\text{eff}} = \alpha_{11}\left(\frac{\partial x_2}{\partial u_2}\right)^2 - 2\alpha_{12}\frac{\partial x_1}{\partial u_2}\frac{\partial x_2}{\partial u_2} + \alpha_{22}\left(\frac{\partial x_1}{\partial u_2}\right)^2, \tag{8.61}$$

$$\alpha_{12,\text{eff}} = -\alpha_{11}\frac{\partial x_2}{\partial u_1}\frac{\partial x_2}{\partial u_2} + \alpha_{12}\left(\frac{\partial x_1}{\partial u_1}\frac{\partial x_2}{\partial u_2} + \frac{\partial x_1}{\partial u_2}\frac{\partial x_2}{\partial u_1}\right) - \alpha_{22}\frac{\partial x_1}{\partial u_1}\frac{\partial x_1}{\partial u_2}, \tag{8.62}$$

$$\alpha_{22,\text{eff}} = \alpha_{11}\left(\frac{\partial x_2}{\partial u_1}\right)^2 - 2\alpha_{12}\frac{\partial x_1}{\partial u_1}\frac{\partial x_2}{\partial u_1} + \alpha_{22}\left(\frac{\partial x_1}{\partial u_1}\right)^2. \tag{8.63}$$

Note that all entries of $\underline{\alpha}$ must be evaluated at $\bar{x}\left(\bar{u}\right)$ when evaluating this effective matrix. The partial derivatives themselves are already specified as functions of $\bar{u}$.

The remaining integrals to compute in (8.42) are less complicated. To compute

$$\int_\Omega \psi\beta\phi\, d\Omega \tag{8.64}$$

we evaluate

$$\int_{(-1,1)^2} \psi\left(\bar{u}\right)\beta\left(\bar{x}\left(\bar{u}\right)\right)\phi\left(\bar{u}\right)\left|\det\left(\underline{J}_{\bar{x}}\right)\right| d\Omega \tag{8.65}$$

which means that we can use

$$\beta_{\text{eff}} = \beta\left(\bar{x}\left(\bar{u}\right)\right)\left|\det\left(\underline{J}_{\bar{x}}\right)\right|. \tag{8.66}$$

Similarly, we replace f with

$$f_{\text{eff}} = f\left(\bar{x}\left(\bar{u}\right)\right)\left|\det\left(\underline{J}_{\bar{x}}\right)\right|. \tag{8.67}$$

For terms involving boundary integrals, the integrals transform differently (they are line integrals). In addition, the integrals transform differently depending on the particular edge. Note that each edge is described by a polynomial parametrization given by one of (8.18)-(8.21). The boundary integrals are all scalar line integrals, so they each transform as one of

$$\int_{\Gamma_i} \eta\, d\Omega = \int_{-1}^{1} \eta\left(\underline{\tilde{X}}_i\bar{p}\left(u_2\right)\right)\left\|\underline{\tilde{X}}_i\underline{\tilde{D}}\bar{p}\left(u_2\right)\right\|_2 du_2, \qquad i = 1,2, \tag{8.68}$$

$$\int_{\Gamma_i} \eta\, d\Omega = \int_{-1}^{1} \eta\left(\underline{\tilde{X}}_i\bar{p}\left(u_1\right)\right)\left\|\underline{\tilde{X}}_i\underline{\tilde{D}}\bar{p}\left(u_1\right)\right\|_2 du_1, \qquad i = 3,4, \tag{8.69}$$

where $\eta\left(\bar{x}\right)$ is a generic placeholder function. Thus, for Robin boundary integrals, the

coefficient γ becomes

$$\gamma_{\text{eff}}^{(i)} = \begin{cases} \gamma\left(\underline{\tilde{X}}_i \bar{p}\left(u_1\right)\right) \| \underline{\tilde{X}}_i \underline{\tilde{D}} \bar{p}\left(u_1\right) \|_2 & i = 1, 2, \\ \gamma\left(\underline{\tilde{X}}_i \bar{p}\left(u_2\right)\right) \| \underline{\tilde{X}}_i \underline{\tilde{D}} \bar{p}\left(u_2\right) \|_2 & i = 3, 4, \end{cases} \tag{8.70}$$

and q becomes

$$q_{\text{eff}}^{(i)} = \begin{cases} q\left(\underline{\tilde{X}}_i \bar{p}\left(u_1\right)\right) \| \underline{\tilde{X}}_i \underline{\tilde{D}} \bar{p}\left(u_1\right) \|_2 & i = 1, 2, \\ q\left(\underline{\tilde{X}}_i \bar{p}\left(u_2\right)\right) \| \underline{\tilde{X}}_i \underline{\tilde{D}} \bar{p}\left(u_2\right) \|_2 & i = 3, 4. \end{cases} \tag{8.71}$$

In principle, one should expect similar effects arising when imposing Dirichlet boundary conditions via the weak form

$$\int_{\Gamma_D} \psi_B \left(\phi - p\right) d\Omega = 0. \tag{8.72}$$

We note that p can be replaced by

$$p_{\text{eff}}^{(i)} = \begin{cases} p\left(\underline{\tilde{X}}_i \bar{p}\left(u_1\right)\right) & i = 1, 2, \\ p\left(\underline{\tilde{X}}_i \bar{p}\left(u_2\right)\right) & i = 3, 4, \end{cases} \tag{8.73}$$

in which the additional factor $\| \underline{\tilde{X}}_i \underline{\tilde{D}} \bar{p}\left(u_j\right) \|_2$ has been ignored. This is because along an edge, we can always choose the weight functions ψ_B to cancel this parametrization factor. Similarly, we can do the same for the term

$$-\int_{\Gamma_D} \psi \bar{n}^T \left(\underline{\alpha} \nabla \phi\right) d\Omega \tag{8.74}$$

by lumping the parametrization factor in with the normal derivative. We do this to save on additional computations. This modification of the weight function to cancel undesirable terms is similar to the choice of using $\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}(x_j)$ rather than $\bar{p}(x_j)$ as weight functions in the preceding chapter.

8.4 A Prototypical Curvilinear Example

To illustrate how the curvilinear map can be integrated into the PDE formulation of Chapter 7, we solve the simple constant coefficient Poisson equation

$$\nabla \cdot \nabla \phi = f \tag{8.75}$$

on a domain Ω whose four vertices are

$$\bar{x}_1 = \begin{bmatrix} \frac{1}{10} \\ 0 \end{bmatrix}, \qquad \bar{x}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{9}{10} \\ \frac{1}{5} \end{bmatrix}, \qquad \bar{x}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \bar{x}_4 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{8.76}$$

and whose interior is the region

$$\Omega = \left\{ \bar{x} \in \mathbb{R}^2 \, : \, \bar{n}_1^T \left(\bar{x} - \bar{x}_1 \right) < 0, \, \bar{n}_2^T \left(\bar{x} - \bar{x}_1 \right) < 0, \, \bar{n}_3^T \left(\bar{x} - \bar{x}_2 \right) < 0, \, \| \bar{x} \|_2 < 1 \right\} \tag{8.77}$$

where

$$\bar{n}_1^T \left(\bar{x}_3 - \bar{x}_1 \right) = 0, \tag{8.78}$$

$$\bar{n}_2^T \left(\bar{x}_2 - \bar{x}_1 \right) = 0, \tag{8.79}$$

$$\bar{n}_3^T \left(\bar{x}_4 - \bar{x}_2 \right) = 0. \tag{8.80}$$

This region has boundary components Γ_1, Γ_2, and Γ_3 defined by straight line segments and Γ_4 defined by a circular arc. We choose three of the sides to be straight line segments for this example because the types of meshes that we focus on in later chapters are constructed in such a way that typically only one edge of each element is curvilinear. That being said, the method is not restricted to such a case and can handle curvilinear boundaries on all edges. In addition, we have chosen $\underline{\alpha} = -\underline{I}$ and $\beta = 0$ so as to emphasize the effect of the map on the effective parameter $\underline{\alpha}_{\text{eff}}$ which is not constant with respect to $\bar{u}$ even though $\underline{\alpha}$ itself is. This way, we can comment on the reduction in sparsity of the operator matrix. In order to avoid Robin boundary conditions further obfuscating the effect of the map on sparsity, we set Dirichlet boundary conditions on each edge. We choose a manufactured solution

$$\phi_{\text{exact}} \left(\bar{x} \right) = \exp \left[- \left(\bar{x} - \bar{c} \right)^T \left(\bar{x} - \bar{c} \right) \right], \qquad \bar{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{8.81}$$

with boundary data $p = \phi_{\text{exact}}$ and forcing function $f = \nabla \cdot \nabla \phi_{\text{exact}}$.

For a tolerance $\epsilon_{\text{tol}} = 10^{-12}$, the Legendre expansions have rank and degree

$$\begin{aligned} K^{(\alpha_{11})} &= 9, & K_{x_1}^{(\alpha_{11})} &= 18, & K_{x_2}^{(\alpha_{11})} &= 12, \\ K^{(\alpha_{12})} &= 9, & K_{x_1}^{(\alpha_{12})} &= 18, & K_{x_2}^{(\alpha_{12})} &= 11, \\ K^{(\alpha_{22})} &= 10, & K_{x_1}^{(\alpha_{22})} &= 18, & K_{x_2}^{(\alpha_{22})} &= 12, \\ K^{(\beta)} &= 1, & K_{x_1}^{(\beta)} &= 0, & K_{x_2}^{(\beta)} &= 0, \end{aligned}$$

$$K^{(f)} = 11, \qquad K_{x_1}^{(f)} = 18, \qquad K_{x_2}^{(f)} = 14,$$
$$K_p^{(1)} = 13, \qquad K_p^{(2)} = 11, \qquad K_p^{(3)} = 10, \qquad K_p^{(4)} = 17,$$

where all ranks and degrees are for the effective parameters. Notice that to represent $\underline{\alpha}_{\text{eff}}$, we require high order polynomials to capture the effect of the map $\bar{x}(\bar{u})$ on $\underline{\alpha}$ even though $\underline{\alpha}$ was representable in terms of constants on Ω. In addition, to reproduce the curvilinear edge represented by the implicit function $\Phi(\bar{x}) = \|\bar{x}\|_2 - 1$, whose zero level set corresponds to Γ_4, we use the projection method described in Section 8.2. The edges are represented by degree $N_1 = 1$, $N_2 = 1$, $N_3 = 1$, and $N_4 = 8$ polynomials (straight edges only require linear polynomials and the circular arc can be represented to the desired tolerance by a degree eight polynomial).

Figure 8.1 illustrates the approximate solution to this problem corresponding to an $L_{u_1} = L_{u_2} = 18$ degree polynomial representation, along with the decay of coefficients used to represent the solution (plotted by computing the entrywise logarithm of absolute values $\log_{10}|\underline{\phi}|$). Figure 8.2 illustrates the pointwise error $\log_{10}|\phi - \phi_{\text{exact}}|$ (sampled with 201 uniformly spaced points in each coordinate of the canonical domain $\bar{u} \in (-1, 1)^2$) which shows that, in a worst case, the approximate solution is exact to 11 digits (reducing the tolerance yields more accurate solutions at greater cost). For this example, the condition number of the saddle point system is roughly $\kappa_1(\underline{S}) \approx 4,715$. Finally, Figure 8.2 also illustrates the sparsity of the saddle point matrix. Even though this problem is a constant coefficient Poisson problem, the curvilinear map introduces a significant number of nonzeros.

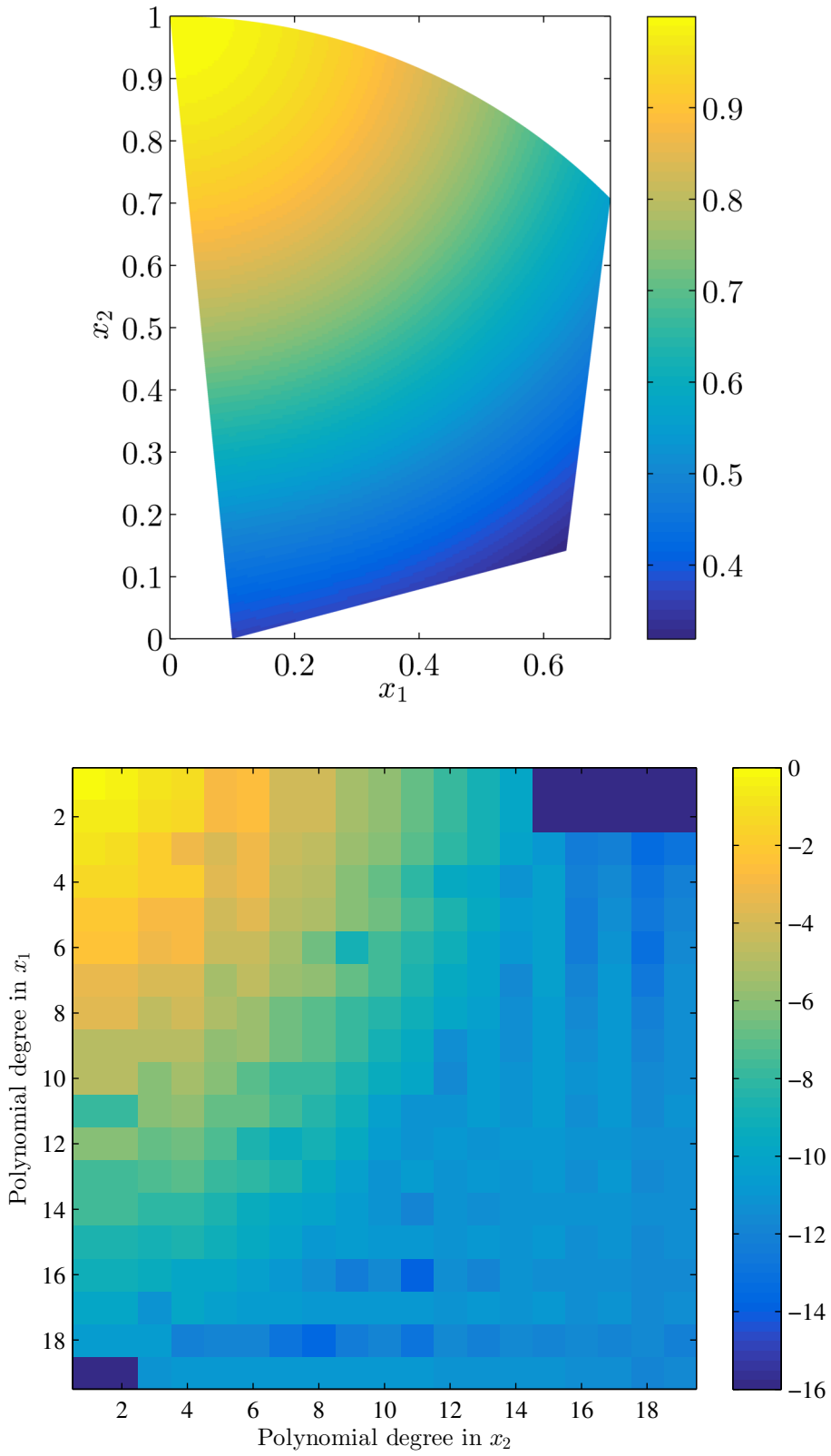

Figure 8.1: (top) Computed solution of the curvilinear problem (8.75) with Dirichlet boundary conditions. (bottom) Log base ten of the absolute value of the entries of the coefficient matrix $\underline{\phi}$.

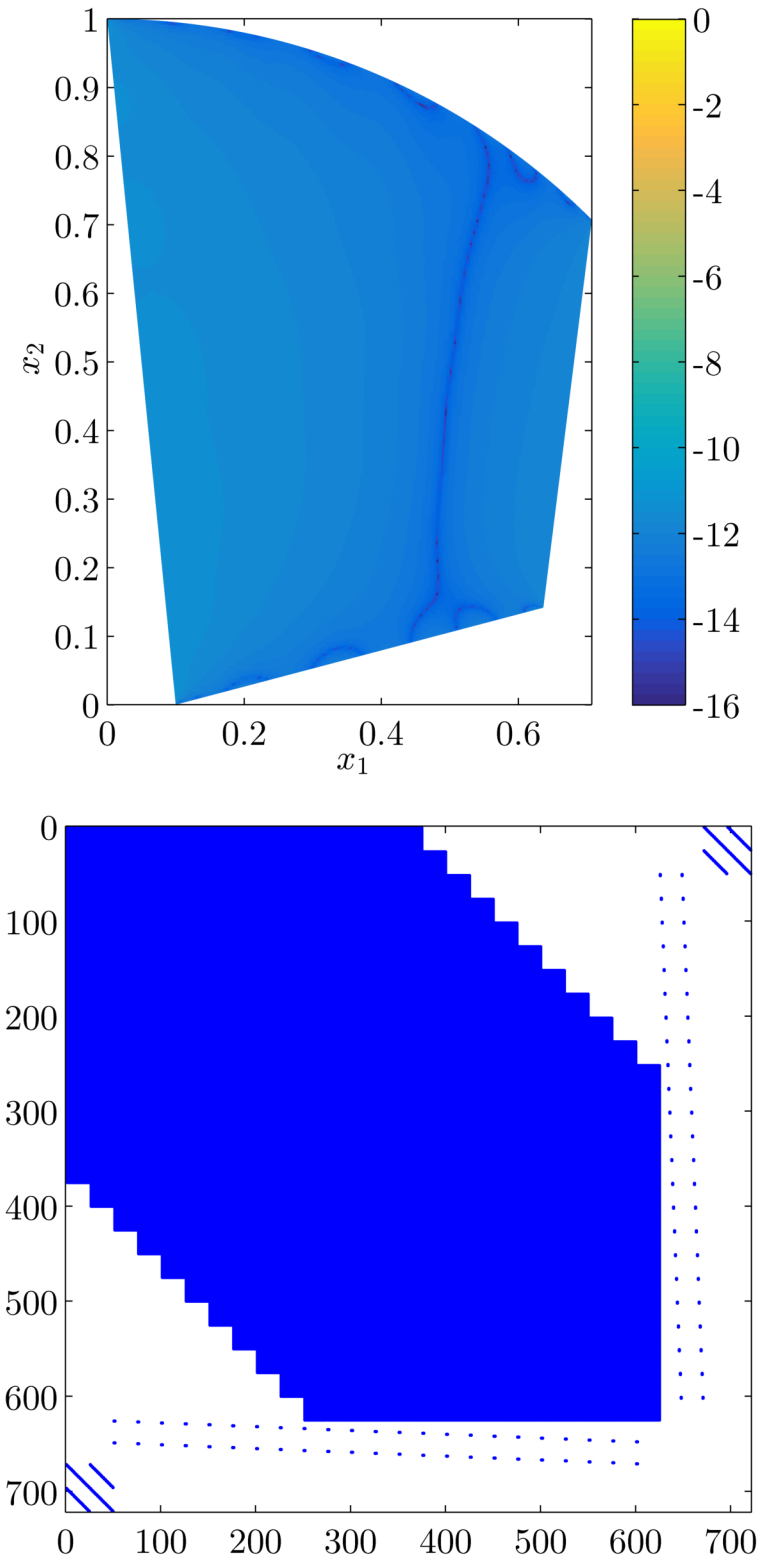

Figure 8.2: (top) Pointwise error $\log_{10}|\phi - \phi_{\text{exact}}|$ between the computed solution in Figure 8.1 and the exact solution. (bottom) Sparsity pattern of the saddle point system used to compute the solution.

Chapter 9

A Non-Conforming Finite Element Method

In Chapters 7 and 8, the PDE

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi = f \tag{9.1}$$

was solved on the canonical domain $(-1,1)^2$, and on mappings of this domain $\bar{x} : (-1,1)^2 \to \Omega$, subject to various boundary conditions. This chapter solves (9.1) on more general domains Ω that cannot be expressed by a single curvilinear quadrilateral map. The approach, as in the one-dimensional case, is to partition the domain Ω into a finite set of disjoint subdomains Ω_j (with a total of N_e such subdomains) so that $\Omega = \bigcup_{j=1}^{N_e} \Omega_j$ with $\Omega_j \cap \Omega_{j'} = \emptyset$ for all $j \neq j'$. Each subdomain Ω_j (called an element) is required to result from a map $\bar{x}_j : (-1,1)^2 \to \Omega_j$, each of which is of the transfinite interpolation type described in Chapter 8.

There are many ways one can construct a partition of a domain Ω into quadrilaterals. The literature [62, 156] cover a wide variety of methods, including structured and unstructured partitions (which are called meshes). According to [62], a structured mesh is one where "each vertex [...] can be readily defined as an array of indices" and, "by extension, any mesh having a high degree of ordering (for example a Cartesian grid) is said to be structured." This definition is ambiguous by design and allows for the possibility of irregularity in the mesh. In addition, it does not specify the geometry of individual elements. In this thesis, only a subset of structured meshes are considered. In particular, only Cartesian structured meshes (whose elements must belong to an orthogonal Cartesian grid) and near-Cartesian structured meshes with a large number of elements taken from an orthogonal Cartesian grid with possible exceptions occurring near the boundary of the domain Ω (and possibly at interfaces) are distinguished. Mesh generation algorithms to produce such meshes belong to

the class of superposition methods [157].

This chapter describes a particular superposition algorithm that can be used to mesh domain Ω. The method starts from a partition of Ω into subdomains $\hat{\Omega}_i$, each defined by the zero level set of a function Φ_i. A uniform Cartesian structured mesh is laid over Ω. Elements from this mesh sufficiently near interfaces between subdomains $\hat{\Omega}_i$ and sufficiently near the boundary of Ω are identified. The vertices of identified elements are projected to these interfaces. Then additional elements are added at the interfaces to improve the quality of the mesh in these regions [158]. A smoothing [66] and mesh optimization step [67, 68] follow to produce a conforming mesh that is near-Cartesian (it is structured away from boundaries and interfaces). As a final step, structured elements in the mesh are grouped together to form larger elements. Then edges of elements on curvilinear boundaries or interfaces are made curvilinear using the projection technique described in Chapter 8. This method produces a final non-conforming mesh.

The remainder of the chapter explains how to solve (9.1) on such a mesh. The techniques described in Chapters 7 and 8 can be applied to each element Ω_j of the mesh to produce local operator matrices. Since many elements come from a structured Cartesian grid, their local operator matrices tend to be sparse. Near interfaces the operator matrices tend not to be sparse, but these elements are needed to produce accurate solutions wherever parameters $\underline{\alpha}$ or β are discontinuous. In addition, local operator matrices tend not to be sparse near boundaries of the domain, however these elements are needed to accurately represent the geometry of the problem. Once local operator matrices are computed, the basis functions across element subdomains are connected together by imposing continuity constraints. Non-conforming edges in the mesh require special attention. The chapter describes a systematic approach to impose continuity along all edges of the mesh and emphasizes how to produce a global constraint matrix of full rank.

Finally, the chapter concludes with examples from electrostatics and electromagnetic scattering. The examples use *hp*-adaption to showcase the fact that basis functions on adjacent elements can have different polynomial degrees, and that the mesh can be non-conforming. In both types of problems, the method provides accurate solutions.

9.1 Quadrilateral Mesh Generation

To generate a near-Cartesian mesh, we employ a four step process. To specify this four step process, we require the user to provide: a tolerance $0 < \epsilon_{\text{tol}} \ll 1$ to which iterative computations are performed (unless otherwise specified); a set of fixed vertices $\{\bar{x}_{\text{fixed},k}\}$ which are usually corners in the geometry where a vertex in the mesh is required (this set

may be empty); a set of implicit functions Φ_i such that

$$\hat{\Omega}_i = \left\{ \bar{x} \in \mathbb{R}^2 \, : \, \Phi_i \left(\bar{x} \right) < 0 \right\}, \tag{9.2}$$

which specifies the partition of domain Ω; and the base point vertex $\bar{x}_{\text{bound}}$ and side length H of an axis-aligned bounding box Ω_{box} whose other vertices are $\bar{x}_{\text{bound}} + H\bar{e}_1$, $\bar{x}_{\text{bound}} + H\bar{e}_2$, and $\bar{x}_{\text{bound}} + H(\bar{e}_1 + \bar{e}_2)$. The bounding box should contain Ω (that is, $\Omega \subset \Omega_{\text{box}}$). The implicit functions Φ_i need not be signed distance functions, but we use them wherever possible[1]. The user should also specify how many steps of smoothing and mesh optimization to perform (usually some small number less than five), as well as the minimum and maximum level of refinement of the bounding box (denoted l_{min} and l_{max} respectively). We consider the box itself to be level 0 and each additional level corresponds to a uniform refinement of the previous level. Refinement is performed by dividing each square belonging to a given level into four similar squares (level 1 has four squares, level 2 has sixteen, etc., as in a quadtree). The mesh corresponding to the maximum level of refinement is used as a starting Cartesian structured mesh in the algorithm and the minimum level is used at the end of the algorithm when grouping structured elements into larger elements (elements are grouped only up to elements belonging to the minimum level). The choice of these parameters is crucial in constructing a useful mesh and a poor choice can lead to a poor mesh. For example, if the maximum level is not chosen large enough, the mesh may not properly resolve fine features in the geometry of the problem.

To illustrate the steps in the algorithm, we use an example with tolerance $\epsilon_{\text{tol}} = 10^{-12}$ and no fixed vertices. We specify

$$\Phi_1(\bar{x}) = \|\bar{x}\|_2 - 1, \tag{9.3}$$

$$\Phi_2(\bar{x}) = -\Phi_1(\bar{x}), \tag{9.4}$$

so that $\hat{\Omega}_1$ corresponds to the interior of the unit circle, and $\hat{\Omega}_2$ corresponds to the exterior. We choose bounding box parameters

$$\bar{x}_{\text{bound}} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \qquad H = 4, \tag{9.5}$$

so that $\Omega_{\text{box}} = (-2, 2)^2$. We set the number of smoothing and optimization steps to 3 and specify a maximum refinement level $l_{\text{max}} = 4$ and a minimum level $l_{\text{min}} = 0$. As we will see,

[1] For a calculus of signed distance functions that can be used to build more complicated domains from simple implicit functions, see [154].

the maximum level is chosen fine enough to resolve the unit circle adequately.

Step 1 Classify Elements and Project Vertices onto Interfaces

In the first step of the algorithm, we begin with the Cartesian mesh obtained from performing uniform refinements of the initial box Ω_{box}. The vertices in such a mesh are given by

$$\bar{x}_{m,n} = \bar{x}_{\text{bound}} + h(m\bar{e}_1 + n\bar{e}_2) \tag{9.6}$$

where $h = H \cdot 2^{-l_{\max}}$ and $m, n = 0, 1, 2, ..., 2^{l_{\max}}$. We view each set of points $\bar{x}_{m,n}$, $\bar{x}_{m,n+1}$, $\bar{x}_{m+1,n}$, and $\bar{x}_{m+1,n+1}$ as the vertices of a square element Ω_j in the mesh. We sample each implicit function Φ_i at all vertices. If an element has all four of its vertices satisfying $\Phi_i(\bar{x}) < 0$, then we classify this element as belonging to subdomain $\hat{\Omega}_i$. Those elements that lie entirely outside of Ω are discarded in the process (these are elements for which $\Phi_i(\bar{x}) \geq 0$ for all i).

After such a process, there will be small bands of unclassified elements intersecting the boundaries of subdomains $\hat{\Omega}_i$. To classify those elements, we perform a more careful check by computing an approximate area fraction of the element that lies inside each subdomain. If $|\Omega_j|$ is the area of Ω_j, then the area fraction of Ω_j contained in $\hat{\Omega}_i$ is

$$A_{ij}^{(f)} = \frac{1}{|\Omega_j|} \int_{\Omega_j} \mathbb{1}_{\hat{\Omega}_i} d\Omega \tag{9.7}$$

$$= \frac{1}{|\Omega_j|} \int_{\Omega_j} [1 - u(\Phi_i(\bar{x}))] d\Omega \tag{9.8}$$

where $\mathbb{1}_{\hat{\Omega}_i}$ is the unit indicator function on $\hat{\Omega}_i$ which we represent using the Heaviside step function $u(t)$. In practice, we use a first order approximation to the Heaviside step function [154] given by

$$u(t) \approx \begin{cases} 0 & t < -\epsilon \\ \frac{1}{2} + \frac{t}{2\epsilon} + \frac{1}{2\pi} \sin\left(\frac{\pi t}{\epsilon}\right) & -\epsilon \leq t \leq \epsilon \\ 1 & \epsilon < t. \end{cases} \tag{9.9}$$

We choose $\epsilon = 1.5h$ and compute integral (9.8) numerically using Gauss-Legendre quadrature with six nodes along each dimension. Since

$$A_{0j}^{(f)} + \sum_i A_{ij}^{(f)} = 1 \tag{9.10}$$

where $A_{0j}^{(f)}$ corresponds to the area fraction outside of Ω, once we have computed all $A_{ij}^{(f)}$ for $i > 0$, we also know $A_{0j}^{(f)}$. We then classify the boundary elements by largest area fraction.

Note that these classification checks are a potential weak point of the algorithm. For example, it is possible for an element to satisfy the four vertex conditions but still intersect the boundary. One potentially costly fix is to sample more points along edges of each element. Instead, we choose this inexpensive test and require the user to choose $l_{\max}$ to be large enough, and consequently the spacing h to be small enough, to capture the finest features of the boundaries of $\hat{\Omega}_i$. In particular, one should choose h as a function of the curvature of the boundary. Regions of high curvature require smaller h. This can sometimes have negative effects because the resulting mesh is uniform. Thus, if the boundary has only a few regions of large curvature, but otherwise mostly small curvature, then the mesh must be fine to resolve those regions of high curvature, and will be over-refined in regions of low curvature. At the end of the algorithm, we show how to avoid these effects in the interior of subdomains. However, this still causes the boundary to be refined to the same level everywhere, even where this level of refinement may not be needed.

The classification of elements is used to determine which vertices of the Cartesian mesh to project onto the boundaries of $\hat{\Omega}_i$. A vertex sharing four elements of the same classification is considered fixed while others are projected. The projection is computed using the same approach as in Chapter 8. That is, if a free vertex belongs to subdomain $\hat{\Omega}_i$ (we have already checked which subdomain each vertex belongs to), then we use Φ_i in iteration (8.25). If we do not know the gradient of Φ_i, we approximate its gradient using (8.38). If the norm of the gradient is zero at some point during the iteration, we perturb the unit vector in the gradient computation by a random direction (each component uniformly distributed on the interval $(0, 1)$). This is acceptable because we simply want to project the vertex onto the boundary of $\hat{\Omega}_i$ and are not concerned with whether the vertex is projected to the nearest point on the boundary. We terminate the iteration if the distance between the projected vertex and the boundary falls below the user specified tolerance ϵ_{tol} or if a number of iterations exceeding 20 is reached. Typically the number of iterations required for convergence is much smaller[2]. We then compute the distance of the projected vertices to the user specified set of fixed points

[2]As a safety precaution, if the iteration terminates having reached 20 iterations without converging, we use bisection to compute the projection onto the boundary. This is slower, but is more robust, particularly for functions Φ_i that are not differentiable. Starting with vertex $\bar{x}$, we compute the normalized direction $\bar{d} = \nabla\Phi_i(\bar{x})/\|\nabla\Phi_i(\bar{x})\|_2$ and solve for the scalar τ such that $\Phi_i(\bar{x} + \tau\bar{d}) = 0$. This is a one-dimensional root-finding problem. We search for the root $\tau \in (-\sqrt{2}h, \sqrt{2}h)$.

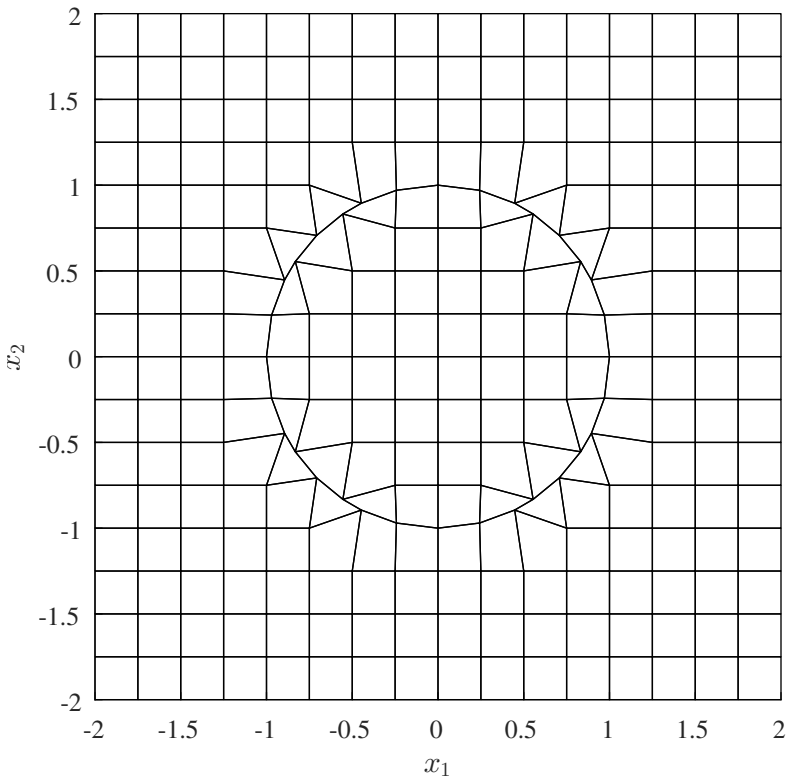

Figure 9.1: Structured Cartesian mesh after projection of vertices onto interfaces.

$\{\bar{x}_{\text{fixed},k}\}$. We set the closest projected vertex to its corresponding fixed point to ensure that all points in the fixed set belong to the set of vertices in the mesh.

Figure 9.1 illustrates the result of this first step of the algorithm as applied to the unit circle example. Note that elements away from the interface remain structured but vertices near the interface have been projected onto the unit circle. All elements are quadrilateral, but many of the elements that include projected vertices are degenerate because of the projection phase.

Step 2 Add Pillow Layer Elements

Next, new elements are added to the mesh along all boundaries of subdomains $\hat{\Omega}_i$. These elements are added to the mesh in layers by a process called pillowing [158]. Every vertex $\bar{x}$ on the boundary of $\hat{\Omega}_i$ is duplicated, then projected a distance $\sqrt{2}h/4$ away from the boundary (one fourth of the length of the diagonal of a structured element in the mesh) and into $\hat{\Omega}_i$ using

$$\bar{x}_{\text{new}} = \bar{x} - \frac{\sqrt{2}}{4} h \frac{\nabla \Phi_i(\bar{x})}{\|\nabla \Phi_i(\bar{x})\|_2}. \tag{9.11}$$

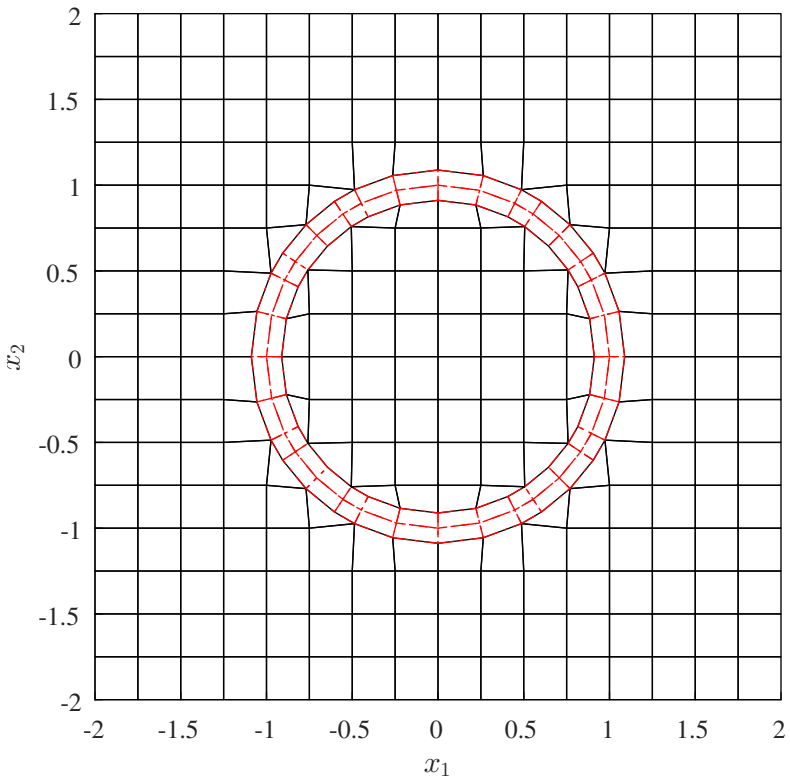

Figure 9.2: Addition of pillow layers. Original elements from the structured Cartesian mesh are shown with black edges while new elements in the pillow layers are shown with red dashed edges.

It is important to replace $\bar{x}$ with $\bar{x}_{\text{new}}$ for all elements belonging to $\hat{\Omega}_i$. This process is repeated for every $\hat{\Omega}_i$, resulting in a layer of new elements on boundaries of Ω and double layers of elements on interfaces. We make sure to classify the new elements appropriately according to the same classification scheme as in Step 1.

Figure 9.2 shows the mesh for the unit circle example after the pillow layer step. Note that since the unit circle is an interface, there are two layers of elements added. By virtue of using the same points on the shared interface, the pillow layer elements are conforming across the interface even though the original projected mesh was not. Elements from the original mesh connecting to the pillow layer continue to be degenerate.

The connectivity of new elements and of the original mesh should be determined. We generate four arrays that do this. The first array $\underline{X}$ lists the locations of the N_v vertices in the mesh as an $\mathbb{R}^{2\times N_v}$ matrix. The second array $\underline{V}$ lists which four vertices belong to each of the N_e elements. To do this, we use an $\mathbb{N}^{N_e\times 4}$ array. The row number in this array implicitly refers to the element while the column number implicitly refers to the local vertex number as illustrated in Figure 9.3. The entry $\underline{V}_{i,j}$ corresponding

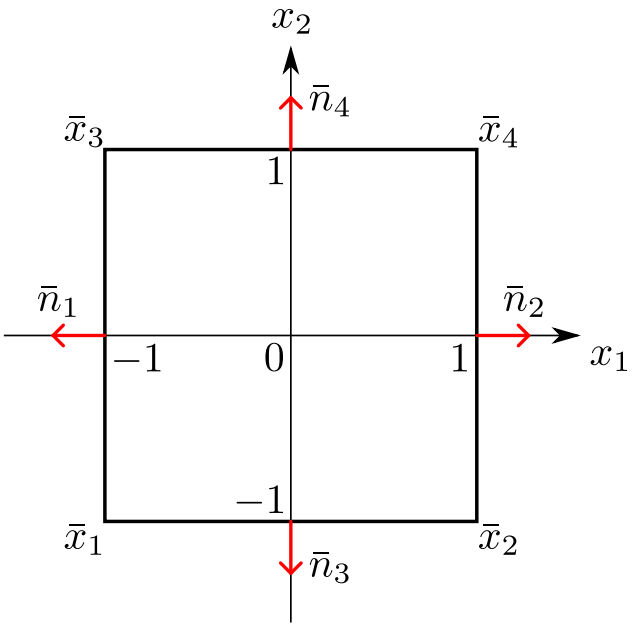

Figure 9.3: Local canonical element vertex numbering and edge numbering. Vertices have coordinates $\bar{x}_1 = -\bar{e}_1 - \bar{e}_2$, $\bar{x}_2 = \bar{e}_1 - \bar{e}_2$, $\bar{x}_3 = -\bar{e}_1 + \bar{e}_2$, and $\bar{x}_4 = \bar{e}_1 + \bar{e}_2$ and edges have unit normals $\bar{n}_1 = -\bar{e}_1$, $\bar{n}_2 = \bar{e}_1$, $\bar{n}_3 = -\bar{e}_2$, and $\bar{n}_4 = \bar{e}_2$.

to local vertex j of the ith element is an integer k that refers to the coordinates of the kth global vertex stored in column k of $\underline{X}$. We produce another $\mathbb{N}^{N_e \times 4}$ array $\underline{E}$ that stores the neighbors of each element across each edge. The row number in this array implicitly refers to the element while the column number implicitly refers to the local edge number as illustrated in Figure 9.3. Here, the entry $\underline{E}_{i,j}$ corresponding to local edge j of the ith element is an integer k that refers to element k that shares that edge. If no element shares edge $\underline{E}_{i,j}$, we store a zero instead. Similarly, we also store an $\mathbb{N}^{N_e \times 4}$ array $\underline{E}^{(n)}$ which specifies the local number of the shared edge. The row number and column number implicitly encode the same data as in $\underline{E}$ but the entry $\underline{E}^{(n)}_{i,j}$ specifies the local edge number of the neighboring edge (an integer between 1 and 4). If the edge is not shared, we store a zero instead.

Step 3 Perform Smoothing and Local Mesh Optimization

To improve the quality of elements near the boundary, we begin by applying Laplacian smoothing [66]. That is, for every vertex $\bar{x}_m$ adjacent to or on a boundary, we compute its new location

$$\bar{x}_{\text{new}} = \sum_{n \in \mathcal{N}_m} \bar{x}_n \tag{9.12}$$

where $\mathcal{N}_m$ is the set of indices corresponding to vertices $\bar{x}_n$ sharing an edge connecting to vertex $\bar{x}_m$. We do this for every relevant vertex, then project those vertices that be-

longed to boundaries and interfaces back onto their respective interfaces or boundaries using the projection scheme of Step 1. Vertices that were part of the user defined set of fixed nodes are not changed.

A small number of iterations of Laplacian smoothing is effective for meshes of a single convex domain Ω but can result in meshes with inverted elements otherwise (for example, when meshing a re-entrant corner). An inverted element is one whose Jacobian determinant (corresponding to the transfinite interpolation map) changes sign. To return the mesh to a state with no inverted elements, we use a mesh untangling algorithm [67]. Typically, only a small number of vertices result in inverted elements. To identify these vertices, we visit every element with a vertex that was projected in Step 1, along with all pillow layer elements. For each element, we compute four signed areas corresponding to triangles in the quadrilateral. Triangle 1 has local vertices 1, 2, and 3; triangle 2 has local vertices 2, 4, and 1; triangle 3 has local vertices 3, 1, and 4; and triangle 4 has local vertices 4, 3, and 2 (these are right triangles corresponding to each vertex in the local canonical element in Figure 9.3 with vertices ordered counterclockwise). If a triangle has vertex coordinates $\bar{p}_1$, $\bar{p}_2$, and $\bar{p}_3$ numbered counterclockwise, then its edges are given by vectors $\bar{p}_2 - \bar{p}_1$ and $\bar{p}_3 - \bar{p}_1$. In addition, its signed area is

$$A^{(t)} = \frac{1}{2}\det(\underline{T}), \qquad \underline{T} = \begin{bmatrix} \bar{p}_2 - \bar{p}_1 & \bar{p}_3 - \bar{p}_1 \end{bmatrix}. \tag{9.13}$$

We compute the four signed areas of the four triangles defined for a single quadrilateral element. If any of the signed areas is negative, the element is possibly inverted.

The untangling algorithm is performed for every vertex belonging to possibly inverted elements, except those vertices belonging to a boundary or to the user specified set of fixed nodes. Once a vertex $\bar{x}$ is chosen, we perform the untangling algorithm on the set of elements that share this vertex. In practice, this is a small number of elements. The algorithm begins by solving an optimization problem to find a location $\bar{x}_{\text{untangled}}$ to move the vertex $\bar{x}$ to in order to make all signed areas of the triangles corresponding to the adjacent elements positive. Suppose elements $j \in \mathcal{J}$ share vertex $\bar{x}$. Then we solve

$$\bar{x}_{\text{untangled}} = \arg\min_{\bar{x}} \sum_{\substack{j\in\mathcal{J} \\ i=1,2,3,4}} |A_{ij}^{(t)} - \beta A| - (A_{ij}^{(t)} - \beta A) \tag{9.14}$$

where $A_{ij}^{(t)}$ is the area of the ith triangle corresponding to the jth element (computed according to (9.13)), $\beta = \epsilon_{\text{tol}}^{1/4}$ is a tolerance that we choose to avoid finding a new vertex position $\bar{x}_{\text{untangled}}$ that leaves certain areas $A_{ij}^{(t)}$ equal to zero, and A is the average area of the elements in $\mathcal{J}$. We choose β this way because we terminate the

iterative optimization routine used to solve this problem when a tolerance $\epsilon_{\text{tol}}^{1/2}$ is met ($\bar{x}$ is used as the initial iterate). Evidently, some $A_{ij}^{(t)}$ and A depend on the location of vertex $\bar{x}$. This objective function is inspired by the objective

$$\sum_{\substack{j\in\mathcal{J}\\ i=1,2,3,4}} |A_{ij}^{(t)}| - A_{ij}^{(t)} \tag{9.15}$$

which is zero when all triangles have positive signed area and which is greater than zero when any signed area is negative. Unfortunately, if some signed areas are zero while others are positive, this simpler objective is minimized, but can correspond to a mesh with degenerate elements. To avoid this situation, we add the term βA to obtain (9.14). In practice, we compute the average area A using

$$A = \frac{1}{2|\mathcal{J}|} \sum_{\substack{j\in\mathcal{J}\\ i=1,2,3,4}} A_{ij}^{(t)} \tag{9.16}$$

where $|\mathcal{J}|$ is the total number of elements sharing vertex $\bar{x}$. We divide by 2 because performing $\sum_{i=1}^{4} A_{ij}^{(t)}$ for a given quadrilateral j double counts its area. We solve this optimization problem using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm with backtracking line search and gradients approximated by finite differences [90].

If the untangling algorithm succeeds, we use a second mesh optimization algorithm to improve the quality of the elements that were untangled. This optimization method belongs to the target-matrix paradigm of mesh optimization methods [68]. The method works when all elements are untangled. We solve the optimization problem

$$\bar{x}_{\text{optimized}} = \arg\min_{\bar{x}} \sum_{\substack{j\in\mathcal{J}\\ i=1,2,3,4}} \mu(\underline{T}_{ij}) \tag{9.17}$$

with initial iterate $\bar{x}_{\text{untangled}}$. The matrices $\underline{T}_{ij}$ are those that arise when computing the area of triangle i for element j as in (9.13) and, as a consequence, are functions of location $\bar{x}$. We choose the shape metric

$$\mu(\underline{T}) = \begin{cases} \text{cond}_F(\underline{T}) & \det(\underline{T}) > 0 \\ \infty & \text{otherwise} \end{cases} \tag{9.18}$$

where $\text{cond}_F(\underline{T}) = \|\underline{T}\|_F \|\underline{T}^{-1}\|_F$ is the condition number in the Frobenius norm. This tends to produce elements of roughly the same shape (in terms of angles and aspect

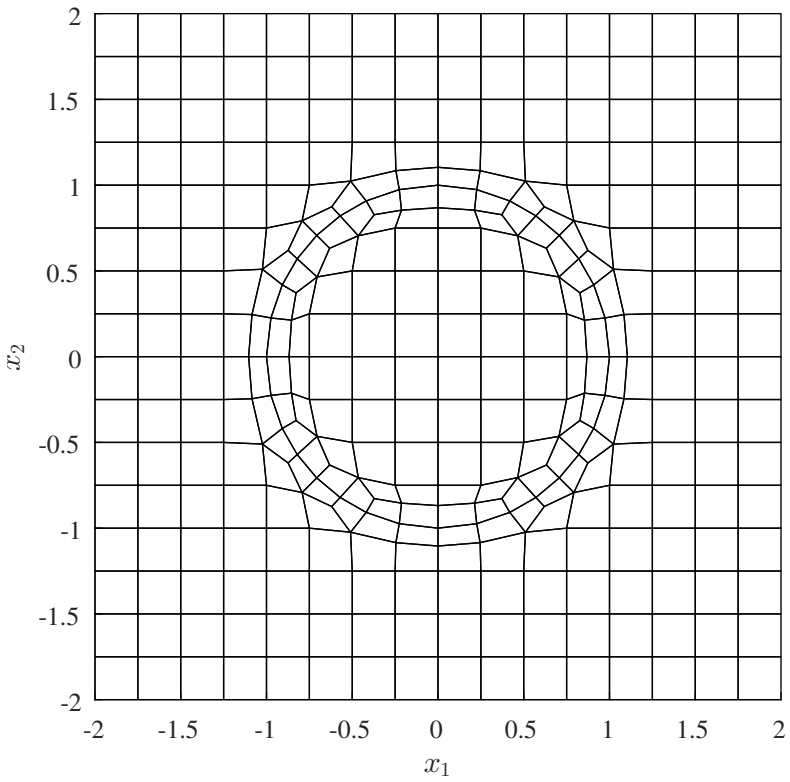

Figure 9.4: Mesh after three iterations of Laplacian smoothing followed by mesh untangling and mesh optimization.

ratio) and disregards size and orientation properties. The optimization is performed using the same tolerance and BFGS method as for the untangling method.

In practice, we perform a small number of iterations (less than or equal to five) of Laplacian smoothing followed by the mesh optimization procedure (here one application of Laplacian smoothing (9.12) followed by one optimization phase is considered one iteration). Figure 9.4 illustrates the mesh for the unit circle example after this smoothing and optimization step. Element quality near the interface has improved when compared with Figure 9.2.

Step 4 Coarsen the Mesh and Construct Curvilinear Edges

Next, we coarsen the mesh by grouping elements that belong to the structured part of the mesh. We only group elements according to a quadtree structure. That is, if we are grouping elements on level l of the tree, we want the group to be an element on level $l-1$ of the tree. For example, it would be possible to group the first two elements of the bottom two rows of Figure 9.4 but not the second and third elements on the bottom two rows. In addition to this grouping rule, we also make sure to only group elements that all belong to the same classification. This prevents grouping elements

across interfaces. We repeat the grouping phase for each level in the tree, starting from level $l_{\max}$ and backtracking to level $l_{\min}$. We do not impose a 2:1 rule common in most quadtree meshes for finite element methods [62]. We maintain a tree structure which indicates which elements in the tree are leaves (i.e. elements) in the coarsened mesh.

Finally, we generate curvilinear edges for all elements that have an edge on a boundary or interface (these are always elements on the finest level of the tree $l_{\max}$). We compute a curvilinear edge for any edge shared with the boundary of Ω (corresponding to a boundary edge) or for any edge shared with a neighbor of different classification (corresponding to an interface edge). We use the connectivity array $\underline{E}$ to determine boundary edges (a zero entry) and neighbors (a nonzero entry). Curvilinear edges are constructed using the projection-based approach for implicitly defined boundaries as described in Section 8.2. For each edge, the vertices serve as the points $\bar{x}_{\text{start}}$ and $\bar{x}_{\text{end}}$ for the curvilinear projection scheme (we always specify the vertices in increasing local numbering order). We store the matrix $\underline{\tilde{X}}_i^{(j)}$ of Legendre coefficients for element j and edge i. For each straight edge in the mesh,

$$\underline{\tilde{X}}_i^{(j)} = \begin{bmatrix} \bar{x}_{\text{start}} & \bar{x}_{\text{end}} \end{bmatrix} \underline{\tilde{V}}^{-T} \tag{9.19}$$

where $\underline{\tilde{V}}$ is the degree one generalized Vandermonde matrix of Legendre polynomials evaluated at points -1 and 1, as given in (2.32). With this information, each element has a well defined transfinite interpolation map $\bar{x}_j : (-1,1)^2 \to \Omega_j$.

Figure 9.5 illustrates the final mesh for the unit circle example after the coarsening and curvilinear edge construction step. Notice how elements away from the interface are grouped together. Here, elements on level $l_{\max} = 4$ have been grouped and exist on levels $l = 2, 3$. No elements exist on levels $l = 0, 1$, despite $l_{\min}$ being set to zero because there is no opportunity for such coarsening (the interface prevents this possibility). The elements in the corners of Figure 9.5 show that the 2:1 rule has not been adhered to. In addition, while it may be difficult to see, upon close inspection, the edges of elements shared with the interface are curvilinear whereas they were not in any of the previous steps of the algorithm.

This four step procedure results in a mesh for which, typically, a large number of elements are structured. In addition, because all steps were performed to a specific user specified tolerance, the mesh represents the desired geometry to this tolerance. This is crucial when computing high accuracy solutions because errors in the representation of the geometry can have a significant effect on accuracy. In the event that a conforming mesh is desired, we simply leave out the coarsening portion of Step 4.

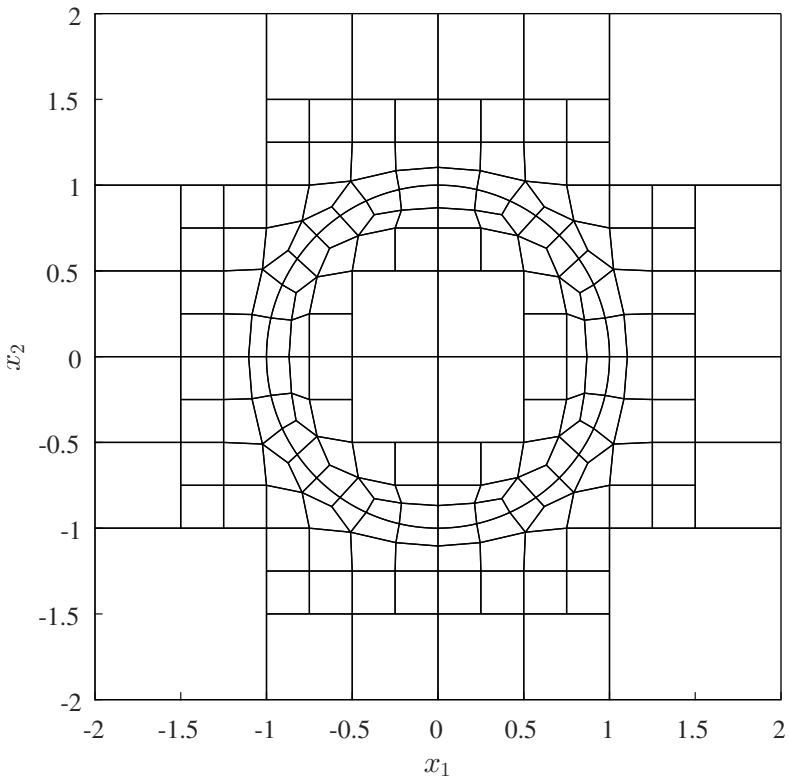

Figure 9.5: Final mesh after quadtree coarsening and computation of curvilinear edges.

9.2 A Conforming Finite Element Method

Before addressing the full generality of the finite element method on a non-conforming mesh, it is helpful to first see how to treat the method on a conforming mesh. This will be a special case of the method presented later for non-conforming meshes. By separating the presentation, we can identify which aspects differ when adding non-conformity to the mesh. In order to solve (9.1) on a potentially complicated domain Ω, we begin by computing a conforming near-Cartesian structured quadrilateral mesh, as in Section 9.1 (without the coarsening procedure), resulting in arrays $\underline{X}$, $\underline{V}$, $\underline{E}$, and $\underline{E}^{(n)}$ which describe its connectivity, along with matrices $\underline{\tilde{X}}_i^{(j)}$ used to describe the geometry of the edges of each element. From the connectivity data, we produce a linked list with N_v nodes. Each node corresponds to a vertex in the mesh and points to an array containing all edges that are incident to the vertex (the edges are labeled by the element to which they belong, as well as their local edge number). We call this the vertex-to-edge incidence list and will use it to ensure that the constraint matrix we assemble remains full rank (this is similar to the vertex-to-edge incidence list in Chapter 7 but contains more data now that the mesh is comprised of more than one element).

We assign each element subdomain Ω_j its appropriate parameters $\underline{\alpha}$, β, and f. We do

this for boundary functions γ, q, and p as well, but only for edges in the connectivity array $\underline{E}$ that contain a zero entry (these are the edges that belong to the boundary of domain Ω). As a final preparatory step, we compute Legendre expansions for each element's effective parameters $\underline{\alpha}_{\text{eff}}$, β_{eff}, f_{eff}, as well as for boundary functions γ_{eff}, q_{eff}, p_{eff}. We can do this because each element has its transfinite interpolation map $\bar{x}_j : (-1,1)^2 \to \Omega_j$ defined at the conclusion of the mesh generation phase.

Using the maps $\bar{x}_j$, we define effective sets of basis functions

$$\bar{N}_j(\bar{x}) = \begin{cases} \bar{N}\left(u_{j,2}(\bar{x}_j)\right) \otimes \bar{N}\left(u_{j,1}(\bar{x}_j)\right) & \bar{x} \in \Omega_j \\ 0 & \text{otherwise} \end{cases} \tag{9.20}$$

where $\bar{u}_j : \Omega_j \to (-1,1)^2$ is the inverse map of $\bar{x}_j$. Each set of basis functions is nonzero on its corresponding subdomain Ω_j and consists of transformed polynomials of degree L_{j,u_1} and L_{j,u_2}. In practice, the inverse of each transfinite interpolation map is not explicitly known and therefore we only work with the forward map[3]. If we concatenate all basis functions together and write ϕ as a linear combination with coefficients $\bar{\phi}$ such that

$$\bar{N}(\bar{x}) = \begin{bmatrix} \bar{N}_1(\bar{x}) \\ \bar{N}_2(\bar{x}) \\ \vdots \\ \bar{N}_{N_e}(\bar{x}) \end{bmatrix}, \qquad \bar{\phi} = \begin{bmatrix} \bar{\phi}_1 \\ \bar{\phi}_2 \\ \vdots \\ \bar{\phi}_{N_e} \end{bmatrix}, \tag{9.21}$$

and then apply, for example, Galerkin's method, the operator matrix partitions into a block diagonal matrix

$$\underline{A} = \begin{bmatrix} \underline{A}_1 & & & \\ & \underline{A}_2 & & \\ & & \ddots & \\ & & & \underline{A}_{N_e} \end{bmatrix} \tag{9.22}$$

[3]Occasionally the inverse map is needed to evaluate a basis function at a specified point $\bar{x}_{\text{desired}}$. In such a case, we must apply root-finding techniques to the function $\bar{f}_j(\bar{u}) = \bar{x}_{\text{desired}} - \bar{x}_j(\bar{u})$ to find the corresponding $\bar{u}_{\text{desired}}$ that yields $\bar{f}_j(\bar{u}_{\text{desired}}) = 0$. Because the transfinite interpolation is a vector function of polynomials, there are root-finding possibilities beyond the standard iterative fixed point techniques (which subsume Newton's method) [50]. However, all of the computations in this chapter are performed without solving this nonlinear problem.

and the forcing term partitions into

$$\bar{b} = \begin{bmatrix} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_{N_e} \end{bmatrix} \tag{9.23}$$

where each block corresponds to a local operator matrix and local forcing term assembled as described in Chapters 7 and 8. To see why this block structure arises, we evaluate the weighted residual

$$\int_\Omega \psi \left[-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi\right] d\Omega = \int_\Omega \psi f \, d\Omega. \tag{9.24}$$

Computing the integral split over each subdomain yields

$$\sum_{j=1}^{N_e} \int_{\Omega_j} \psi \left[-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi\right] d\Omega = \sum_{j=1}^{N_e} \int_{\Omega_j} \psi f \, d\Omega \tag{9.25}$$

$$\sum_{j=1}^{N_e} \int_{\Omega_j} \nabla \psi^T \underline{\alpha} \nabla \phi + \psi \beta \phi \, d\Omega - \oint_{\partial \Omega_j} \psi \bar{n}^T \underline{\alpha} \nabla \phi \, d\Omega = \sum_{j=1}^{N_e} \int_{\Omega_j} \psi f \, d\Omega. \tag{9.26}$$

Applying Robin boundary conditions gives

$$\sum_{j=1}^{N_e} \underbrace{\int_{\Omega_j} \nabla \psi^T \underline{\alpha} \nabla \phi + \psi \beta \phi \, d\Omega + \int_{\Gamma_{R,j}} \psi \gamma \phi \, d\Omega}_{\text{terms associated with } \underline{A}_j} - \int_{\partial \Omega_j \setminus \Gamma_{R,j}} \psi \bar{n}^T \underline{\alpha} \nabla \phi \, d\Omega =$$

$$\sum_{j=1}^{N_e} \underbrace{\int_{\Omega_j} \psi f \, d\Omega + \int_{\Gamma_{R,j}} \psi q \, d\Omega}_{\text{terms associated with } \bar{b}_j}. \tag{9.27}$$

Substituting $\phi = \bar{N}(\bar{x})^T \bar{\phi}$ with $\bar{N}(\bar{x})$ and $\bar{\phi}$ as in (9.21) and repeating the equation with ψ replaced by every entry in $\bar{N}(\bar{x})$ yields

$$\underline{A} \bar{\phi} + \underline{C}^T \bar{\nu} = \bar{b}. \tag{9.28}$$

All off-diagonal blocks of $\underline{A}$ are zero because none of the subdomains Ω_j intersect each other and each set of basis functions in $\bar{N}(\bar{x})$ is only nonzero on a single subdomain. As we have seen in Chapter 6, the remaining unassigned boundary terms in (9.27) contribute to $\underline{C}^T \bar{\nu}$.

To construct the constraint matrix $\underline{C}$, we explicitly enforce Dirichlet boundary conditions, as well as inter-element continuity. To do so, we consider all $j = 1, 2, ..., N_e$, elements

sequentially, and for each element consider each edge $i = 1, 2, 3, 4$, in turn. If edge i of element j belongs to Γ_D, we append the set of equations

$$\begin{bmatrix} 0 & \cdots & 0 & \underline{C}_i & 0 & \cdots & 0 \end{bmatrix} \bar{\phi} = \bar{d}_i \tag{9.29}$$

to the matrix $\underline{C}$ and vector $\bar{d}$. Here, the nonzero $\underline{C}_i$ block belongs to the jth block column, and $\underline{C}_i$ and $\bar{d}_i$ are assembled as in Chapters 7 and 8. To ensure that we have not introduced any redundant equations in performing this operation, we remove this edge from all vertex lists in the vertex-to-edge incidence list. As in Section 7.5, we remove the first row of (9.29) if one vertex list in the incidence list becomes empty. If two vertex lists become empty, we remove the first two rows of (9.29). This preserves the full row rank of the constraint matrix $\underline{C}$.

Enforcing inter-element continuity is related but slightly more complicated. As in Chapter 6, continuity between elements sharing boundary Γ_i can be enforced in a weak sense via

$$\int_{\Gamma_i} \bar{N}_B(\bar{x}) \left[\phi_j(\bar{x}) - \phi_{j'}(\bar{x})\right] d\Omega = 0 \tag{9.30}$$

where ϕ_j is used to signify the solution on element j and $\phi_{j'}$ the solution on its neighboring element j' that shares common boundary Γ_i, with i denoting the ith local boundary for element j (element j' has a corresponding local element edge number i'). $\bar{N}_B$ is a vector of weight functions along edge Γ_i that we specify. Given this edge labeling, a convenient way to write the boundary integral is

$$\int_{\Gamma_i} \bar{N}_B(\bar{x}) \phi_j(\bar{x}) \, d\Omega - \int_{\Gamma_{i'}} \bar{N}_B(\bar{x}) \phi_{j'}(\bar{x}) \, d\Omega = 0. \tag{9.31}$$

Without loss of generality, we let element j have degree L_{j,u_1} and L_{j,u_2} polynomial basis functions whereas we let element j' have degree L_{j',u_1} and L_{j',u_2} polynomial basis functions. When the degrees on both elements match, we can choose $\bar{N}_B$ to be a weighted vector of Legendre polynomials on the shared edge and use the results from Section 7.5 for Dirichlet boundary conditions to obtain

$$\begin{bmatrix} 0 & \cdots & 0 & \underline{C}_i & 0 & \cdots & 0 & -\underline{C}_{i'} & 0 & \cdots & 0 \end{bmatrix} \bar{\phi} = 0 \tag{9.32}$$

where $\underline{C}_i$ appears in the jth block column and $-\underline{C}_{i'}$ appears in the j'th block column[4].

[4]It may also be necessary to change the sign of entries associated with odd basis functions in the second constraint block if the parametrizations of the edges on each element are oriented opposite to each other. This is simple to check by taking the inner product of the two edge vectors. Since the inner product returns the magnitude of the two edge vectors multiplied by the cosine of the angle between them, we know that

As with Dirichlet boundary conditions, we append this block row equation to the existing constraint matrix $\underline{C}$ and vector $\bar{d}$. The forms of the block entries are identical to those arising from Dirichlet boundary constraints, and we need to follow the same procedure to ensure that we preserve full rank. When imposing this continuity constraint, we remove the element-edge combinations (j,i) and (j',i') from the vertex-to-edge incidence list wherever they appear and modify the corresponding constraint equations if a vertex list in the vertex-to-edge incidence list becomes empty.

Unfortunately, when the polynomial degree on neighboring elements differ, we must modify the matrices $\underline{C}_i$ used to compute the inter-element continuity constraints. This is because the number of rows in $\underline{C}_i$ and $\underline{C}_{i'}$ differ when the degree of the basis functions differ. To find conforming matrices, we interpret lower degree polynomial expansions as being embedded into higher degree polynomial expansions so that both $\phi_j(\bar{x})$ and $\phi_{j'}(\bar{x})$ can be written using a common number of basis functions. We accomplish this by using new matrices of coefficients

$$\underline{\phi}_{j,\text{pad}} = \underline{P}_{j,u_1}\underline{\phi}_j\underline{P}_{j,u_2}^T, \tag{9.33}$$

$$\underline{\phi}_{j',\text{pad}} = \underline{P}_{j',u_1}\underline{\phi}_{j'}\underline{P}_{j',u_2}^T, \tag{9.34}$$

where

$$\underline{P}_{m,u_n} = \begin{bmatrix} \underline{I} \\ \underline{0} \end{bmatrix} \tag{9.35}$$

has identity block $\underline{I}$ with size $(L_{m,u_n}+1)\times(L_{m,u_n}+1)$ and zero block with size $(L_{u_n,\max}-L_{m,u_n})\times(L_{m,u_n}+1)$, where $L_{u_n,\max}=\max(L_{j,u_n},L_{j',u_n})$ and $m=j$ or j' and $n=1$ or 2. By padding the coefficient matrices in this way, we can use expansions

$$\phi_j(\bar{u}) = \left(\bar{N}(u_2)\otimes\bar{N}(u_1)\right)^T \text{vec}\left(\underline{P}_{j,u_1}\underline{\phi}_j\underline{P}_{j,u_2}^T\right), \tag{9.36}$$

$$\phi_{j'}(\bar{u}) = \left(\bar{N}(u_2)\otimes\bar{N}(u_1)\right)^T \text{vec}\left(\underline{P}_{j',u_1}\underline{\phi}_{j'}\underline{P}_{j',u_2}^T\right), \tag{9.37}$$

where the basis functions $\bar{N}(u_2)$ are of degree $L_{u_2,\max}$ and $\bar{N}(u_1)$ are of degree $L_{u_1,\max}$. Since the number of basis functions are now common to both elements, we can specify $\bar{N}_B$ compatibly as weighted Legendre polynomials of the same degree.

As an example, consider the first term in the boundary integral (9.31) on edge Γ_i with $i=1,2$, given by

$$\int_{\Gamma_i}\bar{N}_B(\bar{x})\,\phi_j(\bar{x})\,d\Omega = \int_{\Gamma_i}\bar{p}(u_2)\left(\bar{N}(u_2)\otimes\bar{N}((-1)^i)\right)^T\text{vec}\left(\underline{P}_{j,u_1}\underline{\phi}_j\underline{P}_{j,u_2}^T\right)d\Omega. \tag{9.38}$$

the edge parametrizations point in opposite directions if the returned value is negative.

Using property (5.13), we obtain

$$\int_{\Gamma_i} \bar{N}_B(\bar{x})\,\phi_j(\bar{x})\,d\Omega = \int_{\Gamma_i} \bar{p}(u_2)\left(\bar{N}(u_2)\otimes\bar{N}((-1)^i)\right)^T\left(\underline{P}_{j,u_2}\otimes\underline{P}_{j,u_1}\right)\bar{\phi}_j d\Omega. \tag{9.39}$$

Then, using the same manipulations as in (7.147)-(7.149), along with property (5.12), we find that

$$\int_{\Gamma_i} \bar{N}_B(\bar{x})\,\phi_j(\bar{x})\,d\Omega = \frac{1}{\sqrt{2}}\underline{\tilde{S}}^T\underbrace{\left(\underline{P}_{j,u_2}\otimes\left(\bar{e}_1+(-1)^i\bar{e}_2\right)^T\underline{P}_{j,u_1}\right)}_{\underline{C}_{p,ij}}\bar{\phi}_j. \tag{9.40}$$

A similar analysis yields

$$\int_{\Gamma_i} \bar{N}_B(\bar{x})\,\phi_j(\bar{x})\,d\Omega = \frac{1}{\sqrt{2}}\underline{\tilde{S}}^T\underbrace{\left(\left(\bar{e}_1+(-1)^i\bar{e}_2\right)^T\underline{P}_{j,u_2}\otimes\underline{P}_{j,u_1}\right)}_{\underline{C}_{p,ij}}\bar{\phi}_j \tag{9.41}$$

for boundary edges Γ_i with $i = 3, 4$. Since the second integral in (9.31) is of the same form, we find that the appropriate constraint equations are

$$\left[\begin{array}{ccccccccccc} 0 & \cdots & 0 & \underline{C}_{p,ij} & 0 & \cdots & 0 & -\underline{C}_{p,i'j'} & 0 & \cdots & 0 \end{array}\right]\bar{\phi} = 0 \tag{9.42}$$

which take into account any possible polynomial degree mismatch between elements. We are able to remove the factor $\frac{1}{\sqrt{2}}\underline{\tilde{S}}^T$ by multiplication by its inverse. As in the matching polynomial degree case, we need to check for rank deficiency when appending such matrices to the global constraint matrix $\underline{C}$ and vector $\bar{d}$. This is because the padding of coefficients never changes the lower order equations, which are those which cause rank problems. The same procedure for modifying equations to preserve full rank using the vertex-to-edge incidence list continues to apply. Once the global constraint matrix and vector have been assembled, we solve the saddle point system

$$\left[\begin{array}{cc} \underline{A} & \underline{C}^T \\ \underline{C} & 0 \end{array}\right]\left[\begin{array}{c} \bar{\phi} \\ \bar{\nu} \end{array}\right] = \left[\begin{array}{c} \bar{b} \\ \bar{d} \end{array}\right] \tag{9.43}$$

which is not singular. Note that each block in the saddle point matrix is composed of blocks derived from the analysis of a single element as performed in Chapters 7 and 8.

9.3 Including Non-Conforming Constraints

When we solve (9.1) using a non-conforming mesh produced using the procedure described in Section 9.1, the details regarding the assembly of the operator matrix $\underline{A}$ and forcing vector $\bar{b}$ remain unchanged. In addition, imposing Dirichlet boundary conditions also remains unchanged. The added difficulty arises in imposing inter-element continuity constraints between adjacent elements.

To fix ideas, we consider imposing a single set of continuity constraints between two adjacent elements. Without loss of generality, we will assume that element j belongs to a coarser level of the quadtree than element j'. We also allow for both elements to belong to the same level. This case has already been handled by the conforming continuity constraints of the previous section but we will see that our formulation coincides with this result as a special case. We consider constraints of the form

$$\int_{\Gamma_{i'}} \bar{N}_B(\bar{x})[\phi_j(\bar{x}) - \phi_{j'}(\bar{x})]d\Omega = 0 \tag{9.44}$$

where $\Gamma_{i'}$ corresponds to the i'th edge of the finer element j'. This edge makes up only a portion of the edge Γ_i corresponding to edge i of the coarser element j. We perform the same type of padding as in (9.36)-(9.37) to represent $\phi_j(\bar{x})$ and $\phi_{j'}(\bar{x})$ with the same number of basis functions. We choose the weight functions

$$\bar{N}_B(\bar{x}) = \|\underline{\tilde{X}}_{i'}\underline{\tilde{D}}\bar{p}\,(u_{k'})\,\|_2^{-1}\sqrt{2}\underline{\tilde{S}}^{-T}\bar{p}(u_{k'}) \tag{9.45}$$

with $k' = 2$ if $i' = 1, 2$, and $k' = 1$ if $i' = 3, 4$. These are the same weight functions we have used previously to obtain constraint matrices $\underline{C}_{p,ij}$ in the conforming case (the norm term is used to cancel the variable transformation when computing the line integral and the term $\sqrt{2}\underline{\tilde{S}}^{-T}$ simplifies the constraints). With this choice,

$$\int_{\Gamma_{i'}} \bar{N}_B(\bar{x})\phi_{j'}(\bar{x})\,d\Omega = \underline{C}_{p,i'j'}\bar{\phi}_{j'} \tag{9.46}$$

which provides the discretization for the second term in (9.44). Thus far, this is no different from the development for the conforming case.

The difficulty arises when considering the first term in (9.44). In particular, the local coordinate $u_{k'} \in (-1, 1)$ appearing in the weight functions $\bar{N}_B(\bar{x})$ is not the same as the local coordinate u_k on edge Γ_i of the coarser element. This means that we cannot directly use $\underline{C}_{p,ij}\bar{\phi}_j$ when discretizing the first term in (9.44). The key idea is to relate the two sets of coordinates. We can write $u_k = au_{k'} + b$ where constants a and b are chosen so that

$u_{k'} \in (-1,1)$ parametrizes only the shared portion of edges $\Gamma_{i'}$ and Γ_i (we will later explain how to select a and b for elements in a quadtree mesh). To fix ideas, suppose that, for now, $k = 2$ (which means that $i = 1, 2$). Then ϕ_j restricted to either of those two edges can be written as

$$\phi_j(\bar{x}) = \left(\bar{N}\left(u_2\right) \otimes \bar{N}((-1)^i)\right)^T \operatorname{vec}\left(\underline{P}_{j,u_1} \underline{\phi}_j \underline{P}_{j,u_2}^T\right) \tag{9.47}$$

$$= \left(\bar{N}\left(a u_{k'} + b\right) \otimes \bar{N}((-1)^i)\right)^T \operatorname{vec}\left(\underline{P}_{j,u_1} \underline{\phi}_j \underline{P}_{j,u_2}^T\right). \tag{9.48}$$

In our development, we will exploit the fact that

$$\bar{p}(ax + b) = \underline{L}\bar{p}(x) \tag{9.49}$$

where $\underline{L}$ is a lower triangular matrix, $x \in (-1,1)$, and a and b are chosen so that $ax + b$ lies inside $(-1,1)$. We will explain how to compute the entries of $\underline{L}$ shortly. Since $\bar{N}(x) = \tilde{\underline{S}}\bar{p}(x)$ and $\bar{N}(ax+b) = \tilde{\underline{S}}\bar{p}(ax+b)$, multiplying (9.49) by $\tilde{\underline{S}}$ yields

$$\bar{N}(ax + b) = \tilde{\underline{S}}\underline{L}\bar{p}(x) \tag{9.50}$$

$$= \tilde{\underline{S}}\underline{L}\underbrace{\tilde{\underline{S}}^{-1}\tilde{\underline{S}}}_{\underline{I}}\bar{p}(x) \tag{9.51}$$

$$= \underbrace{\tilde{\underline{S}}\underline{L}\tilde{\underline{S}}^{-1}}_{\tilde{\underline{L}}}\bar{N}(x). \tag{9.52}$$

Therefore, if we know $\underline{L}$, combining (9.52) with $x = u_{k'}$ and (9.48) gives

$$\phi_j(\bar{x}) = \left(\tilde{\underline{L}}\bar{N}(u_{k'}) \otimes \bar{N}((-1)^i)\right)^T \operatorname{vec}\left(\underline{P}_{j,u_1} \underline{\phi}_j \underline{P}_{j,u_2}^T\right) \tag{9.53}$$

which we can use to evaluate the first term in (9.44).

Using properties (5.11)-(5.13), we write

$$\phi_j(\bar{x}) = \left(\tilde{\underline{L}}\tilde{\underline{S}}\bar{p}(u_{k'}) \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)\right)^T \operatorname{vec}\left(\underline{P}_{j,u_1} \underline{\phi}_j \underline{P}_{j,u_2}^T\right) \tag{9.54}$$

$$= \left(\bar{p}(u_{k'})^T \tilde{\underline{S}}^T \tilde{\underline{L}}^T \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T\right) \operatorname{vec}\left(\underline{P}_{j,u_1} \underline{\phi}_j \underline{P}_{j,u_2}^T\right) \tag{9.55}$$

$$= \left(\bar{p}(u_{k'})^T \tilde{\underline{S}}^T \tilde{\underline{L}}^T \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T\right)\left(\underline{P}_{j,u_2} \otimes \underline{P}_{j,u_1}\right) \operatorname{vec}(\underline{\phi}_j) \tag{9.56}$$

$$= \left(\bar{p}(u_{k'})^T \tilde{\underline{S}}^T \tilde{\underline{L}}^T \underline{P}_{j,u_2} \otimes \frac{1}{\sqrt{2}}\left(\bar{e}_1 + (-1)^i \bar{e}_2\right)^T \underline{P}_{j,u_1}\right) \bar{\phi}_j. \tag{9.57}$$

We substitute this last equation into the first term of (9.44) along with (9.45) to obtain

$$\int_{\Gamma_{i'}} \bar{N}_B(\bar{x})\phi_j(\bar{x})\,d\Omega = \int_{-1}^{1} \underline{\tilde{S}}^{-T}\bar{p}(u_{k'})\left(\bar{p}(u_{k'})^T\underline{\tilde{S}}^T\underline{\tilde{L}}^T\underline{P}_{j,u_2}\otimes\left(\bar{e}_1+(-1)^i\bar{e}_2\right)^T\underline{P}_{j,u_1}\right)\bar{\phi}_j\,du_{k'} \tag{9.58}$$

$$= \left(\underline{\tilde{L}}^T\underline{P}_{j,u_2}\otimes\left(\bar{e}_1+(-1)^i\bar{e}_2\right)^T\underline{P}_{j,u_1}\right)\bar{\phi}_j. \tag{9.59}$$

Notice that this matrix is similar to $\underline{C}_{p,ij}$. In fact, using (5.27), we have

$$\int_{\Gamma_{i'}} \bar{N}_B(\bar{x})\phi_j(\bar{x})\,d\Omega = \left(\underline{\tilde{L}}^T\otimes 1\right)\underbrace{\left(\underline{P}_{j,u_2}\otimes\left(\bar{e}_1+(-1)^i\bar{e}_2\right)^T\underline{P}_{j,u_1}\right)}_{\underline{C}_{p,ij}}\bar{\phi}_j \tag{9.60}$$

$$= \underline{\tilde{L}}^T\underline{C}_{p,ij}\bar{\phi}_j. \tag{9.61}$$

Similar manipulations when $k=1$ and $i=3,4$, yields

$$\int_{\Gamma_{i'}} \bar{N}_B(\bar{x})\phi_j(\bar{x})\,d\Omega = \left(\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T\underline{P}_{j,u_2}\otimes\underline{\tilde{L}}^T\underline{P}_{j,u_1}\right)\bar{\phi}_j \tag{9.62}$$

$$= \left(1\otimes\underline{\tilde{L}}^T\right)\underbrace{\left(\left(\bar{e}_1+(-1)^i\,\bar{e}_2\right)^T\underline{P}_{j,u_2}\otimes\underline{P}_{j,u_1}\right)}_{\underline{C}_{p,ij}}\bar{\phi}_j \tag{9.63}$$

$$= \underline{\tilde{L}}^T\underline{C}_{p,ij}\bar{\phi}_j. \tag{9.64}$$

Thus, to account for non-conforming constraints, the only change to the conforming constraint matrices is to take the product with the transpose of $\underline{\tilde{L}}$. To summarize, (9.44) is discretized as

$$\underline{\tilde{L}}^T\underline{C}_{p,ij}\bar{\phi}_j - \underline{C}_{p,i'j'}\bar{\phi}_{j'} = 0 \tag{9.65}$$

or, when there are multiple elements, as

$$\begin{bmatrix} 0 & \cdots & 0 & \underline{\tilde{L}}^T\underline{C}_{p,ij} & 0 & \cdots & 0 & -\underline{C}_{p,i'j'} & 0 & \cdots & 0 \end{bmatrix}\bar{\phi} = 0 \tag{9.66}$$

where $\underline{\tilde{L}}^T\underline{C}_{p,ij}$ appears in block column j and $-\underline{C}_{p,i'j'}$ appears in block column j'. When edges are conforming $\underline{\tilde{L}} = \underline{I}$ and we have the same constraint equations as presented in the previous section.

To fully characterize (9.66), we need to compute $\underline{L}$ in (9.49). A direct approach is to multiply (9.49) by $\bar{p}(x)^T$ from the right. Integrating, we obtain

$$\underline{L} = \int_{-1}^{1} \bar{p}(ax+b)\bar{p}(x)^T dx. \tag{9.67}$$

This expression confirms that $\underline{L}$ is in fact lower triangular, because the entry in the ith row and jth column is given by

$$(\underline{L})_{ij} = \int_{-1}^{1} p_{i-1}(ax+b)p_{j-1}(x)\,dx. \tag{9.68}$$

When $j > i$, polynomial $p_{i-1}(ax+b)$ has degree strictly smaller than $p_{j-1}(x)$ and, as a consequence, they are orthogonal (meaning that the integral is zero). This means that the strict upper part of $\underline{L}$ is zero. Rather than directly evaluate the remaining nonzero entries, we use the three-term recurrence relation (2.36) to devise a recursive algorithm to compute $\underline{L}$.

To do so, we rewrite the recurrence relation by isolating the term xp_k. This gives

$$xp_k(x) = \frac{k+1}{\sqrt{(2k+1)(2k+3)}}p_{k+1}(x) + \frac{k}{\sqrt{(2k-1)(2k+1)}}p_{k-1}(x), \qquad k = 0, 1, 2, ... \tag{9.69}$$

Repeating this expression for $k = 0, 1, ..., n$, yields the equation

$$x\bar{p}(x) = \underline{J}_{n+1}\bar{p}(x) + \sqrt{\beta_{n+1}}p_{n+1}(x)\bar{e}_{n+1} \tag{9.70}$$

where

$$\underline{J}_{n+1} = \begin{bmatrix} 0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & 0 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & 0 & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_n} \\ & & & \sqrt{\beta_n} & 0 \end{bmatrix} \tag{9.71}$$

is the truncated Jacobi matrix and $\sqrt{\beta_k} = k/\sqrt{(2k+1)(2k-1)}$. Typically, (9.70) is encountered when computing Gauss-Legendre quadrature rules via eigenvalue routines [159]. Instead, here we rewrite (9.70) with the argument $ax+b$ to obtain

$$(ax+b)\bar{p}(ax+b) = \underline{J}_{n+1}\bar{p}(ax+b) + \sqrt{\beta_{n+1}}p_{n+1}(ax+b)\bar{e}_{n+1}, \tag{9.72}$$

then substitute (9.49) to get

$$(ax+b)\underline{L}\bar{p}(x) = \underline{J}_{n+1}\underline{L}\bar{p}(x) + \sqrt{\beta_{n+1}}p_{n+1}(ax+b)\bar{e}_{n+1}. \tag{9.73}$$

Using (9.70) to remove the term $x\bar{p}(x)$ yields

$$a\underline{L}[\underline{J}_{n+1}\bar{p}(x) + \sqrt{\beta_{n+1}}p_{n+1}(x)\bar{e}_{n+1}] + b\underline{L}\bar{p}(x) = \underline{J}_{n+1}\underline{L}\bar{p}(x) + \sqrt{\beta_{n+1}}p_{n+1}(ax+b)\bar{e}_{n+1}. \quad (9.74)$$

Finally, multiplying from the right by $\bar{p}(x)^T$ and integrating over $(-1,1)$ yields the matrix equation

$$a\underline{L}\underline{J}_{n+1} + b\underline{L} = \underline{J}_{n+1}\underline{L} + \bar{e}_{n+1}\underbrace{\sqrt{\beta_{n+1}}\int_{-1}^{1} p_{n+1}(ax+b)\bar{p}(x)^T dx}_{\bar{p}_{n+1}^T}. \quad (9.75)$$

The term $\sqrt{\beta_{n+1}}p_{n+1}(x)\bar{e}_{n+1}$ has vanished because $p_{n+1}(x)$ is orthogonal to all entries in $\bar{p}(x)$. Letting $\underline{\hat{A}} = \underline{J}_{n+1} - b\underline{I}$ and $\underline{\hat{B}} = a\underline{J}_{n+1}$ shows that

$$\underline{\hat{A}}\underline{L} - \underline{L}\underline{\hat{B}} = -\bar{e}_{n+1}\bar{p}_{n+1}^T \quad (9.76)$$

which is a Sylvester equation for $\underline{L}$ [25].

Since entry $(\underline{L})_{11} = 1$ (because $p_0(ax+b) = p_0(x) = 1/\sqrt{2}$), $\underline{\hat{A}}$ and $\underline{\hat{B}}$ are tridiagonal, and $\bar{e}_{n+1}\bar{p}_{n+1}^T$ is almost entirely zero, we can solve for the rows of $\underline{L}$ sequentially without computing $\bar{p}_{n+1}$. If we denote the kth row of $\underline{L}$ by $\bar{l}_k^T$, we note that row k of (9.76) is

$$(\underline{\hat{A}})_{k,k-1}\bar{l}_{k-1}^T + (\underline{\hat{A}})_{k,k}\bar{l}_k^T + (\underline{\hat{A}})_{k,k+1}\bar{l}_{k+1}^T - \bar{l}_k^T\underline{\hat{B}} = 0 \quad (9.77)$$

as long as k is not the last row (then the right hand side would be nonzero). Since we know that $\bar{l}_1^T = \bar{e}_1^T$, this allows us to compute

$$\bar{l}_{k+1}^T = \frac{1}{(\underline{\hat{A}})_{k,k+1}}\left[\bar{l}_k^T\underline{\hat{B}} - (\underline{\hat{A}})_{k,k}\bar{l}_k^T - (\underline{\hat{A}})_{k,k-1}\bar{l}_{k-1}^T\right] \quad (9.78)$$

sequentially for $k = 1, 2, ...$ (we ignore zero indexed terms). To compute the last row, simply enlarge $\underline{L}$, $\underline{\hat{A}}$, and $\underline{\hat{B}}$ by one row and column but stop the recurrence relation one iteration short of completion. This is a stable algorithm requiring $\mathcal{O}(n^2)$ computations and storage to obtain the $\mathcal{O}(n^2)$ entries in $\underline{L}$. This complexity is comparable to the method in [69].

We also take this opportunity to remark that it is possible to perform the matrix-vector product with $\underline{L}$ efficiently when n is large. This is because $\underline{L}$ has displacement rank 1 (the right hand side of (9.76) is a rank 1 matrix). Note that $\underline{\hat{A}}$ and $\underline{\hat{B}}$ are simultaneously diagonalizable. In particular, computing the eigenvalue decomposition

$$\underline{J}_{n+1}\underline{V} = \underline{V}\underline{\Lambda} \quad (9.79)$$

of $\underline{J}_{n+1}$ where $\underline{V}$ is the matrix whose columns are eigenvectors of $\underline{J}_{n+1}$ and $\underline{\Lambda}$ is the diagonal matrix with corresponding eigenvalues along the diagonal, we directly obtain the eigenvectors of both $\underline{\hat{A}}$ and $\underline{\hat{B}}$. Their eigenvalues are $\underline{\Lambda}_{\hat{A}} = \underline{\Lambda} - b\underline{I}$ and $\underline{\Lambda}_{\hat{B}} = a\underline{\Lambda}$ respectively. Note that since $\underline{J}_{n+1}$ is symmetric, the eigenvector matrix $\underline{V}$ can be chosen orthonormal so that $\underline{V}^T\underline{V} = \underline{I}$. If we multiply (9.76) from the left by $\underline{V}^T$, from the right by $\underline{V}$, and substitute $\underline{L} = \underline{V}\hat{\underline{L}}\underline{V}^T$, we obtain

$$\underline{V}^T\underline{\hat{A}}\underline{V}\hat{\underline{L}}\underbrace{\underline{V}^T\underline{V}}_{\underline{I}} - \underbrace{\underline{V}^T\underline{V}}_{\underline{I}}\hat{\underline{L}}\underline{V}^T\underline{\hat{B}}\underline{V} = -\underbrace{\underline{V}^T\bar{e}_{n+1}}_{\hat{e}}\underbrace{\bar{p}_{n+1}^T\underline{V}}_{\hat{p}^T}. \tag{9.80}$$

Since $\underline{\hat{A}}\underline{V} = \underline{V}\underline{\Lambda}_{\hat{A}}$ and $\underline{\hat{B}}\underline{V} = \underline{V}\underline{\Lambda}_{\hat{B}}$, we have

$$\underline{\Lambda}_{\hat{A}}\hat{\underline{L}} - \hat{\underline{L}}\underline{\Lambda}_{\hat{B}} = -\hat{e}\hat{p}^T \tag{9.81}$$

from which we can directly express the entries of $\hat{\underline{L}}$. They are

$$(\hat{\underline{L}})_{ij} = -\frac{(\hat{e})_i(\hat{p})_j}{(\lambda_{\hat{A}})_i - (\lambda_{\hat{B}})_j}. \tag{9.82}$$

Matrices of this form are called Cauchy-like [25] and their matrix-vector product can be computed in $\mathcal{O}(n)$ operations using the fast multipole method (FMM) [58, 59, 60, 61]. Similarly, the matrix-vector products with $\underline{V}$ and $\underline{V}^T$ can also be accelerated by FMM since $\underline{J}_{n+1}$ is tridiagonal [55]. For details regarding the FMM as applied in this and other contexts, see Appendix B. For this method to be efficient, we have to be able to compute vector $\bar{p}_{n+1}$ efficiently. Recall that this vector is a scaled vector of Legendre coefficients for the function $p_{n+1}(ax + b)$. We can sample this function efficiently using $\mathcal{O}(1)$ evaluation of Legendre polynomials [160] and use those samples as in Section 4.2 and Appendix A to compute the Legendre coefficients with $\mathcal{O}(n(\log n)^2)$ operations.

Now that we have a way to compute matrix $\underline{L}$ given parameters a and b, we can use it to assemble all constraint equations in a non-conforming mesh as produced in Section 9.1. Like with the conforming case, (9.66) has full row rank when taken alone, but can lead to redundant equations when taken together in a global constraint matrix. A systematic procedure can be followed to restore full row rank in the global case (similar to the method described in the previous section). First, we construct a set of vertex-to-edge incidence lists (one for each level of the quadtree) which enumerate distinct vertices that belong to elements of a given level of the tree and track which edges of elements on that level are incident to each vertex. Then, we construct the global constraint matrix in a local fashion, visiting each element edge (beginning with all elements belonging to the finest level of the tree, then

moving to the next coarser level, etc.) and imposing continuity to an adjacent leaf element in the tree. The tree data structure allows us to determine the parameters a and b in the map along each edge so as to compute the appropriate change of basis matrix $\underline{L}$. Each time continuity is imposed along an edge, we verify if all edges on that given level of the tree have been visited for a given vertex in the vertex-to-edge incidence list. For each vertex whose list has been completely visited, we have one redundant equation in the global constraint matrix, and can remove the lowest order equation from the new set of local constraints to restore full rank to the global system.

There is also the possibility of losing global full rank when multiple fine elements share a common edge with a single coarse element. In particular, when an edge must be constrained to match a lower polynomial degree on an adjacent element, we zero certain basis functions through the padding matrices (9.35). If a second set of constraint equations requires a similar reduction in the degree along the edge, naively imposing the local constraints leads to a second set of redundant constraints zeroing the same basis functions. Instead, we track which basis functions have been zeroed on an edge and discard those constraints which would lead to redundancy. These are always the higher order equations in the local constraint and do not interfere with the removal of the lowest order constraints described above.

It remains to explain how parameters a and b are selected at each edge. By virtue of sequentially imposing constraints element by element starting at the finest level of the tree and moving to the next level of the tree after all finer element constraints have been imposed, we always impose constraints in the form (9.66) (that is, always from the perspective of a finer element connecting to a coarser element). The quadtree lets us query the neighbor of an element on the same level of the tree (which may not be a leaf). We can then trace up the quadtree (by going to the parent of the neighbor, then its parent, etc.) until a leaf is found. If no leaf is found, this edge corresponds to a coarse edge relative to a previous finer level and has already been constrained. If a leaf is found, we must impose the constraint. We keep track of the sequence of nodes in the quadtree that were visited in finding the leaf to compute a and b. Because each level of the quadtree is comprised of squares constructed by subdividing a square on the previous level into four squares, the transformation used to represent the coarse basis functions in terms of fine basis functions is of the form

$$ax + b = \{[(x + s_1)/2 + s_2]/2 + \cdots\}/2. \tag{9.83}$$

The signs $s_i \in \{-1, 1\}$ in this composition of functions $(x + s_i)/2$ depend on the sequence of nodes used to find the neighbor leaf node starting from the fine neighbor. Figure 9.6 illustrates one possible configuration of fine and coarse elements and the sequence of signs

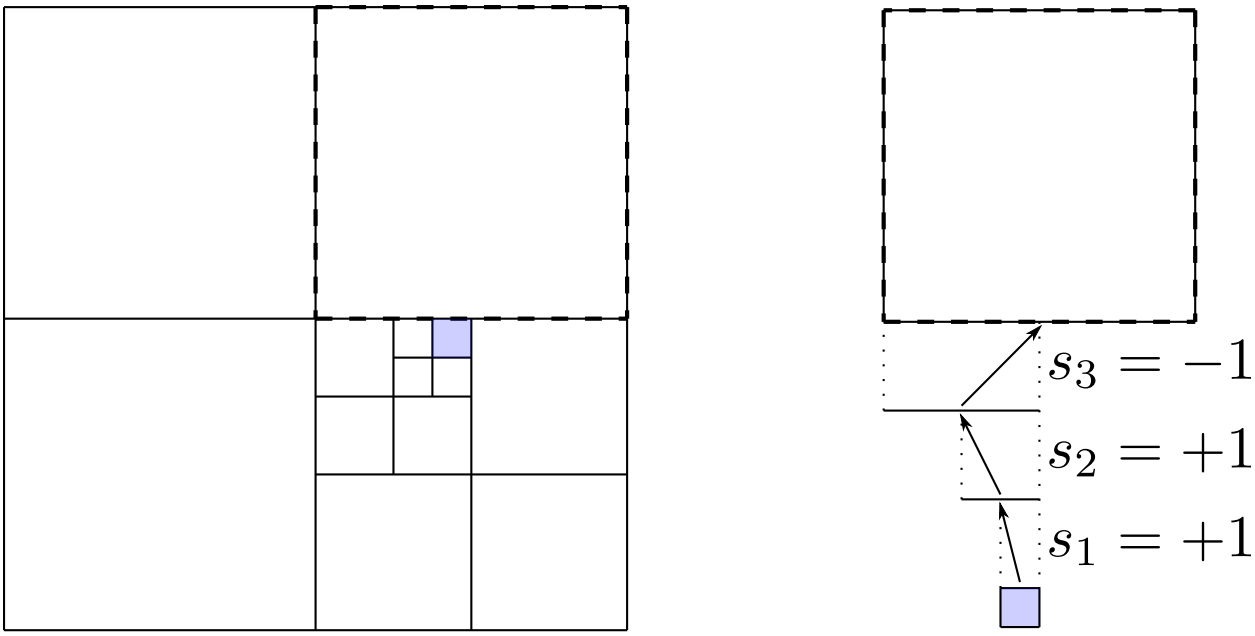

Figure 9.6: (left) Example quadtree mesh where continuity between a fine element (shaded) and neighboring element above it (dashed outline) must be imposed. (right) Schematic illustrating the selection of signs s_i for the associated edge configuration: s_3 is negative because the fine element shares a portion of the left bisection of the coarse edge, s_2 is positive because the fine element shares a portion of the right bisection of the previous bisection, and s_1 is positive because the fine element edge corresponds to the right bisection of the previous bisection.

required to determine a and b for a given edge constraint. If there is a difference of m levels between the fine element to the coarse leaf neighbor, then

$$a = \frac{1}{2^m}, \qquad b = \frac{1}{2^m} \sum_{i=1}^{m} s_i 2^{i-1}, \tag{9.84}$$

where s_1 is the sign corresponding to the first step up the quadtree and s_m corresponds to the mth step required to finally reach the neighboring leaf.

With the global constraint matrix $\underline{C}$ assembled, we obtain a saddle point system of form (9.43) where each block depends upon multiple elements (rather than a single element). One benefit of such a construction is that the change of basis matrix $\tilde{\underline{L}}$ along each edge yields a quantitative way of assessing whether the saddle point system is invertible in finite precision. In particular, it quantifies whether the mismatch in adjacent polynomial degree and element size is too severe, leading to extreme ill-conditioning of the global saddle point system and suggests what type of refinement to avoid in order to compute accurate solutions.

This ill-conditioning arises because the matrix $\tilde{\underline{L}}$ tends to have decaying diagonal entries when the number of levels m between coarse and fine element is sufficiently large and/or the

polynomial degree is high enough. The diagonals decay because row $k+1$ of $\underline{L}$ corresponds to the coefficients in a Legendre expansion of the function $p_k(ax+b)$. When $x \in (-1,1)$ and a and b are chosen as in (9.84), then $ax+b$ represents a small subinterval of $(-1,1)$ (particularly when the number of levels m is large). As a consequence, $p_k(ax+b)$ varies slowly over $(-1,1)$ compared to $p_k(x)$ and only a small number of low degree Legendre polynomials make a significant contribution to the expansion (higher degree polynomials contribute, but substantially less) causing entries near the diagonal of $\underline{L}$ to decay in magnitude. Matrix $\underline{\tilde{L}}$ inherits this property from $\underline{L}$.

Since the constraints along edges involve the product of the transpose of $\underline{\tilde{L}}$ (which is upper triangular with entries in the final rows possibly decaying), corresponding rows of the global constraint matrix $\underline{C}$ may become near zero, causing an effective loss of full row rank in finite precision when certain combinations of polynomial degree and edge refinements are present. Introducing a 2:1 mesh refinement rule common to quadtree-based finite element meshes can alleviate this problem with low or moderate degree polynomials on neighboring elements, but is ineffective at high polynomial degree (even at degree 32, the diagonal has decayed to approximately 10^{-10}). Instead, we can monitor the magnitude of the diagonal and discard any constraints which have decayed below a user specified threshold. This results in a non-conforming finite element method. Alternatively, we can use the diagonal to indicate where in the mesh to prohibit further refinement of degree or element size.

9.4 Adaptive Finite Element Considerations

In practice, when a solution ϕ is not known *a priori*, we may need to solve a sequence of saddle point problems, each assembled using the methods of the previous section, to produce an approximate solution which is accurate to a given tolerance. For this purpose, we describe an *hp*-adaptive algorithm similar to the one described towards the end of Section 4.4. First, we need to compute an error indicator on each element j, which we label η_j. We sum the magnitude squared of each entry in the last two rows and columns of the coefficient matrix $\underline{\phi}_j$ and take the square root. This is a measure of how much the coefficients have decayed in each local expansion which we use as indicator η_j. Note that this is one possible extension of the one-dimensional error indicator which took the square root of the squares of the last two entries in the coefficient vector $\bar{\phi}_j$. As in one-dimension, we use two rows and two columns instead of only one to avoid mischaracterizing error when the solution has certain even or odd symmetries. If all error indicators η_j fall below the user specified tolerance ϵ_{tol}, we stop the *hp*-adaptive process. If they do not, we use a refinement criterion to determine which elements to refine (for simplicity, in the following we set $L_{j,u_1} = L_{j,u_2}$).

Different refinement criteria exist, each suitable for a different class of problem. Here we describe three possible criteria. We describe the first two briefly so as to motivate the third which we use in practice in the examples to follow. The first refinement criterion is uniform refinement, in which all elements in the mesh are refined, regardless of the error indicators η_j. This type of refinement scheme may not efficiently distribute degrees of freedom, particularly for solutions with large regions of slow variation and small regions with rapid variation. In these situations, uniform refinement requires more degrees of freedom than necessary to reach a specified error tolerance.

The second refinement criterion computes the total error

$$\eta^2 = \sum_{j=1}^{N_e} \eta_j^2 \tag{9.85}$$

and finds the elements with largest contributions to η^2. Then elements corresponding to the smallest possible set of element indices $\mathcal{J}_{\max}$ that add to a fixed percentage Θ of the total error

$$\Theta \approx \frac{1}{\eta^2} \sum_{j \in \mathcal{J}_{\max}} \eta_j^2 \tag{9.86}$$

are chosen for refinement. The user must specify their choice of $0 < \Theta \leq 1$ before the algorithm begins. This method is intended to allow for a more efficient use of degrees of freedom in meeting a required error tolerance. Unfortunately, for problems with singular behavior, the method may only refine a small number of elements at each iteration, causing each solution step of the saddle point matrix to contain roughly the same number of unknowns[5].

Finally, a third refinement criterion refines a fixed fraction of the total number of elements at each step. This is the method we use in our examples. In this type of refinement strategy, we sort elements by decreasing error indicator η_j. We then refine a fixed fraction of the total number of elements (those with the largest error indicators). By choosing a fixed fraction, we can ensure that we always have a significant number of additional degrees of freedom after each refinement step, but we will not have an optimal mesh requiring the least number of degrees of freedom for a given error tolerance. Typically, the choice of fraction, which we denote $0 < \varrho < 1$, is made so that the final cost of the *hp*-adaptive algorithm is twice the cost of the final solve of the saddle point system. If N is the initial number of degrees of freedom, the choice of ϱ depends on the asymptotic cost of the solution process $\mathcal{O}(N^{\varpi})$, as

[5]The refinement criterion described in Section 4.4 behaves similarly. Recall that this criterion finds the element with the largest indicator $\eta_{\max}$ and uses it to select elements with indicators larger than $\zeta\eta_{\max}$ for refinement ($\zeta < 1$ is a user specified threshold).

well as a multiplier m_{hp} for how many additional degrees of freedom are used per refined element (for example, if a refinement results in four times as many degrees of freedom for a given element, then $m_{hp} = 4$).

To see how to choose ϱ, we begin by counting the number of degrees of freedom at each iteration. In the initial phase of the algorithm, we have $N_0 = N$ degrees of freedom. In the first iteration, we have

$$N_1 = (1 - \varrho)N + m_{hp}\varrho N \tag{9.87}$$

since ϱN counts how many degrees of freedom belong to the elements that must be refined, $m_{hp}\varrho N$ counts how many degrees of freedom are now used for those refined elements, and $(1 - \varrho)N$ counts the number of degrees of freedom belonging to unrefined elements. It is useful to write

$$N_1 = \underbrace{[(1 - \varrho) + m_{hp}\varrho]}_{\varsigma} N. \tag{9.88}$$

By doing so, we note that in the second iteration, there are

$$N_2 = \varsigma N_1 \tag{9.89}$$

$$= \varsigma^2 N \tag{9.90}$$

degrees of freedom, so that, in general, the kth iteration has $N_k = \varsigma^k N$ degrees of freedom. At each iteration, we perform a linear system solve that costs $\mathcal{O}(N_k^\varpi)$ operations. Thus, roughly speaking, the total cost C_{total} of the hp-adaptive algorithm after M solves is

$$C_{\text{total}} \approx \sum_{k=0}^{M-1} N_k^\varpi \tag{9.91}$$

$$= \sum_{k=0}^{M-1} (\varsigma^k N)^\varpi \tag{9.92}$$

$$= N^\varpi \sum_{k=0}^{M-1} \varsigma^{k\varpi} \tag{9.93}$$

which is equivalent to

$$C_{\text{total}} \approx N^\varpi \frac{1 - \varsigma^{M\varpi}}{1 - \varsigma^\varpi} \tag{9.94}$$

after computing the geometric sum. Multiplying both sides by the denominator and simplifying the right hand side gives

$$(1 - \varsigma^\varpi) C_{\text{total}} \approx N^\varpi - \varsigma^{M\varpi} N^\varpi. \tag{9.95}$$

We notice that the initial solve costs $C_{\text{initial}} \approx N^{\varpi}$ and that the final solve costs

$$C_{\text{final}} \approx N_{M-1}^{\varpi} \tag{9.96}$$
$$= (\varsigma^{M-1} N)^{\varpi} \tag{9.97}$$
$$= \varsigma^{(M-1)\varpi} N^{\varpi} \tag{9.98}$$

which can be written as $C_{\text{final}} \approx \varsigma^{-\varpi} \varsigma^{M\varpi} N^{\varpi}$. This last equation shows that

$$\varsigma^{M\varpi} N^{\varpi} \approx \varsigma^{\varpi} C_{\text{final}} \tag{9.99}$$

which we use in (9.95) to see that

$$(1 - \varsigma^{\varpi}) C_{\text{total}} \approx C_{\text{initial}} - \varsigma^{\varpi} C_{\text{final}}. \tag{9.100}$$

If we choose $\varsigma^{\varpi} = 2$, then

$$C_{\text{total}} \approx 2C_{\text{final}} - C_{\text{initial}} \tag{9.101}$$

and the total cost of the *hp*-adaptive algorithm is approximately twice the cost of the final linear system solve.

We now choose ϱ and ϖ so that $\varsigma^{\varpi} = 2$. For static problems and low frequency problems, it is possible using multigrid methods or domain decomposition methods to have $\varpi = 1$ so that the cost of the linear system solve is linear in the number of degrees of freedom [71, 72]. In such a case, $\varsigma^{\varpi} = 2$ implies

$$(1 - \varrho) + m_{hp} \varrho = 2 \tag{9.102}$$

which we solve for ϱ to obtain

$$\varrho = \frac{1}{m_{hp} - 1}. \tag{9.103}$$

In this thesis, we always refine so that $m_{hp} = 2^d$ where d is the dimension of the problem. That is, in one dimension, we always doubled the number of degrees of freedom when refining an element and, in this chapter, we will always quadruple the number of degrees of freedom when refining an element. Thus the fixed fraction of elements to refine in such a two-dimensional scenario is one third if we want the approximate total cost of the *hp*-adaptive algorithm to be twice that of the final linear system solve. Note that by modifying the choices we have made, we can devise alternative fixed fraction schemes.

To complete the description of the *hp*-adaptive algorithm, we need to specify how to choose between *h*- and *p*-refinement once an element is marked for refinement according to the fixed fraction refinement criterion above. We use an approach similar to the one

described in Section 4.4. That is, we estimate the rate of decay of Legendre coefficients for each element selected for refinement using a least squares method. If the rate of decay is judged rapid, then we perform p-adaption, otherwise we choose h-adaption. We double the polynomial degrees L_{j,u_1} and L_{j,u_2} when performing p-adaption (resulting in a quadrupling of the number of degrees of freedom for the element) or subdivide the element into four elements by bisecting each edge of the original element and setting the polynomial degree of the new elements to the same degree as the original element when performing h-adaption (again resulting in a quadrupling of the number of degrees of freedom). This is where $m_{hp} = 2^d$ with $d = 2$ arises.

To estimate the decay rate, for each element j to refine, we compute Legendre coefficients

$$\underline{\phi}_{j,\text{Leg}} = \tilde{\underline{S}}^T \underline{\phi}_j \tilde{\underline{S}} \tag{9.104}$$

from the integrated Legendre coefficients. We then assume that

$$|(\underline{\phi}_{j,\text{Leg}})_{mn}| \approx \theta^{\sqrt{m^2+n^2}}. \tag{9.105}$$

This is the two-dimensional analog to (4.156). There is no theory paper that directly addresses such decay for Legendre expansion coefficients in two dimensions, however, there is such a paper for Chebyshev polynomials [161] which mirrors the one-dimensional result [109] and which supports such an assumption. In practice, such decay of Legendre expansion coefficients is observed for smooth functions similar to those considered in [161]. Taking the logarithm yields

$$\log |(\underline{\phi}_{j,\text{Leg}})_{mn}| \approx \sqrt{m^2+n^2} \underbrace{\log \theta}_{a_1}. \tag{9.106}$$

We estimate a_1 using the linear least squares fit $\log |(\underline{\phi}_{j,\text{Leg}})_{mn}| \approx \sqrt{m^2+n^2} a_1 + a_2$. If $(\underline{M})_{mn} = \sqrt{(m-1)^2+(n-1)^2}$, we construct

$$\underline{W} = \begin{bmatrix} \text{vec}(\underline{M}) & \bar{e} \end{bmatrix}, \qquad \bar{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \qquad \bar{y} = \text{vec}(\log |\underline{\phi}_{j,\text{Leg}}|), \tag{9.107}$$

where $\bar{e}$ is the vector of all ones and $\log |\cdot|$ is meant to be applied entrywise to $\underline{\phi}_{j,\text{Leg}}$. Then $\underline{W}\bar{a} \approx \bar{y}$ and we solve this over-determined system using QR factorization. Once a_1 is estimated, we compute $\theta = e^{a_1}$ and choose p-refinement if θ is less than a user specified threshold $0 < \theta_t < 1$. Otherwise, h-refinement is performed. In our examples, we choose $\theta_t = 1/2$.

9.5 Bounded Electrostatic Examples

In this section, we solve three examples to illustrate how the *hp*-adaptive method behaves for static problems. Each example naturally lends itself to a two-dimensional model and analysis. In particular, we begin with a simple Poisson problem that computes an eigenmode of the unit disk [4]. Since the solution to this problem is known, we use it to show that the method is accurate. For the second and third examples, we solve electrostatic problems from [12] to illustrate how the *hp*-adaptive method behaves in the presence of re-entrant corners.

9.5.1 Eigenfunction of the Laplacian on a Disk

For the first example, we solve the Poisson equation

$$-\nabla \cdot \nabla \phi = \chi_{m,n}^2 J_m\left(\chi_{m,n}\|\bar{x}\|_2\right)\cos\left(m\,\mathrm{atan2}\left(x_2, x_1\right)\right) \tag{9.108}$$

on the domain $\Omega = \{\bar{x} \in \mathbb{R}^2 : \|\bar{x}\|_2 < 1\}$ subject to zero Dirichlet boundary conditions. Here, J_m is the Bessel function of the first kind of integer order m, $\chi_{m,n}$ is the nth zero of J_m, and $\mathrm{atan2}\left(x_2, x_1\right)$ is used to denote the angle in cylindrical coordinates measured from the x_1 axis (commonly denoted as φ). For our particular example, we choose $m = 6$ and $n = 2$ so that $\chi_{6,2} \approx 13.589290170541217$ (this is correct to 17 significant figures). Using the notation for our prototypical PDE, this Poisson equation has $\underline{\alpha} = \underline{I}$, $\beta = 0$,

$$f = \chi_{m,n}^2 J_m\left(\chi_{m,n}\|\bar{x}\|_2\right)\cos\left(m\,\mathrm{atan2}\left(x_2, x_1\right)\right), \tag{9.109}$$

Dirichlet data $p = 0$ on $\partial\Omega$, and an exact solution $\phi_{\text{exact}} = f/\chi_{m,n}^2$. Thus, ϕ_{exact} is an eigenfunction of the Laplacian with eigenvalue $\chi_{m,n}^2$.

To solve this problem, we perform five iterations of *hp*-adaption with a fixed fraction refinement of one third starting from a mesh produced using the procedure described in Section 9.1. We use distance function $\Phi_1(\bar{x}) = \|\bar{x}\|_2 - 1$ to describe the single subdomain $\hat{\Omega}_1$ for this problem. We choose bounding box parameters

$$\bar{x}_{\text{bound}} = \begin{bmatrix} -1.5 \\ -1.5 \end{bmatrix}, \qquad H = 3, \tag{9.110}$$

and choose maximum level of refinement $l_{\max} = 4$ and minimum level of refinement $l_{\min} = 0$. We do not specify any fixed vertices for the mesh. Two iterations of smoothing and mesh optimization are used. All computations are performed to a tolerance of $\epsilon_{\text{tol}} = 10^{-12}$. Figure 9.7 illustrates the computed solution using the finite element method described in this

chapter, as well as the polynomial degree $L_{j,u_1} = L_{j,u_2}$ used on each element to represent ϕ after five iterations of hp-adaption. Degree $L_{j,u_1} = L_{j,u_2} = 8$ basis functions on each element were used on the starting mesh. We note that the adaption algorithm favors p-adaption in this case as one would expect for a smooth solution. In fact, no element was refined using h-adaption at any iteration of the algorithm.

Figure 9.8 illustrates the pointwise error $\log_{10} |\phi - \phi_{\text{exact}}|$ between the computed solution ϕ and the exact solution ϕ_{exact} (sampled with 21 uniformly spaced points in each coordinate of the canonical domain $\bar{u} \in (-1, 1)^2$ for each element), as well as the sparsity pattern of the associated saddle point system used to compute ϕ. The error plot demonstrates that the approximate solution ϕ is accurate to 11 digits or better throughout the domain Ω. Note that this accuracy is achieved despite the fact that the circular boundary is not perfectly modeled by the mesh. The error is a complicated function of the accuracy of resolving the boundary, accuracy in computing the entries of the saddle point matrix via Legendre expansions, and ability of the basis functions on the mesh to represent the exact solution. When any of these three factors is compromised, the total error increases.

9.5.2 Shielded Microstrip Line

As a second example, we solve for the electrostatic potential ϕ of an infinitely long shielded microstrip line whose schematic representation is given on page 99 of [12]. Figure 9.9 illustrates a portion of the shielded microstrip line. The structure consists of: an infinitely long base conductor of width $2b = 10$; a dielectric layer (called the substrate) of thickness $a = b/5$, which lies directly upon the base conductor; and, an infinitely long, thin conducting strip of width $2a$ and thickness $c = a/10$, which lies directly upon the dielectric, centered along the midline of the structure. This system is translationally symmetric along its length in terms of geometry and electrostatic behavior, and possesses mirror symmetry about its midline; it is modeled and analyzed based on half of its two-dimensional cross section. The electrostatic potential ϕ is governed by

$$-\nabla \cdot (\underline{\epsilon}_r \nabla \phi) = 0. \tag{9.111}$$

The resulting simplified two-dimensional domain Ω, as illustrated in Figure 9.9, is defined using two subdomains. The implicit functions

$$\tilde{\Phi}_1(\bar{x}) = \max\{\max[\max(-x_2 + a, x_1 - b), x_2 - b], -x_1\}, \tag{9.112}$$

$$\tilde{\Phi}_2(\bar{x}) = \max(x_2 - a - c, x_1 - a), \tag{9.113}$$

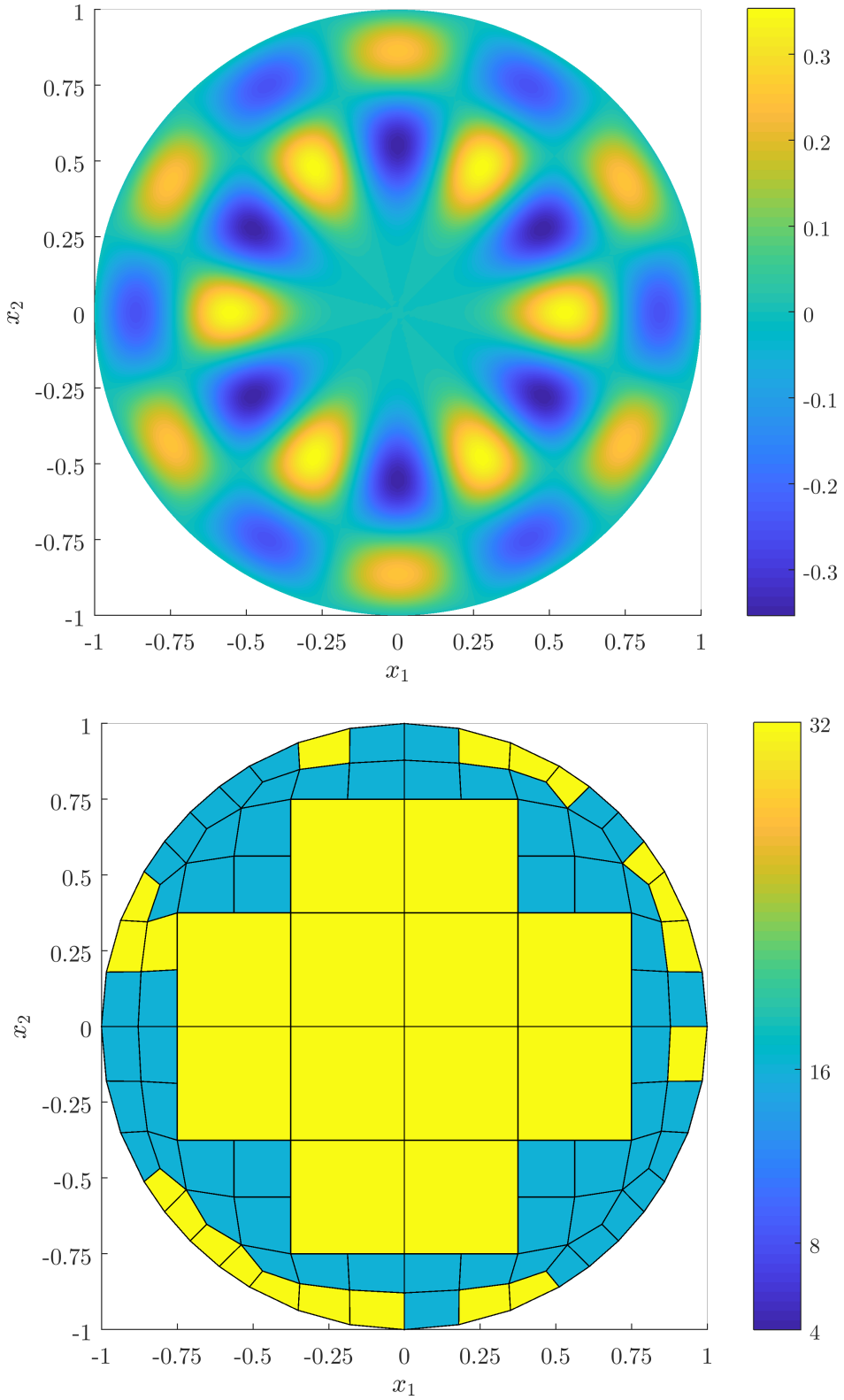

Figure 9.7: (top) Approximate solution ϕ to Poisson's equation (9.108) with zero Dirichlet boundary conditions. (bottom) Polynomial degree $L_{j,u_1} = L_{j,u_2}$ of the basis functions used to represent ϕ on each element after five iterations of *hp*-adaption.

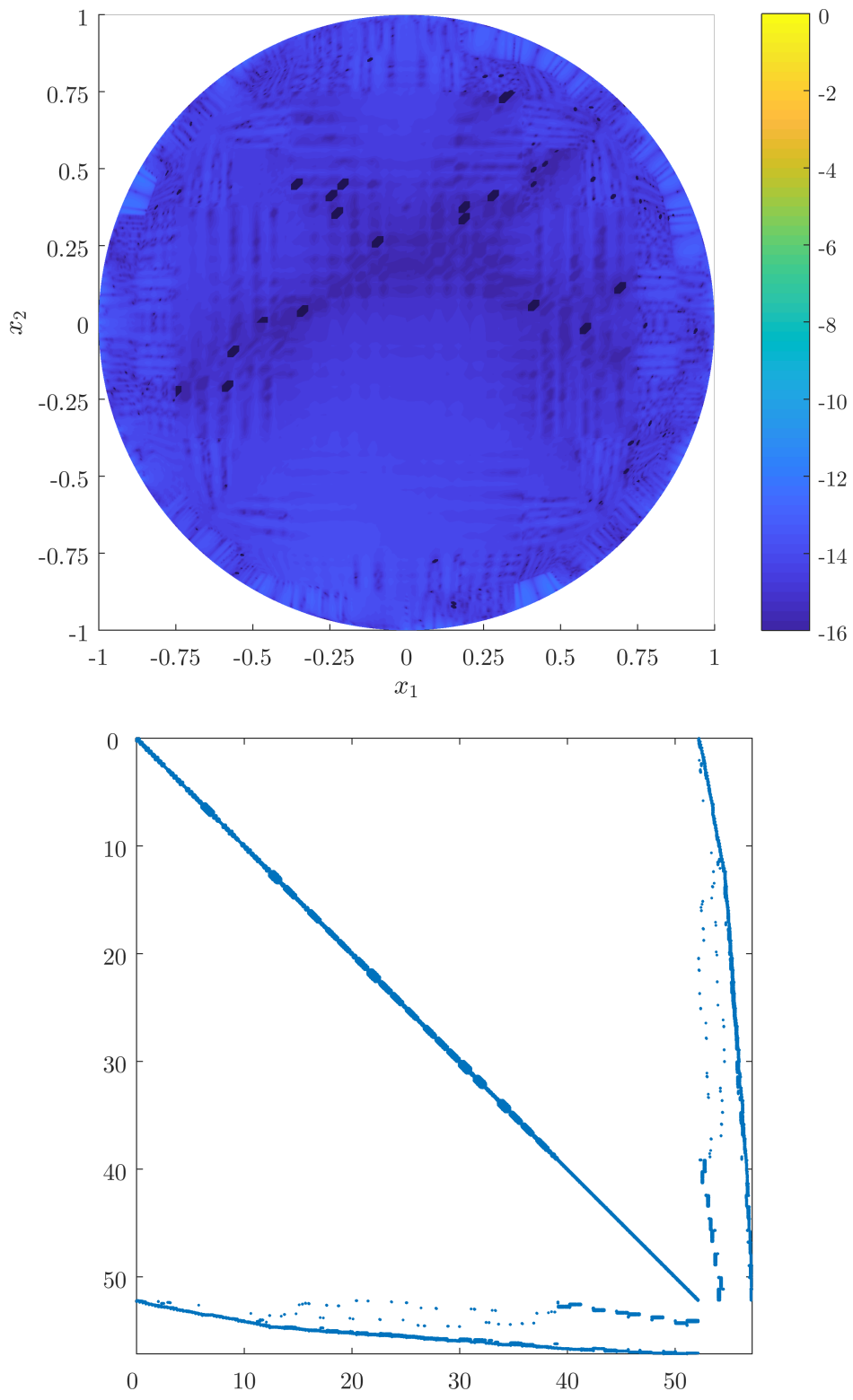

Figure 9.8: (top) Pointwise error $\log_{10} |\phi - \phi_{\text{exact}}|$ of the approximate solution in Figure 9.7. (bottom) Sparsity pattern for the saddle point matrix used to compute the approximate solution. The scales of the row number and column number axes are in thousands.

are used to define the first subdomain $\hat{\Omega}_1$ described by implicit function

$$\Phi_1(\bar{x}) = \max(\tilde{\Phi}_1(\bar{x}), -\tilde{\Phi}_2(\bar{x})). \tag{9.114}$$

The second subdomain $\hat{\Omega}_2$ is described using the implicit function

$$\Phi_2(\bar{x}) = \max\{\max[\max(-x_2, x_1 - b), x_2 - a], -x_1\} \tag{9.115}$$

and corresponds to the dielectric substrate. To compute an initial non-conforming mesh, we use a global bounding box with parameters

$$\bar{x}_{\text{bound}} = \frac{b}{2^{l_{\max}}}\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \qquad H = b + 2\frac{b}{2^{l_{\max}}}, \tag{9.116}$$

with maximum level of refinement $l_{\max} = 7$ and minimum level of refinement $l_{\min} = 0$. The maximum level of refinement is chosen so that the mesh spacing is fine enough to resolve the inner conductor of the microstrip line (limited by dimension c). Since the geometry has corners, we provide a list of fixed vertices that must appear in the mesh. The fixed vertices are

$$\{\bar{x}_{\text{fixed},k}\} = \left\{\begin{bmatrix} b \\ b \end{bmatrix}, \begin{bmatrix} 0 \\ b \end{bmatrix}, \begin{bmatrix} 0 \\ a+c \end{bmatrix}, \begin{bmatrix} a \\ a+c \end{bmatrix}, \begin{bmatrix} a \\ a \end{bmatrix}, \begin{bmatrix} b \\ a \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} b \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ a \end{bmatrix}\right\}. \tag{9.117}$$

The boundary projections for the problem are computed to a tolerance $\epsilon_{\text{tol}} = 10^{-12}$ and two iterations of smoothing and mesh optimization are performed.

For our problem, in subdomain $\hat{\Omega}_1$ we have $\underline{\epsilon}_{r,1} = \underline{I}$ whereas in subdomain $\hat{\Omega}_2$ we have $\underline{\epsilon}_{r,2} = 10\underline{I}$. Since (9.111) is homogeneous, we can ignore the scale factor ϵ_0. In the notation of our generic PDE, $\underline{\alpha} = \underline{\epsilon}_r$, $\beta = 0$, and $f = 0$. We impose a homogeneous Neumann boundary condition on

$$\Gamma_R = \left\{\bar{x} \in \mathbb{R}^2 : x_1 = 0\right\} \tag{9.118}$$

with parameters $\gamma = q = 0$. We do this because $x_1 = 0$ is a plane of symmetry. Similarly, we impose a homogeneous Dirichlet boundary condition on

$$\Gamma_{D,1} = \left\{\bar{x} \in \mathbb{R}^2 : x_1 = b \text{ or } x_2 = 0 \text{ or } x_2 = b\right\} \tag{9.119}$$

where $p_1 = 0$ and an inhomogeneous Dirichlet boundary condition on

$$\Gamma_{D,2} = \left\{\bar{x} \in \mathbb{R}^2 : 0 < x_1 < a \text{ and } x_2 = a \text{ or } 0 < x_1 < a \text{ and } x_2 = a + c\right\} \tag{9.120}$$

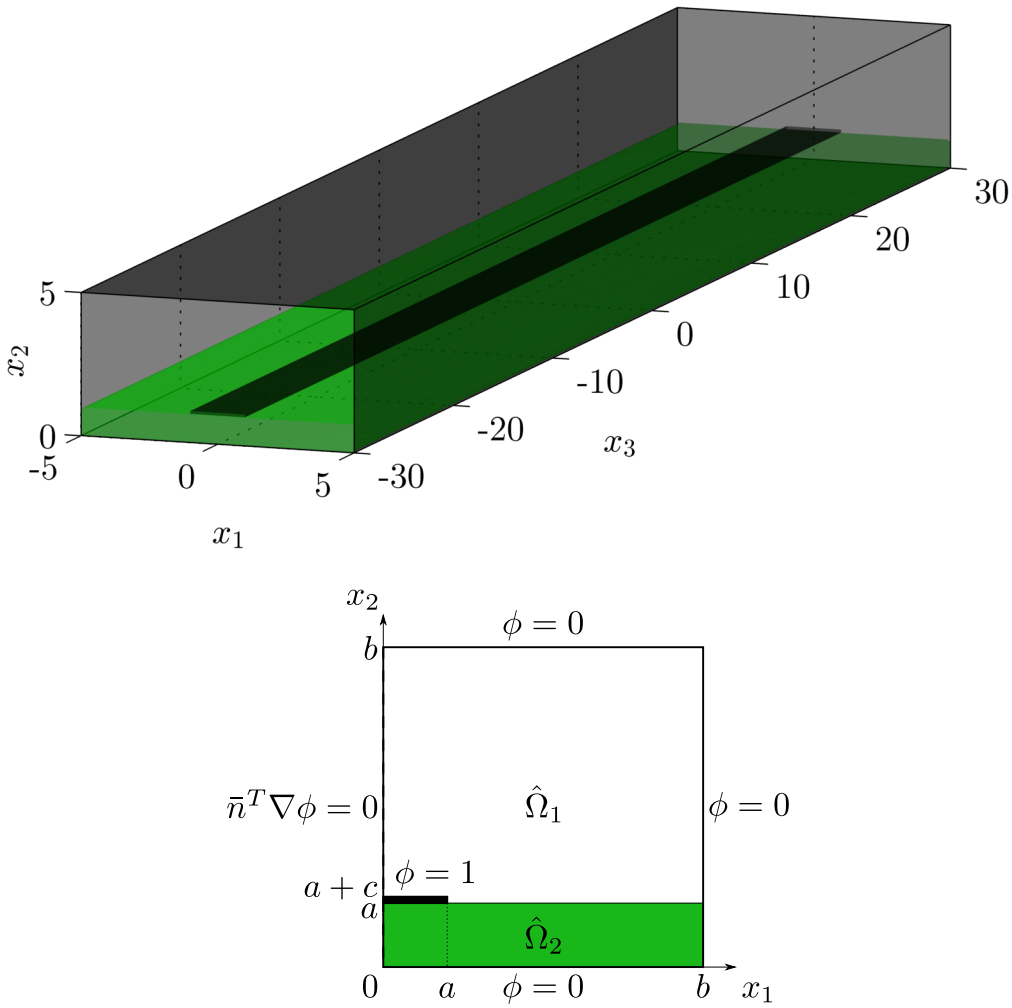

Figure 9.9: (top) Three-dimensional geometry of the shielded microstrip line. The line is infinite in the x_3 dimension but has been truncated for visualization. (bottom) Two-dimensional cross section of the shielded microstrip line exploiting mirror symmetry about the $x_1 = 0$ plane.

where $p_2 = 1$. The first Dirichlet boundary condition grounds the electrostatic potential ϕ on the outer conductor (we use reference potential zero) while the second Dirichlet boundary condition sets the inner conductor to a potential of one volt.

To compute an approximate solution, we perform five iterations of *hp*-adaption using the fixed fraction refinement criterion in Section 9.4 (we refine one third of all elements at each iteration). The elements begin with polynomial degree $L_{j,u_1} = L_{j,u_2} = 4$ basis functions. Figure 9.10 illustrates the computed solution using the finite element method described in this chapter, as well as the polynomial degree $L_{j,u_1} = L_{j,u_2}$ used on each element to represent ϕ. The mesh and solution were computed using the same specified tolerance $\epsilon_{\text{tol}} = 10^{-12}$. Notice that the *hp*-adaptive algorithm prioritizes *p*-adaption away from the

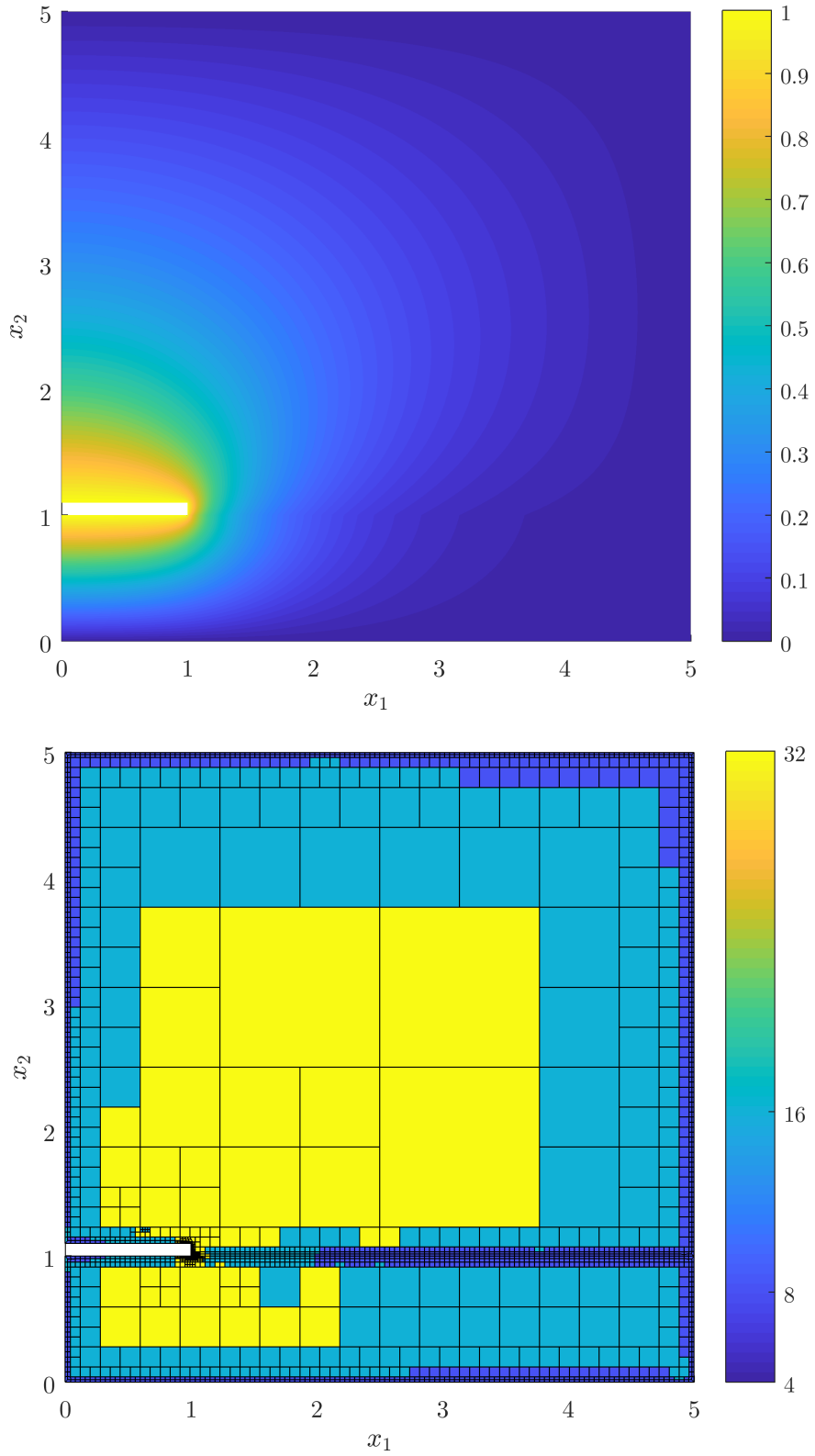

Figure 9.10: (top) Approximate solution ϕ to the variable coefficient Laplace equation for the shielded microstrip line. (bottom) Polynomial degree $L_{j,u_1} = L_{j,u_2}$ of the basis functions used to represent ϕ on each element after five iterations of *hp*-adaption.

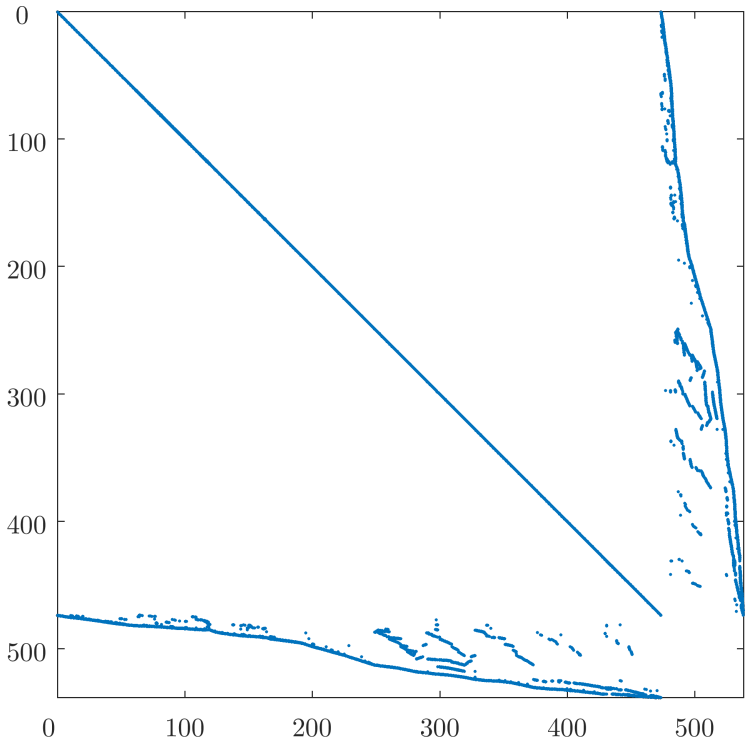

Figure 9.11: Sparsity pattern for the saddle point matrix used to compute the approximate solution to the shielded microstrip line problem. The scales of the row number and column number axes are in thousands.

re-entrant corner of the inner conductor and favors h-adaption near the re-entrant corner. The corner corresponds to a region where the solution varies less smoothly. This example illustrates a mesh with a variety of non-conforming edge configurations. The solution in Figure 9.10 shows that continuity is correctly imposed along those non-conforming edges.

In addition, Figure 9.11 shows the sparsity pattern of the associated saddle point system used to compute ϕ. The final saddle point system describes 538,365 linear equations. Note that the constraint matrix can become quite complicated, but that the systematic procedure described in Section 9.3 ensures that this matrix has full rank so that the saddle point system has a unique solution.

9.5.3 Coaxial Waveguide Discontinuity

As a third example, we solve for the electrostatic potential ϕ in an axisymmetric geometry whose schematic representation is given on page 102 of [12]. Figure 9.12 reproduces this geometry, illustrating a portion of two coaxial waveguides with identical outer conductors but different inner conductor radii joined together. The inner conductors are solid (not hollow). This problem is axisymmetric about its central axis in terms of geometry and electrostatic

behavior; it is modeled and analyzed based on half of its two-dimensional cut-section. The electrostatic potential ϕ is governed by

$$-\nabla \cdot (\underline{\epsilon}_r \nabla \phi) = 0 \tag{9.121}$$

with all derivatives expressed in cylindrical coordinates, where x_1 and x_2 represent the axial and radial coordinates respectively.

To specify the cut-section model, the domain Ω, as illustrated in Figure 9.12, is implicitly defined by taking the set difference of two regions. These regions are

$$\tilde{\Omega}_1 = \left\{ \bar{x} \in \mathbb{R}^2 : 0 < x_1 < b,\, c < x_2 < a + c \right\}, \tag{9.122}$$

$$\tilde{\Omega}_2 = \left\{ \bar{x} \in \mathbb{R}^2 : x_1 > b/2,\, x_2 < a \right\}, \tag{9.123}$$

respectively, with lengths $a = 2$, $b = 5$, and $c = 1$. Then $\Omega = \tilde{\Omega}_1 \backslash \tilde{\Omega}_2$. The corresponding implicit functions for these regions are

$$\tilde{\Phi}_1(\bar{x}) = \max\left\{ \max\left[\max\left(-x_1, x_1 - b\right), c - x_2 \right], x_2 - a - c \right\}, \tag{9.124}$$

$$\tilde{\Phi}_2(\bar{x}) = \max\left(b/2 - x_1, x_2 - a\right), \tag{9.125}$$

so that the implicit function describing $\Omega = \hat{\Omega}_1$ is

$$\Phi_1(\bar{x}) = \max(\tilde{\Phi}_1(\bar{x}), -\tilde{\Phi}_2(\bar{x})) \tag{9.126}$$

which we use to generate the mesh. Since the geometry has corners, we provide a list of fixed vertices

$$\{\bar{x}_{\text{fixed},k}\} = \left\{ \begin{bmatrix} 0 \\ c \end{bmatrix}, \begin{bmatrix} b/2 \\ c \end{bmatrix}, \begin{bmatrix} b/2 \\ a \end{bmatrix}, \begin{bmatrix} b \\ a \end{bmatrix}, \begin{bmatrix} b \\ a + c \end{bmatrix}, \begin{bmatrix} 0 \\ a + c \end{bmatrix} \right\}. \tag{9.127}$$

We use a bounding box with parameters

$$\bar{x}_{\text{bound}} = \begin{bmatrix} -1.1 \\ -1.1 \end{bmatrix}, \qquad H = 8, \tag{9.128}$$

and a maximum level of refinement $l_{\max} = 6$ with minimum level of refinement $l_{\min} = 0$. Projections are computed to a tolerance $\epsilon_{\text{tol}} = 10^{-12}$ and two iterations of smoothing and mesh optimization are performed.

For our problem, we choose $\underline{\epsilon}_r = 10\underline{I}$ which we treat as a scalar (as in the previous

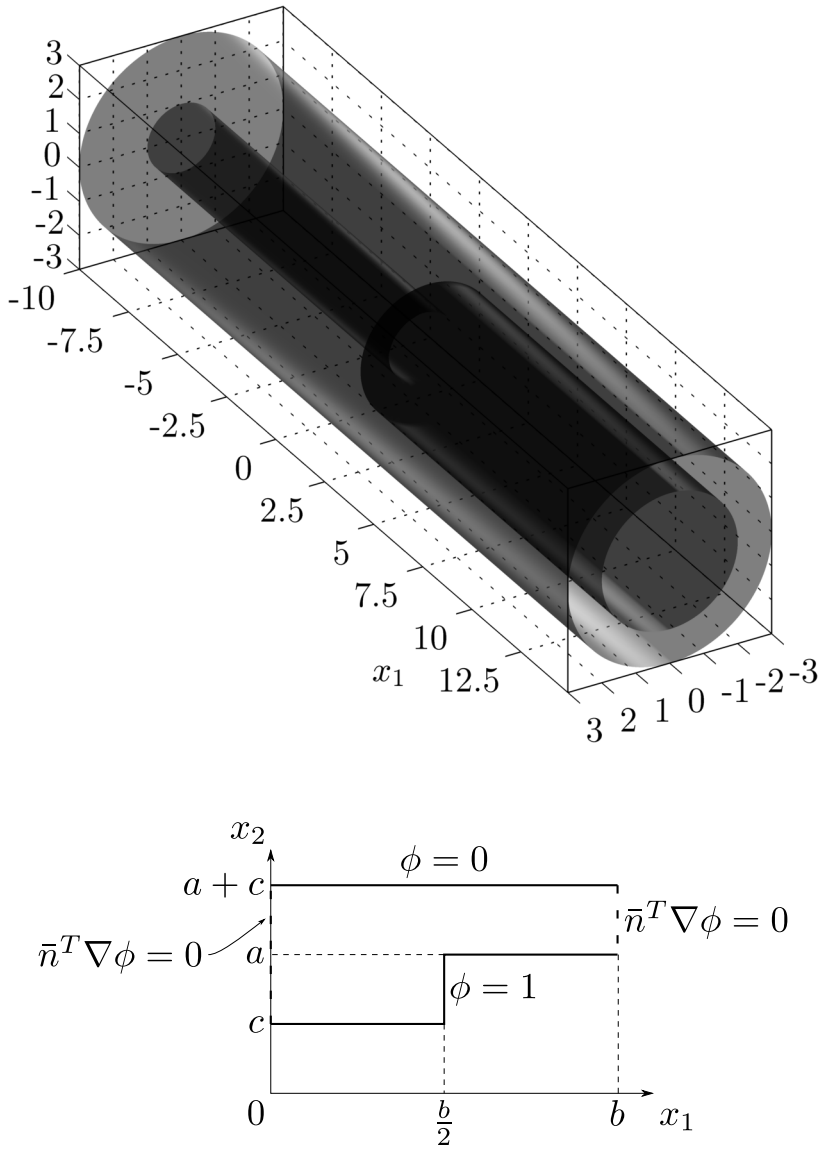

Figure 9.12: (top) Three-dimensional geometry of the coaxial waveguide problem. The line is infinite in the x_1 dimension but has been truncated for visualization. (bottom) Two-dimensional cross section in the axial x_1 and radial x_2 plane. Axisymmetric symmetry is exploited and the domain is truncated using homogeneous Neumann boundary conditions away from the discontinuity.

example, there is no need to include the permittivity of free space in our calculations). In the notation of our generic PDE, $\underline{\alpha} = x_2\underline{\epsilon}_r$, $\beta = 0$, and $f = 0$ (note that $\underline{\alpha}$ varies spatially). We impose homogeneous Neumann boundary conditions on

$$\Gamma_R = \left\{\bar{x} \in \mathbb{R}^2 : x_1 = 0 \cup x_1 = b\right\} \tag{9.129}$$

with parameters $\gamma = q = 0$. We do this to approximate the fact that far enough away from the discontinuity of the waveguide, the solution should not vary substantially in the axial direction. Similarly, we impose a homogeneous Dirichlet boundary condition on

$$\Gamma_{D,1} = \left\{\bar{x} \in \mathbb{R}^2 : x_2 = a + c\right\} \tag{9.130}$$

where $p_1 = 0$ and an inhomogeneous Dirichlet boundary condition on

$$\Gamma_{D,2} = \left\{\bar{x} \in \mathbb{R}^2 : 0 < x_1 < b/2,\, x_2 = c \cup x_1 = b/2,\, c < x_2 < a \cup b/2 < x_1 < b,\, x_2 = a\right\} \tag{9.131}$$

where $p_2 = 1$. The first Dirichlet boundary condition grounds the electrostatic potential ϕ on the outer conductor to zero and the second sets the inner conductor to a potential of one volt.

To compute an approximate solution, we perform five iterations of *hp*-adaption using the fixed fraction refinement criterion as described in Section 9.4. The elements begin with polynomial degree $L_{j,u_1} = L_{j,u_2} = 4$ basis functions. Figure 9.13 illustrates the computed solution using the finite element method described in this chapter, as well as the polynomial degree $L_{j,u_1} = L_{j,u_2}$ used on each element to represent ϕ. The mesh and solution were computed using the same specified tolerance $\epsilon_{\text{tol}} = 10^{-12}$. Note that, like the shielded microstrip line example, *h*-adaption is preferred near the re-entrant corner whereas *p*-adaption dominates away from the corner where the solution varies smoothly. In addition, Figure 9.14 illustrates the sparsity pattern of the associated saddle point system used to compute ϕ. The saddle point system describes 65,048 linear equations.

9.6 Unbounded Time-Harmonic Scattering Example

In the final part of this chapter, we address time-harmonic electromagnetic scattering from perfect electric conductors. The finite element method described in this chapter solves problems on bounded domains. To extend methods that compute solutions on bounded domains to approximate solutions on unbounded domains, two common approaches are typically used [12]. The first type of approach imposes absorbing boundary conditions at

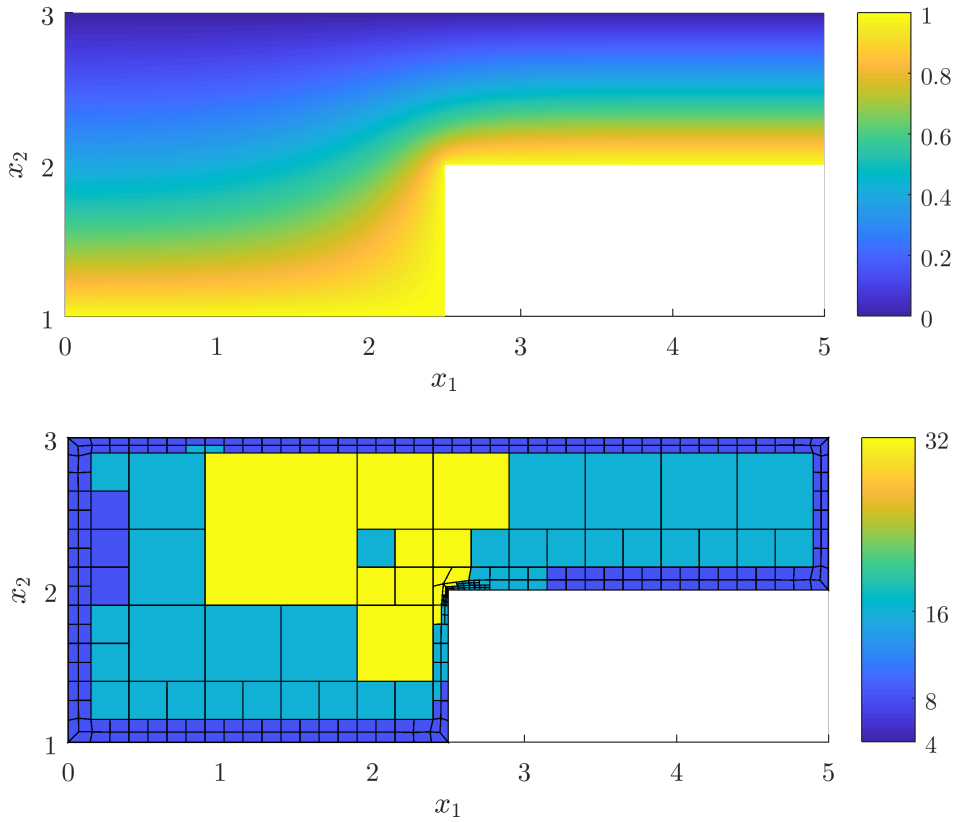

Figure 9.13: (top) Approximate solution ϕ to the variable coefficient Poisson equation for the waveguide discontinuity problem which describes the electromagnetic behavior within two mismatched coaxial waveguides in the vicinity of their junction interface. (bottom) Polynomial degree $L_{j,u_1} = L_{j,u_2}$ of the basis functions used to represent ϕ on each element after five iterations of *hp*-adaption.

a fictitious surface that bounds the region of interest. The second type, rather than use a fictitious surface, adds a fictitious layer. Among the many possible layer techniques, Perfectly Matched Layers (PML) have received widespread attention [86]. We choose to use a PML formulation in this thesis because introducing absorbing boundary conditions requires that we modify the boundary conditions, which leads to additional terms in the local operator matrices of elements adjacent to the boundary whereas, as we will show, the PML can be imposed by modifying the coefficient $\underline{\alpha}$ for a band of boundary adjacent elements. In the latter case, the boundary condition imposed on the fictitious boundary can be a simple homogeneous boundary condition of the types we have already seen.

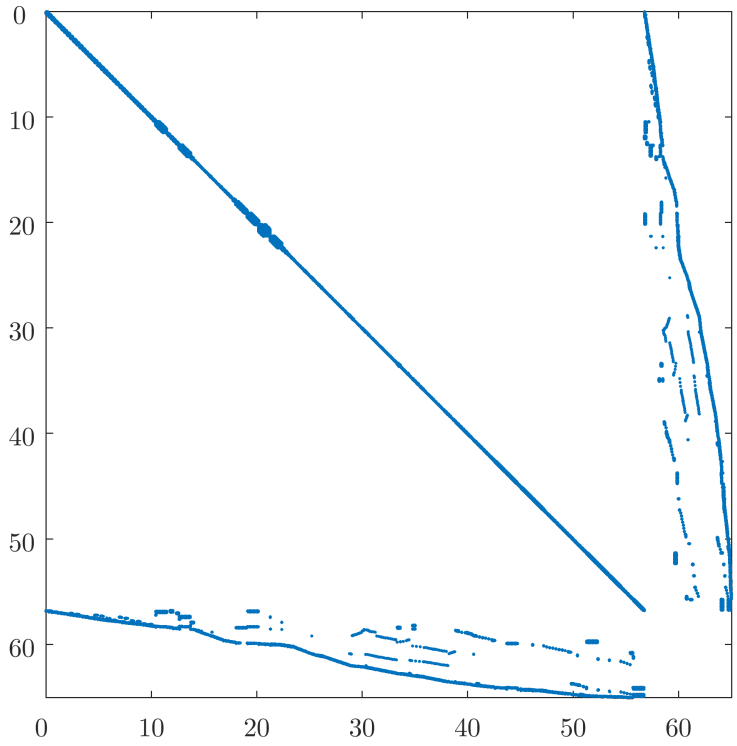

Figure 9.14: Sparsity pattern for the saddle point matrix used to compute the approximate solution to the waveguide discontinuity problem. The scales of the row number and column number axes are in thousands.

9.6.1 Perfectly Matched Layers

For now, we consider a one-dimensional PML description as in [87] to motivate the complex coordinate stretching interpretation of PML. We consider a sinusoidal traveling wave

$$v(z,t) = \cos(\omega t - kz) \tag{9.132}$$

$$= \Re\{e^{-\jmath kz} e^{\jmath \omega t}\} \tag{9.133}$$

and work only with its phasor representation $V(z) = e^{-\jmath kz}$ (this wave travels in the positive z direction) where $\jmath$ denotes the imaginary unit. We then consider performing the transformation

$$z = \tilde{z} - \jmath f(\tilde{z}) \tag{9.134}$$

where $f(\tilde{z}) > 0$ and both f and $\tilde{z}$ are real valued. This gives

$$V(\tilde{z}) = e^{-\jmath k(\tilde{z} - \jmath f(\tilde{z}))} \tag{9.135}$$

$$= e^{-\jmath k\tilde{z}} e^{-kf(\tilde{z})} \tag{9.136}$$

which contains an oscillatory factor $e^{-\jmath k\tilde{z}}$ of the same form as $V(z)$, along with an exponential damping factor $e^{-kf(\tilde{z})}$. Thus, under this type of transformation, a purely oscillatory traveling wave becomes a damped oscillatory wave

$$v(\tilde{z},t) = e^{-kf(\tilde{z})}\cos(\omega t - k\tilde{z}). \tag{9.137}$$

In particular, if we choose the damping function to be linear, i.e., $f(\tilde{z}) = a\tilde{z}$, with a damping factor $a = \sigma_z/k$, we obtain a frequency independent exponential decay

$$v(\tilde{z},t) = e^{-\sigma_z\tilde{z}}\cos(\omega t - k\tilde{z}). \tag{9.138}$$

The amplitude of σ_z determines how rapidly the wave decays as it travels in the positive $\tilde{z}$ direction. If the wave is traveling in the negative z direction, we apply the same transformation. This results in an exponential factor of $e^{\sigma_z\tilde{z}}$ multiplying the oscillatory factor $e^{jk\tilde{z}}$. While counterintuitive, this is still exponential damping, not growth, because the corresponding wave travels in the negative direction. Note that if the function f vanishes for certain values of $\tilde{z}$, then—for those values of $\tilde{z}$—the purely oscillatory function $V(z)$ remains purely oscillatory (the transformation is an identity map). Thus, we can simulate unbounded wave problems by performing a transformation with $f(\tilde{z}) = 0$ in regions where we are interested in the wave-like behavior of solutions, and $f(\tilde{z}) > 0$ in layers adjacent to this region to have the waves exponentially attenuate. At the edge of these layers, we can impose any boundary conditions deemed appropriate as they will effectively be acting on waves of zero amplitude.

The macroscopic Faraday and Ampère laws for media satisfying the linear constitutive relations $\bar{D} = \underline{\epsilon}\bar{E}$ and $\bar{B} = \underline{\mu}\bar{H}$ are

$$\nabla\times\bar{E} = -\frac{\partial\bar{B}}{\partial t}, \tag{9.139}$$

$$\nabla\times\bar{H} = \bar{J} + \frac{\partial\bar{D}}{\partial t}, \tag{9.140}$$

where $\bar{E}$ is the electric field intensity, $\bar{H}$ is the magnetic field intensity, $\bar{D}$ is the electric flux density, $\bar{B}$ is the magnetic flux density, and $\bar{J}$ is the current density. If we set the current density to zero and choose time-harmonic solutions

$$\bar{E}(\bar{x},t) = \Re\{\bar{E}(\bar{x})\,e^{\jmath\omega t}\}, \tag{9.141}$$

$$\bar{H}(\bar{x},t) = \Re\{\bar{H}(\bar{x})\,e^{\jmath\omega t}\}, \tag{9.142}$$

we obtain the time-harmonic equations

$$\nabla \times \bar{E} = -\jmath\omega\underline{\mu}\bar{H}, \tag{9.143}$$

$$\nabla \times \bar{H} = \quad \jmath\omega\underline{\epsilon}\bar{E}, \tag{9.144}$$

which exhibit wave-like solutions in unbounded domains[6]. To include PML for the two curl equations, we perform one-dimensional PML transformations in each of the three spatial variables x_1, x_2, and x_3. That is, we let

$$x_i = \tilde{x}_i - \jmath f_i(\tilde{x}_i) \tag{9.145}$$

where each function f_i is real and nonnegative and $i = 1, 2, 3$. To treat the curl operators, we also need to determine how partial derivatives transform. We apply the chain rule to a generic scalar function $g(\bar{x})$ to obtain

$$\frac{\partial g}{\partial \tilde{x}_i} = \sum_{j=1}^{3} \frac{\partial g}{\partial x_j}\frac{\partial x_j}{\partial \tilde{x}_i}. \tag{9.146}$$

Using the transformations (9.145) gives

$$\frac{\partial g}{\partial \tilde{x}_i} = \sum_{j=1}^{3} \frac{\partial g}{\partial x_j}\frac{\partial}{\partial \tilde{x}_i}\left[\tilde{x}_j - \jmath f_j(\tilde{x}_j)\right] \tag{9.147}$$

$$= \frac{\partial g}{\partial x_i}\left[1 - \jmath f_i'(\tilde{x}_i)\right]. \tag{9.148}$$

Thus, partial derivatives in physical space $\bar{x}$ transform like

$$\frac{\partial g}{\partial x_i} = \frac{1}{1 - \jmath f_i'(\tilde{x}_i)}\frac{\partial g}{\partial \tilde{x}_i}. \tag{9.149}$$

Let $s_i = 1 - \jmath f_i'(\tilde{x}_i)$. Then the curl operator transforms from

$$\nabla \times \bar{E}(\bar{x}) = \begin{bmatrix} 0 & -\dfrac{\partial}{\partial x_3} & \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_3} & 0 & -\dfrac{\partial}{\partial x_1} \\ -\dfrac{\partial}{\partial x_2} & \dfrac{\partial}{\partial x_1} & 0 \end{bmatrix} \begin{bmatrix} E_{x_1}(\bar{x}) \\ E_{x_2}(\bar{x}) \\ E_{x_3}(\bar{x}) \end{bmatrix} \tag{9.150}$$

[6]We often leave out the explicit dependence on space and/or time for $\bar{E}(\bar{x}, t)$ or $\bar{H}(\bar{x}, t)$ (respectively $\bar{E}(\bar{x})$ or $\bar{H}(\bar{x})$) when the context is clear.

to

$$\tilde{\nabla}_s \times \bar{E}\left(\tilde{x}\right) = \begin{bmatrix} 0 & -\frac{1}{s_3}\frac{\partial}{\partial \tilde{x}_3} & \frac{1}{s_2}\frac{\partial}{\partial \tilde{x}_2} \\ \frac{1}{s_3}\frac{\partial}{\partial \tilde{x}_3} & 0 & -\frac{1}{s_1}\frac{\partial}{\partial \tilde{x}_1} \\ -\frac{1}{s_2}\frac{\partial}{\partial \tilde{x}_2} & \frac{1}{s_1}\frac{\partial}{\partial \tilde{x}_1} & 0 \end{bmatrix} \begin{bmatrix} E_{x_1}\left(\tilde{x}\right) \\ E_{x_2}\left(\tilde{x}\right) \\ E_{x_3}\left(\tilde{x}\right) \end{bmatrix}. \tag{9.151}$$

If we define the vector $\bar{s}$ with entries s_i, and the diagonal matrices

$$\underline{S} = \mathrm{diag}\left(\bar{s}\right), \qquad \underline{\Lambda} = \underline{S}^{-1} \det\left(\underline{S}\right) \underline{S}^{-1}, \tag{9.152}$$

then

$$\tilde{\nabla}_s \times \bar{E}\left(\bar{x}\right) = \underline{S}^{-1}\underline{\Lambda}^{-1}\tilde{\nabla} \times \left[\underline{S}\bar{E}\left(\bar{x}\right)\right]. \tag{9.153}$$

To confirm this fact, we multiply these matrices and operators. First,

$$\underline{S}^{-1}\underline{\Lambda}^{-1} = \begin{bmatrix} \frac{1}{s_1} & 0 & 0 \\ 0 & \frac{1}{s_2} & 0 \\ 0 & 0 & \frac{1}{s_3} \end{bmatrix} \begin{bmatrix} \frac{s_1}{s_2 s_3} & 0 & 0 \\ 0 & \frac{s_2}{s_1 s_3} & 0 \\ 0 & 0 & \frac{s_3}{s_1 s_2} \end{bmatrix} \tag{9.154}$$

$$= \begin{bmatrix} \frac{1}{s_2 s_3} & 0 & 0 \\ 0 & \frac{1}{s_1 s_3} & 0 \\ 0 & 0 & \frac{1}{s_1 s_2} \end{bmatrix}. \tag{9.155}$$

Then

$$\tilde{\nabla} \times \left[\underline{S}\bar{E}\left(\tilde{x}\right)\right] = \begin{bmatrix} 0 & -\frac{\partial}{\partial \tilde{x}_3} & \frac{\partial}{\partial \tilde{x}_2} \\ \frac{\partial}{\partial \tilde{x}_3} & 0 & -\frac{\partial}{\partial \tilde{x}_1} \\ -\frac{\partial}{\partial \tilde{x}_2} & \frac{\partial}{\partial \tilde{x}_1} & 0 \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} E_{x_1}\left(\tilde{x}\right) \\ E_{x_2}\left(\tilde{x}\right) \\ E_{x_3}\left(\tilde{x}\right) \end{bmatrix} \tag{9.156}$$

$$= \begin{bmatrix} 0 & -s_2\frac{\partial}{\partial \tilde{x}_3} & s_3\frac{\partial}{\partial \tilde{x}_2} \\ s_1\frac{\partial}{\partial \tilde{x}_3} & 0 & -s_3\frac{\partial}{\partial \tilde{x}_1} \\ -s_1\frac{\partial}{\partial \tilde{x}_2} & s_2\frac{\partial}{\partial \tilde{x}_1} & 0 \end{bmatrix} \begin{bmatrix} E_{x_1}\left(\tilde{x}\right) \\ E_{x_2}\left(\tilde{x}\right) \\ E_{x_3}\left(\tilde{x}\right) \end{bmatrix} \tag{9.157}$$

where we have used the fact that s_i is only a function of $\tilde{x}_i$ which allows the factors of s_i to commute with the partial differential operators. Finally, combining these results gives

$$\underline{S}^{-1}\underline{\Lambda}^{-1}\tilde{\nabla}\times\left[\underline{S}\bar{E}\left(\tilde{x}\right)\right]=\left[\begin{array}{ccc}\frac{1}{s_2s_3} & 0 & 0\\ 0 & \frac{1}{s_1s_3} & 0\\ 0 & 0 & \frac{1}{s_1s_2}\end{array}\right]\left[\begin{array}{ccc}0 & -s_2\frac{\partial}{\partial\tilde{x}_3} & s_3\frac{\partial}{\partial\tilde{x}_2}\\ s_1\frac{\partial}{\partial\tilde{x}_3} & 0 & -s_3\frac{\partial}{\partial\tilde{x}_1}\\ -s_1\frac{\partial}{\partial\tilde{x}_2} & s_2\frac{\partial}{\partial\tilde{x}_1} & 0\end{array}\right]\left[\begin{array}{c}E_{x_1}\left(\tilde{x}\right)\\ E_{x_2}\left(\tilde{x}\right)\\ E_{x_3}\left(\tilde{x}\right)\end{array}\right] \tag{9.158}$$

which yields

$$\underline{S}^{-1}\underline{\Lambda}^{-1}\tilde{\nabla}\times\left[\underline{S}\bar{E}\left(\tilde{x}\right)\right]=\left[\begin{array}{ccc}0 & -\frac{1}{s_3}\frac{\partial}{\partial\tilde{x}_3} & \frac{1}{s_2}\frac{\partial}{\partial\tilde{x}_2}\\ \frac{1}{s_3}\frac{\partial}{\partial\tilde{x}_3} & 0 & -\frac{1}{s_1}\frac{\partial}{\partial\tilde{x}_1}\\ -\frac{1}{s_2}\frac{\partial}{\partial\tilde{x}_2} & \frac{1}{s_1}\frac{\partial}{\partial\tilde{x}_1} & 0\end{array}\right]\left[\begin{array}{c}E_{x_1}\left(\tilde{x}\right)\\ E_{x_2}\left(\tilde{x}\right)\\ E_{x_3}\left(\tilde{x}\right)\end{array}\right]. \tag{9.159}$$

This last equation agrees with (9.151).

Now that we have shown (9.153), we substitute this identity into the transformed time-harmonic Faraday law

$$\tilde{\nabla}_s\times\bar{E}\left(\tilde{x}\right)=-\jmath\omega\underline{\mu}\bar{H}\left(\tilde{x}\right) \tag{9.160}$$

to obtain

$$\underline{S}^{-1}\underline{\Lambda}^{-1}\tilde{\nabla}\times\left[\underline{S}\bar{E}\left(\tilde{x}\right)\right]=-\jmath\omega\underline{\mu}\bar{H}\left(\tilde{x}\right) \tag{9.161}$$

$$\tilde{\nabla}\times\left[\underline{S}\bar{E}\left(\tilde{x}\right)\right]=-\jmath\omega\underline{\Lambda}\underline{S}\underline{\mu}\bar{H}\left(\tilde{x}\right). \tag{9.162}$$

If $\underline{S}$ and $\underline{\mu}$ commute (which is the case, for example, when $\underline{\mu}$ is diagonal), then

$$\tilde{\nabla}\times\underbrace{\left[\underline{S}\bar{E}\left(\tilde{x}\right)\right]}_{\bar{E}_{\mathrm{PML}}(\tilde{x})}=-\jmath\omega\underbrace{\left[\underline{\Lambda}\underline{\mu}\right]}_{\underline{\mu}_{\mathrm{PML}}}\underbrace{\left[\underline{S}\bar{H}\left(\tilde{x}\right)\right]}_{\bar{H}_{\mathrm{PML}}(\tilde{x})}. \tag{9.163}$$

A similar procedure applied to

$$\tilde{\nabla}_s\times\bar{H}\left(\tilde{x}\right)=\jmath\omega\underline{\epsilon}\bar{E}\left(\tilde{x}\right) \tag{9.164}$$

yields

$$\tilde{\nabla}\times\bar{H}_{\mathrm{PML}}\left(\tilde{x}\right)=\jmath\omega\underline{\epsilon}_{\mathrm{PML}}\bar{E}_{\mathrm{PML}}\left(\tilde{x}\right) \tag{9.165}$$

with $\underline{\epsilon}_{\text{PML}} = \underline{\Lambda}\underline{\epsilon}$. By performing these operations, we have rewritten Faraday and Ampère's laws using the PML coordinate transformations but in a form where the curl operators do not involve the transformation parameters s_i. In fact, any formulation capable of solving anisotropic Maxwell's equations with complex coefficients can incorporate PML by solving for the auxiliary fields $\bar{H}_{\text{PML}}$ and/or $\bar{E}_{\text{PML}}$ by modifying the permittivity and permeability tensors. Actually, in the region of interest, $\underline{S} = \underline{I}$ since $s_i = 1 - \jmath f_i'(\tilde{x}_i)$ (this is because f_i appearing in $x_i = \tilde{x}_i - \jmath f_i(\tilde{x}_i)$ is zero there, and by consequence, so is its derivative). This means that in the physical domain

$$\bar{H}_{\text{PML}}(\tilde{x}) = \underline{S}\bar{H}(\bar{x} + \jmath\bar{f}(\tilde{x})) \tag{9.166}$$

$$= \underline{I}\bar{H}(\bar{x}) \tag{9.167}$$

$$= \bar{H}(\bar{x}) \tag{9.168}$$

and similarly, $\bar{E}_{\text{PML}}(\tilde{x}) = \bar{E}(\bar{x})$. Thus there is no need to perform any transformation to compute the physical solution after solving the PML equations. Of course, the solution $\bar{H}_{\text{PML}}$ and/or $\bar{E}_{\text{PML}}$ is nonphysical wherever the functions $f_i(\tilde{x}_i)$ are nonzero.

9.6.2 Scattering from an Infinite Circular Cylinder

In this final section, we solve a two-dimensional electromagnetic scattering problem where we assume that the scatterer is infinitely long in the x_3 direction and that its unit normal $\bar{n}$ is orthogonal to this axis of translation. This means $\bar{E}$ and $\bar{H}$ are independent of x_3. By making such an assumption, the Faraday and Ampère laws decouple into two sets of three equations each rather than yield a set of six coupled equations. These two sets are called the TM_{x_3} and TE_{x_3} mode equations respectively. In this section, we choose to focus on the TE_{x_3} mode although an analogous development applies for the TM_{x_3} mode as well. In the TE_{x_3} mode, $\bar{E}$ and $\bar{H}$ satisfy

$$\bar{E}(\bar{x}) = \begin{bmatrix} E_{x_1}(x_1, x_2) \\ E_{x_2}(x_1, x_2) \\ 0 \end{bmatrix}, \qquad \bar{H}(\bar{x}) = \begin{bmatrix} 0 \\ 0 \\ H_{x_3}(x_1, x_2) \end{bmatrix}. \tag{9.169}$$

Then rewriting Ampère's law as

$$\underline{\epsilon}^{-1}\nabla \times \bar{H} = \jmath\omega\bar{E}, \tag{9.170}$$

taking the curl, and substituting Faraday's law yields

$$\nabla \times \left(\underline{\epsilon}^{-1}\nabla \times \bar{H}\right) = \omega^2\underline{\mu}\bar{H}. \tag{9.171}$$

Evaluating the inner curl first, we obtain

$$\nabla \times \bar{H} = \begin{bmatrix} 0 & -\dfrac{\partial}{\partial x_3} & \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_3} & 0 & -\dfrac{\partial}{\partial x_1} \\ -\dfrac{\partial}{\partial x_2} & \dfrac{\partial}{\partial x_1} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ H_{x_3}(x_1, x_2) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial H_{x_3}}{\partial x_2} \\ -\dfrac{\partial H_{x_3}}{\partial x_1} \\ 0 \end{bmatrix}. \tag{9.172}$$

Then, restricting $\underline{\epsilon}$ to be diagonal,

$$\underline{\epsilon}^{-1} \nabla \times \bar{H} = \begin{bmatrix} \dfrac{1}{\epsilon_{11}} \dfrac{\partial H_{x_3}}{\partial x_2} \\ -\dfrac{1}{\epsilon_{22}} \dfrac{\partial H_{x_3}}{\partial x_1} \\ 0 \end{bmatrix}. \tag{9.173}$$

Applying the second curl gives

$$\nabla \times \left(\underline{\epsilon}^{-1} \nabla \times \bar{H} \right) = \begin{bmatrix} 0 & -\dfrac{\partial}{\partial x_3} & \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_3} & 0 & -\dfrac{\partial}{\partial x_1} \\ -\dfrac{\partial}{\partial x_2} & \dfrac{\partial}{\partial x_1} & 0 \end{bmatrix} \begin{bmatrix} \dfrac{1}{\epsilon_{11}} \dfrac{\partial H_{x_3}}{\partial x_2} \\ -\dfrac{1}{\epsilon_{22}} \dfrac{\partial H_{x_3}}{\partial x_1} \\ 0 \end{bmatrix} \tag{9.174}$$

which yields

$$\nabla \times \left(\underline{\epsilon}^{-1} \nabla \times \bar{H} \right) = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{\partial}{\partial x_1} \left(\dfrac{1}{\epsilon_{22}} \dfrac{\partial H_{x_3}}{\partial x_1} \right) - \dfrac{\partial}{\partial x_2} \left(\dfrac{1}{\epsilon_{11}} \dfrac{\partial H_{x_3}}{\partial x_2} \right) \end{bmatrix} \tag{9.175}$$

since the vector $\underline{\epsilon}^{-1} \nabla \times \bar{H}$ is independent of x_3. As a consequence, for the TE_{x_3} mode, $\bar{H}$ satisfies

$$-\frac{\partial}{\partial x_1} \left(\frac{1}{\epsilon_{22}} \frac{\partial H_{x_3}}{\partial x_1} \right) - \frac{\partial}{\partial x_2} \left(\frac{1}{\epsilon_{11}} \frac{\partial H_{x_3}}{\partial x_2} \right) = \omega^2 \mu_{33} H_{x_3} \tag{9.176}$$

and an identical result holds when we replace H_{x_3}, ϵ_{11}, ϵ_{22}, and μ_{33} with their corresponding PML counterparts $\bar{H}_{\mathrm{PML}}$, $\underline{\epsilon}_{\mathrm{PML}}$, and $\underline{\mu}_{\mathrm{PML}}$.

Using a normalization similar to the one-dimensional case described in Section 5.3, we

substitute

$$\hat{t} = \frac{1}{L\sqrt{\epsilon_0\mu_0}}t, \qquad \hat{x} = \frac{1}{L}\bar{x}, \qquad \hat{H} = \frac{1}{H_{\text{norm}}}\bar{H}, \qquad \hat{E} = \frac{1}{H_{\text{norm}}}\sqrt{\frac{\epsilon_0}{\mu_0}}\bar{E}, \tag{9.177}$$

and corresponding angular frequency $\hat{\omega} = L\sqrt{\epsilon_0\mu_0}\omega$ (a direct consequence of the change of variable from time t to $\hat{t}$) to render (9.176) dimensionless. Typically, we associate L with the wavelength of the phenomena, and H_{norm} with the peak amplitude of $\bar{H}$. Through this normalization, ϵ_{11}, ϵ_{22}, and μ_{33} become their corresponding components of the relative permittivity and permeability tensors and factors ϵ_0 and μ_0 from the absolute permittivity and permeability are scaled out (we drop the circumflex notation from now on and perform all computations in the dimensionless space).

We denote the scatterer's cross section in the x_1x_2-plane by Ω_s with boundary Γ_s and the surrounding space by $\Omega = \mathbb{R}^2\backslash\Omega_s$. Rather than solve for the total field H_{x_3} in Ω, we will assume a given incident field H_i and solve for the scattered field H_s such that $H_{x_3} = H_i + H_s$. For our example, we choose a plane wave in free space

$$H_i(\bar{x}) = e^{-\jmath\omega\bar{k}^T\bar{x}} \tag{9.178}$$

where

$$\bar{k} = \begin{bmatrix} k_{x_1} \\ k_{x_2} \end{bmatrix} = \begin{bmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{bmatrix} \tag{9.179}$$

is a unit vector describing the direction of propagation of the wave and φ_i is the angle of incidence measured from the x_1 axis. Since H_{x_3} satisfies (9.176) in Ω, we must solve the PDE

$$-\frac{\partial}{\partial x_1}\left(\frac{1}{\epsilon_{22}}\frac{\partial H_s}{\partial x_1}\right) - \frac{\partial}{\partial x_2}\left(\frac{1}{\epsilon_{11}}\frac{\partial H_s}{\partial x_2}\right) - \omega^2\mu_{33}H_s = f \tag{9.180}$$

where, restricting $\underline{\epsilon}$ to be independent of $\bar{x}$,

$$f = \frac{\partial}{\partial x_1}\left(\frac{1}{\epsilon_{22}}\frac{\partial H_i}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\frac{1}{\epsilon_{11}}\frac{\partial H_i}{\partial x_2}\right) + \omega^2\mu_{33}H_i \tag{9.181}$$

$$= \omega^2\left(-\frac{1}{\epsilon_{22}}k_{x_1}^2 - \frac{1}{\epsilon_{11}}k_{x_2}^2 + \mu_{33}\right)H_i. \tag{9.182}$$

Equation (9.180) can be rewritten in the form of the generic PDE

$$-\nabla\cdot(\underline{\alpha}\nabla\phi) + \beta\phi = f \tag{9.183}$$

with

$$\underline{\alpha} = \begin{bmatrix} \dfrac{1}{\epsilon_{22}} & 0 \\ 0 & \dfrac{1}{\epsilon_{11}} \end{bmatrix}, \qquad \beta = -\omega^2 \mu_{33}, \qquad f = -(\omega^2 \bar{k}^T \underline{\alpha} \bar{k} + \beta) H_i, \qquad \phi = H_s, \tag{9.184}$$

and solved using the techniques described in this chapter once we have specified the appropriate boundary conditions (this is an anisotropic form of Helmholtz's equation).

Since the scatterer is a perfect electric conductor, the appropriate boundary condition for the scatterer is

$$\bar{n} \times \bar{E} = 0 \qquad \text{on } \Gamma_s. \tag{9.185}$$

We must express this boundary condition in terms of $\bar{H}$ because (9.176) is written in terms of H_{x_3}. We do so by using (9.170) to replace $\bar{E}$ in the boundary condition to obtain

$$\bar{n} \times \left(\frac{1}{\jmath\omega} \underline{\epsilon}^{-1} \nabla \times \bar{H} \right) = 0 \qquad \text{on } \Gamma_s. \tag{9.186}$$

We ignore the factor $1/\jmath\omega$ since this boundary condition is homogeneous. By explicit computation, we find that

$$\bar{n} \times \left(\underline{\epsilon}^{-1} \nabla \times \bar{H} \right) = \begin{bmatrix} 0 & 0 & n_2 \\ 0 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \begin{bmatrix} \dfrac{1}{\epsilon_{11}} \dfrac{\partial H_{x_3}}{\partial x_2} \\ -\dfrac{1}{\epsilon_{22}} \dfrac{\partial H_{x_3}}{\partial x_1} \\ 0 \end{bmatrix} \tag{9.187}$$

simplifies to

$$\bar{n} \times \left(\underline{\epsilon}^{-1} \nabla \times \bar{H} \right) = -\left(n_1 \frac{1}{\epsilon_{22}} \frac{\partial H_{x_3}}{\partial x_1} + n_2 \frac{1}{\epsilon_{11}} \frac{\partial H_{x_3}}{\partial x_2} \right) \tag{9.188}$$

$$= -\bar{n}^T \left(\underline{\alpha} \nabla H_{x_3} \right) \tag{9.189}$$

so that the perfect electric boundary condition can be applied as a Robin boundary condition. Using the total field $H_{x_3} = H_i + H_s$ yields

$$\bar{n}^T \left(\underline{\alpha} \nabla H_{x_3} \right) = 0 \tag{9.190}$$

$$\bar{n}^T \left(\underline{\alpha} \nabla H_s \right) = -\bar{n}^T \left(\underline{\alpha} \nabla H_i \right) \tag{9.191}$$

$$= -\left(n_1 \frac{1}{\epsilon_{22}} \frac{\partial H_i}{\partial x_1} + n_2 \frac{1}{\epsilon_{11}} \frac{\partial H_i}{\partial x_2} \right). \tag{9.192}$$

Substituting the incident field gives

$$\bar{n}^T \left(\underline{\alpha} \nabla H_s\right) = \jmath\omega \left(n_1 \frac{1}{\epsilon_{22}} k_{x_1} + n_2 \frac{1}{\epsilon_{11}} k_{x_2}\right) H_i \tag{9.193}$$

$$= \jmath\omega \bar{n}^T \underline{\alpha} \bar{k} H_i \tag{9.194}$$

so that $\gamma = 0$ and $q = \jmath\omega \bar{n}^T \underline{\alpha} \bar{k} H_i$ on Γ_s.

Because we only consider time-harmonic solutions, there is an additional boundary condition on the scattered solution H_s needed at infinity for this PDE to be well-posed [29]. When $\underline{\epsilon} = \epsilon \underline{I}$ and $\underline{\mu} = \mu \underline{I}$, the Silver-Müller radiation condition

$$\lim_{\|\bar{x}\|_2 \to \infty} \|\bar{x}\|_2 \left[\nabla \times \bar{H}_s + \jmath\omega\sqrt{\mu\epsilon}\hat{x} \times \bar{H}_s\right] = 0 \tag{9.195}$$

must hold uniformly in all unit directions $\hat{x}$. This boundary condition removes the possibility of incoming waves which would not be physically relevant for a scattered wave (the scattered wave should originate at the scatterer Ω_s and propagate away from it, not towards it). However, since the finite element method described in this chapter holds only for bounded domains Ω, rather than attempt to impose such a boundary condition, we instead enclose the scatterer Ω_s with PML.

To fix ideas, we consider an incident wave with angle $\varphi_i = 0$ and angular frequency $\omega = 2\pi$ (corresponding to a normalized wavelength of 1) incident upon an infinitely long circular cylinder surrounded by free space (this means that nondimensional $\epsilon_{11} = \epsilon_{22} = \mu_{33} = 1$). The cylinder boundary Γ_s is the unit circle centered at the origin whose signed distance function is

$$\Phi_1\left(\bar{x}\right) = 1 - \|\bar{x}\|_2. \tag{9.196}$$

This function can be used to describe the domain

$$\Omega = \left\{\bar{x} \in \mathbb{R}^2 : \|\bar{x}\|_2 > 1\right\}. \tag{9.197}$$

To approximate this unbounded domain, we consider the truncated domain

$$\Omega_{\text{truncated}} = \Omega \cap \left\{\bar{x} \in \mathbb{R}^2 : \|\bar{x}\|_\infty < L_{\text{box}}\right\} \tag{9.198}$$

where $L_{\text{box}} = 5$ and construct a mesh with bounding box parameters

$$\bar{x}_{\text{bound}} = \begin{bmatrix} -L_{\text{box}} \\ -L_{\text{box}} \end{bmatrix}, \qquad H = 2L_{\text{box}}, \tag{9.199}$$

maximum level of refinement $l_{\max} = 5$, and minimum level of refinement $l_{\min} = 3$. During the meshing procedure we do not project boundary edges that lie on the boundary of the bounding box. Projections to Γ_s are computed to a tolerance $\epsilon_{\text{tol}} = 10^{-12}$. The domain $\Omega_{\text{truncated}}$ is an axis-aligned square with side lengths $2L_{\text{box}}$, centered at the origin, with the unit disk removed.

We then specify the PML functions

$$s_i = 1 - \jmath f_i'(x_i) \tag{9.200}$$

with

$$f_i'(x_i) = \begin{cases} \dfrac{\sigma_i}{\omega} & x_i > L_{\text{box}} - w_i \text{ or } x_i < -L_{\text{box}} + w_i \\ 0 & \text{otherwise} \end{cases} \tag{9.201}$$

where σ_i is a constant whose magnitude determines how scattered waves decay in the PML and w_i is the thickness of the PML. The factor ω in the denominator means that the attenuation in the PML is independent of frequency. For our example, we choose $w_i = 1.25$ which suggests that

$$\sigma_i = -\frac{\log(\epsilon_{\text{tol}})}{2w_i} \approx 12 \tag{9.202}$$

is a good choice to have the wave decay to the appropriate tolerance after traveling into and out of the PML. Using such PML results in four regions with different values for $\underline{\alpha}$, β, and f. The free space region has both $s_1 = 1$ and $s_2 = 1$, whereas region 2 has $s_1 = 1$ and $s_2 \neq 1$, region 3 has $s_1 \neq 1$ and $s_2 = 1$, and region 4 has both $s_1 \neq 1$ and $s_2 \neq 1$. In region 1, we set the forcing function to

$$f = -(\omega^2 \bar{k}^T \underline{\alpha} \bar{k} + \beta) H_i \tag{9.203}$$

as described earlier, with forcing functions in the other regions set to zero. The boundary condition on Γ_s is given by (9.194) and the boundary condition on $\partial\Omega_{\text{truncated}} \backslash \Gamma_s$ is a homogeneous Neumann boundary condition. We have chosen the PML thicknesses w_i, attenuation coefficients σ_i, and the PML's distance to the scatterer such that this boundary condition has a negligible effect on the scattered solution.

The exact solution to this scattering problem is known [4, 8]. Briefly, it is obtained by solving Helmholtz's equation in cylindrical coordinates via separation of variables which results in an expansion of Bessel functions in the radial direction, multiplied by sinusoids in the angular direction, for the scattered field. Imposing the radiation condition at infinity forces linear combinations of Bessel functions that yield Hankel functions of the second kind. By expanding the plane wave H_i in a cylindrical wave expansion, applying the boundary condition on the unit circle, and matching coefficients in the incident and scattered field, we

find that

$$H_{x_3} = \sum_{n=0}^{\infty} d_n \jmath^{-n} \left[J_n \left(\omega \|\bar{x}\|_2 \right) + c_n H_n^{(2)} \left(\omega \|\bar{x}\|_2 \right) \right] \cos \left(n \operatorname{atan2} \left(x_2, x_1 \right) \right) \tag{9.204}$$

where

$$d_n = \begin{cases} 2 & n > 0 \\ 1 & n = 0, \end{cases} \qquad c_n = -\frac{J_n' \left(\omega \right)}{H_n^{(2)\prime} \left(\omega \right)}, \tag{9.205}$$

and J_n is the Bessel function of the first kind of integer order n, $H_n^{(2)}$ is the Hankel function of the second kind of integer order n, and J_n' and $H_n^{(2)\prime}$ are their derivatives with respect to their arguments. Derivatives of these special functions are evaluated using

$$J_0' \left(z \right) = -J_1 \left(z \right), \tag{9.206}$$

$$H_0^{(2)\prime} \left(z \right) = -H_1^{(2)} \left(z \right), \tag{9.207}$$

and

$$C_n' \left(z \right) = \frac{1}{2} \left[C_{n-1} \left(z \right) - C_{n+1} \left(z \right) \right] \tag{9.208}$$

where C_n is either J_n or $H_n^{(2)}$, and z is a generic argument, as described in [162]. When used for comparison to the computed solution, we take a finite number of terms from the expansion. This expansion converges in a disk whose radius increases when increasing the number of terms taken from the series. In our experiments, we take the first 75 terms for a solution accurate to 13 digits or better throughout the domain $\Omega_{\text{truncated}}$.

Figure 9.15 illustrates the approximate solution $H_{x_3} = H_i + H_s$ as well as the mesh after five iterations of the *hp*-adaptive algorithm. Degree $L_{u_1} = L_{u_2} = 4$ polynomials were used on the initial mesh. In practice, we solve for $\phi = H_s$ and then add it to the known incident field H_i to get the total field represented in the figure. Note that in the initial mesh, there are one element thick PML surrounding the domain. In the approximate solution, along the interface between the PML and free space, one can observe the rapid decay of the scattered field H_s (whose magnitude is roughly one quarter the magnitude of the incident field at that interface) because, to the eye, H_{x_3} simply appears to be the incident field H_i in the PML. Since the factors s_i enter into the coefficients $\underline{\alpha}$ and β, the choice of PML function that we have made leads to this rapid variation in the computed field ϕ. A smooth transition between 0 and σ_i/ω can reduce this rapid variation. We elect to keep the PML transition sharp to demonstrate that high polynomial degree is sufficient to resolve the rapid decay.

Figure 9.16 illustrates the pointwise error $\log_{10} |H_{x_3} - H_{x_3,\text{exact}}|$ between the computed solution H_{x_3} and the exact solution $H_{x_3,\text{exact}}$ (sampled with 21 uniformly spaced points in

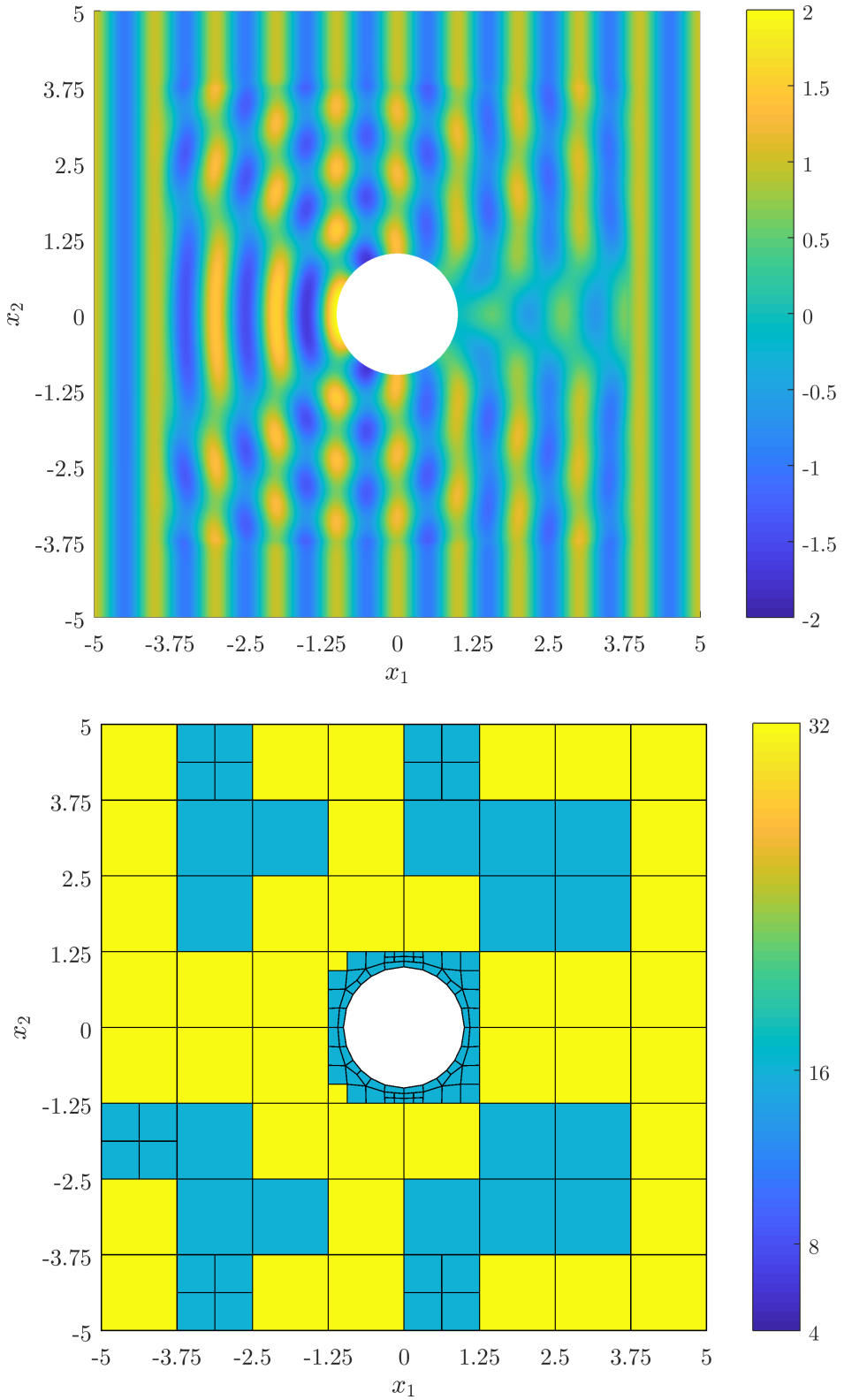

Figure 9.15: (top) Approximate solution $H_{x_3} = H_i + H_s$ to the scattering problem for the circular cylinder obtained by adding (9.178) to the solution of (9.183). (bottom) Mesh used to compute the solution after five iterations of *hp*-adaption.

each coordinate of the canonical domain $\bar{u} \in (-1,1)^2$ on each element), as well as the sparsity pattern of the associated saddle point system used to compute H_s. The solution is accurate to 9 digits or better throughout the free space region, but grossly incorrect inside the PML. This is expected because the PML attenuate the scattered field in those regions. Note that the error tends to be larger over elements with degree 16 than elements with degree 32. Subsequent iterations in the *hp*-adaptive algorithm identify this fact and reduce the error accordingly. There are 214 element refinements performed during the five iterations of *hp*-adaption, only 9 of which are of *h*-type. This is expected because the solution is smooth everywhere and in such situations, *p*-adaption is preferred. This adaptive process results in a final saddle point system describing 79,918 linear equations.

One frequently computed quantity of interest associated with scattering solutions is the radar cross section (RCS). In two dimensions, the RCS is given by

$$\sigma_{2\mathrm{D}}(\varphi, \varphi_i) = \lim_{\|\bar{x}\|_2 \to \infty} 2\pi \|\bar{x}\|_2 \frac{|H_s|^2}{|H_i|^2} \tag{9.209}$$

and is a function of the observation angle φ as well as the angle of incidence φ_i of the incident wave H_i [4]. Since our scatterer is invariant under rotations, there is no need to compute this quantity for various φ_i and we fix it to zero. A convenient way to calculate this quantity is through Green's representation formula for Helmholtz's equation (see [163] for a derivation and the relevant integral equation theory). The formula is

$$H_s(\bar{x}) = \oint_{\Gamma_s} \left(\bar{n}'^T \nabla' G_{2\mathrm{D}}(\bar{x}, \bar{x}')\right) H_s(\bar{x}') - \left(\bar{n}'^T \nabla' H_s(\bar{x}')\right) G_{2\mathrm{D}}(\bar{x}, \bar{x}')\, d\Omega' \tag{9.210}$$

for $\bar{x} \in \Omega$ where

$$G_{2\mathrm{D}}(\bar{x}, \bar{x}') = -\frac{\jmath}{4} H_0^{(2)}(\omega \|\bar{x} - \bar{x}'\|_2) \tag{9.211}$$

is the Green's function for Helmholtz's equation in two dimensions and integration is performed with respect to the primed coordinates (as are derivatives). Using the large argument asymptotic expression [164]

$$\lim_{z \to \infty} H_n^{(2)}(z) \approx \sqrt{\frac{2}{\pi z}} e^{-\jmath\left(z - n\frac{\pi}{2} - \frac{\pi}{4}\right)} \tag{9.212}$$

in Green's representation formula yields

$$\lim_{\|\bar{x}\|_2 \to \infty} H_s(\bar{x}) = \sqrt{\frac{\jmath\omega}{8\pi\|\bar{x}\|_2}} e^{-\jmath\omega\|\bar{x}\|_2} \underbrace{\oint_{\Gamma_s} \left[\hat{x}^T \bar{n}'\, H_s(\bar{x}') - \frac{1}{\jmath\omega}\left(\bar{n}'^T \nabla' H_s(\bar{x}')\right)\right] e^{\jmath\omega \hat{x}^T \bar{x}'} d\Omega'}_{H_{\mathrm{far}}} \tag{9.213}$$

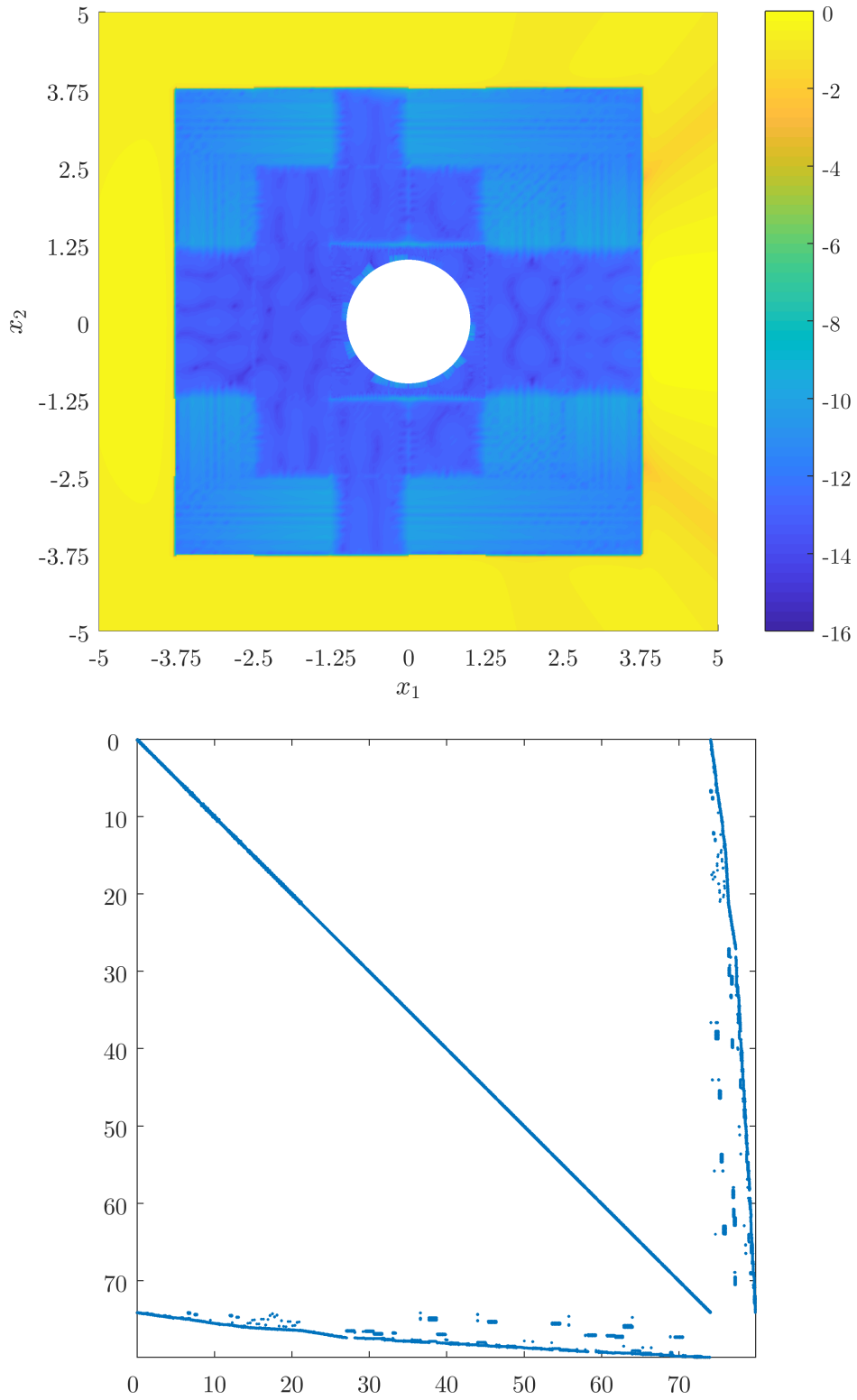

Figure 9.16: (top) Pointwise error $\log_{10}|H_{x_3} - H_{x_3,\text{exact}}|$ of the approximate solution in Figure 9.15. (bottom) Sparsity pattern for the saddle point matrix used to compute the approximate solution. The scales of the row number and column number axes are in thousands.

which gives the scattered far field in terms of the scattered near field. Taking the magnitude gives

$$|H_s|^2 = \sqrt{\frac{\jmath\omega}{8\pi\|\bar{x}\|_2}} e^{-\jmath\omega\|\bar{x}\|_2} H_{\text{far}} \left[\sqrt{\frac{-\jmath\omega}{8\pi\|\bar{x}\|_2}} e^{\jmath\omega\|\bar{x}\|_2} H_{\text{far}}^* \right] \tag{9.214}$$

$$= \frac{\omega}{8\pi\|\bar{x}\|_2} |H_{\text{far}}|^2. \tag{9.215}$$

Similarly, the magnitude of the incident field is

$$|H_i|^2 = e^{-\jmath\omega\bar{k}^T\bar{x}}[e^{\jmath\omega\bar{k}^T\bar{x}}] = 1 \tag{9.216}$$

which means that

$$\sigma_{\text{2D}}(\varphi, \varphi_i) = \lim_{\|\bar{x}\|_2 \to \infty} 2\pi\|\bar{x}\|_2 \left[\frac{\omega}{8\pi\|\bar{x}\|_2} |H_{\text{far}}|^2 \right] \tag{9.217}$$

$$= \frac{\omega}{4} |H_{\text{far}}|^2. \tag{9.218}$$

Thus, to calculate the RCS, we first compute H_s for a given incident wave (in our case with incidence angle $\varphi_i = 0$), then we compute the far field integral

$$H_{\text{far}}(\varphi) = \oint_{\Gamma_s} \left[\hat{x}^T \bar{n}' H_s(\bar{x}') - \frac{1}{\jmath\omega} \left(\bar{n}'^T \nabla' H_s(\bar{x}') \right) \right] e^{\jmath\omega\hat{x}^T\bar{x}'} d\Omega' \tag{9.219}$$

where

$$\hat{x} = \begin{bmatrix} \cos(\varphi) \\ \sin(\varphi) \end{bmatrix} \tag{9.220}$$

is the only quantity in the integrand that depends on φ. To calculate the RCS from the finite element solution we have already computed, we split the integral across the N_Γ elements whose curvilinear edges represent the scatterer:

$$H_{\text{far}}(\varphi) = \sum_{j=1}^{N_\Gamma} \int_{\Gamma_j} \left[\hat{x}^T \bar{n}' H_s(\bar{x}') - \frac{1}{\jmath\omega} \left(\bar{n}'^T \nabla' H_s(\bar{x}') \right) \right] e^{\jmath\omega\hat{x}^T\bar{x}'} d\Omega'. \tag{9.221}$$

Each of these integrals are parametrized along their respective edges using their respective element's transfinite interpolation map as described in Chapter 8. To illustrate, suppose that one of these edges is parametrized by

$$\bar{x}_1(-1, u_2) = \underline{\tilde{X}}_1 \bar{p}(u_2) \tag{9.222}$$

(see Chapter 8 for the other possible parametrizations). Then its corresponding integral can be expressed as

$$\int_{-1}^{1}\left[\hat{x}^T\bar{n}'\,H_s\left(\underline{\tilde{X}}_1\bar{p}\,(u_2)\right)-\frac{1}{\jmath\omega}\left(\bar{n}'^T\underline{J}_{\bar{x}'}^{-T}\nabla_{\bar{u}}H_s\left(\underline{\tilde{X}}_1\bar{p}\,(u_2)\right)\right)\right]e^{\jmath\omega\hat{x}^T\bar{x}'}\|\underline{\tilde{X}}_1\underline{\tilde{D}}\bar{p}\,(u_2)\,\|_2 du_2. \tag{9.223}$$

One-dimensional numerical integration can then be applied to resolve this integral. In practice, this integrand is smooth and high order Clenshaw-Curtis quadrature can be used to guarantee that the error in computing the integral is smaller than the error in the near field solution H_s [110].

To assess the accuracy of the computed RCS, we also compute the RCS $\sigma_{\text{2D,exact}}$ using the exact solution (9.204). Again, we only need to compute the integral for H_{far}. In this case, since the boundary Γ_s is smooth and the integrand for H_{far} is periodic (the integral is performed around a closed loop for a continuous function), we can use the Trapezoidal rule to obtain exponential convergence [165]. Figure 9.17 illustrates $10\log_{10}(\sigma_{\text{2D}})$ (the approximate RCS measured in decibels (dB)) as well as the pointwise error $\log_{10}|\sigma_{\text{2D}}-\sigma_{\text{2D,exact}}|$ between the approximate RCS σ_{2D} and the exact RCS $\sigma_{\text{2D,exact}}$. For both calculations, we compute the RCS at 1000 equally spaced points in the observation angle φ. The approximate RCS is verified to have approximately 12 digits or better of accuracy.

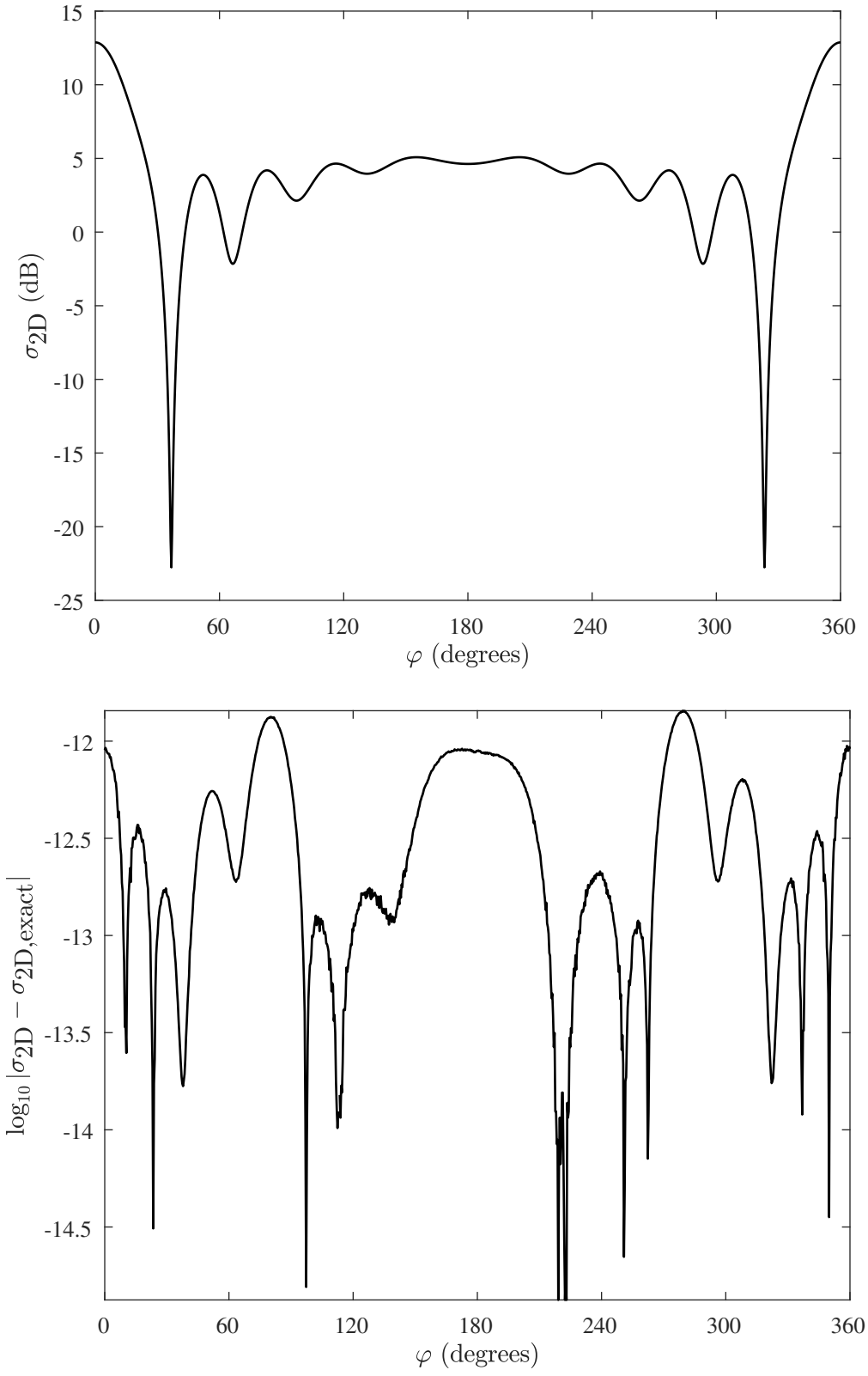

Figure 9.17: (top) Radar cross section (9.209) of the circular cylinder in decibels. (bottom) Pointwise error $\log_{10}|\sigma_{2D} - \sigma_{2D,\text{exact}}|$ in the radar cross section.

Chapter 10

Domain Decomposition for Non-Conforming Problems

The solutions produced in the previous chapter were computed using sparse direct solvers [147]. These methods are suitable for problems with smaller numbers of unknowns. In such cases, the amount of additional memory required by fill-in during the LU factorization does not tend to exceed the total amount of random access memory (RAM) of a workstation running the computations. The eigenfunction of the disk, the coaxial join, and the scattering problem from the previous chapter are good examples of this kind of behavior. However, the shielded microstrip line problem from the previous chapter comes close to exceeding the working memory of the four core Intel Xeon E5-1603 processor workstation used to compute the solution. The final factorization in the *hp*-adaption algorithm for that example costs close to 14.5 gigabytes (GB) of RAM (the workstation itself has 32 GB, some of which is needed to run the operating system and other applications). Note that the number of unknowns in the shielded microstrip line example is only about ten times as many as required for the other examples.

This type of memory constraint is particularly stringent for high frequency scattering problems. In such problems, a large number of unknowns are needed to capture the wave-like nature of the solution throughout the computational domain. This chapter explains how to solve the saddle point system produced in the previous chapter using an iterative domain decomposition algorithm. The domain decomposition method requires significantly less memory to solve the saddle point system and can be applied to problems with significantly more unknowns as a result. It is a type of FETI-DP algorithm [79] that is more general than the standard approach used for static problems [78, 80]. The generalization is required so that the number of iterations of the method depends only weakly on the frequency for Helmholtz problems.

The first part of this chapter describes how FETI-DP is related to classic null space and range space methods [114] used to solve saddle point systems and their associated preconditioners [166]. It then explains how to construct a sparse basis for the null space of the constraint matrix for a saddle point system with non-conforming continuity constraints. This, or a partial basis for the null space, is needed in any FETI-DP type domain decomposition algorithm, including the one developed in this chapter. The particular structure of the basis for the null space permits additional structure to be exploited in the algorithm. The final theory section of the chapter explains how to choose the partial basis for the null space when solving Poisson or Helmholtz problems. The choice is made so that the number of iterations of the method depends only weakly on the frequency of the problem. The final part of the chapter uses test problems to demonstrate the convergence behavior of the algorithm and applies the algorithm to challenging high frequency scattering problems.

10.1 The Dual-Primal Algorithm

The goal of this chapter is to solve the saddle point system

$$\begin{bmatrix} \underline{A} & \underline{C}^T \\ \underline{C} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \bar{d} \end{bmatrix} \tag{10.1}$$

arising from the non-conforming finite element method described in Chapter 9. We have seen in previous chapters that such a system can be obtained from either a variational formulation or the method of weighted residuals. In a variational formulation, the discretized Lagrangian is

$$\mathcal{L}(\bar{\phi}, \bar{\nu}) = \frac{1}{2}\bar{\phi}^T \underline{A}\bar{\phi} - \bar{\phi}^T\bar{b} + \bar{\nu}^T(\underline{C}\bar{\phi} - \bar{d}) \tag{10.2}$$

where $\bar{\phi}$ are called primal variables and $\bar{\nu}$ are called dual variables (also called Lagrange multipliers) [88]. Rather than solve for the primal variables directly as in a classical finite element method using the null space method, a dual-primal algorithm first explicitly enforces a subset of the constraints, then solves for a remaining set of dual variables from which the primal variables are finally obtained. This is where the abbreviation DP (which stands for Dual-Primal) in FETI-DP comes from[1].

The dual-primal algorithm proceeds as follows. First, the constraint matrix is partitioned

[1]The dual-primal algorithm should not be confused with the important class of numerical optimization algorithms called primal-dual methods which solve simultaneously for the primal and dual variables [90].

so that (10.1) becomes

$$\begin{bmatrix} \underline{A} & \underline{C}_p^T & \underline{C}_d^T \\ \underline{C}_p & 0 & 0 \\ \underline{C}_d & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu}_p \\ \bar{\nu}_d \end{bmatrix} = \begin{bmatrix} \bar{b} \\ \bar{d}_p \\ \bar{d}_d \end{bmatrix} \tag{10.3}$$

where $\bar{\nu}$ and $\bar{d}$ have been partitioned accordingly. We will assume that the constraint equations corresponding to the block $\underline{C}_p$ are meant to be explicitly enforced. To enforce these constraints, we use a partial null space method where we let

$$\bar{\phi} = \underline{Z}\bar{\phi}_p + \bar{\phi}_z \tag{10.4}$$

with $\underline{C}_p\underline{Z} = 0$ such that $\underline{Z}$ is a basis for the null space of $\underline{C}_p$. We call this a partial null space method because the standard null space method would choose $\underline{Z}$ as the basis for the null space of the entire constraint matrix $\underline{C}$. Substituting (10.4) into the second block row of (10.3) yields

$$\underline{C}_p(\underline{Z}\bar{\phi}_p + \bar{\phi}_z) = \bar{d}_p \tag{10.5}$$

$$\underbrace{\underline{C}_p\underline{Z}}_{0}\bar{\phi}_p + \underline{C}_p\bar{\phi}_z = \bar{d}_p \tag{10.6}$$

so that choosing $\bar{\phi}_z$ to satisfy

$$\underline{C}_p\bar{\phi}_z = \bar{d}_p \tag{10.7}$$

reduces the size of the saddle point system to solve. In particular, performing the same substitution into the first block row of (10.3) and multiplying from the left by $\underline{Z}^T$ gives

$$\underline{Z}^T\underline{A}(\underline{Z}\bar{\phi}_p + \bar{\phi}_z) + \underbrace{\underline{Z}^T\underline{C}_p^T}_{0}\bar{\nu}_p + \underline{Z}^T\underline{C}_d^T\bar{\nu}_d = \underline{Z}^T\bar{b} \tag{10.8}$$

$$\underline{Z}^T\underline{A}\underline{Z}\bar{\phi}_p + \underline{Z}^T\underline{C}_d^T\bar{\nu}_d = \underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z). \tag{10.9}$$

Similarly, substituting (10.4) into the third block row of (10.3) yields

$$\underline{C}_d(\underline{Z}\bar{\phi}_p + \bar{\phi}_z) = \bar{d}_d \tag{10.10}$$

$$\underline{C}_d\underline{Z}\bar{\phi}_p = \bar{d}_d - \underline{C}_d\bar{\phi}_z. \tag{10.11}$$

Equation (10.9) together with (10.11) yields the reduced saddle point system

$$\begin{bmatrix} \underline{Z}^T\underline{A}\underline{Z} & \underline{Z}^T\underline{C}_d^T \\ \underline{C}_d\underline{Z} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi}_p \\ \bar{\nu}_d \end{bmatrix} = \begin{bmatrix} \underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z) \\ \bar{d}_d - \underline{C}_d\bar{\phi}_z \end{bmatrix}. \tag{10.12}$$

We abbreviate the entries as $\hat{\underline{A}} = \underline{Z}^T \underline{A} \underline{Z}$, $\hat{\underline{C}} = \underline{C}_d \underline{Z}$, $\hat{b} = \underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z)$, and $\hat{d} = \bar{d}_d - \underline{C}_d \bar{\phi}_z$ so that

$$\begin{bmatrix} \hat{\underline{A}} & \hat{\underline{C}}^T \\ \hat{\underline{C}} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi}_p \\ \bar{\nu}_d \end{bmatrix} = \begin{bmatrix} \hat{b} \\ \hat{d} \end{bmatrix}. \tag{10.13}$$

In order to compute the right hand side of this saddle point system, we need to compute $\bar{\phi}_z$. Since $\underline{C}_p \bar{\phi}_z = \bar{d}_p$ and $\underline{C}_p$ has full row rank ($\underline{C}$ has full row rank by construction and $\underline{C}_p$ is comprised of a subset of its rows), its Moore-Penrose pseudoinverse is

$$\underline{C}_p^+ = \underline{C}_p^T (\underline{C}_p \underline{C}_p^T)^{-1}. \tag{10.14}$$

This means that choosing $\bar{\phi}_z = \underline{C}_p^+ \tilde{\phi}$ requires that $\tilde{\phi}$ satisfy

$$\underline{C}_p \bar{\phi}_z = \bar{d}_p \tag{10.15}$$

$$\underline{C}_p \underline{C}_p^+ \tilde{\phi} = \bar{d}_p \tag{10.16}$$

$$\underbrace{\underline{C}_p \underline{C}_p^T (\underline{C}_p \underline{C}_p^T)^{-1}}_{\underline{I}} \tilde{\phi} = \bar{d}_p. \tag{10.17}$$

In other words, we can choose

$$\bar{\phi}_z = \underline{C}_p^+ \bar{d}_p \tag{10.18}$$

in the right hand side of the reduced saddle point system (10.13).

In the dual-primal algorithm, rather than solve the reduced saddle point system directly, we use the range space method [114] to solve for the dual variables $\bar{\nu}_d$. That is, we consider cases for which $\hat{\underline{A}}$ is invertible (it is in all of the problems that we solve in this chapter) and isolate $\bar{\phi}_p$ in the first block row of (10.13). This is done by multiplying the first block row by $\hat{\underline{A}}^{-1}$ from the left to obtain

$$\underbrace{\hat{\underline{A}}^{-1} \hat{\underline{A}}}_{\underline{I}} \bar{\phi}_p + \hat{\underline{A}}^{-1} \hat{\underline{C}}^T \bar{\nu}_d = \hat{\underline{A}}^{-1} \hat{b}. \tag{10.19}$$

We then multiply by $\hat{\underline{C}}$ from the left which gives

$$\underbrace{\hat{\underline{C}} \bar{\phi}_p}_{\hat{d}} + \hat{\underline{C}} \hat{\underline{A}}^{-1} \hat{\underline{C}}^T \bar{\nu}_d = \hat{\underline{C}} \hat{\underline{A}}^{-1} \hat{b} \tag{10.20}$$

where we have used the second block row of (10.13) to simplify the first term. We obtain a

linear system in dual unknowns $\bar{\nu}_d$ given by

$$\underline{\hat{C}}\underline{\hat{A}}^{-1}\underline{\hat{C}}^T\bar{\nu}_d = \underline{\hat{C}}\underline{\hat{A}}^{-1}\hat{b} - \hat{d}. \tag{10.21}$$

This is equivalent to performing one step of block Gaussian elimination on (10.13) to zero the $\underline{\hat{C}}$ block. In a dual-primal algorithm, we solve (10.21) first, then solve

$$\underline{\hat{A}}\bar{\phi}_p = \hat{b} - \underline{\hat{C}}^T\bar{\nu}_d \tag{10.22}$$

for $\bar{\phi}_p$. These two solves consist of applying block backward substitution after the block Gaussian elimination step has been performed. Finally, to construct the original primal unknowns, we substitute $\bar{\phi}_p$ into (10.4).

In practice, (10.21) is solved using an iterative preconditioned Krylov subspace method [167]. Instead of using the left preconditioner

$$\underline{P}^{-1} = (\underline{\hat{C}}^+)^T\underline{\hat{A}}\underline{\hat{C}}^+ \tag{10.23}$$

(see, for example, [166]) we use a modified preconditioner. Since

$$\underline{P}^{-1}\underline{\hat{C}}\underline{\hat{A}}^{-1}\underline{\hat{C}}^T = (\underline{\hat{C}}^+)^T\hat{A}\underbrace{\underline{\hat{C}}^+\underline{\hat{C}}}_{\text{projection}}\underline{\hat{A}}^{-1}\underline{\hat{C}}^T, \tag{10.24}$$

we replace $\underline{\hat{C}}^+ = \underline{\hat{C}}^T(\underline{\hat{C}}\underline{\hat{C}}^T)^{-1}$ in $\underline{P}^{-1}$ with

$$\underline{\hat{C}}^\star = \underline{W}\underline{\hat{C}}^T(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)^{-1} \tag{10.25}$$

to obtain

$$\underline{P}^{-1} = (\underline{\hat{C}}^\star)^T\underline{\hat{A}}\underline{\hat{C}}^\star. \tag{10.26}$$

We choose $\underline{\hat{C}}^\star$ this way so that $\underline{\hat{C}}^\star\underline{\hat{C}}$ preserves the projection property of $\underline{\hat{C}}^+\underline{\hat{C}}$. Any matrix $\underline{M}$ satisfying $\underline{M}^2 = \underline{M}$ is a projection matrix and

$$(\underline{\hat{C}}^\star\underline{\hat{C}})^2 = \underbrace{[\underline{W}\underline{\hat{C}}^T(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)^{-1}\underline{\hat{C}}]}_{\underline{\hat{C}}^\star\underline{\hat{C}}}\underbrace{[\underline{W}\underline{\hat{C}}^T(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)^{-1}\underline{\hat{C}}]}_{\underline{\hat{C}}^\star\underline{\hat{C}}} \tag{10.27}$$

$$= \underbrace{\underline{W}\underline{\hat{C}}^T(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)^{-1}}_{\underline{\hat{C}}^\star}\underbrace{(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)(\underline{\hat{C}}\underline{W}\underline{\hat{C}}^T)^{-1}}_{\underline{I}}\underline{\hat{C}} \tag{10.28}$$

$$= \underline{\hat{C}}^\star\underline{\hat{C}}. \tag{10.29}$$

Note that when $\underline{W} = \underline{I}$, $\underline{\hat{C}}^\star = \underline{\hat{C}}^+$ and $\underline{\hat{C}}^+\underline{\hat{C}}$ is a projector onto the range of $\underline{\hat{C}}^T$. This

projector is orthogonal (since it is equal to its transpose). Other choices of $\underline{W}$ result in projections that are not orthogonal, but that produce more effective preconditioners. That is, they reduce the number of iterations required by the preconditioned Krylov subspace method to converge to a given tolerance. As a final note, in the preconditioner we follow the common practice [70] of using only the part of $\hat{\underline{A}}$ corresponding to the operator $-\nabla \cdot (\underline{\alpha} \nabla \phi)$ rather than the full operator. This means that we leave out any contributions from the term $\beta\phi$ or the Robin boundary conditions when forming $\hat{\underline{A}}$ in $\underline{P}^{-1}$.

10.2 A Sparse Basis for the Null Space

In order to complete the description of the dual-primal algorithm, we need to choose how to partition the constraint equations into blocks $\underline{C}_p$ and $\underline{C}_d$ and describe how to construct an associated basis $\underline{Z}$ for the null space of $\underline{C}_p$. These choices will inform how to choose the weight matrix $\underline{W}$ in the preconditioner. One particularly important constraint on the basis $\underline{Z}$ is that it should be sparse, otherwise we risk taking the original sparse finite element problem and transforming it into a dense problem. Fortunately, such a basis exists and can be computed in a straightforward manner.

Choosing the subset of constraints $\underline{C}_p$ to enforce explicitly depends on how we wish to perform domain decomposition. In the following, we will consider each element to belong to its own domain in the decomposition algorithm. We restrict our attention to this case because we are describing a finite element method aimed at computing high accuracy solutions with potentially high polynomial degree on each element. This is not typical of domain decomposition algorithms applied to finite element methods of low polynomial degree because the number of unknowns for a single element in such cases is small. Instead, several elements are grouped together into domains and the solution of the resulting local problems may be computed efficiently using direct methods in parallel.

In a typical FETI-DP method, the constraint matrix $\underline{C}_p$ corresponds to constraints enforcing continuity at cross points of the decomposition (vertices where several domains meet) and may include continuity of the average of the solution along boundaries between domains [71]. By virtue of imposing continuity constraints in the way described in Chapter 9, we have a hierarchy of constraints along each edge whose lowest order constraints fit into the standard FETI-DP framework. To enforce continuity at vertices between elements, we include the first constraint equation along each edge to form $\underline{C}_p$. Note that care must be taken in situations where certain constraints were discarded in assembling $\underline{C}$. Having been discarded, no constraint from the associated edge should be added to $\underline{C}_p$.

However, in contrast with a standard FETI-DP method, the hierarchy of constraints along

each edge means that we can include however many constraints we wish along a given edge. In practice, we keep this number, called l, fixed across the whole mesh (with the understanding that if the first l_{discard} constraints have been discarded along an edge, we only include $l - l_{\text{discard}}$ constraints corresponding to that edge). Note that if l exceeds the polynomial degree for all elements, then $\underline{C}_p = \underline{C}$ and the domain decomposition method reduces to the null space method. That is, we globally assemble the finite element problem and there is no domain decomposition to speak of. If $l = 0$, then $\underline{C}_d = \underline{C}$ and the domain decomposition method reduces to the range space method, but then the domain decomposition method does not possess a coarse space and the number of iterations required for the Krylov subspace method to converge can be large[2]. Later, we will select l to control the number of iterations required by the domain decomposition algorithm.

Once $\underline{C}_p$ is chosen, we construct a basis for its null space. Since $\underline{C}_p^+ \underline{C}_p$ is a projector onto the range of $\underline{C}_p^T$, $\underline{I} - \underline{C}_p^+ \underline{C}_p$ is a projector onto the null space of $\underline{C}_p$. To see why,

$$\underline{C}_p(\underline{I} - \underline{C}_p^+ \underline{C}_p) = \underline{C}_p(\underline{I} - \underline{C}_p^T(\underline{C}_p \underline{C}_p^T)^{-1} \underline{C}_p) \tag{10.30}$$

$$= \underline{C}_p - \underbrace{\underline{C}_p \underline{C}_p^T (\underline{C}_p \underline{C}_p^T)^{-1}}_{\underline{I}} \underline{C}_p \tag{10.31}$$

$$= 0 \tag{10.32}$$

so that the columns of $\underline{I} - \underline{C}_p^+ \underline{C}_p$ are all in the null space of $\underline{C}_p$. Unfortunately, there are more columns in $\underline{I} - \underline{C}_p^+ \underline{C}_p$ than required for a basis for the null space. This is because $\underline{C}_p \in \mathbb{R}^{m \times n}$ with $m < n$. Since $\underline{C}_p$ has full row rank, $\text{rank}(\underline{C}_p) = m$. The rank-nullity theorem states that

$$\underbrace{\text{rank}(\underline{C}_p)}_{m} + \text{null}(\underline{C}_p) = n \tag{10.33}$$

meaning that a basis for the null space should have $\text{null}(\underline{C}_p) = n - m$ columns. The matrix $\underline{I} - \underline{C}_p^+ \underline{C}_p$ is square with n columns, so that there are m dependent columns if we are to use it to construct a basis for the null space.

It turns out that a simple change of basis makes selecting columns for the basis of the null space a straightforward process. The change of basis is related to the one given in (4.69) which converted the two first integrated Legendre polynomial basis functions into

[2]For example, in Section 10.5.2, we solve a test Helmholtz problem to understand how convergence behaves as a function of element degree, number of elements in the mesh, and parameter l for different wavenumbers k. For a low frequency problem with $k = 1$ and degree $p = 8$ elements, the number of iterations grows as the number of elements is increased when $l = 0$ (even exceeding 1000 iterations when there are 1024 elements). In addition, even though the relative residual decreases, the difference between the computed solution and a solution computed via direct solver is $\mathcal{O}(1)$. When $l = 1$ for this same problem, the number of iterations never exceeds 25 and the computed solutions agree up to 10 digits.

interpolatory polynomials. On a given element, the two-dimensional basis functions are related to one-dimensional basis functions $\bar{N}(u_k) = \underline{B}\tilde{N}(u_k)$ for $k = 1, 2$, via the tensor product. This means that to convert to integrated Legendre polynomials with first two basis functions interpolatory, the expansion

$$\phi_j(\bar{x}) = \left(\bar{N}(u_2) \otimes \bar{N}(u_1)\right)^T \bar{\phi}_j \tag{10.34}$$

$$= \left(\bar{N}(u_2)^T \otimes \bar{N}(u_1)^T\right) \bar{\phi}_j \tag{10.35}$$

becomes

$$\phi_j(\bar{x}) = \left(\tilde{N}(u_2)^T \underline{B}^T \otimes \tilde{N}(u_1)^T \underline{B}^T\right) \bar{\phi}_j \tag{10.36}$$

$$= \left(\tilde{N}(u_2)^T \otimes \tilde{N}(u_1)^T\right) \underbrace{(\underline{B} \otimes \underline{B})\, \bar{\phi}_j}_{\tilde{\phi}_j} \tag{10.37}$$

where we have used (5.12) and the fact that $\underline{B} = \underline{B}^T$. If each element has its own polynomial degree, it is necessary to change the size of $\underline{B}$ accordingly to obtain

$$\tilde{\phi}_j = \underline{B}_j \bar{\phi}_j \tag{10.38}$$

where $\underline{B}_j = \underline{B} \otimes \underline{B}$ with the size of $\underline{B}$ chosen appropriately. Then the block diagonal matrix

$$\underline{B}_z = \begin{bmatrix} \underline{B}_1 & & & \\ & \underline{B}_2 & & \\ & & \ddots & \\ & & & \underline{B}_{N_e} \end{bmatrix} \tag{10.39}$$

can be used to convert all coefficients $\bar{\phi}$ from coefficients in the usual basis functions to coefficients $\tilde{\phi} = \underline{B}_z \bar{\phi}$ in the interpolatory basis functions. Since $\underline{B}_z$ is block diagonal, and each $\underline{B}_j$ is a Kronecker product of matrices $\underline{B}$, the matrix $\underline{B}_z$ inherits the orthogonality, symmetry, and involutory properties of $\underline{B}$. In particular, $\underline{B}_z = \underline{B}_z^T$ and $\underline{B}_z^{-1} = \underline{B}_z$. Among other things, this last property means that $\bar{\phi} = \underline{B}_z \tilde{\phi}$.

We use this change of basis to construct a basis for the null space of $\underline{C}_p$. Rather than work with $\underline{C}_p$ directly, we start from the second block row of (10.3) and use $\bar{\phi} = \underline{B}_z \tilde{\phi}$ to obtain

$$\underbrace{\underline{C}_p \underline{B}_z}_{\underline{C}_{\text{int}}} \tilde{\phi} = \bar{d}_p \tag{10.40}$$

which is a set of constraints on the unknowns corresponding to interpolatory basis functions.

If we compute the projector onto the null space of $\underline{C}_{\text{int}}$, we obtain

$$\underline{I} - \underline{C}_{\text{int}}^{+}\underline{C}_{\text{int}} = \underline{I} - (\underline{C}_p\underline{B}_z)^T[(\underline{C}_p\underline{B}_z)(\underline{C}_p\underline{B}_z)^T]^{-1}(\underline{C}_p\underline{B}_z) \tag{10.41}$$

$$= \underline{I} - \underline{B}_z\underline{C}_p^T(\underline{C}_p\underline{B}_z\underline{B}_z\underline{C}_p^T)^{-1}\underline{C}_p\underline{B}_z \tag{10.42}$$

where we have used $\underline{B}_z = \underline{B}_z^T$. Since $\underline{B}_z\underline{B}_z = \underline{I}$,

$$\underline{I} - \underline{C}_{\text{int}}^{+}\underline{C}_{\text{int}} = \underbrace{\underline{I}}_{\underline{B}_z\underline{B}_z} - \underline{B}_z\underline{C}_p^T(\underline{C}_p\underbrace{\underline{B}_z\underline{B}_z}_{\underline{I}}\underline{C}_p^T)^{-1}\underline{C}_p\underline{B}_z. \tag{10.43}$$

Then factoring $\underline{B}_z$ from both the left and right, and replacing $\underline{C}_p^{+} = \underline{C}_p^T(\underline{C}_p\underline{C}_p^T)^{-1}$ yields

$$\underline{I} - \underline{C}_{\text{int}}^{+}\underline{C}_{\text{int}} = \underline{B}_z(\underline{I} - \underline{C}_p^T(\underline{C}_p\underline{C}_p^T)^{-1}\underline{C}_p)\underline{B}_z \tag{10.44}$$

$$= \underline{B}_z(\underline{I} - \underline{C}_p^{+}\underline{C}_p)\underline{B}_z. \tag{10.45}$$

Because of the correspondence of columns of this matrix with interpolatory basis functions, we can keep a subset of the columns of this matrix corresponding to distinct edges in the mesh to obtain a basis for the null space of $\underline{C}_{\text{int}}$, which we call $\underline{Z}_{\text{int}}$. Denoting the set of columns to keep by $\mathcal{J}$, we have

$$\underline{Z}_{\text{int}} = [\underline{B}_z(\underline{I} - \underline{C}_p^{+}\underline{C}_p)\underline{B}_z]\underline{I}(:, \mathcal{J}) \tag{10.46}$$

where $\underline{I}(:, \mathcal{J})$ corresponds to the columns of the identity matrix given by the set $\mathcal{J}$. Since $\underline{C}_{\text{int}}\underline{Z}_{\text{int}} = 0$ and $\underline{C}_{\text{int}} = \underline{C}_p\underline{B}_z$, we have

$$\underline{C}_p\underbrace{\underline{B}_z\underline{Z}_{\text{int}}}_{\underline{Z}} = 0. \tag{10.47}$$

This means that the basis for the null space $\underline{Z}$ of the original constraint matrix $\underline{C}_p$ is

$$\underline{Z} = (\underline{I} - \underline{C}_p^{+}\underline{C}_p)\underline{B}_z\underline{I}(:, \mathcal{J}). \tag{10.48}$$

Thus, to form $\underline{Z}$, we select columns $\mathcal{J}$ from $\underline{B}_z$, then apply the projector onto the null space of the constraint matrix $\underline{C}_p$ to those columns[3]. In practice, $\underline{Z}$ constructed in this way is sparse.

Choosing the set $\mathcal{J}$ can be performed systematically because of the direct correspondence

[3] I am not aware of a systematic way to directly sample the columns of $\underline{I} - \underline{C}_p^{+}\underline{C}_p$ without first performing this change of variables. Since a basis for the null space is not unique, there may be other valid approaches to construct $\underline{Z}$ that I am not aware of.

between edge unknowns in the interpolatory basis and columns of (10.45). Starting from the finest level of the quadtree, we visit each element. For each element, we visit each edge (we keep track of each edge that has been visited). We then check if the edge belongs to a boundary where a boundary condition is meant to be imposed. If it does, and the edge requires a Dirichlet boundary condition, we discard the first $l-1$ edge unknowns along that edge. If the edge does not require a boundary condition and it has yet to be visited, we find its neighbor. If the neighbor is on the same level of the tree, then we keep the first $l-1$ edge functions of the element and discard the first $l-1$ edge functions of the neighbor. If the neighbor is on a coarser level of the tree, then we discard the first $l-1$ edge functions of the element and keep all edge functions of the neighbor.

We must also consider the vertex unknowns. If the neighbor is on a coarser level, we discard the vertex unknowns of the element corresponding to that shared edge. If the neighbor is on the same level or the current edge has multiple smaller neighbors on a finer level, then one vertex unknown per vertex must be kept, but all other vertex unknowns corresponding to the same vertex must be discarded. We classify all of these kept unknowns as type 2. They are associated with explicitly enforcing the constraints in $\underline{C}_p$ along edges and at vertices. All interior unknowns are kept and all edge unknowns that were not discarded or already classified are kept. We classify these unknowns as type 1. They are unknowns that are only coupled through the matrix $\underline{C}_d$ or that are coupled only with other unknowns on an element-by-element basis. We make sure that the splitting into two types of unknowns is reflected in the index set $\mathcal{J}$ so that

$$\underline{Z} = \begin{bmatrix} \underline{Z}_1 & \underline{Z}_2 \end{bmatrix} \tag{10.49}$$

is partitioned into two blocks.

As an example, Figure 10.1 illustrates which edge and vertex unknowns to keep in set $\mathcal{J}$ for a given quadtree mesh. For this example, we have assumed that the boundaries of the square have Dirichlet boundary conditions imposed. The particular set of unknowns shown in Figure 10.1 is selected by following a z-ordering of the quadtree from the finest level up to the coarsest level (indicated by the numbering). The edges for each element are visited in the local order defined in Figure 9.3. For clarity, the edge and vertex diagrams are separated, but in practice, the edges and vertices can be tracked simultaneously.

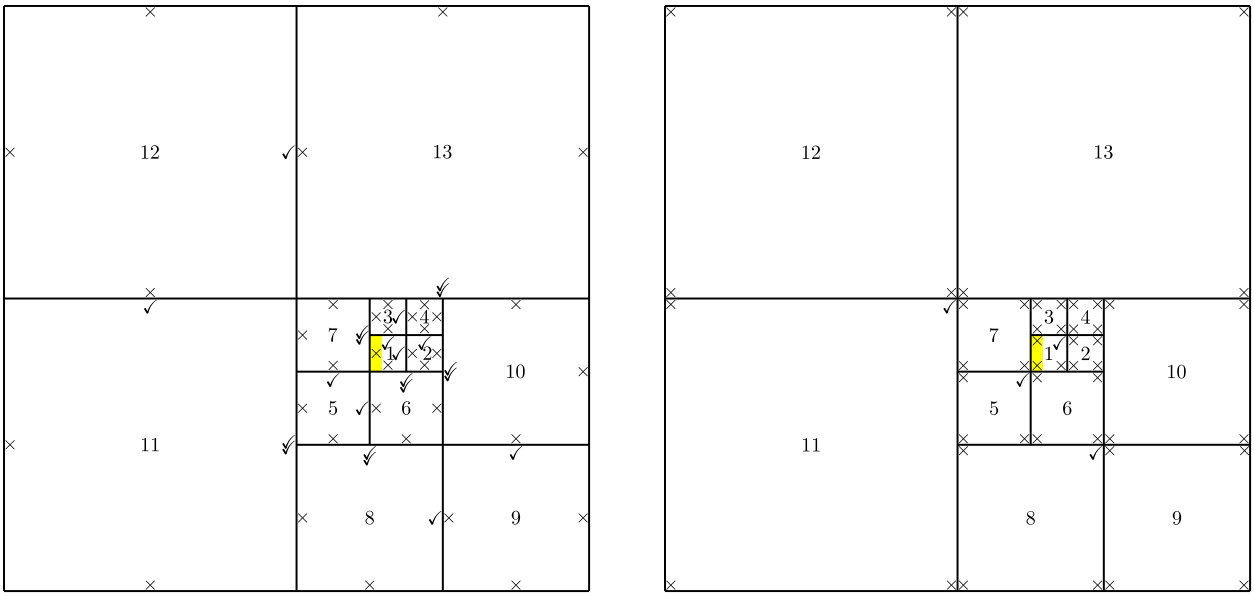

Figure 10.1: (left) Illustration of edge unknowns to keep when constructing $\underline{Z}$. The first $l-1$ edge unknowns are discarded on edges marked with an $\times$ and kept on edges marked with a ✓. A double ✓ indicates edges where all edge unknowns are kept. (right) Illustration of vertex unknowns to keep. An $\times$ indicates that the corresponding vertex unknown is discarded whereas a ✓ indicates that it is kept. The first edge and vertices visited by the algorithm are highlighted in yellow.

10.3 Consequences of this Choice of Basis for the Null Space

The splitting of $\underline{C}$ into $\underline{C}_p$ and $\underline{C}_d$ and the construction of the sparse basis $\underline{Z}$ for the null space of $\underline{C}_p$ as described in the previous section leads to additional structure in the reduced saddle point system (10.13). In particular, by virtue of the partitioning (10.49) of $\underline{Z}$, component $\hat{\underline{A}} = \underline{Z}^T \underline{A} \underline{Z}$ can be written as

$$\hat{\underline{A}} = \begin{bmatrix} \underline{Z}_1 & \underline{Z}_2 \end{bmatrix}^T \underline{A} \begin{bmatrix} \underline{Z}_1 & \underline{Z}_2 \end{bmatrix} \tag{10.50}$$

$$= \begin{bmatrix} \underline{Z}_1^T \\ \underline{Z}_2^T \end{bmatrix} \underline{A} \begin{bmatrix} \underline{Z}_1 & \underline{Z}_2 \end{bmatrix} \tag{10.51}$$

$$= \begin{bmatrix} \underline{Z}_1^T \underline{A} \underline{Z}_1 & \underline{Z}_1^T \underline{A} \underline{Z}_2 \\ \underline{Z}_2^T \underline{A} \underline{Z}_1 & \underline{Z}_2^T \underline{A} \underline{Z}_2 \end{bmatrix}, \tag{10.52}$$

component $\hat{\underline{C}} = \underline{C}_d \underline{Z}$ can be written as

$$\hat{\underline{C}} = \underline{C}_d \begin{bmatrix} \underline{Z}_1 & \underline{Z}_2 \end{bmatrix} = \begin{bmatrix} \underline{C}_d \underline{Z}_1 & \underline{C}_d \underline{Z}_2 \end{bmatrix}, \tag{10.53}$$

and component $\hat{b} = \underline{Z}^T(\bar{b} - \underline{A}\bar{\phi}_z)$ can be written as

$$\hat{b} = \begin{bmatrix} \underline{Z}_1^T \\ \underline{Z}_2^T \end{bmatrix} (\bar{b} - \underline{A}\bar{\phi}_z) \tag{10.54}$$

$$= \begin{bmatrix} \underline{Z}_1^T(\bar{b} - \underline{A}\bar{\phi}_z) \\ \underline{Z}_2^T(\bar{b} - \underline{A}\bar{\phi}_z) \end{bmatrix}. \tag{10.55}$$

We give the submatrices in these expressions shorthand notation $\underline{A}_{ij} = \underline{Z}_i^T \underline{A} \underline{Z}_j$, $\underline{C}_j = \underline{C}_d \underline{Z}_j$, and $\bar{b}_i = \underline{Z}_i^T(\bar{b} - \underline{A}\bar{\phi}_z)$ so that

$$\hat{\underline{A}} = \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{12}^T & \underline{A}_{22} \end{bmatrix}, \qquad \hat{\underline{C}} = \begin{bmatrix} \underline{C}_1 & \underline{C}_2 \end{bmatrix}, \qquad \hat{b} = \begin{bmatrix} \bar{b}_1 \\ \bar{b}_2 \end{bmatrix}. \tag{10.56}$$

By construction, $\underline{A}_{11}$ is block diagonal with each block corresponding to edge and interior unknowns for a given element that have not been constrained by $\underline{C}_p$. Ideally, we would like to exploit this structure when solving (10.21) using a Krylov subspace method with preconditioner (10.26).

First, we consider (10.21). To do so, we need to compute the block matrix inverse of $\hat{\underline{A}}$. This can be done by starting from the augmented block matrix

$$\left[\begin{array}{cc|cc} \underline{A}_{11} & \underline{A}_{12} & \underline{I} & 0 \\ \underline{A}_{12}^T & \underline{A}_{22} & 0 & \underline{I} \end{array}\right] \tag{10.57}$$

and performing block matrix operations to transform the left portion of the augmented system into the identity matrix (as one would do in a simpler case to obtain the inverse of a two-by-two matrix). Subtracting $\underline{A}_{12}^T \underline{A}_{11}^{-1}$ times the first block row from the second gives

$$\left[\begin{array}{cc|cc} \underline{A}_{11} & \underline{A}_{12} & \underline{I} & 0 \\ 0 & \underline{A}_{22} - \underline{A}_{12}^T \underline{A}_{11}^{-1} \underline{A}_{12} & -\underline{A}_{12}^T \underline{A}_{11}^{-1} & \underline{I} \end{array}\right]. \tag{10.58}$$

Letting $\underline{K} = \underline{A}_{22} - \underline{A}_{12}^T \underline{A}_{11}^{-1} \underline{A}_{12}$, and multiplying the second block row by $\underline{K}^{-1}$ gives

$$\left[\begin{array}{cc|cc} \underline{A}_{11} & \underline{A}_{12} & \underline{I} & 0 \\ 0 & \underline{I} & -\underline{K}^{-1} \underline{A}_{12}^T \underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right]. \tag{10.59}$$

Subtracting $\underline{A}_{12}$ times the second block row from the first block row gives

$$\left[\begin{array}{cc|cc} \underline{A}_{11} & 0 & \underline{I} + \underline{A}_{12} \underline{K}^{-1} \underline{A}_{12}^T \underline{A}_{11}^{-1} & -\underline{A}_{12} \underline{K}^{-1} \\ 0 & \underline{I} & -\underline{K}^{-1} \underline{A}_{12}^T \underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right]. \tag{10.60}$$

Finally, multiplying the first block row by $\underline{A}_{11}^{-1}$ gives

$$\left[\begin{array}{cc|cc} \underline{I} & 0 & \underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & -\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1} \\ 0 & \underline{I} & -\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right] \tag{10.61}$$

so that

$$\left[\begin{array}{cc} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{12}^{T} & \underline{A}_{22} \end{array}\right]^{-1} = \left[\begin{array}{cc} \underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & -\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1} \\ -\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right]. \tag{10.62}$$

Using this block matrix inversion formula, we find that

$$\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^{T} = \left[\begin{array}{cc} \underline{C}_1 & \underline{C}_2 \end{array}\right] \left[\begin{array}{cc} \underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & -\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1} \\ -\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right] \left[\begin{array}{c} \underline{C}_1^{T} \\ \underline{C}_2^{T} \end{array}\right] \tag{10.63}$$

yields

$$\begin{aligned} &\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^{T} = \\ &\quad \underline{C}_1[\underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}]\underline{C}_1^{T} - \underline{C}_1\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{C}_2^{T} - \underline{C}_2\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}\underline{C}_1^{T} + \underline{C}_2\underline{K}^{-1}\underline{C}_2^{T}. \end{aligned} \tag{10.64}$$

Similarly,

$$\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{b} = \left[\begin{array}{cc} \underline{C}_1 & \underline{C}_2 \end{array}\right] \left[\begin{array}{cc} \underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & -\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1} \\ -\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1} & \underline{K}^{-1} \end{array}\right] \left[\begin{array}{c} \bar{b}_1 \\ \bar{b}_2 \end{array}\right] \tag{10.65}$$

which gives

$$\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{b} = \underline{C}_1[\underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}]\bar{b}_1 - \underline{C}_1\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\bar{b}_2 - \underline{C}_2\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}\bar{b}_1 + \underline{C}_2\underline{K}^{-1}\bar{b}_2. \tag{10.66}$$

Using these expressions in (10.21) yields

$$\begin{aligned} &\left\{\underline{C}_1[\underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}]\underline{C}_1^{T} - \underline{C}_1\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{C}_2^{T} - \underline{C}_2\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}\underline{C}_1^{T} + \underline{C}_2\underline{K}^{-1}\underline{C}_2^{T}\right\}\bar{\nu}_d \\ &\quad = \underline{C}_1[\underline{A}_{11}^{-1}+\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}]\bar{b}_1 - \underline{C}_1\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\bar{b}_2 - \underline{C}_2\underline{K}^{-1}\underline{A}_{12}^{T}\underline{A}_{11}^{-1}\bar{b}_1 + \underline{C}_2\underline{K}^{-1}\bar{b}_2 - \hat{d}. \end{aligned} \tag{10.67}$$

In a Krylov subspace method, we only need to compute the action of $\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^{T}$ on a vector. In this last equation, we see that the main computational cost in doing so arises from computing the action of $\underline{A}_{11}^{-1}$ and $\underline{K}^{-1}$ on a vector. Since $\underline{A}_{11}$ is block diagonal, its inverse applied to

a vector can be performed in parallel across all elements. Appendix B describes how to apply the inverse of each block efficiently. In contrast, $\underline{K} = \underline{A}_{22} - \underline{A}_{12}^T \underline{A}_{11}^{-1} \underline{A}_{12}$ is not block diagonal, and the size of this square matrix is governed by the choice of parameter l. As a rule of thumb, the number of rows in $\underline{K}$ is equal to l times the number of edges in the mesh. For many problems, this is considerably smaller than the total number of unknowns. In addition, $\underline{K}$ is sparse so that a sparse direct factorization can be applied [147].

Next, we consider the preconditioner (10.26). To do so, we need to choose the weight matrix $\underline{W}$ in (10.25). We choose

$$\underline{W} = \begin{bmatrix} \underline{W}_1 & \underline{W}_2 \\ 0 & \underline{I} \end{bmatrix} \tag{10.68}$$

with $\underline{W}_1$ a particular block diagonal matrix related to $\underline{A}_{11}$ and $\underline{W}_2$ related to $\underline{A}_{11}$ and $\underline{A}_{12}$. To better understand how to choose $\underline{W}_1$ and $\underline{W}_2$ for a problem with a non-conforming mesh, it is informative to first consider a problem with a conforming mesh. This considerably simplifies our development thus far because, in such a case, the matrix $\underline{C}_2$ is zero. In a conforming mesh, the remaining constraints in $\underline{C}_d$ after applying the partial null space method are independent of the constraints in $\underline{C}_p$ because each set of constraints along an edge involves only one other edge. In addition, the constraints along each edge are hierarchical so do not involve unknowns already constrained. However, when non-conforming edges are present, there is coupling between different sets of fine edge constraints through their shared coarse edge. The coupling causes previously constrained unknowns to enter into constraints that have yet to be explicitly imposed. Since $\underline{C}_2 = 0$ in the conforming case, (10.67) reduces to

$$\underline{C}_1[\underline{A}_{11}^{-1} + \underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^T\underline{A}_{11}^{-1}]\underline{C}_1^T\bar{\nu}_d = \underline{C}_1[\underline{A}_{11}^{-1} + \underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^T\underline{A}_{11}^{-1}]\bar{b}_1 - \underline{C}_1\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\bar{b}_2 - \hat{d} \tag{10.69}$$

which has the same structure as the system of equations arising in the FETI-DP method (the matrices themselves can differ depending on our choice of parameter l). For this reason, our choice of $\underline{W}_1$ in (10.68) will be related to the Dirichlet preconditioner used for FETI-DP [78]. Combining (10.26) with (10.25) yields the preconditioner

$$\underline{P}^{-1} = (\hat{\underline{C}}\underline{W}\hat{\underline{C}}^T)^{-T}\hat{\underline{C}}\underline{W}^T\hat{\underline{A}}\underline{W}\hat{\underline{C}}^T(\hat{\underline{C}}\underline{W}\hat{\underline{C}}^T)^{-1}. \tag{10.70}$$

There are two components to the preconditioner. The first is

$$\hat{\underline{C}}\underline{W}\hat{\underline{C}}^T = \begin{bmatrix} \underline{C}_1 & \underline{C}_2 \end{bmatrix} \begin{bmatrix} \underline{W}_1 & \underline{W}_2 \\ 0 & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{C}_1^T \\ \underline{C}_2^T \end{bmatrix}. \tag{10.71}$$

When $\underline{C}_2 = 0$, this simplifies to

$$\hat{\underline{C}}\underline{W}\hat{\underline{C}}^T = \underline{C}_1\underline{W}_1\underline{C}_1^T. \tag{10.72}$$

The second component is

$$\hat{\underline{C}}\underline{W}^T\hat{\underline{A}}\underline{W}\hat{\underline{C}}^T = \begin{bmatrix} \underline{C}_1 & \underline{C}_2 \end{bmatrix} \begin{bmatrix} \underline{W}_1^T & 0 \\ \underline{W}_2^T & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{12}^T & \underline{A}_{22} \end{bmatrix} \begin{bmatrix} \underline{W}_1 & \underline{W}_2 \\ 0 & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{C}_1^T \\ \underline{C}_2^T \end{bmatrix}. \tag{10.73}$$

When $\underline{C}_2 = 0$, this simplifies to

$$\hat{\underline{C}}\underline{W}^T\hat{\underline{A}}\underline{W}\hat{\underline{C}}^T = \underline{C}_1\underline{W}_1^T\underline{A}_{11}\underline{W}_1\underline{C}_1^T. \tag{10.74}$$

Together, the two components form the preconditioner

$$\underline{P}^{-1} = (\underline{C}_1\underline{W}_1\underline{C}_1^T)^{-T}\underline{C}_1\underline{W}_1^T\underline{A}_{11}\underline{W}_1\underline{C}_1^T(\underline{C}_1\underline{W}_1\underline{C}_1^T)^{-1} \tag{10.75}$$

which is the form of preconditioner encountered in the FETI-DP method.

In literature regarding the FETI-DP method [71], the lumped preconditioner corresponds to choosing $\underline{W}_1 = \underline{I}$. This preconditioner is not very effective at keeping the number of iterations in a Krylov subspace method small. However, it is inexpensive to apply to a matrix, in particular because the matrix $\underline{C}_1\underline{C}_1^T$ is diagonal for conforming meshes (and thus its inverse is simple to compute). The lumped preconditioner is not effective because the product $\underline{C}_1\underline{A}_{11}\underline{C}_1^T$ only involves entries in $\underline{A}_{11}$ related to edge unknowns and ignores entries that correspond to interior unknowns (this is because $\underline{C}_1$ only constrains edge unknowns when imposing continuity constraints). Recall that $\underline{A}_{11}$ is a block diagonal matrix whose blocks $\underline{A}^{(j)}$ correspond with unknowns for the jth element. If we partition $\underline{A}^{(j)}$ as

$$\underline{A}^{(j)} = \begin{bmatrix} \underline{A}_{ee}^{(j)} & \underline{A}_{ei}^{(j)} \\ (\underline{A}_{ei}^{(j)})^T & \underline{A}_{ii}^{(j)} \end{bmatrix} \tag{10.76}$$

where subscript e denotes entries corresponding to edge unknowns and subscript i denotes entries corresponding to interior unknowns, then the product with $\underline{C}_1$ only involves the blocks $\underline{A}_{ee}^{(j)}$. However, if we choose the weight matrix $\underline{W}_1$ to be block diagonal with blocks

$$\underline{W}^{(j)} = \begin{bmatrix} \underline{I} & 0 \\ -(\underline{A}_{ii}^{(j)})^{-1}(\underline{A}_{ei}^{(j)})^T & 0 \end{bmatrix}, \tag{10.77}$$

then the product $\underline{W}_1^T \underline{A}_{11} \underline{W}_1$ results in a block diagonal matrix with blocks

$$\underline{W}^{(j)T} \underline{A}^{(j)} \underline{W}^{(j)} = \begin{bmatrix} \underline{I} & -\underline{A}_{ei}^{(j)} (\underline{A}_{ii}^{(j)})^{-1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \underline{A}_{ee}^{(j)} & \underline{A}_{ei}^{(j)} \\ (\underline{A}_{ei}^{(j)})^T & \underline{A}_{ii}^{(j)} \end{bmatrix} \begin{bmatrix} \underline{I} & 0 \\ -(\underline{A}_{ii}^{(j)})^{-1} (\underline{A}_{ei}^{(j)})^T & 0 \end{bmatrix}. \quad (10.78)$$

In this last equation, note that the matrix $\underline{A}_{11}$ is symmetric so that $(\underline{A}_{ii}^{(j)})^{-1}$ is its own transpose. Multiplying these matrices yields

$$\underline{W}^{(j)T} \underline{A}^{(j)} \underline{W}^{(j)} = \begin{bmatrix} \underline{A}_{ee}^{(j)} - \underline{A}_{ei}^{(j)} (\underline{A}_{ii}^{(j)})^{-1} (\underline{A}_{ei}^{(j)})^T & 0 \\ 0 & 0 \end{bmatrix} \quad (10.79)$$

which has transferred the effect of the interior entries to the edge entries. This choice of $\underline{W}_1$ corresponds to the Dirichlet preconditioner in the FETI-DP method. The Dirichlet preconditioner tends to require fewer iterations to converge compared with the lumped preconditioner, but also involves solving a block diagonal system with blocks $\underline{A}_{ii}^{(j)}$ (see Appendix B for details on how to apply the inverse of each block efficiently). Note that $\underline{C}_1 \underline{W}_1 \underline{C}_1^T$ continues to reduce to $\underline{C}_1 \underline{C}_1^T$ since the edge-edge block in (10.77) is the identity matrix.

Often, a diagonal scaling is introduced to the weight matrix when dealing with problems where parameter $\underline{\alpha}$ is discontinuous [168]. This is done by taking (10.77) and scaling by a diagonal matrix to obtain

$$\underline{W}_{\text{scaled}}^{(j)} = \begin{bmatrix} \underline{I} & 0 \\ -(\underline{A}_{ii}^{(j)})^{-1} (\underline{A}_{ei}^{(j)})^T & 0 \end{bmatrix} \begin{bmatrix} \underline{D}^{(j)} & 0 \\ 0 & \underline{I} \end{bmatrix} \quad (10.80)$$

$$= \begin{bmatrix} \underline{D}^{(j)} & 0 \\ -(\underline{A}_{ii}^{(j)})^{-1} (\underline{A}_{ei}^{(j)})^T \underline{D}^{(j)} & 0 \end{bmatrix}. \quad (10.81)$$

Note that

$$\underline{W}_{\text{scaled}}^{(j)T} \underline{A}^{(j)} \underline{W}_{\text{scaled}}^{(j)} = \begin{bmatrix} \underline{D}^{(j)} [\underline{A}_{ee}^{(j)} - \underline{A}_{ei}^{(j)} (\underline{A}_{ii}^{(j)})^{-1} (\underline{A}_{ei}^{(j)})^T] \underline{D}^{(j)} & 0 \\ 0 & 0 \end{bmatrix}. \quad (10.82)$$

In the following, we are going to use a heuristic argument to explain how to choose $\underline{D}^{(j)}$. To do so, imagine that block matrices $\underline{D}$, $\underline{A}_{ee}$, $\underline{A}_{ei}$, and $\underline{A}_{ii}^{-1}$ are formed so that

$$\underline{W}_1^T \underline{A}_{11} \underline{W}_1 = \underline{D} (\underline{A}_{ee} - \underline{A}_{ei} \underline{A}_{ii}^{-1} \underline{A}_{ei}^T) \underline{D}. \quad (10.83)$$

Then the preconditioner (10.75) becomes

$$\underline{P}^{-1} = (\underline{C}_1 \underline{D} \underline{C}_1^T)^{-T} \underline{C}_1 \underline{D} (\underline{A}_{ee} - \underline{A}_{ei} \underline{A}_{ii}^{-1} \underline{A}_{ei}^T) \underline{D} \underline{C}_1^T (\underline{C}_1 \underline{D} \underline{C}_1^T)^{-1}. \quad (10.84)$$

Compare this expression with the second matrix term in (10.69) given by

$$\underline{C}_1[\underline{A}_{11}^{-1}\underline{A}_{12}\underline{K}^{-1}\underline{A}_{12}^T\underline{A}_{11}^{-1}]\underline{C}_1^T = \underline{C}_1[\underline{A}_{11}^{-1}\underline{A}_{12}(\underline{A}_{22} - \underline{A}_{12}^T\underline{A}_{11}^{-1}\underline{A}_{12})^{-1}\underline{A}_{12}^T\underline{A}_{11}^{-1}]\underline{C}_1^T. \tag{10.85}$$

Since $\underline{P}^{-1}$ is supposed to behave like the inverse of the coupling term (10.85) in (10.69), we choose $\underline{D}$ to behave like $\underline{\hat{A}}^{-1}$. Then $(\underline{C}_1\underline{D}\underline{C}_1^T)^{-T}$ in (10.84) scales like $\underline{\hat{A}}$ ($\underline{C}_1$ is unitless), whose counterpart in (10.85) is $\underline{A}_{11}^{-1}$. Similarly, $\underline{D}$ in (10.84) scales like $\underline{\hat{A}}^{-1}$, whose counterpart in (10.85) is $\underline{A}_{12}$. Finally, $\underline{A}_{ee} - \underline{A}_{ei}\underline{A}_{ii}^{-1}\underline{A}_{ei}^T$ in (10.84) has counterpart $(\underline{A}_{22} - \underline{A}_{12}^T\underline{A}_{11}^{-1}\underline{A}_{12})^{-1}$ in (10.85). Choosing

$$\underline{D}^{(j)} = [\text{diag}(\underline{A}_{ee}^{(j)})]^{-1} \tag{10.86}$$

is one simple way to achieve this scaling[4].

Now that we have seen how to select the weight matrix (10.68) in preconditioner (10.70) for a conforming mesh, we consider the non-conforming case. We will continue to choose $\underline{W}_1$ in the way specified for the conforming case by selecting the Dirichlet preconditioner for $\underline{W}_1$ (possibly with diagonal scaling) but now need to select $\underline{W}_2$. We choose $\underline{W}_2$ to simplify the product $\underline{W}^T\underline{\hat{A}}\underline{W}$ in (10.70) to a block diagonal matrix whose second term counteracts the term $\underline{C}_2\underline{K}^{-1}\underline{C}_2^T$ in (10.67). That is,

$$\underline{W}^T\underline{\hat{A}}\underline{W} = \begin{bmatrix} \underline{W}_1^T & 0 \\ \underline{W}_2^T & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{12}^T & \underline{A}_{22} \end{bmatrix} \begin{bmatrix} \underline{W}_1 & \underline{W}_2 \\ 0 & \underline{I} \end{bmatrix} \tag{10.87}$$

$$= \begin{bmatrix} \underline{W}_1^T\underline{A}_{11}\underline{W}_1 & \underline{W}_1^T(\underline{A}_{11}\underline{W}_2 + \underline{A}_{12}) \\ (\underline{W}_2^T\underline{A}_{11} + \underline{A}_{12}^T)\underline{W}_1 & \underline{W}_2^T\underline{A}_{11}\underline{W}_2 + \underline{A}_{12}^T\underline{W}_2 + \underline{W}_2^T\underline{A}_{12} + \underline{A}_{22} \end{bmatrix}. \tag{10.88}$$

For this to be block diagonal,

$$\underline{A}_{11}\underline{W}_2 + \underline{A}_{12} = 0, \tag{10.89}$$

which means that $\underline{W}_2 = -\underline{A}_{11}^{-1}\underline{A}_{12}$. Then

$$\underline{W}^T\underline{\hat{A}}\underline{W} = \begin{bmatrix} \underline{W}_1^T\underline{A}_{11}\underline{W}_1 & 0 \\ 0 & \underline{K} \end{bmatrix} \tag{10.90}$$

since

$$\underline{W}_2^T\underline{A}_{11}\underline{W}_2 + \underline{A}_{12}^T\underline{W}_2 + \underline{W}_2^T\underline{A}_{12} + \underline{A}_{22} = \underline{A}_{12}^T\underline{A}_{11}^{-1}\underline{A}_{11}\underline{A}_{11}^{-1}\underline{A}_{12} - \underline{A}_{12}^T\underline{A}_{11}^{-1}\underline{A}_{12} - \underline{A}_{12}^T\underline{A}_{11}^{-1}\underline{A}_{12} + \underline{A}_{22} \tag{10.91}$$

[4]Other authors choose to scale differently [78]. I have found that these different scalings lead to similar behavior in terms of the number of iterations required for the Krylov subspace method to converge.

and $\underline{K} = \underline{A}_{22} - \underline{A}_{12}^T \underline{A}_{11}^{-1} \underline{A}_{12}$. Using this expression in (10.73) gives

$$\hat{\underline{C}} \underline{W}^T \hat{\underline{A}} \underline{W} \hat{\underline{C}}^T = \begin{bmatrix} \underline{C}_1 & \underline{C}_2 \end{bmatrix} \begin{bmatrix} \underline{W}_1^T \underline{A}_{11} \underline{W}_1 & 0 \\ 0 & \underline{K} \end{bmatrix} \begin{bmatrix} \underline{C}_1^T \\ \underline{C}_2^T \end{bmatrix} \tag{10.92}$$

$$= \underline{C}_1 \underline{W}_1^T \underline{A}_{11} \underline{W}_1 \underline{C}_1^T + \underline{C}_2 \underline{K} \underline{C}_2^T. \tag{10.93}$$

Similarly, (10.71) becomes

$$\hat{\underline{C}} \underline{W} \hat{\underline{C}}^T = \begin{bmatrix} \underline{C}_1 & \underline{C}_2 \end{bmatrix} \begin{bmatrix} \underline{W}_1 & -\underline{A}_{11}^{-1} \underline{A}_{12} \\ 0 & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{C}_1^T \\ \underline{C}_2^T \end{bmatrix} \tag{10.94}$$

$$= \underline{C}_1 \underline{W}_1 \underline{C}_1^T + \underline{C}_2 \underline{C}_2^T - \underline{C}_1 \underline{A}_{11}^{-1} \underline{A}_{12} \underline{C}_2^T. \tag{10.95}$$

In practice, we omit the term $\underline{C}_1 \underline{A}_{11}^{-1} \underline{A}_{12} \underline{C}_2^T$ when using $\hat{\underline{C}} \underline{W} \hat{\underline{C}}^T$ in preconditioner (10.70) as we have found empirically that it has no significant impact on the number of iterations of the Krylov subspace method, nor on the accuracy of the computed solution to the PDE. We note that removing the term $\underline{C}_2 \underline{C}_2^T$ can have a significant impact on both the number of iterations and accuracy of the solution, worsening both considerably. Curiously, removing the term $\underline{C}_2 \underline{K} \underline{C}_2^T$ does not have a significant impact on the number of iterations, but causes the solution to be inaccurate at non-conforming edges[5]. For these reasons, we use the preconditioner

$$\underline{P}^{-1} = (\underline{C}_1 \underline{W}_1 \underline{C}_1^T + \underline{C}_2 \underline{C}_2^T)^{-T} (\underline{C}_1 \underline{W}_1^T \underline{A}_{11} \underline{W}_1 \underline{C}_1^T + \underline{C}_2 \underline{K} \underline{C}_2^T) (\underline{C}_1 \underline{W}_1 \underline{C}_1^T + \underline{C}_2 \underline{C}_2^T)^{-1} \tag{10.96}$$

in practice for non-conforming mesh problems with $\underline{W}_1$ chosen as in a Dirichlet preconditioner. Notice that the term $\underline{C}_2 \underline{K} \underline{C}_2^T$ in the preconditioner tends to behave in the opposite manner as the term $\underline{C}_2 \underline{K}^{-1} \underline{C}_2^T$ in the system (10.67) that is to be solved. We note that in the non-conforming case, the matrix $\underline{C}_2 \underline{C}_2^T$ is not diagonal (as opposed to the matrix $\underline{C}_1 \underline{W}_1 \underline{C}_1^T$). It is possible to recover this diagonal property by grouping elements sharing a non-conforming edge together into a single subdomain of the domain decomposition[6].

[5] For example, in Section 10.6.1, the solution with this term is accurate to 10 digits throughout the domain but removing this term causes the solution accuracy to degrade to 4 digits along non-conforming edges. In either case, the residual used to determine when to stop the Krylov subspace method decreases in the same way so that both methods appear to be equally effective, but only a look at the true error reveals the weakness of the second approach.

[6] I do not do this in this thesis. Instead, each block in the block diagonal matrix $\underline{A}_{11}$ corresponds to a single element so that the structure of the element operator matrices can be exploited.

10.4 Considerations for the Helmholtz Equation

The domain decomposition method, as described thus far, is effective for Poisson problems with $\underline{\alpha}$ positive definite and parameters $\gamma = \beta = 0$ subject to Dirichlet boundary conditions for any choice of parameter l greater than zero. In such a case, the matrix $\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^T$ in system (10.21) and its associated preconditioner $\underline{P}^{-1}$ are both symmetric, positive definite. As a result, we use the preconditioned conjugate gradients method (PCG) as a Krylov subspace method [167] to solve (10.21). When $l = 1$ and all elements possess degree p on a conforming mesh, we find experimentally that

$$\kappa_2(\underline{L}_P^T\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^T\underline{L}_P) \leq C(1 + \log(p^2))^2 \tag{10.97}$$

with $C > 0$ a constant independent of degree p, mesh size h, and parameter $\underline{\alpha}$ (we never compute the Cholesky decomposition $\underline{P}^{-1} = \underline{L}_P\underline{L}_P^T$ in practice). This bound shows that the number of iterations of PCG depends weakly on the polynomial degree, and consequently, the size of the discrete problem to solve[7]. This type of convergence result is similar to the standard FETI-DP method [78]. Unfortunately, this result does not hold for the Helmholtz problem with $\beta = -k^2$ when the wavenumber $k > 0$ becomes large. However, unlike a standard FETI-DP method, it is possible to recover such behavior with our method by increasing parameter l as a function of mesh size h and wavenumber k.

The key is to control dispersion errors for the coarse problem

$$\hat{\underline{A}}\bar{\phi}_p = \hat{b}. \tag{10.98}$$

This coarse problem arises from (10.13) when ignoring the additional constraints $\hat{\underline{C}}\bar{\phi}_p = \hat{d}$ and dual variables $\bar{\nu}_d$. On a uniform mesh of infinite extent with mesh size h, this is possible as long as

$$l \geq \frac{1}{2}[kh + (kh)^{1/3} - 1]. \tag{10.99}$$

In practice, we set l to the ceiling of the right hand side of (10.99) since l must be an integer. In addition, we make sure that l is never zero so that the partial null space method actually provides a loosely coupled coarse problem (when $l = 0$, there is no coupling between elements in the coarse problem). The constraint (10.99) is based on Theorem 3.3 from [14]. There,

[7] In fact, this bound can be used to estimate the number of iterations required for the PCG iteration to meet a desired error tolerance; see Corollary 2.2.1 and Lemma 2.3.2 along with the discussion of Section 2.3 of [167]. In practice, since the condition number of the preconditioned system is small, I use the relative residual as a surrogate for the relative error measured in the $\underline{L}_P^T\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^T\underline{L}_P$ norm.

the criterion appears as

$$p \geq \frac{1}{2}[kh + (kh)^{1/3} - 1] \tag{10.100}$$

where p is the polynomial degree of the elements in the uniform mesh. The relation comes from analyzing the error in the discrete dispersion relation. This error enters a regime of superexponential decay only for p satisfying this criterion. If this criterion is not met, the error does not decay, and may even increase for particular values of p. In practice, we observe experimentally that a similar phenomenon holds when solving (10.98) on finite meshes, even when p is larger than l so long as criterion (10.99) holds. For non-conforming meshes, we choose l using the largest element size h in the mesh[8].

To illustrate this phenomenon, we consider a hypothetical Helmholtz problem with $\underline{\alpha} = \underline{I}$, $\beta = -k^2$, $k = 10.75$, and $f = 0$ on domain $\Omega = (-1, 1)^2$ with Dirichlet boundary conditions such that

$$p(\bar{x}) = -\frac{\jmath}{4} H_0^{(2)}(k\|\bar{x} - \bar{x}_c\|_2) \tag{10.101}$$

on $\partial\Omega$ with

$$\bar{x}_c = \begin{bmatrix} -2 \\ 1 \end{bmatrix}. \tag{10.102}$$

The solution $\phi(\bar{x})$ on Ω is $p(\bar{x})$. We subdivide Ω into four square elements of equal size using degree 64 polynomials on each element to represent ϕ. Figure 10.2 illustrates the computed solution solving (10.98) when l is set to five and six. Condition (10.99) is satisfied when $l \gtrsim 5.9785$ so that the coarse problem poorly approximates the true solution in the first case, but is suitable for the second. Notice that in both cases the solution is not continuous across the element boundaries, but that this discontinuity is much more pronounced in the $l = 5$ case. Figure 10.2 also illustrates the eigenvalues of the associated preconditioned problem. When l is less than five, these eigenvalues are less clustered (some are even negative) and their corresponding solutions are less effective at capturing the true solution. We have chosen to show the $l = 5$ case to demonstrate that condition (10.99) is relatively sharp.

Increasing parameter l increases the size of matrix $\underline{K}$ whose inverse applied to a vector is required when using PCG applied to (10.67). The size of matrix $\underline{K}$ reflects how connected the coarse problem is (recall that the coarse problem arises from imposing some, but not all, continuity conditions between elements) and, for this reason, we often refer to $\underline{K}$ as the coarse problem. In a sense, this method is similar to the FETI-DPH method [81] which is a modification of the FETI-DP method meant to address Helmholtz problems. In the FETI-DPH method, convergence for high frequency Helmholtz problems is achieved by adding additional constraints to the original FETI-DP formulation. In the method presented in this

[8]There may be room for substantial improvement regarding this choice.

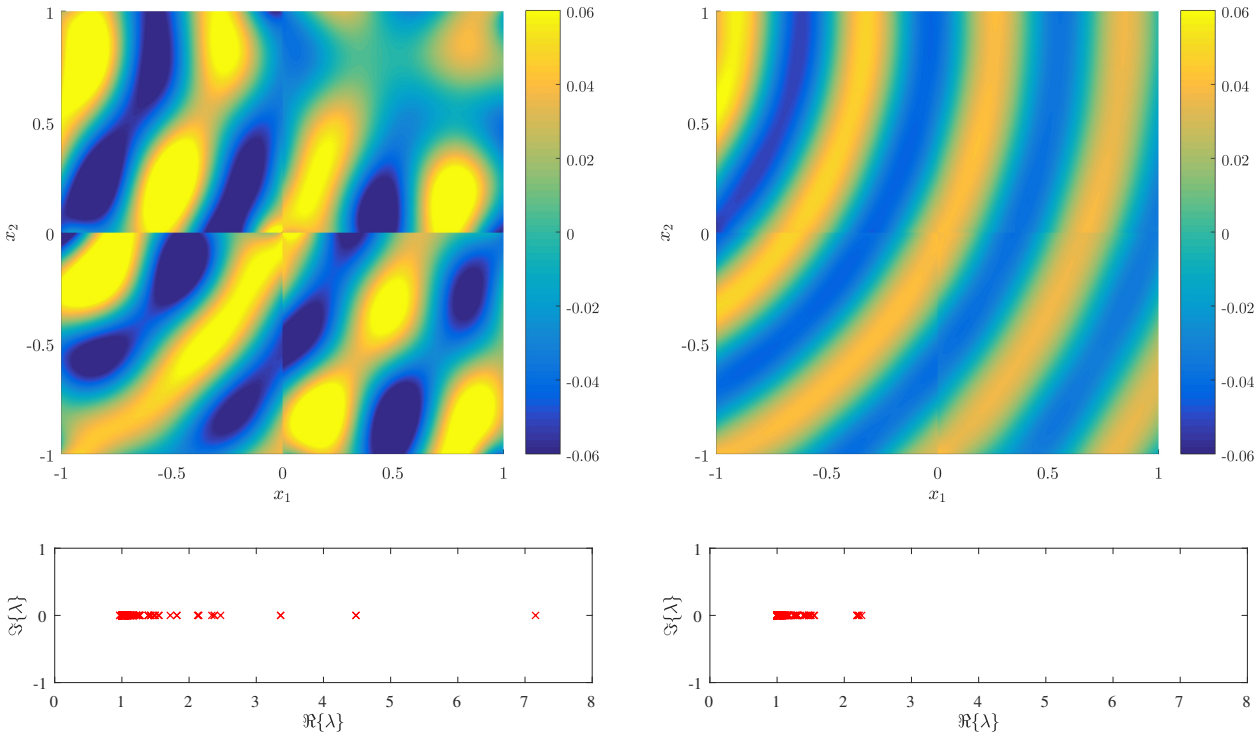

Figure 10.2: (top left) Real part of the solution to the coarse problem when $l = 5$. (top right) Real part of the solution to the coarse problem when $l = 6$. (bottom left) Eigenvalues of the preconditioned problem when $l = 5$. (bottom right) Eigenvalues of the preconditioned problem when $l = 6$.

thesis, we have a natural hierarchy of constraints along all edges that makes it straightforward to increase the connectedness of the coarse problem. Standard FETI-DP does not exploit such a hierarchy. Therefore, the FETI-DPH method appends constraint equations to $\underline{C}$ of the form $\underline{Q}_b^T \underline{C}$ which are then eliminated when applying the range space method (and consequently grow the size of the coarse problem). The matrix $\underline{Q}_b$ consists of plane waves sampled along edges in the domain decomposition. In practice, because it is unclear *a priori* how to choose the directions of these plane waves, many are chosen along each edge and rank-revealing QR factorizations are performed to select a set of orthogonal columns for $\underline{Q}_b$ such that it is not rank deficient.

As with the FETI-DPH method, increasing the size of the coarse problem is not the only change to the domain decomposition method required for the iterative method to converge. Equation (10.67) becomes indefinite for large enough k so that PCG should be replaced with the preconditioned generalized minimum residual method (PGMRES) or some other suitable Krylov subspace method (for example, the biconjugate gradient stabilized method) [167]. In addition, the domain decomposition method is susceptible to spurious resonant frequencies associated with each domain in the decomposition. This is problematic because

at such frequencies, the matrix $\underline{A}_{11}$ is singular, whereas the original saddle point system is not. One possible solution is to choose the domains in the decomposition small enough so that all domains have resonances at frequencies larger than k. This is the approach used in FETI-DPH [81].

Alternatively, instead of limiting the size of domains, it is possible to change the discretization in such a way that each subdomain problem becomes uniquely solvable without changing the primal solution [84, 85]. The idea is to add a Robin boundary condition with $\gamma = \pm \jmath k$ on a portion of each element's boundary. The signs are chosen so that on a shared edge between two elements, one element edge has positive sign while the other has negative sign. This creates a transmission boundary condition between the two elements.

To describe how this works within the framework of the current domain decomposition method, consider a domain Ω partitioned into two elements Ω_1 and Ω_2 that share a common boundary Γ. If ϕ_1 represents the solution in element 1 and ϕ_2 represents the solution in element 2, then continuity along Γ implies

$$\phi_1 = \phi_2 \tag{10.103}$$

and continuity of the normal flux implies

$$\begin{aligned} \bar{n}_1^T \underline{\alpha} \nabla \phi_1 &= \bar{n}_1^T \underline{\alpha} \nabla \phi_2 && (10.104) \\ &= -\bar{n}_2^T \underline{\alpha} \nabla \phi_2 && (10.105) \end{aligned}$$

where $\bar{n}_1$ denotes the unit outward normal from domain Ω_1 and $\bar{n}_2$ denotes the unit outward normal from domain Ω_2. In the standard finite element formulation, we have imposed the continuity constraint explicitly through constraint equations and imposed the normal flux continuity weakly. We can just as easily impose a linear combination of these two constraints weakly. As long as the strong continuity condition together with the new weak condition imply the original weak condition, the solution to the finite element problem will be the same. For example, adding $\jmath k$ times (10.103) to (10.104) yields

$$\underbrace{\bar{n}_1^T \underline{\alpha} \nabla \phi_1 + \jmath k \phi_1}_{q_1} = \underbrace{-\bar{n}_2^T \underline{\alpha} \nabla \phi_2 + \jmath k \phi_2}_{-q_2} . \tag{10.106}$$

Then (10.103) and (10.106) imply (10.104).

To see how such a condition changes the weak form, we start with

$$\int_\Omega \psi [-\nabla \cdot (\underline{\alpha} \nabla \phi) - k^2 \phi] d\Omega = \int_\Omega \psi f \, d\Omega \tag{10.107}$$

and split the integral over the two elements. Using the product rule and the divergence theorem for each domain yields

$$\int_{\Omega_1} \nabla\psi_1^T \underline{\alpha} \nabla\phi_1 - k^2\psi_1\phi_1 d\Omega - \oint_{\partial\Omega_1} \psi_1 \bar{n}_1^T \underline{\alpha} \nabla\phi_1 \, d\Omega$$
$$+ \int_{\Omega_2} \nabla\psi_2^T \underline{\alpha} \nabla\phi_2 - k^2\psi_2\phi_2 d\Omega - \oint_{\partial\Omega_2} \psi_2 \bar{n}_2^T \underline{\alpha} \nabla\phi_2 \, d\Omega = \int_{\Omega_1} \psi_1 f \, d\Omega + \int_{\Omega_2} \psi_2 f \, d\Omega. \quad (10.108)$$

Focusing on the boundary terms on the left hand side that are common to the shared edge Γ, we replace

$$\bar{n}_1^T \underline{\alpha} \nabla\phi_1 = q_1 - \jmath k \phi_1, \quad (10.109)$$
$$\bar{n}_2^T \underline{\alpha} \nabla\phi_2 = q_2 + \jmath k \phi_2, \quad (10.110)$$

as per (10.106). This gives

$$\int_\Gamma \psi_1 \bar{n}_1^T \underline{\alpha} \nabla\phi_1 \, d\Omega + \int_\Gamma \psi_2 \bar{n}_2^T \underline{\alpha} \nabla\phi_2 \, d\Omega = \int_\Gamma \psi_1 (q_1 - \jmath k \phi_1) \, d\Omega + \int_\Gamma \psi_2 (q_2 + \jmath k \phi_2) \, d\Omega \quad (10.111)$$

which we split into four integrals

$$\int_\Gamma \psi_1 \bar{n}_1^T \underline{\alpha} \nabla\phi_1 \, d\Omega + \int_\Gamma \psi_2 \bar{n}_2^T \underline{\alpha} \nabla\phi_2 \, d\Omega =$$
$$- \jmath k \int_\Gamma \psi_1 \phi_1 \, d\Omega + \jmath k \int_\Gamma \psi_2 \phi_2 \, d\Omega + \int_\Gamma \psi_1 q_1 \, d\Omega + \int_\Gamma \psi_2 q_2 \, d\Omega. \quad (10.112)$$

The first two terms on the right hand side are precisely the type of terms that contribute to the local operator matrices when imposing Robin boundary conditions (with parameters $\gamma_1 = \jmath k$ and $\gamma_2 = -\jmath k$). Unlike in the Robin boundary treatment of earlier chapters, we must have $q_1 = -q_2$ for the solution to exhibit the correct behavior at the boundary. In addition, we do not know the precise values of either q_1 or q_2, meaning that we cannot treat these terms as forcing terms like standard Robin boundary conditions. However, letting $\nu_{\text{new}} = q_1 = -q_2$, we note that

$$\int_\Gamma \psi_1 q_1 \, d\Omega + \oint_\Gamma \psi_2 q_2 \, d\Omega = \int_\Gamma \psi_1 \nu_{\text{new}} \, d\Omega - \int_\Gamma \psi_2 \nu_{\text{new}} \, d\Omega \quad (10.113)$$
$$= \int_\Gamma (\psi_1 - \psi_2) \nu_{\text{new}} \, d\Omega. \quad (10.114)$$

This is precisely of the same form as boundary terms obtained in the original weak formu-

lation where

$$\bar{n}_1^T \underline{\alpha} \nabla \phi_1 = \nu, \tag{10.115}$$

$$\bar{n}_2^T \underline{\alpha} \nabla \phi_2 = -\nu, \tag{10.116}$$

with ν_{new} replacing ν. Thus, to add the transmission condition to an edge between elements, we add a Robin term with $\gamma = \pm \jmath k$ to each element's local operator matrix $\underline{A}_j$ (the signs must be opposite to each other) and otherwise keep the saddle point system structure regarding constraint matrices unchanged. Notice that the meaning of the primal solution ϕ has not changed, but the Lagrange multiplier functions have changed so that $\nu_{\text{new}} = \nu + \jmath k \phi$.

When there are more than two elements, we must choose the signs for the Robin boundary coefficients appropriately so that every element has at least one edge where a transmission condition is imposed. In addition, we make sure to do so in such a way that no element has a mix of positive and negative coefficients. This is because the proof of uniqueness for the interior Helmholtz problem with real k depends on γ being a nonzero constant on the subset of the boundary where the Robin boundary condition is imposed. In particular, to see why, start with the Helmholtz equation

$$-\nabla \cdot \nabla \phi - k^2 \phi = f \qquad \text{in } \Omega \tag{10.117}$$

subject to boundary conditions

$$\phi = p \qquad \text{on } \Gamma_D, \tag{10.118}$$

$$\bar{n}^T \nabla \phi + \gamma \phi = q \qquad \text{on } \Gamma_R, \tag{10.119}$$

where Ω is a bounded domain and k is real. Assume two solutions ϕ_1 and ϕ_2 satisfy these three conditions and take their difference to obtain the homogeneous equation

$$-\nabla \cdot \nabla \phi_\star - k^2 \phi_\star = 0 \qquad \text{in } \Omega \tag{10.120}$$

subject to homogeneous boundary conditions

$$\phi_\star = 0 \qquad \text{on } \Gamma_D, \tag{10.121}$$

$$\bar{n}^T \nabla \phi_\star + \gamma \phi_\star = 0 \qquad \text{on } \Gamma_R, \tag{10.122}$$

where $\phi_\star = \phi_1 - \phi_2$. By showing that $\phi_\star = 0$, we find that $\phi_1 = \phi_2$ so that a solution to the original inhomogeneous Helmholtz problem is unique. To do so, we substitute $\phi_\star$ and $(\phi_\star)^*$

into Green's second identity where * denotes the complex conjugate. This gives

$$\int_\Omega \phi_\star \nabla \cdot \nabla (\phi_\star)^* - (\phi_\star)^* \nabla \cdot \nabla \phi_\star d\Omega = \oint_{\partial\Omega} \phi_\star \bar{n}^T \nabla (\phi_\star)^* - (\phi_\star)^* \bar{n}^T \nabla \phi_\star d\Omega. \tag{10.123}$$

Using (10.120) to remove the divergence and gradients on the left hand side and (10.121) on the right hand side gives

$$\int_\Omega \phi_\star (-k^2 \phi_\star)^* + (\phi_\star)^* (k^2 \phi_\star)\, d\Omega = \int_{\Gamma_R} \phi_\star \bar{n}^T \nabla (\phi_\star)^* - (\phi_\star)^* \bar{n}^T \nabla \phi_\star d\Omega \tag{10.124}$$

$$0 = \int_{\Gamma_R} \phi_\star \bar{n}^T \nabla (\phi_\star)^* - (\phi_\star)^* \bar{n}^T \nabla \phi_\star d\Omega. \tag{10.125}$$

Now using (10.122) to replace the normal derivatives yields

$$\int_{\Gamma_R} \phi_\star (-\gamma \phi_\star)^* + (\phi_\star)^* (\gamma \phi_\star)\, d\Omega = 0 \tag{10.126}$$

$$\int_{\Gamma_R} (\gamma - \gamma^*) |\phi_\star|^2 d\Omega = 0. \tag{10.127}$$

Since $\gamma - \gamma^* = 2\jmath \Im\{\gamma\}$ where $\Im\{\gamma\}$ is the imaginary part of γ, and γ is a constant of a single sign, we can factor it out of the integral to obtain

$$2\jmath \Im\{\gamma\} \int_{\Gamma_R} |\phi_\star|^2 d\Omega = 0 \tag{10.128}$$

which implies that $\phi_\star = 0$ on Γ_R if $\Im\{\gamma\} \neq 0$. If γ was not a constant of a single sign, we would not be able to factor it out of the integral to show that $\phi_\star = 0$ on a portion of the boundary. A similar substitution of (10.122) to replace $\phi_\star$ and $(\phi_\star)^*$ in (10.125) yields $\bar{n}^T \nabla \phi_\star = 0$ on Γ_R. Since both $\phi_\star$ and its normal derivative are zero on a portion of the boundary, Holmgren's theorem states that $\phi_\star$ must be identically zero inside Ω (see Theorems 2.1, 2.2, and 2.3 in [169] for a proof of this fact for the Helmholtz equation using its corresponding Green's representation formula). Since $\phi_\star = 0$ in Ω, this means that $\phi_1 = \phi_2$ for the inhomogeneous Helmholtz problem and that the solution is unique. Existence follows from the Green's representation formula.

There are different ways to ensure that each element in the mesh has at least one edge with a Robin boundary condition imposed and that all such Robin boundary conditions for a given element share the same sign. One method is described in [85]. A second method follows from the fact that a mesh can be represented as an undirected graph where each element corresponds to a vertex in the graph and adjacent elements are connected in the

graph via an edge. Assigning signs to all elements such that each element has at least one neighbor with a different sign is equivalent to the weak 2-coloring problem [170]. All graphs have a weak 2-coloring. One way to produce such a coloring (which is not unique) is to construct a spanning tree of the graph (choose an arbitrary vertex in the graph and perform a breadth-first-search). Label all elements in even levels of the tree with one sign and all other elements with the opposite sign (or vice versa). This produces the weak 2-coloring. When imposing the Robin boundary conditions, only do so along edges that are shared by elements of opposite sign and use the appropriate sign for $\gamma = \pm \jmath k$ that was determined in the coloring process. This algorithm guarantees that each local element problem is solvable. It should be noted that there are other methods involving two Lagrange multipliers that can also be used to guarantee uniqueness of the local element problems (the basic approach is described in [85]). However, we do not use such an approach because it modifies the structure of the constraint matrix and does not fit into the framework of the domain decomposition method described thus far.

The resulting saddle point system after adding Robin boundary terms at element interfaces possesses the same structure as (10.1). That is, only the $\underline{A}$ block has been modified by the Robin terms (all other terms remain unchanged, although the interpretation of the Lagrange multipliers has changed). However, such a modification makes an original real symmetric Helmholtz problem complex symmetric. In practice, applying the same domain decomposition method to this modified saddle point system requires roughly twice as many iterations to converge than for the unmodified saddle point system. However, the modified method converges even at spurious resonant frequencies where the unmodified approach can fail catastrophically. If one must use the modified approach for robust simulation, it is possible to increase l to reduce the number of iterations in the domain decomposition method at the cost of increasing the size of the coarse problem.

10.5 Convergence Tests

In the previous section, we have made some unsubstantiated claims regarding the convergence behavior of the domain decomposition method. In this section, we provide numerical evidence for these claims. We begin by demonstrating that the domain decomposition method behaves like the standard FETI-DP method [78] when applied to the Poisson equation but that it has the added flexibility to increase the size of the coarse problem which effectively reduces the number of iterations required for convergence. Similar tests are used for the Helmholtz equation to show that increasing the size of the coarse problem is necessary to retain small numbers of iterations when the wavenumber is increased. In the following

discussion, we use the zero vector for initial iterate in all applications of PCG or PGMRES.

10.5.1 Convergence Tests for the Poisson Equation

We begin by demonstrating the number of iterations required for the domain decomposition method to solve (6.1) with $\underline{\alpha} = \rho(\bar{x})\underline{I}$ and $\beta = 0$ on $\Omega = (-1, 1)^2$ subject to zero Dirichlet boundary conditions on $\partial\Omega$. We also report the maximum eigenvalue of the preconditioned system. We use a test problem based on the one presented in Table 2 of [78] where the element size and polynomial degree are varied. The test problem is not identical to that in [78] in the sense that we test higher polynomial degrees, and also larger numbers of elements (in part because of the quadtree structure of our mesh which forces certain numbers of elements when performing uniform refinement). This test shows that the number of iterations depends weakly on discontinuities in parameter ρ, the element degree, and the number of elements in the mesh. This favorable performance matches the performance presented in [78].

In our tests, the domain Ω is partitioned into a regular grid of $N = 2^{2n}$ elements (where $n = 1, 2, 3, 4, 5$) with each element possessing a local degree p expansion (where $p = 8, 16, 32, 64$). If the elements in the grid are numbered as $k = i + (j - 1)2^n$ for $i, j = 1, 2, ..., 2^n$, then the parameter ρ is chosen to be piecewise constant with corresponding value $\rho_{ij} = 10^{6(i-j)/32}$ over each element so that, on the finest grid, the maximum and minimum values of ρ are 10^6 and 10^{-6} respectively[9]. For all problems, we set the right hand side $\bar{b}$ randomly. That is, each entry is randomly sampled from the uniform distribution on the open interval $(0, 1)$. We use the Dirichlet preconditioner for $\underline{W}_1$ with diagonal scaling and report the number of iterations required for the relative residual to be reduced by ten orders of magnitude. We also compute the maximum eigenvalue $\lambda_{\max}$ of the preconditioned matrix $\underline{P}^{-1}\hat{\underline{C}}\hat{\underline{A}}^{-1}\hat{\underline{C}}^T$. In all tests, the smallest eigenvalue is 1 to within at least two digits so we do not list it. We repeat the test using parameter $l = 1, 2, 3, 4$. Table 10.1 shows the number of iterations and maximum eigenvalue for each combination of degree p, number of elements N, and parameter l.

We note that for a given l and p, the number of iterations converges to some fixed value as the number of elements increases. The same behavior is observed in the maximum eigenvalue which controls the range of the spectrum of the preconditioned system. In fact, the two quantities are intimately related. This is because, for any symmetric, positive definite

[9]This is another way in which our test differs from the one in [78]. In that paper, $\rho_{ij} = 10^{(i-j)/4}$. We have modified parameter ρ so that on our finest grid, the maximum and minimum values of ρ coincide with the maximum and minimum values on their finest grid. Their finest grid contains 576 elements whereas ours contains 1024.

Table 10.1: Number of iterations for PCG to reach convergence and maximum eigenvalue $\lambda_{\max}$ of the preconditioned system as functions of degree p, number of elements N, and parameter l for the Poisson test problem.

		$l=1$		$l=2$		$l=3$		$l=4$	
p	N	Iterations	$\lambda_{\max}$	Iterations	$\lambda_{\max}$	Iterations	$\lambda_{\max}$	Iterations	$\lambda_{\max}$
8	4	9	4.02	8	1.54	6	1.23	6	1.10
	16	18	4.91	11	1.70	8	1.30	6	1.12
	64	23	5.11	11	1.75	8	1.32	7	1.13
	256	23	5.16	11	1.76	8	1.33	7	1.13
	1024	23	5.18	11	1.76	8	1.33	7	1.13
16	4	11	5.84	10	2.08	8	1.61	7	1.38
	16	20	7.27	13	2.31	10	1.71	8	1.44
	64	27	7.62	13	2.37	11	1.75	9	1.45
	256	28	7.70	13	2.39	11	1.76	9	1.46
	1024	28	7.73	13	2.40	11	1.76	9	1.46
32	4	12	8.14	11	2.79	9	2.17	9	1.86
	16	22	10.25	15	3.08	12	2.30	10	1.94
	64	33	10.75	16	3.17	13	2.34	11	1.96
	256	33	10.87	15	3.19	13	2.36	11	1.97
	1024	33	10.91	16	3.20	13	2.36	11	1.97
64	4	16	10.94	12	3.67	11	2.91	10	2.51
	16	26	13.83	17	4.03	14	3.06	12	2.61
	64	38	14.53	18	4.13	15	3.11	13	2.64
	256	39	14.71	18	4.15	15	3.13	13	2.65
	1024	39	14.73	18	4.16	15	3.13	13	2.65

system

$$\underline{A}\bar{x} = \bar{b}, \tag{10.129}$$

the relative residual is related to the eigenvalues of $\underline{A}$ by

$$\frac{\|\bar{b} - \underline{A}\bar{x}_k\|_2}{\|\bar{b}\|_2} \leq \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} \max_{z \in \sigma(\underline{A})} |p_k(z)| \tag{10.130}$$

where $\bar{x}_k$ is the kth iterate, $\sigma(\underline{A})$ is the set of all eigenvalues of $\underline{A}$, and p_k is any degree k polynomial such that $p_k(0) = 1$ (combine Corollary 2.2.1 with Lemma 2.3.2 in [167], for example). This inequality holds when $\bar{x}_0 = 0$ (which is the case in all of our examples) and is a statement related to the fact that the relative residual measured in the 2-norm is less than the square root of the condition number of $\underline{A}$ measured in the 2-norm times the relative error in the solution measured in the $\underline{A}$ norm. In our problem, $\underline{A}$ is the preconditioned

matrix $\underline{L}_P^T \hat{\underline{C}} \hat{\underline{A}}^{-1} \hat{\underline{C}}^T \underline{L}_P$ with corresponding right hand side. In such a scenario, $\lambda_{\min} = 1$ and all eigenvalues are contained within the interval $[1, \lambda_{\max}]$ so that

$$\frac{\|\bar{b} - \underline{A}\bar{x}_k\|_2}{\|\bar{b}\|_2} \leq \sqrt{\lambda_{\max}} \max_{z \in [1, \lambda_{\max}]} |p_k(z)|. \tag{10.131}$$

We can estimate the size of the relative residual by choosing the polynomial

$$p_k(z) = \frac{(\frac{\lambda_{\max}+1}{2} - z)^k}{(\frac{\lambda_{\max}+1}{2})^k} \tag{10.132}$$

which is zero at the midpoint of the interval $[1, \lambda_{\max}]$, and satisfies $p_k(0) = 1$. This polynomial has its maximum absolute value at either endpoint of the interval, meaning that

$$\max_{z \in [1, \lambda_{\max}]} |p_k(z)| = \left| \frac{(\frac{\lambda_{\max}-1}{2})^k}{(\frac{\lambda_{\max}+1}{2})^k} \right| \tag{10.133}$$

$$= \left| \frac{\lambda_{\max} - 1}{\lambda_{\max} + 1} \right|^k. \tag{10.134}$$

Thus

$$\frac{\|\bar{b} - \underline{A}\bar{x}_k\|_2}{\|\bar{b}\|_2} \leq \sqrt{\lambda_{\max}} \left| \frac{\lambda_{\max} - 1}{\lambda_{\max} + 1} \right|^k. \tag{10.135}$$

The right hand side of this inequality is a monotonically increasing function of $\lambda_{\max} \geq 1$ for any k. Thus the smaller $\lambda_{\max}$ is, the smaller the relative residual is after k iterations. Since our stopping criterion is based on the relative residual being smaller than 10^{-10}, this shows that the maximum eigenvalue controls the number of iterations. It is possible to improve this estimate [167] to obtain

$$\frac{\|\bar{b} - \underline{A}\bar{x}_k\|_2}{\|\bar{b}\|_2} \leq 2\sqrt{\lambda_{\max}} \left| \frac{\sqrt{\lambda_{\max}} - 1}{\sqrt{\lambda_{\max}} + 1} \right|^k \tag{10.136}$$

although the conclusion is the same in both cases.

Returning to Table 10.1, we also note that for fixed p and N, the number of iterations can be reduced by increasing l (although the cost of each iteration increases because increasing l increases the size of matrix $\underline{K}$ which must be factored). If N and l are fixed, increasing p leads to a mild increase in the number of iterations. To explain this increase, we consider another test based on Table 6 and Figure 4 in [78]. This test demonstrates that (10.97) holds. In particular, we set $\underline{\alpha} = \underline{I}$ and fix $n = 6$. We then compute $\lambda_{\max}$ for p ranging from 6 to 32 in increments of 2. We repeat this process for $l = 1, 2, 3, 4$. Figure 10.3 illustrates how $\lambda_{\max}$ increases as a function of degree p for each possible l. The figure also illustrates

least squares fits to the data such that

$$\sqrt{\lambda_{\max}} \approx A + B \log_{10}(p^2) \tag{10.137}$$

where we only use data with $p \in \{14, 16, ..., 32\}$ to obtain the fit. We do this because the data for $l = 3, 4$, does not appear to have entered the asymptotic regime for smaller values of p. Having computed parameters A and B for the least squares fit, we can change the base of the logarithm to determine constant C in (10.97). To do so,

$$A + B \log_{10}(p^2) = A + A\frac{B}{A} \log_{10}(p^2) \tag{10.138}$$

$$= A\left[1 + \frac{B}{A} \log_{10}(p^2)\right] \tag{10.139}$$

$$= A\left[1 + \frac{B}{A} \log_{10}(p^2)\frac{\log_{10}(D)}{\log_{10}(D)}\right] \tag{10.140}$$

$$= A\left[1 + \frac{B}{A} \log_{10}(D)\frac{\log_{10}(p^2)}{\log_{10}(D)}\right]. \tag{10.141}$$

Since

$$\frac{\log_{10}(p^2)}{\log_{10}(D)} = \log_D(p^2), \tag{10.142}$$

we choose D such that

$$\frac{B}{A} \log_{10}(D) = 1. \tag{10.143}$$

This gives

$$D = 10^{A/B}. \tag{10.144}$$

Having chosen the base D, we find that (10.137) becomes

$$\sqrt{\lambda_{\max}} \approx A\left[1 + \log_D(p^2)\right] \tag{10.145}$$

$$\lambda_{\max} \approx A^2\left[1 + \log_D(p^2)\right]^2. \tag{10.146}$$

Since $\lambda_{\min} = 1$,

$$\kappa_2(\underline{L}_P^T \underline{\hat{C}}\, \underline{\hat{A}}^{-1} \underline{\hat{C}}^T \underline{L}_P) = \frac{\lambda_{\max}}{\lambda_{\min}} = \lambda_{\max} \tag{10.147}$$

so that by (10.146), $C = A^2$ in (10.97). The least squares fits become

$$\kappa_2(\underline{L}_P^T \underline{\hat{C}}\, \underline{\hat{A}}^{-1} \underline{\hat{C}}^T \underline{L}_P) \approx C(1 + \log_D(p^2))^2 \tag{10.148}$$

with constants $C \approx 0.473, 0.354, 0.244, 0.191$ and bases $D \approx 6.15, 31.6, 26.8, 23.1$ for $l =$

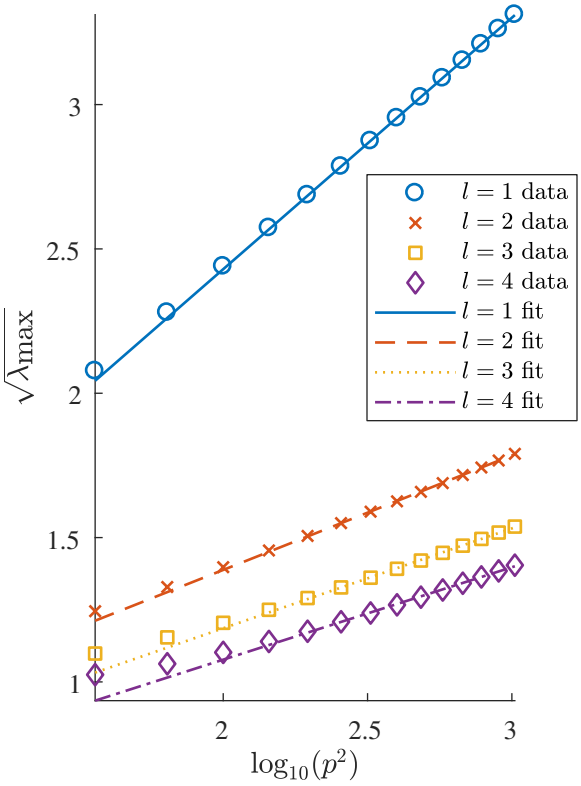

Figure 10.3: Dependence of the maximum eigenvalue of the preconditioned system $\lambda_{\max}$ on polynomial degree p for the Poisson problem as a function of parameter l. Lines represent linear least squares fits to the data ignoring the first 4 data points from each set.

$1, 2, 3, 4$, respectively.

Note that increasing l from 1 to 2 appears to change the slope of the fits in Figure 10.3 but that further increases in l tend primarily to decrease the offset of the fits. This is reflected in constants C and D as well as in the number of iterations in Table 10.1. That is, the number of iterations are roughly halved when changing l from 1 to 2 but only decrease mildly when going from 2 to 3 or 4. This means that for relatively low degree p, there is little value in increasing l beyond 2 when solving Poisson problems. This result is not true when solving Helmholtz problems.

10.5.2 Convergence Tests for the Helmholtz Equation

We now perform similar tests for the Helmholtz equation and address the added complexities involved. We solve (6.1) with $\underline{\alpha} = \underline{I}$, $\beta = -k^2$, and $f = 0$ on $\Omega = (-1, 1)^2$ subject to the same Dirichlet boundary condition on $\partial\Omega$ described in Section 10.4. We use the same discretizations as described in Section 10.5.1 and vary element degree p, number of elements N, and parameter l in the same ways. We repeat this process for several values of k from

Table 10.2: Number of iterations for PGMRES to reach convergence as a function of degree p, number of elements N, parameter l, and wavenumber k, for the Helmholtz test problem.

		$k=11$				$k=21$				$k=31$			
p	N	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$	$l=1$	$l=2$	$l=3$	$l=4$
8	4	61	42	29	22	69	58	47	35	72	58	50	40
	16	79	42	20	**9**	-	-	69	42	-	-	-	95
	64	88	**27**	**10**	**7**	-	-	**30**	**10**	-	-	98	34
	256	**-**	**15**	**9**	**7**	-	**51**	**11**	**7**	-	-	**26**	**9**
	1024	**-**	**12**	**8**	**6**	**-**	**21**	**9**	**7**	-	**54**	**11**	**7**
16	4	65	45	32	24	-	93	76	61	-	-	-	-
	16	85	43	21	**12**	-	-	73	46	-	-	-	-
	64	-	**32**	**13**	**9**	-	-	**33**	**12**	-	-	-	36
	256	**-**	**19**	**11**	**9**	-	**62**	**13**	**9**	-	-	**30**	**11**
	1024	**-**	**15**	**11**	**8**	**-**	**26**	**11**	**9**	-	**67**	**13**	**9**
32	4	67	46	34	25	-	94	78	63	-	-	-	-
	16	91	46	24	**14**	-	-	78	49	-	-	-	-
	64	-	**34**	**15**	**11**	-	-	**37**	**15**	-	-	-	38
	256	**-**	**22**	**13**	**11**	-	**72**	**15**	**11**	-	-	**33**	**13**
	1024	**-**	**17**	**13**	**11**	**-**	**31**	**13**	**11**	-	**79**	**15**	**11**
64	4	69	47	35	26	-	95	80	64	-	-	-	-
	16	-	53	27	**15**	-	-	83	52	-	-	-	-
	64	-	**38**	**16**	**14**	-	-	**42**	**17**	-	-	-	43
	256	**-**	**25**	**15**	**13**	-	**75**	**18**	**14**	-	-	**37**	**16**
	1024	**-**	**20**	**15**	**12**	**-**	**36**	**15**	**13**	-	**89**	**18**	**13**

1 to 101 in increments of 10. We do not use spatially varying $\underline{\alpha}$ so that the effect of k on the number of iterations is clear. Table 10.2 shows the number of iterations required to reduce the relative residual by ten orders of magnitude for the cases $k = 11, 21, 31$. We use the unmodified domain decomposition method without adding Robin boundary terms. Problems where the number of iterations exceeds 100 are marked with a dash. Problems where condition (10.99) is satisfied are marked in bold. In Table 10.2, we do not show the eigenvalue with maximum absolute value because it is not necessarily an indicator of the condition number of the preconditioned system now that the problem can be indefinite.

We note that when N is small (the first and sometimes second row for each new value of p in Table 10.2) the method converges but requires a relatively large number of iterations. This is not the regime we are interested in because the solutions corresponding to these simulations tend not to be accurate. For high accuracy, condition (10.99) does a good job of predicting which problems tend to converge with a small number of iterations. However,

there is a small number of cases where this criterion is met, but the method fails to converge in under 100 iterations. In our data set (which includes the data shown in Table 10.2 along with similar data for k up to 101) there are 26 such failures out of a total of 252 cases where l meets the criterion. It is interesting to note that, in every failure case, increasing l by 1 leads to convergence within 100 iterations. We attribute these failures to the fact that criterion (10.99) is based on dispersion analysis for an infinite grid and holds only when $kh \gg 1$ [14]. For a more refined criterion, the same paper contains the exact dispersion relation on an infinite grid. We find that the numbers of iterations in Table 10.2 are small wherever the dispersion errors are small (taking care to replace p by l in the exact dispersion relation, as was done in Section 10.4). These observations suggest that for the preconditioner to be effective, dispersion errors must be controlled for the coarse problem. Similar behavior is observed using criteria from [17] where dispersion error is controlled on finite unstructured grids (although the criteria depend on two implicit constants).

As a final comment on the data in Table 10.2, it is important to consider the dispersion errors of the coarse problem rather than the global problem when determining the efficacy of the preconditioner. For example, if we use criterion (10.100) instead of (10.99) to choose when to use the domain decomposition method, then there are 445 tests which fail to converge out of a total of 724 cases satisfying the global dispersion criterion. Increasing l by 1 in such failure cases does not lead to convergence.

The data in Table 10.2 was collected using grids and wavenumbers that avoid spurious resonant frequencies. In our last convergence test, we construct an example to illustrate that spurious resonant frequencies can adversely affect convergence. First, we solve the same PDE as in the previous test, but with $N = 64$, $p = 32$, $l = 3$, and $k = 16.55$ with both the unmodified domain decomposition method and the modified method with added Robin boundary terms. The wavenumber is close to, but not exactly, the spurious resonant frequency $k \approx 16.56157163134991$ (which was computed numerically). Parameter l was chosen to satisfy (10.99). Figure 10.4 illustrates the relative residual as a function of number of iterations for both methods. We terminate both methods when the relative residual has decreased by ten orders of magnitude. The behavior in Figure 10.4 is typical in the sense that the modified method requires more iterations to converge than the unmodified method (roughly twice as many). In both cases, the error in the computed solution measured in the H^1 norm is on the order of 10^{-16}.

Next, we repeat the same experiment but with k set to the spurious resonant frequency. The unmodified method fails catastrophically after one iteration and returns a solution with error on the order of 10^5. Additional iterations can reduce the norm of the preconditioned residual but do not succeed in reducing the error. The modified method converges largely in

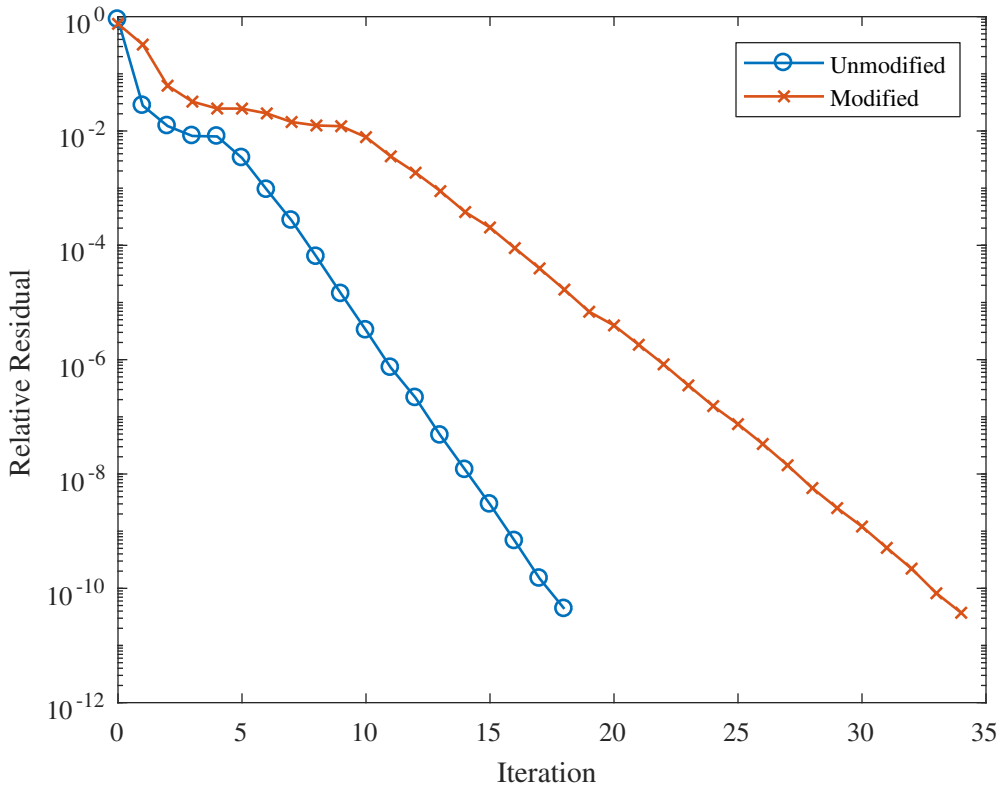

Figure 10.4: Preconditioned relative residual $\|\underline{P}^{-1}\bar{r}_k\|_2/\|\underline{P}^{-1}\bar{r}_0\|_2$ versus iteration number for the unmodified and modified domain decomposition method applied to a test Helmholtz problem. Vector $\bar{r}_k$ is the residual vector corresponding to (10.67) at the kth iteration.

the same way as depicted in Figure 10.4 with a solution whose error is on the order of 10^{-16}.

10.6 Unbounded Time-Harmonic Scattering Examples

In this section, we apply the domain decomposition method to challenging Helmholtz problems. We use the unmodified domain decomposition method with $\underline{W}_1$ set to the Dirichlet preconditioner with no diagonal scaling rather than the modified method with added Robin boundary conditions because, in practice, using the method at spurious resonant frequencies is rare. In addition, in light of the discussion in the previous section regarding criterion (10.99), we choose parameter l such that

$$l \geq \frac{1}{2}[kh + (kh)^{1/3} - 1] + 1 \tag{10.149}$$

where we have built in a safety factor of 1 with h chosen as the longest edge length in the mesh and l rounded up to the nearest integer greater than or equal to 1. Unlike with the

convergence tests in the previous section, all examples in this section use non-conforming meshes.

For the first four examples, we solve unbounded two-dimensional TM_{x_3} mode electromagnetic scattering problems. We assume that $\underline{\epsilon}$ and $\underline{\mu}$ are diagonal but spatially varying. The derivation of the governing PDE for the TM_{x_3} mode mirrors the derivation for the TE_{x_3} mode presented in Section 9.6.2. Rather than repeat a similar derivation, exchanging the roles of $\bar{E}$ with $\bar{H}$ and $\underline{\epsilon}$ with $\underline{\mu}$ in (9.176) gives

$$-\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{22}}\frac{\partial E_{x_3}}{\partial x_1}\right)-\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{11}}\frac{\partial E_{x_3}}{\partial x_2}\right)=\omega^2\epsilon_{33}E_{x_3} \tag{10.150}$$

which is the Helmholtz equation governing the TM_{x_3} mode. As in Section 9.6.2, we use the same change of variables (9.177) to make quantities in (10.150) dimensionless. We also assume that the total field $E_{x_3} = E_i + E_s$ can be written as the sum of a known incident field E_i and unknown scattered field E_s. Substituting this expression into (10.150) yields

$$-\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{22}}\frac{\partial E_s}{\partial x_1}\right)-\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{11}}\frac{\partial E_s}{\partial x_2}\right)-\omega^2\epsilon_{33}E_s=\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{22}}\frac{\partial E_i}{\partial x_1}\right)+\frac{\partial}{\partial x_1}\left(\frac{1}{\mu_{11}}\frac{\partial E_i}{\partial x_2}\right)+\omega^2\epsilon_{33}E_i \tag{10.151}$$

which can be written in the generic form

$$-\nabla\cdot(\underline{\alpha}\nabla\phi)+\beta\phi=f \tag{10.152}$$

with

$$\underline{\alpha}=\begin{bmatrix}\frac{1}{\mu_{22}} & 0\\ 0 & \frac{1}{\mu_{11}}\end{bmatrix},\qquad \beta=-\omega^2\epsilon_{33},\qquad f=\nabla\cdot(\underline{\alpha}\nabla E_i)-\beta E_i,\qquad \phi=E_s. \tag{10.153}$$

Since the problems are unbounded, we use the same PML technique as described in Section 9.6.1 to solve these unbounded problems on bounded domains.

10.6.1 Dielectric Cylinder

We begin with a classical example of scattering from an infinitely long dielectric cylinder [8] which we use to demonstrate the accuracy of the method for a non-conforming mesh. The cylinder is coaxial with the x_3 axis, has radius $a = 1$, and is enclosed inside domain $\Omega = (-4, 4)^2$. We choose $\epsilon_{33} = 4$ inside the cylinder and $e_{33} = 1$ outside. Both $\mu_{11} = \mu_{22} = 1$

throughout the domain. We select an incident plane wave

$$E_i = e^{-\jmath\omega\bar{k}^T\bar{x}} \tag{10.154}$$

with unit vector $\bar{k}$ so that

$$f = \omega^2(\epsilon_{33} - \bar{k}^T\underline{\alpha}\bar{k})E_i. \tag{10.155}$$

The forcing function f is only non-zero inside the cylinder since outside $\epsilon_{33} = 1$ and $\bar{k}^T\underline{\alpha}\bar{k} = 1$. We choose $\bar{k} = \bar{e}_1$ corresponding to propagation of the incident field in the positive x_1 direction and set $\omega = 4 \cdot 2\pi$ so that the wavelength outside the cylinder is $\lambda = 1/4$. This means that Ω is $32\lambda \times 32\lambda$ in size.

To absorb the scattered wave, we use PML with thicknesses $w_i = 1$ and decay rates $\sigma_i = 15$. The PML modify ϵ_{33}, μ_{11}, and μ_{22} wherever σ_i are nonzero. Decay rate σ_1 is nonzero in the two bands $x_1 \in (-4, -3)$ and $x_1 \in (3, 4)$ whereas σ_2 is nonzero in bands $x_2 \in (-4, -3)$ and $x_2 \in (3, 4)$. The mesh (shown in Figure 10.5) is constructed as in Section 9.1 using 6 levels of refinement as a maximum level and 4 levels of refinement as a minimum level. All boundary projections and Legendre expansions are computed to a tolerance of 10^{-12}. Each element in the mesh uses a degree 32 polynomial expansion to represent the scattered field. We solve the discrete problem using the domain decomposition method and terminate the iterative process when the preconditioned relative residual has decreased by 10 orders of magnitude.

Figure 10.5 illustrates the relative residual reduction as a function of iteration number for this problem. There is a total of 640,332 unknowns in $\bar{\phi}$ and the coarse problem matrix $\underline{K}$ is square with 9,909 rows. Factorization of this matrix represents the main bottleneck of the domain decomposition method for high frequency problems because the size of the coarse problem grows as the frequency is increased. This is because l is one factor that determines the size of $\underline{K}$ and l must grow as the frequency grows to continue to satisfy the dispersion criterion (10.149).

Figure 10.6 shows the real part of the computed scattered field E_s and the associated logarithm of the pointwise error $|E_s - E_{s,\text{exact}}|$. The exact solution is computed using the classical method of eigenfunction expansions (see, for example, [8]) using the sum

$$E_{s,\text{exact}}(\rho, \varphi) = \begin{cases} 2\sum_{i=0}^{\infty}{}' a_i H_i^{(2)}(\omega\rho)\cos(i\varphi) & \rho \geq a \\ -E_i + 2\sum_{i=0}^{\infty}{}' c_i J_i(\omega_r\rho)\cos(i\varphi) & \rho < a \end{cases} \tag{10.156}$$

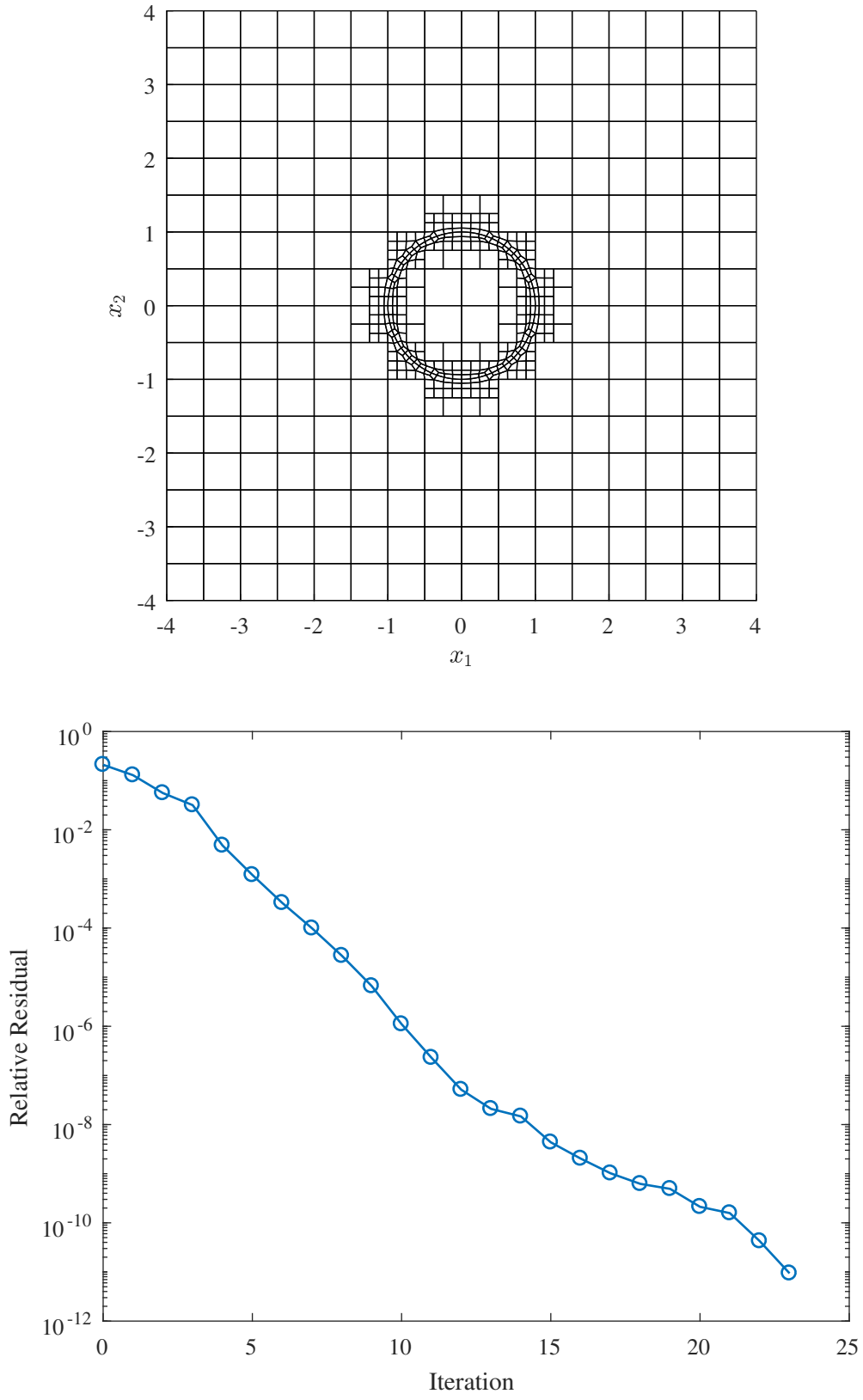

Figure 10.5: (top) Non-conforming mesh used to solve the dielectric cylinder problem. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the dielectric cylinder problem.

where

$$a_i = -\jmath^{-i} \frac{J_i'(\omega a) J_i(\omega_r a) - \sqrt{\epsilon_r} J_i(\omega a) J_i'(\omega_r a)}{H_i^{(2)\prime}(\omega a) J_i(\omega_r a) - \sqrt{\epsilon_r} H_i^{(2)}(\omega a) J_i'(\omega_r a)}, \tag{10.157}$$

$$c_i = \frac{\jmath^{-(i+1)}}{\pi \omega a} \frac{2}{H_i^{(2)\prime}(\omega a) J_i(\omega_r a) - \sqrt{\epsilon_r} H_i^{(2)}(\omega a) J_i'(\omega_r a)}, \tag{10.158}$$

and $\omega_r = \sqrt{\epsilon_r}\omega$, $\epsilon_r = 4$, $\rho = \|\bar{x}\|_2$, and $\varphi = \text{atan2}(x_2, x_1)$. Recall that J_i is the order i Bessel function of the first kind and $H_i^{(2)}$ is the order i Hankel function of the second kind. Primes on the Bessel and Hankel functions denote the derivative with respect to their arguments, and primes on the summation symbols require halving the $i = 0$ term. In computing the error, we use 75 terms in the expansion which is enough to have the series converge in a disk large enough to include domain Ω. In both the solution and error figures, we have shown the PML region to emphasize that the scattered field decays to zero (in most later examples, we do not show the PML since the solution is not physically relevant there). Note that the error is computed to approximately 10 digits of accuracy throughout the physical domain. The error in the PML is large, but this is expected because the computed field is unphysical there. To achieve such high accuracy, it is necessary to capture the curvilinear boundary and to use PML with parameters w_i and σ_i chosen appropriately (as we have done here).

In addition to the scattered field, Figure 10.7 illustrates the computed RCS of the dielectric cylinder and its corresponding error, each computed at 1000 evenly spaced observation angles. The RCS is given by

$$\sigma_{2\text{D}}(\varphi, \varphi_i) = \lim_{\|\bar{x}\|_2 \to \infty} 2\pi \|\bar{x}\|_2 \frac{|E_s|^2}{|E_i|^2} \tag{10.159}$$

$$= \frac{\omega}{4} |E_{\text{far}}|^2 \tag{10.160}$$

with the far field integral

$$E_{\text{far}}(\varphi) = \oint_\Gamma [\hat{x}^T \bar{n}' E_s(\bar{x}') - (\jmath\omega)^{-1} \bar{n}'^T \nabla' E_s(\bar{x}')] e^{\jmath\omega \hat{x}^T \bar{x}'} d\Omega' \tag{10.161}$$

and

$$\hat{x} = \begin{bmatrix} \cos(\varphi) \\ \sin(\varphi) \end{bmatrix}. \tag{10.162}$$

This RCS is effectively the same as the one presented in Section 9.6.2 with $\bar{E}$ replacing $\bar{H}$. We integrate over the boundary Γ of the cylinder. Like in Section 9.6.2, the dielectric cylinder is invariant under rotations about the x_3 axis so that the RCS need only be computed for a single incident angle φ_i. Therefore, the solution in Figure 10.6 is used to compute the RCS

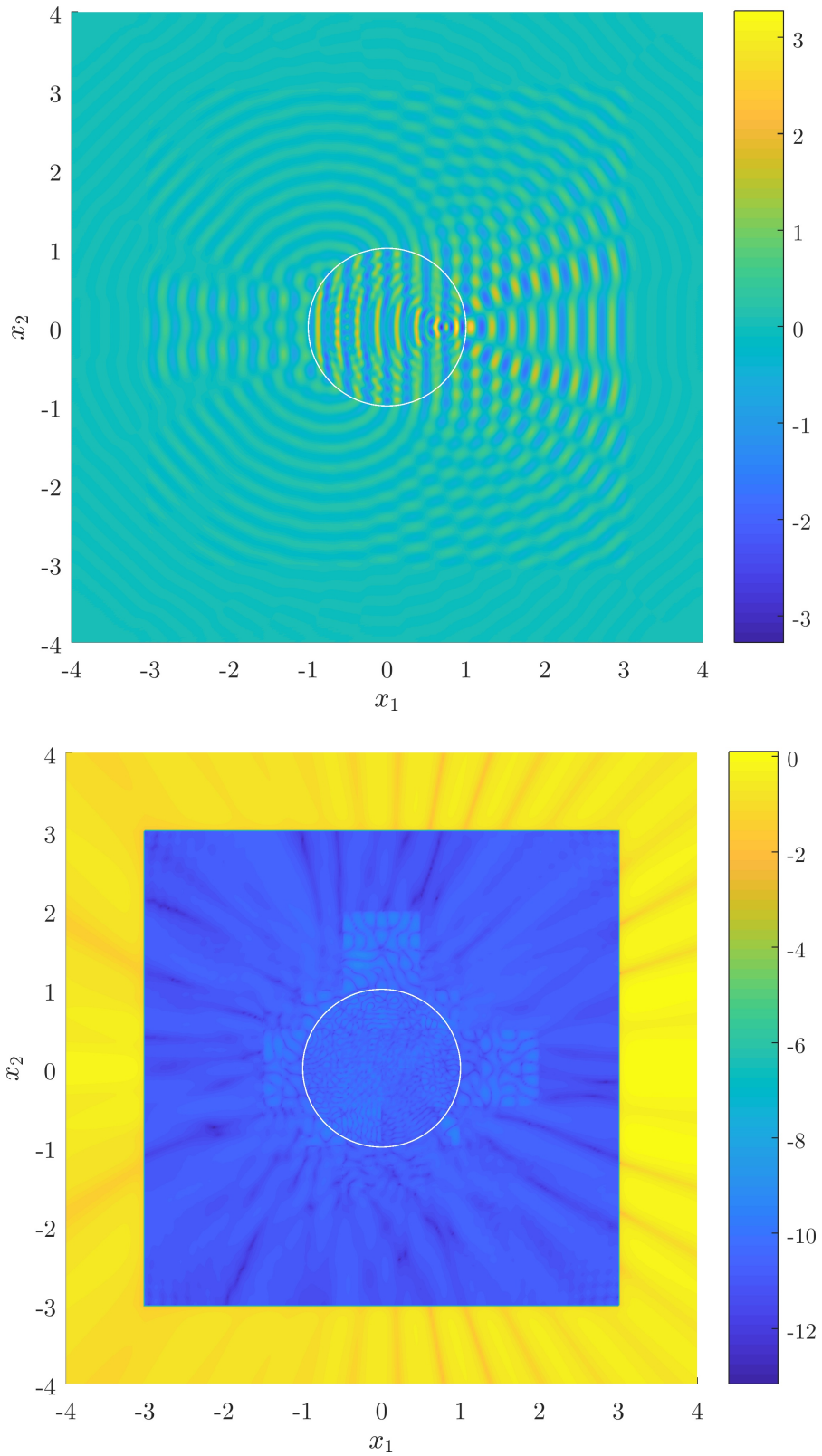

Figure 10.6: (top) Real part of the scattered field $\Re\{E_s\}$ for the dielectric cylinder problem. (bottom) Logarithm of the absolute value of the error in the scattered field $\log_{10}|E_s - E_{s,\text{exact}}|$ for the dielectric cylinder problem.

shown in Figure 10.7. The computed RCS and the exact RCS are calculated in the same way as in Section 9.6.2. Figure 10.7 shows that the RCS determined from the computed scattered field is accurate to roughly 9 digits.

10.6.2 Luneburg Lens

For the next two examples, we consider lens problems based on those in [171]. We use these examples to demonstrate that the method can be applied to problems with more complicated spatial variation of permittivity. In addition, we show how the method behaves when the frequency of the problem is high (respectively, the wavelength is small) compared to the size of the domain. First, we consider a two-dimensional Luneburg lens which has spatially varying permittivity that focuses incident plane waves to a single point. We solve for the scattered field in the TM_{x_3} mode due to the same incident field as in Section 10.6.1 but now with frequency $\omega = 24 \cdot 2\pi$ so that the wavelength of the incident field is $\lambda = 1/24$. We solve for the scattered field in domain $\Omega = (-1.25, 1.25)^2$ which is $60\lambda \times 60\lambda$ in size. The permittivity of the lens is given by

$$\epsilon_{33} = \begin{cases} 2 - \left(\dfrac{\|\bar{x}\|_2}{a} \right)^2 & \|\bar{x}\|_2 \leq a \\ 1 & \text{otherwise} \end{cases} \tag{10.163}$$

with radius $a = 0.45$. The PML used to terminate the domain have thicknesses $w_i = 0.3125$ and decay rates $\sigma_i = 44$ in the bands where the PML are nonzero. All elements use degree 20 expansions. Otherwise, all other parameters are the same as in Section 10.6.1.

Figure 10.8 illustrates the real part of the computed total field $E_{x_3} = E_i + E_s$ for the Luneburg lens. The PML region is not shown. Clearly, the plane wave has been focused to the point

$$\bar{x} = \begin{bmatrix} a \\ 0 \end{bmatrix}. \tag{10.164}$$

The figure also shows the preconditioned relative residual as a function of the number of iterations of the domain decomposition method. The criterion (10.149) leads to a small number of iterations. There are 310,464 unknowns in $\bar{\phi}$ with 18,281 rows in the coarse problem matrix $\underline{K}$. Compared to the coarse problem in Section 10.6.1, $\underline{K}$ is now roughly twice as large because of the near doubling of the problem size in terms of wavelength λ.

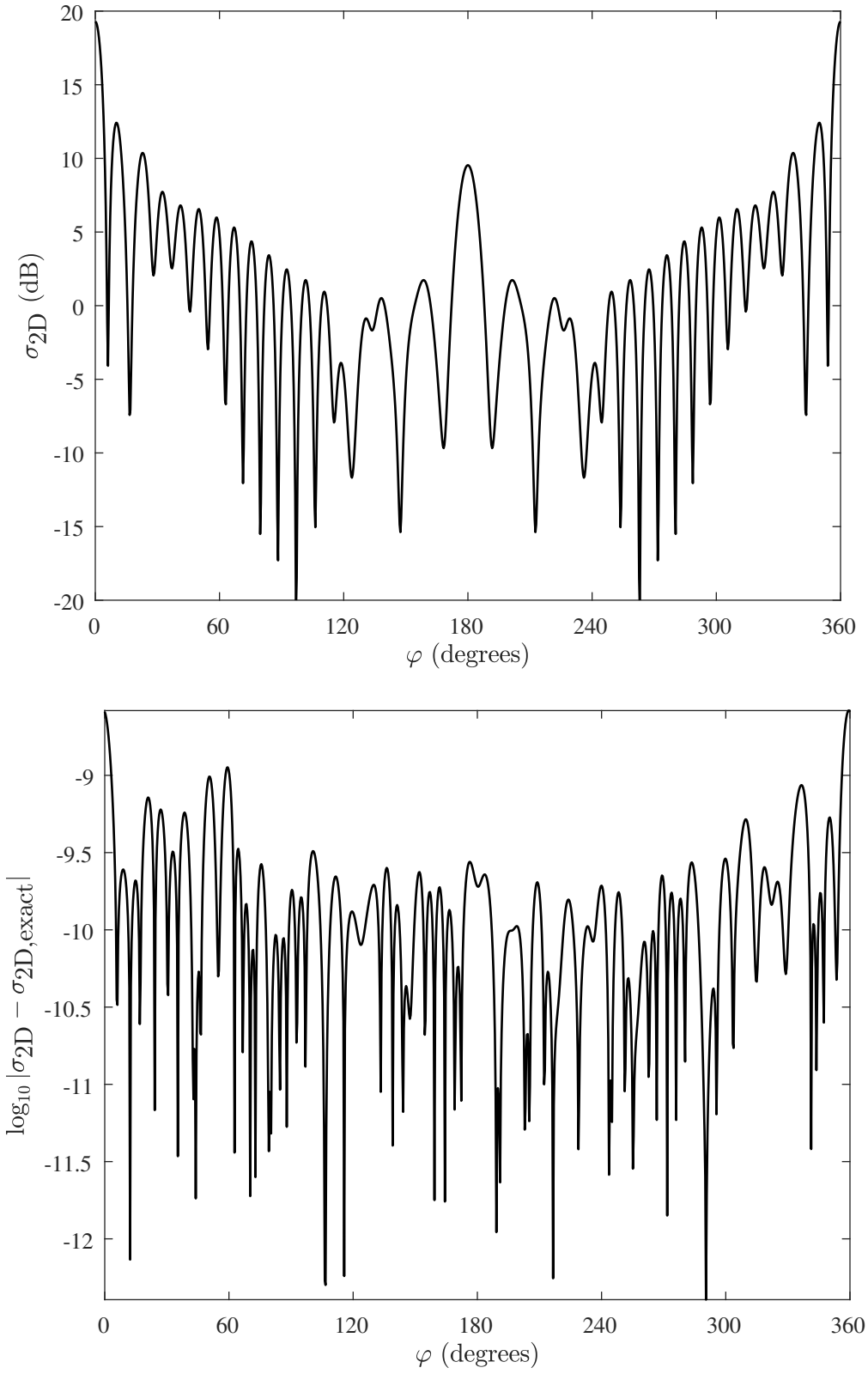

Figure 10.7: (top) RCS of the dielectric cylinder. (bottom) Logarithm of the absolute value of the error in the RCS of the dielectric cylinder.

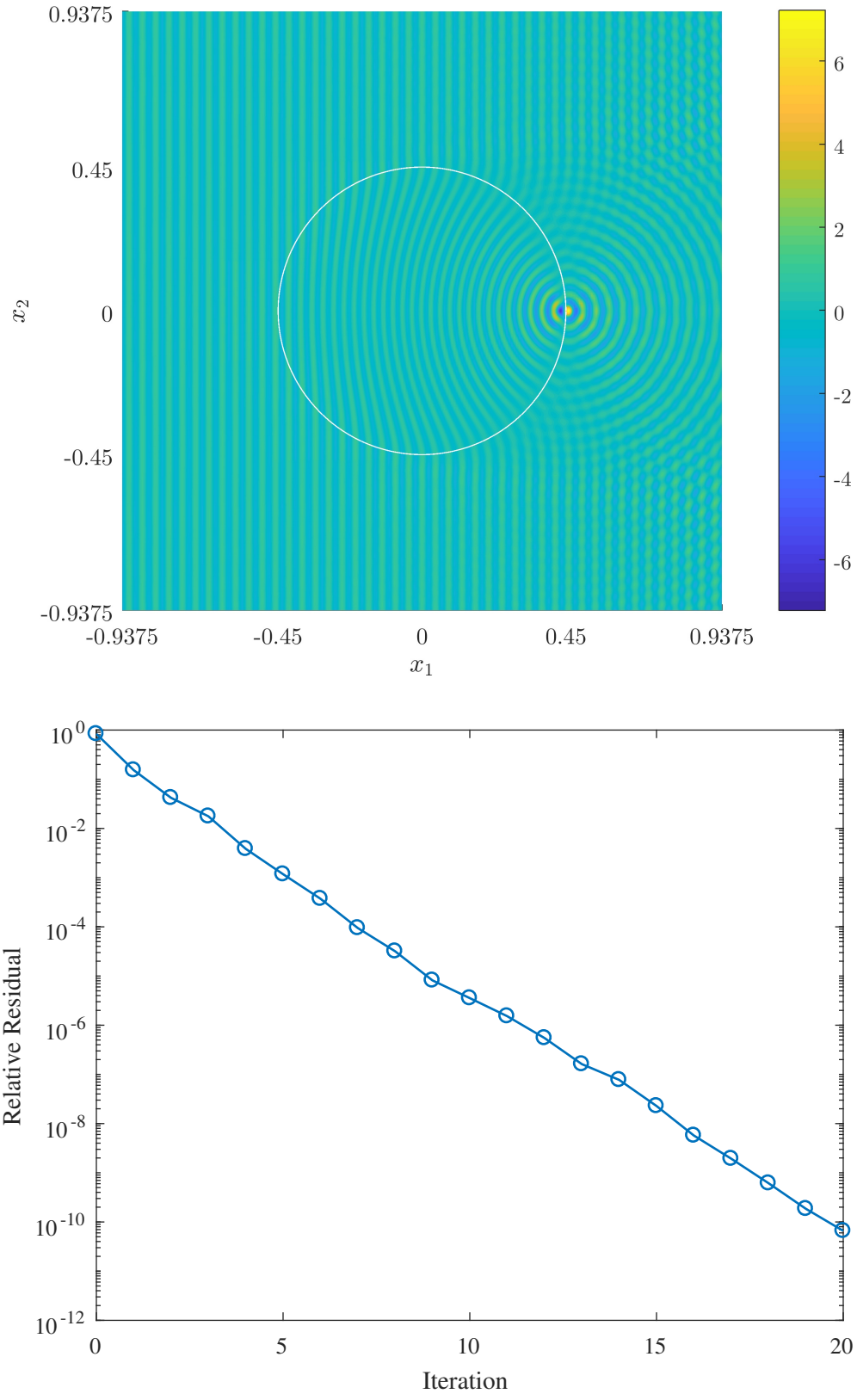

Figure 10.8: (top) Real part of the total field $\Re\{E_{x_3}\}$ for the Luneburg lens problem. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the Luneburg lens problem.

10.6.3 Eaton Lens

The second lens example is a two-dimensional Eaton lens that is meant to bend electromagnetic beam fields by 90 degrees. Again, we work with the TM_{x_3} mode. The lens has spatially varying permittivity

$$\epsilon_{33} = \begin{cases} [n(\|\bar{x}\|_2)]^2 & \|\bar{x}\|_2 \leq a \\ 1 & \text{otherwise} \end{cases} \tag{10.165}$$

where the radially symmetric index of refraction n satisfies

$$n^2 = \frac{a}{n\|\bar{x}\|_2} - \sqrt{\left(\frac{a}{n\|\bar{x}\|_2}\right)^2 - 1}. \tag{10.166}$$

We use radius $a = 0.45$ and compute n for a given $\|\bar{x}\|_2$ when needed using Newton's method with initial iterate set to 1. We use an incident Gaussian beam

$$E_i = H_0^{(2)}(\omega\|\bar{x} - \bar{x}_c\|_2)e^{-\omega/2} \tag{10.167}$$

with complex center

$$\bar{x}_c = \begin{bmatrix} -0.76 - 0.5\jmath \\ 0.275 \end{bmatrix} \tag{10.168}$$

and frequency $\omega = 48 \cdot 2\pi$. This corresponds to a wavelength $\lambda = 1/48$ outside the lens. We solve for the scattered field in domain $\Omega = (-0.75, 0.75)^2$ which is $72\lambda \times 72\lambda$ in size.

The PML have thicknesses $w_i = 0.1875$ and decay rates $\sigma_i = 37$. Since the index of refraction is singular at the origin, we truncate it at a radius of $1/30$ and leave it constant inside that radius (we choose the constant so that ϵ_{33} is continuous). We resolve both this artificial radial interface and the radial interface of the lens using a mesh that is one level finer than the meshes in Sections 10.6.1 and 10.6.2. We do this to avoid computing large Legendre expansions for β (which is not smooth at these two radii, only continuous). We compute the Legendre coefficients at a weaker error tolerance of 10^{-6}. We also terminate the domain decomposition algorithm once the relative residual has decreased by 6 orders of magnitude. We use these less stringent tolerances for this example since we have already compromised the accuracy of the solution by truncating the index of refraction. Degree 24 polynomial expansions are used for all elements. All other parameters are the same as used in Section 10.6.2.

Figure 10.9 illustrates the real part of the total field $E_{x_3} = E_i + E_s$ for the Eaton lens problem. Clearly, the Gaussian beam is bent by 90 degrees around the origin. The figure also shows the preconditioned relative residual as a function of the number of iterations. Again,

a modest number of iterations is required to compute the solution. There are 1,717,500 unknowns in $\bar{\phi}$ with 58,629 rows in the coarse problem matrix $\underline{K}$. This growth in the size of $\underline{K}$ relative to the Luneburg lens problem in Section 10.6.2 is due to two factors. First, increasing the problem size in terms of λ causes $\underline{K}$ to increase in size. Second, increasing the number of elements (the mesh for this problem is finer) introduces new edges which also contribute to increasing the size of $\underline{K}$.

10.6.4 Photonic Crystal Waveguide

In the next example, we consider a photonic crystal waveguide with geometry described in [172]. In that paper, the dielectric rods that constitute the photonic crystal are spatially varying Gaussian cylinders. We choose to replace those cylinders with conventional dielectric cylinders of constant permittivity (see, for example, [173]). Our method is capable of handling both problems, but the circular rods are more challenging. This is because the mesh must resolve the circular boundary of each cylinder in the circular rod problem whereas the mesh need not do so for the Gaussian cylinders. We use this example to show how the domain decomposition method behaves when many fine geometric features must be captured by the mesh.

The photonic crystal is comprised of a 20×20 array of dielectric cylinders, each of radius $a = 4/475$. These cylinders are located inside domain $\Omega = (-0.8, 0.8)^2$. The centers of the rods are spaced uniformly in the region $[-0.4, 0.4]^2$. A channel is removed along the 11th row and 15th column so as to form a type of waveguide bend. See the field plot in Figure 10.11 for a schematic depiction of the centers of the rods. The rods each have permittivity $\epsilon_{33} = 12.25$ corresponding to silicon and are immersed in free space with $\epsilon_{33} = 1$. Again, we work in the TM_{x_3} mode. The incident field is a plane wave with frequency $\omega = 50$ corresponding to a free space wavelength $\lambda \approx 1/8$ so that the domain is approximately $13\lambda \times 13\lambda$ in size. This frequency is chosen to lie in the first bandgap of the photonic crystal.

The PML have thicknesses $w_i = 0.2$ and decay rates $\sigma_i = 37$. The mesh has a maximum level of refinement of 9 and a minimum level of refinement of 4. All elements have degree 8 polynomial basis functions and all Legendre expansions are computed to a tolerance of 10^{-6}. The iterative method is terminated once the relative residual is reduced by 6 orders of magnitude. All other unspecified parameters are chosen as in Section 10.6.1.

Figure 10.10 illustrates the mesh used to resolve the photonic crystal waveguide problem. The mesh is very fine near dielectric cylinders and coarse away from them. The figure shows a detailed view of the mesh in the vicinity of one of the cylinders. Curvilinear elements are used to model the circular boundary. Similar mesh detail is used to model the boundaries

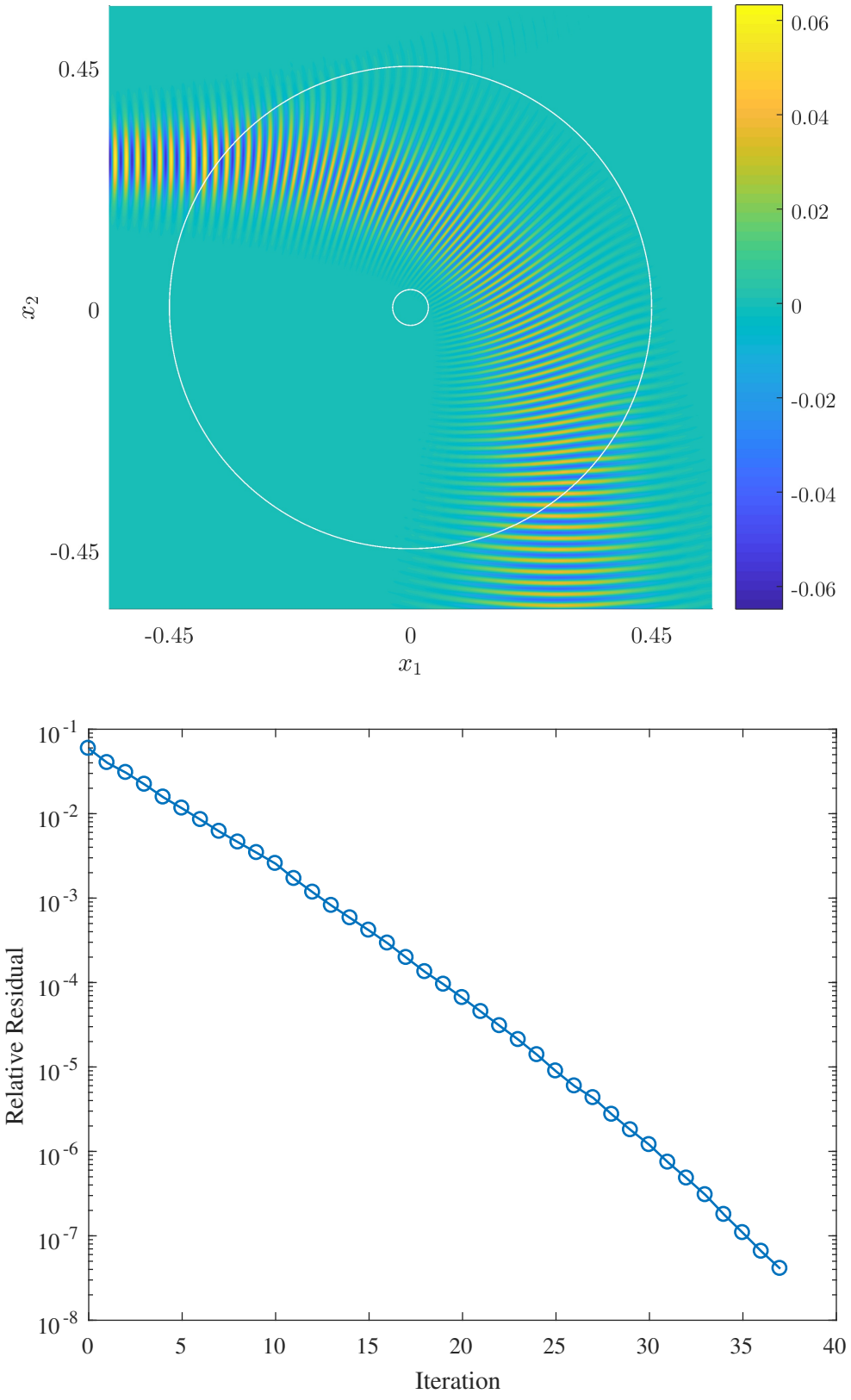

Figure 10.9: (top) Real part of the total field $\Re\{E_{x_3}\}$ for the Eaton lens problem. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the Eaton lens problem.

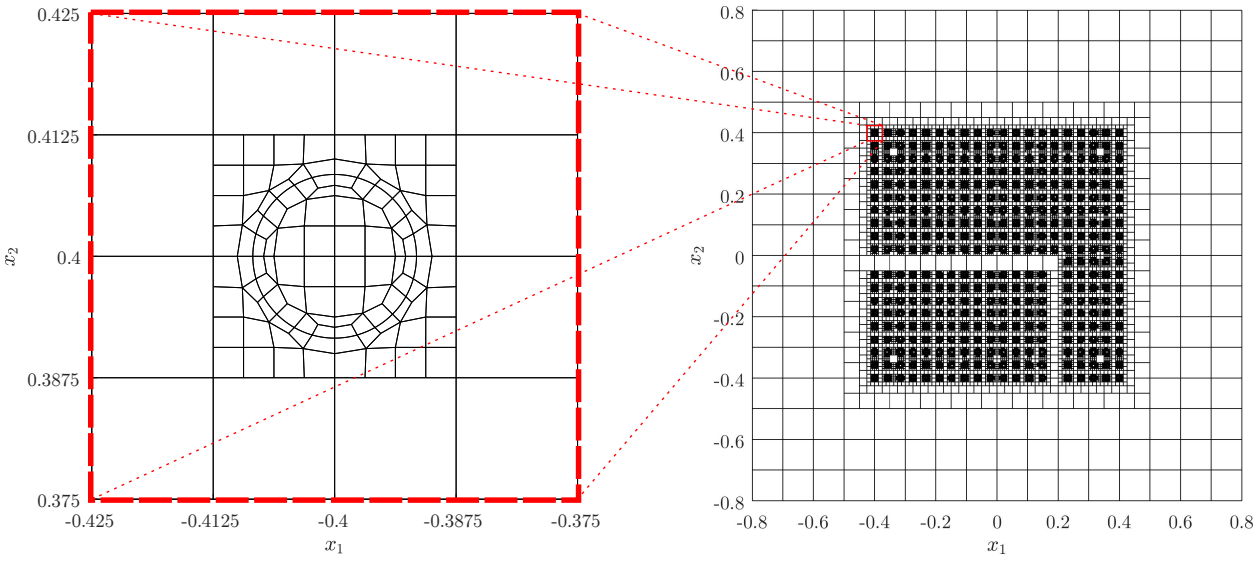

Figure 10.10: Mesh used to compute the scattered field due to a photonic crystal waveguide. A detailed view of the mesh near one dielectric rod is shown on the left with the full mesh on the right.

of the remaining 375 cylinders.

Figure 10.11 shows the total field $E_{x_3} = E_i + E_s$ for the photonic crystal waveguide problem. By virtue of choosing an incident field with frequency in the first bandgap of the photonic crystal, the total field is attenuated in the crystal, but propagates through the waveguide channel. The figure also shows the reduction in the preconditioned relative residual for the domain decomposition method used to produce the solution. There are 3,662,658 unknowns in $\bar{\phi}$ with 324,387 rows in the coarse problem matrix $\underline{K}$. A weakness of the domain decomposition method is evident in this example. In particular, when the mesh must capture many fine features, the size of the coarse problem can grow rapidly. This is because each edge in the mesh contributes roughly l rows to the size of the coarse problem and the number of edges is much larger when fine geometric features are present in the mesh.

10.6.5 Perfect Electric Conducting Ogival Cylinder

In the next example, we consider the TE_{x_3} mode rather than the TM_{x_3} mode for an example of scattering from an infinitely long, perfect electric conducting ogival cylinder. We use this example to show that the domain decomposition method can be applied to geometries including corners and to provide comparisons to the existing literature [12]. We have previously considered an example in the TE_{x_3} mode in Section 9.6.2 and refer the reader to the development there regarding the appropriate PDE problem to solve and the accompanying boundary conditions. We write the total field $H_{x_3} = H_i + H_s$ in terms of a known incident

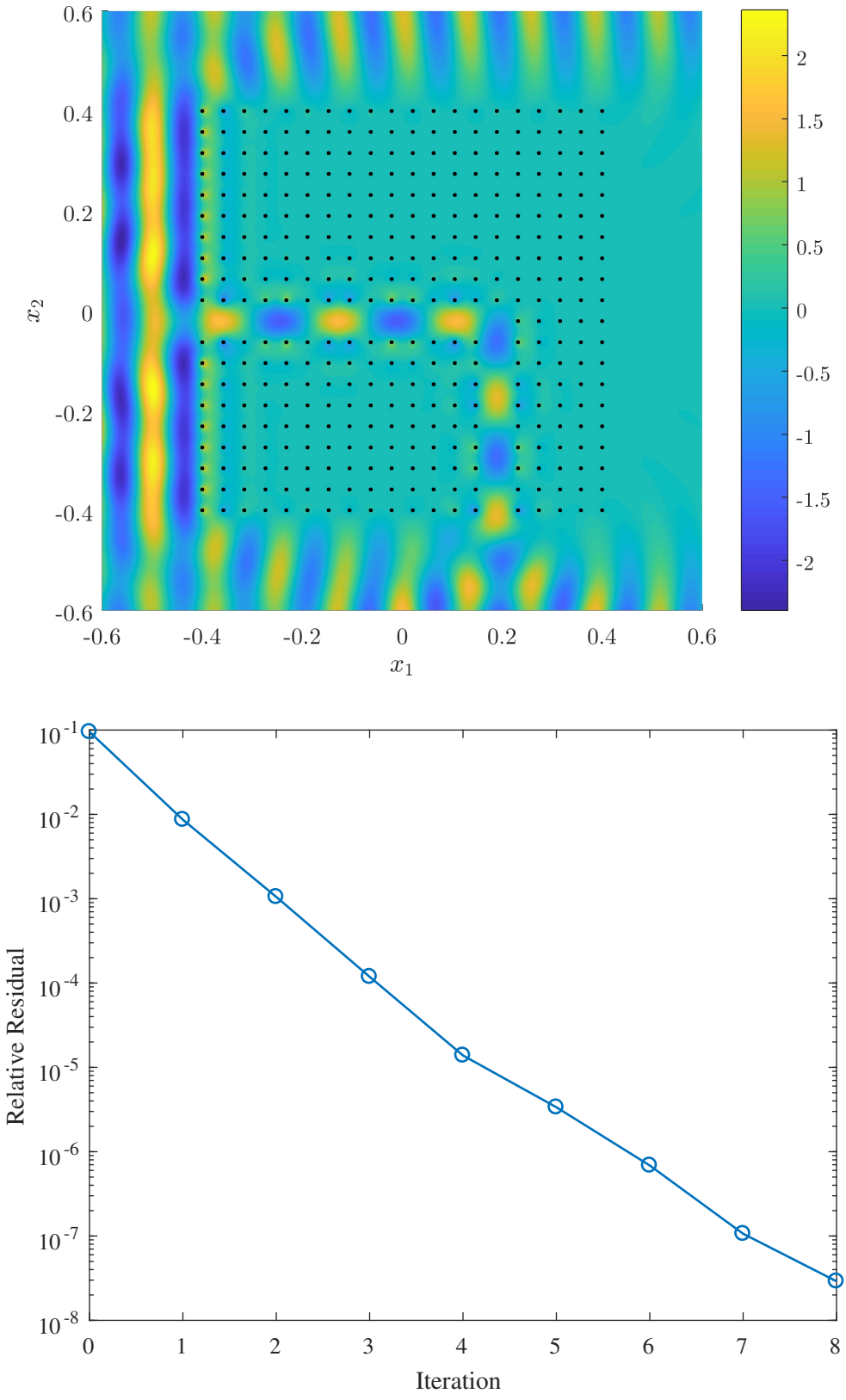

Figure 10.11: (top) Real part of the total field $\Re\{E_{x_3}\}$ for the photonic crystal waveguide problem. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the photonic crystal waveguide problem.

field H_i and unknown scattered field H_s, then solve the PDE problem

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi = f \tag{10.169}$$

with corresponding parameters

$$\underline{\alpha} = \begin{bmatrix} \dfrac{1}{\epsilon_{22}} & 0 \\ 0 & \dfrac{1}{\epsilon_{11}} \end{bmatrix}, \qquad \beta = -\omega^2 \mu_{33}, \qquad f = \nabla \cdot (\underline{\alpha} \nabla H_i) - \beta H_i, \qquad \phi = H_s, \tag{10.170}$$

for H_s subject to boundary condition

$$\bar{n}^T (\underline{\alpha} \nabla H_s) = -\bar{n}^T (\underline{\alpha} \nabla H_i) \tag{10.171}$$

on the perfect electric conductor boundary.

The two boundary components of the ogive are circular arcs. Defining parameters

$$h_o = \frac{2.07}{2}, \qquad w_o = \frac{5}{2}, \qquad \rho_o = \frac{h_o^2 + w_o^2}{2h_o}, \tag{10.172}$$

both arcs have radius ρ_o. The upper arc is comprised of points satisfying $x_2 \geq 0$ on the circle with center

$$\bar{x}_{c,\text{upper}} = \begin{bmatrix} 0 \\ h_o - \rho_o \end{bmatrix} \tag{10.173}$$

whereas the lower arc is comprised of points satisfying $x_2 \leq 0$ on the circle with center

$$\bar{x}_{c,\text{lower}} = \begin{bmatrix} 0 \\ -(h_o - \rho_o) \end{bmatrix}. \tag{10.174}$$

The ogive measures $2w_o$ along the x_1 axis and $2h_o$ along the x_2 axis. We let Ω_o be the interior of the ogive and solve for the scattered field in domain $\Omega = (-3.5, 3.5)^2 \setminus \Omega_o$. We assume free space conditions $\underline{\epsilon} = \underline{I}$ and $\underline{\mu} = \underline{I}$ throughout Ω. We choose the incident field to be the plane wave given by

$$H_i = e^{\jmath \omega \bar{k}^T \bar{x}} \tag{10.175}$$

with unit vector

$$\bar{k} = \begin{bmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{bmatrix} \tag{10.176}$$

and incident angle φ_i. Since H_i is a plane wave and the problem is in free space, the forcing function f is zero. We choose frequency $\omega = 2\pi$ for the incident wave, but will vary the unit

vector $\bar{k}$ using the incidence angle φ_i. The Neumann boundary condition (10.171) can be enforced with parameters

$$\gamma = 0, \qquad q = -\jmath\omega\bar{n}^T\underline{\alpha}\bar{k}H_i, \tag{10.177}$$

with inward pointing normal to the ogive (which is outward pointing to Ω) given by

$$\bar{n}(\bar{x}) = \frac{-1}{\sqrt{x_1^2 + (x_2 \pm (h_o - \rho_o))^2}} \begin{bmatrix} x_1 \\ x_2 \pm (h_o - \rho_o) \end{bmatrix} \tag{10.178}$$

where the sign changes depending on if the point $\bar{x}$ belongs to the upper or lower arc of the ogive.

A PML region with thicknesses $w_i = 0.4375$ and decay rates $\sigma_i = 16$ is chosen to attenuate the scattered field. The mesh uses a maximum level of refinement of 7 and a minimum level of refinement of 4 with fixed points

$$\{\bar{x}_{\text{fixed},k}\} = \left\{ \begin{bmatrix} w_o \\ 0 \end{bmatrix}, \begin{bmatrix} -w_o \\ 0 \end{bmatrix} \right\} \tag{10.179}$$

corresponding to the corners of the ogive. The mesh is illustrated in Figure 10.12. In all results, the solution is computed with degree 16 polynomials on all elements, and coefficient expansions and boundary representations are computed to a tolerance of 10^{-6}. The domain decomposition algorithm is terminated once the preconditioned relative residual is reduced by 6 orders of magnitude.

First, we show results for a single angle of incidence $\varphi_i = 0$. Figure 10.12 shows the preconditioned relative residual as a function of the number of iterations for the domain decomposition method while Figure 10.13 illustrates the real part of the scattered field. There are 269,926 unknowns in $\bar{\phi}$ with 6,158 rows in the coarse problem matrix $\underline{K}$.

Next, we compute the RCS of the ogive. We do so for 360 equally spaced incident angles φ_i and 360 observation angles in $[0, 2\pi)$. The RCS is computed in the same way as in Section 9.6.2. We choose the boundary of the ogive as Γ when computing the far field integral. Since the ogive is not invariant under rotations about the x_3 axis, we compute the bistatic RCS illustrated in Figure 10.13. We do so by computing H_s for a given incident angle φ_i using the domain decomposition method, then compute H_{far} in (9.213) and σ_{2D} in (9.218) at the 360 observation angles φ. We repeat this process for all 360 incident angles.

Finally, for comparison purposes, we reproduce Figure 4.23(b) and 4.25(b) in [12]. The first of these figures demonstrates the ratio of the magnitude of the total field $|H_{x_3}|$ and the incident field $|H_i|$ observed at the surface of the ogive for the incident angle $\varphi_i = 0$. Figure 10.14 reproduces Figure 4.23(b) in [12] where s is the distance along the upper arc

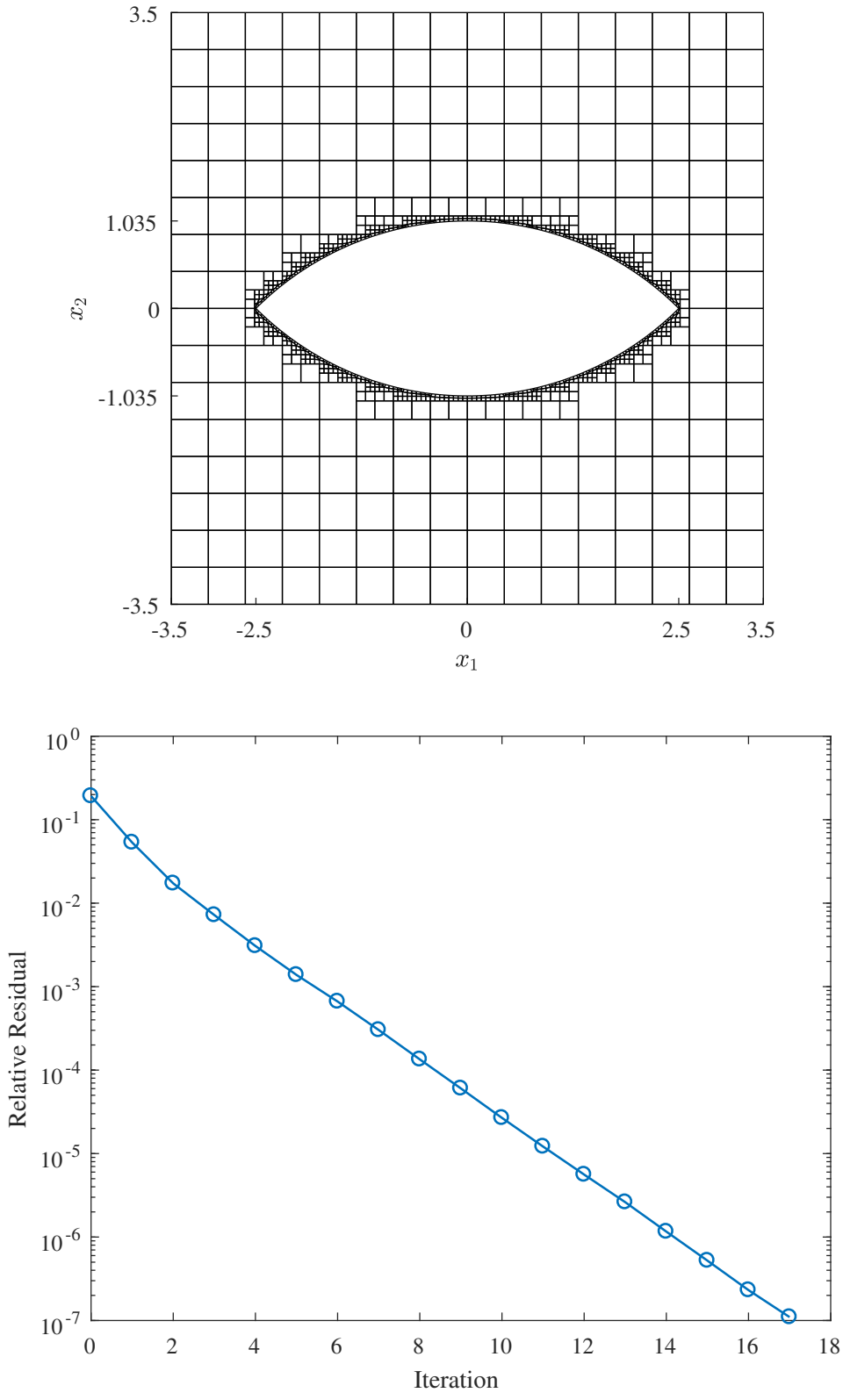

Figure 10.12: (top) Non-conforming mesh used to solve the perfect electric conducting ogival cylinder problem. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the perfect electric conducting ogival cylinder problem.

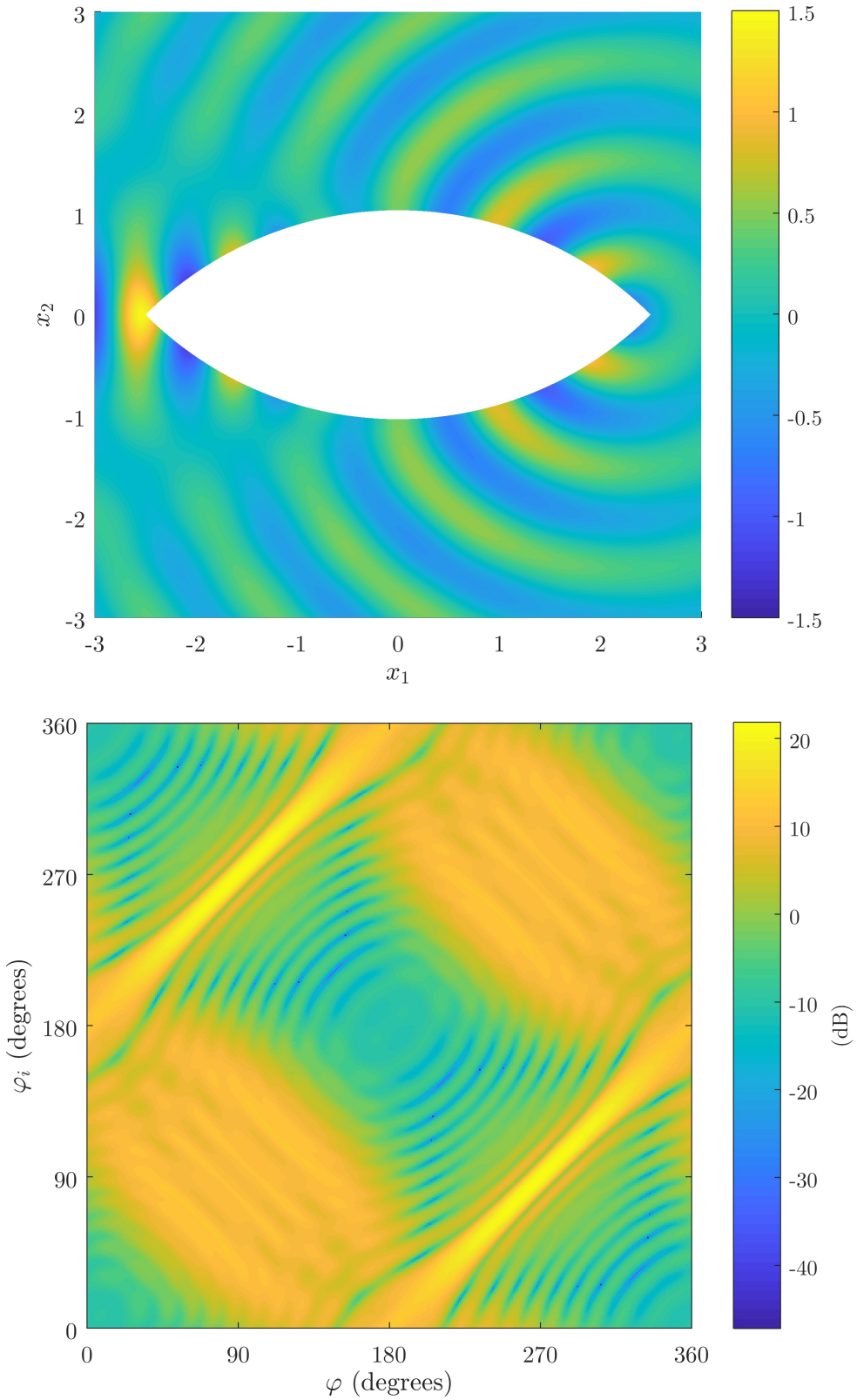

Figure 10.13: (top) Real part of the scattered field $\Re\{H_s\}$ for the perfect electric conducting ogival cylinder problem. (bottom) Bistatic RCS for the perfect electric conducting ogival cylinder problem.

of the ogive (or lower arc due to symmetry of the solution) measured from its leading edge (the rightmost corner of the ogive). We also include the monostatic RCS (where incident and observation angles are equal) for angles between 0 and 90 degrees in Figure 10.14. This corresponds to sampling the bistatic RCS in Figure 10.13 along the line $\varphi_i = \varphi$. Only angles 0 to 90 are required to characterize the monostatic RCS because of symmetries of the ogive. No symmetries were exploited when computing the bistatic RCS, but Figure 10.13 shows that such symmetries exist. The monostatic RCS in Figure 10.14 can be directly compared with Figure 4.25(b) in [12]; the result is visually indistinguishable.

10.6.6 Electromagnetic Cloak

As a final example, we consider the electromagnetic cloak in [174]. We use this example to show how the method behaves with anisotropic, spatially varying permittivity and permeability and again demonstrate the accuracy of the method. The cloak is meant to eliminate scattering from objects inside an infinite cylinder coaxial with the x_3 axis that has radius R_1. It is comprised of a layer of anisotropic and spatially varying permittivity and permeability immediately adjacent to the outer surface of the cylinder. In practice, anything can be placed inside the cylinder, but we will assume that the cylinder is a perfect electric conductor. The layer has outer radius $R_2 > R_1$ and its permittivity tensor in cylindrical coordinates has components

$$\epsilon_\rho = \frac{\rho - R_1}{\rho}, \tag{10.180}$$

$$\epsilon_\varphi = \frac{1}{\epsilon_\rho}, \tag{10.181}$$

$$\epsilon_{33} = \left(\frac{R_2}{R_2 - R_1}\right)^2 \epsilon_\rho, \tag{10.182}$$

where $\rho = \|\bar{x}\|_2$ and $\varphi = \text{atan2}(x_2, x_1)$ are the standard cylindrical coordinates. To obtain the permittivity tensor in Cartesian coordinates, we use the transformation

$$\underbrace{\begin{bmatrix} E_{x_1} \\ E_{x_2} \\ E_{x_3} \end{bmatrix}}_{\bar{E}_{\bar{x}}} = \underbrace{\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\underline{R}^T} \underbrace{\begin{bmatrix} E_\rho \\ E_\varphi \\ E_{x_3} \end{bmatrix}}_{\bar{E}_{\bar{\rho}}} \tag{10.183}$$

from vectors $\bar{E}_{\bar{\rho}}$ expressed with unit vectors in cylindrical coordinates to vectors $\bar{E}_{\bar{x}}$ expressed with unit vectors in Cartesian coordinates. Note that the transformation matrix $\underline{R}^T$ is a

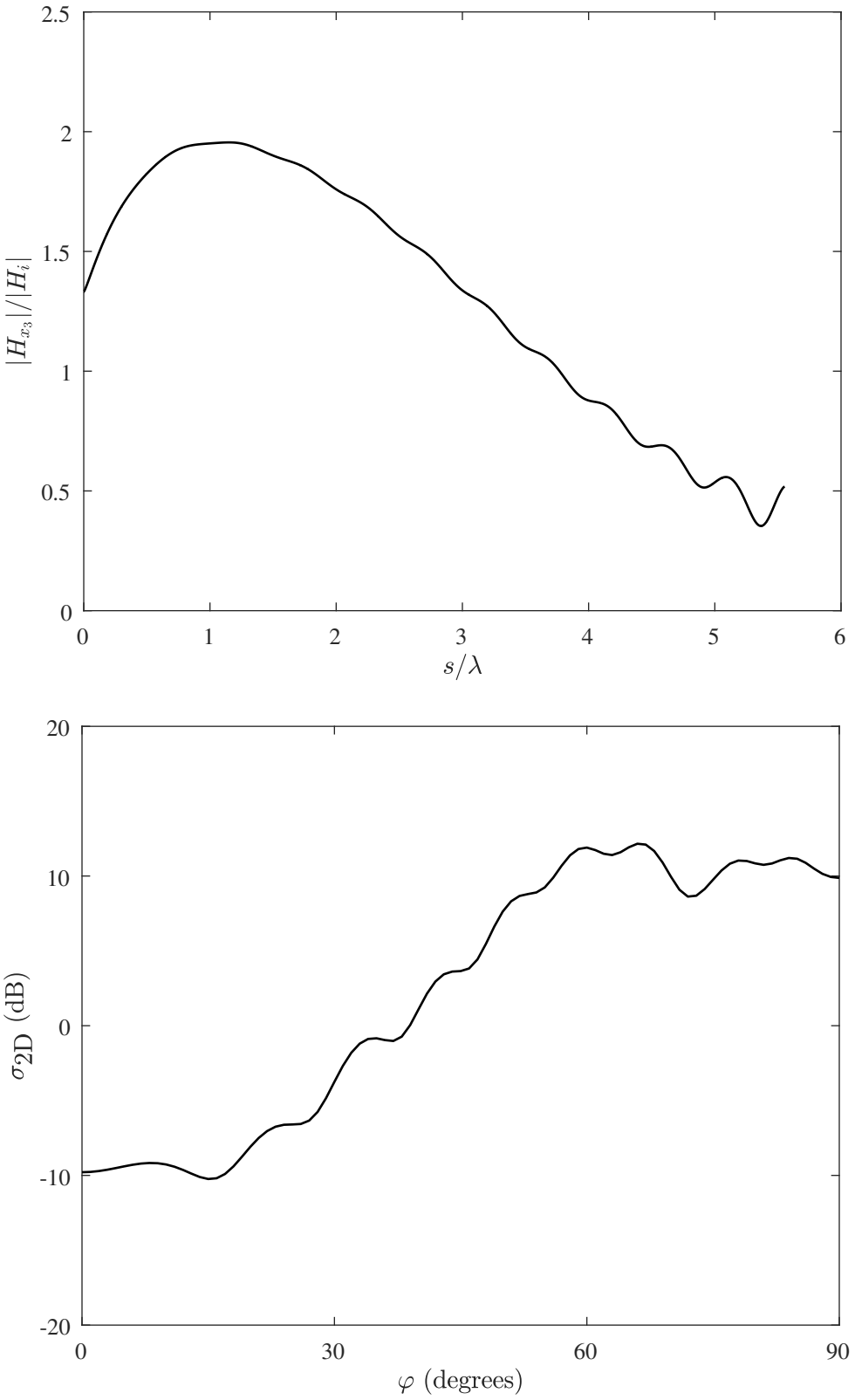

Figure 10.14: (top) Ratio of the absolute values of the total field and the incident field observed at the surface of the perfect electric conducting ogival cylinder. (bottom) Monostatic RCS for the perfect electric conducting ogival cylinder.

rotation matrix that is orthogonal. This means that $\underline{R}^T\underline{R} = \underline{R}\underline{R}^T = \underline{I}$. Since the constitutive relation using cylindrical unit vectors is

$$\bar{D}_{\bar{\rho}} = \underline{\epsilon}_{\bar{\rho}}\bar{E}_{\bar{\rho}} \tag{10.184}$$

where $\underline{\epsilon}_{\bar{\rho}}$ is the permittivity tensor in cylindrical coordinates, we find that multiplying from the right by $\underline{R}^T$ and using (10.183) gives

$$\underbrace{\underline{R}^T\bar{D}_{\bar{\rho}}}_{\bar{D}_{\bar{x}}} = \underline{R}^T\underline{\epsilon}_{\bar{\rho}}\bar{E}_{\bar{\rho}}. \tag{10.185}$$

In addition, multiplying (10.183) by $\underline{R}$ gives $\bar{E}_{\bar{\rho}} = \underline{R}\bar{E}_{\bar{x}}$ so that substituting this last expression into (10.185) yields

$$\bar{D}_{\bar{x}} = \underbrace{\underline{R}^T\underline{\epsilon}_{\bar{\rho}}\underline{R}}_{\underline{\epsilon}_{\bar{x}}}\bar{E}_{\bar{x}}. \tag{10.186}$$

Therefore, to obtain the permittivity tensor $\underline{\epsilon}_{\bar{x}}$ in Cartesian coordinates, we compute

$$\underline{\epsilon}_{\bar{x}} = \underline{R}^T\underline{\epsilon}_{\bar{\rho}}\underline{R} \tag{10.187}$$

$$= \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \epsilon_\rho & 0 & 0 \\ 0 & \epsilon_\varphi & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix}\begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.188}$$

$$= \begin{bmatrix} \epsilon_\rho\cos(\varphi) & -\epsilon_\varphi\sin(\varphi) & 0 \\ \epsilon_\rho\sin(\varphi) & \epsilon_\varphi\cos(\varphi) & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix}\begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.189}$$

$$= \begin{bmatrix} \epsilon_\rho\cos^2(\varphi) + \epsilon_\varphi\sin^2(\varphi) & (\epsilon_\rho - \epsilon_\varphi)\sin(\varphi)\cos(\varphi) & 0 \\ (\epsilon_\rho - \epsilon_\varphi)\sin(\varphi)\cos(\varphi) & \epsilon_\rho\sin^2(\varphi) + \epsilon_\varphi\cos^2(\varphi) & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix} \tag{10.190}$$

so that

$$\underline{\epsilon} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & 0 \\ \epsilon_{12} & \epsilon_{22} & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix} \tag{10.191}$$

with

$$\epsilon_{11} = \epsilon_\rho \cos^2(\varphi) + \epsilon_\varphi \sin^2(\varphi), \tag{10.192}$$

$$\epsilon_{12} = (\epsilon_\rho - \epsilon_\varphi) \sin(\varphi) \cos(\varphi), \tag{10.193}$$

$$\epsilon_{22} = \epsilon_\rho \sin^2(\varphi) + \epsilon_\varphi \cos^2(\varphi). \tag{10.194}$$

The cloak also has permeability $\underline{\mu} = \underline{\epsilon}$. These expressions for the permittivity and permeability are valid inside the layer where $R_1 < \rho \leq R_2$. Note that ϵ_φ is singular as ρ approaches R_1. In our computations, we replace ϵ_φ in (10.181) with

$$\epsilon_\varphi = \frac{\rho}{\rho - R_1 + \delta} \tag{10.195}$$

where $\delta = 10^{-8}$ is a small perturbation that smooths ϵ_φ at the singularity. By including the regularization parameter δ, we compromise the electromagnetic cloak's ability to mask the perfect electric conductor but, as we will see, the effect is small. Outside of the cloak, where $\rho > R_2$, we have free space with $\underline{\epsilon} = \underline{\mu} = \underline{I}$.

Even though the permittivity and permeability are no longer diagonal, their structure (10.191) continues to permit separate analysis of the TM_{x_3} and TE_{x_3} modes. We will focus on the TE_{x_3} mode; analysis for the TM_{x_3} mode is analogous. Following the same procedure as outlined in Section 9.6.2, but using the new structure of the permittivity and permeability tensors, we obtain

$$-\nabla \cdot (\underline{\alpha} \nabla \phi) + \beta \phi = f \tag{10.196}$$

with corresponding parameters

$$\underline{\alpha} = \frac{1}{\epsilon_{11}\epsilon_{22} - \epsilon_{12}^2} \begin{bmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{12} & \epsilon_{22} \end{bmatrix}, \qquad \beta = -\omega^2 \mu_{33}, \qquad f = \nabla \cdot (\underline{\alpha} \nabla H_i) - \beta H_i, \qquad \phi = H_s. \tag{10.197}$$

The parameter $\underline{\alpha}$ simplifies considerably because

$$\epsilon_{11}\epsilon_{22} - \epsilon_{12}^2 = [\epsilon_\rho \cos^2(\varphi) + \epsilon_\varphi \sin^2(\varphi)][\epsilon_\rho \sin^2(\varphi) + \epsilon_\varphi \cos^2(\varphi)] - [(\epsilon_\rho - \epsilon_\varphi) \sin(\varphi) \cos(\varphi)]^2. \tag{10.198}$$

Multiplying, squaring, and collecting like terms gives

$$\epsilon_{11}\epsilon_{22} - \epsilon_{12}^2 = \epsilon_\rho \epsilon_\varphi [\cos^4(\varphi) + 2\sin^2(\varphi)\cos^2(\varphi) + \sin^4(\varphi)] \tag{10.199}$$

$$= \epsilon_\rho \epsilon_\varphi \underbrace{[\cos^2(\varphi) + \sin^2(\varphi)]^2}_{1}. \tag{10.200}$$

In fact, using (10.180) and (10.181) shows that

$$\epsilon_\rho \epsilon_\varphi = 1 \tag{10.201}$$

and that

$$\epsilon_{11}\epsilon_{22} - \epsilon_{12}^2 = 1. \tag{10.202}$$

When regularizing ϵ_φ, this equality is only approximate[10]. For this reason, we use

$$\underline{\alpha} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{12} & \epsilon_{22} \end{bmatrix} \tag{10.203}$$

in our calculations.

The formulation in terms of the standard PDE (10.196) is incomplete. This is because the weak form of (10.196) weakly enforces continuity of the normal flux

$$\bar{n}^T \underline{\alpha} \nabla \phi = \bar{n}^T \underline{\alpha} \nabla H_s \tag{10.204}$$

of the scattered field H_s whereas continuity must be enforced for the flux of the total field $H_{x_3} = H_i + H_s$. This has not been a problem for other scattering examples in this thesis because those examples have continuous $\underline{\alpha}$. When $\underline{\alpha}$ is continuous, continuity of the flux (10.204) implies that the gradient ∇H_s itself is continuous across element interfaces, and since the gradient of the incident field is continuous, so too is the gradient of the total field and consequently the total flux. In this example, parameter $\underline{\alpha}$ is discontinuous where the cloak layer and free space meet so we cannot conclude that the gradient of the scattered field is continuous across that interface. Therefore, we must make sure that the weak form weakly forces the continuity condition

$$\bar{n} \times (\bar{E}_1 - \bar{E}_2) = 0 \tag{10.205}$$

where $\bar{E}_1$ denotes the electric field intensity outside the cloak layer, $\bar{E}_2$ denotes the field inside the cloak layer, and $\bar{n}$ is the unit normal vector pointing into the cloak. We saw in Section 9.6.2 that $\bar{n} \times \bar{E}$ can be written in terms of the normal flux of H_{x_3} so that the tangential continuity condition (10.205) becomes

$$\bar{n}^T [\underline{\alpha}_2 \nabla (H_{x_3})_2 - \underline{\alpha}_1 \nabla (H_{x_3})_1] = 0. \tag{10.206}$$

[10]To recover the equality, ϵ_ρ needs to also contain the regularization parameter. In practice, both approaches produce effective electromagnetic cloaks.

To see how to enforce this condition, we start from the functional for the total field and reduce it to one for the scattered field.

The functional for the Helmholtz equation for the total field is given by

$$F(H_{x_3}) = \frac{1}{2}\int_\Omega (\nabla H_{x_3})^T \underline{\alpha} \nabla H_{x_3} + \beta H_{x_3}^2 d\Omega. \tag{10.207}$$

For clarity of exposition, we ignore boundary conditions in our derivation, although they can be added without difficulty. In Chapter 6, we saw that finding the stationary point of such a functional implied that

$$-\nabla \cdot (\underline{\alpha} \nabla H_{x_3}) + \beta H_{x_3} = 0 \qquad \text{in } \Omega, \tag{10.208}$$

$$\bar{n}^T [\underline{\alpha}_2 \nabla (H_{x_3})_2 - \underline{\alpha}_1 \nabla (H_{x_3})_1] = 0 \qquad \text{on } \Gamma, \tag{10.209}$$

where Γ was an interface internal to domain Ω where $\underline{\alpha}$ was discontinuous. This functional imposes the correct continuity condition on the total field. Substituting $H_{x_3} = H_i + H_s$ into the functional gives

$$F(H_s) = \frac{1}{2}\int_\Omega (\nabla (H_i + H_s))^T \underline{\alpha} \nabla (H_i + H_s) + \beta (H_i + H_s)^2 d\Omega \tag{10.210}$$

$$= \frac{1}{2}\int_\Omega (\nabla H_s)^T \underline{\alpha} \nabla H_s + \beta {H_s}^2 d\Omega + \int_\Omega (\nabla H_s)^T \underline{\alpha} \nabla H_i + \beta H_s H_i d\Omega \tag{10.211}$$

where we have ignored terms that are independent of H_s (they will vanish when taking the first variation with respect to the unknown scattered field H_s). Applying integration by parts to remove the gradient from H_s in the second integral yields

$$F(H_s) = \frac{1}{2}\int_\Omega (\nabla H_s)^T \underline{\alpha} \nabla H_s + \beta {H_s}^2 d\Omega + \int_\Omega H_s [-\nabla \cdot (\underline{\alpha} \nabla H_i) + \beta H_i] d\Omega + \int_\Gamma H_s \bar{n}^T (\underline{\alpha} \nabla H_i) d\Omega. \tag{10.212}$$

Notice how the first integral contains the contributions for the operator matrix and the second integral contains the contribution of the forcing function in (10.197) for (10.196). The remaining boundary integral is what is missing from our formulation. When there is a discontinuity in $\underline{\alpha}$, two such terms arise (one from each side of the interface) so that

$$\int_\Gamma H_s \bar{n}^T (\underline{\alpha}_2 - \underline{\alpha}_1) \nabla H_i d\Omega \tag{10.213}$$

must be added to the functional. For the cloak problem, we add a Robin boundary condition with parameters

$$\gamma = 0, \qquad q = \bar{n}^T (\underline{\alpha}_2 - \underline{\alpha}_1) \nabla H_i, \tag{10.214}$$

on the interface between the cloak and free space regions in addition to the standard explicit continuity condition on H_s. This amounts to adding a forcing term to the right hand side of the saddle point system (in other words, we modify the forcing vector $\bar{b}$ but can leave the operator matrix $\underline{A}$, constraint matrix $\underline{C}$, and constraint vector $\bar{d}$ unchanged).

As usual, there is an equivalent formulation using the weighted residual method. We start from the standard weighted residual for the scattered field equation, but add a boundary term involving the normal flux of the incident field at the boundary where $\underline{\alpha}$ is discontinuous to both sides of the equation. The term added to the side of the equation with the operator terms is lumped together with the preexisting boundary term and only changes the meaning of the Lagrange multipliers. In the original scattered field equations, the Lagrange multipliers represent the flux of the scattered field, but with this added term, the Lagrange multipliers now represent the flux of the total field. The term added to the side of the equations associated with forcing terms is discretized and gives rise to the same added forcing term as in the variational formulation.

In our example, we choose a plane wave incident field

$$H_i = e^{-\jmath\omega\bar{k}^T\bar{x}} \tag{10.215}$$

with unit vector $\bar{k} = \bar{e}_1$ and frequency $\omega = 20 \cdot 2\pi$ corresponding to a free space wavelength $\lambda = 1/20$. This choice leads to the imposition of Robin boundary conditions with parameters

$$\gamma = 0, \qquad q = -\jmath\omega\bar{n}^T(\underline{\alpha}_2 - \underline{\alpha}_1)\bar{k}H_i, \tag{10.216}$$

on the interface between the cloak and free space regions (with $\bar{n}$ pointing into the cloak, $\underline{\alpha}_1$ corresponding to free space, and $\underline{\alpha}_2$ corresponding to the cloak). Another Robin boundary condition with parameters

$$\gamma = 0, \qquad q = \jmath\omega\bar{n}^T\underline{\alpha}\bar{k}H_i, \tag{10.217}$$

must be applied to the boundary of the perfect electric conducting cylinder (as in Section 9.6.2). The forcing function outside the cloak is zero, but inside has the form

$$f = \nabla \cdot (\underline{\alpha}\nabla H_i) - \beta H_i \tag{10.218}$$

$$= \nabla \cdot (\underline{\alpha}[-\jmath\omega\bar{k}H_i]) - \beta H_i. \tag{10.219}$$

Since

$$\underline{\alpha}\bar{k} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{12} & \epsilon_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \end{bmatrix}, \tag{10.220}$$

and $\beta = -\omega^2\mu_{33}$, the forcing function becomes

$$f = -\jmath\omega\left\{\frac{\partial}{\partial x_1}(\epsilon_{11}H_i) + \frac{\partial}{\partial x_2}(\epsilon_{12}H_i)\right\} + \omega^2\mu_{33}H_i \tag{10.221}$$

$$= -\jmath\omega\left\{\frac{\partial\epsilon_{11}}{\partial x_1}H_i + \epsilon_{11}\frac{\partial H_i}{\partial x_1} + \frac{\partial\epsilon_{12}}{\partial x_2}H_i + \epsilon_{12}\frac{\partial H_i}{\partial x_2}\right\} + \omega^2\mu_{33}H_i \tag{10.222}$$

where we have taken the divergence and applied the product rule. Taking the partial derivatives of H_i, this gives

$$f = -\jmath\omega\left\{\frac{\partial\epsilon_{11}}{\partial x_1}H_i + \epsilon_{11}(-\jmath\omega H_i) + \frac{\partial\epsilon_{12}}{\partial x_2}H_i\right\} + \omega^2\mu_{33}H_i \tag{10.223}$$

$$= \left[\omega^2(\mu_{33} - \epsilon_{11}) - \jmath\omega\left\{\frac{\partial\epsilon_{11}}{\partial x_1} + \frac{\partial\epsilon_{12}}{\partial x_2}\right\}\right]H_i. \tag{10.224}$$

The partial derivatives of ϵ_{11} and ϵ_{12} must also be evaluated. Using the product rule applied to (10.192) gives

$$\frac{\partial\epsilon_{11}}{\partial x_1} = \frac{\partial\epsilon_\rho}{\partial x_1}\cos^2(\varphi) + \epsilon_\rho\frac{\partial}{\partial x_1}[\cos^2(\varphi)] + \frac{\partial\epsilon_\varphi}{\partial x_1}\sin^2(\varphi) + \epsilon_\varphi\frac{\partial}{\partial x_1}[\sin^2(\varphi)]. \tag{10.225}$$

Applying the chain rule to each derivative term, we obtain

$$\frac{\partial\epsilon_{11}}{\partial x_1} = \frac{\partial\epsilon_\rho}{\partial\rho}\frac{\partial\rho}{\partial x_1}\cos^2(\varphi) + \epsilon_\rho\frac{\partial}{\partial\varphi}[\cos^2(\varphi)]\frac{\partial\varphi}{\partial x_1} + \frac{\partial\epsilon_\varphi}{\partial\rho}\frac{\partial\rho}{\partial x_1}\sin^2(\varphi) + \epsilon_\varphi\frac{\partial}{\partial\varphi}[\sin^2(\varphi)]\frac{\partial\varphi}{\partial x_1} \tag{10.226}$$

$$= \frac{\partial\epsilon_\rho}{\partial\rho}\frac{\partial\rho}{\partial x_1}\cos^2(\varphi) + \epsilon_\rho[-2\cos(\varphi)\sin(\varphi)]\frac{\partial\varphi}{\partial x_1} + \frac{\partial\epsilon_\varphi}{\partial\rho}\frac{\partial\rho}{\partial x_1}\sin^2(\varphi) + \epsilon_\varphi[2\sin(\varphi)\cos(\varphi)]\frac{\partial\varphi}{\partial x_1} \tag{10.227}$$

$$= \frac{\partial\rho}{\partial x_1}\left[\frac{\partial\epsilon_\rho}{\partial\rho}\cos^2(\varphi) + \frac{\partial\epsilon_\varphi}{\partial\rho}\sin^2(\varphi)\right] + 2\frac{\partial\varphi}{\partial x_1}(\epsilon_\varphi - \epsilon_\rho)\cos(\varphi)\sin(\varphi). \tag{10.228}$$

Similarly, using the product rule and chain rule on (10.193) gives

$$\frac{\partial\epsilon_{12}}{\partial x_2} = \left(\frac{\partial\epsilon_\rho}{\partial x_2} - \frac{\partial\epsilon_\varphi}{\partial x_2}\right)\sin(\varphi)\cos(\varphi) + (\epsilon_\rho - \epsilon_\varphi)\frac{\partial}{\partial x_2}[\sin(\varphi)\cos(\varphi)] \tag{10.229}$$

$$= \left(\frac{\partial\epsilon_\rho}{\partial\rho}\frac{\partial\rho}{\partial x_2} - \frac{\partial\epsilon_\varphi}{\partial\rho}\frac{\partial\rho}{\partial x_2}\right)\sin(\varphi)\cos(\varphi) + (\epsilon_\rho - \epsilon_\varphi)\frac{\partial}{\partial\varphi}[\sin(\varphi)\cos(\varphi)]\frac{\partial\varphi}{\partial x_2} \tag{10.230}$$

$$= \left(\frac{\partial\epsilon_\rho}{\partial\rho} - \frac{\partial\epsilon_\varphi}{\partial\rho}\right)\frac{\partial\rho}{\partial x_2}\sin(\varphi)\cos(\varphi) + (\epsilon_\rho - \epsilon_\varphi)[\cos^2(\varphi) - \sin^2(\varphi)]\frac{\partial\varphi}{\partial x_2}. \tag{10.231}$$

To complete the description of these derivatives, we evaluate

$$\frac{\partial \rho}{\partial x_1} = \frac{\partial}{\partial x_1}[\|\bar{x}\|_2] \tag{10.232}$$

$$= \frac{x_1}{\|\bar{x}\|_2}, \tag{10.233}$$

as well as

$$\frac{\partial \rho}{\partial x_2} = \frac{\partial}{\partial x_2}[\|\bar{x}\|_2] \tag{10.234}$$

$$= \frac{x_2}{\|\bar{x}\|_2}, \tag{10.235}$$

and

$$\frac{\partial \varphi}{\partial x_1} = \frac{\partial}{\partial x_1}\text{atan}\left(\frac{x_2}{x_1}\right) \tag{10.236}$$

$$= -\frac{x_2}{\|\bar{x}\|_2^2}, \tag{10.237}$$

as well as

$$\frac{\partial \varphi}{\partial x_2} = \frac{\partial}{\partial x_2}\text{atan}\left(\frac{x_2}{x_1}\right) \tag{10.238}$$

$$= \frac{x_1}{\|\bar{x}\|_2^2}. \tag{10.239}$$

In addition, using (10.180), we have

$$\frac{\partial \epsilon_\rho}{\partial \rho} = \frac{\partial}{\partial \rho}\left[\frac{\rho - R_1}{\rho}\right] \tag{10.240}$$

$$= \frac{R_1}{\rho^2}, \tag{10.241}$$

and, using (10.195),

$$\frac{\partial \epsilon_\varphi}{\partial \rho} = \frac{\partial}{\partial \rho}\left[\frac{\rho}{\rho - R_1 + \delta}\right] \tag{10.242}$$

$$= \frac{\delta - R_1}{(\rho + \delta - R_1)^2}. \tag{10.243}$$

Since $\rho = \|\bar{x}\|_2$ and $\varphi = \text{atan2}(x_2, x_1)$, combining (10.224), (10.228), (10.231), and (10.232)-(10.243) gives f in terms of known functions of $\bar{x}$ (μ_{33} and ϵ_{11} are given by (10.182) and (10.192) respectively while H_i is given by (10.215)).

In our example, we choose the radius of the perfect electric conducting cylinder as $R_1 = 0.1$ and the outer radius of the cloak as $R_2 = 2R_1$. We solve for the scattered field H_s in domain $\Omega = (-0.8, 0.8)^2 \setminus \Omega_c$ where Ω_c is the interior of the perfect electric conducting cylinder of radius R_1. The domain Ω is $32\lambda \times 32\lambda$ in size. A PML region with thicknesses $w_i = 0.2$ and decay rates $\sigma_i = 70$ is chosen to attenuate the scattered field. The mesh uses a maximum of 6 levels or refinement and a minimum of 4 levels of refinement. Degree 10 polynomials are used for basis functions outside the cloak layer and the degree is increased to 20 in a graded fashion approaching the perfect conducting cylinder. The grading is determined by the polynomial degree used to compute the Legendre expansions to represent the forcing function on each element (but limited to a maximum degree of 20). The degree distribution and mesh are shown in Figure 10.15. All Legendre expansions to represent variable parameters and curvilinear boundaries are computed to a tolerance of 10^{-12}.

The domain decomposition method is run until the preconditioned relative residual is reduced by ten orders of magnitude. Figure 10.15 also shows the preconditioned relative residual as a function of the number of iterations used to compute the scattered field. There are a total of 117,791 unknowns in $\bar{\phi}$ and 9,676 rows in the coarse problem matrix $\underline{K}$. Figure 10.16 illustrates the real part of the computed total field $H_{x_3} = H_i + H_s$ for the cloak problem as well as the magnitude of the scattered field. The magnitude of the scattered field is the error in the total field outside the cloak layer since the exact solution there is the incident field. Note that, as expected, the error in the cloak is large (on the order of 1 depicted in yellow) but that, immediately outside the cloak, the error is on the order of 10^{-5} (depicted in green) and that the error decays to zero in the PML (depicted in blue). The fact that we only recover 5 digits of accuracy is a consequence of the choice of the regularization parameter δ and not the finite element method itself. Decreasing δ decreases the error, but makes computing the Legendre expansions near the cylinder increasingly costly. The error is small in the PML because the PML cause the scattered wave to decay to zero which happens to be the desired solution for this example (in almost all other scattering problems, this decay would cause the error in the PML to be large; see Sections 9.6.2 or 10.6.1 for examples).

Finally, Figure 10.17 shows the RCS of the cloak, computed in the same way as in Section 9.6.2. We choose the surface for the far field integral to be the interface between the cloak and free space regions. The RCS confirms that the cloak is effective at reducing reflected waves, as expected based on the device design. In fact, the solution computed here is significantly more accurate than the one computed in [174].

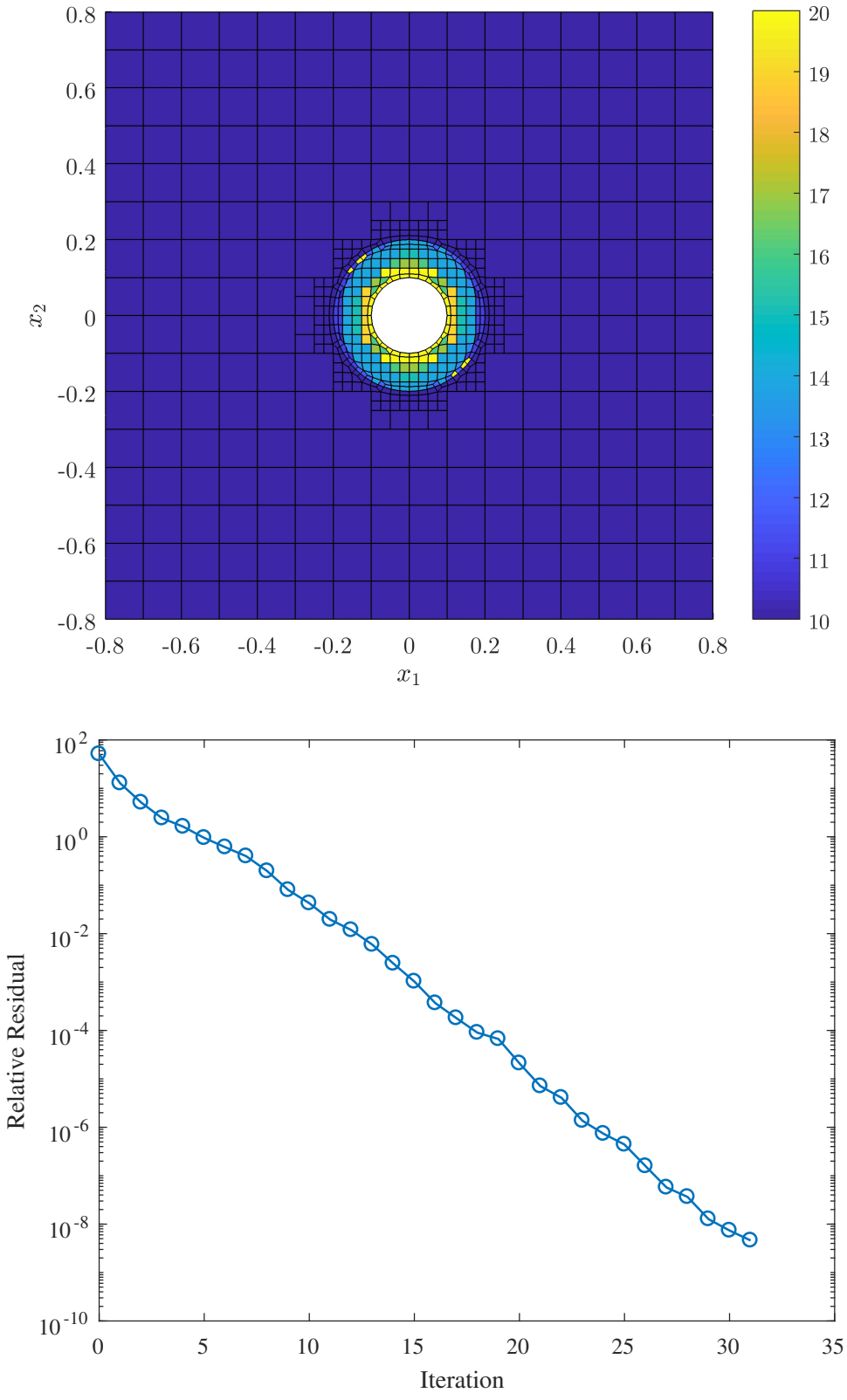

Figure 10.15: (top) Non-conforming mesh used to solve the electromagnetic cloak problem. Color denotes the polynomial degree of basis functions on each element. (bottom) Preconditioned relative residual as a function of the number of iterations used to compute the scattered field for the electromagnetic cloak problem.

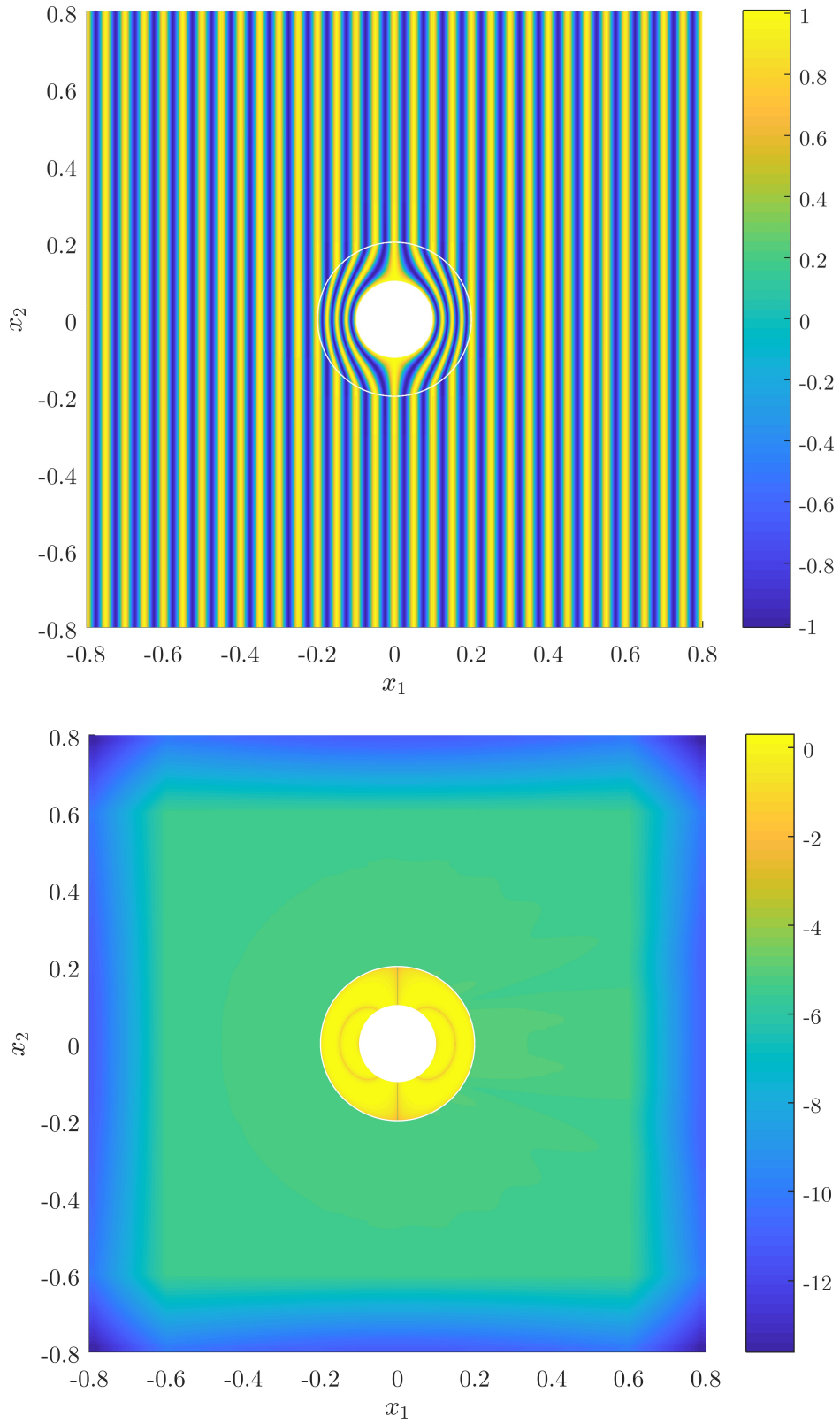

Figure 10.16: (top) Real part of the total field $\Re\{H_{x_3}\}$ for the electromagnetic cloak problem. (bottom) Logarithm of the absolute value of the scattered field $\log_{10}|H_s|$ for the electromagnetic cloak problem.

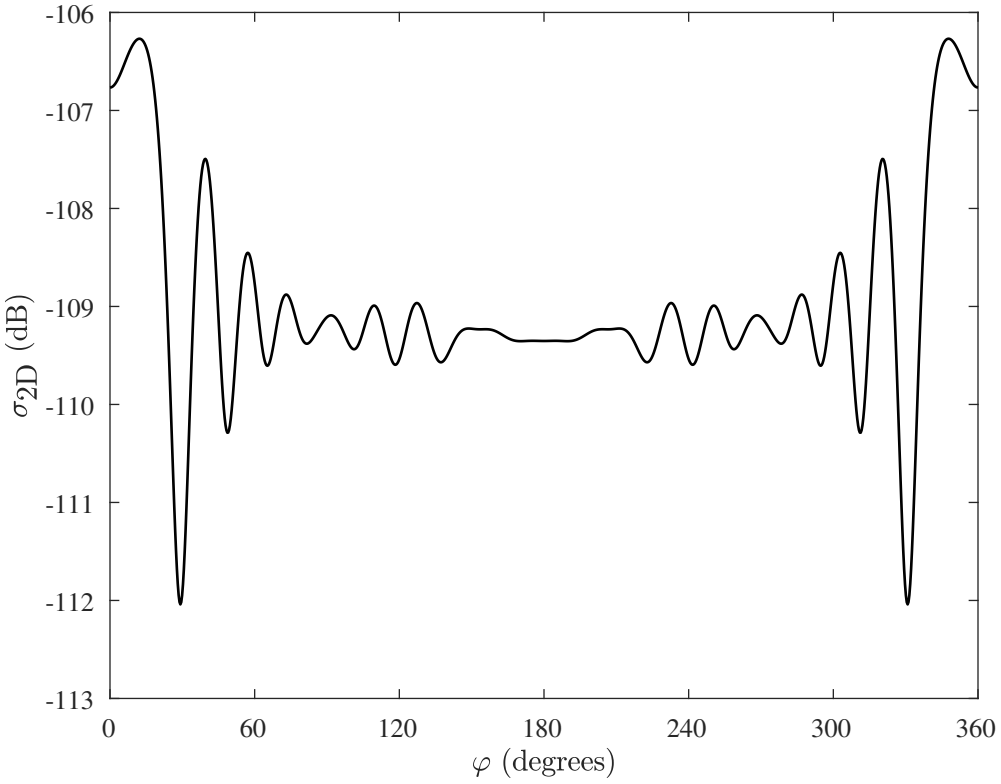

Figure 10.17: RCS of the electromagnetic cloak.

Chapter 11

Conclusion

The objective of the research reported in this thesis was to develop a high accuracy finite element method for the solution of Poisson and Helmholtz problems related to Maxwell's equations. This work has explored how to achieve this objective from first principles. In particular, the thesis first describes the development of a one-dimensional, single element spectral method dependent upon integrated Legendre basis functions and Legendre expansions computed using the fast Legendre transform. Then this single domain method is extended to a finite element method whose worst case computational complexity is $\mathcal{O}(N(\log N)^2)$ where N is the total number of unknowns.

The remainder of the thesis is focused on extending these results to higher-dimensional problems. First, the analog of the one-dimensional method for a single element is extended to a single square element, then to a planar curvilinear quadrilateral. Since curvilinear elements possess undesirable local sparsity patterns, a mesh generation procedure using curvilinear quadrilaterals only near curvilinear interfaces and boundaries was developed. To allow for h-adaption, a crucial component of the one-dimensional finite element method, the quadrilateral mesh is permitted to be non-conforming. This required a systematic development of constraint equations capable of handling continuity constraints between elements with arbitrary polynomial degree basis function mismatch and levels of refinement. To solve the associated saddle point system arising from the resulting finite element method, a domain decomposition method was also developed. This domain decomposition method leverages the fact that local element subproblems can be solved exploiting the structure of the one-dimensional basis functions. Finally, challenging high frequency electromagnetics problems were modeled and analyzed to demonstrate the ability of the finite element method to compute high accuracy solutions.

The main contributions of the theories and methods presented in this thesis are:

1. To utilize Legendre expansions to represent spatially varying parameters in a one-

dimensional finite element method and to use the fast Legendre transform to efficiently compute these expansions, and expansions for forcing functions.

2. To develop a systematic approach to allow arbitrary element refinement and polynomial degree mismatch between elements in higher-dimensional extensions of the one-dimensional finite element method in Contribution 1.

3. To develop a domain decomposition method specially tailored to the type of discretizations arising from Contribution 2 which can be used to efficiently solve the associated linear systems iteratively.

4. To explain and demonstrate how the domain decomposition method in Contribution 3 is suitable for both static and high frequency electromagnetic problems.

These contributions were essential to meeting the objective of this thesis: the development of a high accuracy finite element method for the solution of Poisson and Helmholtz problems.

11.1 Future Work

Since the domain decomposition method in this thesis is a generalization of the FETI-DP method and shares features similar to the FETI-DPH method meant to solve Helmholtz problems, it suffers from a similar computational bottleneck. In particular, the size of the coarse problem can be large for problems with fine mesh features or for high frequency problems. For this reason, future work is aimed towards reducing the size of the coarse problem. One way to do so when fine mesh features are present is to relax the constraint that each element belong to its own subdomain. This is particularly important near corner singularities, where a large number of refinements in an hp-adaptive method results in a high concentration of small elements near the singularity, but where the polynomial degree of basis functions need not be high. In such a scenario, it makes sense to treat these low degree elements together in a single subdomain of the domain decomposition, like in traditional domain decomposition methods [71], rather than each element as an independent subdomain. Another approach towards reducing the size of the coarse space involves varying the number of constraints explicitly imposed along edges in the mesh (denoted by parameter l). In this thesis, the parameter l is constant across the entire mesh, selected in accordance with the largest edge in the mesh. For non-uniformly refined meshes, this restriction may be more costly than needed and future work includes verifying whether l can be varied from edge to edge without significantly degrading the performance of the domain decomposition method.

In addition, the mesh generation procedure described in this thesis focuses on reducing the number of elements whose transfinite interpolation maps are not affine. The problem with such an approach is that the quality of the elements near curvilinear boundaries and interfaces can be poor. The number of terms in the Legendre expansions representing effective parameters for each element quantify this quality. Ideally, these Legendre expansions should have a small number of terms if the original parameters are smooth. However, if a curvilinear element has a transfinite map which is far from affine, the effective parameter will need to represent this variation and the resulting Legendre expansion can require many terms. This slows the convergence of fast solvers. Future work includes allowing more elements from the mesh to possess non-affine transfinite interpolation maps, but whose overall quality is higher than those in the current mesh generation procedure.

All computations in this thesis have been performed using MATLAB on a workstation with a four core Intel Xeon E5-1603 processor and 32 GB of RAM. Future work includes implementing the domain decomposition algorithm in C/C++ using MPI so as to test the parallel scalability of the method. This is particularly crucial when implementing three-dimensional problems with higher polynomial degree basis functions than those used in this thesis.

In three dimensions, Poisson and Helmholtz problems can be discretized similarly to their two-dimensional counterparts because of the tensor-product nature of basis functions. For each element, one obtains operator matrices comprised of sums of terms of the form $\underline{A} \otimes \underline{B} \otimes \underline{C}$ rather than $\underline{A} \otimes \underline{B}$ where the individual matrices $\underline{A}$, $\underline{B}$, or $\underline{C}$ are matrices arising in the one-dimensional formulation. These systems can also be solved using the fast method of [49]. Three-dimensional Legendre expansions can also be computed similarly using one- and two-dimensional techniques previously described [175]. In addition, the mesh generation techniques in this thesis are not dimension dependent, so they too can be generalized to three dimensions. Quadtrees are replaced by octrees and transfinite interpolation extends easily to three dimensions. All projections required to compute curvilinear boundaries in the mesh are dimension independent. Furthermore, continuity constraints are treated in much the same way as in two dimensions, although rather than impose constraints along edges, one must impose constraints along faces. The same one-dimensional properties of Legendre polynomials are used when imposing continuity along non-conforming faces. Finally, the domain decomposition method remains unchanged, although in three dimensions one must decide which constraints to use to form the coarse problem (face constraints, in addition to the usual vertex and edge constraints may need to be considered). In fact, the methods described in this thesis were developed specifically with an extension to higher dimensions in mind so that the main difficulty between a two- and three-dimensional implementation

would be one of book-keeping and programming rather than of new mathematical theory.

However, there are added theory complications when extending the method to the curl-curl equation for three-dimensional electromagnetic problems. In particular, crucial to the success of the method in this thesis is the development of a strong understanding of weak imposition of continuity constraints for non-conforming meshes. For the curl-curl equation, the appropriate inter-element continuity constraints must be tangential, and future work involves adapting the techniques from this thesis to such a setting.

Bibliography

[1] United Nations, International year of light and light-based technologies, 2015., http://www.light2015.org/Home/About/Resources.html, [Online; accessed February 2014]. Cited on pages 14 and 21.

[2] J. C. Maxwell, On physical lines of force, Philosophical Magazine 21-23 (1861) 161–174, 281–291, 338–348, 12–24, 85–95. Cited on page 14.

[3] O. Heaviside, Electrical Papers: Volume I, MacMillan, 1892, Ch. 30, p. 449. Cited on page 14.

[4] C. A. Balanis, Advanced Engineering Electromagnetics, John Wiley & Sons, 2012. Cited on pages 14, 226, 248, and 251.

[5] J. J. Bowman, T. B. A. Senior, P. L. E. Uslenghi (Eds.), Electromagnetic Acoustic Scattering by Simple Shapes, North-Holland Publishing Company, 1969. Cited on page 14.

[6] R. F. Harrington, Time-Harmonic Electromagnetic Fields, IEEE Press, 2001. doi:10.1109/9780470546710. Cited on page 14.

[7] J. D. Jackson, Classical Electrodynamics, John Wiley & Sons, 1999. Cited on page 14.

[8] J.-M. Jin, Theory and Computation of Electromagnetic Fields, IEEE Press, 2010. doi:10.1002/9780470874257. Cited on pages 14, 248, 290, and 291.

[9] J. A. Stratton, Electromagnetic Theory, McGraw-Hill, 1941. doi:10.1002/9781119134640. Cited on page 14.

[10] A. Bondeson, T. Rylander, P. Ingelstrom, Computational Electromagnetics, Springer, 2000. doi:10.1007/978-1-4614-5351-2. Cited on page 14.

[11] R. F. Harrington, Field Computation by Moment Methods, IEEE Press, 1993. doi:10.1109/9780470544631. Cited on page 14.

[12] J. Jin, The Finite Element Method in Electromagnetics, 3rd Edition, Wiley, 2014. Cited on pages 14, 18, 19, 19, 22, 55, 59, 60, 76, 103, 179, 226, 227, 233, 236, 301, 304, 304, and 307.

[13] A. Taflove, S. C. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, Artech House, 2000. Cited on page 14.

[14] M. Ainsworth, Discrete dispersion relation for *hp*-version finite element approximation at high wavenumber, SIAM J. Numer. Anal. 42 (2004) 553–575. doi:10.1137/S0036142903423460. Cited on pages 15, 21, 274, and 288.

[15] M. Ainsworth, H. A. Wajid, Dispersive and dissipative behavior of the spectral element method, SIAM J. Numer. Anal. 47 (2009) 3910–3937. doi:10.1137/080724976. Cited on page 15.

[16] F. Ihlenburg, Finite Element Analysis of Acoustic Scattering, Springer, 1991. doi:10.1007/b98828. Cited on page 15.

[17] J. M. Melenk, S. Sauter, Wavenumber explicit convergence analysis for Galerkin discretizations of the Helmholtz equation, SIAM J. Numer. Anal. 49 (2011) 1210–1243. doi:10.1137/090776202. Cited on pages 15 and 288.

[18] A. A. Oberai, P. M. Pinksy, A numerical comparison of finite element methods for the Helmholtz equation, Journal of Computational Acoustics 8 (2000) 211–221. doi:10.1142/S0218396X00000133. Cited on page 15.

[19] L. Zhu, H. Wu, Preasymptotic error analysis of CIP-FEM and FEM for Helmholtz equation with high wave number. Part II: *hp* version, SIAM J. Numer. Anal. 51 (2013) 1828–1852. doi:10.1137/120874643. Cited on page 15.

[20] A. Lieu, G. Gabard, H. Bériot, A comparison of high-order polynomial and wave-based methods for Helmholtz problems, Journal of Computational Physics 321 (2016) 105–125. doi:10.1016/j.jcp.2016.05.045. Cited on page 15.

[21] W. F. Mitchell, How high a degree is high enough for high order finite elements?, Procedia Computer Science 51 (2015) 246–255. doi:10.1016/j.procs.2015.05.235. Cited on page 15.

[22] P. E. J. Vos, S. J. Sherwin, R. M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretizations, Journal of Computational Physics 229 (2010) 5161–5181. doi:10.1016/j.jcp.2010.03.031. Cited on page 15.

[23] M. J. Gander, G. Wanner, From Euler, Ritz, and Galerkin to modern computing, SIAM Review 54 (2012) 627–666. doi:10.1137/100804036. Cited on page 16.

[24] P. P. Silvester, Finite element solution of homogeneous waveguide problems, Alta Frequenza 38 (1969) 313–317. Cited on page 18.

[25] G. H. Golub, C. F. V. Loan, Matrix Computations, 4th Edition, John Hopkins University Press, 2012. Cited on pages 18, 22, 54, 104, 144, 217, 218, 343, 362, 368, 369, 385, 387, 387, 390, and 392.

[26] J. C. Nedelec, Mixed finite elements in $\mathbb{R}^3$, Numer. Math. 35 (1980) 315–341. doi:10.1007/bf01396415. Cited on page 18.

[27] P. G. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland, 1978. doi:10.1016/s0168-2024(08)x7014-6. Cited on pages 18 and 22.

[28] P. P. Silvester, R. L. Ferrari, Finite Elements for Electrical Engineers, 3rd Edition, Cambridge University Press, 1996. doi:10.1017/cbo9781139170611. Cited on pages 19, 22, 55, and 60.

[29] P. Monk, Finite Element Methods for Maxwell's Equations, Oxford University Press, 2003. doi:10.1093/acprof:oso/9780198508885.001.0001. Cited on pages 19, 22, and 247.

[30] L. Demkowicz, P. Monk, L. Vardepetyan, W. Rachowicz, de Rham diagram for *hp* finite element spaces, Computers and Mathematics with Applications 39 (2000) 29–38. doi:10.1016/s0898-1221(00)00062-6. Cited on page 19.

[31] M. Ainsworth, J. Coyle, Hierarchic finite element bases on unstructured tetrahedral meshes, International Journal for Numerical Methods in Engineering 58 (2003) 2103–2130. doi:10.1002/nme.847. Cited on page 19.

[32] F. Fuentes, B. Keith, L. Demkowicz, S. Nagaraj, Orientation embedded high order shape functions for the exact sequence elements of all shapes, Computers and Mathematics 70 (2015) 353–458. doi:10.1016/j.camwa.2015.04.027. Cited on page 19.

[33] P. Solin, K. Segeth, I. Dolezal, High-Order Finite Element Methods, Chapman-Hall/CRC, 2004. doi:10.1201/9780203488041. Cited on pages 19 and 22.

[34] J. P. Webb, Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements, IEEE Transactions on Antennas and Propagation 47 (1999) 1244–1253. doi:10.1109/8.791939. Cited on page 19.

[35] A. Zdunek, W. Rachowicz, J. Kurtz, L. Demkowicz, M. Paszyriski, D. Pardo, Computing with *hp*-Adaptive Finite Elements: Volume 2, Chapman-Hall/CRC, 2008. doi:10.1201/9781420011692. Cited on page 19.

[36] S. J. Sherwin, G. E. Karniadakis, A triangular spectral element method; applications to the incompressible Navier-Stokes equations, Computer Methods in Applied Mechanics and Engineering 123 (1995) 189–229. doi:10.1016/0045-7825(94)00745-9. Cited on page 19.

[37] G. E. Karniadakis, S. J. Sherwin, Spectral/*hp* Element Methods for CFD, Oxford University Press, 1999. doi:10.1093/acprof:oso/9780198528692.001.0001. Cited on pages 19, 20, 75, and 136.

[38] M. Dubiner, Spectral methods on triangles and other domains, Journal of Scientific Computing 6 (1991) 345–390. doi:10.1007/bf01060030. Cited on pages 19 and 133.

[39] C. F. Dunkl, Y. Xu, Orthogonal Polynomials of Several Variables, Cambridge University Press, 2001. doi:10.1017/cbo9781107786134. Cited on pages 19 and 135.

[40] C. Canuto, A. Quarteroni, M. Y. Hussaini, T. A. Zang, Spectral Methods: Fundamentals in Single Domains, Springer, 2006. doi:10.1007/978-3-540-30726-6. Cited on pages 19 and 39.

[41] E. A. Coutsias, T. Hagstrom, D. Torres, An efficient spectral method for ordinary differential equations with rational function coefficients, Mathematics of Computation 65 (1996) 661–635. doi:10.1090/s0025-5718-96-00704-1. Cited on page 19.

[42] L. Greengard, Spectral integration and two-point boundary value problems, SIAM Journal of Numerical Analysis 28 (1991) 1071–1080. doi:10.21236/ada199805. Cited on page 19.

[43] J. Shen, T. Tang, L.-L. Wang, Spectral Methods: Algorithms, Analysis and Applications, Springer, 2011. doi:10.1007/978-3-540-71041-7. Cited on page 19.

[44] S. Beuchler, J. Schoberl, New shape functions for triangular *p*-FEM using integrated Jacobi polynomials, Numerische Mathematik 103 (2006) 339–366. doi:10.1007/s00211-006-0681-2. Cited on page 19.

[45] S. Beuchler, V. Pillwein, J. Schoberl, S. Zaglmayr, Numerical and Symbolic Scientific Computing, Springer, 2012, Ch. Sparsity optimized high order finite element functions on simplices, pp. 21–44. doi:10.1007/978-3-7091-0794-2_2. Cited on page 19.

[46] C. Canuto, R. Nochetto, M. Verani, Contraction and optimality properties of adaptive Legendre-Galerkin methods: The one-dimensional case, Computers & Mathematics with Applications 67 (2014) 752–770. doi:10.1016/j.camwa.2013.05.025. Cited on page 19.

[47] C. Canuto, V. Simoncini, M. Verani, Contraction and optimality properties of an adaptive Legendre-Galerkin method: The multi-dimensional case, Journal of Scientific Computing 63 (2014) 769–798. doi:10.1007/s10915-014-9912-3. Cited on page 19.

[48] B. Szabó, I. Babuška, Finite Element Analysis, Wiley, 1991. Cited on pages 19, 22, 62, and 75.

[49] J. Shen, Efficient spectral-Galerkin method I. Direct solvers for the second and fourth order equations using Legendre polynomials, SIAM J. Sci. Comput. 15 (1994) 1489–1505. doi:10.1137/0915089. Cited on pages 20, 20, 62, 75, 322, 378, and 385.

[50] A. Townsend, L. N. Trefethen, An extension of Chebfun to two dimensions, SIAM J. Sci. Comput. 35 (2013) C495–C518. doi:10.1137/130908002. Cited on pages 20, 144, 144, 144, 144, 144, and 208.

[51] A. Townsend, M. Webb, S. Olver, Fast polynomial transforms based on Toeplitz and Hankel matrices, Mathematics of Computation 87 (2018) 1913–1934. doi:10.1090/mcom/3277. Cited on pages 20, 62, 78, 342, 342, 342, 343, 359, 366, and 376.

[52] J. A. Gaunt, The triplets of helium, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 228 (1929) 151–196. doi:10.1098/rspa.1929.0037. Cited on pages 20 and 80.

[53] C. F. Borges, W. B. Gragg, A parallel divide and conquer algorithm for the generalized real symmetric definite tridiagonal eigenproblem, Tech. rep., Naval Postgraduate School (1992). doi:10.21236/ada262297. Cited on pages 20, 388, 388, and 394.

[54] M. Gu, S. C. Eisenstat, A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem, SIAM J. Matrix Anal. Appl. 15 (1994) 1266–1276. doi:10.1137/S089547989223924X. Cited on pages 20, 388, 388, 400, and 401.

[55] M. Gu, S. C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl. 16 (1995) 172–191. doi:10.1137/S0895479892241287. Cited on pages 20, 218, 388, 388, and 400.

[56] R.-C. Li, Solving secular equations stably and efficiently, Tech. rep., University of California at Berkeley (1993). doi:10.21236/ada608792. Cited on pages 20 and 388.

[57] J. D. Rutter, A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem, Tech. rep., University of California at Berkeley (1994). Cited on pages 20, 388, and 388.

[58] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J. Sci. Stat. Comput. 9 (1988) 669–686. doi:10.1137/0909044. Cited on pages 20, 218, 405, and 418.

[59] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, Journal of Computational Physics 155 (1999) 468–498. doi:10.1006/jcph.1999.6355. Cited on pages 20, 218, 405, and 418.

[60] A. Dutt, M. Gu, V. Rokhlin, Fast algorithms for polynomial interpolation, integration, and differentiation, SIAM J. Numer. Anal. 33 (1996) 1689–1711. doi:10.1137/0733082. Cited on pages 20, 218, 405, and 421.

[61] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 135 (1987) 280–292. doi:10.1006/jcph.1997.5706. Cited on pages 20, 218, 405, and 416.

[62] P. J. Frey, P.-L. George, Mesh Generation: Application to Finite Elements, 2nd Edition, ISTE and Wiley, 2008. doi:10.1002/9780470611166. Cited on pages 20, 132, 195, 195, and 206.

[63] R. Schneiders, A grid-based algorithm for the generation of hexahedral element meshes, Engineering with Computers 12 (1996) 168–177. doi:10.1007/BF01198732. Cited on page 20.

[64] P.-O. Persson, Mesh generation for implicit geometries, Ph.D. thesis, Massachusetts Institute of Technology (2005). http://hdl.handle.net/1721.1/27866. Cited on pages 20 and 185.

[65] S. J. Owen, M. L. Staten, M. C. Sorensen, Parallel hex meshing from volume fractions, in: W. R. Quadros (Ed.), Proceedings of the 20th International Meshing Roundtable, Springer, 2011. doi:10.1007/978-3-642-24734-7_9. Cited on page 20.

[66] Y. Ohtake, A. Belyaev, I. Bogaevski, Mesh regularization and adaptive smoothing, Computer-Aided Design 33 (2001) 789–800. doi:10.1016/S0010-4485(01)00095-1. Cited on pages 20, 196, and 202.

[67] P. M. Knupp, Hexahedral and tetrahedral mesh untangling, Engineering with Computers 17 (2001) 261–268. doi:10.1007/s003660170006. Cited on pages 20, 196, and 203.

[68] P. Knupp, Introducing the target-matrix paradigm for mesh optimization via node-movement, in: S. Shontz (Ed.), Proceedings of the 19th International Meshing Roundtable, Springer, 2010, pp. 67–83. doi:10.1007/978-3-642-15414-0_5. Cited on pages 20, 196, and 204.

[69] H. E. Salzer, A recurrence scheme for converting from one orthogonal expansion into another, Communications of the ACM 16 (1973) 705–707. doi:10.1145/355611.362548. Cited on pages 20 and 217.

[70] H. Yserentant, Preconditioning indefinite discretization matrices, Numer. Math. 54 (1988) 719–734. doi:10.1007/BF01396490. Cited on pages 20 and 261.

[71] A. Toselli, O. Widlund, Domain Decomposition Methods – Algorithms and Theory, Springer, 2005. doi:10.1007/b137868. Cited on pages 20, 224, 261, 270, and 321.

[72] W. Hackbusch, Multi-Grid Methods and Applications, Springer, 2003. doi:10.1007/978-3-662-02427-0. Cited on pages 20 and 224.

[73] L. L. Thompson, A review of finite-element methods for time-harmonic acoustics, J. Acoust. Soc. Am. 119 (2006) 1315–1330. doi:10.1121/1.2164987. Cited on page 20.

[74] Y. A. Erlangga, Advances in iterative methods and preconditioners for the Helmholtz equation, Arch. Comput. Methods Eng. 15 (2008) 37–66. doi:10.1007/s11831-007-9013-7. Cited on page 20.

[75] B. Engquist, L. Ying, Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation, Communications on Pure and Applied Mathematics 64 (2011) 697–735. doi:10.1002/cpa.20358. Cited on page 21.

[76] J. Poulson, B. Engquist, S. Li, L. Ying, A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations, SIAM J. Sci. Comput. 35 (2013) C194–C212. doi:10.1137/120871985. Cited on page 21.

[77] C. Farhat, F.-X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, International Journal for Numerical Methods in Engineering 32 (1991) 1205–1227. doi:10.1002/nme.1620320604. Cited on page 21.

[78] A. Klawonn, L. F. Pavarino, O. Rheinbach, Spectral element FETI-DP and BDDC preconditioners with multi-element subdomains, Comput. Methods Appl. Mech. Engrg. 198 (2008) 511–523. doi:10.1016/j.cma.2008.08.017. Cited on pages 21, 256, 269, 272, 274, 281, 282, 282, 282, 282, and 284.

[79] J. Mandel, B. Sousedík, BDDC and FETI-DP under minimalist assumptions, Computing 81 (2007) 269–280. doi:10.1007/s00607-007-0254-y. Cited on pages 21 and 256.

[80] D. Stefanica, FETI and FETI-DP methods for spectral and mortar spectral elements: a performance comparison, Journal of Scientific Computing 17 (2002) 629–638. doi:10.1023/A:1015130915856. Cited on pages 21 and 256.

[81] C. Farhat, P. Avery, R. Tezaur, J. Li, FETI-DPH: a dual-primal domain decomposition method for acoustic scattering, Journal of Computational Acoustics 13 (2005) 499–524. doi:10.1142/S0218396X05002761. Cited on pages 21, 275, and 277.

[82] M. J. Gander, H. Zhang, Domain decomposition methods for the Helmholtz equation: A numerical investigation, in: R. Bank, M. Holst, O. Widlund, J. Xu (Eds.), Domain Decomposition Methods in Science and Engineering XX, Vol. 91 of Lecture Notes in Computational Science and Engineering, Springer, 2013, pp. 215–222. doi:10.1007/978-3-642-35275-1_24. Cited on page 21.

[83] F. B. Belgacem, The mortar finite element method with Lagrange multipliers, Numerische Mathematik 84 (1999) 173–197. doi:10.1007/s002110050468. Cited on page 21.

[84] J.-D. Benamou, B. Després, A domain decomposition method for the Helmholtz equation and related optimal control problems, Journal of Computational Physics 136 (1997) 68–82. doi:10.1006/jcph.1997.5742. Cited on pages 21 and 277.

[85] C. Farhat, A. Macedo, M. Lesoinne, F.-X. Roux, F. Magoulès, A. de La Bourdonnaie, Two-level domain decomposition methods with Lagrange multipliers for the fast iterative solution of acoustic scattering problems, Comput. Methods Appl. Mech. Engrg. 184 (2000) 213–239. doi:10.1016/S0045-7825(99)00229-7. Cited on pages 21, 277, 280, and 281.

[86] J.-P. Berenger, A perfectly matched layer for the absorption of electromagnetic waves, Journal of Computational Physics 114 (1994) 185–200. doi:10.1006/jcph.1994.1159. Cited on pages 21 and 237.

[87] S. Johnson, Notes on perfectly matched layers (PMLs), Tech. rep., Massachusetts Institute of Technology (2010). http://math.mit.edu/~stevenj/notes.html. Cited on pages 21 and 238.

[88] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004. doi:10.1017/CBO9780511804441. Cited on pages 22, 48, and 257.

[89] R. Fletcher, Practical Methods of Optimization, John Wiley & Sons, 1987. doi:10.1002/9781118723203. Cited on page 22.

[90] J. Nocedal, S. Wright, Numerical Optimization, Springer, 2006. doi:10.1007/b98874. Cited on pages 22, 204, and 257.

[91] J. W. Demmel, Applied Numerical Linear Algebra, SIAM, 1996. doi:10.1137/1.9781611971446. Cited on pages 22, 390, and 392.

[92] L. N. Trefethen, D. Bau, Numerical Linear Algebra, SIAM, 1997. doi:10.1137/1.9780898719574. Cited on page 22.

[93] D. S. Watkins, Fundamentals of Matrix Computations, John Wiley & Sons, 2002. doi:10.1002/0471249718. Cited on page 22.

[94] P. B. Bochev, M. D. Gunzberger, Least-Squares Finite Element Methods, Springer, 2009. doi:10.1007/978-3-540-70529-1_330. Cited on page 22.

[95] S. C. Brenner, L. R. Scott, The Mathematical Theory of Finite Element Methods, Springer, 2008. doi:10.1007/978-1-4757-4338-8_7. Cited on page 22.

[96] L. Demkowicz, Computing with *hp*-Adaptive Finite Elements: Volume 1, Chapman-Hall/CRC, 2007. doi:10.1201/9781420011685. Cited on page 22.

[97] K.-J. Bathe, Finite Element Procedures in Engineering Analysis, Prentice-Hall, 1982. Cited on page 22.

[98] W. B. Bickford, A First Course in the Finite Element Method, Irwin, 1994. Cited on page 22.

[99] O. C. Zienkiewicz, R. L. Taylor, The Finite Element Method: Volume 1, Butterworth-Heinemann, 2000. Cited on page 22.

[100] G. Szego, Orthogonal Polynomials, American Mathematical Society, 1975. doi:10.1090/coll/023. Cited on pages 22, 30, and 31.

[101] S. C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, McGraw-Hill, 2012. Cited on page 24.

[102] S. C. Chapra, R. P. Canale, Numerical Methods for Engineers, McGraw-Hill, 1998. Cited on page 24.

[103] S. S. Rao, Applied Numerical Methods for Engineers and Scientists, Prentice Hall, 2002. Cited on page 24.

[104] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes in C, Cambridge University Press, 1988. Cited on page 24.

[105] A. Ralston, P. Rabinowitz, A First Course in Numerical Analysis, Dover, 2001. Cited on page 24.

[106] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer, 2000. doi:10.1007/b98885. Cited on pages 24 and 26.

[107] E. Isaacson, H. B. Keller, Analysis of Numerical Methods, Dover, 1994. Cited on pages 24 and 38.

[108] M. Cox, Reliable determination of interpolating polynomials, Numerical Algorithms 5 (1993) 133–154. doi:10.1007/bf02215677. Cited on pages 25 and 55.

[109] L. N. Trefethen, Approximation Theory and Approximation Practice, Cambridge University Press, 2012. Cited on pages 28, 56, 61, 61, 79, and 225.

[110] L. N. Trefethen, Is Gauss Quadrature better than Clenshaw-Curtis?, SIAM Review 50 (2008) 67–87. doi:10.1137/060659831. Cited on pages 28 and 254.

[111] J. Waldvogel, Fast construction of the Fejer and Clenshaw-Curtis quadrature rules, BIT Numerical Mathematics 43 (2003) 001–018. doi:10.1007/s10543-006-0045-4. Cited on page 32.

[112] N. Higham, Stability analysis of algorithms for solving confluent Vandermonde-like systems, SIAM Journal of Applied Matrix Analysis 11 (1990) 23–41. doi:10.1137/0611002. Cited on pages 32 and 77.

[113] R. Courant, D. Hilbert, Methods of Mathematical Physics: Volume 1, John Wiley & Sons, 1989. doi:10.1002/9783527617210. Cited on page 46.

[114] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numerica 14 (1999) 1–137. doi:10.1017/s0962492904000212. Cited on pages 51, 53, 257, and 259.

[115] J.-P. Berrut, L. N. Trefethen, Barycentric Lagrange interpolation, SIAM Review 46 (2004) 501–517. doi:10.1137/s0036144502417715. Cited on page 56.

[116] L. N. Trefethen, Spectral Methods in MATLAB, SIAM, 2000. doi:10.1137/1.9780898719598. Cited on pages 61 and 162.

[117] Symmetries of orthogonal polynomials, http://dlmf.nist.gov/18.6, [Online; accessed August 2015]. Cited on page 69.

[118] I. Bogaert, Iteration-free computation of Gauss–Legendre quadrature nodes and weights, SIAM Journal on Scientific Computing 36 (2014) A1008–A1026. doi:10.1137/140954969. Cited on page 78.

[119] B. K. Alpert, V. Rokhlin, A fast algorithm for the evaluation of Legendre expansions, SIAM Journal on Scientific and Statistical Computing 12 (1991) 158–179. doi:10.1137/0912009. Cited on pages 78 and 342.

[120] N. Hale, A. Townsend, A fast, simple, and stable Chebyshev-Legendre transform using an asymptotic formula, SIAM Journal on Scientific Computing 36 (2014) A148–A167. doi:10.1137/130932223. Cited on pages 78 and 342.

[121] D. Potts, G. Steidl, M. Tasche, Fast algorithms for discrete polynomial transforms, Mathematics of Computation 67 (1998) 1577–1590. doi:10.1090/s0025-5718-98-00975-2. Cited on pages 78 and 342.

[122] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (2009) 455–500. doi:10.1137/07070111x. Cited on page 80.

[123] Properties of the $3j$ symbol, http://dlmf.nist.gov/34.3, [Online; accessed August 2015]. Cited on page 80.

[124] J. C. Adams, On the expression of the product of any two Legendre's coefficients by means of a series of Legendre's coefficients, Proc. R. Soc. Lond. 27 (1878) 63–71. doi:10.1098/rspl.1878.0016. Cited on page 80.

[125] P. Houston, E. Süli, A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations, Comput. Methods Appl. Mech. Engrg. 194 (2005) 229–243. doi:10.1016/j.cma.2004.04.009. Cited on pages 91 and 91.

[126] C. Bernardi, Indicateurs d'erreur en h-n version des éléments spectraux, Modélisation Mathémathique et Analyse Numérique 30 (1996) 1–38. doi:10.1051/m2an/1996300100011. Cited on page 91.

[127] P. J. Davis, Interpolation and Approximation, Dover Publications, 1975. Cited on page 91.

[128] T. P. Wihler, An hp-adaptive strategy based on continuous Sobolev embeddings, Journal of Computational and Applied Mathematics 235 (2011) 2731–2739. doi:10.1002/pamm.201110004. Cited on page 92.

[129] B. Gustafsson, H.-O. Kreiss, J. Oliger, Time Dependent Problems and Difference Methods, Wiley-Interscience, 1995. doi:10.1002/9781118548448. Cited on page 102.

[130] R. J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM, 2007. doi:10.1137/1.9780898717839. Cited on page 102.

[131] E. Hairer, S. P. Norsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer, 1993. doi:10.1007/978-3-540-78862-1. Cited on page 102.

[132] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer, 1996. doi:10.1007/978-3-642-05221-7. Cited on page 102.

[133] D. A. French, A space-time finite element method for the wave equation, Computer Methods in Applied Mechanics and Engineering 107 (1993) 145–157. doi:10.1016/0045-7825(93)90172-t. Cited on page 102.

[134] G. M. Hulbert, T. J. R. Hughes, Space-time finite element methods for second-order hyperbolic equations, Computer Methods in Applied Mechanics and Engineering 84 (1990) 327–348. doi:10.1016/0045-7825(90)90082-w. Cited on page 102.

[135] D. K. Cheng, Field and Wave Electromagnetics, Addison-Wesley, 1989. Cited on page 116.

[136] R. Kashyap, Fiber Bragg Gratings, Elsevier, 2010. doi:10.1016/c2009-0-16830-7. Cited on page 116.

[137] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, M. Gross, Polyhedral finite elements using harmonic basis functions, in: Proceedings of the Symposium on Geometry Processing, 2008. doi:10.1111/j.1467-8659.2008.01293.x. Cited on page 132.

[138] T. Mukherjee, J. P. Webb, Hierarchical bases for polygonal finite elements, IEEE Transactions on Magnetics 51 (2015) 1–4. doi:10.1109/TMAG.2014.2345497. Cited on page 132.

[139] T. Mukherjee, J. P. Webb, Polygonal finite elements of arbitrary order, IEEE Transactions on Magnetics 52 (2016) 1–4. doi:10.1109/TMAG.2015.2487245. Cited on page 132.

[140] S.-W. Cheng, T. K. Dey, J. R. Shewchuk, Delaunay Mesh Generation, CRC Press, 2012. doi:10.1201/b12987. Cited on page 132.

[141] G. Munschy, P. Pluvinage, Résolution de l'équation de Schrodinger des atomes à deux électrons. ii. Méthode rigoureuse. États s symétriques, J. Phys. Radium 18 (1957) 157–160. doi:10.1051/jphysrad:01957001803015700. Cited on page 133.

[142] J. Proriol, Sur une famille de polynomes à deux variables orthogonaux dans un triangle, CR Acad. Sci. Paris 245 (1957) 2459–2461. Cited on page 133.

[143] T. Koornwinder, Two-variable analogues of the classical orthogonal polynomials, in: R. A. Askey (Ed.), Theory and Application of Special Functions, Academic Press, 1975, pp. 435–495. doi:10.1016/b978-0-12-064850-4.50015-x. Cited on page 133.

[144] S. Alisauskas, Coupling coefficients of SO(n) and integrals involving Jacobi and Gegenbauer polynomials, J. Phys. A: Math. Gen. 35 (2002) 7323–7345. doi:10.1088/0305-4470/35/34/307. Cited on page 135.

[145] J. S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods, Springer, 2008. doi:10.1007/978-0-387-72067-8. Cited on page 136.

[146] A. Townsend, S. Olver, The automatic solution of partial differential equations using a global spectral method, Journal of Computational Physics 299 (2015) 106–123. doi:10.1016/j.jcp.2015.06.031. Cited on pages 162 and 169.

[147] T. A. Davis, Direct Methods for Sparse Linear Systems, SIAM, 2006. doi:10.1137/1.9780898718881. Cited on pages 162, 256, and 269.

[148] L. C. Evans, Partial Differential Equations, American Mathematical Society, 1997. doi:10.1090/gsm/019. Cited on page 169.

[149] P. J. Roache, Code verification by the method of manufactured solutions, J. Fluids Eng. 124 (2001) 4–10. doi:10.1115/1.1436090. Cited on page 169.

[150] S. A. Coons, Surfaces for computer-aided design of space forms, Tech. rep., Massachusetts Institute of Technology (1967). doi:10.21236/ad0663504. Cited on page 178.

[151] W. J. Gordon, Blending-function methods of bivariate and multivariate interpolation and approximation, SIAM J. Numer. Anal. 8 (1971) 158–177. doi:10.1137/0708019. Cited on page 178.

[152] W. J. Gordon, C. A. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, International Journal for Numerical Methods in Engineering 7 (1973) 461–477. doi:10.1002/nme.1620070405. Cited on pages 178 and 178.

[153] G. Farin, Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Academic Press, 1997. doi:10.1016/c2009-0-22351-8. Cited on page 178.

[154] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003. doi:10.1007/b98879. Cited on pages 183, 197, and 198.

[155] W. Rudin, Principles of Mathematical Analysis, McGraw-Hill, 1976. Cited on page 183.

[156] G. F. Carey, Computational Grids: Generation, Adaptation, and Solution Strategies, Taylor & Francis, 1997. Cited on page 195.

[157] R. Schneiders, Algorithms for quadrilateral and hexahedral mesh generation, in: Proceedings of the VKI Lecture Series on Computational Fluid Dynamics, 2000. Cited on page 196.

[158] S. J. Owen, T. R. Shelton, Validation of grid-based hex meshes with computational solid mechanics, in: J. Sarrate, M. Staten (Eds.), Proceedings of the 22nd International Meshing Roundtable, Springer, 2014, pp. 39–56. doi:10.1007/978-3-319-02335-9_3. Cited on pages 196 and 200.

[159] W. Gautschi, Orthogonal Polynomials: Computation and Approximation, Oxford University Press, 2004. Cited on pages 216 and 425.

[160] I. Bogaert, B. Michiels, J. Fostier, $\mathcal{O}(1)$ computation of Legendre polynomials and Gauss–Legendre nodes and weights for parallel computing, SIAM Journal on Scientific Computing 34 (2012) C83–C101. doi:10.1137/110855442. Cited on page 218.

[161] L. N. Trefethen, Multivariate polynomial approximation in the hypercube, Proceedings of the American Mathematical Society 145 (2017) 4837–4844. doi:10.1090/proc/13623. Cited on pages 225 and 225.

[162] Recurrence relations and derivatives of Bessel functions, http://dlmf.nist.gov/10.6, [Online; accessed December 2016]. Cited on page 249.

[163] D. Colton, R. Kress, Integral Equation Methods in Scattering Theory, Wiley-Interscience Publication, 1983. doi:10.1137/1.9781611973167. Cited on page 251.

[164] Definitions of Bessel functions, http://dlmf.nist.gov/10.2, [Online; accessed July 2018]. Cited on page 251.

[165] L. N. Trefethen, J. A. C. Weideman, The exponentially convergent trapezoidal rule, SIAM Review 56 (2014) 385–458. doi:10.1137/130932132. Cited on page 254.

[166] H. C. Elman, Preconditioning for the steady-state Navier-Stokes equations with low viscosity, SIAM J. Sci. Comput. 20 (1999) 1299–1316. doi:10.1137/S1064827596312547. Cited on pages 257 and 260.

[167] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations, SIAM, 2004. doi:10.1137/1.9781611970944. Cited on pages 260, 274, 274, 276, 283, and 284.

[168] D. J. Rixen, C. Farhat, A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems, International Journal for Numerical Methods in Engineering 44 (1999) 489–516. doi:10.1002/(SICI)1097-0207(19990210)44:4<489::AID-NME514>3.0.CO;2-Z. Cited on page 271.

[169] D. Colton, R. Kress, Inverse Acoustic and Electromagnetic Scattering Theory, 3rd Edition, Springer, 2013. doi:10.1007/978-1-4614-4942-3. Cited on page 280.

[170] M. Naor, L. Stockmeyer, What can be computed locally?, SIAM J. Comput. 24 (1995) 1259–1277. doi:10.1137/S0097539793254571. Cited on page 281.

[171] F. Vico, L. Greengard, M. Ferrando, Fast convolution with free-space Green's functions, Journal of Computational Physics 323 (2016) 191–203. doi:10.1016/j.jcp.2016.07.028. Cited on page 295.

[172] A. Gillman, A. H. Barnett, P.-G. Martinsson, A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media, BIT Numer. Math. 55 (2015) 141–170. doi:10.1007/s10543-014-0499-8. Cited on page 299.

[173] A. Sharkawy, S. Shi, D. W. Prather, Implementations of optical vias in high-density photonic crystal optical networks, Journal of Micro/Nanolithography, MEMS, and MOEMS 2 (2003) 300–308. doi:10.1117/1.1610480. Cited on page 299.

[174] S. A. Cummer, B.-I. Popa, D. Schurig, D. R. Smith, J. Pendry, Full-wave simulations of electromagnetic cloaking structures, Physical Review E 74 (2006) 036621(1–5). doi:10.1103/physreve.74.036621. Cited on pages 307 and 316.

[175] B. Hashemi, L. N. Trefethen, Chebfun in three dimensions, SIAM J. Sci. Comput. 39 (2017) C341–C363. doi:10.1137/16M1083803. Cited on page 322.

[176] C. V. Loan, Computational Frameworks for the Fast Fourier Transform, SIAM, 1992. doi:10.1137/1.9781611970999. Cited on pages 343 and 352.

[177] J. G. Proakis, D. G. Manolakis, Digital Signal Processing, Prentice-Hall, 1996. Cited on page 343.

[178] G. E. Andrews, R. Askey, R. Roy, Special Functions, Cambridge University Press, 1999. doi:10.1017/cbo9781107325937. Cited on pages 353, 354, and 354.

[179] Interrelations and limit relations of orthogonal polynomials, https://dlmf.nist.gov/18.7, [Online; accessed May 2018]. Cited on page 353.

[180] Recurrence relations and derivatives of orthogonal polynomials, https://dlmf.nist.gov/18.9, [Online; accessed May 2018]. Cited on page 356.

[181] C. F. V. Loan, The ubiquitous Kronecker product, Journal of Computational and Applied Mathematics 123 (2000) 85–100. doi:10.1016/s0377-0427(00)00393-9. Cited on page 379.

[182] V. Simoncini, Computational methods for linear matrix equations, SIAM Review 58 (2016) 377–441. doi:10.1137/130912839. Cited on page 379.

[183] D. Fortunato, A. Townsend, Fast Poisson solvers for spectral methods, preprint (2017). https://arxiv.org/abs/1710.11259. Cited on page 379.

[184] E. Jarlebring, G. Mele, D. Palitta, E. Ringh, Krylov methods for low-rank commuting generalized Sylvester equations, Numerical Linear Algebra with Applications (2018). doi:10.1002/nla.2176. Cited on page 379.

[185] J.-P. Chehab, M. Raydan, An implicit preconditioning strategy for large-scale generalized Sylvester equations, Applied Mathematics and Computation 217 (2011) 8793–8803. doi:10.1016/j.amc.2011.03.148. Cited on page 379.

[186] A. Bouhamidi, K. Jbilou, A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications, Applied Mathematics and Computation 206 (2008) 687–694. doi:10.1016/j.amc.2008.09.022. Cited on page 379.

[187] B. Bialecki, G. Fairweather, A. Karageorghis, Matrix decomposition algorithms for elliptic boundary value problems: a survey, Numer Algor 56 (2011) 253–295. doi:10.1007/s11075-010-9384-y. Cited on page 385.

[188] S. Li, X. Liao, J. Liu, H. Jiang, New fast divide-and-conquer algorithms for the symmetric tridiagonal eigenvalue problem, Numer. Linear Algebra Appl. 23 (2016) 656–673. doi:10.1002/nla.2046. Cited on page 388.

[189] J. W. Demmel, O. A. Marques, B. N. Parlett, C. Vömel, Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers, SIAM J. Sci. Comput. 30 (2008) 1508–0526. doi:10.1137/070688778. Cited on page 392.

[190] T. Z. Boulmezaoud, J. M. Urquiza, On the eigenvalues of the spectral second order differentiation operator and application to the boundary observability of the wave equation, Journal of Scientific Computing 31 (2007) 307–345. doi:10.1007/s10915-006-9106-8. Cited on page 422.

[191] G. N. Watson, A Treatise on the Theory of Bessel Functions, Cambridge University Press, 1922. Cited on pages 422, 422, 422, and 428.

[192] J. Gard, E. Zakrajšek, Method for evaluation of zeros of Bessel functions, Inst. Maths Applics 11 (1973) 57–72. doi:10.1093/imamat/11.1.57. Cited on page 423.

[193] Definitions and basic properties of spherical Bessel functions, https://dlmf.nist.gov/10.47, [Online; accessed July 2018]. Cited on page 426.

[194] Relations to other functions for Bessel functions, https://dlmf.nist.gov/10.16, [Online; accessed July 2018]. Cited on page 426.

[195] K. F. Lee, R. Wong, Asymptotic expansion of the modified Lommel polynomials $h_{n,\nu}(x)$ and their zeros, Proceedings of the AMS 142 (2014) 3953–3964. doi:10.1090/s0002-9939-2014-12134-4. Cited on page 430.

[196] G. Matviyenko, On the evaluation of Bessel functions, Applied and Computational Harmonic Analysis 1 (1993) 116–135. doi:10.1006/acha.1993.1009. Cited on page 430.

[197] Uniform asymptotic expansions for large order Bessel functions, https://dlmf.nist.gov/10.20, [Online; accessed July 2018]. Cited on page 430.

Appendix A

The Fast Legendre Transform

This appendix describes the Fast Legendre Transform (FLT) which converts an approximation of a continuous function $\phi(x)$ defined for $x \in (-1, 1)$ via interpolatory Lagrange polynomials $l_i(x)$ at Chebyshev nodes

$$x_i = \cos\left(\frac{\pi}{n} i\right), \qquad i = 0, 1, 2, ..., n, \tag{A.1}$$

into an equivalent expansion in terms of orthonormal Legendre polynomials $p_i(x)$. That is, the algorithm converts between samples $\phi(x_i)$ of the function and Legendre coefficients $\phi_{i,\text{Leg}}$ so that the two representations

$$\phi(x) \approx \sum_{i=0}^{n} \phi(x_i) l_i(x), \qquad \phi(x) \approx \sum_{i=0}^{n} \phi_{i,\text{Leg}} p_i(x), \tag{A.2}$$

are equivalent. Many such algorithms exist to perform this task with near linear computational complexity [119, 120, 121]. In particular, the method described in [51] performs this task with near linear computational complexity $\mathcal{O}(n(\log n)^2)$. The method exploits the Fast Fourier Transform (FFT) and connections between Chebyshev polynomials, Legendre polynomials, and interpolation in Chebyshev nodes to perform the transform[1]. In this document, we focus on the transform taking samples to coefficients because the reverse process is similar. For details regarding the inverse transform, see [51].

The key steps in the algorithm are to:

1. Sample the function at the Chebyshev nodes
2. Use an FFT to calculate Chebyshev coefficients from the samples
3. Convert the Chebyshev coefficients into Legendre coefficients through a series of FFTs.

[1] I favor the method of [51] because of its simple implementation.

Steps 1 and 2 describe a Fast Chebyshev Transform (FCT). The FCT converts function values $f(x_i)$ evaluated at Chebyshev nodes into coefficients $\phi_{i,\text{Cheb}}$ in the Chebyshev expansion

$$\phi(x) \approx \sum_{i=0}^{n} \phi_{i,\text{Cheb}} T_i(x) \tag{A.3}$$

where $T_i(x)$ are the Chebyshev polynomials of the first kind. If $\bar{\phi}_{\text{Leg}}$ is the vector containing the Legendre coefficients and $\bar{\phi}_{\text{Cheb}}$ the vector containing the Chebyshev coefficients, then Step 3 exploits the fact that

$$\bar{\phi}_{\text{Leg}} = \underline{D}_2(\underline{H} \circ \underline{T}^T)\underline{D}_1 \bar{\phi}_{\text{Cheb}} \tag{A.4}$$

where $\underline{D}_1$ and $\underline{D}_2$ are diagonal matrices, $\underline{H}$ is a numerically low rank symmetric Hankel matrix that is almost positive semidefinite (we will elaborate later on precisely how it almost possesses this property), $\underline{T}$ is a Toeplitz matrix, and $\circ$ denotes the Hadamard product (multiplication entrywise). The method first computes a rank-revealing LDLT decomposition of $\underline{H}$ that exploits its Hankel and positive semidefinite nature to obtain a factorization

$$\underline{H} \approx \sum_{k=1}^{r} d_k \bar{l}_k \bar{l}_k^T \tag{A.5}$$

where r is the numerical rank of the matrix. Then using this factorization, (A.4) becomes

$$\bar{\phi}_{\text{Leg}} = \underline{D}_2 \left[\sum_{k=1}^{r} d_k \text{diag}(\bar{l}_k) \underline{T}^T \text{diag}(\bar{l}_k) \right] \underline{D}_1 \bar{\phi}_{\text{Cheb}}. \tag{A.6}$$

All multiplications with diagonal matrices are performed with $\mathcal{O}(n)$ operations and each multiplication with the Toeplitz matrix is performed using an FFT with $\mathcal{O}(n \log n)$ operations by embedding the Toeplitz matrix inside a larger circulant matrix. In [51], the authors prove that r is approximately $\log n$ for the matrix $\underline{H}$ arising in the FLT, so that the total cost of this approach is $\mathcal{O}(n(\log n)^2)$.

In the following, we will take for granted the fact that the Discrete Fourier Transform (DFT) and its inverse, which we will define shortly, can be computed efficiently using the FFT and inverse FFT. The interested reader unfamiliar with how the FFT implements the DFT efficiently may find the following matrix computations and digital signal processing literature helpful [25, 176, 177]. Otherwise, we will work from first principles, using only a handful of special function relations to describe the algorithm given by Steps 1, 2 and 3.

A.1 The Fast Chebyshev Transform

We begin with a description of the FCT. The goal is to find a Chebyshev expansion in terms of Chebyshev polynomials of the first kind $T_j(x) = \cos(j\cos^{-1}(x))$ that interpolates the function ϕ at the Chebyshev nodes. We can determine the coefficients in the expansion by writing out the $n+1$ interpolation equations

$$\phi(x_i) = \sum_{j=0}^{n} \phi_{j,\text{Cheb}} T_j(x_i), \qquad i = 0, 1, 2, ..., n, \tag{A.7}$$

where the nodes x_i satisfy (A.1). In matrix form, letting $\bar{\phi}$ be the vector containing the samples $\phi(x_i)$ and $\underline{V}$ the Vandermonde matrix with entries $(\underline{V})_{ij} = T_{j-1}(x_{i-1})$, we have

$$\bar{\phi} = \underline{V}\bar{\phi}_{\text{Cheb}}. \tag{A.8}$$

This shows that solving the linear system (A.8) yields the vector of Chebyshev coefficients $\bar{\phi}_{\text{Cheb}}$, as desired from the FCT. Unfortunately, without exploiting the special structure of $\underline{V}$, solving the system is not fast.

To exploit this structure, first, we notice that the entries of the Vandermonde matrix are

$$(\underline{V})_{ij} = \cos\left((j-1)\cos^{-1}\left(\cos\left(\frac{\pi}{n}(i-1)\right)\right)\right) \tag{A.9}$$

$$= \cos\left(\frac{\pi}{n}(j-1)(i-1)\right). \tag{A.10}$$

Next, we compare these entries with the entries in a DFT matrix. A DFT matrix $\underline{F}_m$ of size $m \times m$ has entries

$$(\underline{F}_m)_{ij} = \omega_m^{(i-1)(j-1)} \tag{A.11}$$

with $\omega_m = e^{-2\pi\jmath/m}$. Choosing $m = 2n$, the DFT matrix has entries

$$(\underline{F}_{2n})_{ij} = \omega_{2n}^{(i-1)(j-1)} \tag{A.12}$$

$$= \left(e^{-2\pi\jmath/2n}\right)^{(i-1)(j-1)} \tag{A.13}$$

$$= e^{-\jmath(\pi/n)(i-1)(j-1)} \tag{A.14}$$

$$= \cos\left(\frac{\pi}{n}(j-1)(i-1)\right) - \jmath\sin\left(\frac{\pi}{n}(j-1)(i-1)\right). \tag{A.15}$$

Notice how the real part of $\underline{F}_{2n}$ for $i, j = 0, 1, 2, ..., n$ is equivalent to the Vandermonde matrix $\underline{V}$. We will exploit this fact to compute the FCT. That is, instead of solving the system (A.8), we will find a way to relate this problem to multiplication with the matrix

$\underline{F}_{2n}$, which can be performed in $\mathcal{O}(n \log n)$ operations using the FFT.

In doing so, we will need three fundamental properties of $\underline{F}_m$. The first property is that

$$\underline{F}_m = \underline{F}_m^T \tag{A.16}$$

since interchanging i and j in (A.11) gives $(\underline{F}_m)_{ij} = (\underline{F}_m)_{ji}$. The second property is that

$$\underline{F}_m^* \underline{F}_m = m\underline{I} \tag{A.17}$$

so that

$$\underline{F}_m^{-1} = \frac{1}{m}\underline{F}_m^* \tag{A.18}$$

where * denotes the conjugate transpose. This second property follows from multiplying the conjugate transpose of the $(p+1)$th column of $\underline{F}_m$ with the $(q+1)$th column. This gives

$$\begin{bmatrix} (\omega_m^{0\cdot p})^* & (\omega_m^{1\cdot p})^* & (\omega_m^{2\cdot p})^* & \cdots & (\omega_m^{(m-1)\cdot p})^* \end{bmatrix} \begin{bmatrix} \omega_m^{0\cdot q} \\ \omega_m^{1\cdot q} \\ \omega_m^{2\cdot q} \\ \vdots \\ \omega_m^{(m-1)\cdot q} \end{bmatrix} = \sum_{k=0}^{m-1} e^{\frac{2\pi\jmath}{m}k(p-q)}. \tag{A.19}$$

If $p = q$, then

$$\sum_{k=0}^{m-1} e^{\frac{2\pi\jmath}{m}k(p-q)} = \sum_{k=0}^{m-1} 1 \tag{A.20}$$

$$= m \tag{A.21}$$

whereas, if $p \neq q$, the geometric sum yields

$$\sum_{k=0}^{m-1} e^{\frac{2\pi\jmath}{m}k(p-q)} = \frac{1 - \left[e^{\frac{2\pi\jmath}{m}(p-q)}\right]^m}{1 - e^{\frac{2\pi\jmath}{m}(p-q)}} \tag{A.22}$$

$$= \frac{1 - e^{2\pi\jmath(p-q)}}{1 - e^{\frac{2\pi\jmath}{m}(p-q)}}. \tag{A.23}$$

Since $p - q$ is an integer, $e^{2\pi\jmath(p-q)} = 1$ and the numerator of (A.23) is zero. This directly proves (A.17) since p and q are arbitrary integers. Finally, the third property is that

$$\underline{T}_m \underline{F}_m = \underline{F}_m^*, \tag{A.24}$$

$$\underline{F}_m \underline{T}_m = \underline{F}_m^*, \tag{A.25}$$

where

$$\underline{T}_m = \begin{bmatrix} 1 & 0 \\ 0 & \underline{E}_{m-1} \end{bmatrix}, \qquad \underline{E}_{m-1} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{bmatrix}, \tag{A.26}$$

and $\underline{E}_{m-1}$ has $m-1$ rows and columns. By direct computation,

$$(\underline{T}_m \underline{F}_m)_{ij} = \begin{cases} \omega_m^{0\cdot(j-1)} & i = 1 \\ \omega_m^{(m-i+1)(j-1)} & i > 1. \end{cases}$$

Since

$$\omega_m^{(m-i+1)(j-1)} = \underbrace{\omega_m^{m(j-1)}}_{1} \underbrace{\omega_m^{-(i-1)(j-1)}}_{\left(\omega_m^{(i-1)(j-1)}\right)^*}, \tag{A.27}$$

this gives (A.24). Similar direct computation gives

$$(\underline{F}_m \underline{T}_m)_{ij} = \begin{cases} \omega_m^{(i-1)\cdot 0} & j = 1 \\ \omega_m^{(i-1)(n-j+1)} & j > 1 \end{cases}$$

from which (A.25) follows.

To use the DFT matrix to compute the FCT, we study the structure of $\underline{F}_{2n}$ in greater detail. First, from (A.15), the first row has entries

$$(\underline{F}_{2n})_{1j} = \cos\left(\frac{\pi}{n}(j-1)(1-1)\right) - \jmath \sin\left(\frac{\pi}{n}(j-1)(1-1)\right) \tag{A.28}$$

$$= \cos(0) - \jmath \sin(0) \tag{A.29}$$

$$= 1. \tag{A.30}$$

The $(n+1)$th row has entries

$$(\underline{F}_{2n})_{n+1,j} = \cos\left(\frac{\pi}{n}(j-1)(n+1-1)\right) - \jmath \sin\left(\frac{\pi}{n}(j-1)(n+1-1)\right) \tag{A.31}$$

$$= \cos(\pi(j-1)) - \jmath \sin(\pi(j-1)) \tag{A.32}$$

$$= (-1)^{j-1} \tag{A.33}$$

which oscillate between plus and minus one. Similar properties hold for the corresponding columns by the transpose property (A.16). Defining the cosine matrix $\underline{C}$ and sine matrix $\underline{S}$

with entries

$$(\underline{C})_{ij} = \cos\left(\frac{\pi}{n}(j-1)(1-1)\right), \qquad (\underline{S})_{ij} = \sin\left(\frac{\pi}{n}(j-1)(1-1)\right), \tag{A.34}$$

respectively, valid for $2 \le i, j \le n$, then $\underline{F}_{2n}$ has the block structure

$$\underline{F}_{2n} = \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} - \jmath\underline{S} & \bar{v} & \underline{X}^T \\ 1 & \bar{v}^T & (-1)^n & \bar{y}^T \\ \bar{e} & \underline{X} & \bar{y} & \underline{Z} \end{bmatrix} \tag{A.35}$$

where $\bar{e}$ is the vector of all ones, and $\bar{v}$ has alternating entries $(\bar{v})_i = (-1)^i$. Matrices $\underline{X}$, $\underline{Z}$ and vector $\bar{y}$ remain to be determined. To determine them, we use property (A.24) with

$$\underline{T}_{2n} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{E} \\ 0 & 0 & 1 & 0 \\ 0 & \underline{E} & 0 & 0 \end{bmatrix} \tag{A.36}$$

partitioned in accordance with (A.35) and $\underline{E}$ defined as in (A.26), chosen with appropriate size. Taking the product $\underline{T}_{2n}\underline{F}_{2n}$ yields

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{E} \\ 0 & 0 & 1 & 0 \\ 0 & \underline{E} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} - \jmath\underline{S} & \bar{v} & \underline{X}^T \\ 1 & \bar{v}^T & (-1)^n & \bar{y}^T \\ \bar{e} & \underline{X} & \bar{y} & \underline{Z} \end{bmatrix} = \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \underline{E}\bar{e} & \underline{E}\underline{X} & \underline{E}\bar{y} & \underline{E}\underline{Z} \\ 1 & \bar{v}^T & (-1)^n & \bar{y}^T \\ \underline{E}\bar{e} & \underline{E}(\underline{C} - \jmath\underline{S}) & \underline{E}\bar{v} & \underline{E}\underline{X}^T \end{bmatrix} \tag{A.37}$$

which we compare to

$$\underline{F}^*_{2n} = \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} + \jmath\underline{S} & \bar{v} & \underline{X}^* \\ 1 & \bar{v}^T & (-1)^n & \bar{y}^* \\ \bar{e} & (\underline{X}^T)^* & (\bar{y}^T)^* & \underline{Z}^* \end{bmatrix}. \tag{A.38}$$

Comparison of the (2,2) blocks gives

$$\underline{E}\underline{X} = \underline{C} + \jmath\underline{S} \tag{A.39}$$

$$\underline{X} = \underline{E}(\underline{C} + \jmath\underline{S}) \tag{A.40}$$

since $\underline{E}^2 = \underline{I}$. Similarly, comparison of the (2,3) blocks gives

$$\underline{E}\bar{y} = \bar{v} \tag{A.41}$$

$$\bar{y} = \underline{E}\bar{v} \tag{A.42}$$

and comparison of the (2,4) blocks, together with (A.40) gives

$$\underline{E}\underline{Z} = \underline{X}^* \tag{A.43}$$

$$\underline{Z} = \underline{E}[\underline{E}(\underline{C} + \jmath\underline{S})]^* \tag{A.44}$$

$$= \underline{E}(\underline{C} - \jmath\underline{S})\underline{E}. \tag{A.45}$$

Thus

$$\underline{F}_{2n} = \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} - \jmath\underline{S} & \bar{v} & (\underline{C} + \jmath\underline{S})\underline{E} \\ 1 & \bar{v}^T & (-1)^n & \bar{v}^T\underline{E} \\ \bar{e} & \underline{E}(\underline{C} + \jmath\underline{S}) & \underline{E}\bar{v} & \underline{E}(\underline{C} - \jmath\underline{S})\underline{E} \end{bmatrix}. \tag{A.46}$$

To relate this matrix to the Vandermonde matrix $\underline{V}$, we apply a transform to separate the $\underline{C}$ and $\underline{S}$ matrices. The transform is related to conjugate even vectors: vectors $\bar{y}$ satisfying the property that the complex conjugate of $\bar{y}$ is equal to $\underline{T}_m\bar{y}$. That is, $\bar{y}^* = \bar{y}^T\underline{T}_m$. This property is relevant in the context of the DFT because the DFT of a real vector is conjugate even. For example, if $\bar{y} = \underline{F}_m\bar{x}$ with $\bar{x} \in \mathbb{R}^m$, then

$$\bar{y}^* = \bar{x}^*\underline{F}_m^* \tag{A.47}$$

$$= \bar{x}^T\underline{F}_m\underline{T}_m \tag{A.48}$$

$$= \bar{x}^T\underline{F}_m^T\underline{T}_m \tag{A.49}$$

$$= \bar{y}^T\underline{T}_m \tag{A.50}$$

where the second line follows from (A.25) and the third line follows from (A.16). When $m = 2n$, then an arbitrary conjugate even vector

$$\bar{y} = \begin{bmatrix} a_1 + \jmath b_1 \\ \bar{a}_2 + \jmath\bar{b}_2 \\ a_3 + \jmath b_3 \\ \bar{a}_4 + \jmath\bar{b}_4 \end{bmatrix} \tag{A.51}$$

has real components a_1, $\bar{a}_2$, a_3, $\bar{a}_4$, b_1, $\bar{b}_2$, b_3, $\bar{b}_4$ constrained by $\bar{y}^* = \bar{y}^T \underline{T}_{2n}$. That is,

$$\begin{bmatrix} a_1 - \jmath b_1 \\ \bar{a}_2 - \jmath \bar{b}_2 \\ a_3 - \jmath b_3 \\ \bar{a}_4 - \jmath \bar{b}_4 \end{bmatrix}^T = \begin{bmatrix} a_1 + \jmath b_1 \\ \bar{a}_2 + \jmath \bar{b}_2 \\ a_3 + \jmath b_3 \\ \bar{a}_4 + \jmath \bar{b}_4 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{E} \\ 0 & 0 & 1 & 0 \\ 0 & \underline{E} & 0 & 0 \end{bmatrix} \tag{A.52}$$

$$= \begin{bmatrix} a_1 + \jmath b_1 \\ \underline{E}(\bar{a}_4 + \jmath \bar{b}_4) \\ a_3 + \jmath b_3 \\ \underline{E}(\bar{a}_2 + \jmath \bar{b}_2) \end{bmatrix}^T \tag{A.53}$$

so that, equating like entries gives

$$b_1 = 0, \tag{A.54}$$

$$\bar{a}_4 = \underline{E}\bar{a}_2, \tag{A.55}$$

$$\bar{b}_4 = -\underline{E}\bar{b}_2, \tag{A.56}$$

$$b_3 = 0, \tag{A.57}$$

with a_1, $\bar{a}_2$, a_3, and $\bar{b}_2$ free. This means that for (A.51) to be conjugate even, it must be of the form

$$\bar{y} = \begin{bmatrix} a_1 \\ \bar{a}_2 + \jmath \bar{b}_2 \\ a_3 \\ \underline{E}\bar{a}_2 - \jmath \underline{E}\bar{b}_2 \end{bmatrix}. \tag{A.58}$$

Rather than have $\bar{a}_2$ and $\bar{b}_2$ appear twice in $\bar{y}$, it is convenient to work with the vector

$$\bar{y}_{ce} = \begin{bmatrix} a_1 & \bar{a}_2^T & a_3 & \bar{b}_2^T \end{bmatrix}^T. \tag{A.59}$$

The two representations are connected via the linear transformation

$$\bar{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & \jmath \underline{I} \\ 0 & 0 & 1 & 0 \\ 0 & \underline{E} & 0 & -\jmath \underline{E} \end{bmatrix}}_{\underline{U}_{2n}} \bar{y}_{ce}. \tag{A.60}$$

The inverse transformation is

$$\underline{U}_{2n}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\underline{I} & 0 & \frac{1}{2}\underline{E} \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{\jmath}{2}\underline{I} & 0 & \frac{\jmath}{2}\underline{E} \end{bmatrix}. \tag{A.61}$$

Finally, if we have $\bar{y} = \underline{F}_{2n}\bar{x}$, then we can rewrite this expression in terms of $\bar{y}_{ce} = \underline{U}_{2n}^{-1}\bar{y}$ and $\bar{x}_{ce} = \underline{U}_{2n}^{-1}\bar{x}$ through the process

$$\underline{U}_{2n}^{-1}\bar{y} = \underline{U}_{2n}^{-1}\underline{F}_{2n}\bar{x} \tag{A.62}$$

$$\bar{y}_{ce} = \underline{U}_{2n}^{-1}\underline{F}_{2n}\underbrace{\underline{U}_{2n}\underline{U}_{2n}^{-1}}_{\underline{I}}\bar{x} \tag{A.63}$$

$$= \underline{U}_{2n}^{-1}\underline{F}_{2n}\underline{U}_{2n}\bar{x}_{ce}. \tag{A.64}$$

Our approach to computing the inverse of $\underline{V}$ lies in the observation that

$$\underline{U}_{2n}^{-1}\underline{F}_{2n}\underline{U}_{2n} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\underline{I} & 0 & \frac{1}{2}\underline{E} \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{\jmath}{2}\underline{I} & 0 & \frac{\jmath}{2}\underline{E} \end{bmatrix} \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} - \jmath\underline{S} & \bar{v} & (\underline{C} + \jmath\underline{S})\underline{E} \\ 1 & \bar{v}^T & (-1)^n & \bar{v}^T\underline{E} \\ \bar{e} & \underline{E}(\underline{C} + \jmath\underline{S}) & \underline{E}\bar{v} & \underline{E}(\underline{C} - \jmath\underline{S})\underline{E} \end{bmatrix} \underline{U}_{2n} \tag{A.65}$$

$$= \begin{bmatrix} 1 & \bar{e}^T & 1 & \bar{e}^T \\ \bar{e} & \underline{C} & \bar{v} & \underline{C}\underline{E} \\ 1 & \bar{v}^T & (-1)^n & \bar{v}^T\underline{E} \\ 0 & -\underline{S} & 0 & \underline{S}\underline{E} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & \jmath\underline{I} \\ 0 & 0 & 1 & 0 \\ 0 & \underline{E} & 0 & -\jmath\underline{E} \end{bmatrix} \tag{A.66}$$

$$= \begin{bmatrix} 1 & 2\bar{e}^T & 1 & 0 \\ \bar{e} & 2\underline{C} & \bar{v} & 0 \\ 1 & 2\bar{v}^T & (-1)^n & 0 \\ 0 & 0 & 0 & -2\jmath\underline{S} \end{bmatrix}. \tag{A.67}$$

Notice how the upper three by three block is closely related to $\underline{V}$ for the FCT. In fact,

$$\underline{V} = \underbrace{\begin{bmatrix} 1 & 2\bar{e}^T & 1 \\ \bar{e} & 2\underline{C} & \bar{v} \\ 1 & 2\bar{v}^T & (-1)^n \end{bmatrix}}_{\tilde{\underline{V}}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2}\underline{I} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\underline{D}} \tag{A.68}$$

so that $\underline{V}^{-1} = \underline{D}^{-1}\tilde{\underline{V}}^{-1}$. Inverting $\underline{D}$ is trivial (it is diagonal), and we perform the inverse of $\tilde{\underline{V}}$ by selecting only those rows and columns relevant to the upper three by three block of the inverse of $\underline{U}_{2n}^{-1}\underline{F}_{2n}\underline{U}_{2n}$. That is,

$$\tilde{\underline{V}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\underline{U}_{2n}^{-1}\underline{F}_{2n}\underline{U}_{2n}\right)^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \underline{I} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.69}$$

Since $\left(\underline{U}_{2n}^{-1}\underline{F}_{2n}\underline{U}_{2n}\right)^{-1} = \underline{U}_{2n}^{-1}\underline{F}_{2n}^{-1}\underline{U}_{2n}$, we obtain

$$\tilde{\underline{V}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\underline{I} & 0 & \frac{1}{2}\underline{E} \\ 0 & 0 & 1 & 0 \end{bmatrix} \underline{F}_{2n}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \underline{I} & 0 \\ 0 & 0 & 1 \\ 0 & \underline{E} & 0 \end{bmatrix}. \tag{A.70}$$

Alternatively, one can use

$$\tilde{\underline{V}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underline{F}_{2n}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \underline{I} & 0 \\ 0 & 0 & 1 \\ 0 & \underline{E} & 0 \end{bmatrix} \tag{A.71}$$

which is equivalent due to certain symmetries.

To summarize, solving the system (A.8) is equivalent to applying the transformations

$$\bar{\phi}_{\text{Cheb}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2\underline{I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underline{F}_{2n}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \underline{I} & 0 \\ 0 & 0 & 1 \\ 0 & \underline{E} & 0 \end{bmatrix} \bar{\phi} \tag{A.72}$$

where we partition

$$\bar{\phi} = \begin{bmatrix} \phi_0 \\ \tilde{\phi} \\ \phi_n \end{bmatrix}. \tag{A.73}$$

In practice, we do not form these intermediate matrices. Instead, we use the vector

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \underline{I} & 0 \\ 0 & 0 & 1 \\ 0 & \underline{E} & 0 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \tilde{\phi} \\ \phi_n \end{bmatrix} = \begin{bmatrix} \phi_0 \\ \tilde{\phi} \\ \phi_n \\ \underline{E}\tilde{\phi} \end{bmatrix} \tag{A.74}$$

as input to an inverse FFT routine (noting that $\underline{E}\tilde{\phi}$ is a reversal of order of the entries in $\tilde{\phi}$), then postprocess the output by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2\underline{I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \underline{I} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2\underline{I} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.75}$$

which is equivalent to discarding the last $n-1$ entries of the output vector, and scaling the 2nd to nth entries by a factor of two. Since the inverse FFT can be computed in $\mathcal{O}(2n\log(2n)) = \mathcal{O}(n\log(n))$ operations, this is a fast method for converting samples at Chebyshev nodes into Chebyshev coefficients. There are more efficient methods which exploit additional symmetries of the vector (A.74) which are called Fast Cosine Transforms, but we direct the interested reader to [176] for details.

A.2 Legendre to Chebyshev Connection Coefficients

The last section explained how to efficiently convert from samples of a function $\bar{\phi}$ at Chebyshev nodes to Chebyshev coefficients $\bar{\phi}_{\text{Cheb}}$. In this section, we describe how Chebyshev coefficients are related to Legendre coefficients $\bar{\phi}_{\text{Leg}}$. For now, we will talk only about orthogonal Legendre polynomials, not orthonormal ones. Switching between the two is trivial and will be done only at the end of our development. To begin, we define vectors of Chebyshev polynomials and Legendre polynomials

$$\bar{T}(x) = \begin{bmatrix} T_0(x) \\ T_1(x) \\ \vdots \\ T_n(x) \end{bmatrix}, \qquad \bar{P}(x) = \begin{bmatrix} P_0(x) \\ P_1(x) \\ \vdots \\ P_n(x) \end{bmatrix}, \tag{A.76}$$

respectively. Since we have computed a polynomial

$$p(x) = \bar{\phi}_{\text{Cheb}}^T \bar{T}(x) \tag{A.77}$$

in the previous section, we now investigate how to rewrite this polynomial as

$$p(x) = \bar{\phi}_{\text{Leg}}^T \bar{P}(x). \tag{A.78}$$

Equating the two expressions, multiplying from the right by $\bar{P}(x)^T$, and integrating yields

$$\bar{\phi}_{\text{Cheb}}^T \bar{T}(x) = \bar{\phi}_{\text{Leg}}^T \bar{P}(x) \tag{A.79}$$

$$\bar{\phi}_{\text{Cheb}}^T \underbrace{\int_{-1}^{1} \bar{T}(x)\bar{P}(x)^T dx}_{\underline{\tilde{L}}} = \bar{\phi}_{\text{Leg}}^T \underbrace{\int_{-1}^{1} \bar{P}(x)\bar{P}(x)^T dx}_{\underline{D}} \tag{A.80}$$

where $\underline{\tilde{L}}$ is a lower triangular matrix, and $\underline{D}$ is a diagonal matrix. Then taking the transpose and isolating $\bar{\phi}_{\text{Leg}}$ yields

$$(\bar{\phi}_{\text{Cheb}}^T \underline{\tilde{L}})^T = (\bar{\phi}_{\text{Leg}}^T \underline{D})^T \tag{A.81}$$

$$\underline{\tilde{L}}^T \bar{\phi}_{\text{Cheb}} = \underline{D} \bar{\phi}_{\text{Leg}} \tag{A.82}$$

$$\underbrace{\underline{D}^{-1} \underline{\tilde{L}}^T}_{\underline{L}^T} \bar{\phi}_{\text{Cheb}} = \bar{\phi}_{\text{Leg}}. \tag{A.83}$$

The key to computing $\underline{L}^T \bar{\phi}_{\text{Cheb}}$ is thus understanding the structure of $\underline{L}$. In special functions literature [178], the entries in $\underline{L}$ are called the connection coefficients between the Chebyshev and Legendre polynomials. This is because

$$\bar{T}(x) = \underline{L} \bar{P}(x), \tag{A.84}$$

which describes precisely how to express each Chebyshev polynomial in terms of a linear combination of Legendre polynomials. To confirm that this $\underline{L}$ is the same as the one connecting the coefficients, multiply (A.84) by $\bar{P}(x)^T$ and integrate, as in (A.80).

Both Chebyshev and Legendre polynomials can be viewed as special cases of a more general family of polynomials called the ultraspherical (or Gegenbauer) polynomials $C_i^\lambda(x)$. Here, λ is a parameter which differentiates between families of polynomials. In particular, when $\lambda = 1/2$, the ultraspherical polynomials coincide with the Legendre polynomials. That is, $C_i^{1/2}(x) = P_i(x)$. Similarly, (see [179], for example)

$$\lim_{\lambda \to 0} \frac{1}{\lambda} C_i^\lambda(x) = \frac{2}{i} T_i(x), \qquad i \geq 1. \tag{A.85}$$

For this reason, in this section, we derive the connection coefficients between ultraspherical polynomials of two different parameters λ and μ, then specialize to the Chebyshev and

Legendre case by taking the limit as λ approaches zero and setting $\mu = 1/2$.

The starting point is the generating function [178] for ultraspherical polynomials

$$(1 - 2xr + r^2)^{-\lambda} = \sum_{i=0}^{\infty} C_i^{\lambda}(x) r^i. \tag{A.86}$$

Letting $x = \cos\theta = (e^{\jmath\theta} + e^{-\jmath\theta})/2$, we note that

$$1 - 2xr + r^2 = 1 - (e^{\jmath\theta} + e^{-\jmath\theta})r + r^2 \tag{A.87}$$

$$= (1 - re^{\jmath\theta})(1 - re^{-\jmath\theta}). \tag{A.88}$$

Then

$$(1 - 2\cos\theta r + r^2)^{-\lambda} = (1 - re^{\jmath\theta})^{-\lambda}(1 - re^{-\jmath\theta})^{-\lambda} \tag{A.89}$$

and we can apply the binomial theorem to each term separately to obtain

$$(1 - re^{\jmath\theta})^{-\lambda} = \sum_{k=0}^{\infty} \binom{-\lambda}{k} (-re^{\jmath\theta})^k \tag{A.90}$$

$$= \sum_{k=0}^{\infty} \frac{(-\lambda)_k}{k!} (-1)^k r^k e^{\jmath k\theta} \tag{A.91}$$

where $(-\lambda)_k = (-\lambda)(-\lambda - 1)\cdots(-\lambda - k + 1)$ is the Pochhammer symbol. Similarly,

$$(1 - re^{-\jmath\theta})^{-\lambda} = \sum_{l=0}^{\infty} \binom{-\lambda}{l} (-re^{-\jmath\theta})^l \tag{A.92}$$

$$= \sum_{l=0}^{\infty} \frac{(-\lambda)_l}{l!} (-1)^l r^l e^{-\jmath l\theta}. \tag{A.93}$$

Multiplying these two series, and collecting terms containing r^i, we obtain

$$C_i^{\lambda}(\cos\theta) r^i = \sum_{j=0}^{i} \left(\frac{(-\lambda)_j}{j!} (-1)^j r^j e^{\jmath j\theta} \right) \left(\frac{(-\lambda)_{i-j}}{(i-j)!} (-1)^{i-j} r^{i-j} e^{-\jmath(i-j)\theta} \right) \tag{A.94}$$

$$= \sum_{j=0}^{i} \left(\frac{(-\lambda)_j (-1)^j}{j!} \frac{(-\lambda)_{i-j} (-1)^{i-j}}{(i-j)!} e^{\jmath(2j-i)\theta} \right) r^i. \tag{A.95}$$

If $\Gamma(z)$ denotes the Gamma function [178] satisfying $\Gamma(z+1) = z\Gamma(z)$, then

$$(-\lambda)_j (-1)^j = \frac{\Gamma(j + \lambda)}{\Gamma(\lambda)}, \qquad (-\lambda)_{i-j} (-1)^{i-j} = \frac{\Gamma(i - j + \lambda)}{\Gamma(\lambda)}, \tag{A.96}$$

$j! = \Gamma(j+1)$, $(i-j)! = \Gamma(i-j+1)$, and

$$C_i^\lambda(\cos\theta) = \frac{1}{[\Gamma(\lambda)]^2}\sum_{j=0}^{i}\frac{\Gamma(j+\lambda)}{\Gamma(j+1)}\frac{\Gamma(i-j+\lambda)}{\Gamma(i-j+1)}e^{\jmath(2j-i)\theta}. \tag{A.97}$$

Making one additional adjustment, we recognize that if i is odd, there are an even number of terms to sum, and we can group them together in pairs. That is, for example, if $j = 0$ and $j = i$, we have the two terms

$$\underbrace{\frac{\Gamma(\lambda)}{\Gamma(1)}\frac{\Gamma(i+\lambda)}{\Gamma(i+1)}e^{\jmath(-i)\theta}}_{j=0\text{ term}} + \underbrace{\frac{\Gamma(i+\lambda)}{\Gamma(i+1)}\frac{\Gamma(i-i+\lambda)}{\Gamma(i-i+1)}e^{\jmath(2i-i)\theta}}_{j=i\text{ term}} \tag{A.98}$$

appearing in the sum. They simplify to

$$\frac{\Gamma(\lambda)}{\Gamma(1)}\frac{\Gamma(i+\lambda)}{\Gamma(i+1)}e^{-\jmath i\theta} + \frac{\Gamma(i+\lambda)}{\Gamma(i+1)}\frac{\Gamma(\lambda)}{\Gamma(1)}e^{\jmath i\theta} = \frac{\Gamma(\lambda)}{\Gamma(1)}\frac{\Gamma(i+\lambda)}{\Gamma(i+1)}(e^{-\jmath i\theta} + e^{\jmath i\theta}) \tag{A.99}$$

$$= \frac{\Gamma(\lambda)}{\Gamma(1)}\frac{\Gamma(i+\lambda)}{\Gamma(i+1)}2\cos(i\theta). \tag{A.100}$$

Similar simplification applies to all pairs $j = k$ and $j = i - k$, giving

$$\underbrace{\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}e^{\jmath(2k-i)\theta}}_{j=k\text{ term}} + \underbrace{\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}\frac{\Gamma(i-(i-k)+\lambda)}{\Gamma(i-(i-k)+1)}e^{\jmath(2(i-k)-i)\theta}}_{j=i-k\text{ term}} \tag{A.101}$$

which becomes

$$\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}2\cos((i-2k)\theta). \tag{A.102}$$

When i is even, there is an odd number of terms to sum. Similar pairings occur, leaving the $j = i/2$ term alone. However, this term has the form

$$\frac{\Gamma(i/2+\lambda)}{\Gamma(i/2+1)}\frac{\Gamma(i-i/2+\lambda)}{\Gamma(i-i/2+1)}e^{\jmath(0)\theta} \tag{A.103}$$

which means that we can write it as

$$\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}\cos((i-2k)\theta) \tag{A.104}$$

with $k = i/2$ (notice the factor of 2 missing). This gives

$$C_i^\lambda(\cos\theta) = \frac{1}{[\Gamma(\lambda)]^2}\sum_{k=0}^{i/2}\nu_k\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}\cos((i-2k)\theta) \tag{A.105}$$

where

$$\nu_k = \begin{cases} 1 & k = i/2,\ i \text{ even}, \\ 2 & \text{otherwise}. \end{cases} \tag{A.106}$$

In the next step of our process, we find a way to replace the cosine functions in (A.105) with ultraspherical polynomials C_j^μ where μ need not be the same as λ. The key is to take the derivative of (A.105) with respect to θ. Since (see [180], for example)

$$\frac{d}{dx}C_i^\lambda(x) = 2\lambda C_{i-1}^{\lambda+1}(x), \tag{A.107}$$

we have

$$\frac{d}{d\theta}C_i^\lambda(\cos\theta) = \frac{1}{[\Gamma(\lambda)]^2}\sum_{k=0}^{i/2}\nu_k\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}\frac{d}{d\theta}\cos((i-2k)\theta) \tag{A.108}$$

$$2\lambda C_{i-1}^{\lambda+1}(\cos\theta)(-\sin\theta) = \frac{1}{[\Gamma(\lambda)]^2}\sum_{k=0}^{\lfloor i/2\rfloor}2\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}(-\sin((i-2k)\theta)(i-2k)) \tag{A.109}$$

$$C_{i-1}^{\lambda+1}(\cos\theta) = \frac{1}{\lambda[\Gamma(\lambda)]^2}\sum_{k=0}^{\lfloor i/2\rfloor}(i-2k)\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}\frac{\sin((i-2k)\theta)}{\sin\theta}. \tag{A.110}$$

Since the $k = i/2$ term is constant, its derivative is zero. This causes the sum to run to the floor of $i/2$ denoted as $\lfloor i/2\rfloor$ and allows us to replace all remaining ν_k with 2. Also, note that

$$U_{i-2k-1}(\cos\theta) = \frac{\sin((i-2k)\theta)}{\sin\theta} \tag{A.111}$$

is the definition of the order $i-2k-1$ Chebyshev polynomial of the second kind and that the Chebyshev polynomial of the second kind is an ultraspherical polynomial with $\lambda = 1$. This means that

$$C_{i-1}^{\lambda+1}(x) = \frac{1}{\lambda[\Gamma(\lambda)]^2}\sum_{k=0}^{\lfloor i/2\rfloor}(2k-i)\frac{\Gamma(k+\lambda)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+1)}C_{i-2k-1}^1(x) \tag{A.112}$$

where we have used $x = \cos\theta$. We can set $\lambda+1 \to \lambda$ and $i-1 \to i$ to obtain

$$C_i^\lambda(x) = \frac{1}{(\lambda-1)[\Gamma(\lambda-1)]^2}\sum_{k=0}^{\lfloor i/2\rfloor}(i+1-2k)\frac{\Gamma(k+\lambda-1)}{\Gamma(k+1)}\frac{\Gamma(i-k+\lambda)}{\Gamma(i+1-k+1)}C_{i-2k}^1(x). \tag{A.113}$$

Note that the upper index of the sum does not increase because $i-2k-1$ is negative when

$k \geq i/2$, meaning that terms exceeding this index drop out of the sum. We now repeat a two step process of differentiation with respect to x, and raising and lowering of indices. This yields the sequence

$$C_i^\lambda(x) = \frac{1}{(\lambda-1)(\lambda-2)[\Gamma(\lambda-2)]^2} \cdot \sum_{k=0}^{\lfloor i/2 \rfloor} (i+2-2k) \frac{\Gamma(k+\lambda-2)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i+2-k+1)} (1) C_{i-2k}^2(x), \quad \text{(A.114)}$$

$$C_i^\lambda(x) = \frac{1}{(\lambda-1)(\lambda-2)(\lambda-3)[\Gamma(\lambda-3)]^2} \cdot \sum_{k=0}^{\lfloor i/2 \rfloor} (i+3-2k) \frac{\Gamma(k+\lambda-3)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i+3-k+1)} (1)(2) C_{i-2k}^3(x), \quad \text{(A.115)}$$

$$C_i^\lambda(x) = \frac{1}{(\lambda-1)(\lambda-2)(\lambda-3)(\lambda-4)[\Gamma(\lambda-4)]^2} \cdot \sum_{k=0}^{\lfloor i/2 \rfloor} (i+4-2k) \frac{\Gamma(k+\lambda-4)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i+4-k+1)} (1)(2)(3) C_{i-2k}^4(x), \quad \text{(A.116)}$$

and so on which gives rise to the pattern summarized by

$$C_i^\lambda(x) = \frac{1}{\Gamma(\lambda)} \frac{\Gamma(\mu+1)}{\mu\Gamma(\lambda-\mu)} \sum_{k=0}^{\lfloor i/2 \rfloor} (i+\mu-2k) \frac{\Gamma(k+\lambda-\mu)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i+\mu-k+1)} C_{i-2k}^\mu(x) \quad \text{(A.117)}$$

for μ an integer. The validity of this expression can be proved via induction. Since the right hand side of (A.117) is a rational function of μ which holds at infinitely many integers and the left hand side is independent of μ, we conclude that the equation must hold for all real values of μ, including $\mu = 1/2$.

At this point, we have determined the connection coefficients between ultraspherical polynomials $C_i^\lambda(x)$ and $C_i^\mu(x)$ (they are the coefficients in the sum (A.117)). Equipped with (A.117), we are ready to specialize to the case connecting Chebyshev polynomials of the first kind to Legendre polynomials. To do so, we let $\mu = 1/2$ to obtain Legendre polynomials on the right hand side. This gives

$$C_i^\lambda(x) = \frac{2}{\Gamma(\lambda)} \frac{\Gamma(3/2)}{\Gamma(\lambda-1/2)} \sum_{k=0}^{\lfloor i/2 \rfloor} (i+1/2-2k) \frac{\Gamma(k+\lambda-1/2)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+3/2)} P_{i-2k}(x). \quad \text{(A.118)}$$

We then divide by λ and take the limit as λ approaches zero to recover Chebyshev polynomials on the left hand side. Choosing $i \geq 1$, this gives

$$\lim_{\lambda \to 0} \frac{1}{\lambda} C_i^{\lambda}(x) = \lim_{\lambda \to 0} \frac{2}{\Gamma(\lambda+1)} \frac{\Gamma(3/2)}{\Gamma(\lambda-1/2)} \cdot \sum_{k=0}^{\lfloor i/2 \rfloor} (i+1/2-2k) \frac{\Gamma(k+\lambda-1/2)}{\Gamma(k+1)} \frac{\Gamma(i-k+\lambda)}{\Gamma(i-k+3/2)} P_{i-2k}(x) \quad \text{(A.119)}$$

which becomes

$$\frac{2}{i} T_i(x) = \frac{2}{\Gamma(1)} \frac{\Gamma(3/2)}{\Gamma(-1/2)} \sum_{k=0}^{\lfloor i/2 \rfloor} (i+1/2-2k) \frac{\Gamma(k-1/2)}{\Gamma(k+1)} \frac{\Gamma(i-k)}{\Gamma(i-k+3/2)} P_{i-2k}(x) \quad \text{(A.120)}$$

$$T_i(x) = \frac{i}{\Gamma(1)} \frac{\Gamma(3/2)}{\Gamma(-1/2)} \sum_{k=0}^{\lfloor i/2 \rfloor} (i+1/2-2k) \frac{\Gamma(k-1/2)}{\Gamma(k+1)} \frac{\Gamma(i-k)}{\Gamma(i-k+3/2)} P_{i-2k}(x). \quad \text{(A.121)}$$

We now use properties of the Gamma function

$$\Gamma(1) = 1, \qquad \Gamma(-1/2) = -2\sqrt{\pi}, \qquad \Gamma(3/2) = \sqrt{\pi}/2, \quad \text{(A.122)}$$

to see that

$$T_i(x) = -\frac{i}{4} \sum_{k=0}^{\lfloor i/2 \rfloor} (i+1/2-2k) \frac{\Gamma(k-1/2)}{\Gamma(k+1)} \frac{\Gamma(i-k)}{\Gamma(i-k+3/2)} P_{i-2k}(x). \quad \text{(A.123)}$$

Changing index so that $i-2k = j$ (and consequently replacing k by $(i-j)/2$ for $i-j$ even), we obtain

$$T_i(x) = -\frac{i}{4} \sum_{\substack{j \leq i \\ i-j \text{ even}}} (j+1/2) \frac{\Gamma\left(\frac{i-j}{2} - \frac{1}{2}\right)}{\Gamma\left(\frac{i-j}{2}+1\right)} \frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} P_j(x) \quad \text{(A.124)}$$

for $i \geq 1$. The case $i = 0$ is trivial, because both $T_0(x) = P_0(x) = 1$. This completes the characterization of connection coefficients between Chebyshev polynomials $T_i(x)$ and Legendre polynomials $P_i(x)$.

Combining (A.124) with the $i = 0$ case allows us to explicitly describe the entries of $\underline{L}$ such that $\bar{T}(x) = \underline{L}\bar{P}(x)$. The entries are

$$(\underline{L})_{i+1,j+1} = \begin{cases} 1 & i = j = 0 \\ -\frac{1}{4} i (j+1/2) \frac{\Gamma\left(\frac{i-j}{2} - \frac{1}{2}\right)}{\Gamma\left(\frac{i-j}{2}+1\right)} \frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} & j \leq i,\ i-j \text{ even} \\ 0 & \text{otherwise} \end{cases} \quad \text{(A.125)}$$

with $0 \le i, j \le n$. The key insight of [51] is to observe that this $\underline{L}$ can be factorized as

$$\underline{L} = \underline{D}_1(\underline{H} \circ \underline{T})\underline{D}_2 \tag{A.126}$$

where

$$\underline{D}_1 = -\frac{1}{4}\begin{bmatrix} \frac{4}{\sqrt{\pi}} & & & & \\ & 1 & & & \\ & & 2 & & \\ & & & \ddots & \\ & & & & n \end{bmatrix}, \qquad \underline{D}_2 = \frac{1}{2}\begin{bmatrix} 1 & & & & \\ & 3 & & & \\ & & 5 & & \\ & & & \ddots & \\ & & & & 2n+1 \end{bmatrix}, \tag{A.127}$$

and

$$(\underline{T})_{i+1,j+1} = \begin{cases} \dfrac{\Gamma\left(\frac{i-j}{2} - \frac{1}{2}\right)}{\Gamma\left(\frac{i-j}{2} + 1\right)} & j \le i,\ i-j \text{ even} \\ 0 & \text{otherwise,} \end{cases} \qquad (\underline{H})_{i+1,j+1} = \begin{cases} 1 & i = j = 0 \\ \dfrac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2} + \frac{3}{2}\right)} & \text{otherwise.} \end{cases} \tag{A.128}$$

Note that, from (A.125), $\underline{D}_1$ arises from the term $-\frac{1}{4}i$, $\underline{D}_2$ arises from the term $j + \frac{1}{2} = (2j+1)/2$, and the Hankel matrix $\underline{H}$ and Toeplitz matrix $\underline{T}$ cover the remaining Gamma functions. Hankel matrices have constant antidiagonals (arising from terms containing the sum of indices $i + j$) whereas Toeplitz matrices have constant diagonals (arising from terms with the difference $i - j$). The first entry in $\underline{D}_1$ is chosen after specifying $\underline{D}_2$, $\underline{H}$, and $\underline{T}$ so that $(\underline{L})_{11} = 1$, as required by (A.125). We remark that $\underline{T}$ is lower triangular and that $\underline{H}$ is symmetric.

A.3 Factorization of the Hankel Part

In order to perform the multiplication $\underline{L}^T \bar{\phi}_{\text{Cheb}}$ quickly for large n, we need to exploit the structure of matrix $\underline{H}$. To do so, we first show that a large portion of this matrix is positive semidefinite. This can be done using the Beta function

$$\mathrm{B}(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt \tag{A.129}$$

defined for $\Re\{x\}, \Re\{y\} > 0$. The Beta function is related to the Gamma function through the relation

$$\mathrm{B}(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \tag{A.130}$$

which we use with $x = (i+j)/2$ and $y = 3/2$ to obtain

$$\mathrm{B}\left(\frac{i+j}{2}, \frac{3}{2}\right) = \frac{\Gamma\left(\frac{i+j}{2}\right)\Gamma\left(\frac{3}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)}. \tag{A.131}$$

We see that an alternative expression for the entries in the Hankel matrix $\underline{H}$ is given by

$$\frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} = \frac{\mathrm{B}\left(\frac{i+j}{2}, \frac{3}{2}\right)}{\Gamma\left(\frac{3}{2}\right)}. \tag{A.132}$$

Because of the restriction on the real part of $(i+j)/2$, we will leave out the case $i = j = 0$. Thus we treat only cases satisfying $1 \leq i, j \leq n$ (in theory we could allow one of i or j to be zero, but this will not lead to a square submatrix of $\underline{H}$). Using (A.129), we have

$$\frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} = \frac{1}{\Gamma\left(\frac{3}{2}\right)}\int_0^1 t^{(i+j-2)/2}(1-t)^{1/2}dt. \tag{A.133}$$

Making the substitution $t = x^2$ with $dt = 2x\,dx$, we obtain

$$\frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} = \frac{1}{\Gamma\left(\frac{3}{2}\right)}\int_0^1 x^{i+j-2}(1-x^2)^{1/2}2x\,dx \tag{A.134}$$

$$= \frac{4}{\sqrt{\pi}}\int_0^1 x^{i+j-1}(1-x^2)^{1/2}dx \tag{A.135}$$

where we have used (A.122). Partitioning $\underline{H}$ as

$$\underline{H} = \begin{bmatrix} h_{11} & \bar{h}_{21}^T \\ \bar{h}_{21} & \underline{H}_{22} \end{bmatrix} \tag{A.136}$$

by separating the first row and column from the rest of $\underline{H}$, we have just shown that the entries of $\underline{H}_{22}$ can be written as

$$(\underline{H}_{22})_{ij} = \frac{4}{\sqrt{\pi}}\int_0^1 x^{i+j-1}(1-x^2)^{1/2}dx. \tag{A.137}$$

We use this expression to show that $\underline{H}_{22}$ is positive semidefinite. Choosing an arbitrary vector $\bar{y}$, we have

$$\bar{y}^T\underline{H}_{22}\bar{y} = \sum_{i,j=1}^{n} y_i y_j (\underline{H}_{22})_{ij} \tag{A.138}$$

which gives

$$\bar{y}^T \underline{H}_{22} \bar{y} = \sum_{i,j=1}^{n} y_i y_j \frac{4}{\sqrt{\pi}} \int_0^1 x^{i+j-1} (1-x^2)^{1/2} dx. \tag{A.139}$$

Bringing the sum into the integral and writing $x^{i+j-1} = x^i x^j x^{-1}$, we have

$$\bar{y}^T \underline{H}_{22} \bar{y} = \frac{4}{\sqrt{\pi}} \int_0^1 \left[\sum_{i,j=1}^{n} y_i y_j x^i x^j \right] x^{-1} (1-x^2)^{1/2} dx \tag{A.140}$$

$$= \frac{4}{\sqrt{\pi}} \int_0^1 \left[\sum_{i=1}^{n} y_i x^i \right]^2 x^{-1} (1-x^2)^{1/2} dx. \tag{A.141}$$

Finally, $\left[\sum_{i=1}^{n} y_i x^i\right]^2 x^{-1}$ is a polynomial in x and both this polynomial and $(1-x^2)^{1/2}$ are positive over the interval $(0,1)$, meaning that the integral itself must be greater than or equal to zero. From this, we conclude that

$$\bar{y}^T \underline{H}_{22} \bar{y} \geq 0 \tag{A.142}$$

which is precisely what is meant by matrix $\underline{H}_{22}$ being positive semidefinite.

The fact that we have partitioned $\underline{H}$ does not prevent fast multiplication of $\underline{L}^T \bar{\phi}_{\text{Cheb}}$. If we partition $\underline{T}$ accordingly, we have

$$\underline{T} = \begin{bmatrix} t_{11} & 0 \\ \bar{t}_{21} & \underline{T}_{22} \end{bmatrix} \tag{A.143}$$

with $\underline{T}_{22}$ lower triangular and also Toeplitz. Then

$$\underline{L}^T \bar{\phi}_{\text{Cheb}} = \left[\underline{D}_1 (\underline{H} \circ \underline{T}) \underline{D}_2\right]^T \bar{\phi}_{\text{Cheb}} \tag{A.144}$$

$$= \underline{D}_2 (\underline{H} \circ \underline{T})^T \underline{D}_1 \bar{\phi}_{\text{Cheb}} \tag{A.145}$$

$$= \underline{D}_2 (\underline{H} \circ \underline{T}^T) \underline{D}_1 \bar{\phi}_{\text{Cheb}} \tag{A.146}$$

since $\underline{H}$ is symmetric. Applying the diagonal matrices is simple, so we need only consider the matrix multiplication of $\underline{H} \circ \underline{T}^T$ with a vector. Since

$$\underline{H} \circ \underline{T}^T = \begin{bmatrix} h_{11} & \bar{h}_{21}^T \\ \bar{h}_{21} & \underline{H}_{22} \end{bmatrix} \circ \begin{bmatrix} t_{11} & \bar{t}_{21}^T \\ 0 & \underline{T}_{22}^T \end{bmatrix} \tag{A.147}$$

$$= \begin{bmatrix} h_{11} t_{11} & \bar{h}_{21}^T \circ \bar{t}_{21}^T \\ 0 & \underline{H}_{22} \circ \underline{T}_{22}^T \end{bmatrix}, \tag{A.148}$$

its multiplication with a vector yields

$$\begin{bmatrix} h_{11}t_{11} & \bar{h}_{21}^T \circ \bar{t}_{21}^T \\ 0 & \underline{H}_{22} \circ \underline{T}_{22}^T \end{bmatrix} \begin{bmatrix} \phi_1 \\ \bar{\phi}_2 \end{bmatrix} = \begin{bmatrix} h_{11}t_{11}\phi_1 + (\bar{h}_{21} \circ \bar{t}_{21})^T \bar{\phi}_2 \\ (\underline{H}_{22} \circ \underline{T}_{22}^T)\bar{\phi}_2 \end{bmatrix}. \tag{A.149}$$

The first entry on the right hand side can be computed in $\mathcal{O}(n)$ operations since it involves an inner product and a small number of scalar operations. We now focus on how to compute the second vector component efficiently.

To do so, as stated in the introduction of this chapter, we are going to exploit the fact that $\underline{H}_{22}$ is symmetric, positive semidefinite to compute its factorization into a sum of rank 1 terms. We use a pivoted LDLT algorithm to compute the rank-revealing factorization. The algorithm is based on Sections 4.2.7 and 4.2.8 of [25]. First, we set $\underline{H}_{22}^{(0)} = \underline{H}_{22}$ and find the entry $(\underline{H}_{22})_{ij}$ with largest magnitude. Then, we use this entry as a pivot to compute

$$\underline{H}_{22}^{(1)} = \underline{H}_{22}^{(0)} - \frac{1}{(\underline{H}_{22})_{ij}} \underline{H}_{22}(:, j)\underline{H}_{22}(i, :) \tag{A.150}$$

where $\underline{H}_{22}(:, j)$ denotes the jth column of $\underline{H}_{22}$ and $\underline{H}_{22}(i, :)$ denotes the ith row. This process is then repeated to produce a sequence of matrices $\underline{H}_{22}^{(0)}$, $\underline{H}_{22}^{(1)}$, $\underline{H}_{22}^{(2)}$,..., until the pivot at the rth step is below a user specified tolerance (we choose $10^{-14} \log n$).

In practice, since $\underline{H}_{22}$ is symmetric, positive semidefinite, the pivot always comes from the diagonal. This is because for any symmetric, positive semidefinite matrix $\underline{A}$ satisfying $\bar{x}^T \underline{A} \bar{x} \geq 0$,

$$|a_{ij}| \leq \frac{a_{ii} + a_{jj}}{2}. \tag{A.151}$$

To see why, choose $\bar{x} = \bar{e}_i + \bar{e}_j$ where $\bar{e}_i$ is the ith unit vector and $\bar{e}_j$ is the jth unit vector. Then

$$(\bar{e}_i + \bar{e}_j)^T \underline{A}(\bar{e}_i + \bar{e}_j) \geq 0 \tag{A.152}$$

$$a_{ii} + 2a_{ij} + a_{jj} \geq 0. \tag{A.153}$$

Similarly, choosing $\bar{x} = \bar{e}_i - \bar{e}_j$ yields

$$(\bar{e}_i - \bar{e}_j)^T \underline{A}(\bar{e}_i - \bar{e}_j) \geq 0 \tag{A.154}$$

$$a_{ii} - 2a_{ij} + a_{jj} \geq 0. \tag{A.155}$$

Inequality (A.153) yields

$$a_{ij} \geq -\frac{a_{ii} + a_{jj}}{2} \tag{A.156}$$

whereas (A.155) yields

$$a_{ij} \leq \frac{a_{ii} + a_{jj}}{2}. \tag{A.157}$$

Taken together, these two inequalities yield (A.151). Now, using (A.151),

$$\max_{i,j} |a_{ij}| \leq \max_{i,j} \frac{a_{ii} + a_{jj}}{2}. \tag{A.158}$$

By choosing $\bar{x} = \bar{e}_i$ or $\bar{x} = \bar{e}_j$ in $\bar{x}^T \underline{A} \bar{x} \geq 0$, we know that $a_{ii} \geq 0$ and $a_{jj} \geq 0$. Thus to maximize the sum of a_{ii} and a_{jj}, we simply choose $i = j$ and find that

$$\max_{i,j} |a_{ij}| \leq \max_{i} a_{ii}. \tag{A.159}$$

This maximum is achieved for $i = j$ so that

$$\max_{i,j} |a_{ij}| = \max_{i} a_{ii} \tag{A.160}$$

for any positive semidefinite matrix.

In addition, the matrices $\underline{H}_{22}^{(0)}$, $\underline{H}_{22}^{(1)}$, $\underline{H}_{22}^{(2)}$,..., are each symmetric, positive semidefinite, and their rank decreases by 1 after each iteration. To see why, assume without loss of generality that the pivot is a_{11} in a positive semidefinite matrix $\underline{A}$ (we can always perform a symmetric permutation of $\underline{A}$ to make this last statement true). Then partition $\underline{A}$ as

$$\underline{A} = \begin{bmatrix} a_{11} & \bar{a}^T \\ \bar{a} & \underline{A}_{22} \end{bmatrix} \tag{A.161}$$

and perform the step

$$\begin{bmatrix} a_{11} & \bar{a}^T \\ \bar{a} & \underline{A}_{22} \end{bmatrix} - \frac{1}{a_{11}} \begin{bmatrix} a_{11} \\ \bar{a} \end{bmatrix} \begin{bmatrix} a_{11} & \bar{a}^T \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \underline{A}_{22} - \frac{1}{a_{11}} \bar{a}\bar{a}^T \end{bmatrix}. \tag{A.162}$$

In the setting of our algorithm, this corresponds to, for example, taking $\underline{H}_{22}^{(0)}$ and computing $\underline{H}_{22}^{(1)}$. If $\underline{A}$ had full rank, by virtue of introducing a zero row and column, $\underline{A}_{22} - \frac{1}{a_{11}} \bar{a}\bar{a}^T$ has at most one rank less. In addition, $\underline{A}_{22} - \frac{1}{a_{11}} \bar{a}\bar{a}^T$ appears in the factorization

$$\begin{bmatrix} a_{11} & \bar{a}^T \\ \bar{a} & \underline{A}_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{1}{a_{11}} \bar{a} & \underline{I} \end{bmatrix}}_{\underline{B}} \underbrace{\begin{bmatrix} a_{11} & 0 \\ 0 & \underline{A}_{22} - \frac{1}{a_{11}} \bar{a}\bar{a}^T \end{bmatrix}}_{\underline{C}} \begin{bmatrix} 1 & 0 \\ \frac{1}{a_{11}} \bar{a} & \underline{I} \end{bmatrix}^T. \tag{A.163}$$

From this last equation, we note that

$$\bar{x}^T \underline{A} \bar{x} = \bar{x}^T \underline{B} \underline{C} \underbrace{\underline{B}^T \bar{x}}_{\bar{y}} \tag{A.164}$$

for arbitrary real $\bar{x}$. Then, since $\underline{A}$ is positive semidefinite, we have

$$\bar{y}^T \underline{C} \bar{y} \geq 0. \tag{A.165}$$

Note that $\underline{B}$ is invertible (it has unit entries on the diagonal and is lower triangular), which means that we can produce any real $\bar{y}$ by choosing $\bar{x} = \underline{B}^{-T} \bar{y}$. Thus $\underline{C}$ is also positive semidefinite. Finally, choosing

$$\bar{y} = \begin{bmatrix} 0 \\ \bar{z} \end{bmatrix} \tag{A.166}$$

with $\bar{z}$ real and arbitrary, we observe that

$$\bar{z}^T \left(\underline{A}_{22} - \frac{1}{a_{11}} \bar{a} \bar{a}^T \right) \bar{z} \geq 0. \tag{A.167}$$

In the context of our algorithm, this shows that $\underline{H}_{22}^{(1)}$ is positive semidefinite given $\underline{H}_{22}^{(0)}$ positive semidefinite and the same holds for each subsequent matrix $\underline{H}_{22}^{(2)}$, $\underline{H}_{22}^{(3)}$, etc.

Taking these two observations into account, a naive implementation of the rank revealing LDLT algorithm is presented in Algorithm A.1 which can terminate before n iterations when the rank of $\underline{A}$ is less than n. The algorithm yields a factorization

$$\underline{A} = \sum_{k=1}^{r} d_k \bar{l}_k \bar{l}_k^T \tag{A.168}$$

which can also be written as $\underline{A} = \underline{L} \underline{D} \underline{L}^T$ with $\underline{D}$ a diagonal matrix containing the inverse of the pivot values and $\underline{L} = \begin{bmatrix} \bar{l}_1 & \bar{l}_2 & \cdots & \bar{l}_r \end{bmatrix}$ containing the pivot columns. The method is naive because the computational complexity of line 20 is $\mathcal{O}(rn^2)$ which we can reduce if the rank r of $\underline{A}$ is much smaller than n. We do this in practice by not updating (A.150) in its entirety. Instead, we only update the diagonal and those columns needed in the factorization.

To see how this is done, suppose that at iteration k, the algorithm decides that index p contains the pivot. To update the diagonal of $\underline{A}$, we need to set

$$a_{ii} = a_{ii} - a_{ip} a_{ip} / a_{pp}. \tag{A.169}$$

This has linear computational complexity since i runs from 1 to n. If the rank of $\underline{A}$ is r, this

Algorithm A.1 Naive implementation of the pivoted LDLT factorization for a symmetric, positive semidefinite matrix $\underline{A}$.

```
1  n = size(A,1);
2  tol = 10^-14*log(n);
3
4  d = [];
5  L = [];
6  for k = 1:n
7      % Compute the pivot and pivot index
8      [pivot,index] = max(diag(A));
9
10     % Terminate the iterative process if the pivot is small
11     if pivot < tol
12         break;
13     end
14
15     % Store the pivot and pivot column
16     L = [L, A(:,index)];
17     d = [d; 1./pivot];
18
19     % Update the matrix
20     A = A - A(:,index)*A(index,:)./pivot;
21 end
```

costs $\mathcal{O}(rn)$ operations since we repeat this computation at each iteration, and there are a total of r iterations. To take advantage of this fact, we store the diagonal separately and update it in this way. Similarly, instead of updating the rest of $\underline{A}$, we only update columns that become selected by the new pivot. To do this, we note that at iteration k, column p of the updated $\underline{A}$ (denoted by $\bar{a}_p$) is of the form

$$\bar{a}_p - \sum_{j=1}^{k-1} \frac{1}{d_j} \bar{l}_j (\bar{l}_j)_p \tag{A.170}$$

so that, once we come to such an iteration, we perform this summation, then store this new vector as $\bar{l}_k$. Algorithm A.2 shows pseudocode describing this process. In particular, the diagonal update is performed on line 25, and the pivot column update is performed on lines 18 to 22. Note that the complexity of the pivot column update is now approximately

$$\sum_{k=1}^{r} \left\{ \sum_{j=1}^{k-1} 2n \right\} = 2n \left[r(r+1)/2 - r \right] \tag{A.171}$$

Algorithm A.2 Faster implementation of the pivoted LDLT factorization for a symmetric, positive semidefinite matrix $\underline{A}$. The algorithm is fast when the rank r of $\underline{A}$ is much smaller than size n of the matrix.

```
n = size(A,1);
tol = 10^-14*log(n);

d = [];
L = [];
diagonal = diag(A); % Only compute the diagonal in practice
for k = 1:n
    % Compute the pivot and pivot index
    [pivot,index] = max(diagonal);

    % Terminate the iterative process if the pivot is small
    if pivot < tol
        break;
    end

    % Store the pivot and pivot column
    d = [d; 1./pivot];
    L = [L, A(:,index)];
    for j = 1:k-1
        % Update the pivot column accordingly
        L(:,end) = L(:,end) - L(:,j)*L(index,j)*d(j);
    end

    % Update the diagonal
    diagonal = diagonal - (L(:,end).^2)./pivot;
end
```

which is $\mathcal{O}(r^2 n)$ computations rather than $\mathcal{O}(rn^2)$ as in the naive approach. The paper [51] shows that $\underline{H}_{22}$ has rank $\mathcal{O}(\log n)$ so that applying Algorithm A.2 costs $\mathcal{O}(n(\log n)^2)$ operations when applied to $\underline{H}_{22}$.

Finally, we note that we have not exploited the fact that $\underline{H}_{22}$ is Hankel. We do so to avoid storing all of $\underline{H}_{22}$. This is because a Hankel matrix is fully characterized by a vector of $2n-1$ entries (n entries along the first column and $n-1$ entries along the last row so as to avoid double counting the last entry in the first column). Thus, in practice, we compute these $2n-1$ entries and store them in a vector $\bar{h}_{22}$ (we use an ordering beginning with the first column, then appending the $n-1$ entries from the last row). From this vector, we can obtain column j of $\underline{H}_{22}$ by choosing entries j to $j+n-1$ of $\bar{h}_{22}$. In addition, we can select the diagonal of $\underline{H}_{22}$ by choosing odd entries of $\bar{h}_{22}$ from 1 to $2n-1$ (there are a total of n

such entries). An algorithm in the style of Algorithm A.2 suitable for Hankel matrices would index the columns and diagonal of $\underline{A}$ in this way (and may eliminate the inner loop in favor of a vectorized implementation).

A.4 Fast Toeplitz Products

With the factorization of

$$\underline{H}_{22} = \sum_{k=1}^{r} d_k \bar{l}_k \bar{l}_k^T \tag{A.172}$$

as computed by the rank revealing LDLT algorithm in Section A.3, we consider the final problem of performing the matrix-vector product

$$(\underline{H}_{22} \circ \underline{T}_{22}^T)\bar{\phi}_2 \tag{A.173}$$

from (A.149). Using the factorization, we have

$$\underline{H}_{22} \circ \underline{T}_{22}^T = \left(\sum_{k=1}^{r} d_k \bar{l}_k \bar{l}_k^T \right) \circ \underline{T}_{22}^T \tag{A.174}$$

$$= \sum_{k=1}^{r} d_k (\bar{l}_k \bar{l}_k^T \circ \underline{T}_{22}^T) \tag{A.175}$$

which is a sum of Hadamard products of rank one matrices and a Toeplitz matrix. We can distribute the Hadamard product because for any three matrices $\underline{A}$, $\underline{B}$, and $\underline{C}$,

$$(\underline{A} + \underline{B}) \circ \underline{C} = \underline{A} \circ \underline{C} + \underline{B} \circ \underline{C}. \tag{A.176}$$

This is because the Hadamard product is an entrywise product (the entry $(\underline{A} \circ \underline{B})_{ij} = a_{ij} b_{ij}$ and scalar multiplication is distributive). Since the entries of a rank one term are $(\bar{l}_k \bar{l}_k^T)_{ij} = (\bar{l}_k)_i (\bar{l}_k)_j$, then

$$(\bar{l}_k \bar{l}_k^T \circ \underline{T}_{22}^T)_{ij} = (\bar{l}_k)_i (\bar{l}_k)_j (\underline{T}_{22}^T)_{ij} \tag{A.177}$$

which corresponds to a scaling of rows of $\underline{T}_{22}^T$ by $(\bar{l}_k)_i$ and scaling of columns by $(\bar{l}_k)_j$. Thus, rather than writing a Hadamard product, we can equivalently write

$$\bar{l}_k \bar{l}_k^T \circ \underline{T}_{22}^T = \text{diag}(\bar{l}_k) \underline{T}_{22}^T \text{diag}(\bar{l}_k) \tag{A.178}$$

where the diagonal matrices perform the row and column scaling. This means that

$$\underline{H}_{22} \circ \underline{T}_{22}^T = \sum_{k=1}^{r} d_k \text{diag}(\bar{l}_k) \underline{T}_{22}^T \text{diag}(\bar{l}_k) \tag{A.179}$$

so that, in a matrix-vector product, the crucial operation is the matrix-vector product of the Toeplitz matrix with a vector (products with the diagonal matrices $\mathrm{diag}(\bar{l}_k)$ only cost $\mathcal{O}(n)$ operations, as does scaling by d_k). Thus our focus is on performing the r Toeplitz vector products with $\underline{T}_{22}^T$ efficiently.

Rather than explain how to perform the multiplication with $\underline{T}_{22}^T$, we will start by using a generic Toeplitz matrix $\underline{T}$, then later specialize to $\underline{T}_{22}^T$. The idea is to embed the Toeplitz matrix $\underline{T}$ inside a larger circulant matrix $\underline{C}$ for which fast matrix-vector multiplies exist that exploit the FFT. That is, we construct

$$\underline{C} = \begin{bmatrix} \underline{T} & \underline{C}_{12} \\ \underline{C}_{21} & \underline{C}_{22} \end{bmatrix} \tag{A.180}$$

so that

$$\begin{bmatrix} \underline{T} & \underline{C}_{12} \\ \underline{C}_{21} & \underline{C}_{22} \end{bmatrix} \begin{bmatrix} \bar{x} \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{T}\bar{x} \\ \underline{C}_{21}\bar{x} \end{bmatrix}. \tag{A.181}$$

Notice that we obtain the Toeplitz matrix-vector product $\underline{T}\bar{x}$ as long as we consider only the first n rows of the output of this circulant matrix-vector product.

How then does one embed the Toeplitz matrix $\underline{T}$ into a larger circulant matrix $\underline{C}$? Recall that a Toeplitz matrix has constant diagonals, whereas a circulant matrix is a special type of Toeplitz matrix with "wraparound" [25]. For example,

$$\underline{T} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} \\ t_1 & t_0 & t_{-1} \\ t_2 & t_1 & t_0 \end{bmatrix}, \qquad \underline{C} = \begin{bmatrix} c_0 & c_4 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_4 & c_3 \\ c_3 & c_2 & c_1 & c_0 & c_4 \\ c_4 & c_3 & c_2 & c_1 & c_0 \end{bmatrix}. \tag{A.182}$$

Notice how $\underline{T}$ is specified by $2n-1$ numbers (for example, its first column and first row taking care not to repeat the overlap of the two vectors) whereas $\underline{C}$ is specified by its first column only. Using this small example of $\underline{T}$, we show how to perform the embedding (A.180). We can write

$$\underline{C} = \left[\begin{array}{ccc|cc} t_0 & t_{-1} & t_{-2} & t_2 & t_1 \\ t_1 & t_0 & t_{-1} & t_{-2} & t_2 \\ t_2 & t_1 & t_0 & t_{-1} & t_{-2} \\ \hline t_{-2} & t_2 & t_1 & t_0 & t_{-1} \\ t_{-1} & t_{-2} & t_2 & t_1 & t_0 \end{array}\right] \tag{A.183}$$

which shows that the seed to the circulant matrix (its first column) should be the first column

of $\underline{T}$ augmented by the reversal of the first row (excluding its first entry). If $\underline{T}$ is square with n rows, then $\underline{C}$ will be square with $2n-1$ rows (it is possible to embed $\underline{T}$ into larger circulant matrices, but we will always choose this approach). We will refer to the first column of $\underline{C}$ as $\bar{c}$ which we call the seed vector.

We now explain how to perform the matrix-vector product with $\underline{C}$ efficiently following [25]. We will assume $\underline{C}$ is an $m \times m$ circulant matrix with seed vector

$$\bar{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{m-1} \end{bmatrix}. \tag{A.184}$$

The key is to write $\underline{C}$ as the sum of m shift matrices, each appropriately scaled. That is, defining

$$\underline{\mathcal{D}}_m = \begin{bmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \end{bmatrix} \tag{A.185}$$

as the $m \times m$ circular downshift matrix, we can write

$$\underline{C} = \sum_{k=0}^{m-1} c_k \underline{\mathcal{D}}_m^k \tag{A.186}$$

where $\underline{\mathcal{D}}_m^0 = \underline{I}$. If we knew the eigendecomposition $\underline{\mathcal{D}}_m = \underline{V}\underline{\Lambda}\underline{V}^{-1}$ with $\underline{V}$ the matrix whose columns are eigenvectors of $\underline{\mathcal{D}}_m$ and $\underline{\Lambda} = \text{diag}(\bar{\lambda})$ where $\bar{\lambda}$ is the vector containing the eigenvalues of $\underline{\mathcal{D}}_m$, then

$$\underline{\mathcal{D}}_m^k = \underbrace{(\underline{V}\underline{\Lambda}\underline{V}^{-1})(\underline{V}\underline{\Lambda}\underline{V}^{-1})\cdots(\underline{V}\underline{\Lambda}\underline{V}^{-1})}_{k \text{ times}} \tag{A.187}$$

$$= \underline{V}\underline{\Lambda}\underbrace{\underline{V}^{-1}\underline{V}}_{\underline{I}}\underline{\Lambda}\underbrace{\underline{V}^{-1}\underline{V}}_{\underline{I}}\cdots\underbrace{\underline{V}^{-1}\underline{V}}_{\underline{I}}\underline{\Lambda}\underline{V}^{-1} \tag{A.188}$$

$$= \underline{V}\underline{\Lambda}^k\underline{V}^{-1}. \tag{A.189}$$

Using this fact in (A.186) yields

$$\underline{C} = \sum_{k=0}^{m-1} c_k \underline{V} \underline{\Lambda}^k \underline{V}^{-1} \tag{A.190}$$

$$= \underline{V} \left[\sum_{k=0}^{m-1} c_k \underline{\Lambda}^k \right] \underline{V}^{-1}. \tag{A.191}$$

Notice that the sum is taken over diagonal matrices so that this last expression is itself an eigendecomposition of $\underline{C}$. That is, $\underline{C}$ is diagonalized by the same eigenvectors as $\underline{\mathcal{D}}_m$. In addition, we note that

$$\sum_{k=0}^{m-1} c_k \underline{\Lambda}^k = \operatorname{diag}\left[\sum_{k=0}^{m-1} c_k \bar{\lambda}^k \right] \tag{A.192}$$

where $\bar{\lambda}^k$ is to be understood in an entrywise sense. That is,

$$\bar{\lambda}^k = \underbrace{\bar{\lambda} \circ \bar{\lambda} \circ \cdots \circ \bar{\lambda}}_{k \text{ times}}. \tag{A.193}$$

Then we can write

$$\sum_{k=0}^{m-1} c_k \bar{\lambda}^k = \underbrace{\begin{bmatrix} \bar{\lambda}^0 & \bar{\lambda}^1 & \bar{\lambda}^2 & \cdots & \bar{\lambda}^{m-1} \end{bmatrix}}_{\tilde{\underline{V}}} \bar{c}. \tag{A.194}$$

We now show that the eigenvectors and eigenvalues of $\underline{\mathcal{D}}_m$, and consequently $\underline{C}$ are related to the DFT so that products with $\underline{V}$, $\tilde{\underline{V}}$, and their inverses can be performed using the FFT. Take column $j+1$ of the DFT matrix $\underline{F}_m$ from (A.11) and multiply by $\underline{\mathcal{D}}_m$. This gives

$$\begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & 1 & \end{bmatrix} \underbrace{\begin{bmatrix} \omega_m^{0\cdot j} \\ \omega_m^{1\cdot j} \\ \omega_m^{2\cdot j} \\ \vdots \\ \omega_m^{(m-1)\cdot j} \end{bmatrix}}_{\bar{v}_j} = \begin{bmatrix} \omega_m^{(m-1)\cdot j} \\ \omega_m^{0\cdot j} \\ \omega_m^{1\cdot j} \\ \vdots \\ \omega_m^{(m-2)\cdot j} \end{bmatrix}. \tag{A.195}$$

Now rewrite the right hand side as a scalar times the original vector. In particular, notice that

$$\omega_m^{-j} \begin{bmatrix} \omega_m^{0\cdot j} \\ \omega_m^{1\cdot j} \\ \omega_m^{2\cdot j} \\ \vdots \\ \omega_m^{(m-1)\cdot j} \end{bmatrix} = \begin{bmatrix} \omega_m^{(-1)\cdot j} \\ \omega_m^{0\cdot j} \\ \omega_m^{1\cdot j} \\ \vdots \\ \omega_m^{(m-2)\cdot j} \end{bmatrix} \tag{A.196}$$

and that the first entry can be rewritten as $\omega_m^{(m-1)\cdot j}$ since

$$\omega_m^{(m-1)\cdot j} = \underbrace{\omega_m^{m\cdot j}}_{1}\omega_m^{(-1)\cdot j}. \tag{A.197}$$

We have just shown that $\underline{\mathcal{D}}_m \bar{v}_j = \omega_m^{-j}\bar{v}_j$ so that $\bar{v}_j$ is an eigenvector of $\underline{\mathcal{D}}_m$ with corresponding eigenvalue ω_m^{-j}. This holds for all integers $0 \leq j \leq m-1$ so that, arranging the columns $\bar{v}_j$ into a matrix gives $\underline{V} = \underline{F}_m$. Similarly, the eigenvalues can be arranged to produce

$$\bar{\lambda} = \begin{bmatrix} \omega_m^{-0} \\ \omega_m^{-1} \\ \vdots \\ \omega_m^{-(m-1)} \end{bmatrix}. \tag{A.198}$$

From (A.194), we see that $\tilde{\underline{V}}$ has entries

$$\tilde{\underline{V}} = \begin{bmatrix} \omega_m^{-0\cdot 0} & \omega_m^{-0\cdot 1} & \cdots & \omega_m^{-0\cdot(m-1)} \\ \omega_m^{-1\cdot 0} & \omega_m^{-1\cdot 1} & \cdots & \omega_m^{-1\cdot(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_m^{-(m-1)\cdot 0} & \omega_m^{-(m-1)\cdot 1} & \cdots & \omega_m^{-(m-1)\cdot(m-1)} \end{bmatrix}. \tag{A.199}$$

This is precisely the complex conjugate of $\underline{F}_m$. Since $\underline{F}_m$ is symmetric (recall (A.16)), $\tilde{\underline{V}} = \underline{F}_m^*$. Finally, using (A.18), we see that $\tilde{\underline{V}} = m\underline{F}_m^{-1}$.

Equipped with this knowledge regarding $\underline{V}$ and $\tilde{\underline{V}}$, we proceed to compute the matrix-vector product $\bar{y} = \underline{C}\bar{x}$. Given (A.191)-(A.194), we compute the product in stages. That is,

$$\bar{y} = \underline{C}\bar{x} \tag{A.200}$$

$$= \underline{V}\mathrm{diag}(\underbrace{\tilde{\underline{V}}\bar{c}}_{\bar{w}})\underbrace{\underline{V}^{-1}\bar{x}}_{\bar{z}}. \tag{A.201}$$

First, we compute $\bar{z}$ by applying an inverse FFT to $\bar{x}$. We also compute $\bar{w}$ by applying an inverse FFT to $\bar{c}$ and scaling by m. Then we compute the product $\mathrm{diag}(\bar{w})\bar{z} = \bar{w} \circ \bar{z}$ and take the FFT of this resulting vector. This process gives $\bar{y}$.

We now specialize to the case where $\underline{T} = \underline{T}_{22}^T$. Since $\underline{T}_{22}^T$ is an $n \times n$ matrix, the corresponding circulant matrix has size $m = 2n-1$. Each FFT and inverse FFT costs $\mathcal{O}(m\log m) = \mathcal{O}(n\log n)$ operations. The Hadamard product of $\bar{w}$ and $\bar{z}$ costs $\mathcal{O}(m) = \mathcal{O}(n)$ operations so that the total cost of multiplying $\underline{T}_{22}^T$ with a vector is $\mathcal{O}(n\log n)$ operations.

In practice, we only need to apply the inverse FFT to compute $\bar{w}$ once, but need to perform the circulant matrix-vector product r times (recall (A.179)). This leads to an overall cost of $\mathcal{O}(rn \log n)$ for this stage of the algorithm, and since $r \approx \log n$, we continue to have $\mathcal{O}(n(\log n)^2)$ computational complexity overall.

A.5 Computing the Hankel and Toeplitz Seed Vectors

We have made one glaring omission in presenting the algorithm. The entries (A.128) in the Hankel and Toeplitz matrices depend on the evaluation of the Gamma function. In the algorithms that we have presented thus far, we only have to evaluate the first column and last row of $\underline{H}$ and the first row of $\underline{T}^T$ to characterize each matrix. However, we have to make sure that we can perform these calculations with near linear complexity, otherwise the $\mathcal{O}(n(\log n)^2)$ algorithm will not be of any use. The key will be to use

$$\Gamma(z+1) = z\Gamma(z) \tag{A.202}$$

allowing a recursive algorithm with computational complexity $\mathcal{O}(n)$.

We start with the Hankel matrix. For now, we ignore the $i = j = 0$ entry and focus on

$$\frac{\Gamma\left(\frac{i+j}{2}\right)}{\Gamma\left(\frac{i+j}{2}+\frac{3}{2}\right)} \tag{A.203}$$

for the first column of $\underline{H}$. In the first column, $j = 0$ so that entries are of the form

$$h_i = \frac{\Gamma\left(\frac{i}{2}\right)}{\Gamma\left(\frac{i}{2}+\frac{3}{2}\right)} \tag{A.204}$$

with $i \geq 1$. Note that if we try to compute

$$h_{i+2} = \frac{\Gamma\left(\frac{i+2}{2}\right)}{\Gamma\left(\frac{i+2}{2}+\frac{3}{2}\right)} = \frac{\Gamma\left(\frac{i}{2}+1\right)}{\Gamma\left(\frac{i}{2}+\frac{3}{2}+1\right)}, \tag{A.205}$$

we can use (A.202) in both the numerator and denominator to obtain

$$h_{i+2} = \underbrace{\frac{\Gamma\left(\frac{i}{2}\right)}{\Gamma\left(\frac{i}{2}+\frac{3}{2}\right)}}_{h_i} \frac{\frac{i}{2}}{\left(\frac{i}{2}+\frac{3}{2}\right)} \tag{A.206}$$

where we have identified the recursion. Simplifying this expression yields

$$h_{i+2} = h_i \frac{i}{i+3} \tag{A.207}$$

which indicates that at each iteration of the recursion, we multiply by a number smaller than 1 whose value approaches 1 in the limit $i \to \infty$. This recursion is stable. For the recursion to be of any use, we need to explicitly compute the base cases h_1 and h_2. The first is

$$h_1 = \frac{\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(2\right)} = \sqrt{\pi} \tag{A.208}$$

while the second is

$$h_2 = \frac{\Gamma\left(1\right)}{\Gamma\left(\frac{5}{2}\right)} = \frac{4}{3\sqrt{\pi}}. \tag{A.209}$$

Both can be computed from knowledge of $\Gamma(1) = 1$, $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, and property (A.202). In theory, we also need to compute the last row of $\underline{H}$. Rather than index i from 1 to $n-2$ (used to compute the first column of $\underline{H}$) we index from 1 to $2n-2$. This change includes the last row of $\underline{H}$. We can add the $i = j = 0$ entry $h_0 = 1$ to the beginning of this vector to obtain the complete Hankel matrix seed vector of length $2n+1$. Although we have computed the seed for $\underline{H}$, it is trivial to obtain the corresponding seed for the submatrix $\underline{H}_{22}$ needed in the LDLT factorization; simply leave out the first two entries of the seed.

Next, we consider the Toeplitz matrix $\underline{T}^T$. Recall that the first row and column of $\underline{T}^T$ are needed to specify the Toeplitz matrix. From (A.128), we note that since $\underline{T}$ is lower triangular, its transpose (which is upper triangular) has zeros in its first column. This gives $t_i = 0$ for $i > 0$ in (A.182). Thus we need only compute the first row of $\underline{T}^T$. By taking the transpose, we interchange i and j in (A.128) giving

$$\frac{\Gamma\left(\frac{j-i}{2} - \frac{1}{2}\right)}{\Gamma\left(\frac{j-i}{2} + 1\right)}$$

when $j - i$ is even. The first row has $i = 0$ so that when j is even, we have

$$t_{-j} = \frac{\Gamma\left(\frac{j}{2} - \frac{1}{2}\right)}{\Gamma\left(\frac{j}{2} + 1\right)} \tag{A.210}$$

(the negative subscript on t agrees with the description of Toeplitz matrices in (A.182)). We

explicitly compute

$$t_0 = \frac{\Gamma\left(-\frac{1}{2}\right)}{\Gamma(1)} = -2\sqrt{\pi} \tag{A.211}$$

and notice that the remaining entries have already been computed in the seed for $\underline{H}$. In particular,

$$t_{-j} = \begin{cases} h_{j-1} & j \text{ even} \\ 0 & \text{otherwise} \end{cases} \tag{A.212}$$

and $0 < j \leq n$. With t_0 and $t_i = 0$ for $0 < i \leq n$ we have fully specified the seed vector for $\underline{T}^T$ (the vector has $2n+1$ entries). In practice, we also need the seed for the submatrix $\underline{T}_{22}^T$ but this can be obtained directly from the seed for $\underline{T}^T$ by eliminating t_n and t_{-n}, giving a seed of length $2n-1$.

A.6 Finalizing the Algorithm

Given the theoretical development of the previous five sections, we are prepared to state how the fast algorithm is implemented in practice. Roughly speaking, the steps are:

1. Sample the function ϕ at Chebyshev nodes.
2. Take the inverse FFT of the samples concatenated with the reversal of the interior samples.
3. Keep the first set of entries and scale all but the first and last by 2 to obtain Chebyshev coefficients.
4. Construct the Hankel seed vector recursively and use it to populate the Toeplitz seed vector.
5. Scale the Chebyshev coefficients by the entries in scaling matrix $\underline{D}_1$ (see (A.127)).
6. Take the inner product of that vector with the first row of $\underline{H} \circ \underline{T}^T$ and store this scalar y_1.
7. Compute the rank revealing LDLT factorization of the submatrix $\underline{H}_{22}$.
8. Compute the inverse FFT of the seed to the circulant matrix that embeds the submatrix $\underline{T}_{22}^T$.
9. For all $r \approx \log n$ rank 1 components of the LDLT factorization: take the vector from Step 5, scale by the rank 1 vector, compute the multiplication by $\underline{T}_{22}^T$ (pad the vector

with zeros, take the inverse FFT, scale by the vector in Step 8, take the FFT, then discard the padded entries), and scale by the rank 1 vector.

10. Sum all r vectors from Step 9 scaled by their respective pivots to obtain vector $\bar{y}_2$.

11. Concatenate y_1 from Step 6 and $\bar{y}_2$ from Step 10 and scale by the entries in the scaling matrix $\underline{D}_2$ (see (A.127)).

Steps 1-3 produce Chebyshev coefficients $\bar{\phi}_{\text{Cheb}}$ from a vector of samples at Chebyshev nodes. Then Steps 4-11 compute Legendre coefficients $\bar{\phi}_{\text{Leg}} = \underline{D}_2(\underline{H} \circ \underline{T}^T)\underline{D}_1\bar{\phi}_{\text{Cheb}}$. No step has computational complexity exceeding $\mathcal{O}(n(\log n)^2)$ and, in practice, it is possible to combine steps to avoid certain storage costs. In a MATLAB implementation of the fast algorithm, the method is only faster than the direct method described in Section 4.2 when n exceeds 256, so we switch between the two algorithms accordingly.

The astute reader will recall that, in this appendix, we have computed the Legendre coefficients for an expansion in orthogonal Legendre polynomials, whereas in the thesis, an expansion in orthonormal Legendre polynomials is required. To obtain such an expansion we note that orthonormal Legendre polynomials $p_k(x)$ are related to orthogonal Legendre polynomials $P_k(x)$ via the relation

$$p_k(x) = \frac{P_k(x)}{\|P_k(x)\|_{L^2}} \tag{A.213}$$

where

$$\|P_k(x)\|_{L^2} = \left(\int_{-1}^{1} [P_k(x)]^2 dx\right)^{\frac{1}{2}} \tag{A.214}$$

$$= \sqrt{\frac{2}{2k+1}}. \tag{A.215}$$

Thus, forming the vector $\bar{p}(x)$ of orthonormal Legendre polynomials and the diagonal matrix $\underline{D}$, given by

$$\bar{p}(x) = \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_n(x) \end{bmatrix}, \qquad \underline{D} = \begin{bmatrix} \sqrt{\frac{2}{1}} & & & \\ & \sqrt{\frac{2}{3}} & & \\ & & \ddots & \\ & & & \sqrt{\frac{2}{2n+1}} \end{bmatrix}, \tag{A.216}$$

respectively, we have $\bar{P}(x) = \underline{D}\bar{p}(x)$. A function $p(x)$ expanded in orthogonal Legendre polynomials can then be converted to an expansion in terms of orthonormal Legendre polynomials

as follows:

$$p(x) = \bar{\phi}_{\text{Leg}}^T \bar{P}(x) \tag{A.217}$$

$$= \underbrace{\bar{\phi}_{\text{Leg}}^T \underline{D}}_{\bar{\phi}_{\text{Leg},n}^T} \bar{p}(x). \tag{A.218}$$

This means that the coefficients in the orthonormal expansion are $\bar{\phi}_{\text{Leg},n} = \underline{D}\bar{\phi}_{\text{Leg}}$. This scaling is performed in $\mathcal{O}(n)$ operations, which is negligible compared to the cost of the fast transform.

To conclude this appendix, we note that at the core of the FLT is the idea of applying a matrix-vector product of the form $\underline{D}_2(\underline{H} \circ \underline{T})\underline{D}_1\bar{x}$ with $\underline{D}_1$ and $\underline{D}_2$ diagonal, $\underline{H}$ Hankel, symmetric, positive semidefinite, and numerically low rank, and $\underline{T}$ Toeplitz. The connection coefficients between Chebyshev and Legendre polynomials satisfy such properties, as do their inverse coefficients, as well as connection coefficients between other classes of polynomials [51]. We take this opportunity to mention that the tensor $\underline{\underline{T}}$ of integrals of triples of Legendre polynomials from Section 4.3 also exhibits this property. That is, each frontal slice $\underline{T}_k$ of $\underline{\underline{T}}$ can be written in the form $\underline{D}_2(\underline{H} \circ \underline{T})\underline{D}_1$. In particular, we can choose

$$\underline{D}_1 = \begin{bmatrix} \sqrt{\frac{1}{2}} & & & \\ & \sqrt{\frac{3}{2}} & & \\ & & \ddots & \\ & & & \sqrt{\frac{2n+1}{2}} \end{bmatrix}, \qquad \underline{D}_2 = 2\sqrt{\frac{2k+1}{2}}\underline{D}_1, \tag{A.219}$$

and

$$(\underline{T})_{i+1,j+1} = \begin{cases} A(s-i)A(s-j) & i+j+k \text{ even}, \ |i-j| \leq k, \\ 0 & \text{otherwise}, \end{cases} \tag{A.220}$$

with

$$(\underline{H})_{i+1,j+1} = \begin{cases} \dfrac{A(s-k)}{(2s+1)A(s)} & i+j+k \text{ even}, \ i+j \geq k, \\ 0 & \text{otherwise}, \end{cases} \tag{A.221}$$

for $0 \leq i, j \leq n$ where A and s are defined as in Section 4.3. One can check that $\underline{T}$ is a symmetric, Toeplitz matrix of bandwidth k. Additionally, $\underline{H}$ is a symmetric, Hankel matrix of low rank. When k is small, this is of virtually no use because k is the bandwidth of $\underline{T}_k$ and a naive fast matrix-vector product is already possible. However, when k is large, this may be useful. This is interesting because we have explicitly constructed our algorithms to avoid scenarios with large k, but exploiting this factorization may allow us to loosen the

restriction on large k. We have not exploited this structure in the thesis, but aim to conduct additional investigations along these lines in future work.

Appendix B

Fast Legendre Solvers

This appendix describes how fast Legendre solvers can be applied to solve the local subproblems that arise in the domain decomposition method proposed in Chapter 10. While the appendix focuses on these solvers in the context of domain decomposition, they are equally applicable to the single domain problems presented in Chapters 7 and 8. In fact, the central idea of the method, partial diagonalization via eigendecomposition, was initially proposed to solve Poisson and Helmholtz problems using single domain spectral methods [49]. Since Chapter 10 focused on Poisson and Helmholtz problems, this appendix also focuses on these types of problems.

B.1 Generalized Sylvester Equations

Throughout the thesis, we have encountered systems of the form

$$\sum_{i=1}^{K} \left(\underline{B}_i^T \otimes \underline{A}_i \right) \bar{\phi} = \bar{f} \tag{B.1}$$

where matrices $\underline{A}_i$ and $\underline{B}_i$ and right hand side $\bar{f} = \text{vec}(\underline{F})$ were known, and vector $\bar{\phi} = \text{vec}(\underline{\Phi})$ was unknown. For example, the construction of such a linear system subject to additional constraint equations was the focus of Chapter 7 and the techniques developed there were used in Chapters 8 through 10. For some explicit examples, see Section 7.6 where each operator matrix is described in the form

$$\underline{A} = \sum_{i=1}^{K} \left(\underline{B}_i^T \otimes \underline{A}_i \right). \tag{B.2}$$

Notice that (B.1) can be obtained by applying the vectorization operator vec$(\cdot)$ to the matrix equation

$$\sum_{i=1}^{K} \underline{A}_i \underline{\Phi} \underline{B}_i = \underline{F} \tag{B.3}$$

using the property (5.13) to introduce Kronecker products. Equation (B.3) is called a generalized Sylvester equation [181]. Only certain cases of (B.3) have corresponding efficient solution techniques [182]. Finding new techniques to solve these types of equations is an active area of research [183, 184]. Typical specializations include methods applicable when the number of terms K in (B.3) is small (at most three, typically) or when the matrices in (B.3) have specific low rank properties. When systems do not meet these types of specialized criteria, Krylov subspace methods and other iterative techniques are used [185, 186]. In these methods, the choice of preconditioner is crucial.

Since the systems (B.3) in this thesis do not always have special rank properties, we will use a Krylov subspace method to solve them. We use PCG or PGMRES as in Chapter 10 to do so. It is important to note that there is no need to form the matrix (B.2) explicitly when performing matrix vector products. Instead, the product is performed using one-dimensional matrices $\underline{A}_i$ and $\underline{B}_i$ only. That is,

$$\underline{A}\bar{\phi} = \text{vec}\left(\sum_{i=1}^{K} \underline{A}_i \underline{\Phi} \underline{B}_i\right). \tag{B.4}$$

Since $\underline{A}_i$ and $\underline{B}_i$ are typically banded matrices, as long as their bandwidth is small compared to the size $L+1$ of the matrix $\underline{\Phi}$, their product with $\underline{\Phi}$ can be computed in $\mathcal{O}(L^2)$ operations. This is linear in the number of entries in vector $\bar{\phi}$. In this way, the matrix-vector product $\underline{A}\bar{\phi}$ can be computed efficiently. In the following section, we describe how to precondition such a Krylov subspace method.

B.2 Preconditioners for the Helmholtz Problem

As preconditioner, we use direct solvers for special three term generalized Sylvester equations. We will consider two special cases: one designed to solve the Helmholtz problem on a square domain subject to homogeneous Dirichlet boundary conditions, and one designed to solve the same problem with homogeneous Neumann boundary conditions. The first case is required when applying the Dirichlet preconditioner from Chapter 10 (used for each local subproblem), and the second case is required when applying $\underline{A}_{11}^{-1}$ in the domain decomposition algorithm in Chapter 10 (used as a component of the preconditioner for each local subproblem).

In the first case, we must solve the saddle point system

$$\begin{bmatrix} \underline{S}_{DL}\underline{S}_{DL}^T \otimes \tilde{\underline{S}}\tilde{\underline{S}}^T + \tilde{\underline{S}}\tilde{\underline{S}}^T \otimes \underline{S}_{DL}\underline{S}_{DL}^T - k^2\tilde{\underline{S}}\tilde{\underline{S}}^T \otimes \tilde{\underline{S}}\tilde{\underline{S}}^T & \underline{C}^T \\ \underline{C} & 0 \end{bmatrix} \begin{bmatrix} \bar{\phi} \\ \bar{\nu} \end{bmatrix} = \begin{bmatrix} \bar{b} \\ 0 \end{bmatrix} \tag{B.5}$$

where constraints $\underline{C}\bar{\phi} = 0$ impose homogeneous Dirichlet boundary conditions on the boundary of the canonical square domain (see Section 7.6 for such an example). To solve this system using the null space method, we let

$$\bar{\phi} = \underline{Z}\bar{\phi}_p + \bar{\phi}_z \tag{B.6}$$

with $\underline{C}\underline{Z} = 0$ and $\underline{C}\bar{\phi}_z = 0$. For this problem, the choice

$$\underline{Z} = \underbrace{\begin{bmatrix} 0 \\ \underline{I} \end{bmatrix}}_{\underline{Z}_I} \otimes \begin{bmatrix} 0 \\ \underline{I} \end{bmatrix}, \qquad \bar{\phi}_z = 0, \tag{B.7}$$

satisfies both conditions when the zero block in $\underline{Z}_I$ is 2-by-$(L-1)$ where L is the polynomial degree (equal in both the x_1 and x_2 dimensions) of the basis used to represent the solution on the square and $\underline{I}$ is the $(L-1)$-by-$(L-1)$ identity matrix. A simple way to see how this choice for $\underline{Z}$ and $\bar{\phi}_z$ makes sense is to rewrite (B.6) as

$$\mathrm{vec}(\underline{\Phi}) = (\underline{Z}_I \otimes \underline{Z}_I)\,\mathrm{vec}(\underline{\Phi}_p) \tag{B.8}$$

where $\bar{\phi}_p = \mathrm{vec}(\underline{\Phi}_p)$. Then using (5.13), we note that

$$\mathrm{vec}(\underline{\Phi}) = \mathrm{vec}(\underline{Z}_I\underline{\Phi}_p\underline{Z}_I^T) \tag{B.9}$$

$$\underline{\Phi} = \underline{Z}_I\underline{\Phi}_p\underline{Z}_I^T. \tag{B.10}$$

This means that

$$\underline{\Phi} = \begin{bmatrix} 0 \\ \underline{I} \end{bmatrix} \underline{\Phi}_p \begin{bmatrix} 0 & \underline{I} \end{bmatrix} \tag{B.11}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & \underline{\Phi}_p \end{bmatrix}. \tag{B.12}$$

In words, the null space method zeros the coefficients in $\underline{\Phi}$ corresponding to basis functions that are nonzero on the boundary of the square, and leaves the coefficients that correspond to basis functions that are zero on the boundary free. Carrying the null space method forward,

we obtain

$$(\underline{Z}_I \otimes \underline{Z}_I)^T \left(\underline{S}_{DL}\underline{S}_{DL}^T \otimes \underline{\tilde{S}}\underline{\tilde{S}}^T + \underline{\tilde{S}}\underline{\tilde{S}}^T \otimes \underline{S}_{DL}\underline{S}_{DL}^T - k^2 \underline{\tilde{S}}\underline{\tilde{S}}^T \otimes \underline{\tilde{S}}\underline{\tilde{S}}^T\right)(\underline{Z}_I \otimes \underline{Z}_I)\,\bar{\phi}_p = (\underline{Z}_I \otimes \underline{Z}_I)^T\,\bar{b}. \quad \text{(B.13)}$$

Using properties (5.11) and (5.12) gives

$$(\underline{B} \otimes \underline{A}_d + \underline{A}_d \otimes \underline{B} - k^2 \underline{A}_d \otimes \underline{A}_d)\bar{\phi}_p = (\underline{Z}_I \otimes \underline{Z}_I)^T\,\bar{b} \quad \text{(B.14)}$$

where $\underline{A}_d = \underline{Z}_I^T \underline{\tilde{S}}\underline{\tilde{S}}^T \underline{Z}_I$ and $\underline{B} = \underline{Z}_I^T \underline{S}_{DL}\underline{S}_{DL}^T \underline{Z}_I$. Then, if $\bar{b} = \mathrm{vec}(\hat{\underline{F}})$, using (5.13), we obtain

$$\underline{A}_d \underline{\Phi}_p \underline{B} + \underline{B}\underline{\Phi}_p \underline{A}_d - k^2 \underline{A}_d \underline{\Phi}_p \underline{A}_d = \underbrace{\underline{Z}_I^T \hat{\underline{F}} \underline{Z}_I}_{\underline{F}_d}. \quad \text{(B.15)}$$

This matrix equation is a three term generalized Sylvester equation (compare with (B.3)) for the unknown $\underline{\Phi}_p$. The matrices $\underline{A}_d$ and $\underline{B}$ are truncated versions of $\underline{\tilde{S}}\underline{\tilde{S}}^T$ and $\underline{S}_{DL}\underline{S}_{DL}^T$ respectively. This is because

$$\underline{Z}_I^T \underline{H} \underline{Z}_I = \begin{bmatrix} 0 & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{H}_{11} & \underline{H}_{12} \\ \underline{H}_{12}^T & \underline{H}_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \underline{I} \end{bmatrix} \quad \text{(B.16)}$$

$$= \underline{H}_{22} \quad \text{(B.17)}$$

for any symmetric matrix $\underline{H}$ and both $\underline{A}_d$ and $\underline{B}$ are formed by this type of product. Thus, by truncated version, we mean specifically that $\underline{A}_d$ and $\underline{B}$ are submatrices obtained by discarding the first two rows and columns of $\underline{\tilde{S}}\underline{\tilde{S}}^T$ and $\underline{S}_{DL}\underline{S}_{DL}^T$ respectively. Since $\underline{S}_{DL}\underline{S}_{DL}^T = \underline{I} - \bar{e}_1\bar{e}_1^T$, this means that $\underline{B} = \underline{I}$. The matrix equation (B.15) becomes

$$\underline{A}_d \underline{\Phi}_p + \underline{\Phi}_p \underline{A}_d - k^2 \underline{A}_d \underline{\Phi}_p \underline{A}_d = \underline{F}_d \quad \text{(B.18)}$$

using this observation. In addition, $\underline{A}_d$ is pentadiagonal (it inherits this structure from $\underline{\tilde{S}}\underline{\tilde{S}}^T$).

In the homogeneous Neumann boundary case, a similar generalized Sylvester equation arises. In that case, we must solve

$$(\underline{S}_{DL}\underline{S}_{DL}^T \otimes \underline{\tilde{S}}\underline{\tilde{S}}^T + \underline{\tilde{S}}\underline{\tilde{S}}^T \otimes \underline{S}_{DL}\underline{S}_{DL}^T - k^2 \underline{\tilde{S}}\underline{\tilde{S}}^T \otimes \underline{\tilde{S}}\underline{\tilde{S}}^T)\bar{\phi} = \bar{b} \quad \text{(B.19)}$$

subject to no constraints. The corresponding matrix equation is

$$\underline{\tilde{S}}\underline{\tilde{S}}^T \underline{\Phi} \underline{S}_{DL}\underline{S}_{DL}^T + \underline{S}_{DL}\underline{S}_{DL}^T \underline{\Phi} \underline{\tilde{S}}\underline{\tilde{S}}^T - k^2 \underline{\tilde{S}}\underline{\tilde{S}}^T \underline{\Phi} \underline{\tilde{S}}\underline{\tilde{S}}^T = \underline{F}_n \quad \text{(B.20)}$$

where $\bar{b} = \text{vec}(\underline{F}_n)$. To solve a problem with the same form as (B.18), we need to exploit the fact that

$$\underline{S}_{DL}\underline{S}_{DL}^T = \begin{bmatrix} 0 & 0 \\ 0 & \underline{I} \end{bmatrix} \tag{B.21}$$

where the first diagonal entry is scalar, and the identity matrix is L-by-L. To do so, we partition

$$\tilde{\underline{S}}\tilde{\underline{S}}^T = \begin{bmatrix} 1 & \bar{s}^T \\ \bar{s} & \underline{S} \end{bmatrix}, \qquad \underline{\Phi} = \begin{bmatrix} \phi_{11} & \bar{\phi}_{12}^T \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix}, \qquad \underline{F}_n = \begin{bmatrix} f_{11} & \bar{f}_{12}^T \\ \bar{f}_{21} & \underline{F}_{22} \end{bmatrix}, \tag{B.22}$$

accordingly. We then multiply (B.20) by

$$\underline{G} = \begin{bmatrix} 1 & 0 \\ -\bar{s} & \underline{I} \end{bmatrix} \tag{B.23}$$

from the left and by its transpose on the right. Under this product, the first term in (B.20) is

$$\underline{G}\tilde{\underline{S}}\tilde{\underline{S}}^T\underline{\Phi}\underline{S}_{DL}\underline{S}_{DL}^T\underline{G}^T = \begin{bmatrix} 1 & 0 \\ -\bar{s} & \underline{I} \end{bmatrix}\begin{bmatrix} 1 & \bar{s}^T \\ \bar{s} & \underline{S} \end{bmatrix}\begin{bmatrix} \phi_{11} & \bar{\phi}_{12}^T \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \underline{I} \end{bmatrix}\begin{bmatrix} 1 & -\bar{s}^T \\ 0 & \underline{I} \end{bmatrix} \tag{B.24}$$

$$= \begin{bmatrix} 1 & \bar{s}^T \\ 0 & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix}\begin{bmatrix} \phi_{11} & \bar{\phi}_{12}^T \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \underline{I} \end{bmatrix} \tag{B.25}$$

$$= \begin{bmatrix} 1 & \bar{s}^T \\ 0 & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix}\begin{bmatrix} 0 & \bar{\phi}_{12}^T \\ 0 & \underline{\Phi}_{22} \end{bmatrix} \tag{B.26}$$

$$= \begin{bmatrix} 0 & \bar{\phi}_{12}^T + \bar{s}^T\underline{\Phi}_{22} \\ 0 & (\underline{S} - \bar{s}\bar{s}^T)\underline{\Phi}_{22} \end{bmatrix}. \tag{B.27}$$

Similarly, the second term is

$$\underline{G}\underline{S}_{DL}\underline{S}_{DL}^T\underline{\Phi}\tilde{\underline{S}}\tilde{\underline{S}}^T\underline{G}^T = \begin{bmatrix} 0 & 0 \\ 0 & \underline{I} \end{bmatrix}\begin{bmatrix} \phi_{11} & \bar{\phi}_{12}^T \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \bar{s} & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix} \tag{B.28}$$

$$= \begin{bmatrix} 0 & 0 \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \bar{s} & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix} \tag{B.29}$$

$$= \begin{bmatrix} 0 & 0 \\ \bar{\phi}_{21} + \underline{\Phi}_{22}\bar{s} & \underline{\Phi}_{22}(\underline{S} - \bar{s}\bar{s}^T) \end{bmatrix}, \tag{B.30}$$

and the third term is

$$\underline{G}\tilde{\underline{S}}\tilde{\underline{S}}^T\underline{\Phi}\tilde{\underline{S}}\tilde{\underline{S}}^T\underline{G}^T = \begin{bmatrix} 1 & \bar{s}^T \\ 0 & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix} \begin{bmatrix} \phi_{11} & \bar{\phi}_{12}^T \\ \bar{\phi}_{21} & \underline{\Phi}_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \bar{s} & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix} \tag{B.31}$$

$$= \begin{bmatrix} \phi_{11} + \bar{s}^T\bar{\phi}_{21} & \bar{\phi}_{12}^T + \bar{s}^T\underline{\Phi}_{22} \\ (\underline{S} - \bar{s}\bar{s}^T)\bar{\phi}_{21} & (\underline{S} - \bar{s}\bar{s}^T)\underline{\Phi}_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \bar{s} & \underline{S} - \bar{s}\bar{s}^T \end{bmatrix} \tag{B.32}$$

$$= \begin{bmatrix} \phi_{11} + \bar{s}^T\bar{\phi}_{21} + \bar{\phi}_{12}^T\bar{s} + \bar{s}^T\underline{\Phi}_{22}\bar{s} & (\bar{\phi}_{12}^T + \bar{s}^T\underline{\Phi}_{22})(\underline{S} - \bar{s}\bar{s}^T) \\ (\underline{S} - \bar{s}\bar{s}^T)(\bar{\phi}_{21} + \underline{\Phi}_{22}\bar{s}) & (\underline{S} - \bar{s}\bar{s}^T)\underline{\Phi}_{22}(\underline{S} - \bar{s}\bar{s}^T) \end{bmatrix}. \tag{B.33}$$

The right hand side is given by

$$\underline{G}\underline{F}_n\underline{G}^T = \begin{bmatrix} 1 & 0 \\ -\bar{s} & \underline{I} \end{bmatrix} \begin{bmatrix} f_{11} & \bar{f}_{12}^T \\ \bar{f}_{21} & \underline{F}_{22} \end{bmatrix} \begin{bmatrix} 1 & -\bar{s}^T \\ 0 & \underline{I} \end{bmatrix} \tag{B.34}$$

$$= \begin{bmatrix} f_{11} & \bar{f}_{12}^T \\ \bar{f}_{21} - \bar{s}f_{11} & \underline{F}_{22} - \bar{s}\bar{f}_{12}^T \end{bmatrix} \begin{bmatrix} 1 & -\bar{s}^T \\ 0 & \underline{I} \end{bmatrix} \tag{B.35}$$

$$= \begin{bmatrix} f_{11} & \bar{f}_{12}^T - f_{11}\bar{s}^T \\ \bar{f}_{21} - \bar{s}f_{11} & \underline{F}_{22} - \bar{s}\bar{f}_{12}^T - \bar{f}_{21}\bar{s}^T + \bar{s}f_{11}\bar{s}^T \end{bmatrix}. \tag{B.36}$$

Combining (B.27), (B.30), (B.33), and (B.36) according to (B.20) gives four matrix equations in the unknowns $\underline{\Phi}_{22}$, $\bar{\phi}_{21}$, $\bar{\phi}_{12}^T$, and ϕ_{11}. They are

$$(\underline{S} - \bar{s}\bar{s}^T)\underline{\Phi}_{22} + \underline{\Phi}_{22}(\underline{S} - \bar{s}\bar{s}^T) - k^2(\underline{S} - \bar{s}\bar{s}^T)\underline{\Phi}_{22}(\underline{S} - \bar{s}\bar{s}^T) = \underline{F}_{22} - \bar{s}\bar{f}_{12}^T - \bar{f}_{21}\bar{s}^T + \bar{s}f_{11}\bar{s}^T, \tag{B.37}$$

and

$$\bar{\phi}_{12}^T + \bar{s}^T\underline{\Phi}_{22} - k^2(\bar{\phi}_{12}^T + \bar{s}^T\underline{\Phi}_{22})(\underline{S} - \bar{s}\bar{s}^T) = \bar{f}_{12}^T - f_{11}\bar{s}^T, \tag{B.38}$$

$$\bar{\phi}_{21} + \underline{\Phi}_{22}\bar{s} - k^2(\underline{S} - \bar{s}\bar{s}^T)(\bar{\phi}_{21} + \underline{\Phi}_{22}\bar{s}) = \bar{f}_{21} - \bar{s}f_{11}, \tag{B.39}$$

$$-k^2(\phi_{11} + \bar{s}^T\bar{\phi}_{21} + \bar{\phi}_{12}^T\bar{s} + \bar{s}^T\underline{\Phi}_{22}\bar{s}) = f_{11}. \tag{B.40}$$

The equations are ordered in this way because the first can be solved independently of the other three, and its solution used to solve the subsequent two, whose solutions can then be used to solve the fourth.

Letting $\underline{A}_n = \underline{S} - \bar{s}\bar{s}^T$, it becomes clear that (B.37) is a generalized Sylvester equation for $\underline{\Phi}_{22}$ given by

$$\underline{A}_n\underline{\Phi}_{22} + \underline{\Phi}_{22}\underline{A}_n - k^2\underline{A}_n\underline{\Phi}_{22}\underline{A}_n = \underline{F}_{22} - \bar{s}\bar{f}_{12}^T - \bar{f}_{21}\bar{s}^T + \bar{s}f_{11}\bar{s}^T, \tag{B.41}$$

whose right hand side is known. In addition, since

$$\bar{s} = -\frac{1}{\sqrt{3}}\bar{e}_2, \tag{B.42}$$

the matrix $\underline{A}_n$ is pentadiagonal (the submatrix $\underline{S}$ is pentadiagonal, and $\underline{A}_n = \underline{S} - (1/3)\bar{e}_2\bar{e}_2^T$). Note that this generalized Sylvester equation is of the same form as the Dirichlet problem (B.18). To solve (B.38), we isolate terms with $\bar{\phi}_{12}^T$ to obtain

$$\bar{\phi}_{12}^T(\underline{I} - k^2\underline{A}_n) = \bar{f}_{12}^T - f_{11}\bar{s}^T - \bar{s}^T\underline{\Phi}_{22}(\underline{I} - k^2\underline{A}_n). \tag{B.43}$$

Multiplying by the inverse of $\underline{I} - k^2\underline{A}_n$ yields

$$\bar{\phi}_{12}^T = (\bar{f}_{12}^T - f_{11}\bar{s}^T)(\underline{I} - k^2\underline{A}_n)^{-1} - \bar{s}^T\underline{\Phi}_{22}. \tag{B.44}$$

Since $\underline{I} - k^2\underline{A}_n$ is pentadiagonal, its inverse applied to a vector can be performed with linear computational complexity in the degree L. Similarly, in (B.39) we isolate for $\bar{\phi}_{21}$ to obtain

$$(\underline{I} - k^2\underline{A}_n)\bar{\phi}_{21} = \bar{f}_{21} - \bar{s}f_{11} - (\underline{I} - k^2\underline{A}_n)\underline{\Phi}_{22}\bar{s} \tag{B.45}$$

and multiply by the same inverse to get

$$\bar{\phi}_{21} = (\underline{I} - k^2\underline{A}_n)^{-1}(\bar{f}_{21} - \bar{s}f_{11}) - \underline{\Phi}_{22}\bar{s}. \tag{B.46}$$

Finally, ϕ_{11} is obtained from (B.40) giving

$$\phi_{11} = -\frac{f_{11}}{k^2} - \bar{s}^T\bar{\phi}_{21} - \bar{\phi}_{12}^T\bar{s} - \bar{s}^T\underline{\Phi}_{22}\bar{s}. \tag{B.47}$$

In both the Dirichlet and Neumann cases, when $k = 0$, we obtain a solver for Poisson equations. However, care must be taken in the Neumann case because (B.47) is invalid under those circumstances (since f_{11}/k^2 results in division by zero). The correct approach in such a situation is to reconsider (B.40) and see that the compatibility condition $f_{11} = 0$ must hold when $k = 0$. This leaves ϕ_{11} unspecified, which represents the coefficient of the constant function in the expansion of the solution. We are free to choose any constant for ϕ_{11} as constant functions are in the null space of the discretized Laplacian when subject to homogeneous Neumann boundary conditions[1].

In practice, solving the Neumann problem is only a building block used to invert $\underline{A}_{11}$

[1] I choose $\phi_{11} = 0$.

(the block diagonal matrix arising from the domain decomposition method in Chapter 10). This is because $\underline{A}_{11} = \underline{Z}_1^T \underline{A} \underline{Z}_1$ where $\underline{A}$ is block diagonal with each block corresponding to a homogeneous Neumann problem. Ideally, we could treat this in the same way that the Dirichlet case was treated, but the structure of $\underline{Z}_1$ is not necessarily of Kronecker product form. For this reason, we use $\underline{Z}_1^T \underline{A}^+ \underline{Z}_1$ as a preconditioner to $\underline{A}_{11}$ where $\underline{A}^+$ is shorthand for applying the process outlined above for solving a homogeneous Neumann problem for each element. When $\underline{A}$ is composed of blocks of the matrix in (B.19), then this preconditioner converges in a small number of iterations independent of the polynomial degree of basis functions. The number of iterations does depend on the choice of parameter l in the domain decomposition method. When l is small, the problem is close to a homogeneous Neumann problem, whereas when l is large, it is close to a homogeneous Dirichlet problem (when l is zero, the problem is a homogeneous Neumann problem and when l is equal to the polynomial degree of basis functions, then it is a homogeneous Dirichlet problem). For example, when l is one or two, this method converges in 4 iterations, independent of the polynomial degree.

B.3 Partial Diagonalization

In either the Dirichlet or Neumann case, the key step to solving each subdomain problem is to solve a matrix equation of the form

$$\underline{A}\underline{\Phi} + \underline{\Phi}\underline{A} - k^2 \underline{A}\underline{\Phi}\underline{A} = \underline{F} \tag{B.48}$$

where $\underline{A}$ is a pentadiagonal matrix related to $\tilde{\underline{S}}\tilde{\underline{S}}^T$, $\underline{\Phi}$ is a matrix of unknown coefficients, k is the wavenumber of the Helmholtz problem, and $\underline{F}$ are known coefficients related to the forcing function (see (B.18) and (B.41) respectively). We use a partial diagonalization technique presented in [49] to do so efficiently. The method is part of a family of techniques [187] which includes the well known fast Poisson solver [25] (a method that solves the Sylvester equation arising from a second order finite difference discretization of Poisson's equation). The idea behind these methods is to exploit the eigenvalue decomposition of $\underline{A}$. In our case, we compute

$$\underline{A}\underline{V} = \underline{V}\underline{\Lambda}. \tag{B.49}$$

Since $\underline{A}$ is symmetric, the eigenvalues in the diagonal matrix $\underline{\Lambda}$ are real and the eigenvectors in $\underline{V}$ are orthonormal. This means that

$$\underline{V}^T \underline{A} \underline{V} = \underline{\Lambda}. \tag{B.50}$$

To solve the generalized Sylvester equation, we let

$$\underline{\Phi} = \underline{X}\underline{V}^T \tag{B.51}$$

where $\underline{X}$ is a new unknown matrix. Substituting this expression into (B.48) gives

$$\underline{A}\underline{X}\underline{V}^T + \underline{X}\underline{V}^T\underline{A} - k^2\underline{A}\underline{X}\underline{V}^T\underline{A} = \underline{F}. \tag{B.52}$$

Multiplying from the right by $\underline{V}$ leads to

$$\underline{A}\underline{X}\underbrace{\underline{V}^T\underline{V}}_{\underline{I}} + \underline{X}\underbrace{\underline{V}^T\underline{A}\underline{V}}_{\underline{\Lambda}} - k^2\underline{A}\underline{X}\underbrace{\underline{V}^T\underline{A}\underline{V}}_{\underline{\Lambda}} = \underbrace{\underline{F}\underline{V}}_{\underline{\hat{F}}} \tag{B.53}$$

which explains our choice of substitution (B.51). This also explains why we call this a partial diagonalization method because we have only diagonalized matrices on the right in each term of the generalized Sylvester equation. A full diagonalization would repeat a similar process to diagonalize matrices on the left as well. To solve for $\underline{X}$, we partition

$$\underline{X} = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_n \end{bmatrix}, \qquad \underline{\hat{F}} = \begin{bmatrix} \bar{f}_1 & \bar{f}_2 & \cdots & \bar{f}_n \end{bmatrix}, \tag{B.54}$$

and note that each column of $\underline{X}$ can be solved for independently. The number of columns n is either $L-1$ or L depending on whether this is a Dirichlet or Neumann problem. We can solve for each column of $\underline{X}$ independently because (B.53) becomes

$$\begin{bmatrix} \underline{A}\bar{x}_1 & \underline{A}\bar{x}_2 & \cdots & \underline{A}\bar{x}_n \end{bmatrix} + \begin{bmatrix} \lambda_1\bar{x}_1 & \lambda_2\bar{x}_2 & \cdots & \lambda_n\bar{x}_n \end{bmatrix} \\ - k^2 \begin{bmatrix} \lambda_1\underline{A}\bar{x}_1 & \lambda_2\underline{A}\bar{x}_2 & \cdots & \lambda_n\underline{A}\bar{x}_n \end{bmatrix} = \begin{bmatrix} \bar{f}_1 & \bar{f}_2 & \cdots & \bar{f}_n \end{bmatrix} \tag{B.55}$$

under the column partitioning. In other words,

$$\underline{A}\bar{x}_j + \lambda_j\bar{x}_j - k^2\lambda_j\underline{A}\bar{x}_j = \bar{f}_j \tag{B.56}$$

$$[\lambda_j\underline{I} + (1-k^2\lambda_j)\underline{A}]\bar{x}_j = \bar{f}_j \tag{B.57}$$

for $j = 1, 2, ..., n$. Notice that the matrix $\lambda_j\underline{I} + (1-k^2\lambda_j)\underline{A}$ inherits the structure of $\underline{A}$, which was pentadiagonal. This means that it costs $\mathcal{O}(n)$ operations to solve each of these systems. Since there are a total of n such systems, the total cost of this step is $\mathcal{O}(n^2)$. This is linear in the number of unknowns since $\underline{X}$ is an n-by-n matrix. However, this is only effective if computing the eigenvalues and eigenvectors in (B.49) and performing the

eigenvector multiplications

$$\hat{\underline{F}} = \underline{F}\underline{V}, \qquad \underline{\Phi} = \underline{X}\underline{V}^T, \tag{B.58}$$

can be done efficiently. If done naively, it costs $\mathcal{O}(n^3)$ operations to compute the eigenvalues and eigenvectors, and $\mathcal{O}(n^3)$ operations to compute the multiplications with eigenvectors [25]. For this reason, we now investigate how to compute the eigenvalues and eigenvectors efficiently when n is large. To do so, we need to exploit the structure of $\underline{A}$.

B.4 Divide and Conquer Eigensolvers

Rather than work directly with $\underline{A}$, we begin by permuting its rows and columns. That is, rather than directly solve (B.48), we let

$$\underline{\Phi} = \underline{P}_{2,n/2}\tilde{\underline{\Phi}}\underline{P}_{2,n/2}^T \tag{B.59}$$

where $\underline{P}_{p,r}$ is the mod-p perfect shuffle permutation matrix of size pr [25]. The particular shuffle matrix that we use is given by

$$\underline{P}_{2,n/2}^T = \underline{I}([(1:2:n),\ (2:2:n)],:) \tag{B.60}$$

using a MATLAB-like notation to index the rows of the identity matrix. Making such a substitution has the effect of separating odd entries from even entries in (B.48) when multiplying by $\underline{P}_{2,n/2}^T$ from the left and by $\underline{P}_{2,n/2}$ from the right. That is (B.48) becomes

$$\begin{aligned}\underline{P}_{2,n/2}^T\underline{A}\underline{P}_{2,n/2}\tilde{\underline{\Phi}}\underline{P}_{2,n/2}^T\underline{P}_{2,n/2} + \underline{P}_{2,n/2}^T\underline{P}_{2,n/2}\tilde{\underline{\Phi}}\underline{P}_{2,n/2}^T\underline{A}\underline{P}_{2,n/2} \\ - k^2\underline{P}_{2,n/2}^T\underline{A}\underline{P}_{2,n/2}\tilde{\underline{\Phi}}\underline{P}_{2,n/2}^T\underline{A}\underline{P}_{2,n/2} = \underline{P}_{2,n/2}^T\underline{F}\underline{P}_{2,n/2}\end{aligned} \tag{B.61}$$

which is

$$\underline{\tilde{A}}\tilde{\underline{\Phi}} + \tilde{\underline{\Phi}}\underline{\tilde{A}} - k^2\underline{\tilde{A}}\tilde{\underline{\Phi}}\underline{\tilde{A}} = \underline{\tilde{F}} \tag{B.62}$$

with $\underline{\tilde{A}} = \underline{P}_{2,n/2}^T\underline{A}\underline{P}_{2,n/2}$ and $\underline{\tilde{F}} = \underline{P}_{2,n/2}^T\underline{F}\underline{P}_{2,n/2}$. The structure of $\underline{\tilde{A}}$ is now

$$\underline{\tilde{A}} = \begin{bmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{bmatrix} \tag{B.63}$$

with $\underline{T}_1$ and $\underline{T}_2$ both symmetric and tridiagonal matrices. Thus, by performing this odd-even perfect shuffle permutation, we have obtained a problem where $\underline{\tilde{A}}$ is symmetric tridiagonal, and in fact, block diagonal with two blocks each tridiagonal.

We will apply the partial diagonalization method of Section B.3 to (B.62) because fast eigensolvers exist for symmetric tridiagonal matrices [53, 54, 55, 57]. Since $\underline{\tilde{A}}$ is block diagonal, we can compute its eigendecomposition by computing the eigendecomposition of each block. That means that

$$\underline{\tilde{A}}\underline{\tilde{V}} = \underline{\tilde{V}}\underline{\tilde{\Lambda}} \tag{B.64}$$

can be computed as

$$\begin{bmatrix} \underline{\tilde{T}}_1 & 0 \\ 0 & \underline{\tilde{T}}_2 \end{bmatrix} \begin{bmatrix} \underline{\tilde{V}}_1 & 0 \\ 0 & \underline{\tilde{V}}_2 \end{bmatrix} = \begin{bmatrix} \underline{\tilde{V}}_1 & 0 \\ 0 & \underline{\tilde{V}}_2 \end{bmatrix} \begin{bmatrix} \underline{\tilde{\Lambda}}_1 & 0 \\ 0 & \underline{\tilde{\Lambda}}_2 \end{bmatrix} \tag{B.65}$$

by performing the independent eigendecompositions $\underline{\tilde{T}}_1\underline{\tilde{V}}_1 = \underline{\tilde{V}}_1\underline{\tilde{\Lambda}}_1$ and $\underline{\tilde{T}}_2\underline{\tilde{V}}_2 = \underline{\tilde{V}}_2\underline{\tilde{\Lambda}}_2$. For this reason, we consider how to solve the generic symmetric tridiagonal eigendecomposition problem $\underline{T}\underline{V} = \underline{V}\underline{\Lambda}$ which alllows us to treat the block diagonal case.

There are several fast methods to perform the eigendecomposition

$$\underline{T}\underline{V} = \underline{V}\underline{\Lambda} \tag{B.66}$$

efficiently, including the method of multiple relatively robust representations and divide and conquer algorithms [188]. In this appendix, we focus on a particular divide and conquer algorithm (there are several variations on the main idea) which treats a symmetric tridiagonal matrix $\underline{T}$ as the sum of a block diagonal matrix and a rank 1 term. The method is closest to the ones described in [54, 57], but uses the root-finding techniques described in [53] and the eigenvector matrix product acceleration idea of [55]. Other root-finding techniques, such as the method described in [56] could be used instead.

For now, we will let $n = 2m$ so that the matrix $\underline{T}$ can be split into two parts of equal size. When this is not the case, a similar algorithm applies, although the splitting will no longer result in submatrices of equal size. To derive a divide and conquer algorithm, we rewrite

$$\underline{T} = \begin{bmatrix} a_1 & b_1 & & & & & & \\ b_1 & a_2 & \ddots & & & & & \\ & \ddots & \ddots & b_{m-1} & & & & \\ & & b_{m-1} & a_m & b_m & & & \\ & & & b_m & a_{m+1} & b_{m+1} & & \\ & & & & b_{m+1} & a_{m+2} & \ddots & \\ & & & & & \ddots & \ddots & b_{2m-1} \\ & & & & & & b_{2m-1} & a_{2m} \end{bmatrix} \tag{B.67}$$

as

$$\underline{T} = \begin{bmatrix} a_1 & b_1 & & & & & & \\ b_1 & a_2 & \ddots & & & & & \\ & \ddots & \ddots & b_{m-1} & & & & \\ & & b_{m-1} & a_m - b_m & 0 & & & \\ & & & 0 & a_{m+1} - b_m & b_{m+1} & & \\ & & & & b_{m+1} & a_{m+2} & \ddots & \\ & & & & & \ddots & \ddots & b_{2m-1} \\ & & & & & & b_{2m-1} & a_{2m} \end{bmatrix} + \underbrace{b_m}_{\rho} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^T \tag{B.68}$$

which is of the form

$$\underline{T} = \begin{bmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{bmatrix} + \rho \begin{bmatrix} \bar{e}_m \\ \bar{e}_1 \end{bmatrix} \begin{bmatrix} \bar{e}_m \\ \bar{e}_1 \end{bmatrix}^T \tag{B.69}$$

with $\underline{T}_1$ and $\underline{T}_2$ both symmetric, tridiagonal matrices of half the size of $\underline{T}$. For the moment, suppose that we can compute the eigendecompositions

$$\underline{T}_1 \underline{U}_1 = \underline{U}_1 \underline{\Lambda}_1, \tag{B.70}$$

$$\underline{T}_2 \underline{U}_2 = \underline{U}_2 \underline{\Lambda}_2, \tag{B.71}$$

then forming

$$\underline{U} = \begin{bmatrix} \underline{U}_1 & 0 \\ 0 & \underline{U}_2 \end{bmatrix} \tag{B.72}$$

and multiplying (B.69) from the left by $\underline{U}^T$ and from the right by $\underline{U}$ yields

$$\underline{U}^T \underline{T} \underline{U} = \begin{bmatrix} \underline{U}_1^T & 0 \\ 0 & \underline{U}_2^T \end{bmatrix} \begin{bmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{bmatrix} \begin{bmatrix} \underline{U}_1 & 0 \\ 0 & \underline{U}_2 \end{bmatrix} + \rho \begin{bmatrix} \underline{U}_1^T & 0 \\ 0 & \underline{U}_2^T \end{bmatrix} \begin{bmatrix} \bar{e}_m \\ \bar{e}_1 \end{bmatrix} \begin{bmatrix} \bar{e}_m \\ \bar{e}_1 \end{bmatrix}^T \begin{bmatrix} \underline{U}_1 & 0 \\ 0 & \underline{U}_2 \end{bmatrix} \tag{B.73}$$

$$= \begin{bmatrix} \underline{U}_1^T \underline{T}_1 \underline{U}_1 & 0 \\ 0 & \underline{U}_2^T \underline{T}_2 \underline{U}_2 \end{bmatrix} + \rho \begin{bmatrix} \underline{U}_1^T \bar{e}_m \\ \underline{U}_2^T \bar{e}_1 \end{bmatrix} \begin{bmatrix} \bar{e}_m^T \underline{U}_1 & \bar{e}_1^T \underline{U}_2 \end{bmatrix}. \tag{B.74}$$

By (B.70) and (B.71), the first term is diagonalized, giving

$$\underline{U}^T \underline{T} \underline{U} = \begin{bmatrix} \underline{\Lambda}_1 & 0 \\ 0 & \underline{\Lambda}_2 \end{bmatrix} + \rho \begin{bmatrix} \underline{U}_1^T \bar{e}_m \\ \underline{U}_2^T \bar{e}_1 \end{bmatrix} \begin{bmatrix} \bar{e}_m^T \underline{U}_1 & \bar{e}_1^T \underline{U}_2 \end{bmatrix}. \tag{B.75}$$

Letting $\bar{u}_1 = \underline{U}_1^T \bar{e}_m$ and $\bar{u}_2 = \underline{U}_2^T \bar{e}_1$ with

$$\hat{\underline{D}} = \begin{bmatrix} \underline{\Lambda}_1 & 0 \\ 0 & \underline{\Lambda}_2 \end{bmatrix}, \qquad \hat{u} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix}, \tag{B.76}$$

we obtain

$$\underline{U}^T \underline{T} \underline{U} = \hat{\underline{D}} + \rho \hat{u} \hat{u}^T. \tag{B.77}$$

Suppose that we use a permutation $\underline{P}$ to order the diagonal entries of $\underline{D}$ so that $d_1 < d_2 < \cdots < d_n$. Then

$$\underline{P}^T \underline{U}^T \underline{T} \underline{U} \underline{P} = \underbrace{\underline{P}^T \hat{\underline{D}} \underline{P}}_{\underline{D}} + \rho \underbrace{(\underline{P}^T \hat{u})}_{\bar{u}} \underbrace{(\hat{u}^T \underline{P})}_{\bar{u}^T} \tag{B.78}$$

is a diagonal plus rank 1 matrix whose eigenvalues and eigenvectors can be computed efficiently (we will see how to do so in a moment). Suppose that the eigendecomposition

$$\hat{\underline{V}}^T (\underline{D} + \rho \bar{u} \bar{u}^T) \hat{\underline{V}} = \hat{\underline{\Lambda}} \tag{B.79}$$

is computed, then (B.78) becomes

$$\hat{\underline{V}}^T \underline{P}^T \underline{U}^T \underline{T} \underline{U} \underline{P} \hat{\underline{V}} = \hat{\underline{\Lambda}}. \tag{B.80}$$

Comparing with (B.66) shows that the eigenvalues $\underline{\Lambda} = \hat{\underline{\Lambda}}$ and that the eigenvectors $\underline{V} = \underline{U} \underline{P} \hat{\underline{V}}$ (this last fact is true because each matrix is orthogonal so that the product is as well).

Notice that to compute such a decomposition, we need to compute the two smaller decompositions (B.70) and (B.71), as well as the larger decomposition (B.79). To compute the smaller decompositions, we apply this approach recursively. We can do so because $\underline{T}_1$ and $\underline{T}_2$ are both symmetric tridiagonal matrices, just like the original matrix $\underline{T}$. In practice, we perform this divide step until the matrices $\underline{T}_1$ and $\underline{T}_2$ become smaller than some fixed size[2]. For these small tridiagonal matrices, we compute their eigendecompositions directly using the QR iteration [25].

The larger decomposition is computed exploiting the structure of $\underline{D} + \rho \bar{u} \bar{u}^T$ where $d_1 < d_2 < \cdots < d_n$ and $\rho < 0$. In our application of the method, ρ is always negative because it comes from the off-diagonal of the pentadiagonal matrix $\underline{\tilde{S}} \underline{\tilde{S}}^T$ which is all negative. This is true at every step of the recursive process since only two entries on the diagonal of the submatrices, not off-diagonal, are modified when performing the divide step (see (B.68)). To

[2] I choose $m = 32$, although the choice depends on the implementation of the method and the hardware used to perform the computations. For example, [91] uses $m = 25$.

compute the eigenvalues of $\underline{D} + \rho\bar{u}\bar{u}^T$, we find values of λ such that

$$\det(\underline{D} + \rho\bar{u}\bar{u}^T - \lambda\underline{I}) = 0. \tag{B.81}$$

Suppose $\underline{D} - \lambda\underline{I}$ is invertible (meaning that $\lambda \neq d_i$ for any d_i), then

$$\begin{aligned}
\det(\underline{D} + \rho\bar{u}\bar{u}^T - \lambda\underline{I}) &= \det(\underline{D} - \lambda\underline{I} + \rho\bar{u}\bar{u}^T) && \text{(B.82)}\\
&= \det\left((\underline{D} - \lambda\underline{I})(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T)\right) && \text{(B.83)}\\
&= \det(\underline{D} - \lambda\underline{I})\det(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T). && \text{(B.84)}
\end{aligned}$$

Since $\underline{D} - \lambda\underline{I}$ is invertible, $\det(\underline{D} - \lambda\underline{I}) \neq 0$, which means that

$$\det(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T) = 0. \tag{B.85}$$

To express this determinant in a form amenable to root-finding techniques, notice that for any $\bar{x}$ and $\bar{y}$,

$$\begin{aligned}
\begin{bmatrix} \underline{I} & 0 \\ \bar{y}^T & 1 \end{bmatrix}\begin{bmatrix} \underline{I} + \bar{x}\bar{y}^T & \bar{x} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \underline{I} & 0 \\ -\bar{y}^T & 1 \end{bmatrix} &= \begin{bmatrix} \underline{I} + \bar{x}\bar{y}^T & \bar{x} \\ \bar{y}^T + \bar{y}^T\bar{x}\bar{y}^T & \bar{y}^T\bar{x} + 1 \end{bmatrix}\begin{bmatrix} \underline{I} & 0 \\ -\bar{y}^T & 1 \end{bmatrix} && \text{(B.86)}\\
&= \begin{bmatrix} \underline{I} & \bar{x} \\ 0 & \bar{y}^T\bar{x} + 1 \end{bmatrix}. && \text{(B.87)}
\end{aligned}$$

Taking the determinant of both sides, we note that

$$\begin{aligned}
\det\left(\begin{bmatrix} \underline{I} & 0 \\ \bar{y}^T & 1 \end{bmatrix}\begin{bmatrix} \underline{I} + \bar{x}\bar{y}^T & \bar{x} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \underline{I} & 0 \\ -\bar{y}^T & 1 \end{bmatrix}\right) &= \det\left(\begin{bmatrix} \underline{I} & \bar{x} \\ 0 & \bar{y}^T\bar{x} + 1 \end{bmatrix}\right) && \text{(B.88)}\\
\det\left(\begin{bmatrix} \underline{I} & 0 \\ \bar{y}^T & 1 \end{bmatrix}\right)\det\left(\begin{bmatrix} \underline{I} + \bar{x}\bar{y}^T & \bar{x} \\ 0 & 1 \end{bmatrix}\right)\det\left(\begin{bmatrix} \underline{I} & 0 \\ -\bar{y}^T & 1 \end{bmatrix}\right) &= \det\left(\begin{bmatrix} \underline{I} & \bar{x} \\ 0 & \bar{y}^T\bar{x} + 1 \end{bmatrix}\right) && \text{(B.89)}\\
\det\left(\begin{bmatrix} \underline{I} + \bar{x}\bar{y}^T & \bar{x} \\ 0 & 1 \end{bmatrix}\right) &= \bar{y}^T\bar{x} + 1 && \text{(B.90)}
\end{aligned}$$

where the last step comes from noting that the determinant of a triangular matrix is the product of its diagonal entries. Expanding the determinant along the bottom row gives $\det(\underline{I} + \bar{x}\bar{y}^T) = \bar{y}^T\bar{x} + 1$ which, if we let $\bar{x} = \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}$ and $\bar{y} = \bar{u}$, yields

$$\det(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T) = \bar{u}^T\rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u} + 1. \tag{B.91}$$

Thus, when $\underline{D} - \lambda \underline{I}$ is invertible, the eigenvalues of $\underline{D} + \rho \bar{u} \bar{u}^T$ satisfy

$$1 + \rho \bar{u}^T (\underline{D} - \lambda \underline{I})^{-1} \bar{u} = 0 \tag{B.92}$$

which can be written as

$$f(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} = 0. \tag{B.93}$$

This is called the secular equation.

Note that the derivative of the secular equation

$$f'(\lambda) = \rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2} \tag{B.94}$$

is always negative since $\rho < 0$ and u_i^2 and $(d_i - \lambda)^2$ are both positive. This means that $f(\lambda)$ is monotonic and decreasing (except at the poles d_i). For this to be possible, there must be a root $f(\lambda_k) = 0$ between each pair of poles so that

$$\lambda_1 < d_1 < \lambda_2 < d_2 < \lambda_2 < \cdots < \lambda_n < d_n. \tag{B.95}$$

This last statement gives all n roots of the secular equation only when all d_i are distinct (otherwise two terms in (B.93) coincide) and when all u_i are nonzero (otherwise a term in (B.93) vanishes). These two cases can be handled separately using deflation [25, 91]. For certain matrix classes, deflation leads to significant work savings in LAPACK [189]. An optimal implementation of divide and conquer should perform deflation[3]. All other eigenvalues are computed using a root-finding technique applied to the secular equation.

Once the eigenvalues are computed, the eigenvectors can be determined. Since each eigenvector must satisfy

$$(\underline{D} + \rho \bar{u} \bar{u}^T) \bar{v} = \lambda \bar{v}, \tag{B.96}$$

we observe that

$$(\underline{D} - \lambda \underline{I} + \rho \bar{u} \bar{u}^T) \bar{v} = 0 \tag{B.97}$$

$$(\underline{D} - \lambda \underline{I})(\underline{I} + \rho (\underline{D} - \lambda \underline{I})^{-1} \bar{u} \bar{u}^T) \bar{v} = 0. \tag{B.98}$$

Since $\underline{D} - \lambda \underline{I}$ is invertible,

$$(\underline{I} + \rho (\underline{D} - \lambda \underline{I})^{-1} \bar{u} \bar{u}^T) \bar{v} = 0 \tag{B.99}$$

[3] In my implementation, I have not used deflation since it is not necessary to obtain accurate eigendecompositions for the class of matrices encountered in this thesis. However, I suspect deflation can play an important role for these types of matrices and intend to use it in the future.

which gives

$$\bar{v} = -\rho(\underline{D} - \lambda \underline{I})^{-1}\bar{u}(\bar{u}^T\bar{v}) \tag{B.100}$$

$$= -\rho(\bar{u}^T\bar{v})(\underline{D} - \lambda \underline{I})^{-1}\bar{u}. \tag{B.101}$$

Since eigenvectors can be scaled arbitrarily, we ignore the scalar factors and use

$$\bar{v}_j = \frac{(\underline{D} - \lambda_j \underline{I})^{-1}\bar{u}}{\|(\underline{D} - \lambda_j \underline{I})^{-1}\bar{u}\|_2} \tag{B.102}$$

as an orthonormal eigenvector associated with eigenvalue λ_j. Notice that entry i in $\bar{v}_j$ is given by

$$(\bar{v}_j)_i = \frac{\dfrac{u_i}{d_i - \lambda_j}}{\sqrt{\displaystyle\sum_{k=1}^{n} \frac{u_k^2}{(d_k - \lambda_j)^2}}} \tag{B.103}$$

so that the eigenvector matrix

$$\hat{\underline{V}} = \begin{bmatrix} \bar{v}_1 & \bar{v}_2 & \cdots & \bar{v}_n \end{bmatrix} \tag{B.104}$$

can be written as

$$\hat{\underline{V}} = \mathrm{diag}(\bar{u})\hat{\underline{C}}\mathrm{diag}(\bar{n})^{-1} \tag{B.105}$$

where the entries of $\bar{n}$ are given by $\|(\underline{D} - \lambda_i \underline{I})^{-1}\bar{u}\|_2$ and $\hat{\underline{C}}$ is a Cauchy matrix with entries $1/(d_i - \lambda_j)$. Note that it costs $\mathcal{O}(n^2)$ operations to compute the entries of $\bar{n}$ directly (by evaluating the sums in the denominator of (B.103)). This cost can be reduced to $\mathcal{O}(n)$ operations using the fast multipole method (FMM), but we do not do this here since the method in Section B.3 requires the computation of $\mathcal{O}(n^2)$ elements anyway[4]. Matrix-vector multiplication with $\hat{\underline{V}}$ or its transpose can also be computed in $\mathcal{O}(n)$ operations when the FMM is used. Here we do exploit the FMM because we will need to multiply $\mathcal{O}(n)$ vectors by $\hat{\underline{V}}$, which results in a total cost of $\mathcal{O}(n^2)$ operations. We reserve discussion of the FMM in this context for Section B.5.

B.4.1 Numerical Considerations for Divide and Conquer

In practice, we must solve the secular equation using a numerical root-finding algorithm and compute the associated eigenvectors. To do so in a numerically stable way is crucial. To

[4]In practice, I have not performed enough tests to determine if it is worth using the FMM in this context.

compute eigenvalues, we solve the nonlinear secular equation

$$f(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} = 0 \tag{B.106}$$

using the method of Borges and Gragg [53]. The idea is, for each eigenvalue $d_k < \lambda_{k+1} < d_{k+1}$, to approximate f by a simpler rational function

$$g(\lambda) = c_1 + \frac{c_2}{d_k - \lambda} + \frac{c_3}{d_{k+1} - \lambda} \tag{B.107}$$

where the coefficients c_1, c_2, and c_3 are determined by choosing the function g, its derivative, and second derivative to match those of f at a point $\lambda_{k+1}^{(j)}$ and to use the root of this simpler function as an new approximation to λ_{k+1}, called $\lambda_{k+1}^{(j+1)}$. Borges and Gragg show that this iterative process converges monotonically to the eigenvalue λ_{k+1} so that the iteration can be stopped when the change in the iterate $\lambda_{k+1}^{(j)} - \lambda_{k+1}^{(j+1)}$ changes sign (at which point round-off effects dominate improvements to the iterate). The method converges cubically to the eigenvalue λ_{k+1} so that in practice only a small number of iterations (usually less than 5) are needed for convergence. The method is practical since the roots of g can be found by solving a quadratic equation. The new iterate $\lambda_{k+1}^{(j+1)}$ is chosen as the root that belongs to the interval (d_k, d_{k+1}). This method is similar to Newton's method since one interpretation of Newton's method is to find the root of $g(\lambda) = b_1 + b_2\lambda$, called $\lambda_{k+1}^{(j+1)}$, with b_1 and b_2 chosen such that g and its first derivative at $\lambda_{k+1}^{(j)}$ agree with those of f.

To use the method of Borges and Gragg, we first compute the coefficients c_1, c_2, and c_3 by solving the system of equations

$$g(\lambda_{k+1}^{(j)}) = f(\lambda_{k+1}^{(j)}), \tag{B.108}$$

$$g'(\lambda_{k+1}^{(j)}) = f'(\lambda_{k+1}^{(j)}), \tag{B.109}$$

$$g''(\lambda_{k+1}^{(j)}) = f''(\lambda_{k+1}^{(j)}). \tag{B.110}$$

To save on notation, we will simply write λ instead of $\lambda_{k+1}^{(j)}$ in the following derivation. This gives

$$c_1 + \frac{c_2}{d_k - \lambda} + \frac{c_3}{d_{k+1} - \lambda} = f(\lambda), \tag{B.111}$$

$$\frac{c_2}{(d_k - \lambda)^2} + \frac{c_3}{(d_{k+1} - \lambda)^2} = f'(\lambda), \tag{B.112}$$

$$\frac{c_2}{(d_k - \lambda)^3} + \frac{c_3}{(d_{k+1} - \lambda)^3} = \frac{1}{2} f''(\lambda). \tag{B.113}$$

Letting $\delta_k = 1/(d_k - \lambda)$ and $\delta_{k+1} = 1/(d_{k+1} - \lambda)$, we obtain the 3-by-3 linear system

$$\begin{bmatrix} 1 & \delta_k & \delta_{k+1} \\ 0 & \delta_k^2 & \delta_{k+1}^2 \\ 0 & \delta_k^3 & \delta_{k+1}^3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f(\lambda) \\ f'(\lambda) \\ \frac{1}{2} f''(\lambda) \end{bmatrix}. \tag{B.114}$$

In practice, it can dangerous to solve this system directly using a computer due to round-off errors. Instead, we solve the system by hand. First, we examine the precise form of the right hand side terms

$$f(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda}, \tag{B.115}$$

$$f'(\lambda) = \rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2}, \tag{B.116}$$

$$f''(\lambda) = 2\rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^3}, \tag{B.117}$$

which we write as

$$\begin{bmatrix} f(\lambda) \\ f'(\lambda) \\ \frac{1}{2} f''(\lambda) \end{bmatrix} = \begin{bmatrix} 1 + \rho \sum_{i=1}^{n} u_i^2 \delta_i \\ \rho \sum_{i=1}^{n} u_i^2 \delta_i^2 \\ \rho \sum_{i=1}^{n} u_i^2 \delta_i^3 \end{bmatrix} \tag{B.118}$$

where $\delta_i = 1/(d_i - \lambda)$. Performing one step of Gaussian elimination on (B.114), we obtain

$$\begin{bmatrix} 1 & \delta_k & \delta_{k+1} \\ 0 & \delta_k^2 & \delta_{k+1}^2 \\ 0 & 0 & \delta_{k+1}^3 - \delta_k \delta_{k+1}^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f(\lambda) \\ f'(\lambda) \\ \frac{1}{2} f''(\lambda) - \delta_k f'(\lambda) \end{bmatrix} \tag{B.119}$$

which shows that

$$c_3 = \frac{1}{\delta_{k+1}^2 (\delta_{k+1} - \delta_k)} \left[\frac{1}{2} f''(\lambda) - \delta_k f'(\lambda) \right] \tag{B.120}$$

$$= \frac{1}{\delta_{k+1}^2 (\delta_{k+1} - \delta_k)} \left[\rho \sum_{i=1}^{n} u_i^2 \delta_i^3 - \delta_k \rho \sum_{i=1}^{n} u_i^2 \delta_i^2 \right] \tag{B.121}$$

using (B.118). Combining the sums gives

$$c_3 = \frac{\rho}{\delta_{k+1}^2 (\delta_{k+1} - \delta_k)} \sum_{i=1}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \tag{B.122}$$

$$= \frac{\rho}{\delta_{k+1}^2 (\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \tag{B.123}$$

with the last line arising because

$$\delta_i - \delta_k = \frac{1}{d_i - \lambda} - \frac{1}{d_k - \lambda} \tag{B.124}$$

$$= \frac{d_k - d_i}{(d_i - \lambda)(d_k - \lambda)} \tag{B.125}$$

is zero when $i = k$. Note also that the $i = k + 1$ term can also be isolated from the sum to give

$$c_3 = \rho \left[u_{k+1}^2 + \frac{1}{\delta_{k+1}^2(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k, k+1}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right]. \tag{B.126}$$

Thus, to compute c_3, we compute

$$c_3 = \rho u_{k+1}^2 + \rho (d_{k+1} - \lambda)^2 \frac{(d_k - \lambda)(d_{k+1} - \lambda)}{d_k - d_{k+1}} \sum_{\substack{i=1 \\ i \neq k, k+1}}^{n} u_i^2 \frac{1}{(d_i - \lambda)^2} \frac{d_k - d_i}{(d_i - \lambda)(d_k - \lambda)} \tag{B.127}$$

$$= \rho u_{k+1}^2 + \rho \frac{(d_{k+1} - \lambda)^3}{d_k - d_{k+1}} \sum_{\substack{i=1 \\ i \neq k, k+1}}^{n} u_i^2 \frac{d_k - d_i}{(d_i - \lambda)^3} \tag{B.128}$$

where we have replaced the δ terms with their original definitions.

Similarly, from (B.119) together with (B.118) and (B.123), we find that

$$c_2 = \frac{1}{\delta_k^2} \left[f'(\lambda) - \delta_{k+1}^2 c_3 \right] \tag{B.129}$$

$$= \frac{1}{\delta_k^2} \left[\rho \sum_{i=1}^{n} u_i^2 \delta_i^2 - \delta_{k+1}^2 \frac{\rho}{\delta_{k+1}^2(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right] \tag{B.130}$$

$$= \frac{\rho}{\delta_k^2} \left[\sum_{i=1}^{n} u_i^2 \delta_i^2 - \frac{1}{(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right] \tag{B.131}$$

$$= \frac{\rho}{\delta_k^2} \left[u_k^2 \delta_k^2 + \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 - \frac{1}{(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right]. \tag{B.132}$$

Grouping the summation terms together yields

$$c_2 = \frac{\rho}{\delta_k^2} \left[u_k^2 \delta_k^2 + \frac{1}{(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k}}^{n} u_i^2 \delta_i^2 (\delta_{k+1} - \delta_i) \right]. \tag{B.133}$$

Note that $\delta_{k+1} - \delta_i = 0$ when $i = k+1$ so that

$$c_2 = \rho \left[u_k^2 + \frac{1}{\delta_k^2(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i^2 (\delta_{k+1} - \delta_i) \right]. \tag{B.134}$$

Replacing the δ terms with their definitions gives

$$c_2 = \rho u_k^2 + \rho (d_k - \lambda)^2 \frac{(d_{k+1} - \lambda)(d_k - \lambda)}{d_k - d_{k+1}} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \frac{1}{(d_i - \lambda)^2} \frac{d_i - d_{k+1}}{(d_{k+1} - \lambda)(d_i - \lambda)} \tag{B.135}$$

$$= \rho u_k^2 + \rho \frac{(d_k - \lambda)^3}{d_k - d_{k+1}} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \frac{d_i - d_{k+1}}{(d_i - \lambda)^3}. \tag{B.136}$$

Finally, (B.119) together with (B.118), (B.134), and (B.126) yields

$$c_1 = 1 + \rho \sum_{i=1}^{n} u_i^2 \delta_i - \delta_k \rho \left[u_k^2 + \frac{1}{\delta_k^2(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i^2 (\delta_{k+1} - \delta_i) \right]$$
$$- \delta_{k+1} \rho \left[u_{k+1}^2 + \frac{1}{\delta_{k+1}^2(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right]. \tag{B.137}$$

Canceling the $\rho \delta_k u_k^2$ and $\rho \delta_{k+1} u_{k+1}^2$ terms gives

$$c_1 = 1 + \rho \left[\sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i - \frac{1}{\delta_k(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i^2 (\delta_{k+1} - \delta_i) \right.$$
$$\left. - \frac{1}{\delta_{k+1}(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i^2 (\delta_i - \delta_k) \right] \tag{B.138}$$

and collecting summations together yields

$$c_1 = 1 + \rho \frac{1}{\delta_k \delta_{k+1}(\delta_{k+1} - \delta_k)} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i \left[(\delta_{k+1} - \delta_k) \delta_k \delta_{k+1} - \delta_i \delta_{k+1} (\delta_{k+1} - \delta_i) - \delta_i \delta_k (\delta_i - \delta_k) \right]. \tag{B.139}$$

It is possible to factor the term in square brackets since

$$(\delta_{k+1} - \delta_k) \delta_k \delta_{k+1} - \delta_i \delta_{k+1} (\delta_{k+1} - \delta_i) - \delta_i \delta_k (\delta_i - \delta_k) = (\delta_{k+1} - \delta_k)(\delta_i - \delta_k)(\delta_i - \delta_{k+1}). \tag{B.140}$$

This means that

$$c_1 = 1 + \rho \frac{1}{\delta_k \delta_{k+1}} \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \delta_i (\delta_i - \delta_k)(\delta_i - \delta_{k+1}) \tag{B.141}$$

which gives

$$c_1 = 1 + \rho (d_k - \lambda)(d_{k+1} - \lambda) \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \frac{1}{(d_i - \lambda)} \frac{d_k - d_i}{(d_i - \lambda)(d_k - \lambda)} \frac{d_{k+1} - d_i}{(d_i - \lambda)(d_{k+1} - \lambda)} \tag{B.142}$$

$$= 1 + \rho \sum_{\substack{i=1 \\ i \neq k,k+1}}^{n} u_i^2 \frac{(d_k - d_i)(d_{k+1} - d_i)}{(d_i - \lambda)^3} \tag{B.143}$$

when the δ terms are replaced with their definitions.

When computing c_1, c_2, and c_3 using (B.143), (B.136), and (B.128) respectively, each summation should be split into two sums, one for $i < k$ and another for $i > k + 1$. This is done to avoid catastrophic cancellation when adding a mix of positive and negative numbers together. For example, in (B.143), if $i < k$ and $\lambda = \lambda_{k+1}^{(j)}$, then

$$d_k - d_i > 0, \tag{B.144}$$

$$d_{k+1} - d_i > 0, \tag{B.145}$$

$$d_i - \lambda < 0, \tag{B.146}$$

so that each term

$$u_i^2 \frac{(d_k - d_i)(d_{k+1} - d_i)}{(d_i - \lambda)^3} \tag{B.147}$$

with $i < k$ is negative, whereas when $i > k + 1$,

$$d_k - d_i < 0, \tag{B.148}$$

$$d_{k+1} - d_i < 0, \tag{B.149}$$

$$d_i - \lambda > 0, \tag{B.150}$$

so that each term (B.147) is positive.

Once the coefficients c_1, c_2, and c_3 are computed, we solve (B.107) for the next iterate $\lambda_{k+1}^{(j+1)}$. However, in practice, solving for $\lambda_{k+1}^{(j+1)}$ directly can lead to catastrophic cancellation in one of the differences $d_k - \lambda_{k+1}^{(j+1)}$ or $d_{k+1} - \lambda_{k+1}^{(j+1)}$. To avoid such a scenario, we begin the iterative process with initial iterate $\lambda_{k+1}^{(0)} = (d_k + d_{k+1})/2$, which is the midpoint of the interval (d_k, d_{k+1}). Since the Borges and Gragg method converges monotonically to the root, if the next iterate $\lambda_{k+1}^{(1)}$ is smaller than the initial iterate, serious cancellation can only occur

with the difference $d_k - \lambda_{k+1}^{(j)}$ and we let $\tau_-^{(j)} = d_k - \lambda_{k+1}^{(j)}$ and only compute with $\tau_-^{(j)}$ rather than the difference. Otherwise, we let $\tau_+^{(j)} = d_{k+1} - \lambda_{k+1}^{(j)}$ and only compute with $\tau_+^{(j)}$. If the eigenvalue is needed for later computations, it can be recovered as long as we store the final $\tau_\pm$ and the choice of pole d_k or d_{k+1} (indicated by $\pm$). Note that the first eigenvalue λ_1 is not bracketed between two poles. In that case, we use d_1 and d_2 to form (B.107) and always choose d_1 as the pole closest to λ_1 when choosing τ. A good initial guess in this special case is zero since the smallest eigenvalue for the tridiagonal matrix we consider is very close to zero (this would not be the case for a general symmetric tridiagonal matrix). Unfortunately, in this case, convergence is not guaranteed to be monotonic so an alternative stopping criterion is needed. Usual stopping criteria for Newton's method can be applied.

When we choose $\tau_-^{(j)} = d_k - \lambda_{k+1}^{(j)}$, and we wish to find the zeros of (B.107), we must solve

$$c_1 + \frac{c_2}{\tau_-^{(j+1)}} + \frac{c_3}{d_{k+1} - (d_k - \tau_-^{(j+1)})} = 0. \tag{B.151}$$

Letting $\Delta = d_{k+1} - d_k$, we have

$$c_1 + \frac{c_2}{\tau_-^{(j+1)}} + \frac{c_3}{\Delta + \tau_-^{(j+1)}} = 0 \tag{B.152}$$

$$\tau_-^{(j+1)}(\Delta + \tau_-^{(j+1)}) \left[c_1 + \frac{c_2}{\tau_-^{(j+1)}} + \frac{c_3}{\Delta + \tau_-^{(j+1)}} \right] = 0 \tag{B.153}$$

$$\tau_-^{(j+1)}(\Delta + \tau_-^{(j+1)})c_1 + (\Delta + \tau_-^{(j+1)})c_2 + \tau_-^{(j+1)}c_3 = 0 \tag{B.154}$$

$$c_1(\tau_-^{(j+1)})^2 + (\Delta c_1 + c_2 + c_3)\tau_-^{(j+1)} + \Delta c_2 = 0. \tag{B.155}$$

This last equation is a quadratic equation in $\tau_-^{(j+1)}$ with coefficients

$$a = c_1, \tag{B.156}$$

$$b = \Delta c_1 + c_2 + c_3, \tag{B.157}$$

$$c = \Delta c_2. \tag{B.158}$$

Similarly, if we choose $\tau_+^{(j)} = d_{k+1} - \lambda_{k+1}^{(j)}$ instead, then (B.107) becomes

$$c_1 + \frac{c_2}{d_k - (d_{k+1} - \tau_+^{(j+1)})} + \frac{c_3}{\tau_+^{(j+1)}} = 0. \tag{B.159}$$

Using the same definition for Δ, we obtain

$$c_1 + \frac{c_2}{\tau_+^{(j+1)} - \Delta} + \frac{c_3}{\tau_+^{(j+1)}} = 0 \tag{B.160}$$

which yields the quadratic equation

$$c_1(\tau_+^{(j+1)})^2 + (-\Delta c_1 + c_2 + c_3)\tau_+^{(j+1)} - \Delta c_3 = 0 \tag{B.161}$$

with coefficients

$$a = c_1, \tag{B.162}$$

$$b = -\Delta c_1 + c_2 + c_3, \tag{B.163}$$

$$c = -\Delta c_3. \tag{B.164}$$

In either case, one must solve for the roots of a quadratic polynomial. Solving such an equation numerically should be handled with care, again to avoid catastrophic cancellation. In particular, one can start by normalizing the coefficients a, b, and c by dividing by $\max(\{|a|, |b|, |c|\})$. If $a = 0$, then the quadratic is linear and the single root is $\tau = -c/b$. Otherwise, when $b \geq 0$, the roots are

$$\tau_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}, \qquad \tau_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \tag{B.165}$$

and when $b < 0$, the roots are

$$\tau_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \qquad \tau_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}. \tag{B.166}$$

This choice avoids cancellation when combining b with the discriminant. Once the two roots are calculated, we determine which of the two causes $\lambda_{k+1}^{(j+1)}$ to belong to the interval (d_k, d_{k+1}) and set $\tau_\pm^{(j+1)}$ accordingly.

When computing eigenvalue λ_{k+1} iteratively, the most costly step of each iteration is the computation of c_1, c_2, and c_3. Each coefficient requires $\mathcal{O}(n)$ operations to compute their respective sums. All other computations (solving the quadratic equation) require $\mathcal{O}(1)$ operations. There are only a small number of iterations (almost always less than five) required to compute each eigenvalue. This means that $\mathcal{O}(n^2)$ operations are required to compute all n eigenvalues. This can be reduced to $\mathcal{O}(n)$ operations using the FMM if the sums to compute c_1, c_2, and c_3 are performed for all eigenvalues simultaneously [55] (although this is not necessary).

This leaves the computation of the associated eigenvectors. Unfortunately, computing (B.102) directly can lead to a loss of orthogonality. The authors of [54] explain how to compute the eigenvectors in a way that preserves orthogonality. The key idea is to find a

new vector $\bar{u}_{\text{new}}$ such that the computed eigenvalues are the exact eigenvalues of the matrix $\underline{D} + \rho\bar{u}_{\text{new}}\bar{u}_{\text{new}}^T$ since they are not the exact eigenvalues of the original matrix $\underline{D} + \rho\bar{u}\bar{u}^T$ due to the limited precision of our computations. Then the eigenvectors

$$\bar{v}_j = \frac{(\underline{D} - \lambda_j \underline{I})^{-1}\bar{u}_{\text{new}}}{\|(\underline{D} - \lambda_j \underline{I})^{-1}\bar{u}_{\text{new}}\|_2} \tag{B.167}$$

will be the exact eigenvectors of the new problem. As long as the new problem is close to the original problem (which [54] proves is true), then the eigenvectors computed in this way will be orthogonal to high relative accuracy for the original problem. To compute $\bar{u}_{\text{new}}$, we use the fact (derived earlier in (B.84)) that

$$\det(\underline{D} + \rho\bar{u}\bar{u}^T - \lambda\underline{I}) = \det(\underline{D} - \lambda\underline{I})\det(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T). \tag{B.168}$$

Since

$$\det(\underline{D} + \rho\bar{u}\bar{u}^T - \lambda\underline{I}) = \prod_{i=1}^{n}(\lambda_i - \lambda) \tag{B.169}$$

where λ_i are the eigenvalues of $\underline{D} + \rho\bar{u}\bar{u}^T$,

$$\det(\underline{D} - \lambda\underline{I}) = \prod_{i=1}^{n}(d_i - \lambda) \tag{B.170}$$

since $\underline{D}$ is diagonal, and

$$\det(\underline{I} + \rho(\underline{D} - \lambda\underline{I})^{-1}\bar{u}\bar{u}^T) = 1 + \rho\sum_{i=1}^{n}\frac{u_i^2}{d_i - \lambda} \tag{B.171}$$

by our earlier derivation of the secular equation (B.93), we have that

$$\prod_{i=1}^{n}(\lambda_i - \lambda) = \prod_{i=1}^{n}(d_i - \lambda)\left[1 + \rho\sum_{i=1}^{n}\frac{u_i^2}{d_i - \lambda}\right]. \tag{B.172}$$

Extracting the $i = k$ term from the sum gives

$$\prod_{i=1}^{n}(\lambda_i - \lambda) = \prod_{i=1}^{n}(d_i - \lambda)\left[1 + \rho\sum_{\substack{i=1\\i\neq k}}^{n}\frac{u_i^2}{d_i - \lambda}\right] + \rho u_k^2\prod_{\substack{i=1\\i\neq k}}^{n}(d_i - \lambda). \tag{B.173}$$

If we let $\lambda = d_k$, then $\prod_{i=1}^{n}(d_i - d_k) = 0$ and

$$\prod_{i=1}^{n}(\lambda_i - d_k) = \rho u_k^2\prod_{\substack{i=1\\i\neq k}}^{n}(d_i - d_k) \tag{B.174}$$

which implies that

$$u_k^2 = \frac{1}{\rho} \frac{\prod_{i=1}^{n} (\lambda_i - d_k)}{\prod_{\substack{i=1 \\ i \neq k}}^{n} (d_i - d_k)} \tag{B.175}$$

$$= \frac{\lambda_k - d_k}{\rho} \prod_{\substack{i=1 \\ i \neq k}}^{n} \frac{\lambda_i - d_k}{d_i - d_k}. \tag{B.176}$$

By (B.95), $\lambda_i - d_k < 0$ if $i < k$ and $\lambda_i - d_k > 0$ if $i > k$. Similarly, $d_i - d_k < 0$ if $i < k$ and $d_i - d_k > 0$ if $i > k$. Since $\rho < 0$, this implies that the right hand side of (B.176) is positive, so that

$$u_k = \sqrt{\frac{\lambda_k - d_k}{\rho} \prod_{\substack{i=1 \\ i \neq k}}^{n} \frac{\lambda_i - d_k}{d_i - d_k}}. \tag{B.177}$$

Thus, to compute the entries of $\bar{u}_{\text{new}}$, we use this expression with λ_i given by the computed eigenvalues.

It costs $\mathcal{O}(n)$ operations to compute the product in (B.177) and there are n such entries in the vector $\bar{u}_{\text{new}}$ so that the total cost to compute $\bar{u}_{\text{new}}$ is $\mathcal{O}(n^2)$ operations. It is important to use the computed differences τ rather than $d - \lambda$ wherever appropriate in doing so. Since one begins with $\bar{u}$, it is trivial to change the signs of entries in $\bar{u}_{\text{new}}$ to match those in $\bar{u}$. It is possible, but not necessary, to accelerate the computation of $\bar{u}_{\text{new}}$ using the FMM. As mentioned at the end of Section B.4, the multiplication of these eigenvectors with another vector can be performed using the FMM. For this reason, we do not explicitly compute the eigenvectors, but rather, only store the vector $\bar{u}_{\text{new}}$, the normalization vector $\bar{n}$ (whose computation has been previously discussed), and the information regarding eigenvalues (which are not stored directly, but through their offsets $\bar{\tau}$ from the vector $\bar{d}$ and a vector of flags indicating which entry in $\bar{d}$ to use as pole for each entry in $\bar{\tau}$). This has the added benefit of reducing the storage requirements for eigenvectors from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ double precision floating point numbers.

B.4.2 Finalizing the Divide and Conquer Algorithm

Now that we have seen how to compute the eigenvalues and eigenvectors of a diagonal plus rank 1 matrix, we can show how to use the divide and conquer algorithm in practice, and consider its computational complexity. Given the symmetric tridiagonal matrix $\underline{T}$ of size n, we choose a maximum block size n_b for which we are willing to perform a direct eigenvalue decomposition (for example, $n_b = 32$). We divide n by two repeatedly until the result is smaller than n_b. The number of times we have divided will be referred to as the maximum

level of recursion $l_{\max}$. This means that

$$n_b > \frac{n}{2^{l_{\max}}} \tag{B.178}$$

or that

$$l_{\max} = \left\lceil \frac{\log(n) - \log(n_b)}{\log(2)} \right\rceil \tag{B.179}$$

where $\lceil \cdot \rceil$ denotes the ceiling function. We perform the divide step from Section B.4 $l_{\max}$ times which guarantees that the $2^{l_{\max}}$ remaining tridiagonal matrices all have dimension smaller than n_b. If n is even, it is possible to subdivide the matrix into two matrices of equal size, but otherwise, we can assign either block to be one row larger. Even when n is even, once it has been subdivided, its submatrices may have odd sizes and so a systematic way to assign block sizes is needed[5]. This process leads to submatrices at the maximum level that have almost equal sizes (the even/odd issue means that they do not all have exactly the same size). The size is between $n_b/2$ and n_b and depends on the value of n.

Once the blocks have been determined at the maximum level, we compute the $2^{l_{\max}}$ small eigendecompositions directly using the QR iteration and store the eigenvalues and eigenvectors. We then proceed to determine the eigenvalue and eigenvector data associated with pairs of these matrices via their diagonal plus rank 1 connection. To do this, we multiply the unit vectors $\bar{e}_m$ and $\bar{e}_1$ with the eigenvectors to form $\hat{u}$. Then we sort the eigenvalues of the two matrices (taken together) in ascending order, storing the permutation required to do so (it is stored as a vector of integers rather than a matrix). The permutation is applied to $\hat{u}$ to obtain $\bar{u}$ and the eigenvalue and eigenvector data for the diagonal plus rank 1 matrix are computed via the secular equation. This is performed $2^{l_{\max}-1}$ times. The process is repeated at the next level $2^{l_{\max}-2}$ times, and so on, until we reach the root level which is the full size of the matrix. At the intermediate levels j, to compute $\bar{u}^{(j)}$, we need to multiply the unit vectors not just by the eigenvector matrices at the maximum level, but all intermediary eigenvector matrices as well. This is a type of hierarchical matrix-vector product

$$\bar{u}^{(j)} = \left(\underline{V}^{(\text{fine})}\underline{P}^{(l_{\max})}\underline{V}^{(l_{\max})}\underline{P}^{(l_{\max}-1)}\underline{V}^{(l_{\max}-1)}\cdots\underline{P}^{(j-1)}\underline{V}^{(j-1)}\right)^T \bar{e}^{(j)} \tag{B.180}$$

where each $\underline{V}^{(j)}$ and $\underline{P}^{(j)}$ is block diagonal and $\bar{e}^{(j)}$ represents the particular combination of unit vectors and their corresponding value $\rho^{(j)}$. The blocks increase in size until level 1 where they are the full size of the matrix. The first matrix $\underline{V}^{(\text{fine})}$ is different from the other intermediate matrices in that it is comprised of the directly computed eigenvectors at the maximum level, whereas all other matrices are computed via the diagonal plus rank 1 update

[5] In practice, at each subdivision step, I choose the first submatrix to be larger.

method. The final eigenvector matrix for the symmetric tridiagonal matrix $\underline{T}$ is given by

$$\underline{V} = \underline{V}^{(\text{fine})} \underline{P}^{(l_{\max})} \underline{V}^{(l_{\max})} \underline{P}^{(l_{\max}-1)} \underline{V}^{(l_{\max}-1)} \cdots \underline{P}^{(1)} \underline{V}^{(1)} \tag{B.181}$$

but is never explicitly computed. Instead, the blocks in $\underline{V}^{(\text{fine})}$ are stored explicitly (but are all of size smaller than n_b), while each permutation matrix is stored as an integer array, and each block in $\underline{V}^{(j)}$ is stored as a set of vectors $\bar{u}$, $\bar{n}$, $\bar{d}$, $\bar{\tau}$ with a vector of flags indicating which pole in $\bar{d}$ to use to recover the intermediate eigenvalues $\bar{\lambda}$ from $\bar{d}$ and $\bar{\tau}$. This data can be stored in a binary tree data structure to facilitate retrieval. Whenever the eigenvector matrix is required for a matrix-vector product, the product (B.181) is multiplied against a vector in sequence.

To determine the computational complexity of the algorithm, we note that computing the eigenvalues and eigenvectors at the maximum level requires $\mathcal{O}(2^{l_{\max}} n_b^3)$ operations. Since $2^{l_{\max}} n_b \approx n$, this requires $\mathcal{O}(n_b^2 n)$ operations, which is linear in n. We then take into account the cost of computing the eigenvalues and eigenvectors for each diagonal plus rank 1 subproblem. We have seen that each of these problems requires a quadratic number of operations in the size of the associated matrix. This means that $\mathcal{O}(n^2)$ operations are required at the root level, $\mathcal{O}((\frac{n}{2})^2)$ operations are required at the next level (performed twice), and so on. Altogether, this gives an approximate number of operations

$$n^2 + \left(\frac{n}{2}\right)^2 \cdot 2 + \left(\frac{n}{2^2}\right)^2 \cdot 2^2 + \cdots + \left(\frac{n}{2^{l_{\max}}}\right)^2 \cdot 2^{l_{\max}} = n^2 \sum_{i=0}^{l_{\max}-1} \left(\frac{1}{2}\right)^i \tag{B.182}$$

$$= n^2 \frac{1 - (\frac{1}{2})^{l_{\max}}}{1 - (\frac{1}{2})} \tag{B.183}$$

$$= n^2 (2 - 2^{1-l_{\max}}) \tag{B.184}$$

where we have evaluated the geometric sum. Note that this is $\mathcal{O}(n^2)$ operations since $2 - 2^{1-l_{\max}}$ tends towards 2 as $l_{\max}$ increases.

We also need to determine how much each matrix-vector product with (B.181) costs. Assuming that each matrix $\underline{V}^{(j)}$ is multiplied with a vector using the FMM, the cost is linear in the dimension of the matrix. This means that to multiply by $\underline{V}^{(1)}$ requires $\mathcal{O}(n)$ operations, to multiply by $\underline{V}^{(2)}$ costs $\mathcal{O}(\frac{n}{2})$ operations twice, and so on. The approximate total number of operations is then

$$n + \frac{n}{2} \cdot 2 + \frac{n}{2^2} \cdot 2^2 + \cdots + \frac{n}{2^{l_{\max}}} \cdot 2^{l_{\max}} = n l_{\max}. \tag{B.185}$$

Since $l_{\max} \approx \log_2 n$, this means that to multiply by the first $l_{\max}$ matrices $\underline{V}^{(j)}$ requires

$\mathcal{O}(n \log n)$ operations. In addition, there is the additional cost of multiplying by $\underline{V}^{(\text{fine})}$. This requires $\mathcal{O}(2^{l_{\max}} n_b^2)$ operations since there are $2^{l_{\max}}$ matrices of size n_b at the maximum level and direct matrix-vector multiplication is quadratic in the size of these matrices. Since $2^{l_{\max}} n_b \approx n$, this means that $\mathcal{O}(n_b n)$ operations are required for this final step of multiplication. Thus the total cost of a matrix-vector multiplication with (B.181) is $\mathcal{O}(n \log n)$ operations.

We have neglected to comment on the cost of computing the intermediate $\bar{u}$ vectors. They are computed in a similar fashion as described for the matrix-vector product. Since there are

$$\sum_{i=0}^{l_{\max}-1} 2^i = 2^{l_{\max}} - 1 \tag{B.186}$$

such intermediate vectors, and $2^{l_{\max}} \approx n/n_b$, the number of operations to compute these products can be no greater than $\mathcal{O}(n^2 \log n)$. It is possible to improve upon this result but we do not bother here because in the partial diagonalization algorithm of Section B.3, we compute the matrix-vector product with (B.181) n times, which means the total number of operations for that algorithm is $\mathcal{O}(n^2 \log n)$. This matches the cost of computing the eigendecomposition so that we have described a near linear (there are n^2 unknowns to compute in Section B.3) method for partial diagonalization, as desired.

B.5 The Fast Multipole Method

In order to achieve the near linear computational complexity of the previous section, we must be able to perform matrix-vector products $\underline{\hat{C}}\bar{x}$ with Cauchy matrices $\underline{\hat{C}}$ in $\mathcal{O}(n)$ operations where n is the length of vector $\bar{x}$. The entries of $\underline{\hat{C}}$ are given by

$$\underline{\hat{C}}_{ij} = \frac{1}{d_i - \lambda_j} \tag{B.187}$$

and related to the intermediate diagonal plus rank 1 matrices that arise in the divide and conquer algorithm. Recall, from Section B.4, that the values d_i (which we will call target points) arise from the diagonal matrix $\underline{D}$ and the values λ_j (which we will call source points) arise from the eigenvalues of $\underline{D} + \rho \bar{u} \bar{u}^T$ at each step of the divide and conquer method. In this section, we will use the FMM to perform the matrix-vector multiplication in $\mathcal{O}(n)$ operations [58, 59, 60, 61]. The FMM interprets the matrix-vector product

$$\bar{y} = \underline{\hat{C}}\bar{x} \tag{B.188}$$

as point evaluations of a linear combination of kernel functions. That is, by defining the kernel function

$$K(d, \lambda_j) = \frac{1}{d - \lambda_j}, \tag{B.189}$$

the entries y_i of the matrix-vector product (B.188) can be written as

$$y_i = \sum_{j=1}^{n} x_j \frac{1}{d_i - \lambda_j} \tag{B.190}$$

$$= \sum_{j=1}^{n} x_j K(d_i, \lambda_j) \tag{B.191}$$

where x_j are considered source strengths or coefficients in a linear combination of kernel functions due to the source locations λ_j evaluated at the target point d_i. Without loss of generality, we will assume that all d_i and λ_j belong to the interval $(0, 1)$. This is the case for the Dirichlet problem, but not for the Neumann problem of Section B.2. In the Neumann case, the problem can be scaled and treated as though such conditions hold.

The FMM exploits the fact that the kernel function has simple local Taylor series expansions away from the source singularities and simple multipole expansions centered about those same singularities. Rather than work with the kernel directly, these expansions are used wherever they converge rapidly. The key to the FMM is to partition the interval where the source and target points reside using a binary tree structure so that a partition (possibly non-uniform) of the original interval is obtained where each subinterval contains roughly the same number of sources and targets. Then multipole expansions are computed for each subinterval. These expansions are centered at the midpoint of each subinterval, but only converge rapidly away from their respective subintervals. Other subintervals that are sufficiently far away are said to be well-separated. The FMM aggregates multipole expansions of two subintervals in the tree to form a multipole expansion centered at the midpoint of their parent subinterval. This is performed for the whole tree in an upward pass. Then local Taylor expansions are computed from the multipole expansions for subintervals that are well separated on each level. The local expansions for coarse levels of the tree are used to create local expansions on finer levels in a downward pass. The final evaluation of (B.191) is performed by direct evaluation for sources belonging to each subinterval and its two neighbor subintervals, then adding the contribution of the local expansion to account for sources belonging to all other subintervals. By performing this process carefully (there are certain subintervals in a non-uniform partition that must be treated differently), this process can be performed in $\mathcal{O}(n)$ operations for a given tolerance specified by the user. The tolerance changes how many source points should be contained in each subinterval and how many

terms should be used in each multipole and local Taylor expansion. In addition, the tolerance changes the constant hidden by the $\mathcal{O}$ notation, but not the fact that the matrix-vector product can be performed with linear computational complexity.

In order to describe the FMM, we first need to see how multipole and local Taylor expansions are computed. We begin with the multipole expansion of $K(d, \lambda_j)$ centered at point $\lambda_\star$. We rewrite

$$\frac{1}{d-\lambda_j} = \frac{1}{d-\lambda_\star+\lambda_\star-\lambda_j} \tag{B.192}$$

$$= \frac{1}{(d-\lambda_\star)\left[1+\frac{\lambda_\star-\lambda_j}{d-\lambda_\star}\right]} \tag{B.193}$$

$$= \frac{1}{d-\lambda_\star}\frac{1}{1-\frac{\lambda_j-\lambda_\star}{d-\lambda_\star}} \tag{B.194}$$

and assume that

$$\left|\frac{\lambda_j-\lambda_\star}{d-\lambda_\star}\right| < 1 \tag{B.195}$$

so that, by Taylor series, we have

$$\frac{1}{d-\lambda_j} = \frac{1}{d-\lambda_\star}\sum_{m=0}^{\infty}\left(\frac{\lambda_j-\lambda_\star}{d-\lambda_\star}\right)^m \tag{B.196}$$

$$= \sum_{m=0}^{\infty}(\lambda_j-\lambda_\star)^m\frac{1}{(d-\lambda_\star)^{m+1}}. \tag{B.197}$$

This Taylor series converges when $|\lambda_j-\lambda_\star| < |d-\lambda_\star|$, meaning that d must be farther away from the center $\lambda_\star$ than the source λ_j is from the center. Next, we use this type of multipole expansion in (B.191) to obtain

$$y_i = \sum_{j=1}^{n} x_j\left[\sum_{m=0}^{\infty}(\lambda_j-\lambda_\star)^m\frac{1}{(d_i-\lambda_\star)^{m+1}}\right]. \tag{B.198}$$

Notice how the multipole expansion splits into terms dependent on sources x_j and source locations λ_j and terms that are independent of both. By rearranging the sums, we can exploit this fact:

$$y_i = \sum_{m=0}^{\infty}\left[\sum_{j=1}^{n} x_j(\lambda_j-\lambda_\star)^m\right]\frac{1}{(d_i-\lambda_\star)^{m+1}}. \tag{B.199}$$

If a large number of terms indexed by m are needed for this expression to be accurate, then the expansion is not efficient. We can determine how many terms p are needed by performing

an error analysis. We do so by noting that

$$\underbrace{\left|\sum_{j=1}^{n} x_j \frac{1}{d-\lambda_j} - \sum_{m=0}^{p-1}\left[\sum_{j=1}^{n} x_j(\lambda_j-\lambda_\star)^m\right]\frac{1}{(d-\lambda_\star)^{m+1}}\right|}_{\epsilon} = \left|\sum_{m=p}^{\infty}\left[\sum_{j=1}^{n} x_j(\lambda_j-\lambda_\star)^m\right]\frac{1}{(d-\lambda_\star)^{m+1}}\right| \tag{B.200}$$

where ϵ is the absolute error incurred by using only the first p terms in the multipole expansion. Using the triangle inequality, we have

$$\epsilon \leq \sum_{m=p}^{\infty}\sum_{j=1}^{n} |x_j|\,|\lambda_j-\lambda_\star|^m \frac{1}{|d-\lambda_\star|^{m+1}}. \tag{B.201}$$

If $|\lambda_j-\lambda_\star| \leq r$ and $|d-\lambda_\star| \geq \alpha r$ with $\alpha \geq 1$ and $r > 0$, then

$$\frac{1}{|d-\lambda_\star|} \leq \frac{1}{\alpha r}. \tag{B.202}$$

Using these inequalities in the error estimate yields

$$\epsilon \leq \sum_{m=p}^{\infty}\sum_{j=1}^{n} |x_j|\, r^m \frac{1}{(\alpha r)^{m+1}} \tag{B.203}$$

$$\leq \sum_{m=p}^{\infty}\sum_{j=1}^{n} |x_j| \frac{1}{r\alpha^{m+1}}. \tag{B.204}$$

Factoring terms dependent on j, and rearranging the remaining sum, we have

$$\epsilon \leq \underbrace{\sum_{j=1}^{n} |x_j|}_{\|\bar{x}\|_1} \frac{1}{\alpha r}\sum_{m=p}^{\infty}\left(\frac{1}{\alpha}\right)^m \tag{B.205}$$

$$\leq \|\bar{x}\|_1 \frac{1}{\alpha r}\left[\sum_{m=0}^{\infty}\left(\frac{1}{\alpha}\right)^m - \sum_{m=0}^{p-1}\left(\frac{1}{\alpha}\right)^m\right]. \tag{B.206}$$

Since $\alpha \geq 1$, these geometric sums yield

$$\epsilon \leq \|\bar{x}\|_1 \frac{1}{\alpha r}\left[\frac{1}{1-\alpha^{-1}} - \frac{1-\alpha^{-p}}{1-\alpha^{-1}}\right] \tag{B.207}$$

$$\leq \|\bar{x}\|_1 \frac{1}{r}\left[\frac{\alpha^{-p}}{\alpha-1}\right]. \tag{B.208}$$

In the FMM, $\alpha = 3$ when r corresponds to the radius of a subinterval. This is the well separated condition, meaning that multipole expansions are only used on subintervals that

are at least one neighbor away from the subinterval containing the center $\lambda_\star$. In such a case,

$$\epsilon \leq \|\bar{x}\|_1 \frac{1}{2r} \frac{1}{3^p} \tag{B.209}$$

and p is chosen to make ϵ as small as desired.

We cannot use this bound to determine p because it depends on r, meaning that the absolute error increases with decreasing size of subinterval r. However, the relative error does not. This is because

$$\left| \sum_{j=1}^{n} x_j \frac{1}{d - \lambda_j} \right| \leq \sum_{j=1}^{n} |x_j| \frac{1}{|d - \lambda_j|} \tag{B.210}$$

and $|d - \lambda_j| \geq 2r$. This means that

$$\left| \sum_{j=1}^{n} x_j \frac{1}{d - \lambda_j} \right| \leq \frac{\sum_{j=1}^{n} |x_j|}{2r} \tag{B.211}$$

$$\leq \frac{\|\bar{x}\|_1}{2r}. \tag{B.212}$$

This is an upper bound, which means that dividing (B.209) by $\|\bar{x}\|_1/(2r)$ does not yield an upper bound on the relative error. However, in practice, it is typically observed that the upper bound (B.212) is attained so that dividing (B.209) by (B.212) yields the useful approximation

$$\epsilon_{\text{rel}} \approx \frac{1}{3^p} \tag{B.213}$$

which can be used to select the truncation order p. Taking logarithms, we obtain the rule of thumb

$$p \approx \left\lceil -\frac{\log(\epsilon_{\text{rel}})}{\log(3)} \right\rceil. \tag{B.214}$$

Thus, to compute a multipole expansion accurate to a relative tolerance ϵ_{rel} of machine epsilon in double precision we choose $p = 33$. When we perform such a truncation, (B.199) becomes

$$y_i = \sum_{m=0}^{p-1} \underbrace{\left[\sum_{j=1}^{n} x_j (\lambda_j - \lambda_\star)^m \right]}_{c_m} \frac{1}{(d_i - \lambda_\star)^{m+1}} \tag{B.215}$$

which costs $\mathcal{O}(pn)$ operations to compute the multipole coefficients c_m and $\mathcal{O}(pn)$ operations to evaluate the sum at each y_i once the coefficients c_m are known. If n is much larger than p, then this drastically reduces the cost of performing a matrix-vector multiplication since both sets of operations are linear in n. Compared with directly evaluating the matrix-vector

product, which requires $\mathcal{O}(n^2)$ operations, this savings can be substantial when $n \gg p$.

Unfortunately, when the source and target points are not well separated (they are not well separated in our application), then a single multipole expansion as described cannot be used alone to achieve such speedups. A careful use of several multipole expansions about different centers, as well as local Taylor expansions are needed to obtain similar computational complexity results. For this reason, we consider how to convert one multipole expansion about center $\lambda_\star^{(1)}$ to another multipole expansion centered about $\lambda_\star^{(2)}$. This can be achieved by a particular linear transformation applied to the coefficients of the multipole expansion. To see why, let $\lambda_\star = \lambda_\star^{(1)}$ and $c_m = c_m^{(1)}$ and consider rewriting (B.215) about a new center $\lambda_\star^{(2)}$. This gives

$$y_i = \sum_{m=0}^{p-1} c_m^{(1)} \frac{1}{(d_i - \lambda_\star^{(1)})^{m+1}} \tag{B.216}$$

$$= \sum_{m=0}^{p-1} c_m^{(1)} \frac{1}{(d_i - \lambda_\star^{(2)} + \lambda_\star^{(2)} - \lambda_\star^{(1)})^{m+1}} \tag{B.217}$$

$$= \sum_{m=0}^{p-1} c_m^{(1)} \frac{1}{\left[1 + \frac{\lambda_\star^{(2)} - \lambda_\star^{(1)}}{d_i - \lambda_\star^{(2)}}\right]^{m+1}} \frac{1}{(d_i - \lambda_\star^{(2)})^{m+1}}. \tag{B.218}$$

Using the binomial theorem, we obtain

$$y_i = \sum_{m=0}^{p-1} c_m^{(1)} \left[\sum_{k=0}^{\infty} (-1)^k \frac{(m+k)!}{k!m!} \left(\frac{\lambda_\star^{(2)} - \lambda_\star^{(1)}}{d_i - \lambda_\star^{(2)}}\right)^k\right] \frac{1}{(d_i - \lambda_\star^{(2)})^{m+1}} \tag{B.219}$$

$$= \sum_{m=0}^{p-1} c_m^{(1)} \left[\sum_{k=0}^{\infty} \frac{(m+k)!}{k!m!} \left(\frac{\lambda_\star^{(1)} - \lambda_\star^{(2)}}{d_i - \lambda_\star^{(2)}}\right)^k\right] \frac{1}{(d_i - \lambda_\star^{(2)})^{m+1}}. \tag{B.220}$$

Compared with the multipole expansion

$$y_i = \sum_{m=0}^{p-1} c_m^{(2)} \frac{1}{(d_i - \lambda_\star^{(2)})^{m+1}}, \tag{B.221}$$

we observe, by equating coefficients with like powers of $d_i - \lambda_\star^{(2)}$, that

$$c_0^{(2)} = c_0^{(1)}, \tag{B.222}$$

$$c_1^{(2)} = \Delta\lambda c_0^{(1)} + c_1^{(1)}, \tag{B.223}$$

$$c_2^{(2)} = (\Delta\lambda)^2 c_0^{(1)} + 2\Delta\lambda c_1^{(1)} + c_2^{(1)}, \tag{B.224}$$

$$c_3^{(2)} = (\Delta\lambda)^3 c_0^{(1)} + 3(\Delta\lambda)^2 c_1^{(1)} + 3\Delta\lambda c_2^{(1)} + c_3^{(1)}, \tag{B.225}$$

and so on, where $\Delta\lambda = \lambda_\star^{(1)} - \lambda_\star^{(2)}$. This means that the vectors

$$\bar{c}^{(1)} = \begin{bmatrix} c_0^{(1)} \\ c_1^{(1)} \\ \vdots \\ c_{p-1}^{(1)} \end{bmatrix}, \qquad \bar{c}^{(2)} = \begin{bmatrix} c_0^{(2)} \\ c_1^{(2)} \\ \vdots \\ c_{p-1}^{(2)} \end{bmatrix}, \tag{B.226}$$

are connected by the linear transformation

$$\bar{c}^{(2)} = \underline{A}_m^m \bar{c}^{(1)} \tag{B.227}$$

with matrix entries

$$(\underline{A}_m^m)_{ij} = \begin{cases} (\Delta\lambda)^{i-j} \binom{i-1}{j-1} & i \geq j \\ 0 & \text{otherwise.} \end{cases} \tag{B.228}$$

We use the notation $\underline{A}_m^m$ to signify that this matrix takes multipole coefficients to other multipole coefficients. This notation is useful because we will encounter similar matrices which take multipole coefficients to local Taylor coefficients, as well as matrices which take local Taylor coefficients to other local Taylor coefficients.

For the local Taylor expansions, we repeat a similar process to the one used to derive the multipole expansion, but assume that the convergence condition (B.195) is reversed. That is, we rewrite

$$\frac{1}{d-\lambda_j} = \frac{1}{d-\lambda_\star+\lambda_\star-\lambda_j} \tag{B.229}$$

$$= \frac{1}{(\lambda_\star-\lambda_j)\left[\frac{d-\lambda_\star}{\lambda_\star-\lambda_j}+1\right]} \tag{B.230}$$

$$= \frac{1}{\lambda_\star-\lambda_j}\frac{1}{1-\frac{d-\lambda_\star}{\lambda_j-\lambda_\star}} \tag{B.231}$$

and assume that $|d-\lambda_\star| < |\lambda_j-\lambda_\star|$ so that a Taylor series gives the convergent expression

$$\frac{1}{d-\lambda_j} = \frac{1}{\lambda_\star-\lambda_j}\sum_{m=0}^{\infty}\left(\frac{d-\lambda_\star}{\lambda_j-\lambda_\star}\right)^m \tag{B.232}$$

$$= \sum_{m=0}^{\infty}\frac{-1}{(\lambda_j-\lambda_\star)^{m+1}}(d-\lambda_\star)^m. \tag{B.233}$$

In this case, the series converges when d is closer to the center $\lambda_\star$ than the source λ_j is to the center. Similar relative error bounds can be derived for local expansions as for multipole

expansions. For this reason, we also truncate the local expansion after p terms. Then a local Taylor expansion for

$$y_i = \sum_{j=1}^{n} x_j \frac{1}{d_i - \lambda_j} \tag{B.234}$$

is given by

$$y_i = \sum_{j=1}^{n} x_j \left[\sum_{m=0}^{p-1} \frac{-1}{(\lambda_j - \lambda_\star)^{m+1}} (d_i - \lambda_\star)^m \right] \tag{B.235}$$

$$= \sum_{m=0}^{p-1} \underbrace{\left[\sum_{j=1}^{n} \frac{-x_j}{(\lambda_j - \lambda_\star)^{m+1}} \right]}_{b_m} (d_i - \lambda_\star)^m \tag{B.236}$$

where we have rearranged the summations to exploit similar indexing properties we observed with the multipole expansion. In particular, by computing the local Taylor expansion coefficients b_m using $\mathcal{O}(pn)$ operations, it is possible to evaluate the n entries y_i by $\mathcal{O}(pn)$ operations. Compared with the original sum which requires $\mathcal{O}(n^2)$ operations, this can require significantly fewer operations when $p \ll n$.

As with multipole expansions, it is useful to be able to convert local Taylor expansion coefficients $b_m^{(1)}$ for an expansion centered at $\lambda_\star^{(1)}$ to coefficients $b_m^{(2)}$ for a local Taylor expansion centered at $\lambda_\star^{(2)}$. We rewrite (B.236) with $b_m = b_m^{(1)}$ and $\lambda_\star = \lambda_\star^{(1)}$ in terms of the center $\lambda_\star^{(2)}$ to obtain

$$y_i = \sum_{m=0}^{p-1} b_m^{(1)} (d_i - \lambda_\star^{(1)})^m \tag{B.237}$$

$$= \sum_{m=0}^{p-1} b_m^{(1)} (d_i - \lambda_\star^{(2)} + \lambda_\star^{(2)} - \lambda_\star^{(1)})^m. \tag{B.238}$$

Grouping $d_i - \lambda_\star^{(2)}$ together and $\lambda_\star^{(2)} - \lambda_\star^{(1)}$ together, we apply the binomial theorem to obtain

$$y_i = \sum_{m=0}^{p-1} b_m^{(1)} \left[\sum_{k=0}^{m} \binom{m}{k} (d_i - \lambda_\star^{(2)})^{m-k} (\lambda_\star^{(2)} - \lambda_\star^{(1)})^k \right]. \tag{B.239}$$

Since we want to find the coefficients $b_m^{(2)}$ in the local Taylor expansion

$$y_i = \sum_{m=0}^{p-1} b_m^{(2)} (d_i - \lambda_\star^{(2)})^m, \tag{B.240}$$

we equate coefficients corresponding to like powers of $d_i - \lambda_\star^{(2)}$ to obtain

$$b_0^{(2)} = \binom{0}{0} b_0^{(1)} + \binom{1}{1}(-\Delta\lambda) b_1^{(1)} + \binom{2}{2}(-\Delta\lambda)^2 b_2^{(1)} + \binom{3}{3}(-\Delta\lambda)^3 b_3^{(1)} + \cdots \tag{B.241}$$

$$b_1^{(2)} = \binom{1}{0} b_1^{(1)} + \binom{2}{1}(-\Delta\lambda) b_2^{(1)} + \binom{3}{2}(-\Delta\lambda)^2 b_3^{(1)} + \cdots \tag{B.242}$$

$$b_2^{(2)} = \binom{2}{0} b_2^{(1)} + \binom{3}{1}(-\Delta\lambda) b_3^{(1)} + \cdots \tag{B.243}$$

$$b_3^{(2)} = \binom{3}{0} b_3^{(1)} + \cdots \tag{B.244}$$

and so on. If we let

$$\bar{b}^{(1)} = \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \\ \vdots \\ b_{p-1}^{(1)} \end{bmatrix}, \qquad \bar{b}^{(2)} = \begin{bmatrix} b_0^{(2)} \\ b_1^{(2)} \\ \vdots \\ b_{p-1}^{(2)} \end{bmatrix}, \tag{B.245}$$

then

$$\bar{b}^{(2)} = \underline{A}_l^l \bar{b}^{(1)} \tag{B.246}$$

with

$$(\underline{A}_l^l)_{ij} = \begin{cases} (-\Delta\lambda)^{j-i}\binom{j-1}{i-1} & j \geq i \\ 0 & \text{otherwise.} \end{cases} \tag{B.247}$$

As a final matter, we consider converting the coefficients c_m in a multipole expansion about center $\lambda_\star^{(1)}$ into coefficients for a local Taylor expansion with coefficients b_m centered at $\lambda_\star^{(2)}$. Since we are dealing with the multipole expansion

$$y_i = \sum_{m=0}^{p-1} c_m \frac{1}{(d_i - \lambda_\star^{(1)})^{m+1}} \tag{B.248}$$

and want a local expansion

$$y_i = \sum_{m=0}^{p-1} b_m (d_i - \lambda_\star^{(2)})^m, \tag{B.249}$$

we use the Taylor series

$$\underbrace{\frac{1}{(d - \lambda_\star^{(1)})^{m+1}}}_{f(d)} = \sum_{k=0}^{\infty} \frac{f^{(k)}(\lambda_\star^{(2)})}{k!}(d - \lambda_\star^{(2)})^k \tag{B.250}$$

to convert the multipole expansion into local expansion form. The derivatives $f^{(k)}(d)$ are given by

$$f^{(k)}(d) = (-m-1)(-m-2)\cdots(-m-k)(d-\lambda_\star^{(1)})^{-m-1-k} \tag{B.251}$$

$$= (-1)^k \frac{(m+k)!}{m!} \frac{1}{(d-\lambda_\star^{(1)})^{m+k+1}}. \tag{B.252}$$

Substituting (B.252) into (B.250), then substituting the resulting expression into (B.248) gives

$$y_i = \sum_{m=0}^{p-1} c_m \left[\sum_{k=0}^{\infty} (-1)^k \frac{(m+k)!}{m!k!} \frac{1}{(\lambda_\star^{(2)} - \lambda_\star^{(1)})^{m+k+1}} (d_i - \lambda_\star^{(2)})^k \right] \tag{B.253}$$

$$= \sum_{m=0}^{p-1} c_m \left[\sum_{k=0}^{\infty} (-1)^k \binom{m+k}{k} \frac{1}{(-\Delta\lambda)^{m+k+1}} (d_i - \lambda_\star^{(2)})^k \right] \tag{B.254}$$

which we compare with (B.249) to see that

$$b_0 = c_0(-1)^0 \binom{0}{0} \frac{1}{(-\Delta\lambda)^1} + c_1(-1)^0 \binom{1}{0} \frac{1}{(-\Delta\lambda)^2} + c_2(-1)^0 \binom{2}{0} \frac{1}{(-\Delta\lambda)^3} + \cdots \tag{B.255}$$

$$b_1 = c_0(-1)^1 \binom{1}{1} \frac{1}{(-\Delta\lambda)^2} + c_1(-1)^1 \binom{2}{1} \frac{1}{(-\Delta\lambda)^3} + c_2(-1)^1 \binom{3}{1} \frac{1}{(-\Delta\lambda)^4} + \cdots \tag{B.256}$$

and so on. If

$$\bar{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{bmatrix}, \qquad \bar{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{p-1} \end{bmatrix}, \tag{B.257}$$

then

$$\bar{b} = \underline{A}_m^l \bar{c} \tag{B.258}$$

with

$$(\underline{A}_m^l)_{ij} = (-1)^{i-1} \binom{i+j-2}{i-1} \frac{1}{(-\Delta\lambda)^{i+j-1}}. \tag{B.259}$$

Note that when converting multipole coefficients to other multipole coefficients (using $\underline{A}_m^m$), local coefficients to other local coefficients (using $\underline{A}_l^l$), or multipole coefficients to local coefficients (using $\underline{A}_m^l$), we always perform a matrix-vector multiplication with a p-by-p matrix. This means that the complexity of converting these types of expansions to other expansions is $\mathcal{O}(p^2)$.

We now have the tools required to implement the FMM. To understand how the FMM

works, let us consider a simplified example where source and target points are the same and are distributed uniformly over the interval $(0, 1)$. The FMM begins by bisecting this interval, and the resulting subintervals until each subinterval contains no more than $q < n$ points (we will choose q by analyzing the cost of the algorithm). Suppose that the original interval is labeled as belonging to level 0 of the binary tree used to represent the partitioning process. The two subintervals produced by bisecting the interval on level 0 belong to level 1 and so on. Since the points are uniformly distributed, the final partition is comprised of all subintervals belonging to level $l - 1$. There are 2^{l-1} subintervals on level $l - 1$ so that $n \approx 2^{l-1}q$.

The FMM proceeds as follows. First, we compute multipole expansions for each subinterval on level $l-1$ (all multipole expansions are centered at the midpoint of their corresponding subinterval). Since the coefficients of each expansion can be computed in $\mathcal{O}(qp)$ operations (there are only q sources per subinterval instead of n) and there are a total of 2^{l-1} subintervals, this costs $\mathcal{O}(2^{l-1}qp)$ operations.

Second, we compute multipole expansions centered at the midpoints of the parent subintervals. We do this using the multipole to multipole operator $\underline{A}_m^m$. For example, to compute the multipole coefficients for a subinterval on level $l - 2$, we use two multipole to multipole operators (one for each child of the given subinterval) to convert multipole coefficients from level $l - 1$ to ones on level $l - 2$. We repeat this process starting from the finest level and ending at the root of the tree. This is repeated

$$2^{l-1} + 2^{l-2} + \cdots + 2^1 = \sum_{i=1}^{l-1} 2^i \tag{B.260}$$

$$= \frac{1 - 2^l}{1 - 2} - 1 \tag{B.261}$$

$$= 2^l - 2 \tag{B.262}$$

times and each application of the multipole to multipole operator requires $\mathcal{O}(p^2)$ operations.

Third, we use the multipole to local operator $\underline{A}_m^l$ to transfer the effect of well separated subintervals to each subinterval. A subinterval is well separated from another subinterval if they belong to the same level of the tree, their parents are neighbors, but they themselves are not neighbors. There are at most 3 well separated subintervals for each subinterval. We use the multipole to local operator three times, summing the contribution of each together and store the corresponding local Taylor expansion coefficients for each subinterval. This means that we perform $\mathcal{O}(p^2)$ operations $3(2^l - 2)$ times (there are a total of $2^l - 2$ subintervals to perform this step for and each one requires at most 3 multipole to local operators).

Fourth, we use the local to local operator $\underline{A}_l^l$ to transfer the local contributions from a

parent subinterval to its children. We do this starting from the root of the tree (although the root has no local expansion) level by level working toward the leaves of the tree. This costs $\mathcal{O}(p^2)$ operations per operator and is applied to all $2^l - 2$ subintervals.

Finally, we evaluate the local expansion at each leaf subinterval. This costs $\mathcal{O}(qp)$ operations for each subinterval and there are 2^{l-1} leaves. This computes the contribution of each source outside the corresponding subinterval and its (at most) two neighbor leaf subintervals. To obtain the contribution of sources in these remaining three subintervals, we compute the sum directly, which costs $\mathcal{O}(q^2)$ operations per subinterval, or $3q^2$ operations for the three taken together. Since there are 2^{l-1} leaves, the total cost of this final step is $2^{l-1} \cdot 3q^2 + 2^{l-1}qp$ operations.

The approximate total cost of the algorithm as described is

$$\underbrace{2^{l-1}qp}_{\text{step 1}} + \underbrace{(2^l - 2)p^2}_{\text{step 2}} + \underbrace{3(2^l - 2)p^2}_{\text{step 3}} + \underbrace{(2^l - 2)p^2}_{\text{step 4}} + \underbrace{2^{l-1} \cdot 3q^2 + 2^{l-1}qp}_{\text{step 5}}. \tag{B.263}$$

Since $n \approx 2^{l-1}q$, this means that

$$2^l \approx \frac{2n}{q} \tag{B.264}$$

so that the total number of operations is approximately

$$\underbrace{np}_{\text{step 1}} + \underbrace{\frac{2n}{q}p^2}_{\text{step 2}} + \underbrace{3\frac{2n}{q}p^2}_{\text{step 3}} + \underbrace{\frac{2n}{q}p^2}_{\text{step 4}} + \underbrace{3nq + np}_{\text{step 5}}. \tag{B.265}$$

If we choose $q = p$, then this simplifies to

$$\underbrace{np}_{\text{step 1}} + \underbrace{2np}_{\text{step 2}} + \underbrace{6np}_{\text{step 3}} + \underbrace{2np}_{\text{step 4}} + \underbrace{3np + np}_{\text{step 5}} = 15np \tag{B.266}$$

which is linear in the number of source/target points. This is how the FMM achieves linear computational complexity. The source [61] is the first to describe such a FMM for two-dimensional electrostatic problems. The development here applies their ideas to the Cauchy matrix-vector multiplication problem.

In practice, the source and target points need not be the same. In our application, they are different, but are distributed similarly. In addition, they need not be uniformly distributed. In such a case, one uses an adaptive FMM. By this, we mean that the same binary tree is used to partition the interval $(0, 1)$, but that the leaf subintervals may belong to different levels of the tree. We simply stop refining the tree once a subinterval has fewer than $q = p$ source points. An adaptive FMM must take into account the non uniform refinement,

and does so by categorizing subintervals using a set of four lists (each subinterval has four lists). The first list is a list of neighbor subintervals (this list was needed in the simplified FMM as well). The second list is a list of the well separated subintervals for each subinterval (this list was also used in the simplified FMM). The third and fourth lists only arise in an adaptive FMM. The third list is a list of subintervals that are smaller than the current subinterval but whose multipole expansions converge quickly on the subinterval (these are well separated but not by the strict definition given earlier). Finally, the fourth list is a list of subintervals larger than the current subinterval whose multipole expansions cannot be used on the current subinterval (these are not well separated but are also not neighbors). To populate these lists, on each level l, we refer to subintervals 0 to $2^l - 1$ listed from left to right.

To populate List 1, we find a leaf subinterval on level l called j. We then set the left neighbor to $j' = j - 1$. If the left neighbor is less than 0, then there can be no left neighbor. Otherwise, if the neighbor is a leaf subinterval, we store it in List 1 for subinterval j on level l and terminate the search for the left neighbor. If the neighbor does not exist (the tree was not refined to this level), then we need to reduce the level to $l - 1$ and compute the parent of the left neighbor (given by $\lfloor j'/2 \rfloor$ where $\lfloor \cdot \rfloor$ indicates the floor function). Otherwise, we need to search for the neighbor on level $l + 1$ with the child subinterval indexed by $2j' + 1$. We repeat this check until a neighbor is found, or we have determined that no left neighbor exists. A similar process is performed to find the right neighbor. In that case, the neighbor is initialized as $j' = j + 1$. There is no neighbor if $j' > 2^l - 1$. Going up the tree, we set the next candidate neighbor to $2j'$, whereas going down, we set it to $\lfloor j'/2 \rfloor$.

To populate List 2, for every subinterval j on level l, we compute the parent $i = \lfloor j/2 \rfloor$ on level $l - 1$. The neighbors of the parent are $i_R = i + 1$ and $i_L = i - 1$. Their children are $j' = 2i_\star$ and $j' = 2i_\star + 1$ where $i_\star$ can be either i_R or i_L. When j is even, then the possible members of List 2 are $2i_L$, $2i_R$, or $2i_R + 1$ whereas when j is odd, the possible members are $2i_L$, $2i_L + 1$, or $2i_R + 1$. When determining which members are required, we ignore members outside the range 0 to $2^l - 1$, and any member that was not part of the refinement process of the tree.

To populate List 3, we find a leaf subinterval on level l with index j. Its left neighbor is $j' = j - 1$. If j' is less than zero, we stop adding subintervals to List 3. Otherwise, if we find another subinterval that is a leaf, we also stop. If the neighbor is not a leaf but belongs to the tree, we add descendant $2j'$ to List 3 and continue the search with descendant $2j' + 1$ on level $l + 1$. If the neighbor does not belong to the tree, we stop the search process. We also have to perform a similar search for the right neighbor $j' = j + 1$. In this case, we terminate if $j' > 2^l - 1$, if j' is a leaf subinterval, or if j' does not belong to the tree. If j' is not a leaf,

but belongs to the tree, we add descendant $2j'+1$ to List 3 and continue the search with descendant $2j'$ on level $l+1$.

Finally, to populate List 4, we use List 3. For every List 3 at level l for subinterval j that is not empty, we add an entry to List 4. If List 3 contains an entry j' on level l', then List 4 for subinterval j' on level l' should contain subinterval j on level l.

The adaptive FMM proceeds as follows. First, we compute the multipole coefficients at each leaf subinterval. Then we use the multipole to multipole operator $\underline{A}_m^m$ to compute multipole coefficients on parent subintervals by starting from the leaf subintervals and working up the tree. Next, we compute the local expansion coefficients for each subinterval in the tree using the multipole expansions for subintervals in List 2. This uses the multipole to local operator $\underline{A}_m^l$. So far, this is precisely the same as the simplified FMM, although List 2 may be different from the set of well separated subintervals in the simplified FMM. Then contributions from subintervals in List 4 are added to the local expansion by directly computing the local expansion from the sources inside those subintervals. Next, the local to local operator $\underline{A}_l^l$ is used to compute local expansion coefficients in a downward pass, as in the simplified FMM. For each leaf subinterval, the contributions from subintervals in List 3 are added to the local expansion coefficients using the multipole to local operator $\underline{A}_m^l$. Finally, the local expansion is evaluated at target points on each leaf subinterval. In addition, the contributions of subintervals in List 1 and of the leaf subinterval itself are evaluated directly. The sources [58, 59] describe in detail how the adaptive FMM method applies for electrostatic potential problems in two and three dimensions. The adaptive FMM described here for the Cauchy matrix is adapted from these sources.

In practice, each multipole and local expansion uses a center of subinterval j on level l of the tree corresponding to

$$\lambda_\star = \frac{2j+1}{2^{l+1}}. \tag{B.267}$$

In addition, the radius of a subinterval (half its length) on level l is given by

$$r = \frac{1}{2^{l+1}}. \tag{B.268}$$

In our development of the FMM, the radius of a subinterval has not been used. However, in practice, we actually work with scaled multipole and local coefficients to avoid overflow. It turns out that an appropriate scaling depends on the radius of subintervals. To see why scaling becomes necessary, consider the entries (B.259) of $\underline{A}_m^l$. The term $\binom{i+j-2}{i-1}$ grows large rapidly. In addition, so too does $(-\Delta\lambda)^{-(i+j-1)}$ when $\Delta\lambda$ becomes small. When the tree for the adaptive FMM requires many levels, the conversion of multipole expansions at fine levels to local expansions can overflow, causing the algorithm to fail.

To avoid overflow, we rewrite the multipole expansion (B.215) as

$$y_i = \sum_{m=0}^{p-1} \underbrace{\left[\sum_{j=1}^{n} x_j \frac{(\lambda_j - \lambda_\star)^m}{(\beta r)^{m+1}} \right]}_{\hat{c}_m} \left(\frac{\beta r}{d_i - \lambda_\star} \right)^{m+1} \tag{B.269}$$

so that $\beta r/(d_i - \lambda_\star)$ is $\mathcal{O}(1)$. Typically, we choose β to be a small integer since $|d - \lambda_j| \geq 2r$. This means that the scaled multipole coefficients are related to the original multipole coefficients by

$$\hat{c} = (\beta r)^{-1} \underline{D}_{\beta r}^{-1} \bar{c} \tag{B.270}$$

where

$$\hat{c} = \begin{bmatrix} \hat{c}_0 \\ \hat{c}_1 \\ \vdots \\ \hat{c}_{p-1} \end{bmatrix}, \qquad \bar{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{p-1} \end{bmatrix}, \qquad \underline{D}_{\beta r} = \begin{bmatrix} 1 & & & \\ & \beta r & & \\ & & \ddots & \\ & & & (\beta r)^{p-1} \end{bmatrix}. \tag{B.271}$$

Similarly, we rewrite the local expansion (B.236) as

$$y_i = \sum_{m=0}^{p-1} \underbrace{\left[\sum_{j=1}^{n} \frac{-x_j r^m}{(\lambda_j - \lambda_\star)^{m+1}} \right]}_{\hat{b}_m} \left(\frac{d_i - \lambda_\star}{r} \right)^m. \tag{B.272}$$

In this case, there is no need to include a scale factor like β because $|d_i - \lambda_\star| < r$. Then the scaled local coefficients are related to the original local coefficients by

$$\hat{b} = \underline{D}_r \bar{b} \tag{B.273}$$

where

$$\hat{b} = \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \\ \vdots \\ \hat{b}_{p-1} \end{bmatrix}, \qquad \bar{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{bmatrix}, \qquad \underline{D}_r = \begin{bmatrix} 1 & & & \\ & r & & \\ & & \ddots & \\ & & & r^{p-1} \end{bmatrix}. \tag{B.274}$$

Using the connections (B.270) and (B.273), we can derive scaled multipole to multipole,

local to local, and multipole to local operators. For example, from (B.227), we obtain

$$(\beta^{(2)}r^{(2)})^{-1}\underline{D}^{-1}_{\beta^{(2)}r^{(2)}}\bar{c}^{(2)} = \underbrace{(\beta^{(2)}r^{(2)})^{-1}\underline{D}^{-1}_{\beta^{(2)}r^{(2)}}\underline{A}^m_m(\beta^{(1)}r^{(1)})\underline{D}_{\beta^{(1)}r^{(1)}}}_{\underline{\hat{A}}^m_m}(\beta^{(1)}r^{(1)})^{-1}\underline{D}^{-1}_{\beta^{(1)}r^{(1)}}\bar{c}^{(1)} \tag{B.275}$$

$$\hat{c}^{(2)} = \underline{\hat{A}}^m_m\hat{c}^{(1)}. \tag{B.276}$$

Based on (B.228), we find that the scaled entries are

$$(\underline{\hat{A}}^m_m)_{ij} = \begin{cases} (\beta^{(2)}r^{(2)})^{-i}(\beta^{(1)}r^{(1)})^j(\Delta\lambda)^{i-j}\binom{i-1}{j-1} & i \geq j \\ 0 & \text{otherwise.} \end{cases} \tag{B.277}$$

By computing the ratios

$$\gamma = \frac{r^{(2)}}{r^{(1)}}, \qquad \delta = \frac{\Delta\lambda}{r^{(1)}}, \tag{B.278}$$

the entries become

$$(\underline{\hat{A}}^m_m)_{ij} = \begin{cases} (\beta^{(1)}\gamma)^{-i}(\beta^{(2)})^j(\delta)^{i-j}\binom{i-1}{j-1} & i \geq j \\ 0 & \text{otherwise} \end{cases} \tag{B.279}$$

which are independent of the absolute sizes of the subintervals and only dependent on their relative sizes. Similarly, for the local to local operator (B.246) we obtain

$$\underline{D}_{r^{(2)}}\bar{b}^{(2)} = \underbrace{\underline{D}_{r^{(2)}}\underline{A}^l_l\underline{D}^{-1}_{r^{(1)}}}_{\underline{\hat{A}}^l_l}\underline{D}_{r^{(1)}}\bar{b}^{(1)} \tag{B.280}$$

$$\hat{b}^{(2)} = \underline{\hat{A}}^l_l\hat{b}^{(1)} \tag{B.281}$$

with entries (B.247) becoming

$$(\underline{\hat{A}}^l_l)_{ij} = \begin{cases} (r^{(2)})^{i-1}(r^{(1)})^{-(j-1)}(-\Delta\lambda)^{j-i}\binom{j-1}{i-1} & j \geq i \\ 0 & \text{otherwise.} \end{cases} \tag{B.282}$$

Using the scaling

$$\gamma = \frac{r^{(2)}}{r^{(1)}}, \qquad \delta = \frac{-\Delta\lambda}{r^{(1)}}, \tag{B.283}$$

gives

$$(\underline{\hat{A}}_l^l)_{ij} = \begin{cases} \gamma^{i-1}\delta^{j-i}\binom{j-1}{i-1} & j \geq i \\ 0 & \text{otherwise} \end{cases} \tag{B.284}$$

which is independent of the absolute sizes of the subintervals. Finally, for the multipole to local operator (B.258) we obtain

$$\underline{D}_{r^{(2)}}\bar{b} = \underbrace{\underline{D}_{r^{(2)}}\underline{A}_m^l(\beta^{(1)}r^{(1)})\underline{D}_{\beta^{(1)}r^{(1)}}}_{\underline{\hat{A}}_m^l}(\beta^{(1)}r^{(1)})^{-1}\underline{D}^{-1}_{\beta^{(1)}r^{(1)}}\bar{c} \tag{B.285}$$

$$\hat{b} = \underline{\hat{A}}_m^l\hat{c} \tag{B.286}$$

with entries (B.259) becoming

$$(\underline{\hat{A}}_m^l)_{ij} = (r^{(2)})^{i-1}(\beta^{(1)}r^{(1)})^j(-1)^{i-1}\binom{i+j-2}{i-1}\frac{1}{(-\Delta\lambda)^{i+j-1}}. \tag{B.287}$$

Using the scaling (B.283) gives

$$(\underline{\hat{A}}_m^l)_{ij} = \frac{\gamma^{i-1}(\beta^{(1)})^j}{\delta^{i+j-1}}(-1)^{i-1}\binom{i+j-2}{i-1} \tag{B.288}$$

which is independent of the absolute sizes of the subintervals. For similar scaling used in the context of adaptive FMM, see [60].

To complete our discussion of the FMM, we remark that in the divide and conquer algorithm of Section B.4, we also need to multiply by the transpose of Cauchy matrices. The Cauchy matrix $\underline{\hat{C}}$ with entries

$$\underline{\hat{C}}_{ij} = \frac{1}{d_i - \lambda_j} \tag{B.289}$$

has a corresponding transpose matrix $\underline{\hat{C}}^T$ with entries

$$(\underline{\hat{C}}^T)_{ij} = \frac{1}{d_j - \lambda_i}. \tag{B.290}$$

Thus to multiply by $\underline{\hat{C}}^T$, we use the same FMM but interchange the source and targets and multiply the final result by -1. It is crucial that the direct evaluation code used for the self interactions and those of the neighbors in the FMM use the accurately calculated differences $\bar{\tau}$ rather than $\bar{\lambda}$ directly, otherwise large errors can occur when the FMM is used together with the divide and conquer algorithm. This care is not needed when forming multipole or local coefficients from sources because these expansions are always used away from near

singularities in the Cauchy matrix entries.

B.6 Properties of the Dirichlet Eigensystem

In this appendix, we have described a way to perform the local computations required in the domain decomposition method of Chapter 10 using a near linear number of operations. The method relies on a divide and conquer eigendecomposition algorithm and the FMM to compute the eigenvalues and eigenvectors of a symmetric tridiagonal matrix efficiently. In fact, the method described works, with minor modifications, to compute the eigenvalues and eigenvectors of an arbitrary symmetric tridiagonal matrix, not just those arising from the finite element method in this thesis. For this reason, in this final section of Appendix B, we consider the structure of the pentadiagonal matrix for the Dirichlet problem and its associated two symmetric tridiagonal matrices to see if alternative, more specialized algorithms can be exploited.

Recall from Sections B.2 and B.3 that the Dirichlet problem requires the eigensystem corresponding to the matrix $\underline{A}_d$ which is the matrix $\tilde{\underline{S}}\tilde{\underline{S}}^T$ with the first two rows and columns removed. The appendix of [190] shows that the eigenvalues of $\underline{A}_d$ can be related to the roots of Lommel polynomials (see page 294 of [191], for example). Lommel polynomials $R_{m,\nu}(z)$ (which are actually polynomials in z^{-1}) satisfy a recurrence

$$R_{m-1,\nu}(z) + R_{m+1,\nu}(z) = \frac{2(\nu+m)}{z} R_{m,\nu}(z) \tag{B.291}$$

with certain useful base cases

$$R_{1,\nu}(z) = \frac{2\nu}{z}, \qquad R_{0,\nu}(z) = 1, \qquad R_{-1,\nu}(z) = 0, \qquad R_{-2,\nu}(z) = -1, \tag{B.292}$$

given on page 299 of [191]. The Lommel polynomials are intimately related to Bessel functions. Later, we will use the fact that

$$\frac{2\sin(\nu\pi)}{\pi z} R_{m,\nu}(z) = J_{\nu+m}(z)J_{-\nu+1}(z) + (-1)^m J_{-\nu-m}(z)J_{\nu-1}(z) \tag{B.293}$$

where $J_n(z)$ is the Bessel function of the first kind of order n. In addition,

$$J_{\nu+m}(z) = J_\nu(z)R_{m,\nu}(z) - J_{\nu-1}(z)R_{m-1,\nu+1}(z). \tag{B.294}$$

These last two relations are given on page 295 of [191]. In our manipulations, we treat m as an integer and ν as a real number.

To see how Lommel polynomials are related to the matrix $\underline{A}_d$, we use the method in [192] (which is for roots of Bessel functions) to construct an eigenvalue problem for the roots of Lommel polynomials. That is, we let $m = n - 1$ in (B.291) to obtain

$$R_{n-2,\nu}(z) + R_{n,\nu}(z) = \frac{2(\nu + n - 1)}{z} R_{n-1,\nu}(z) \tag{B.295}$$

$$\frac{1}{(\nu + n - 1)} \left[R_{n-2,\nu}(z) + R_{n,\nu}(z) \right] = \frac{2}{z} R_{n-1,\nu}(z). \tag{B.296}$$

Similarly, we let $m = n + 1$ in (B.291) to obtain

$$R_{n,\nu}(z) + R_{n+2,\nu}(z) = \frac{2(\nu + n + 1)}{z} R_{n+1,\nu}(z) \tag{B.297}$$

$$\frac{1}{(\nu + n + 1)} \left[R_{n,\nu}(z) + R_{n+2,\nu}(z) \right] = \frac{2}{z} R_{n+1,\nu}(z). \tag{B.298}$$

We then add (B.296) to (B.298) to find that

$$\frac{1}{(\nu + n - 1)} \left[R_{n-2,\nu}(z) + R_{n,\nu}(z) \right] + \frac{1}{(\nu + n + 1)} \left[R_{n,\nu}(z) + R_{n+2,\nu}(z) \right] = \frac{2}{z} \left[R_{n-1,\nu}(z) + R_{n+1,\nu}(z) \right] \tag{B.299}$$

where another application of (B.291) on the right hand side yields

$$\frac{R_{n-2,\nu}(z) + R_{n,\nu}(z)}{(\nu + n - 1)} + \frac{R_{n,\nu}(z) + R_{n+2,\nu}(z)}{(\nu + n + 1)} = \frac{2}{z} \left[\frac{2(\nu + n)}{z} R_{n,\nu}(z) \right] \tag{B.300}$$

$$\frac{R_{n-2,\nu}(z)}{(\nu + n - 1)} + \frac{2(\nu + n) R_{n,\nu}(z)}{(\nu + n - 1)(\nu + n + 1)} + \frac{R_{n+2,\nu}(z)}{(\nu + n + 1)} = \frac{4(\nu + n)}{z^2} R_{n,\nu}(z). \tag{B.301}$$

Dividing by $4(\nu + n)$ gives

$$\frac{R_{n-2,\nu}(z)}{4(\nu + n)(\nu + n - 1)} + \frac{R_{n,\nu}(z)}{2(\nu + n - 1)(\nu + n + 1)} + \frac{R_{n+2,\nu}(z)}{4(\nu + n)(\nu + n + 1)} = \frac{1}{z^2} R_{n,\nu}(z) \tag{B.302}$$

which is a three-term recurrence relation for Lommel polynomials.

Next, we show how this recurrence is related to the matrix $\underline{A}_d$. To do so, we need to consider the two tridiagonal matrices $\underline{T}_1$ and $\underline{T}_2$ arising from the permuted matrix

$$\underline{P}_{2,n/2}^T \underline{A}_d \underline{P}_{2,n/2} = \begin{bmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{bmatrix} \tag{B.303}$$

where $\underline{P}_{2,n/2}$ is the usual perfect shuffle matrix that takes the pentadiagonal matrix to

two tridiagonal matrices of half the size (see Section B.4 for details). We will obtain each tridiagonal matrix by considering (B.302) with a different value of ν. In each case, we choose $n = 2k + 1$ with $k = 0, 1, 2, ..., K$, which gives equations

$$\frac{R_{-1,\nu}(z)}{4(\nu+1)(\nu)} + \frac{R_{1,\nu}(z)}{2(\nu)(\nu+2)} + \frac{R_{3,\nu}(z)}{4(\nu+1)(\nu+2)} = \frac{1}{z^2}R_{1,\nu}(z), \tag{B.304}$$

$$\frac{R_{1,\nu}(z)}{4(\nu+3)(\nu+2)} + \frac{R_{3,\nu}(z)}{2(\nu+2)(\nu+4)} + \frac{R_{5,\nu}(z)}{4(\nu+3)(\nu+4)} = \frac{1}{z^2}R_{3,\nu}(z), \tag{B.305}$$

up to

$$\frac{R_{2K-1,\nu}(z)}{4(\nu+2K+1)(\nu+2K)} + \frac{R_{2K+1,\nu}(z)}{2(\nu+2K)(\nu+2K+2)} + \frac{R_{2K+3,\nu}(z)}{4(\nu+2K+1)(\nu+2K+2)} = \frac{1}{z^2}R_{2K+1,\nu}(z), \tag{B.306}$$

which we can write using matrix notation as

$$\underline{T}_\nu \tilde{R}_\nu(z) + \frac{1}{4(\nu+2K+1)(\nu+2K+2)}R_{2K+3,\nu}(z)\bar{e}_{K+1} = \frac{1}{z^2}\tilde{R}_\nu(z) \tag{B.307}$$

where we have used the fact that $R_{-1,\nu}(z) = 0$ and defined

$$\underline{T}_\nu = \begin{bmatrix} \frac{1}{2(\nu)(\nu+2)} & \frac{1}{4(\nu+1)(\nu+2)} & \\ \frac{1}{4(\nu+3)(\nu+2)} & \frac{1}{2(\nu+2)(\nu+4)} & \ddots \\ & \ddots & \ddots \end{bmatrix}, \qquad \tilde{R}_\nu(z) = \begin{bmatrix} R_{1,\nu}(z) \\ R_{3,\nu}(z) \\ R_{5,\nu}(z) \\ \vdots \\ R_{2K+1,\nu}(z) \end{bmatrix}, \tag{B.308}$$

where $\underline{T}_\nu$ is tridiagonal. Notice that when z_j is a root of $R_{2K+3,\nu}(z)$ such that $R_{2K+3,\nu}(z_j) = 0$, we obtain the eigenvalue equation

$$\underline{T}_\nu \tilde{R}_\nu(z_j) = \frac{1}{z_j^2}\tilde{R}_\nu(z_j) \tag{B.309}$$

with eigenvector $\tilde{R}_\nu(z_j)$ and corresponding eigenvalue $1/z_j^2$. For $\underline{T}_\nu$ to be related to $\underline{T}_1$ and $\underline{T}_2$, we need to make $\underline{T}_\nu$ symmetric. To do so, we scale $\tilde{R}_\nu(z_j)$ by a diagonal matrix such that

$$\bar{R}_\nu(z_j) = \underline{D}_\nu \tilde{R}_\nu(z_j) \tag{B.310}$$

with

$$\underline{D}_\nu = (-1)^{\nu-\frac{1}{2}} \begin{bmatrix} -\sqrt{1+\nu} & & & & \\ & \sqrt{3+\nu} & & & \\ & & -\sqrt{5+\nu} & & \\ & & & \ddots & \\ & & & & (-1)^K\sqrt{2K+1+\nu} \end{bmatrix}. \tag{B.311}$$

Multiplying (B.309) by $\underline{D}_\nu$ and using $\underline{D}_\nu^{-1}\underline{D}_\nu = \underline{I}$ gives

$$\underline{D}_\nu \underline{T}_\nu \underline{D}_\nu^{-1} \underbrace{\underline{D}_\nu \tilde{R}_\nu(z_j)}_{\bar{R}_\nu(z_j)} = \frac{1}{z_j^2} \underbrace{\underline{D}_\nu \tilde{R}_\nu(z_j)}_{\bar{R}_\nu(z_j)} \tag{B.312}$$

with

$$\underline{D}_\nu \underline{T}_\nu \underline{D}_\nu^{-1} = \begin{cases} \underline{T}_1 & \nu = \frac{1}{2} \\ \underline{T}_2 & \nu = \frac{3}{2}. \end{cases} \tag{B.313}$$

Thus the eigenvalues of $\underline{T}_1$ are $1/z_j^2$ for $R_{2K+3,\frac{1}{2}}(z_j) = 0$ with eigenvectors $\bar{R}_{\frac{1}{2}}(z_j)$ and the eigenvalues of $\underline{T}_2$ are $1/z_j^2$ for $R_{2K+3,\frac{3}{2}}(z_j) = 0$ with eigenvectors $\bar{R}_{\frac{3}{2}}(z_j)$. The union of the two sets of eigenvalues provides the eigenvalues of $\underline{A}_d$, and the eigenvectors of $\underline{A}_d$ are the perfect shuffle permutation $\underline{P}_{2,n/2}$ applied to the eigenvectors of each of $\underline{T}_1$ and $\underline{T}_2$ when padded with zeros (recall that $\underline{P}_{2,n/2}$ is roughly twice the size of $\bar{R}_{\frac{1}{2}}(z_j)$ or $\bar{R}_{\frac{3}{2}}(z_j)$). Note that this development is similar to the classical orthogonal polynomial treatment of Gauss quadrature rules via the truncated Jacobi matrix (see Section 9.3 for an example and [159] for more details) although the Lommel polynomials are less well known.

This development suggests that an alternative method to computing the eigenvalues and eigenvectors of $\underline{A}_d$ is to better understand the Lommel polynomials. In particular, one must be able to compute their roots numerically to obtain eigenvalues, and one must determine if it is possible to efficiently multiply by the eigenvector matrix

$$\underline{V}_\nu = \underline{D}_\nu \underbrace{\begin{bmatrix} R_{1,\nu}(z_1) & R_{1,\nu}(z_2) & R_{1,\nu}(z_3) & \cdots & R_{1,\nu}(z_K) \\ R_{3,\nu}(z_1) & R_{3,\nu}(z_2) & R_{3,\nu}(z_3) & \cdots & R_{3,\nu}(z_K) \\ R_{5,\nu}(z_1) & R_{5,\nu}(z_2) & R_{5,\nu}(z_3) & \cdots & R_{5,\nu}(z_K) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{2K+1,\nu}(z_1) & R_{2K+1,\nu}(z_2) & R_{2K+1,\nu}(z_3) & \cdots & R_{2K+1,\nu}(z_K) \end{bmatrix}}_{\underline{R}_\nu} \tag{B.314}$$

where $\underline{R}_\nu$ is a type of generalized Vandermonde matrix. To see if this is possible, we use

(B.293) to write Lommel polynomials in terms of more familiar functions. In particular, letting $\nu = \frac{1}{2}$, we obtain

$$\frac{2\sin(\frac{\pi}{2})}{\pi z} R_{m,\frac{1}{2}}(z) = J_{m+\frac{1}{2}}(z)J_{\frac{1}{2}}(z) + (-1)^m J_{-m-\frac{1}{2}}(z)J_{-\frac{1}{2}}(z) \tag{B.315}$$

$$R_{m,\frac{1}{2}}(z) = \frac{\pi z}{2}\left[J_{m+\frac{1}{2}}(z)J_{\frac{1}{2}}(z) + (-1)^m J_{-m-\frac{1}{2}}(z)J_{-\frac{1}{2}}(z)\right] \tag{B.316}$$

which shows that the Lommel polynomials we are interested in are related to spherical Bessel functions (since the order of the Bessel functions are integers shifted by one half). Since

$$Y_{m+\frac{1}{2}}(z) = (-1)^{m+1} J_{-m-\frac{1}{2}}(z) \tag{B.317}$$

$$= -(-1)^m J_{-m-\frac{1}{2}}(z) \tag{B.318}$$

and

$$J_{\frac{1}{2}}(z) = \sqrt{\frac{2}{\pi z}}\sin(z), \tag{B.319}$$

$$J_{-\frac{1}{2}}(z) = \sqrt{\frac{2}{\pi z}}\cos(z), \tag{B.320}$$

(see [193] and [194] for these relations), the Lommel polynomial can be written as

$$R_{m,\frac{1}{2}}(z) = \frac{\pi z}{2}\left[J_{m+\frac{1}{2}}(z)\sqrt{\frac{2}{\pi z}}\sin(z) - Y_{m+\frac{1}{2}}(z)\sqrt{\frac{2}{\pi z}}\cos(z)\right] \tag{B.321}$$

$$= \sqrt{\frac{\pi z}{2}}\left[J_{m+\frac{1}{2}}(z)\sin(z) - Y_{m+\frac{1}{2}}(z)\cos(z)\right]. \tag{B.322}$$

Similarly, letting $\nu = \frac{3}{2}$ in (B.293) and using the same spherical Bessel function properties gives

$$\frac{2\sin(\frac{3}{2}\pi)}{\pi z} R_{m,\frac{3}{2}}(z) = J_{m+1+\frac{1}{2}}(z)J_{-\frac{1}{2}}(z) + (-1)^m J_{-m-1-\frac{1}{2}}(z)J_{\frac{1}{2}}(z) \tag{B.323}$$

$$R_{m,\frac{3}{2}}(z) = -\frac{\pi z}{2}\left[J_{m+1+\frac{1}{2}}(z)\sqrt{\frac{2}{\pi z}}\cos(z) + Y_{m+1+\frac{1}{2}}(z)\sqrt{\frac{2}{\pi z}}\sin(z)\right] \tag{B.324}$$

$$= -\sqrt{\frac{\pi z}{2}}\left[J_{m+1+\frac{1}{2}}(z)\cos(z) + Y_{m+1+\frac{1}{2}}(z)\sin(z)\right]. \tag{B.325}$$

Thus, to compute the eigenvalues and eigenvectors of $\underline{A}_d$, we must study the properties of spherical Bessel functions.

Before doing so, we take this opportunity to explain why one might expect Lommel

polynomials to arise in this context. We note that the eigenvalue problem

$$\underline{A}_d\bar{\phi}_k = \frac{1}{\lambda_k}\bar{\phi}_k \tag{B.326}$$

arises when computing eigenfunctions of the BVP

$$\frac{d^2\phi_k}{dz^2} + \lambda_k\phi_k(z) = 0 \tag{B.327}$$

subject to homogeneous Dirichlet boundary conditions $\phi_k(\pm 1) = 0$. It is well known that the nonzero eigenfunctions of (B.327) are

$$\phi_k(z) = \sin\left(\frac{k\pi}{2}(z+1)\right) \tag{B.328}$$

with $k = 1, 2, 3, ...$, and corresponding eigenvalues

$$\lambda_k = \left(\frac{k\pi}{2}\right)^2. \tag{B.329}$$

To see why this problem is related to matrix $\underline{A}_d$, we let $\phi_k(z) = \bar{N}(z)^T\bar{\phi}$ and discretize the weak form of (B.327) using the method of weighted residuals. This gives the discrete problem

$$-\underline{S}_{DL}\underline{S}_{DL}^T\bar{\phi} + \lambda_k\underline{\tilde{S}}\underline{\tilde{S}}^T\bar{\phi} = 0. \tag{B.330}$$

Since the first two entries of $\bar{\phi}$ must be zero for the homogeneous Dirichlet boundary conditions to hold, this reduces to

$$-\underline{I}\bar{\phi}_k + \lambda_k\underline{A}_d\bar{\phi}_k = 0 \tag{B.331}$$

which gives the discrete eigenvalue problem

$$\underline{A}_d\bar{\phi}_k = \frac{1}{\lambda_k}\bar{\phi}_k. \tag{B.332}$$

This development is relevant because analyzing the eigenfunctions (B.328) using the addition formulas

$$\cos(\zeta\cos(\theta)) = 2^\nu\Gamma(\nu)\sum_{m=0}^{\infty}(-1)^m(\nu+2m)\frac{J_{\nu+2m}(\zeta)}{\zeta^\nu}C_{2m}^\nu(\cos(\theta)), \tag{B.333}$$

$$\sin(\zeta\cos(\theta)) = 2^\nu\Gamma(\nu)\sum_{m=0}^{\infty}(-1)^m(\nu+2m+1)\frac{J_{\nu+2m+1}(\zeta)}{\zeta^\nu}C_{2m+1}^\nu(\cos(\theta)), \tag{B.334}$$

from page 369 of [191] reveals the connection to Lommel polynomials. Here, Γ is the Gamma function, and C_n^ν is an ultraspherical (Gegenbauer) polynomial. We need to use both addition formulas because, like with the splitting of $\underline{T}_1$ and $\underline{T}_2$, our analysis splits into two sets of eigenfunctions, those that are even, and those that are odd. To begin, we consider the eigenfunctions with $k = 2l$. Letting $\zeta = l\pi$, $\cos(\theta) = z$, and $\nu = \frac{1}{2}$ in (B.333) gives

$$\cos(l\pi z) = \sqrt{2}\Gamma\left(\frac{1}{2}\right)\sum_{m=0}^{\infty}(-1)^m\left(\frac{1}{2}+2m\right)\frac{J_{2m+\frac{1}{2}}(l\pi)}{\sqrt{l\pi}}C_{2m}^{\frac{1}{2}}(z) \tag{B.335}$$

$$= \sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^m\left(\frac{1}{2}+2m\right)J_{2m+\frac{1}{2}}(l\pi)P_{2m}(z) \tag{B.336}$$

where we have used the facts that $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ and $C_n^{\frac{1}{2}}(z) = P_n(z)$ where $P_n(z)$ is an orthogonal (not orthonormal) Legendre polynomial. Since orthonormal Legendre polynomials $p_n(z)$ are related to orthogonal Legendre polynomials by $P_n(z) = \sqrt{\frac{2}{2n+1}}p_n(z)$, we have

$$\cos(l\pi z) = \sqrt{2\pi}\sum_{m=0}^{\infty}(-1)^m\left(\frac{1}{2}+2m\right)J_{2m+\frac{1}{2}}(l\pi)\sqrt{\frac{2}{2(2m)+1}}p_{2m}(z) \tag{B.337}$$

$$= \sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^m\sqrt{2m+\frac{1}{2}}J_{2m+\frac{1}{2}}(l\pi)p_{2m}(z). \tag{B.338}$$

Integrating both sides with respect to z yields

$$\int_{-1}^{z}\cos(l\pi z')dz' = \sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^m\sqrt{2m+\frac{1}{2}}J_{2m+\frac{1}{2}}(l\pi)\int_{-1}^{z}p_{2m}(z')dz' \tag{B.339}$$

$$\frac{\sin(l\pi z)}{l\pi} = \sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^m\sqrt{2m+\frac{1}{2}}J_{2m+\frac{1}{2}}(l\pi)N_{2m}(z) \tag{B.340}$$

where $N_n(z)$ are the integrated Legendre polynomials. Since

$$\sin(l\pi(z+1)) = \sin(l\pi z + l\pi) \tag{B.341}$$

$$= (-1)^l\sin(l\pi z), \tag{B.342}$$

we have

$$\frac{(-1)^l\sin(l\pi(z+1))}{l\pi} = \sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^m\sqrt{2m+\frac{1}{2}}J_{2m+\frac{1}{2}}(l\pi)N_{2m}(z) \tag{B.343}$$

$$\sin(l\pi(z+1)) = l\pi\sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^{l+m}\sqrt{2m+\frac{1}{2}}J_{2m+\frac{1}{2}}(l\pi)N_{2m}(z). \tag{B.344}$$

To finalize our derivation, we need to rewrite $J_{2m+\frac{1}{2}}(l\pi)$ using Lommel polynomials. We use (B.294) to find that

$$J_{\frac{1}{2}+2m}(l\pi) = J_{\frac{1}{2}}(l\pi)R_{2m,\frac{1}{2}}(l\pi) - J_{-\frac{1}{2}}(l\pi)R_{2m-1,\frac{3}{2}}(l\pi). \tag{B.345}$$

Since

$$J_{\frac{1}{2}}(l\pi) = \sqrt{\frac{2}{l\pi^2}}\underbrace{\sin(l\pi)}_{0}, \tag{B.346}$$

$$J_{-\frac{1}{2}}(l\pi) = \sqrt{\frac{2}{l\pi^2}}\underbrace{\cos(l\pi)}_{(-1)^l}, \tag{B.347}$$

we have that

$$J_{2m+\frac{1}{2}}(l\pi) = \frac{(-1)^{l+1}}{\pi}\sqrt{\frac{2}{l}}R_{2m-1,\frac{3}{2}}(l\pi). \tag{B.348}$$

Substituting this expression into (B.344) gives

$$\sin(l\pi(z+1)) = l\pi\sqrt{\frac{2}{l}}\sum_{m=0}^{\infty}(-1)^{l+m}\sqrt{2m+\frac{1}{2}}\frac{(-1)^{l+1}}{\pi}\sqrt{\frac{2}{l}}R_{2m-1,\frac{3}{2}}(l\pi)N_{2m}(z) \tag{B.349}$$

$$= 2\sum_{m=0}^{\infty}(-1)^{m+1}\sqrt{2m+\frac{1}{2}}R_{2m-1,\frac{3}{2}}(l\pi)N_{2m}(z). \tag{B.350}$$

Since the $m=0$ term is zero ($R_{-1,\frac{3}{2}}(z)=0$ for any z), this gives

$$\sin(l\pi(z+1)) = 2\sum_{m=1}^{\infty}(-1)^{m+1}\sqrt{2m-1+\frac{3}{2}}R_{2m-1,\frac{3}{2}}(l\pi)N_{2m}(z). \tag{B.351}$$

This last expression shows that when k is even in (B.328), Lommel polynomials and integrated Legendre polynomials can be used to represent eigenfunctions of the second derivative operator. Compare this expression with the eigenvector $\bar{R}(z_j)$. The term

$$(-1)^{m+1}\sqrt{2m-1+\frac{3}{2}}R_{2m-1,\frac{3}{2}}(l\pi) \tag{B.352}$$

is closely related to $\bar{R}(z_j)$ because $z_j \approx l\pi$ for small j. This means that the finite sum $\sum_m c_m R_{2m-1,\frac{3}{2}}(z_j)N_{2m}(z)$ with appropriate constants c_m tends to accurately represent the eigenfunctions $\sin(l\pi(z+1))$ for small frequencies as one would expect for eigenvectors arising from a second derivative matrix.

A similar derivation holds for $k=2l-1$ by letting $\zeta=(2l-1)\frac{\pi}{2}$, $\cos(\theta)=z$, and $\nu=\frac{1}{2}$

in (B.334). Rather than repeat a similar derivation, the result is

$$\sin\left((2l-1)\frac{\pi}{2}(z+1)\right) = 2\sum_{m=0}^{\infty}(-1)^{m+1}\sqrt{2m+1+\frac{1}{2}}R_{2m+1,\frac{1}{2}}\left((2l-1)\frac{\pi}{2}\right)N_{2m+1}(z) \tag{B.353}$$

which shows that when k is odd in (B.328), we obtain a similar representation in terms of Lommel polynomials and integrated Legendre polynomials. In this case, we get

$$(-1)^{m+1}\sqrt{2m+1+\frac{1}{2}}R_{2m+1,\frac{1}{2}}\left((2l-1)\frac{\pi}{2}\right) \tag{B.354}$$

corresponding closely with the other set of eigenvectors. By showing these two addition formulas, we gain intuition as to why the discrete eigenfunctions for $\underline{A}_d$ behave the way that they do.

Finally, we close this section with certain suggestions for exploiting the rich analytical structure of the eigenvalues and eigenvectors of $\underline{A}_d$. In theory, the eigenvalues of $\underline{A}_d$ can be computed using Newton's method for each root of $R_{2K+3,\frac{1}{2}}(z)$ and $R_{2K+3,\frac{3}{2}}(z)$ using a small number of iterations. Since these functions can be written using Bessel and sinusoidal functions according to (B.322) and (B.325), they can also be evaluated in $\mathcal{O}(1)$ operations (their derivatives can also be evaluated with the same complexity). For Newton's method to converge, suitable initial iterates are required. Initial iterates are simple to find for smaller roots because of the addition formula derivations above but are not obvious for large roots[6].

Once the eigenvalues are computed, fast multiplications with the eigenvector matrices $\underline{V}_\nu$ are required. The key is determining how to perform the matrix-vector product $\underline{R}_\nu\bar{x}$ efficiently. As an example, consider the $\nu = \frac{1}{2}$ case. Then using (B.322) and the definition of $\underline{R}_\nu$ yields

$$\underline{R}_{\frac{1}{2}} = \underline{J}\underline{D}_s - \underline{Y}\underline{D}_c \tag{B.355}$$

where $\underline{J}$ is a matrix containing Bessel functions of the first kind of odd order plus one half evaluated at the roots of Lommel polynomials, $\underline{Y}$ is a matrix containing Bessel functions of the second kind of odd order plus one half evaluated at the roots of Lommel polynomials, and $\underline{D}_s$ and $\underline{D}_c$ are diagonal matrices with sines and cosines multiplied by $\sqrt{\frac{\pi z}{2}}$, all evaluated at roots of Lommel polynomials. It is difficult to perform the matrix-vector products with $\underline{J}$ and $\underline{Y}$ efficiently. Since $\underline{D}_c$ is effectively zero for a portion of its diagonal entries, it is possible to only compute a portion of $\underline{Y}$, avoiding singular behavior. It remains to be seen whether uniform asymptotic expansions for Bessel functions [196, 197] can be leveraged to perform matrix-vector products with $\underline{J}$ and $\underline{Y}$ with near linear computational complexity.

[6]For large roots, a good asymptotic approximation is needed, which can be found in [195].