In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

Partial FFT Direct Parallel Algorithms for Subsurface Scattering Problems

Ronald Gonzales

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering and Applied Sciences

Department of Mathematics and Statistics

Idaho State University

Spring 2022

To the Graduate Faculty:

The members of the committee appointed to examine the dissertation of Ronald Gonzales find it satisfactory and recommend that it be accepted.

Dr. Yury Gryazin,
Major Adviser

Dr. Bennett Palmer,
Committee Member

Dr. Wenxiang Zhu,
Committee Member

Dr. Ken Bosworth,
Committee Member

Dr. Marco Schoen,
Graduate Faculty Representative

Dedication

I dedicate my dissertation to my family, godsons and friends. Thank you all for your patience and understanding in my absence during my work.

To my parents, thank you for all the love and support. Any success that I may achieve could not have been possible without you. Mom, your years of hard work and dedication to our family opened the door for me to continue my education. Dad, without your support and encouragement I could not have done any of this.

To David, thank you for all the FaceTime calls to keep me sane during my long hours on the computer.

To Christa, thank you for all the love and support throughout my doctoral education.

To Johnathan and the monsters, I hope my dedication encourages you to work hard toward whichever careers you may choose.

Acknowledgment

I would like to thank my mentor, Professor Yury Gryazin, whose knowledge and expertise in applied mathematics proved invaluable during my doctoral education. Your passion for education and the field of applied mathematics inspired my career path and gave me direction for my own passions.

Thank you to the editors, my mother Rachel and my girlfriend Christa, for their tremendous support and assistance during the writing of this dissertation.

My progress in mathematics would not be possible without a firm foundation. I want to thank my high school mathematics teacher, Ron Lagerstrom, for his belief in my abilities and his encouragement.

I would like to acknowledge a fellow doctoral student, Yun Teck Lee, for your collaboration throughout our graduate education.

My gratitude to ISU's Department of Mathematics and Statistics for the years of continuous support, education and an overall excellent resource.

Finally, I would like to thank the defense committee for being so flexible in their schedules and open about expectations.

Contents

List of Figures . . . . . **ix**

List of Tables . . . . . **xi**

List of Acronyms . . . . . **xii**

List of Symbols . . . . . **xiii**

Abstract . . . . . **xiv**

1 Introduction **1**

2 Two-Dimensional Discretization **10**

2.1 Second-Order Accuracy 10

2.2 Fourth-Order Accuracy 11

2.3 Sixth-Order Accuracy 14

2.4 Summary 22

3 Three-Dimensional Discretization **23**

3.1 Second-Order Accuracy 24

3.2 Fourth-Order Accuracy 25

3.3 Sixth-Order Accuracy . . . 27
3.4 Fourth-Order Accuracy for Convection-Diffusion . . . 37
3.5 Summary . . . 40

4 Boundary Conditions . . . 41
4.1 Dirichlet Conditions . . . 42
4.2 Sommerfeld-Like Conditions . . . 44
4.2.1 Fourth-Order Accuracy . . . 45
4.2.2 Seventh-Order Accuracy . . . 46
4.2.3 Ninth-Order Accuracy . . . 47
4.3 Summary . . . 50

5 Direct FFT Solver . . . 51
5.1 Stencils . . . 51
5.2 Coefficients . . . 52
5.2.1 Second-Order . . . 53
5.2.2 Fourth-Order . . . 54
5.2.3 Sixth-Order . . . 54
5.3 Solver . . . 55
5.3.1 Eigenvalues . . . 56
5.3.2 Diagonalization . . . 58
5.3.3 Finding the DST with FFT . . . 63
5.3.4 Block Tridiagonal Solver . . . 65
5.3.5 Sequential Algorithm . . . 66

5.4 Parallel Implementations 66
5.4.1 OpenMP 67
5.4.2 MPI 68
5.4.3 Hybrid 70
5.5 Summary 72

6 Direct Generalized Eigenvalue Solver 73
6.1 Uniform Medium 73
6.2 Air and Soil Medium 74
6.3 Air and Soil Medium with Subsurface Inclusion 79
6.4 Parallel Implementation 83
6.5 Summary 85

7 Direct Partial FFT Solver 86
7.1 Computation Reduction 89
7.1.1 Step One 89
7.1.2 Step Two 91
7.1.3 Step Three 92
7.2 Summary 93

8 Numerical Results 94
8.1 FFT Solver 94
8.1.1 Helmholtz 94
8.1.2 Convection-Diffusion 99

8.2 Generalized Eigenvalue and Partial FFT Solvers 102

8.2.1 Constant k 102

8.2.2 Air and Soil Medium 107

8.2.3 Air and Soil Medium with Subsurface Inclusions 110

8.3 Summary 113

9 Conclusion and Future Work 116

9.1 Future Work 117

10 Bibliography 119

List of Figures

4.1 General 9-Point Stencil for Constant k 42

4.2 3×3 Computational Domain with 9-Point Stencil 42

4.3 Coefficient Matrix $\mathcal{B}$ with Boundary Elements 43

5.1 FFT Solver 9-Point Stencil 52

5.2 FFT Solver 27-Point Stencil 53

5.3 3D Transfer of Data Between MPI Processes 71

6.1 Vertical Cross-Section of Air and Soil Domain 75

6.2 Generalized Eigenvalue Solver Design 79

6.3 Vertical Cross-Section of Air and Soil Domain with Subsurface Inclusion . . 80

7.1 2D Matrix Sparsity Example 90

8.1 Constant k FFT Solver OpenMP vs MPI on 512^3 98

8.2 Variable k FFT Solver OpenMP vs MPI on 512^3 98

8.3 Convection-Diffusion FFT Solver OpenMP 102

8.4 Constant k Partial FFT Solver OpenMP on 1025^2 106

8.5 Constant k Partial FFT Solver OpenMP on 2049^2 107

8.6 Variable k Partial FFT Solver OpenMP on 2049^2 110

8.7 2D Domain with Air, Soil and Rectangular Inclusion 114

8.8 Real Part of 2D Solution with Rectangular Inclusion 114

8.9 Partial FFT Solver OpenMP with Rectangular Inclusion 115

List of Tables

8.1 Comparison of Direct and Iterative Solvers 96

8.2 FFT Solver Convergence 97

8.3 FFT Solver Hybrid on 512^3 99

8.4 FFT Solver Hybrid on Large Grids 99

8.5 FFT Solver Convergence for Convection-Diffusion 100

8.6 FFT Solver OpenMP for Convection-Diffusion 101

8.7 Generalized Eigenvalue Solver Convergence 103

8.8 Partial FFT Solver Convergence 104

8.9 Solver Timing Comparison on Uniform Grids 105

8.10 Solver Timing Comparison on Nonuniform Grids 105

8.11 Partial FFT Solver Residuals for Air and Soil Problem 108

8.12 Generalized Eigenvalue and Partial FFT Solvers OpenMP on 2049^2 109

8.13 Generalized Eigenvalue and Partial FFT Solvers OpenMP on 4097^2 109

8.14 Electrical Parameters 112

8.15 Partial FFT Solver Residuals for Inclusion Problem 113

List of Acronyms

ABC	Absorbing Boundary Condition
API	Application Programming Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DFT	Discrete Fourier Transform
DST	Discrete Sine Transform
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
GPU	Graphics Processing Unit
LAPACK	Linear Algebra PACKage
MIT	Massachusetts Institute of Technology
MPI	Message Passing Interface
RAM	Random Access Memory

List of Symbols

$\mathbb{N}$	Set of natural numbers, the nonnegative integers
$\mathbb{Z}$	Set of integers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
Δ	Laplace operator
∇	Gradient operator
δ_x	Central difference approximation for the first derivative
δ_x^2	Central difference approximation for the second derivative
A^T	Transpose of the matrix A
$\otimes$	Kronecker product

Partial FFT Direct Parallel Algorithms for Subsurface Scattering Problems

Dissertation Abstract - - Idaho State University (2022)

The present research introduces a direct parallel partial fast Fourier transform (FFT) algorithm for the numerical solutions of the two- and three-dimensional Helmholtz equations. The governing equations are discretized by high-order compact finite difference methods. The resulting discretized system is indefinite, making the convergence of most iterative methods deteriorate as frequency increases. For indefinite systems parallel direct approaches are a better alternative, especially for systems with discontinuous and singular right-hand sides. The research focuses on the efficient parallel implementation of the proposed algorithm in both shared (OpenMP) and distributed (MPI) memory environments. The complexity and speed-up of the direct parallel methods are investigated on scattering problems with realistic ranges of parameters in air, soil and mine-like targets.

Key Words: Parallel algorithm, direct solver, FFT, subsurface scattering, mine-like inclusions, OpenMP, MPI, high-order, Sommerfeld-like boundary conditions.

1 Introduction

Increasing the resolution of existing numerical solvers is the focus of research in many areas of science and engineering. The need for improved accuracy, or resolution, increases the time and memory requirements of existing sequential methods rendering them unacceptable in many applications. Performance enhancement of a single processing unit has plateaued in recent decades. Implementation of multiple-core processors began in 2005 [1]. Today the trend of computing performance improvement primarily through the addition of processing units continues. Therefore, the development of parallel algorithms is necessary for high-resolution simulation of many natural phenomena and engineering applications.

The purpose of the research study is to present a novel direct parallel partial FFT solver. This is accomplished by introducing a direct FFT solver followed by a novel generalized eigenvalue solver. Slightly modified versions of these solvers combined produce the partial FFT solver. The basic premise of each solver is the same: transform the right-hand side (RHS), solve the system and transform the solution back to the solution space.

The discretized schemes for the two- and three-dimensional Helmholtz equations are the target systems for the solvers. The equations and boundary conditions are discretized using second-, fourth- and sixth-order compact finite difference schemes. Nabavi presented the two-dimensional case scheme with constant-coefficient [2], which Sutmann extended to the three-dimensional case [3]. The fourth-order approximation is an extension of the work done by Lele [4]. Turkel presented a sixth-order scheme with variable-coefficient in [5] and Gryazin provided this scheme with Sommerfeld-like boundary conditions in [6]. The work focuses on demonstrating the efficient parallel implementations of these direct numerical methods in

shared and distributed memory environments.

The target problems considered in the research are the two- and three-dimensional Helmholtz equations on rectangular domains with Dirichlet or Sommerfeld-like boundary conditions. That is

$$\Delta u(\mathbf{x}) + k^2(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}) \text{ in } \Omega \tag{1.1}$$

$$\Gamma u(\mathbf{x}) = g(\mathbf{x}) \quad \text{for } \mathbf{x} \in \partial\Omega \tag{1.2}$$

where Γ is the differential operator corresponding to either Dirichlet or Sommerfeld-like conditions [7]. The three-dimensional domain is defined as

$$\Omega = \left\{ \mathbf{x} = (x, y, z) \in \mathbb{R}^3 \mid x_l \leq x \leq x_u,\, y_l \leq y \leq y_u,\, z_l \leq z \leq z_u \right\}$$

where $x_l < x_u$, $y_l < y_u$ and $z_l < z_u$. The two-dimensional domain is defined by simply omitting the z direction from Ω. The function k is complex valued. The stability, existence and uniqueness analysis of the problem can be found in [8, 9].

In the case of Sommerfeld-like conditions, Γ from (1.2) is an approximation for Sommerfeld radiation conditions given by

$$\lim_{r\to\infty} r^{(d-1)/2} \left(\frac{\partial}{\partial r} u(\mathbf{x}) - ik(\mathbf{x})u(\mathbf{x}) \right) = 0 \tag{1.3}$$

where $r = ||\mathbf{x}||_2$ with $d = 2$ or $d = 3$ for two- and three-dimensions respectively [7]. Truncating the unbounded domain to a finite domain at the boundary under consideration provides the approximation [10]. That is

$$\nabla u(\mathbf{x}) \cdot \mathbf{n} - ik(\mathbf{x})u(\mathbf{x}) = 0$$

for $\mathbf{x} \in \partial\Omega$ where $\mathbf{n}$ is the outward normal vector of the boundary. This approximation presents a case of absorbing boundary conditions (ABCs) that reduce reflections caused by the boundary, which are considered first-order ABCs [10]. Higher-order ABCs are viable, but are not considered here as they do not have a dramatic influence on the accuracy of the solvers under consideration [11].

The immediate application of the problem given by (1.1,1.2) with k varying in all spatial directions is the forward solution of high-frequency electromagnetic wave propagation, a vital piece in imaging subsurface mine-like targets. Mine-like objects are modeled by $k(\mathbf{x})$ as small inclusions within a medium containing both air and soil. The inclusions are defined by the electrical conductivity and permittivity that differ from the surrounding soil [12]. Solving the forward scattering problem presents a challenge because of the high-frequency required [13], forcing the use of finer computational grids and thus necessitating a highly parallel solver. The numerical results generated here examine applications with realistic ranges of parameters in air, soil and mine-like targets [13]. High-resolution is key in imaging subsurface mine-like targets and is enabled by two contributing factors. These factors include a) the application of fourth- and sixth-order schemes and b) the utilization of finer computational grids through increased computational power via parallel computing.

Robey and Zamora define parallel computing as "the practice of identifying and exposing parallelism in algorithms, expressing this in our software, and understanding the costs, benefits, and limitations of the chosen implementation [1]." One such benefit of running computations in parallel is decreased computation time. Wall-time and central processing unit (CPU) time are two common measures for computation time. CPU-time measures the length of time the CPU was used, but it does not capture the speed-up advantages of com-

putations run in parallel. For computations in parallel, CPU-time reflects the sum of time per processing unit. Thus, CPU-time does not decrease because computations in parallel utilize multiple processing units simultaneously. Wall-time, utilized here, can capture the parallel computing speed-up advantages because it measures the length of time to complete a task.

The goal of implementing parallel algorithms is to observe a reduction in computation time with the addition of processing units, or processes. A process is "an independent unit of computation that has ownership of a portion of memory and control over resources in user space [1]." Ideally, a doubling of the number of processes would reduce the wall-time by half, or linear speed-up. Perfect linear speed-up in computation time is typically not observed in practice as the increase in number of processors also increases the overhead and the number of communications between these processes. Thus, there is a limit to the speed-up as the latency, delay due to communication, can outweigh the benefit of reducing the number of computations done by a process.

To achieve reduction in wall-time the computations must be divided as evenly as possible among the available processes. Consider a for-loop with n iterations and let p be the number of processes. Assume $n > p$ as in practice n will be very large and p is limited by the hardware. If p divides n, the number of iterations performed by each process is simply p/k. On the other hand, if $n \mod p = r \neq 0$ then one of the remaining iterations is given to r process, preventing one process from doing significantly more work than the others.

The simultaneous operations required to reduce wall-time are enabled by hardware such as multi-core processors and graphics processing units (GPUs), now standard in personal computers. Both contain multiple processing units with the ability to performer opera-

tions independently. However, personal computers are often unable to run full-scale high-resolution simulations for scientific application due to lack of memory and processing power. A computer cluster is a collection of computers wired together giving access to significantly more CPUs, GPUs and memory. An individual computer within a cluster is called a node and possesses several processors, often 32, and one or more GPUs.

Two parallel technologies utilized, OpenMP and MPI, are tools that provide the programmer access to multiple processing units. OpenMP and MPI are utilized separately and together in a hybrid implementation. The first parallel tool, OpenMP, is an application programming interface (API) for shared memory architectures. Utilizing OpenMP delivers a relatively simple implementation and excellent reduction in wall-time, but is also restricted. The requirement of shared memory limits a program using strictly OpenMP to a single computer or node, limiting the number of processes that can be utilized and, more importantly, the availability of random access memory (RAM). Some tests require vast amounts of RAM that cannot be provided on a single node. OpenMP now has the ability to access the power of GPUs, but is only considered here for use on CPUs. The provided FFT solver was tested on GPUs via CUDA, a parallel tool for NVIDIA GPUs. The resulting implementations saw a reduction in wall-time, but the performance was underwhelming compared to the Message Passing Interface (MPI) implementation due to the bottleneck created by the data transfer to and from the GPU's memory.

The ability to run large-scale computational grids requires the RAM from multiple nodes, which calls for the use of MPI, today's standard for communication between processes in a distributed memory environment [14]. MPI implementations only allocate the minimum required memory on each process's available memory by dividing the computational domain

as evenly as possible among the processes. In turn, this distribution allows for much larger computational grids as the program is no longer limited by the memory of a single node. Each process independently runs its own copy of the program on its assigned portion of the computational domain. The process passes messages, or transfers data, when necessary.

If k is constant or assumed to only vary in the vertical direction then the resulting stencil pattern formed by the discretization leads to a system that can be solved directly with an FFT solver. This restricting pattern is found when working with strictly Dirichlet conditions or Sommerfeld-like conditions on only the top and bottom boundaries with respect to the vertical direction and Dirichlet on the others. In this case, the FFT solver utilizes the discrete sine transform (DST) to transform the right-hand side, then LU decomposition is used to solve the system and then DST is used to transform the solution. The discrete Fourier transform (DFT) is used to compute the necessary DSTs. Massachusetts Institute of Technology (MIT) supports a widely used library, Fastest Fourier Transform in the West (FFTW), that is implemented for the DFT calculations via FFT [15]. Let N_x, N_y and N_z be the number of grid points in the x, y and z directions respectively. Then, in the three-dimensional case, the direct FFT solver requires $\mathcal{O}\left(N_x \cdot N_y \cdot N_z \log(N)\right)$ operations where $N = \max\{N_x, N_y\}$. The major advantage of this approach is the natural parallelization in the transformations via DST.

The pattern in the stencil is lost when utilizing k that varies in all spatial directions or including Sommerfeld-like conditions on all boundaries. Hence, a more versatile approach is required for the target application problems where k contains an inclusion. The novel generalized eigenvalue and partial FFT solvers can overcome the loss of stencil pattern and effectively approximate the solution to (1.1) with mine-like inclusions in k and Sommerfeld-

like conditions on all boundaries. The partial FFT solver is a combination of the FFT solver and generalized eigenvalue solver that takes advantage of the highly parallel nature of the FFT solver and reduces the number of operations required by the generalized eigenvalue solver to only $\mathcal{O}\left(N_x \cdot N_y \cdot N_z \log(N)\right)$ [16]. A related solver has been developed for the case of constant wave number, k, by Toivanen and Wolfmayr [10]. The lack of variable coefficient k makes this method insufficient for some of the applications under consideration.

If k is assumed to contain an inclusion or the equation (1.1) is coupled with Sommerfeld-like conditions on all boundaries, the resulting discretized system is neither positive definite nor Hermitian [17]. This system causes the convergence of most iterative methods to deteriorate as frequency increases [5, 18, 19]. Turkel approached a similar problem with an iterative method and was limited to a computational grid of only 403^2 and 402^3 in the two- and three-dimensional cases respectively [5]. In two-dimensions, this is immediately evident. With a grid of only 112^2 the iterations approached 3000 to reach the desired residual [5]. In this situation, the parallel direct approach is a better alternative.

The developed algorithms present a stable alternative to highly-efficient Krylov-type methods developed for subsurface scattering problems in the author's previous publication [12] and in [13]. However, if k is assumed to vary only in one spatial direction these direct solvers can be utilized as efficient preconditioners in some general non-constant coefficient problems as shown in [20, 20, 21]. Such an iterative solver in the second-order case for both two- and three-dimensions is provided by Gryazin in [6, 20]. A second-order trilinear finite element approximation for the case of constant k in three-dimensions was given by Elman in [21]. The MPI extension for Fortran90 was used to parallelize this implementation and demonstrated "a large amount of parallelism" [21]. This direct solver was utilized as a

preconditioner for the case with Sommerfeld-like boundary conditions with an average speed-up ranging from 1.77 to 1.90 times when doubling the number of processes. The quality of FFT preconditioners in scattering problems is presented in [22].

Although the focus is the solution of the Helmholtz equation, the numerical methods given here have a variety of applications. The FFT solver is developed to be a generalized solver for a wider class of linear systems. Systems obtained from approximations on three-dimensional 27-point stencils with similar constraints on the stencil coefficients form this class. As an example of the solver's versatility, the direct parallel implementation of a compact fourth-order scheme for a convection-diffusion equation is considered. Convection-diffusion with dominant vertical convection can be considered a first step in the development of a high-resolution parallel compact simulator for atmospheric flow [23]. General statements provided throughout refer to the Helmholtz problem unless explicitly stated that it refers to the convection-diffusion problem.

Numerical experiments with test problems demonstrate the high-efficiency of the parallel solvers' implementations. The second-, fourth- and sixth-order compact schemes are tested for convergence accuracy in the case of constant coefficient k with analytic solutions. In the case of variable k, the resulting residuals are presented to ensure solution accuracy. Parallel performance of these solvers is investigated using decreases in wall-time to measure speed-up. The performance of the FFT solver is compared to the results given by Turkel in [5] and the partial FFT solver is compared to the solver provided by Toivanen in [10].

Chapters 2 and 3 give the second-, fourth- and sixth-order finite difference approximations of the Helmholtz equations in two- and three-dimensions respectively. Chapter 3 considers the fourth-order discretization for the convection-diffusion equation. Chapter 4 states the

Dirichlet and Sommerfeld-like boundary conditions along with a simple two-dimensional example on their implementations. The FFT solver's derivation and parallel implementation is given in Chapter 5. Novel generalized eigenvalue and partial FFT solvers are provided in Chapters 6 and 7 respectively. Chapter 8 presents numerical results that show the efficiency of the solver's parallel implementations.

2 Two-Dimensional Discretization

Chapter 2 develops second-, fourth- and sixth-order compact finite difference schemes for the approximate solution of the two-dimensional Helmholtz equation on rectangular domains. To approximate the solution of (1.1) with boundary conditions (1.2) consider the computational grid

$$\Omega_h = \{(x_i, y_j) \in \mathbb{R}^2 \mid x_i = x_l + ih_x,\, y_j = y_l + jh_y, i = 1, \ldots, N_x,\, j = 1, \ldots, N_y\},$$

where $h_x = (x_u - x_l) / (N_x + 1)$ and $h_y = (y_u - y_l) / (N_y + 1)$ are the step sizes in the x and y directions respectively for Dirichlet boundary conditions. If Sommerfeld-like boundary conditions are considered, the step sizes are defined as $h_x = (x_u - x_l) / (N_x - 1)$ and $h_y = (y_u - y_l) / (N_y - 1)$. In these cases the step sizes are defined differently, however the same discretization applies. The following notation will be used for the first and second central differences at the (i, j)-th grid point

$$\delta_x u_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h_x}, \quad \delta_x^2 u_{i,j} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2}$$

where $u_{i,j} = u(x_i, y_j)$ and $u_{i\pm1,j} = u(x_i \pm h_x, y_j)$ [24].

2.1 Second-Order Accuracy

Taylor series expansion of u at each point (x_i, y_j) gives

$$u_{i\pm1,j} = u_{i,j} \pm h_x \frac{\partial}{\partial x} u_{i,j} + \frac{h_x^2}{2} \frac{\partial^2}{\partial x^2} u_{i,j} \pm \frac{h_x^3}{3!} \frac{\partial^3}{\partial x^3} u_{i,j} + \frac{h_x^4}{4!} \frac{\partial^4}{\partial x^4} u_{i,j} \pm \frac{h_x^5}{5!} \frac{\partial^5}{\partial x^5} u_{i,j} + \frac{h_x^6}{6!} \frac{\partial^6}{\partial x^6} u_{i,j} \ldots \quad (2.1)$$

for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$ [24]. The addition of $u_{i-1,j}$ and $u_{i+1,j}$ will cancel the odd terms, that is

$$u_{i-1,j} + u_{i+1,j} = 2u_{i,j} + h_x^2 \frac{\partial^2}{\partial x^2} u_{i,j} + \mathcal{O}\left(h_x^4\right).$$

Hence $\dfrac{\partial^2}{\partial x^2} u_{i,j} = \delta_x^2 u_{i,j} + \mathcal{O}\left(h_x^2\right)$. The same expansion can be done with respect to y. From these expansions, the following second-order scheme for the approximation of (1.1) is obtained,

$$f_{i,j} = \Delta u_{i,j} + k_{i,j}^2 u_{i,j}$$

$$f_{i,j} = \delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} + k_{i,j}^2 u_{i,j} + \mathcal{O}\left(\max\left\{h_x^2, h_y^2\right\}\right)$$

$$f_{i,j} = \frac{1}{h_y^2} u_{i,j-1} + \frac{1}{h_x^2} u_{i-1,j} + \left(k_{i,j}^2 - \frac{2}{h_x^2} - \frac{2}{h_y^2}\right) u_{i,j}$$

$$+ \frac{1}{h_x^2} u_{i+1,j} + \frac{1}{h_y^2} u_{i,j+1} + \mathcal{O}\left(\max\left\{h_x^2, h_y^2\right\}\right)$$

for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. This equation provides the following coefficients for a 9-point stencil

$$u_{i,j} : k_{i,j}^2 - \frac{2}{h_x^2} - \frac{2}{h_y^2}, \qquad u_{i\pm1,j} : \frac{1}{h_x^2}, \qquad u_{i,j\pm1} : \frac{1}{h_y^2}, \qquad u_{i\pm1,j\pm1} : 0.$$

2.2 Fourth-Order Accuracy

Here, the fourth-order term in the addition of $u_{i-1,j}$ and $u_{i+1,j}$ is included, that is

$$u_{i-1,j} + u_{i+1,j} = 2u_{i,j} + h_x^2 \frac{\partial^2}{\partial x^2} u_{i,j} + \frac{h_x^4}{12} \frac{\partial^4}{\partial x^4} u_{i,j} + \mathcal{O}\left(h_x^6\right)$$

$$\delta_x^2 u_{i,j} = \frac{\partial^2}{\partial x^2} u_{i,j} + \frac{h_x^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j} + \mathcal{O}\left(h_x^4\right)$$

$$\delta_x^2 u_{i,j} = \frac{\partial^2}{\partial x^2} u_{i,j} + \frac{h_x^2}{12} \delta_x^2 \frac{\partial^2}{\partial x^2} u_{i,j} + \mathcal{O}\left(h_x^4\right)$$

$$\delta_x^2 u_{i,j} = \left(1 + \frac{h_x^2}{12} \delta_x^2\right) \frac{\partial^2}{\partial x^2} u_{i,j} + \mathcal{O}\left(h_x^4\right)$$

$$\frac{\partial^2}{\partial x^2} u_{i,j} = \left(1 + \frac{h_x^2}{12} \delta_x^2\right)^{-1} \delta_x^2 u_{i,j} + \mathcal{O}\left(h_x^4\right)$$

$$\frac{\partial^2}{\partial x^2} u_{i,j} = \alpha_x^{-1} \delta_x^2 u_{i,j} + \mathcal{O}\left(h_x^4\right).$$

where $\alpha_x = (1 + h_x^2 \delta_x^2/12)$. The same expansion can be done with respect to y. The operators α_x and α_y are shown to commute by Lemma 2.1.

Lemma 2.1. *The operators* $\left(1 + \frac{h_x^2}{12}\delta_x^2\right)$ *and* $\left(1 + \frac{h_y^2}{12}\delta_y^2\right)$ *commute.*

Proof. Let u be a function of x and y. First note that the operators δ_x^2 and δ_y^2 commute since

$$\begin{aligned}
\delta_x^2 \delta_y^2 u_{i,j} &= \delta_x^2 \left(\frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2} \right) \\
&= \frac{1}{h_y^2} \frac{1}{h_x^2} \left([u_{i-1,j-1} - 2u_{i,j-1} + u_{i+1,j-1}] \right. \\
&\quad \left. -2\,[u_{i-1,j} - 2u_{i,j} + u_{i+1,j}] + [u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}] \right) \\
&= \frac{1}{h_x^2} \frac{1}{h_y^2} \left([u_{i-1,j-1} - 2u_{i-1,j} + u_{i-1,j+1}] \right. \\
&\quad \left. -2\,[u_{i,j-1} - 2u_{i,j} + u_{i,j+1}] + [u_{i+1,j-1} - 2u_{i+1,j} + u_{i+1,j+1}] \right) \\
&= \delta_y^2 \left(\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} \right) = \delta_y^2 \delta_x^2 u_{i,j}.
\end{aligned}$$

Thus

$$\left(1+\frac{h_x^2}{12}\delta_x^2\right)\left(1+\frac{h_y^2}{12}\delta_y^2\right)u_{i,j} = u_{i,j}+\frac{h_y^2}{12}\delta_y^2 u_{i,j}+\frac{h_x^2}{12}\delta_x^2 u_{i,j}+\frac{h_x^2}{12}\frac{h_y^2}{12}\delta_x^2\delta_y^2 u_{i,j}$$

$$= u_{i,j}+\frac{h_x^2}{12}\delta_x^2 u_{i,j}+\frac{h_y^2}{12}\delta_y^2 u_{i,j}+\frac{h_y^2}{12}\frac{h_x^2}{12}\delta_y^2\delta_x^2 u_{i,j}$$

$$= \left(1+\frac{h_y^2}{12}\delta_y^2\right)\left(1+\frac{h_x^2}{12}\delta_x^2\right)u_{i,j}. \qquad \blacksquare$$

The following is obtained by multiplying $\alpha_x\alpha_y$ to both sides of the equation:

$$\Delta u_{i,j} + k_{i,j}^2 u_{i,j} = f_{i,j}$$

$$\alpha_x^{-1}\delta_x^2 u_{i,j} + \alpha_y^{-1}\delta_y^2 u_{i,j} + k_{i,j}^2 u_{i,j} = f_{i,j} + \mathcal{O}\left(\max\left\{h_x^4, h_y^4\right\}\right)$$

$$\alpha_y\delta_x^2 u_{i,j} + \alpha_x\delta_y^2 u_{i,j} + \alpha_x\alpha_y k_j^2 u_{i,j} = \alpha_x\alpha_y f_{i,j} + \mathcal{O}\left(\max\left\{h_x^4, h_y^4\right\}\right). \tag{2.2}$$

The following fourth-order scheme for the approximation of (1.1) is obtained by multiplying out (2.2) and dropping all terms with $h_x^2 h_y^2$. This simplification is justified because the fourth-order approximation scheme is considered and higher-order terms are not needed. Then

$$\left(\delta_x^2+\delta_y^2+\frac{\left(h_x^2+h_y^2\right)}{12}\delta_x^2\delta_y^2\right)u_{i,j} + \left(1+\frac{h_x^2}{12}\delta_x^2+\frac{h_y^2}{12}\delta_y^2\right)k_{i,j}^2 u_{i,j}$$

$$= \left(1+\frac{h_x^2}{12}\delta_x^2+\frac{h_y^2}{12}\delta_y^2\right)f_{i,j} + \mathcal{O}\left(\max\left\{h_x^4, h_y^4\right\}\right)$$

for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. This equation provides the following coefficients for a

9-point stencil

$$u_{i,j} : -\frac{5}{3h_x^2} - \frac{5}{3h_y^2} + \frac{2k_{i,j}^2}{3}\,, \qquad u_{i\pm1,j\pm1} : \frac{1}{12h_y^2} + \frac{1}{12h_x^2}\,,$$

$$u_{i\pm1,j} : \frac{5}{6h_x^2} - \frac{1}{6h_y^2} + \frac{k_{i\pm1,j}^2}{12}\,, \qquad u_{i,j\pm1} : \frac{5}{6h_y^2} - \frac{1}{6h_x^2} + \frac{k_{i,j\pm1}^2}{12}.$$

2.3 Sixth-Order Accuracy

The following subsection gives the sixth-order approximation finite difference scheme, similar to that presented in [5]. Specific notation needed in the calculations followed by necessary lemmas with their proofs are given, concluding with the development of the approximation. For this scheme a uniform grid step in both spatial directions is assumed, that is $h = h_x = h_y$. Define the following notation

$$\nabla_h u_{i,j} = (\delta_x\,,\, \delta_y)\, u_{i,j}\,, \qquad \nabla_h^{1/2} u_{i,j} = \left(\delta_x\delta_y^2\,,\, \delta_x^2\delta_y\right) u_{i,j}\,, \qquad \Delta_h u_{i,j} = \left(\delta_x^2 + \delta_y^2\right) u_{i,j},$$

$$Lu = \Delta u + k^2 u, \quad \text{and} \quad L_h u_{i,j} = \Delta_h u_{i,j} + k_{i,j}^2 u_{i,j}$$

where u is a function of x and y. Note the sum of $u_{i+1,j}$ and $u_{i-1,j}$ from (2.1) gives the following

$$u_{i-1,j} + u_{i+1,j} = 2u_{i,j} + h^2\frac{\partial^2}{\partial x^2}u_{i,j} + \frac{h^4}{12}\frac{\partial^4}{\partial x^4}u_{i,j} + \frac{h^6}{360}\frac{\partial^6}{\partial x^6}u_{i,j} + \mathcal{O}\left(h^8\right)$$

$$\delta_x^2 u_{i,j} = \frac{\partial^2}{\partial x^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j} + \frac{h^4}{360}\frac{\partial^6}{\partial x^6}u_{i,j} + \mathcal{O}\left(h^6\right). \tag{2.3}$$

Lemma 2.2.

$$\frac{\partial}{\partial x}u_{i,j} = \delta_x u_{i,j} + \frac{h^2}{6}\left(\delta_x\delta_y^2 u_{i,j} + \delta_x\left(k^2 u\right)_{i,j} - \frac{\partial}{\partial x}f_{i,j}\right) + \mathcal{O}\left(h^4\right)$$

Proof. Differentiating (1.1) with respect to x gives

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u+k^2u=f$$

$$\frac{\partial^3}{\partial x^3}u+\frac{\partial^3}{\partial x\partial y^2}u+\frac{\partial}{\partial x}\left(k^2u\right)=\frac{\partial}{\partial x}f$$

$$\frac{\partial}{\partial x}f-\frac{\partial^3}{\partial x\partial y^2}u-\frac{\partial}{\partial x}\left(k^2u\right)=\frac{\partial^3}{\partial x^3}u.$$

The difference of $u_{i+1,j}$ and $u_{i-1,j}$ from (2.1) gives

$$\delta_x u_{i,j}=\frac{\partial}{\partial x}u_{i,j}+\frac{h^2}{6}\frac{\partial^3}{\partial x^3}u_{i,j}+\mathcal{O}\left(h^4\right)$$

$$\frac{\partial}{\partial x}u_{i,j}=\delta_x u_{i,j}-\frac{h^2}{6}\frac{\partial^3}{\partial x^3}u_{i,j}+\mathcal{O}\left(h^4\right)$$

$$\frac{\partial}{\partial x}u_{i,j}=\delta_x u_{i,j}-\frac{h^2}{6}\left(\frac{\partial}{\partial x}f_{i,j}-\frac{\partial^3}{\partial x\partial y^2}u_{i,j}-\frac{\partial}{\partial x}\left(k^2u\right)_{i,j}\right)+\mathcal{O}\left(h^4\right)$$

$$\frac{\partial}{\partial x}u_{i,j}=\delta_x u_{i,j}-\frac{h^2}{6}\left(\frac{\partial}{\partial x}f_{i,j}-\delta_x\delta_y^2u_{i,j}-\delta_x\left(k^2u\right)_{i,j}\right)+\mathcal{O}\left(h^4\right)$$

since $\delta_x\delta_y^2u_{i,j}$ and $\delta_x\left(k^2u\right)_{i,j}$ are second-order approximations. ■

The fourth-order approximation for $\dfrac{\partial}{\partial y}u_{i,j}$ is found similarly since the calculations in Lemma 2.2 are not dependent on the spatial direction.

Lemma 2.3.

$$\delta_x^2\delta_y^2u_{i,j}=\frac{\partial^4}{\partial x^2\partial y^2}u_{i,j}+\frac{h^2}{12}\left(\frac{\partial^6}{\partial x^4\partial y^2}+\frac{\partial^6}{\partial x^2\partial y^4}\right)u_{i,j}+\mathcal{O}\left(h^4\right)$$

Proof. The truncation of (2.3) to fourth-order gives

$$\delta_x^2u_{i,j}=\frac{\partial^2}{\partial x^2}u_{i,j}+\frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j}+\mathcal{O}\left(h^4\right). \tag{2.4}$$

It follows

$$\begin{aligned}
\delta_y^2\delta_x^2 u_{i,j} &= \delta_y^2\left(\frac{\partial^2}{\partial x^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j}\right) + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^2}{\partial y^2}\left(\frac{\partial^2}{\partial x^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j}\right) + \frac{h^2}{12}\frac{\partial^4}{\partial y^4}\left(\frac{\partial^2}{\partial x^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j}\right) + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^6}{\partial x^4\partial y^2}u_{i,j} + \frac{h^2}{12}\frac{\partial^6}{\partial x^2\partial y^4}u_{i,j} + \frac{h^4}{144}\frac{\partial^8}{\partial x^4\partial y^4}u_{i,j} + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j} + \frac{h^2}{12}\left(\frac{\partial^6}{\partial x^4\partial y^2} + \frac{\partial^6}{\partial x^2\partial y^4}\right)u_{i,j} + \mathcal{O}\left(h^4\right). \qquad \blacksquare
\end{aligned}$$

Lemma 2.4.

$$\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)\left(k^2u\right)_{i,j} = \frac{12}{h^2}\left(\Delta_h\left(k^2u\right)_{i,j} - \Delta(k^2u)_{i,j}\right) + \mathcal{O}\left(h^2\right)$$

Proof. Note that

$$\begin{aligned}
\delta_x^2\left(k^2u\right)_{i,j} &= \frac{\partial^2}{\partial x^2}\left(k^2u\right)_{i,j} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}\left(k^2u\right)_{i,j} + \mathcal{O}\left(h^4\right) \\
\frac{\partial^4}{\partial x^4}\left(k^2u\right)_{i,j} &= \frac{12}{h^2}\left(\delta_x^2\left(k^2u\right)_{i,j} - \frac{\partial^2}{\partial x^2}\left(k^2u\right)_{i,j}\right) + \mathcal{O}\left(h^2\right)
\end{aligned}$$

by (2.4). Hence

$$\begin{aligned}
\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)\left(k^2u\right)_{i,j} &= \frac{12}{h^2}\left(\delta_x^2\left(k^2u\right)_{i,j} - \frac{\partial^2}{\partial x^2}\left(k^2u\right)_{i,j}\right) \\
&\quad + \frac{12}{h^2}\left(\delta_y^2\left(k^2u\right)_{i,j} - \frac{\partial^2}{\partial y^2}\left(k^2u\right)_{i,j}\right) + \mathcal{O}\left(h^2\right) \\
\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)\left(k^2u\right)_{i,j} &= \frac{12}{h^2}\left(\left(\delta_x^2 + \delta_y^2\right)\left(k^2u\right)_{i,j} - \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)(k^2u)_{i,j}\right) + \mathcal{O}\left(h^2\right). \qquad \blacksquare
\end{aligned}$$

Lemma 2.5.

$$\Delta\left(k^2u\right)_{i,j} = \left(\Delta k^2_{i,j} - k^4_{i,j}\right)u_{i,j} + k^2_{i,j}f_{i,j}$$

$$+ 2\nabla k^2_{i,j}\cdot\left(\nabla_h u_{i,j} + \frac{h^2}{6}\left(\nabla_h^{1/2}u_{i,j} + \nabla_h\left(k^2u\right)_{i,j} - \nabla f_{i,j}\right)\right) + \mathcal{O}\left(h^4\right)$$

Proof. First observe that

$$\begin{aligned}
\Delta\left(k^2u\right) &= \frac{\partial^2}{\partial x^2}\left(k^2u\right) + \frac{\partial^2}{\partial y^2}\left(k^2u\right)\\
&= u\frac{\partial^2}{\partial x^2}k^2 + 2\left(\frac{\partial}{\partial x}k^2\right)\left(\frac{\partial}{\partial x}u\right) + k^2\frac{\partial^2}{\partial x^2}u\\
&\quad + u\frac{\partial^2}{\partial y^2}k^2 + 2\left(\frac{\partial}{\partial y}k^2\right)\left(\frac{\partial}{\partial y}u\right) + k^2\frac{\partial^2}{\partial y^2}u\\
&= u\Delta k^2 + 2\nabla k^2\cdot\nabla u + k^2\Delta u\\
&= u\Delta k^2 + 2\nabla k^2\cdot\nabla u + k^2\left(f - k^2u\right)\\
&= \left(\Delta k^2 - k^4\right)u + 2\nabla k^2\cdot\nabla u + k^2 f.
\end{aligned}$$

It follows

$$\begin{aligned}
\Delta\left(k^2u\right)_{i,j} &= \left(\Delta k^2_{i,j} - k^4_{i,j}\right)u_{i,j} + 2\nabla k^2_{i,j}\cdot\nabla u_{i,j} + k^2_{i,j}f_{i,j}\\
&= \left(\Delta k^2_{i,j} - k^4_{i,j}\right)u_{i,j} + k^2_{i,j}f_{i,j}\\
&\quad + 2\nabla k^2_{i,j}\cdot\left(\delta_x u_{i,j} + \frac{h^2}{6}\left(\delta_x\delta_y^2 u_{i,j} + \delta_x\left(k^2u\right)_{i,j} - \frac{\partial}{\partial x}f_{i,j}\right),\right.\\
&\quad \left.\delta_y u_{i,j} + \frac{h^2}{6}\left(\delta_y\delta_x^2 u_{i,j} + \delta_y\left(k^2u\right)_{i,j} - \frac{\partial}{\partial y}f_{i,j}\right)\right) + \mathcal{O}\left(h^4\right)
\end{aligned}$$

$$= \left(\Delta k_{i,j}^2 - k_{i,j}^4\right) u_{i,j} + k_{i,j}^2 f_{i,j}$$

$$+ 2\nabla k_{i,j}^2 \cdot \left(\nabla_h u_{i,j} + \frac{h^2}{6}\left(\nabla_h^{1/2} u_{i,j} + \nabla_h\left(k^2 u\right)_{i,j} - \nabla f_{i,j}\right)\right) + \mathcal{O}\left(h^4\right)$$

from Lemma 2.2. ■

Now, consider the sixth-order approximation to (1.1). The substitution of (2.3) and the corresponding equation for y gives

$$\begin{aligned} L_h u_{i,j} &= \Delta_h u_{i,j} + k_{i,j}^2 u_{i,j} \\ &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u_{i,j} + \frac{h^2}{12}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right) u_{i,j} + \frac{h^4}{360}\left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right) u_{i,j} \\ &\quad + k_{i,j}^2 u_{i,j} + \mathcal{O}\left(h^6\right) \\ &= L u_{i,j} + \frac{h^2}{12}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right) u_{i,j} + \frac{h^4}{360}\left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right) u_{i,j} + \mathcal{O}\left(h^6\right). \end{aligned}$$

It follows

$$f_{i,j} = L u_{i,j} = L_h u_{i,j} - \frac{h^2}{12}\beta_4 - \frac{h^4}{360}\beta_6 + \mathcal{O}\left(h^6\right) \tag{2.5}$$

where $\beta_4 = \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right) u_{i,j}$ and $\beta_6 = \left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right) u_{i,j}$. The coefficients of β_4 and β_6 show that their approximations must be fourth- and second-order, respectively.

2.3.1 Fourth-Order Approximation of β_4

As defined in (2.5), $\beta_4 = \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right) u_{i,j}$. To approximate β_4 with fourth-order accuracy, consider the second and fourth partial derivatives of the equation (1.1):

$$\frac{\partial^4}{\partial x^4} u + \frac{\partial^4}{\partial x^2 \partial y^2} u + \frac{\partial^2}{\partial x^2}\left(k^2 u\right) = \frac{\partial^2}{\partial x^2} f$$

$$\frac{\partial^4}{\partial x^2 \partial y^2} u + \frac{\partial^4}{\partial y^4} u + \frac{\partial^2}{\partial y^2} \left(k^2 u \right) = \frac{\partial^2}{\partial y^2} f$$

$$\frac{\partial^6}{\partial x^4 \partial y^2} u + \frac{\partial^6}{\partial x^2 \partial y^4} u + \frac{\partial^4}{\partial x^2 \partial y^2} \left(k^2 u \right) = \frac{\partial^4}{\partial x^2 \partial y^2} f.$$

It follows

$$\frac{\partial^4}{\partial x^4} u = \frac{\partial^2}{\partial x^2} f - \frac{\partial^4}{\partial x^2 \partial y^2} u - \frac{\partial^2}{\partial x^2} \left(k^2 u \right)$$

$$\frac{\partial^4}{\partial y^4} u = \frac{\partial^2}{\partial y^2} f - \frac{\partial^4}{\partial x^2 \partial y^2} u - \frac{\partial^2}{\partial y^2} \left(k^2 u \right)$$

$$\frac{\partial^6}{\partial x^4 \partial y^2} u + \frac{\partial^6}{\partial x^2 \partial y^4} u = \frac{\partial^4}{\partial x^2 \partial y^2} f - \frac{\partial^4}{\partial x^2 \partial y^2} \left(k^2 u \right) \tag{2.6}$$

Then

$$\begin{aligned}
\beta_4 &= \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} \right) u_{i,j} \\
&= \frac{\partial^2}{\partial x^2} f_{i,j} - \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} - \frac{\partial^2}{\partial x^2} \left(k^2 u \right)_{i,j} + \frac{\partial^2}{\partial y^2} f_{i,j} - \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} - \frac{\partial^2}{\partial y^2} \left(k^2 u \right)_{i,j} \\
&= \Delta f_{i,j} - 2 \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} - \Delta \left(k^2 u \right)_{i,j}.
\end{aligned}$$

Lemma 2.3 and equation (2.6) give

$$\begin{aligned}
\delta_x^2 \delta_y^2 u_{i,j} &= \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} + \frac{h^2}{12} \left(\frac{\partial^6}{\partial x^4 \partial y^2} + \frac{\partial^6}{\partial x^2 \partial y^4} \right) u_{i,j} + \mathcal{O}\left(h^4 \right) \\
&= \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} + \frac{h^2}{12} \left(\frac{\partial^4}{\partial x^2 \partial y^2} f_{i,j} - \frac{\partial^4}{\partial x^2 \partial y^2} \left(k^2 u \right)_{i,j} \right) + \mathcal{O}\left(h^4 \right) \\
\frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j} &= \delta_x^2 \delta_y^2 u_{i,j} - \frac{h^2}{12} \left(\frac{\partial^4}{\partial x^2 \partial y^2} f_{i,j} - \frac{\partial^4}{\partial x^2 \partial y^2} \left(k^2 u \right)_{i,j} \right) + \mathcal{O}\left(h^4 \right)
\end{aligned}$$

$$= \delta_x^2\delta_y^2 u_{i,j} + \frac{h^2}{12}\left(\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j}\right) + \mathcal{O}\left(h^4\right)$$

since $\delta_x^2\delta_y^2\left(k^2u\right)$ is a second-order approximation. Finally,

$$\beta_4 = \Delta f_{i,j} - 2\frac{\partial^4}{\partial x^2\partial y^2}u_{i,j} - \Delta\left(k^2u\right)_{i,j}$$

$$= \Delta f_{i,j} - 2\delta_x^2\delta_y^2 u_{i,j} - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j}\right) - \Delta\left(k^2u\right)_{i,j} + \mathcal{O}\left(h^4\right).$$

2.3.2 Second-Order Approximation of β_6

As defined in (2.5), $\beta_6 = \left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right)u_{i,j}$. To approximate β_6 with second-order accuracy, consider the fourth derivatives of the equation (1.1),

$$\frac{\partial^6}{\partial x^6}u + \frac{\partial^6}{\partial x^4\partial y^2}u + \frac{\partial^4}{\partial x^4}\left(k^2u\right) = \frac{\partial^4}{\partial x^4}f$$

$$\frac{\partial^4}{\partial x^4}f - \frac{\partial^6}{\partial x^4\partial y^2}u - \frac{\partial^4}{\partial x^4}\left(k^2u\right) = \frac{\partial^6}{\partial x^6}u$$

with $\frac{\partial^6}{\partial y^6}u$ found similarly. It follows

$$\beta_6 = \left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6}\right)u$$

$$= \frac{\partial^4}{\partial x^4}f - \frac{\partial^6}{\partial x^4\partial y^2}u - \frac{\partial^4}{\partial x^4}\left(k^2u\right) + \frac{\partial^4}{\partial y^4}f - \frac{\partial^6}{\partial x^2\partial y^4}u - \frac{\partial^4}{\partial y^4}\left(k^2u\right)$$

$$= -\left(\frac{\partial^6}{\partial x^4\partial y^2}u + \frac{\partial^6}{\partial x^2\partial y^4}u\right) - \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)\left(k^2u\right) + \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)f$$

$$= \frac{\partial^4}{\partial x^2\partial y^2}k^2u - \frac{\partial^4}{\partial x^2\partial y^2}f - \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)\left(k^2u\right) + \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right)f.$$

by (2.6). Lemma 2.4 gives the second-order approximation as follows

$$\beta_6 = \delta_x^2\delta_y^2\left(k^2u\right)_{i,j} - \frac{12}{h^2}\left(\Delta_h\left(k^2u\right)_{i,j} - \Delta\left(k^2u\right)_{i,j}\right)$$
$$+\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} - \frac{\partial^4}{\partial x^2\partial y^2}\right)f_{i,j} + \mathcal{O}\left(h^2\right).$$

The approximated β_4 and β_6 substituted into (2.5) along with Lemma 2.5 gives

$$L_hu_{i,j} - \frac{h^2}{12}\left[\Delta f_{i,j} - 2\delta_x^2\delta_y^2u_{i,j} - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j}\right) - \Delta\left(k^2u\right)_{i,j} + \mathcal{O}\left(h^4\right)\right]$$
$$- \frac{h^4}{360}\left[\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} - \frac{12}{h^2}\left(\Delta_h\left(k^2u\right)_{i,j} - \Delta\left(k^2u\right)_{i,j}\right)\right.$$
$$\left.+\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} - \frac{\partial^4}{\partial x^2\partial y^2}\right)f_{i,j} + \mathcal{O}\left(h^2\right)\right] + \mathcal{O}\left(h^6\right)$$
$$= L_hu_{i,j} + \frac{h^2}{6}\delta_x^2\delta_y^2u_{i,j} + \frac{h^4}{90}\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} + \frac{h^2}{30}\Delta_h\left(k^2u\right)_{i,j}$$
$$+\frac{h^2}{20}\left[\left(\Delta k_{i,j}^2 - k_{i,j}^4\right)u_{i,j} + k_{i,j}^2f_{i,j}\right.$$
$$\left.+2\nabla k_{i,j}^2\cdot\left(\nabla_hu_{i,j} + \frac{h^2}{6}\left(\nabla_h^{1/2}u_{i,j} + \nabla_h\left(k^2u\right)_{i,j} - \nabla f_{i,j}\right)\right) + \mathcal{O}\left(h^4\right)\right]$$
$$- \frac{h^4}{360}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + 4\frac{\partial^4}{\partial x^2\partial y^2}\right)f_{i,j} - \frac{h^2}{12}\Delta f_{i,j} + \mathcal{O}\left(h^6\right)$$
$$= L_hu_{i,j} + \frac{h^2}{6}\delta_x^2\delta_y^2u_{i,j} + \frac{h^4}{90}\delta_x^2\delta_y^2\left(k^2u\right)_{i,j} + \frac{h^2}{30}\Delta_h\left(k^2u\right)_{i,j}$$
$$+\frac{h^2}{20}\left(\Delta k_{i,j}^2 - k_{i,j}^4\right)u_{i,j} + \frac{h^2}{10}\nabla k_{i,j}^2\cdot\left(\nabla_hu_{i,j} + \frac{h^2}{6}\left(\nabla_h^{1/2}u_{i,j} + \nabla_h\left(k^2u\right)_{i,j}\right)\right)$$
$$- \frac{h^4}{360}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + 4\frac{\partial^4}{\partial x^2\partial y^2}\right)f_{i,j} - \frac{h^2}{12}\Delta f_{i,j} + \frac{h^2}{20}k_{i,j}^2f_{i,j} - \frac{h^4}{60}\nabla k_{i,j}^2\cdot\nabla f_{i,j} + \mathcal{O}\left(h^6\right)$$
$$= f_{i,j}.$$

Thus the sixth-order compact approximation is given by

$$L_h u_{i,j} + \frac{h^2}{6}\delta_x^2\delta_y^2 u_{i,j} + \frac{h^4}{90}\delta_x^2\delta_y^2\left(k^2 u\right)_{i,j} + \frac{h^2}{30}\Delta_h\left(k^2 u\right)_{i,j}$$
$$+ \frac{h^2}{20}\left(\Delta k_{i,j}^2 - k_{i,j}^4\right) u_{i,j} + \frac{h^2}{10}\nabla k_{i,j}^2 \cdot \left(\nabla_h u_{i,j} + \frac{h^2}{6}\left(\nabla_h^{1/2} u_{i,j} + \nabla_h\left(k^2 u\right)_{i,j}\right)\right)$$

with the right-hand side

$$\left(1 - \frac{h^2}{20}k_{i,j}^2\right) f_{i,j} + \frac{h^4}{360}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + 4\frac{\partial^4}{\partial x^2 \partial y^2}\right) f_{i,j} + \frac{h^2}{12}\Delta f_{i,j} + \frac{h^4}{60}\nabla k_{i,j}^2 \cdot \nabla f_{i,j}.$$

for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. This scheme provides the following coefficients for a 9-point stencil

$$u_{i,j} : -\frac{10}{3h^2} + \frac{41k_{i,j}^2}{45} + \frac{h^2}{20}\left(\Delta k_{i,j}^2 - k_{i,j}^4\right),$$

$$u_{i\pm1,j} : \frac{2}{3h^2} + \frac{k_{i\pm1,j}^2}{90} \pm \left(\frac{h}{30} + \frac{h^3 k_{i\pm1,j}^2}{120}\right)\frac{\partial}{\partial x}k_{i,j}^2,$$

$$u_{i,j\pm1} : \frac{2}{3h^2} + \frac{k_{i,j\pm1}^2}{90} \pm \left(\frac{h}{30} + \frac{h^3 k_{i,j\pm1}^2}{120}\right)\frac{\partial}{\partial y}k_{i,j}^2,$$

$$u_{i\pm1,j-1} : \frac{1}{6h^2} + \frac{k_{i\pm1,j-1}^2}{90} \pm \frac{h}{120}\frac{\partial}{\partial x}k_{i,j}^2 - \frac{h}{120}\frac{\partial}{\partial y}k_{i,j}^2,$$

$$u_{i\pm1,j+1} : \frac{1}{6h^2} + \frac{k_{i\pm1,j+1}^2}{90} \pm \frac{h}{120}\frac{\partial}{\partial x}k_{i,j}^2 + \frac{h}{120}\frac{\partial}{\partial y}k_{i,j}^2.$$

2.4 Summary

Chapter 2 developed the second-, fourth- and sixth-order approximation schemes for the numerical solution of the two-dimensional Helmholtz equation. In addition, the nine-point stencil coefficients were presented. Chapter 3 develops approximation schemes for the three-dimensional Helmholtz and convection-diffusion equations.

3 Three-Dimensional Discretization

Chapter 3 develops second-, fourth- and sixth-order compact finite difference schemes for the numerical solution of the three-dimensional Helmholtz equation on rectangular domains. To approximate the solution of (1.1) with boundary conditions (1.2), consider the computational grid

$$\Omega_h = \{(x_i, y_j, z_l) \in \mathbb{R}^3 \mid x_i = x_l + ih_x,\ y_j = y_l + jh_y,\ z_l = z_l + lh_z,$$
$$i = 1, \ldots, N_x,\ j = 1, \ldots, N_y,\ l = 1, \ldots, N_z\}, \tag{3.1}$$

where $h_x = (x_u - x_l)/(N_x + 1)$, $h_y = (y_u - y_l)/(N_y + 1)$ and $h_z = (z_u - z_l)/(N_z + 1)$ are the step sizes in the x, y and z directions respectively for Dirichlet boundary conditions. If Sommerfeld-like boundary conditions are considered, the step sizes are defined as $h_x = (x_u - x_l)/(N_x - 1)$, $h_y = (y_u - y_l)/(N_y - 1)$ and $h_z = (z_u - z_l)/(N_z - 1)$. Regardless of the step size definition, the same discretization applies. The following notation will be used for the first and second central differences at the (i, j, l)-th grid point

$$\delta_x u_{i,j,l} = \frac{u_{i+1,j,l} - u_{i-1,j,l}}{2h_x}, \quad \delta_x^2 u_{i,j,l} = \frac{u_{i-1,j,l} - 2u_{i,j,l} + u_{i+1,j,l}}{h_x^2}$$

where $u_{i,j,l} = u(x_i, y_j, z_l)$ and $u_{i\pm1,j,l} = u(x_i \pm h_x, y_j, z_l)$ [24]. The operators δ_y, δ_z, δ_y^2 and δ_z^2 are defined similarly. In addition, the chapter introduces a fourth-order scheme for the convection-diffusion equation with Dirichlet boundary conditions. Note that many of the calculations needed for the development of these schemes are similar to those in the previous chapter for the two-dimensional case and are therefore omitted.

3.1 Second-Order Accuracy

Taylor series expansion of u at each point (x_i, y_j, z_l) gives

$$u_{i\pm1,j,l} = u_{i,j,l} \pm h_x \frac{\partial}{\partial x} u_{i,j,l} + \frac{h_x^2}{2} \frac{\partial^2}{\partial x^2} u_{i,j,l} \pm \frac{h_x^3}{3!} \frac{\partial^3}{\partial x^3} u_{i,j,l} + \frac{h_x^4}{4!} \frac{\partial^4}{\partial x^4} u_{i,j,l} \pm \frac{h_x^5}{5!} \frac{\partial^5}{\partial x^5} u_{i,j,l} + \frac{h_x^6}{6!} \frac{\partial^6}{\partial x^6} u_{i,j,l} \ldots \tag{3.2}$$

for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $l = 1, \ldots, N_z$ [24]. The addition of $u_{i-1,j,l}$ and $u_{i+1,j,l}$ will cancel the odd terms, that is

$$u_{i-1,j,l} + u_{i+1,j,l} = 2u_{i,j,l} + h_x^2 \frac{\partial^2}{\partial x^2} u_{i,j,l} + \mathcal{O}\left(h_x^4\right).$$

Hence $\dfrac{\partial^2}{\partial x^2} u_{i,j,l} = \delta_x^2 u_{i,j,l} + \mathcal{O}\left(h_x^2\right)$. The same expansion can be done with respect to y and z. From these, the following second-order scheme for the approximation of (1.1) is obtained,

$$\begin{aligned} f_{i,j,l} = {} & \frac{1}{h_x^2} u_{i-1,j,l} + \frac{1}{h_y^2} u_{i,j-1,l} + \frac{1}{h_z^2} u_{i,j,l-1} + \left(k_{i,j,l}^2 - \frac{2}{h_x^2} - \frac{2}{h_y^2} - \frac{2}{h_z^2}\right) u_{i,j,l} \\ & + \frac{1}{h_x^2} u_{i+1,j,l} + \frac{1}{h_y^2} u_{i,j+1,l} + \frac{1}{h_z^2} u_{i,j,l+1} + \mathcal{O}\left(\max\left\{h_x^2, h_y^2, h_z^2\right\}\right) \end{aligned} \tag{3.3}$$

for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $l = 1, \ldots, N_z$. This scheme provides the following coefficients for a 27-point stencil

$$u_{i,j,l} : k_{i,j,l}^2 - \frac{2}{h_x^2} - \frac{2}{h_y^2} - \frac{2}{h_z^2}, \qquad u_{i\pm1,j,l} : \frac{1}{h_x^2}, \qquad u_{i,j\pm1,l} : \frac{1}{h_y^2}, \qquad u_{i,j,l\pm1} : \frac{1}{h_z^2},$$

$$u_{i\pm1,j\pm1,l\pm1} = u_{i,j\pm1,l\pm1} = u_{i\pm1,j,l\pm1} = u_{i\pm1,j\pm1,l} : 0.$$

3.2 Fourth-Order Accuracy

The fourth-order approximation given in Section 3.2 uses a strategy similar that of Lele [4]. Here, the fourth-order term in the addition of $u_{i-1,j,l}$ and $u_{i+1,j,l}$ is included, that is

$$u_{i-1,j,l} + u_{i+1,j,l} = 2u_{i,j,l} + h_x^2 \frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h_x^4}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} + \mathcal{O}\left(h_x^6\right)$$

$$\delta_x^2 u_{i,j,l} = \frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h_x^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} + \mathcal{O}\left(h_x^4\right)$$

$$\frac{\partial^2}{\partial x^2} u_{i,j,l} = \left(1 + \frac{h_x^2}{12}\delta_x^2\right)^{-1} \delta_x^2 u_{i,j,l} + \mathcal{O}\left(h_x^4\right)$$

$$\frac{\partial^2}{\partial x^2} u_{i,j,l} = \alpha_x^{-1} \delta_x^2 u_{i,j,l} + \mathcal{O}\left(h_x^4\right).$$

where $\alpha_x = (1 + h_x^2\delta_x^2/12)$. The same expansion can be done with respect to y and z. Note that Lemma 2.1 had no dependency on the spatial direction or dimension, so the operators α_x, α_y and α_z commute. The following is obtained by multiplying $\alpha_x\alpha_y\alpha_z$ to both sides of (1.1)

$$\alpha_y\alpha_z\delta_x^2 u_{i,j,l} + \alpha_x\alpha_z\delta_y^2 u_{i,j,l} + \alpha_x\alpha_y\delta_z^2 u_{i,j,l} + \alpha_x\alpha_y\alpha_z k_{i,j,l}^2 u_{i,j,l} = \alpha_x\alpha_y\alpha_z f_{i,j,l} + \mathcal{O}\left(\max\left\{h_x^4, h_y^4, h_z^4\right\}\right) \quad (3.4)$$

The following fourth-order scheme for the approximation of (1.1) is obtained by multiplying out and dropping all terms with $h_x^2h_y^2$, $h_x^2h_z^2$, $h_y^2h_z^2$ and $h_x^2h_y^2h_z^2$. This is justified because the fourth-order approximation scheme is considered and only the second-order terms need to remain. Consider first

$$\alpha_y\alpha_z\delta_x^2 u_{i,j,l} = \left(1 + h_y^2\frac{\delta_y^2}{12}\right)\left(1 + h_z^2\frac{\delta_z^2}{12}\right)\delta_x^2 u_{i,j,l}$$

$$= \left(\delta_x^2 + h_y^2 \frac{\delta_x^2 \delta_y^2}{12} + h_z^2 \frac{\delta_x^2 \delta_z^2}{12}\right) u_{i,j,l}.$$

Also,

$$\begin{aligned}
\alpha_x \alpha_y \alpha_z k_{i,j,l}^2 u_{i,j,l} &= \left(1 + h_x^2 \frac{\delta_x^2}{12}\right)\left(1 + h_y^2 \frac{\delta_y^2}{12}\right)\left(1 + h_z^2 \frac{\delta_z^2}{12}\right) k_{i,j,l}^2 u_{i,j,l} \\
&= \left(1 + h_x^2 \frac{\delta_x^2}{12}\right)\left(1 + h_y^2 \frac{\delta_y^2}{12} + h_z^2 \frac{\delta_z^2}{12}\right) k_{i,j,l}^2 u_{i,j,l} \\
&= \left(1 + h_x^2 \frac{\delta_x^2}{12} + h_y^2 \frac{\delta_y^2}{12} + h_z^2 \frac{\delta_z^2}{12} + h_x^2 h_y^2 \frac{\delta_x^2 \delta_y^2}{144} + h_x^2 h_z^2 \frac{\delta_x^2 \delta_z^2}{144}\right) k_{i,j,l}^2 u_{i,j,l} \\
&= \left(1 + h_x^2 \frac{\delta_x^2}{12} + h_y^2 \frac{\delta_y^2}{12} + h_z^2 \frac{\delta_z^2}{12}\right) k_{i,j,l}^2 u_{i,j,l}.
\end{aligned}$$

Substituting these and the analogous terms into (3.4) gives

$$\begin{aligned}
&\left(1 + h_x^2 \frac{\delta_x^2}{12} + h_y^2 \frac{\delta_y^2}{12} + h_z^2 \frac{\delta_z^2}{12}\right) f_{i,j,l} = \left(1 + h_x^2 \frac{\delta_x^2}{12} + h_y^2 \frac{\delta_y^2}{12} + h_z^2 \frac{\delta_z^2}{12}\right) k_{i,j,l}^2 u_{i,j,l} \\
&+ \left(\delta_x^2 + h_y^2 \frac{\delta_x^2 \delta_y^2}{12} + h_z^2 \frac{\delta_x^2 \delta_z^2}{12}\right) u_{i,j,l} + \left(\delta_y^2 + h_x^2 \frac{\delta_x^2 \delta_y^2}{12} + h_z^2 \frac{\delta_y^2 \delta_z^2}{12}\right) u_{i,j,l} \\
&+ \left(\delta_z^2 + h_x^2 \frac{\delta_x^2 \delta_z^2}{12} + h_y^2 \frac{\delta_y^2 \delta_z^2}{12}\right) u_{i,j,l} + \mathcal{O}\left(\max\left\{h_x^4, h_y^4, h_z^4\right\}\right)
\end{aligned}$$

for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $l = 1, \ldots, N_z$. This scheme provides the following coefficients for a 27-point stencil

$$u_{i,j,l} : \frac{k_{i,j,l}^2}{2} - \frac{4}{3}\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right), \qquad u_{i\pm1,j\pm1,l\pm1} : 0\,,$$

$$u_{i\pm1,j,l} : \frac{k_{i\pm1,j,l}^2}{12} + \frac{2}{3h_x^2} - \frac{1}{6}\left(\frac{1}{h_y^2} + \frac{1}{h_z^2}\right), \qquad u_{i,j\pm1,l} : \frac{k_{i,j\pm1,l}^2}{12} + \frac{2}{3h_y^2} - \frac{1}{6}\left(\frac{1}{h_x^2} + \frac{1}{h_z^2}\right),$$

$$u_{i,j,l\pm1} : \frac{k_{i,j,l\pm1}^2}{12} + \frac{2}{3h_z^2} - \frac{1}{6}\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right),$$

$$u_{i,j\pm1,l\pm1} : \frac{1}{12}\left(\frac{1}{h_y^2}+\frac{1}{h_z^2}\right), \qquad u_{i\pm1,j,l\pm1} : \frac{1}{12}\left(\frac{1}{h_x^2}+\frac{1}{h_z^2}\right), \qquad u_{i\pm1,j\pm1,l} : \frac{1}{12}\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right).$$

3.3 Sixth-Order Accuracy

Section 3.3 derives the sixth-order approximation finite difference scheme in the three-dimensional case. The work closely follows that presented in [5]. After establishing notation, the necessary lemmas with their proofs are given, followed by the development of the approximation schemes. For this scheme a uniform grid step in both spatial directions is assumed, that is $h = h_x = h_y = h_z$. Define the following notation,

$$\Delta_h u_{i,j,l} = \left(\delta_x^2+\delta_y^2+\delta_z^2\right)u_{i,j,l}\,, \qquad L_h u_{i,j,l} = (\Delta_h + k_{i,j,l}^2)u_{i,j,l} \qquad \nabla_h u_{i,j,l} = (\delta_x\,,\,\delta_y\,,\,\delta_z)\,u_{i,j,l}\,,$$

$$\nabla_h^{1/2} u_{i,j,l} = \left(\delta_x\delta_y^2\,,\,\delta_x^2\delta_y\,,\,\delta_x^2\delta_z\right)u_{i,j,l} + \left(\delta_x\delta_z^2\,,\,\delta_y\delta_z^2\,,\,\delta_y^2\delta_z\right)u_{i,j,l}\,,$$

$$\nabla^4 u = \left(\frac{\partial^4}{\partial x^4}+\frac{\partial^4}{\partial y^4}+\frac{\partial^4}{\partial z^4}\right)u\,, \qquad \nabla^6 u = \left(\frac{\partial^6}{\partial x^6}+\frac{\partial^6}{\partial y^6}+\frac{\partial^6}{\partial z^6}\right)u.$$

Note the sum of $u_{i+1,j,l}$ and $u_{i-1,j,l}$ from (3.2) gives the following

$$u_{i-1,j,l}+u_{i+1,j,l} = 2u_{i,j,l} + h^2\frac{\partial^2}{\partial x^2}u_{i,j,l} + \frac{h^4}{12}\frac{\partial^4}{\partial x^4}u_{i,j,l} + \frac{h^6}{360}\frac{\partial^6}{\partial x^6}u_{i,j,l} + \mathcal{O}\left(h^8\right)$$

$$\delta_x^2 u_{i,j,l} = \frac{\partial^2}{\partial x^2}u_{i,j,l} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}u_{i,j,l} + \frac{h^4}{360}\frac{\partial^6}{\partial x^6}u_{i,j,l} + \mathcal{O}\left(h^6\right). \tag{3.5}$$

Lemma 3.1.

$$\frac{\partial}{\partial x}u_{i,j,l} = \delta_x u_{i,j,l} + \frac{h^2}{6}\left(\delta_x\delta_y^2 u_{i,j,l} + \delta_x\delta_z^2 u_{i,j,l} + \delta_x\left(k^2u\right)_{i,j,l} - \frac{\partial}{\partial x}f_{i,j,l}\right) + \mathcal{O}\left(h^4\right)$$

Proof. Differentiating (1.1) with respect to x gives

$$\frac{\partial}{\partial x}f - \frac{\partial^3}{\partial x\partial y^2}u - \frac{\partial^3}{\partial x\partial z^2}u - \frac{\partial}{\partial x}\left(k^2u\right) = \frac{\partial^3}{\partial x^3}u.$$

The difference of $u_{i+1,j,l}$ and $u_{i-1,j,l}$ from (3.2) gives

$$\delta_x u_{i,j,l} = \frac{\partial}{\partial x} u_{i,j,l} + \frac{h^2}{6} \frac{\partial^3}{\partial x^3} u_{i,j,l} + \mathcal{O}\left(h^4\right)$$

$$\frac{\partial}{\partial x} u_{i,j,l} = \delta_x u_{i,j,l} - \frac{h^2}{6} \left(\frac{\partial}{\partial x} f_{i,j,l} - \frac{\partial^3}{\partial x \partial y^2} u_{i,j,l} - \frac{\partial^3}{\partial x \partial z^2} u_{i,j,l} - \frac{\partial}{\partial x} \left(k^2 u\right)_{i,j,l} \right) + \mathcal{O}\left(h^4\right)$$

$$\frac{\partial}{\partial x} u_{i,j,l} = \delta_x u_{i,j,l} - \frac{h^2}{6} \left(\frac{\partial}{\partial x} f_{i,j,l} - \delta_x \delta_y^2 u_{i,j,l} - \delta_x \delta_z^2 u_{i,j,l} - \delta_x \left(k^2 u\right)_{i,j,l} \right) + \mathcal{O}\left(h^4\right)$$

since $\delta_x \delta_y^2 u_{i,j,l}$, $\delta_x \delta_z^2 u_{i,j,l}$ and $\delta_x \left(k^2 u\right)_{i,j,l}$ are second-order approximations. ■

The fourth-order approximations for $\frac{\partial}{\partial y} u_{i,j,l}$ and $\frac{\partial}{\partial z} u_{i,j,l}$ are found in the same way since the calculations in Lemma 3.1 are not dependent on the spatial direction.

Lemma 3.2.

$$\delta_x^2 \delta_y^2 u_{i,j,l} = \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j,l} + \frac{h^2}{12} \left(\frac{\partial^6}{\partial x^4 \partial y^2} + \frac{\partial^6}{\partial x^2 \partial y^4} \right) u_{i,j,l} + \mathcal{O}\left(h^4\right)$$

Proof. The truncation of (3.5) to fourth-order gives

$$\delta_x^2 u_{i,j,l} = \frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} + \mathcal{O}\left(h^4\right). \tag{3.6}$$

It follows

$$\begin{aligned}
\delta_y^2 \delta_x^2 u_{i,j,l} &= \delta_y^2 \left(\frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} \right) + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^2}{\partial y^2} \left(\frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} \right) + \frac{h^2}{12} \frac{\partial^4}{\partial y^4} \left(\frac{\partial^2}{\partial x^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} \right) + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^6}{\partial x^4 \partial y^2} u_{i,j,l} + \frac{h^2}{12} \frac{\partial^6}{\partial x^2 \partial y^4} u_{i,j,l} + \frac{h^4}{144} \frac{\partial^8}{\partial x^4 \partial y^4} u_{i,j,l} + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^4}{\partial x^2 \partial y^2} u_{i,j,l} + \frac{h^2}{12} \left(\frac{\partial^6}{\partial x^4 \partial y^2} + \frac{\partial^6}{\partial x^2 \partial y^4} \right) u_{i,j,l} + \mathcal{O}\left(h^4\right).
\end{aligned}$$

■

Lemma 3.3.

$$\nabla^4 \left(k^2 u\right)_{i,j,l} = \frac{12}{h^2}\left(\Delta_h \left(k^2 u\right)_{i,j,l} - \Delta(k^2 u)_{i,j,l}\right) + \mathcal{O}\left(h^2\right)$$

Proof. Note that

$$\delta_x^2 \left(k^2 u\right)_{i,j,l} = \frac{\partial^2}{\partial x^2}\left(k^2 u\right)_{i,j,l} + \frac{h^2}{12}\frac{\partial^4}{\partial x^4}\left(k^2 u\right)_{i,j,l} + \mathcal{O}\left(h^4\right)$$

$$\frac{\partial^4}{\partial x^4}\left(k^2 u\right)_{i,j,l} = \frac{12}{h^2}\left(\delta_x^2 \left(k^2 u\right)_{i,j,l} - \frac{\partial^2}{\partial x^2}\left(k^2 u\right)_{i,j,l}\right) + \mathcal{O}\left(h^2\right)$$

by (3.6). Hence

$$\begin{aligned}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + \frac{\partial^4}{\partial z^4}\right)\left(k^2 u\right)_{i,j,l} = {} & \frac{12}{h^2}\left(\delta_x^2 \left(k^2 u\right)_{i,j,l} - \frac{\partial^2}{\partial x^2}\left(k^2 u\right)_{i,j,l}\right) \\ & + \frac{12}{h^2}\left(\delta_y^2 \left(k^2 u\right)_{i,j,l} - \frac{\partial^2}{\partial y^2}\left(k^2 u\right)_{i,j,l}\right) \\ & + \frac{12}{h^2}\left(\delta_z^2 \left(k^2 u\right)_{i,j,l} - \frac{\partial^2}{\partial z^2}\left(k^2 u\right)_{i,j,l}\right) + \mathcal{O}\left(h^2\right)\end{aligned}$$

$$\begin{aligned}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + \frac{\partial^4}{\partial z^4}\right)\left(k^2 u\right)_{i,j} = {} & \frac{12}{h^2}\left(\left(\delta_x^2 + \delta_y^2 + \delta_z^2\right)\left(k^2 u\right)_{i,j}\right. \\ & \left. - \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)(k^2 u)_{i,j}\right) + \mathcal{O}\left(h^2\right). \qquad \blacksquare\end{aligned}$$

Lemma 3.4.

$$\begin{aligned}\Delta\left(k^2 u\right)_{i,j,l} = {} & \left(\Delta k_{i,j,l}^2 - k_{i,j,l}^4\right) u_{i,j,l} + k_{i,j,l}^2 f_{i,j,l} \\ & + 2\nabla k_{i,j,l}^2 \cdot \left(\nabla_h u_{i,j,l} + \frac{h^2}{6}\left(\nabla_h^{1/2} u_{i,j,l} + \nabla_h \left(k^2 u\right)_{i,j,l} - \nabla f_{i,j,l}\right)\right) + \mathcal{O}\left(h^4\right)\end{aligned}$$

Proof. First observe that

$$\begin{aligned}
\Delta\left(k^2u\right) &= \frac{\partial^2}{\partial x^2}\left(k^2u\right) + \frac{\partial^2}{\partial y^2}\left(k^2u\right) + \frac{\partial^2}{\partial z^2}\left(k^2u\right) \\
&= u\frac{\partial^2}{\partial x^2}k^2 + 2\left(\frac{\partial}{\partial x}k^2\right)\left(\frac{\partial}{\partial x}u\right) + k^2\frac{\partial^2}{\partial x^2}u \\
&\quad + u\frac{\partial^2}{\partial y^2}k^2 + 2\left(\frac{\partial}{\partial y}k^2\right)\left(\frac{\partial}{\partial y}u\right) + k^2\frac{\partial^2}{\partial y^2}u \\
&\quad + u\frac{\partial^2}{\partial z^2}k^2 + 2\left(\frac{\partial}{\partial z}k^2\right)\left(\frac{\partial}{\partial z}u\right) + k^2\frac{\partial^2}{\partial z^2}u \\
&= u\Delta k^2 + 2\nabla k^2\cdot\nabla u + k^2\Delta u \\
&= \left(\Delta k^2 - k^4\right)u + 2\nabla k^2\cdot\nabla u + k^2 f.
\end{aligned}$$

It follows

$$\begin{aligned}
\Delta\left(k^2u\right)_{i,j,l} &= \left(\Delta k^2_{i,j,l} - k^4_{i,j,l}\right)u_{i,j,l} + 2\nabla k^2_{i,j,l}\cdot\nabla u_{i,j,l} + k^2_{i,j,l}f_{i,j,l} \\
&= \left(\Delta k^2_{i,j,l} - k^4_{i,j,l}\right)u_{i,j,l} + k^2_{i,j,l}f_{i,j,l} \\
&\quad + 2\nabla k^2_{i,j,l}\cdot\left(\delta_x u_{i,j,l} + \frac{h^2}{6}\left(\delta_x\delta_y^2 u_{i,j,l} + \delta_x\delta_z^2 u_{i,j,l} + \delta_x\left(k^2u\right)_{i,j,l} - \frac{\partial}{\partial x}f_{i,j,l}\right),\right. \\
&\quad \left.\delta_y u_{i,j,l} + \frac{h^2}{6}\left(\delta_y\delta_x^2 u_{i,j,l} + \delta_y\delta_z^2 u_{i,j,l} + \delta_y\left(k^2u\right)_{i,j,l} - \frac{\partial}{\partial y}f_{i,j,l}\right),\right. \\
&\quad \left.\delta_z u_{i,j,l} + \frac{h^2}{6}\left(\delta_z\delta_x^2 u_{i,j,l} + \delta_z\delta_y^2 u_{i,j,l} + \delta_z\left(k^2u\right)_{i,j,l} - \frac{\partial}{\partial z}f_{i,j,l}\right)\right) + \mathcal{O}\left(h^4\right) \\
&= \left(\Delta k^2_{i,j,l} - k^4_{i,j,l}\right)u_{i,j,l} + k^2_{i,j,l}f_{i,j,l}
\end{aligned}$$

$$+ 2\nabla k_{i,j}^2 \cdot \left(\nabla_h u_{i,j,l} + \frac{h^2}{6} \left(\nabla_h^{1/2} u_{i,j,l} + \nabla_h \left(k^2 u \right)_{i,j,l} - \nabla f_{i,j,l} \right) \right) + \mathcal{O}\left(h^4 \right)$$

from Lemma 3.1. ■

Now consider the sixth-order approximation to (1.1). The substitution of (3.5) and the corresponding equations for y and z give

$$\begin{aligned} L_h u_{i,j,l} &= \Delta_h u_{i,j,l} + k_{i,j,l}^2 u_{i,j,l} \\ &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) u_{i,j,l} + \frac{h^2}{12} \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + \frac{\partial^4}{\partial z^4} \right) u_{i,j,l} \\ &\quad + \frac{h^4}{360} \left(\frac{\partial^6}{\partial x^6} + \frac{\partial^6}{\partial y^6} + \frac{\partial^6}{\partial z^6} \right) u_{i,j,l} + k_{i,j,l}^2 u_{i,j,l} + \mathcal{O}\left(h^6 \right) \\ &= L u_{i,j,l} + \frac{h^2}{12} \nabla^4 u_{i,j,l} + \frac{h^4}{360} \nabla^6 u_{i,j,l} + \mathcal{O}\left(h^6 \right). \end{aligned}$$

It follows

$$f_{i,j,l} = L u_{i,j,l} = L_h u_{i,j,l} - \frac{h^2}{12} \beta_4 - \frac{h^4}{360} \beta_6 + \mathcal{O}\left(h^6 \right) \tag{3.7}$$

where $\beta_4 = \nabla^4 u_{i,j,l}$ and $\beta_6 = \nabla^6 u_{i,j,l}$. The coefficients of β_4 and β_6 show that their approximations must be fourth- and second-order, respectively.

3.3.1 Fourth-Order Approximation of β_4

To approximate β_4 with fourth-order accuracy, consider the second and fourth partial derivatives of the equation (1.1):

$$\frac{\partial^4}{\partial x^4} u = \frac{\partial^2}{\partial x^2} f - \frac{\partial^4}{\partial x^2 \partial y^2} u - \frac{\partial^4}{\partial x^2 \partial z^2} u - \frac{\partial^2}{\partial x^2} \left(k^2 u \right)$$

$$\frac{\partial^4}{\partial y^4}u = \frac{\partial^2}{\partial y^2}f - \frac{\partial^4}{\partial x^2\partial y^2}u - \frac{\partial^4}{\partial y^2\partial z^2}u - \frac{\partial^2}{\partial y^2}\left(k^2u\right)$$

$$\frac{\partial^4}{\partial z^4}u = \frac{\partial^2}{\partial z^2}f - \frac{\partial^4}{\partial x^2\partial z^2}u - \frac{\partial^4}{\partial y^2\partial z^2}u - \frac{\partial^2}{\partial z^2}\left(k^2u\right)$$

$$\frac{\partial^6}{\partial x^4\partial y^2}u + \frac{\partial^6}{\partial x^2\partial y^4}u = \frac{\partial^4}{\partial x^2\partial y^2}f - \frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u - \frac{\partial^4}{\partial x^2\partial y^2}\left(k^2u\right) \tag{3.8}$$

Then

$$\begin{aligned}
\beta_4 &= \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + \frac{\partial^4}{\partial z^4}\right)u_{i,j,l} \\
&= \frac{\partial^2}{\partial x^2}f_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j,l} - \frac{\partial^4}{\partial x^2\partial z^2}u_{i,j,l} - \frac{\partial^2}{\partial x^2}\left(k^2u\right)_{i,j,l} \\
&\quad + \frac{\partial^2}{\partial y^2}f_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j,l} - \frac{\partial^4}{\partial y^2\partial z^2}u_{i,j,l} - \frac{\partial^2}{\partial y^2}\left(k^2u\right)_{i,j,l} \\
&\quad + \frac{\partial^2}{\partial z^2}f_{i,j,l} - \frac{\partial^4}{\partial x^2\partial z^2}u_{i,j,l} - \frac{\partial^4}{\partial y^2\partial z^2}u_{i,j,l} - \frac{\partial^2}{\partial z^2}\left(k^2u\right)_{i,j,l} \\
&= \Delta f_{i,j,l} - 2\left(\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial x^2\partial z^2} + \frac{\partial^4}{\partial y^2\partial z^2}\right)u_{i,j,l} - \Delta\left(k^2u\right)_{i,j,l}.
\end{aligned}$$

Lemma 3.2 and equation (3.8) give

$$\begin{aligned}
\delta_x^2\delta_y^2u_{i,j,l} &= \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j,l} + \frac{h^2}{12}\left(\frac{\partial^6}{\partial x^4\partial y^2} + \frac{\partial^6}{\partial x^2\partial y^4}\right)u_{i,j,l} + \mathcal{O}\left(h^4\right) \\
&= \frac{\partial^4}{\partial x^2\partial y^2}u_{i,j,l} \\
&\quad + \frac{h^2}{12}\left(\frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l} - \frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}\left(k^2u\right)_{i,j,l}\right) + \mathcal{O}\left(h^4\right)
\end{aligned}$$

$$\frac{\partial^4}{\partial x^2\partial y^2}u_{i,j,l} = \delta_x^2\delta_y^2u_{i,j,l} - \frac{h^2}{12}\left(\frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l} - \frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}\left(k^2u\right)_{i,j,l}\right) + \mathcal{O}\left(h^4\right)$$

$$= \delta_x^2\delta_y^2 u_{i,j,l} + \frac{h^2}{12}\left(\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l} + \delta_x^2\delta_y^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l}\right) + \mathcal{O}\left(h^4\right)$$

since $\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l}$ and $\delta_x^2\delta_y^2\left(k^2u\right)$ are second-order approximations. Finally,

$$\begin{aligned}
\beta_4 &= \Delta f_{i,j,l} - 2\left(\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial x^2\partial z^2} + \frac{\partial^4}{\partial y^2\partial z^2}\right)u_{i,j,l} - \Delta\left(k^2u\right)_{i,j,l} \\
&= \Delta f_{i,j,l} - 2\delta_x^2\delta_y^2 u_{i,j,l} - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l} + \delta_x^2\delta_y^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l}\right) \\
&\quad - 2\delta_x^2\delta_z^2 u_{i,j,l} - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l} + \delta_x^2\delta_z^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial x^2\partial z^2}f_{i,j,l}\right) \\
&\quad - 2\delta_y^2\delta_z^2 u_{i,j,l} - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l} + \delta_y^2\delta_z^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial y^2\partial z^2}f_{i,j,l}\right) \\
&\quad - \Delta\left(k^2u\right)_{i,j,l} + \mathcal{O}\left(h^4\right) \\
&= \Delta f_{i,j,l} - 2\left(\delta_x^2\delta_y^2 + \delta_x^2\delta_z^2 + \delta_y^2\delta_z^2\right)u_{i,j,l} - \frac{h^2}{2}\delta_x^2\delta_y^2\delta_z^2 u_{i,j,l} - \Delta\left(k^2u\right)_{i,j,l} \\
&\quad - \frac{h^2}{6}\left(\delta_x^2\delta_y^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l}\right) \\
&\quad - \frac{h^2}{6}\left(\delta_x^2\delta_z^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial x^2\partial z^2}f_{i,j,l}\right) \\
&\quad - \frac{h^2}{6}\left(\delta_y^2\delta_z^2\left(k^2u\right)_{i,j,l} - \frac{\partial^4}{\partial y^2\partial z^2}f_{i,j,l}\right) + \mathcal{O}\left(h^4\right).
\end{aligned}$$

3.3.2 Second-Order Approximation of β_6

As defined in (3.7), $\beta_6 = \nabla^6 u_{i,j,l}$. To approximate β_6 with second-order accuracy, consider the fourth derivatives of the equation (1.1),

$$\frac{\partial^6}{\partial x^6}u = \frac{\partial^4}{\partial x^4}f - \frac{\partial^6}{\partial x^4\partial y^2}u - \frac{\partial^6}{\partial x^4\partial z^2}u - \frac{\partial^4}{\partial x^4}\left(k^2u\right)$$

with $\dfrac{\partial^6}{\partial y^6}u$ and $\dfrac{\partial^6}{\partial z^6}u$ given similarly. It follows

$$\begin{aligned}
\beta_6 &= \left(\frac{\partial^6}{\partial x^6}+\frac{\partial^6}{\partial y^6}+\frac{\partial^6}{\partial z^6}\right)u \\
&= \left(\frac{\partial^4}{\partial x^4}+\frac{\partial^4}{\partial y^4}+\frac{\partial^4}{\partial z^4}\right)f-\left(\frac{\partial^4}{\partial x^4}+\frac{\partial^4}{\partial y^4}+\frac{\partial^4}{\partial z^4}\right)\left(k^2u\right) \\
&\quad+\frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u+\frac{\partial^4}{\partial x^2\partial y^2}\left(k^2u\right)-\frac{\partial^4}{\partial x^2\partial y^2}f \\
&\quad+\frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u+\frac{\partial^4}{\partial x^2\partial z^2}\left(k^2u\right)-\frac{\partial^4}{\partial x^2\partial z^2}f \\
&\quad+\frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u+\frac{\partial^4}{\partial y^2\partial z^2}\left(k^2u\right)-\frac{\partial^4}{\partial y^2\partial z^2}f \\
&= \nabla^4 f-\nabla^4\left(k^2u\right)+3\frac{\partial^6}{\partial x^2\partial y^2\partial z^2}u \\
&\quad+\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)\left(k^2u\right)-\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)f
\end{aligned}$$

by (3.8). Lemma 3.3 give the second-order approximation as follows

$$\begin{aligned}
\beta_6 = \left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)\left(k^2u\right)_{i,j,l}-\frac{12}{h^2}\left(\Delta_h\left(k^2u\right)_{i,j,l}-\Delta\left(k^2u\right)_{i,j,l}\right)+3\delta_x^2\delta_y^2\delta_z^2u_{i,j,l} \\
-\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)f_{i,j,l}+\nabla^4 f_{i,j,l}+\mathcal{O}\left(h^2\right).
\end{aligned}$$

The approximated β_4 and β_6 substituted into (3.7), along with Lemma 3.4, gives

$$\begin{aligned}
&L_hu_{i,j,l}-\frac{h^2}{12}\Bigg[\Delta f_{i,j,l}-2\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)u_{i,j,l}-\frac{h^2}{2}\delta_x^2\delta_y^2\delta_z^2u_{i,j,l}-\Delta\left(k^2u\right)_{i,j,l} \\
&-\frac{h^2}{6}\left(\delta_x^2\delta_y^2\left(k^2u\right)_{i,j,l}-\frac{\partial^4}{\partial x^2\partial y^2}f_{i,j,l}\right)-\frac{h^2}{6}\left(\delta_x^2\delta_z^2\left(k^2u\right)_{i,j,l}-\frac{\partial^4}{\partial x^2\partial z^2}f_{i,j,l}\right)
\end{aligned}$$

$$
\begin{aligned}
&-\frac{h^2}{6}\left(\delta_y^2\delta_z^2\left(k^2u\right)_{i,j,l}-\frac{\partial^4}{\partial y^2\partial z^2}f_{i,j,l}\right)+\mathcal{O}\left(h^4\right)\Big] \\
&-\frac{h^4}{360}\Big[\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)\left(k^2u\right)_{i,j,l}-\frac{12}{h^2}\left(\Delta_h\left(k^2u\right)_{i,j,l}-\Delta\left(k^2u\right)_{i,j,l}\right)+3\delta_x^2\delta_y^2\delta_z^2u_{i,j,l} \\
&-\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)f_{i,j,l}+\nabla^4f_{i,j,l}+\mathcal{O}\left(h^2\right)\Big]+\mathcal{O}\left(h^6\right)=f_{i,j,l} \\
&=L_hu_{i,j,l}-\frac{h^2}{12}\Delta f_{i,j,l}+\frac{h^2}{6}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)u_{i,j,l}+\frac{h^4}{30}\delta_x^2\delta_y^2\delta_z^2u_{i,j,l}+\frac{h^2}{30}\Delta_h\left(k^2u\right)_{i,j,l} \\
&+\frac{h^2}{10}\nabla k_{i,j,l}^2\cdot\nabla_hu_{i,j,l}+\frac{h^4}{60}\nabla k_{i,j,l}^2\cdot\left(\nabla_h^{1/2}u_{i,j,l}+\nabla_h\left(k^2u\right)_{i,j,l}-\nabla f_{i,j,l}\right) \\
&+\frac{h^4}{90}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)\left(k^2u\right)_{i,j,l}-\frac{h^4}{90}\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)f_{i,j,l} \\
&+\frac{h^2}{20}\left(\Delta k_{i,j,l}^2-k_{i,j,l}^4\right)u_{i,j,l}+\frac{h^2}{20}k_{i,j,l}^2f_{i,j,l}-\frac{h^4}{360}\nabla^4f_{i,j,l}+\mathcal{O}\left(h^6\right) \\
&=f_{i,j,l}.
\end{aligned}
$$

Thus the sixth-order compact approximation is given by

$$
\begin{aligned}
&L_hu_{i,j,l}+\frac{h^2}{6}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)u_{i,j,l}+\frac{h^2}{20}\left(\Delta k_{i,j,l}^2-k_{i,j,l}^4\right)u_{i,j,l} \\
&\quad+\frac{h^2}{10}\nabla k_{i,j,l}^2\cdot\nabla_hu_{i,j,l}+\frac{h^4}{60}\nabla k_{i,j,l}^2\cdot\left(\nabla_h^{1/2}u_{i,j,l}+\nabla_h\left(k^2u\right)_{i,j,l}\right)+\frac{h^4}{30}\delta_x^2\delta_y^2\delta_z^2u_{i,j,l} \\
&\quad+\frac{h^2}{30}\Delta_h\left(k^2u\right)_{i,j,l}+\frac{h^4}{90}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)\left(k^2u\right)_{i,j,l}
\end{aligned}
$$

with the right-hand side

$$
\begin{aligned}
&\left(1-\frac{h^2}{20}k_{i,j,l}^2\right)f_{i,j,l}+\frac{h^2}{12}\Delta f_{i,j,l}+\frac{h^4}{60}\nabla k_{i,j,l}^2\cdot\nabla f_{i,j,l}+\frac{h^4}{360}\nabla^4f_{i,j,l} \\
&\quad+\frac{h^4}{90}\left(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\right)f_{i,j,l}
\end{aligned}
$$

for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $l = 1, \ldots, N_z$. This scheme provides the following coefficients for a 27-point stencil

$$u_{i,j,l} : -\frac{64}{15h^2} + \frac{14k_{i,j,l}^2}{15} + \frac{h^2}{20}\left(\Delta k_{i,j,l}^2 - k_{i,j,l}^4\right), \quad u_{i\pm1,j\pm1,l\pm1} : \frac{1}{30h^2},$$

$$u_{i\pm1,j,l} : \frac{7}{15h^2} - \frac{k_{i\pm1,j,l}^2}{90} \pm \frac{h}{60}\frac{\partial}{\partial x}k_{i,j,l}^2\left(1 + \frac{h^2}{2}k_{i\pm1,j,l}^2\right),$$

$$u_{i,j\pm1,l} : \frac{7}{15h^2} - \frac{k_{i,j\pm1,l}^2}{90} \pm \frac{h}{60}\frac{\partial}{\partial y}k_{i,j,l}^2\left(1 + \frac{h^2}{2}k_{i,j\pm1,l}^2\right),$$

$$u_{i,j,l\pm1} : \frac{7}{15h^2} - \frac{k_{i,j,l\pm1}^2}{90} \pm \frac{h}{60}\frac{\partial}{\partial z}k_{i,j,l}^2\left(1 + \frac{h^2}{2}k_{i,j,l\pm1}^2\right),$$

$$u_{i,j+1,l\pm1} : \frac{1}{10h^2} + \frac{k_{i,j+1,l\pm1}^2}{90} + \frac{h}{120}\left(\frac{\partial}{\partial y} \pm \frac{\partial}{\partial z}\right)k_{i,j,l}^2,$$

$$u_{i,j-1,l\pm1} : \frac{1}{10h^2} + \frac{k_{i,j-1,l\pm1}^2}{90} - \frac{h}{120}\left(\frac{\partial}{\partial y} \mp \frac{\partial}{\partial z}\right)k_{i,j,l}^2,$$

$$u_{i+1,j,l\pm1} : \frac{1}{10h^2} + \frac{k_{i+1,j,l\pm1}^2}{90} + \frac{h}{120}\left(\frac{\partial}{\partial x} \pm \frac{\partial}{\partial z}\right)k_{i,j,l}^2,$$

$$u_{i-1,j,l\pm1} : \frac{1}{10h^2} + \frac{k_{i-1,j,l\pm1}^2}{90} - \frac{h}{120}\left(\frac{\partial}{\partial x} \mp \frac{\partial}{\partial z}\right)k_{i,j,l}^2,$$

$$u_{i+1,j\pm1,l} : \frac{1}{10h^2} + \frac{k_{i+1,j\pm1,l}^2}{90} + \frac{h}{120}\left(\frac{\partial}{\partial x} \pm \frac{\partial}{\partial y}\right)k_{i,j,l}^2,$$

$$u_{i-1,j\pm1,l} : \frac{1}{10h^2} + \frac{k_{i-1,j\pm1,l}^2}{90} - \frac{h}{120}\left(\frac{\partial}{\partial x} \mp \frac{\partial}{\partial y}\right)k_{i,j,l}^2.$$

3.4 Fourth-Order Accuracy for Convection-Diffusion

Although this research focuses on the Helmholtz equation, the presented numerical methods have a wide range of applications. One such application is the convection-diffusion equation. To demonstrate the FFT solver's versatility, section 3.4 develops only the fourth-order finite difference scheme. The scheme gives the approximate solution of the convection-diffusion equation on rectangular domains with Dirichlet boundary conditions. That is

$$\Delta u(\mathbf{x}) - \mathbf{a} \cdot \nabla u(\mathbf{x}) = f(\mathbf{x}) \quad \text{in } \Omega \tag{3.9}$$

with $\mathbf{a} = (a_1, a_2, a_3) \in \mathbb{R}^3$ where a_1, a_2 and a_3 are the constant convection coefficients in the x, y and z directions respectively. The domain is defined as before, that is (3.1), where $h_x = (x_u - x_l)/(N_x + 1)$, $h_y = (y_u - y_l)/(N_y + 1)$ and $h_z = (z_u - z_l)/(N_z + 1)$.

Three-dimensional Taylor series expansion, (3.2), gives the fourth-order approximations for the first and second derivatives as

$$\frac{\partial}{\partial x} u_{i,j,l} = \delta_x u_{i,j,l} - \frac{{h_x}^2}{6} \frac{\partial^3}{\partial x^3} u_{i,j,l} + \mathcal{O}\left({h_x}^4\right) \tag{3.10}$$

$$\frac{\partial^2}{\partial x^2} u_{i,j,l} = \delta_x^2 u_{i,j,l} - \frac{{h_x}^2}{12} \frac{\partial^4}{\partial x^4} u_{i,j,l} + \mathcal{O}\left({h_x}^4\right). \tag{3.11}$$

Taking the partial derivative with respect to x the equation (3.9) can be written

$$\frac{\partial^2}{\partial x^2} u = f - \frac{\partial^2}{\partial y^2} u - \frac{\partial^2}{\partial z^2} u + a_1 \frac{\partial}{\partial x} u + a_2 \frac{\partial}{\partial y} u + a_3 \frac{\partial}{\partial z} u$$

$$\frac{\partial^3}{\partial x^3} u = \frac{\partial}{\partial x} f - \frac{\partial^3}{\partial x \partial y^2} u - \frac{\partial^3}{\partial x \partial z^2} u + a_1 \frac{\partial^2}{\partial x^2} u + a_2 \frac{\partial^2}{\partial x \partial y} u + a_3 \frac{\partial^2}{\partial x \partial z} u \tag{3.12}$$

$$\frac{\partial^4}{\partial x^4} u = \frac{\partial^2}{\partial x^2} f - \frac{\partial^4}{\partial x^2 \partial y^2} u - \frac{\partial^4}{\partial x^2 \partial z^2} u + a_1 \frac{\partial^3}{\partial x^3} u + a_2 \frac{\partial^3}{\partial x^2 \partial y} u + a_3 \frac{\partial^3}{\partial x^2 \partial z} u. \tag{3.13}$$

The second-order approximation of (3.12) is given as

$$\frac{\partial^3}{\partial x^3}u_{i,j,l} = \frac{\partial}{\partial x}f_{i,j,l} + \left(a_1\delta_x^2 + a_2\delta_x\delta_y + a_3\delta_x\delta_z - \delta_x\delta_y^2 - \delta_x\delta_z^2\right)u_{i,j,l} + \mathcal{O}\left({h_x}^2\right). \tag{3.14}$$

The second-order approximation of (3.13) is found similarly, but it also requires substitution of (3.14). That is

$$\begin{aligned}
\frac{\partial^4}{\partial x^4}u_{i,j,l} &= \frac{\partial^2}{\partial x^2}f_{i,j,l} + a_1\frac{\partial^3}{\partial x^3}u_{i,j,l} + \left(a_2\delta_x^2\delta_y + a_3\delta_x^2\delta_z - \delta_x^2\delta_y^2 - \delta_x^2\delta_z^2\right)u_{i,j,l} + \mathcal{O}\left({h_x}^2\right) \\
&= \frac{\partial^2}{\partial x^2}f_{i,j,l} + a_1\left[\frac{\partial}{\partial x}f_{i,j,l} + \left(a_1\delta_x^2 + a_2\delta_x\delta_y + a_3\delta_x\delta_z - \delta_x\delta_y^2 - \delta_x\delta_z^2\right)u_{i,j,l}\right] \\
&\quad + \left(a_2\delta_x^2\delta_y + a_3\delta_x^2\delta_z - \delta_x^2\delta_y^2 - \delta_x^2\delta_z^2\right)u_{i,j,l} + \mathcal{O}\left({h_x}^2\right) \\
&= \frac{\partial^2}{\partial x^2}f_{i,j,l} + a_1\frac{\partial}{\partial x}f_{i,j,l} + a_1\left(a_1\delta_x^2 + a_2\delta_x\delta_y + a_3\delta_x\delta_z - \delta_x\delta_y^2 - \delta_x\delta_z^2\right)u_{i,j,l} \\
&\quad + \left(a_2\delta_x^2\delta_y + a_3\delta_x^2\delta_z - \delta_x^2\delta_y^2 - \delta_x^2\delta_z^2\right)u_{i,j,l} + \mathcal{O}\left({h_x}^2\right).
\end{aligned} \tag{3.15}$$

Substituting (3.14) into (3.10) gives the fourth-order approximation of the first derivative as

$$\frac{\partial}{\partial x}u_{i,j,l} = \delta_x u_{i,j,l} - \frac{{h_x}^2}{6}\left[\frac{\partial}{\partial x}f_{i,j,l} + \left(a_1\delta_x^2 + a_2\delta_x\delta_y + a_3\delta_x\delta_z - \delta_x\delta_y^2 - \delta_x\delta_z^2\right)u_{i,j,l}\right] + \mathcal{O}\left({h_x}^4\right). \tag{3.16}$$

Similarly, substituting (3.15) into (3.11) gives the fourth-order approximation of the second derivative. That is

$$\begin{aligned}
\frac{\partial^2}{\partial x^2}u_{i,j,l} = -\frac{{h_x}^2}{12}\Bigg[&\frac{\partial^2}{\partial x^2}f_{i,j,l} + a_1\frac{\partial}{\partial x}f_{i,j,l} + a_1\left(a_1\delta_x^2 + a_2\delta_x\delta_y + a_3\delta_x\delta_z - \delta_x\delta_y^2 - \delta_x\delta_z^2\right)u_{i,j,l} \\
&+ \left(a_2\delta_x^2\delta_y + a_3\delta_x^2\delta_z - \delta_x^2\delta_y^2 - \delta_x^2\delta_z^2\right)u_{i,j,l}\Bigg] + \delta_x^2 u_{i,j,l} + \mathcal{O}\left({h_x}^4\right).
\end{aligned} \tag{3.17}$$

Equations (3.16) and (3.17) give the approximated partial derivatives with respect to x, but

the derivatives for the y and z directions are found similarly. Using the same notation for Δ_h and ∇_h from Section 3.3 the fourth-order compact approximation for (3.9) is given by

$$\Delta_h u_{i,j,l} - \mathbf{a}\cdot\nabla_h u_{i,j,l} + \frac{1}{12}\left(a_1 {h_x}^2\delta_x + a_2 {h_y}^2\delta_y + a_3 {h_z}^2\delta_z\right)(\mathbf{a}\cdot\nabla_h u_{i,j,l})$$

$$-\frac{1}{12}\left((a_1+1){h_x}^2\delta_x, (a_2+1){h_y}^2\delta_y, (a_3+1){h_z}^2\delta_z\right)\cdot\left(\delta_y^2+\delta_z^2, \delta_x^2+\delta_z^2, \delta_x^2+\delta_y^2\right)u_{i,j,l}$$

$$-\frac{1}{12}\left({h_x}^2\delta_x^2, {h_y}^2\delta_y^2, {h_z}^2\delta_z^2\right)\cdot(a_2\delta_y + a_3\delta_z, a_1\delta_x + a_3\delta_z, a_1\delta_x + a_2\delta_y)\,u_{i,j,l}$$

with the right-hand side

$$f_{i,j,l} + \frac{1}{12}\left({h_x}^2\frac{\partial^2}{\partial x^2} + {h_y}^2\frac{\partial^2}{\partial y^2} + {h_z}^2\frac{\partial^2}{\partial z^2}\right)f_{i,j,l} - \frac{1}{12}\left(a_1{h_x}^2, a_2{h_y}^2, a_3{h_z}^2\right)\cdot\nabla f_{i,j,l}$$

for $i = 1,\ldots,N_x$, $j = 1,\ldots,N_y$ and $l = 1,\ldots,N_z$. This provides the following 27-point stencil coefficients

$$u_{i,j,l} : -\frac{4}{3}\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right) - \frac{1}{6}\left(a_1^2+a_2^2+a_3^2\right), \qquad u_{i\pm1,j\pm1,l\pm1} : 0\,,$$

$$u_{i\pm1,j,l} : \frac{1}{3}\left(\frac{a_1^2}{4} \mp \frac{a_1}{h_x} + \frac{2}{h_x^2}\right) - \frac{1}{6}\left(1 \mp \frac{a_1 h_x}{2}\right)\left(\frac{1}{h_y^2}+\frac{1}{h_z^2}\right),$$

$$u_{i,j\pm1,l} : \frac{1}{3}\left(\frac{a_2^2}{4} \mp \frac{a_2}{h_y} + \frac{2}{h_y^2}\right) - \frac{1}{6}\left(1 \mp \frac{a_2 h_y}{2}\right)\left(\frac{1}{h_x^2}+\frac{1}{h_z^2}\right),$$

$$u_{i,j,l\pm1} : \frac{2}{3h_z^2} + \frac{a_3^2}{12} \mp \frac{a_3}{3h_z} - \frac{1}{6}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right),$$

$$u_{i,j+1,l\pm1} : \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{1}{h_z^2}+\frac{1}{h_y^2}\right) - \frac{a_2}{24}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{h_y}{h_z^2}+\frac{1}{h_y}\right),$$

$$u_{i,j-1,l\pm1} : \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{1}{h_z^2}+\frac{1}{h_y^2}\right) + \frac{a_2}{24}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{h_y}{{h_z}^2}+\frac{1}{h_y}\right),$$

$$u_{i+1,j,l\pm1} : \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{1}{h_z^2}+\frac{1}{h_x^2}\right) - \frac{a_1}{24}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{h_x}{{h_z}^2}+\frac{1}{h_x}\right),$$

$$u_{i-1,j,l\pm1} : \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{1}{h_z^2}+\frac{1}{h_x^2}\right) + \frac{a_1}{24}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(\frac{h_x}{h_z^2}+\frac{1}{h_x}\right),$$

$$u_{i+1,j\pm1,l} : \frac{1}{12}\left(\frac{1}{h_y^2}+\frac{1}{h_x^2}\right) \pm \frac{a_1 a_2}{48}\left(\frac{h_x}{h_y}+\frac{h_y}{h_x}\right) - \frac{a_1}{24}\left(\frac{h_x}{h_y^2}+\frac{1}{h_x}\right) \mp \frac{a_2}{24}\left(\frac{h_y}{h_x^2}+\frac{1}{h_y}\right),$$

$$u_{i-1,j\pm1,l} : \frac{1}{12}\left(\frac{1}{h_y^2}+\frac{1}{h_x^2}\right) \mp \frac{a_1 a_2}{48}\left(\frac{h_x}{h_y}+\frac{h_y}{h_x}\right) + \frac{a_1}{24}\left(\frac{h_x}{h_y^2}+\frac{1}{h_x}\right) \mp \frac{a_2}{24}\left(\frac{h_y}{h_x^2}+\frac{1}{h_y}\right).$$

3.5 Summary

Chapter 3 developed the second-, fourth- and sixth-order approximation schemes for the solution of the three-dimensional Helmholtz equation. Additionally, the fourth-order approximation scheme was developed for the convection-diffusion equation. Chapter 4 will present the definition and approximations of the boundary conditions considered in the research.

4 Boundary Conditions

Chapter 4 presents the definitions and detailed applications of the Dirichlet and Sommerfeld-like boundary conditions. In the case of Sommerfeld-like boundary conditions, the necessary approximations are provided for the respective scheme's order of approximation. Vector and matrix notations are required. Let $\mathbf{v}_{a:b,c:d} \in \mathbb{C}^{(b-a+1)\cdot(d-c+1)}$ be such that

$$\mathbf{v}_{a:b,c:d} = \begin{bmatrix} v_{a,c} & v_{a+1,c} & \cdots & v_{b,c} & v_{a,c+1} & \cdots & v_{b,c+1} & \cdots & v_{a,d} & \cdots & v_{b,d} \end{bmatrix}^T$$

where $a, b, c, d \in \mathbb{N}$ with $a < b$ and $c < d$. For matrices a similar notation is used. Let $M \in \mathbb{C}^{n\times m}$ where $n, m \in \mathbb{N}$. If $a, b, c, d \in \mathbb{N}$ such that $a < b \leq n$ and $c < d \leq m$ then $M_{a:b,c:d}$ is the sub matrix of matrix M consisting of rows a through b and columns c through d. The sub matrix $M_{a,1:m}$ represents the a-th row of the matrix M and $M_{1:n,c}$ is the c-th column. The vector and matrix notations are utilized throughout.

To illustrate how the boundary conditions are implemented, Chapter 4 works through a simple, two-dimensional Helmholtz example where $N_x = N_y = 3$ and k is assumed constant. Additionally, consider the general 9-point stencil in Figure 4.1. The stencil holds when k is constant with no other assumptions. Figure 4.2 illustrates the domain with different positions of the 9-point stencil corresponding to the first three rows of the matrix in Figure 4.3. The solution vector, $\mathbf{u}_{1:N_x,1:N_y}$, consists of the $N_x \cdot N_y = 9$ elements shown in the interior colored white, while the boundary is shown in blue. This illustration demonstrates the relationship a solution element shares with its neighbors. The only element in the example without influence from the boundary is $u_{2,2}$. An extension of $\mathbf{u}_{1:N_x,1:N_y}$ to $\mathbf{u}_{0:N_x+1,0:N_y+1}$ includes the boundary values. Let $\mathcal{B} \in \mathbb{C}^{N_x\cdot N_y \times (N_x+2)(N_y+2)}$ be the sparse block tridiagonal

Figure 4.1: General 9-Point Stencil for Constant k

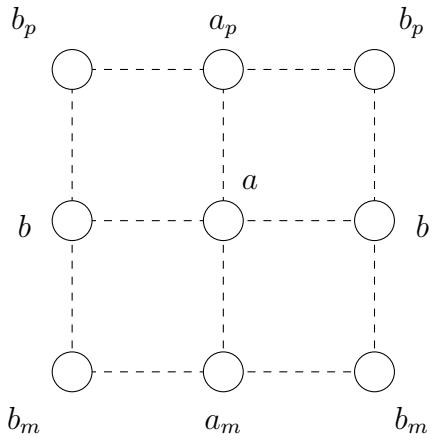
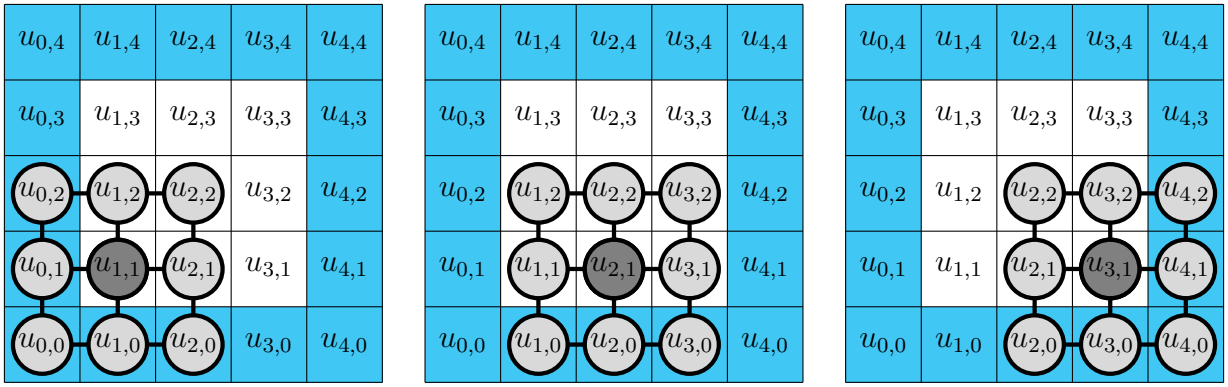


Figure 4.2: 3×3 Computational Domain with 9-Point Stencil

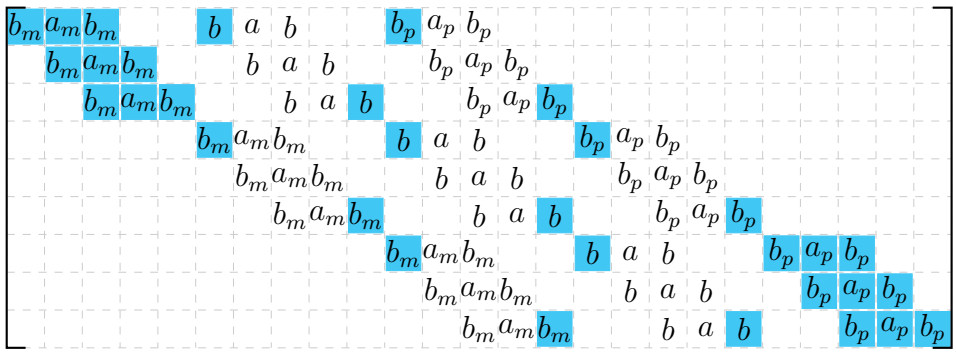


matrix shown in Figure 4.3. Then $\mathcal{B}\mathbf{u}_{0:N_x+1,0:N_y+1}$ represents all the possible positions of the stencil in Figure 4.2. The matrix elements of $\mathcal{B}$ that are highlighted blue in Figure 4.3 will be relocated or approximated appropriately dependent upon the boundary conditions under consideration.

4.1 Dirichlet Conditions

In Dirichlet boundary conditions the solution to (1.1), u, is known on the boundary, that is $u(\mathbf{x}) = g(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$. Since the function is known on the boundary no approximation is

Figure 4.3: Coefficient Matrix $\mathcal{B}$ with Boundary Elements



required. The values of $\mathcal{B}$ corresponding to the boundary can be moved to the right-hand side of the system, demonstrated in the two-dimensional example. Let $\mathbf{f}_{1:N_x,1:N_y} \in \mathbb{C}^{N_x \cdot N_y}$ be such that $\mathcal{B}\mathbf{u}_{0:N_x+1,0:N_y+1} = \mathbf{f}_{1:N_x,1:N_y}$. This system is equivalent to $A\mathbf{u}_{1:N_x,1:N_y} = \tilde{\mathbf{f}}_{1:N_x,1:N_y}$, that is

$$\begin{bmatrix}
a & b & & a_p & b_p & & & & \\
b & a & b & b_p & a_p & b_p & & & \\
 & b & a & & b_p & a_p & & & \\
a_m & b_m & & a & b & & a_p & b_p & \\
b_m & a_m & b_m & b & a & b & b_p & a_p & b_p \\
 & b_m & a_m & & b & a & & b_p & a_p \\
 & & & a_m & b_m & & a & b & \\
 & & & b_m & a_m & b_m & b & a & b \\
 & & & & b_m & a_m & & b & a
\end{bmatrix}
\begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix}
f_{1,1} - b_m u_{0,0} - a_m u_{1,0} - b_m u_{2,0} - b u_{0,1} - b_p u_{0,2} \\
f_{2,1} - b_m u_{1,0} - a_m u_{2,0} - b_m u_{3,0} \\
f_{3,1} - b_m u_{2,0} - a_m u_{3,0} - b_m u_{4,0} - b u_{4,1} - b_p u_{4,2} \\
f_{1,2} - b_m u_{0,1} - b u_{0,2} - b_p u_{0,3} \\
f_{2,2} \\
f_{3,2} - b_m u_{4,1} - b u_{4,2} - b_p u_{4,3} \\
f_{1,3} - b_p u_{0,4} - a_p u_{1,4} - b_p u_{2,4} - b u_{0,3} - b_m u_{0,2} \\
f_{2,3} - b_p u_{1,4} - a_p u_{2,4} - b_p u_{3,4} \\
f_{3,3} - b_p u_{2,4} - a_p u_{3,4} - b_p u_{4,4} - b u_{4,3} - b_m u_{4,2}
\end{bmatrix}$$

where all the known boundary values and their respective stencil coefficient have been relocated to the right-hand side. The solution of this system with k variable in the vertical

direction in both two- and three-dimensions is considered in Chapter 5.

4.2 Sommerfeld-Like Conditions

Section 4.2 describes an approximation for the Sommerfeld radiation conditions, namely Sommerfeld-like boundary conditions. The Sommerfeld radiation condition is given by (1.3) for both two- and three-dimensions. Truncating the unbounded domain to a finite domain at the boundary under consideration provides the approximation, Sommerfeld-like conditions,

$$\nabla u(\mathbf{x}) \cdot \mathbf{n} - ik(\mathbf{x})u(\mathbf{x}) = 0 \tag{4.1}$$

for $\mathbf{x} \in \partial\Omega$ where $\mathbf{n}$ is the outward normal vector of the boundary. Equation (4.1) presents the first-order absorbing boundary conditions [10]. Higher-order ABCs are viable, but not considered here as they do not have a dramatic influence on the accuracy of the solvers [11].

The following subsections give the boundary approximations required for the second-, fourth- and sixth-order schemes from Chapters 2 and 3. The approximations are strictly for the Helmholtz applications and only those for the three-dimensional case are given, although the two-dimensional case is similar. Consider i strictly as $i = \sqrt{-1}$ and not an index; ι is used for indexing.

For the first term in the three-dimensional second-order scheme (3.3), $u_{\iota-1,j,l}/{h_x}^2$ for $\iota = 1, \ldots, N_x$, $j = 1, \ldots, N_y$, $l = 1, \ldots, N_z$, if $\iota = 1$, then the term is on the boundary. With Sommerfeld-like conditions this value is unknown and must be approximated with elements of the solution vector that are not on the boundary. To maintain the order of the scheme these boundary elements require higher-order approximation by at least two due to the division of ${h_x}^2$. The need for higher-order approximation is true regardless of the order

of the scheme or direction of the boundary. Therefore, the approximations of the boundary conditions given in the section are fourth-, seventh- and ninth-order accuracy for the second-, fourth- and sixth-order schemes respectively.

4.2.1 Fourth-Order Accuracy

For the boundaries with respect to the x direction, the subtraction of $u_{\iota+1,j,l}$ and $u_{\iota-1,j,l}$ from the three-dimensional Taylor series expansion (3.2) gives

$$\begin{aligned} u_{\iota+1,j,l} - u_{\iota-1,j,l} &= 2h_x \frac{\partial}{\partial x} u_{\iota,j,l} + \frac{h_x^3}{3} \frac{\partial^3}{\partial x^3} u_{\iota,j,l} + \mathcal{O}\left(h_x^5\right) \\ \delta_x u_{\iota,j,l} &= \frac{\partial}{\partial x} u_{\iota,j,l} + \frac{h_x^2}{6} \delta_x^2 \frac{\partial}{\partial x} u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right) \\ \frac{\partial}{\partial x} u_{\iota,j,l} &= \left(1 + \frac{h_x^2}{6}\delta_x^2\right)^{-1} \delta_x u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right). \end{aligned} \tag{4.2}$$

Written explicitly for the boundaries in the x direction, (4.1) is

$$\nabla u \cdot (1,0,0) - iku = \frac{\partial}{\partial x} u - iku = 0 \tag{4.3}$$

$$\nabla u \cdot (-1,0,0) - iku = -\frac{\partial}{\partial x} u - iku = 0 = \frac{\partial}{\partial x} u + iku. \tag{4.4}$$

On the lower and upper boundaries of the domain the Sommerfeld-like boundary conditions differ only by the sign of iku, which is true for all spatial directions. Assume that $k_{\iota,j,l} = k_{j,l}$ near the boundaries in the x direction. The assumption is valid for the problems under consideration as the inclusion will not be near the boundary. Substituting (4.2) into the equations (4.3) and (4.4) gives the fourth-order approximations for the terms on the x

boundaries as

$$0 = \frac{\partial}{\partial x} u_{\iota,j,l} - ik_{\iota,j,l} u_{\iota,j,l}$$

$$0 = \delta_x u_{\iota,j,l} - ik_{\iota,j,l} u_{\iota,j,l} - ik_{\iota,j,l} \frac{h_x^2}{6} \delta_x^2 u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right)$$

$$0 = \left(3 \pm ih_x k_{j,l}\right) u_{\iota+1,j,l} - \left(3 \mp ih_x k_{j,l}\right) u_{\iota-1,j,l} \pm 4ih_x k_{j,l} u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right)$$

$$u_{\iota\pm1,j,l} = \frac{(3 + ih_x k_{j,l})}{(3 - ih_x k_{j,l})} u_{\iota\mp1,j,l} + \frac{4ih_x k_{j,l}}{(3 - ih_x k_{j,l})} u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right)$$

$$u_{\iota\pm1,j,l} = \beta_{x,j,l} u_{\iota\mp1,j,l} + \alpha_{x,j,l} u_{\iota,j,l} + \mathcal{O}\left(h_x^4\right) \tag{4.5}$$

where $\beta_{x,j,l} = \dfrac{(3 + ih_x k_{j,l})}{(3 - ih_x k_{j,l})}$ and $\alpha_{x,j,l} = \dfrac{4ih_x k_{j,l}}{(3 - ih_x k_{j,l})}$. The coefficients $\beta_{y,\iota,l}$, $\alpha_{y,\iota,l}$, $\beta_{z,\iota,j}$ and $\alpha_{z,\iota,j}$ are found in a similar way.

4.2.2 Seventh-Order Accuracy

The seventh-order approximation for the Sommerfeld-like boundary conditions require the derivatives of (4.3) and (4.4). Maintaining the assumption that the function k is constant with respect to x near these boundaries, the derivatives are

$$\frac{\partial}{\partial x} u \pm iku = 0$$

$$\frac{\partial^2}{\partial x^2} u = \mp ik \frac{\partial}{\partial x} u$$

$$\frac{\partial^3}{\partial x^3} u = \mp ik \frac{\partial^2}{\partial x^2} u = \mp ik \left(\mp ik \frac{\partial}{\partial x} u \right) = -k^2 \frac{\partial}{\partial x} u$$

$$\frac{\partial^4}{\partial x^4} u = \mp ik \frac{\partial^3}{\partial x^3} u = \mp ik \left(-k^2 \frac{\partial}{\partial x} u \right) = \pm ik^3 \frac{\partial}{\partial x} u$$

$$\frac{\partial^5}{\partial x^5}u = \mp ik\frac{\partial^4}{\partial x^4}u = \mp ik\left(\pm ik^3\frac{\partial}{\partial x}u\right) = k^4\frac{\partial}{\partial x}u. \tag{4.6}$$

The seventh-order difference of $u_{\iota+1,j,l}$ and $u_{\iota-1,j,l}$ from (3.2) becomes

$$u_{\iota+1,j,l} - u_{\iota-1,j,l} = 2h_x\frac{\partial}{\partial x}u_{i,j,l} + 2\frac{h_x^3}{3!}\frac{\partial^3}{\partial x^3}u_{i,j,l} + 2\frac{h_x^5}{5!}\frac{\partial^5}{\partial x^5}u_{i,j,l} + \mathcal{O}\left({h_x}^7\right)$$

$$\frac{\partial}{\partial x}u_{\iota,j,l} = \left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!}\right)^{-1}\delta_x u_{\iota,j,l} + \mathcal{O}\left({h_x}^6\right)$$

after the proper substitution. Subsequently,

$$0 = \left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!}\right)^{-1}\delta_x u_{\iota,j,l} \pm ik_{j,l}u_{\iota,j,l} + \mathcal{O}\left({h_x}^6\right)$$

$$0 = u_{\iota+1,j,l} - u_{\iota-1,j,l} \pm 2ih_xk_{j,l}\left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!}\right)u_{\iota,j,l} + \mathcal{O}\left({h_x}^7\right).$$

Thus the seventh-order approximations for the terms on the x boundaries are

$$u_{\iota\pm1,j,l} = u_{\iota\mp1,j,l} + 2ih_xk_{j,l}\left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!}\right)u_{\iota,j,l} + \mathcal{O}\left({h_x}^7\right)$$

$$u_{\iota\pm1,j,l} = \beta_x u_{\iota\mp1,j,l} + \alpha_{x,j,l}u_{\iota,j,l} + \mathcal{O}\left(h_x^7\right) \tag{4.7}$$

where $\beta_x = 1$ and $\alpha_{x,j,l} = 2ih_xk_{j,l}\left(1 - h_x^2k_{j,l}^2/3! + h_x^4k_{j,l}^4/5!\right)$. The coefficients β_y, $\alpha_{y,\iota,l}$, β_z and $\alpha_{z,\iota,j}$ are found in the same way.

4.2.3 Ninth-Order Accuracy

The development of the ninth-order approximation builds upon seventh-order. Continuing from equation (4.6), the sixth and seventh partial derivatives are

$$\frac{\partial^6}{\partial x^6}u = \mp ik\frac{\partial^5}{\partial x^5}u = \mp ik\left(k^4\frac{\partial}{\partial x}u\right) = \mp ik^5\frac{\partial}{\partial x}u$$

$$\frac{\partial^7}{\partial x^7}u = \mp ik\frac{\partial^6}{\partial x^6}u = \mp ik\left(\mp ik^5\frac{\partial}{\partial x}u\right) = -k^6\frac{\partial}{\partial x}u.$$

The ninth-order subtraction of $u_{\iota+1,j,l}$ and $u_{\iota-1,j,l}$ from (3.2) gives

$$u_{\iota+1,j,l} - u_{\iota-1,j,l} = 2h_x\frac{\partial}{\partial x}u_{\iota,j,l} + 2\frac{h_x^3}{3!}\frac{\partial^3}{\partial x^3}u_{\iota,j,l} + 2\frac{h_x^5}{5!}\frac{\partial^5}{\partial x^5}u_{\iota,j,l} + 2\frac{h_x^7}{7!}\frac{\partial^7}{\partial x^7}u_{\iota,j,l} + \mathcal{O}\left({h_x}^9\right)$$

$$\frac{\partial}{\partial x}u_{\iota,j,l} = \left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!} - \frac{h_x^6k_{j,l}^6}{7!}\right)^{-1}\delta_x + \mathcal{O}\left({h_x}^8\right).$$

after the proper substitution. Then

$$0 = \left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!} - \frac{h_x^6k_{j,l}^6}{7!}\right)^{-1}\delta_x u_{\iota,j,l} \pm ik_{j,l}u_{\iota,j,l} + \mathcal{O}\left({h_x}^8\right)$$

$$0 = u_{\iota+1,j,l} - u_{\iota-1,j,l} \pm 2ih_xk_{j,l}\left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!} - \frac{h_x^6k_{j,l}^6}{7!}\right)u_{\iota,j,l} + \mathcal{O}\left({h_x}^9\right). \tag{4.8}$$

Thus the ninth-order approximations for the terms on the x boundaries are

$$u_{\iota\pm1,j,l} = u_{\iota\mp1,j,l} + 2ih_xk_{j,l}\left(1 - \frac{h_x^2k_{j,l}^2}{3!} + \frac{h_x^4k_{j,l}^4}{5!} - \frac{h_x^6k_{j,l}^6}{7!}\right)u_{\iota,j,l} + \mathcal{O}\left({h_x}^9\right)$$

$$u_{\iota\pm1,j,l} = \beta_x u_{\iota\mp1,j,l} + \alpha_{x,j,l}u_{\iota,j,l} + \mathcal{O}\left(h_x^9\right)$$

where $\beta_x = 1$ and $\alpha_{x,j,l} = 2ih_xk_{j,l}\left(1 - h_x^2k_{j,l}^2/3! + h_x^4k_{j,l}^4/5! - h_x^6k_{j,l}^6/7!\right)$. The coefficients β_y, $\alpha_{y,\iota,l}$, β_z and $\alpha_{z,\iota,j}$ are found in the same way.

Return to the two-dimensional example and the matrix $\mathcal{B}$ given in Figure 4.3. The linear system $\mathcal{B}\mathbf{u}_{0:N_x+1,0:N_y+1} = \mathbf{f}_{1:N_x,1:N_y}$ needs to be consolidated to $A\mathbf{u}_{1:N_x,1:N_y} = \mathbf{f}_{1:N_x,1:N_y}$ where $A \in \mathbb{C}^{N_x\cdot N_y \times N_x\cdot N_y}$ and $\mathbf{u}_{1:N_x,1:N_y}$ is the unknown solution vector without the boundary elements. Consolidation is accomplished by using the approximations for the boundary conditions. Define $\alpha_x = \alpha_{x,j,l}$ since $\alpha_{x,j,l}$ is constant in this example, and define α_y, β_x and

β_y similarly. Consider the first row of $\mathcal{B}$ times $\mathbf{u}_{0:N_x+1,0:N_y+1}$. That is

$$
\begin{aligned}
f_{1,1} &= \mathcal{B}_{1,1:(N_x+2)(N_y+2)} \cdot \mathbf{u}_{0:N_x+1,0:N_y+1} \\
&= b_m u_{0,0} + a_m u_{1,0} + b_m u_{2,0} + b u_{0,1} + a u_{1,1} + b u_{2,1} + b_p u_{0,2} + a_p u_{1,2} + b_p u_{2,2} \\
&= b_m \left[\beta_x\beta_y u_{2,2} + \alpha_y\beta_x u_{2,1} + \alpha_x\beta_y u_{1,2} + \alpha_x\alpha_y u_{1,1}\right] + a_m \left[\beta_y u_{1,2} + \alpha_y u_{1,1}\right] \\
&\quad + b_m \left[\beta_y u_{2,2} + \alpha_y u_{2,1}\right] + b \left[\beta_x u_{2,1} + \alpha_x u_{1,1}\right] + a u_{1,1} + b u_{2,1} \\
&\quad + b_p \left[\beta_x u_{2,2} + \alpha_x u_{1,2}\right] + a_p u_{1,2} + b_p u_{2,2} \\
&= \left(a + b_m\alpha_x\alpha_y + a_m\alpha_y + b\alpha_x\right) u_{1,1} + \left(a_p + b_m\alpha_x\beta_y + a_m\beta_y + b_p\alpha_x\right) u_{1,2} \\
&\quad + \left(b + b_m\alpha_y\beta_x + b_m\alpha_y + b\beta_x\right) u_{2,1} + \left(b_p + b_m\beta_y + b_p\beta_x + b_m\beta_x\beta_y\right) u_{2,2} \\
&= a^{(x,y)} u_{1,1} + a_p^{(x,y)} u_{1,2} + b^{(x,y)} u_{2,1} + b_p^{(x,y)} u_{2,2} \qquad (4.9) \\
&= A_{1,1:N_x\cdot N_y} \cdot \mathbf{u}_{1:N_x,1:N_y}.
\end{aligned}
$$

where $a^{(x,y)} = (a + b_m\alpha_x\alpha_y + a_m\alpha_y + b\alpha_x)$, $a_p^{(x,y)} = (a_p + b_m\alpha_x\beta_y + a_m\beta_y + b_p\alpha_x)$, $b^{(x,y)} = (b + b_m\alpha_y\beta_x + b_m\alpha_y + b\beta_x)$ and $b_p^{(x,y)} = (b_p + b_m\beta_y + b_p\beta_x + b_m\beta_x\beta_y)$. The process is repeated with the remaining rows giving the consolidated system $A\mathbf{u}_{1:N_x,1:N_y} = \mathbf{f}_{1:N_x,1:N_y}$. That is

$$
\begin{bmatrix}
a^{(x,y)} & b^{(x,y)} & & a_p^{(x,y)} & b_p^{(x,y)} & & & & & \\
b^{(y)} & a^{(y)} & b^{(y)} & b_p^{(y)} & a_p^{(y)} & b_p^{(y)} & & & & \\
 & b^{(x,y)} & a^{(x,y)} & & b_p^{(x,y)} & a_p^{(x,y)} & & & & \\
a_m^{(x)} & b_m^{(x)} & & a^{(x)} & b^{(x)} & & a_p^{(x)} & b_p^{(x)} & & \\
b_m & a_m & b_m & b & a & b & b_p & a_p & b_p & \\
 & b_m^{(x)} & a_m^{(x)} & & b^{(x)} & a^{(x)} & & b_p^{(x)} & a_p^{(x)} & \\
 & & & a_m^{(x,y)} & b_m^{(x,y)} & & a^{(x,y)} & b^{(x,y)} & & \\
 & & & b_m^{(y)} & a_m^{(y)} & b_m^{(y)} & b^{(y)} & a^{(y)} & b^{(y)} & \\
 & & & & b_m^{(x,y)} & a_m^{(x,y)} & & b^{(x,y)} & a^{(x,y)} &
\end{bmatrix}
\begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix}
=
\begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix}
$$

where the elements of A are defined like those in (4.9).

4.3 Summary

Chapter 4 presented the definitions of the Dirichlet and Sommerfeld-like boundary conditions and developed the approximations for the Sommerfeld-like conditions. A two-dimensional example demonstrated the boundary conditions' implementations. Chapter 5 will present a highly parallel algorithm for the solution of 9- and 27-diagonal linear systems satisfying a set of required conditions.

5 Direct FFT Solver

An efficient parallel direct solver is presented utilizing the second-, fourth- and sixth-order compact schemes given in Chapters 2 and 3. The defining assumption is that the coefficient, k, only varies in the vertical direction. That is $k(x,y) = k(y)$ and $k(x,y,z) = k(z)$ within the domain in two- and three-dimensions, respectively. The solver can be used in the case with Dirichlet conditions on all boundaries or Sommerfeld-like conditions on the top and bottom boundaries with respect to the vertical direction and Dirichlet on the others. The following vector notation is observed. The vector $\mathbf{v}_l \in \mathbb{C}^{N_x \cdot N_y}$ is such that

$$\mathbf{v}_l = \begin{bmatrix} v_{1,1,l} & v_{2,1,l} & \cdots & v_{N_x,1,l} & v_{1,2,l} & v_{2,2,l} & \cdots & v_{N_x,2,l} & \cdots & v_{N_x,N_y,l} \end{bmatrix}^T \tag{5.1}$$

for $l = 1, \ldots, N_z$. In the two-dimensional case $\mathbf{v}_l \in \mathbb{C}^{N_x}$ for $l = 1, \ldots, N_y$.

5.1 Stencils

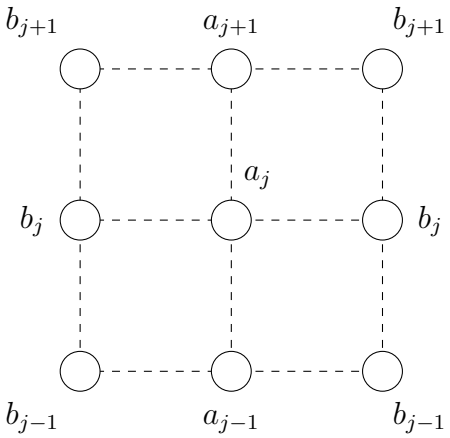
The second-, fourth- and sixth-order compact schemes in two-dimensions can be presented in the stencil form shown below. Figure 5.1 graphically shows the 9-point stencil. Any compact scheme with these stencil coefficient patterns can be expressed at every grid point (i,j) as

$$\sum_{\nu=j-1}^{j+1} \left(b_\nu \left[u_{i-1,\nu} + u_{i+1,\nu} \right] + a_\nu u_{i,\nu} \right) = f_{i,j}.$$

The equation corresponds to the $(i + (j-1) \cdot N_x) - th$ row in the resulting linear system $A\mathbf{u} = \mathbf{f}$ where the vectors $\mathbf{u}$, $\mathbf{f} \in \mathbb{C}^{N_x \cdot N_y}$ are such that $\mathbf{u} = \begin{bmatrix} \mathbf{u}_l & \cdots & \mathbf{u}_{N_y} \end{bmatrix}^T$ is the solution vector and $\mathbf{f} = \begin{bmatrix} \mathbf{f}_l & \cdots & \mathbf{f}_{N_y} \end{bmatrix}^T$ is the right-hand side of the numerical scheme.

Likewise, the compact schemes in the three-dimensional case can be expressed at every

Figure 5.1: FFT Solver 9-Point Stencil



grid point (i, j, l) as

$$\sum_{\nu=l-1}^{l+1} (d_\nu \left[u_{i-1,j-1,\nu} + u_{i-1,j+1,\nu} + u_{i+1,j-1,\nu} + u_{i+1,j+1,\nu}\right]$$

$$+ b_\nu \left[u_{i-1,j,\nu} + u_{i+1,j,\nu}\right] + c_\nu \left[u_{i,j-1,\nu} + u_{i,j+1,\nu}\right] + a_\nu u_{i,j,\nu}) = f_{i,j,l}. \tag{5.2}$$

This stencil expression is illustrated in Figure 5.2. Equations (5.2) corresponds to the $(i + (j-1) \cdot N_x + (l-1) \cdot N_x \cdot N_y) - th$ row in the resulting linear system

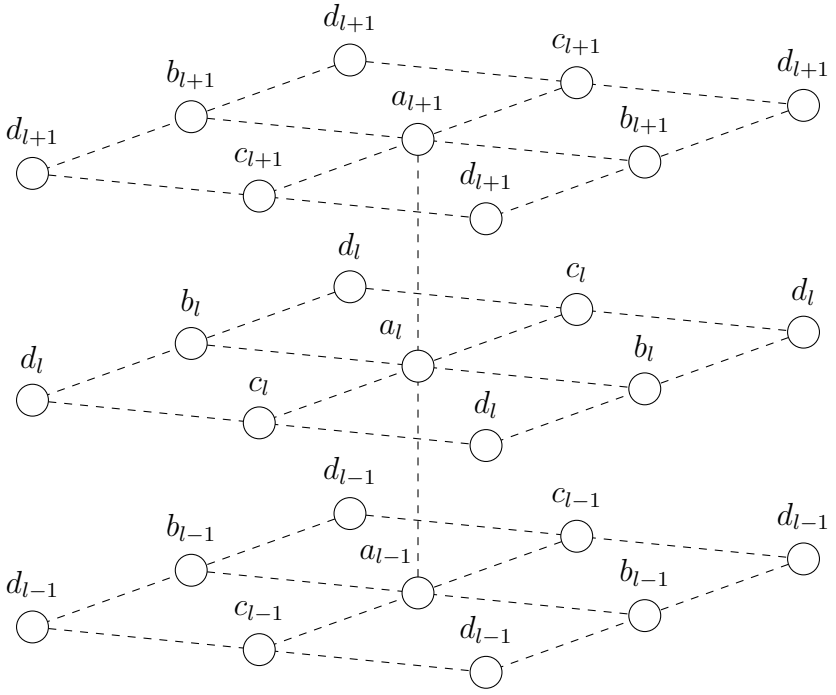
$$A\mathbf{u} = \mathbf{f} \tag{5.3}$$

where $\mathbf{u}, \mathbf{f} \in \mathbb{C}^{N_x \cdot N_y \cdot N_z}$ such that $\mathbf{u} = \begin{bmatrix}\mathbf{u}_l & \cdots & \mathbf{u}_{N_z}\end{bmatrix}^T$ and $\mathbf{f} = \begin{bmatrix}\mathbf{f}_l & \cdots & \mathbf{f}_{N_z}\end{bmatrix}^T$.

5.2 Coefficients

Section 5.2 explicitly defines the coefficients of the compact schemes from Chapters 2 and 3. The coefficients follow from multiplying both sides of the linear systems by h_y^2 in two-dimensions and h_z^2 in three-dimensions. Recall that k varies only vertically in the domain, thus $k_{i,j} = k_j$ and $k_{i,j,l} = k_l$ in two- and three-dimensions respectively. For the convection-

Figure 5.2: FFT Solver 27-Point Stencil



diffusion equation, assume that vertical convection is dominant. That is, $\mathbf{a} = (0, 0, a_3)$. Let $R_{yx} = h_y^2/h_x^2$, $R_{zx} = h_z^2/h_x^2$ and $R_{zy} = h_z^2/h_y^2$.

5.2.1 Second-Order

Two-dimensions:

$$a_j = h_y^2 k_j^2 - 2\left(R_{yx} + 1\right), \qquad b_j = R_{yx}, \qquad a_{j\pm1} = 1, \qquad b_{j\pm1} = 0$$

Three-dimensions:

$$b_l = R_{zx}, \qquad c_l = R_{zy}, \qquad a_{l\pm1} = 1, \qquad a_l = h_z^2 k_l^2 - 2\left(R_{zx} + R_{zy} + 1\right)$$

$$d_l = d_{l\pm1} = c_{l\pm1} = b_{l\pm1} = 0$$

5.2.2 Fourth-Order

Two-dimensions:

$$a_j = -\frac{5R_{yx}}{3} - \frac{5}{3} + \frac{2h_y^2 k_j^2}{3} \qquad b_j = \frac{5R_{yx}}{6} - \frac{1}{6} + \frac{h_y^2 k_j^2}{12}$$

$$a_{j\pm1} = \frac{5}{6} - \frac{R_{yx}}{6} + \frac{h_y^2 k_{j\pm1}^2}{12} \qquad b_{j\pm1} = \frac{R_{yx}}{12} + \frac{1}{12}$$

Three-dimensions:

$$d_{l\pm1} = 0\,, \quad b_{l\pm1} = \frac{R_{zx}+1}{12}\,, \quad c_{l\pm1} = \frac{R_{zy}+1}{12}\,, \quad a_{l\pm1} = \frac{h_z^2 k_{l\pm1}^2}{12} + \frac{1}{6}\left(4 - R_{zx} - R_{zy}\right)$$

$$d_l = \frac{R_{zx}+R_{zy}}{12}\,, \quad b_l = \frac{h_z^2 k_l^2}{12} + \frac{1}{6}\left(4R_{zx} - R_{zy} - 1\right)\,, \quad c_l = \frac{h_z^2 k_l^2}{12} + \frac{1}{6}\left(4R_{zy} - R_{zx} - 1\right)$$

$$a_l = \frac{h_z^2 k_l^2}{2} - \frac{4}{3}\left(R_{zx} + R_{zy} + 1\right)$$

Convection-Diffusion:

$$d_{l\pm1} = 0\,, \quad b_{l\pm1} = \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(1 + R_{zx}\right)\,, \quad c_{l\pm1} = \frac{1}{12}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(1 + R_{zy}\right)$$

$$a_{l\pm1} = \frac{2}{3} + \frac{a_3^2 h_z^2}{12} \mp \frac{a_3 h_z}{3} - \frac{1}{6}\left(1 \mp \frac{a_3 h_z}{2}\right)\left(R_{zx} + R_{zy}\right)$$

$$d_l = \frac{1}{12}\left(R_{zx} + R_{zy}\right)\,, \quad b_l = \frac{2}{3}R_{zx} - \frac{1}{6}\left(R_{zy} + 1\right)\,, \quad c_l = \frac{2}{3}R_{zy} - \frac{1}{6}\left(R_{zx} + 1\right)$$

$$a_l = -\frac{4}{3}\left(R_{zx} + R_{zy} + 1\right) - \frac{h_z^2 a_3^2}{6}$$

5.2.3 Sixth-Order

Recall that in the sixth-order case $h = h_x = h_y = h_z$.

Two-dimensions:

$$a_j = -\frac{10}{3} + \frac{41h^2k_j^2}{45} + \frac{h^4}{20}\left(\frac{\partial^2}{\partial y^2}k_j^2 - k_j^4\right) \qquad b_j = \frac{2}{3} + \frac{h^2k_j^2}{90}$$

$$a_{j\pm1} = \frac{2}{3} + \frac{h^2k_{j\pm1}^2}{90} \pm \left(\frac{h^3}{30} + \frac{h^5k_{j\pm1}^2}{120}\right)\frac{\partial}{\partial y}k_j^2 \qquad b_{j\pm1} = \frac{1}{6} + \frac{h^2k_{j\pm1}^2}{90} \pm \frac{h^3}{120}\frac{\partial}{\partial y}k_j^2$$

Three-dimensions:

$$d_{l\pm1} = \frac{1}{30}\,, \qquad b_{l\pm1} = c_{l\pm1} = \frac{1}{10} + \frac{h^2k_{l\pm1}^2}{90} \pm \frac{h^3}{120}\frac{\partial}{\partial z}k_l^2\,,$$

$$a_{l\pm1} = \frac{7}{15} - \frac{h^2k_{l\pm1}^2}{90} \pm \frac{h^3}{60}\frac{\partial}{\partial z}k_l^2\left(1 + \frac{h^2k_{l\pm1}^2}{2}\right)$$

$$d_l = \frac{1}{10} + \frac{h^2k_l^2}{90}\,, \qquad b_l = c_l = \frac{7}{15} - \frac{h^2k_l^2}{90}\,, \qquad a_l = -\frac{64}{15} + \frac{14h^2k_l^2}{15} + \frac{h^4}{20}\left(\frac{\partial^2}{\partial z^2}k_l^2 - k_l^4\right)$$

5.3 Solver

Section 5.3 describes a highly parallel direct solver that utilizes FFT. Only the three-dimensional case is considered, as the two-dimensional solver can be derived through its inferred similarities. The numerical scheme (5.2) can be presented in block tridiagonal form written as

$$C_1\mathbf{u}_1 + C_{p,1}\mathbf{u}_2 = \mathbf{f}_1$$

$$C_{m,l}\mathbf{u}_{l-1} + C_l\mathbf{u}_l + C_{p,l}\mathbf{u}_{l+1} = \mathbf{f}_l, \qquad \text{for } l = 2, \ldots, N_z - 1$$

$$C_{m,N_z}\mathbf{u}_{N_z-1} + C_{N_z}\mathbf{u}_{N_z} = \mathbf{f}_{N_z}$$

The nine-diagonal matrices $C_{m,l}$, C_l and $C_{p,l}$ are determined by the coefficients in the preceding section, their definition follows. If $n \in \mathbb{Z}$, let $\mathcal{I}$ be the mapping such that $\mathcal{I}(n)$ is the $n \times n$ matrix with ones in the entries where the indices i and j differ by 1, that is $|i - j| = 1$.

For example:

$$\mathcal{I}(4) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Similarly, let $I(n)$ be the $n \times n$ identity matrix. Then

$$C_{m,l} = I(N_y) \otimes [b_{l-1}\mathcal{I}(N_x) + a_{l-1}I(N_x)] + \mathcal{I}(N_y) \otimes [c_{l-1}I(N_x) + d_{l-1}\mathcal{I}(N_x)]$$

$$C_{l} = I(N_y) \otimes [b_{l}\mathcal{I}(N_x) + a_{l}I(N_x)] + \mathcal{I}(N_y) \otimes [c_{l}I(N_x) + d_{l}\mathcal{I}(N_x)]$$

$$C_{p,l} = I(N_y) \otimes [b_{l+1}\mathcal{I}(N_x) + a_{l+1}I(N_x)] + \mathcal{I}(N_y) \otimes [c_{l+1}I(N_x) + d_{l+1}\mathcal{I}(N_x)]$$

5.3.1 Eigenvalues

To develop the eigenvalues and eigenvectors (eigenpairs) of the matrices $C_{m,l}$, C_l and $C_{p,l}$, consider the following lemmas and theorems.

Lemma 5.1. *Let* $\beta_{r,s}^{i,j} = \sin(r\pi i/(N_x+1)) \sin(s\pi j/(N_y+1))$ *where* $r, i \in \{1, \ldots, N_x\}$ *and* $s, j \in \{1, \ldots, N_y\}$. *Define* $\mathbf{v}_{i,j} = \begin{bmatrix} \beta_{1,1}^{i,j} & \beta_{2,1}^{i,j} & \cdots & \beta_{N_x,1}^{i,j} & \beta_{1,2}^{i,j} & \cdots & \beta_{N_x,N_y}^{i,j} \end{bmatrix}^T$. *Then* $\mathbf{v}_{i,j}$ *is an eigenvector of* $B = \mathcal{I}(N_x \cdot N_y)$ *with corresponding eigenvalue* $\lambda_i = 2\cos(\pi i/(N_x+1))$.

Proof. Recall the trigonometric identity $2\sin(\alpha)\cos(\beta) = \sin(\alpha-\beta)+\sin(\alpha+\beta)$ [25]. Then

$$\begin{aligned} \lambda_l \beta_{r,s}^{i,j} &= 2\cos\left(\frac{\pi i}{N_x+1}\right) \sin\left(\frac{r\pi i}{N_x+1}\right) \sin\left(\frac{s\pi j}{N_y+1}\right) \\ &= \left[\sin\left(\frac{(r-1)\pi i}{N_x+1}\right) + \sin\left(\frac{(r+1)\pi l}{N_x+1}\right)\right] \sin\left(\frac{s\pi j}{N_y+1}\right) \\ &= \beta_{r-1,s}^{i,j} + \beta_{r+1,s}^{i,j}. \end{aligned}$$

Note that $\beta^{i,j}_{N_x+1,s} = 0 = \beta^{i,j}_{0,s}$. Thus

$$B\mathbf{v}_{i,j} = \begin{bmatrix} \beta^{i,j}_{2,1} & \beta^{i,j}_{1,1} + \beta^{i,j}_{3,1} & \cdots & \beta^{i,j}_{N_x-1,1} + \beta^{i,j}_{1,1} & \beta^{i,j}_{N_x,2} + \beta^{i,j}_{2,2} & \cdots & \beta^{i,j}_{N_x-1,N_y} \end{bmatrix}^T = \lambda_i \mathbf{v}_{i,j}. \quad \blacksquare$$

Lemma 5.2. *Let $B = (b_{\iota,\kappa}) \in \mathbb{R}^{N_x \cdot N_y \times N_x \cdot N_y}$ be such that $b_{\iota,\kappa} = 1$ when $|\iota - \kappa| = N_x$ and $b_{\iota,\kappa} = 0$ otherwise. Then $\mathbf{v}_{i,j}$, as defined in Lemma 5.1, is an eigenvector of B with corresponding eigenvalue $\lambda_j = 2\cos(\pi j/(N_y+1))$.*

Proof. First, observe that

$$\begin{aligned} \lambda_j \beta^{i,j}_{r,s} &= 2\cos\left(\frac{\pi j}{N_y+1}\right)\sin\left(\frac{r\pi i}{N_x+1}\right)\sin\left(\frac{s\pi j}{N_y+1}\right) \\ &= \left[\sin\left(\frac{(s-1)\pi j}{N_y+1}\right) + \sin\left(\frac{(s+1)\pi j}{N_y+1}\right)\right]\sin\left(\frac{r\pi i}{N_x+1}\right) \\ &= \beta^{i,j}_{r,s-1} + \beta^{i,j}_{r,s+1}. \end{aligned}$$

Note that $\beta^{i,j}_{r,N_y+1} = 0 = \beta^{i,j}_{r,0}$. Thus

$$B\mathbf{v}_{i,j} = \begin{bmatrix} \beta^{i,j}_{1,2} & \beta^{i,j}_{2,2} & \cdots & \beta^{i,j}_{N_x,2} & \beta^{i,j}_{1,1} + \beta^{i,j}_{1,3} & \cdots & \beta^{i,j}_{N_x,N_y-1} \end{bmatrix}^T = \lambda \mathbf{v}_{i,j}. \quad \blacksquare$$

Theorem 5.1. *Let $\mathbf{v}$ and λ_i be an eigenpair of the matrix A_i for $i = 1, \ldots, n$ where $n \in \mathbb{N}$ such that $n > 1$. Then $\sum_{i=1}^{n} \lambda_i$ is an eigenvalue of the matrix $\sum_{i=1}^{n} A_i$ with corresponding eigenvector $\mathbf{v}$.*

Proof. $\left(\sum_{i=1}^{n} A_i\right)\mathbf{v} = \sum_{i=1}^{n} A_i\mathbf{v} = \sum_{i=1}^{n} \lambda_i \mathbf{v} = \left(\sum_{i=1}^{n} \lambda_i\right)\mathbf{v}$ $\blacksquare$

Theorem 5.2. *Let $B_1, B_2 \in \mathbb{R}^{n\times n}$ for $n \in \mathbb{N}$ such that $n > 1$. If $(\lambda_1, \mathbf{v})$ and $(\lambda_2, \mathbf{v})$ are eigenpairs of B_1 and B_2 respectively then $(\lambda_2\lambda_1, \mathbf{v})$ is an eigenpair of B_1B_2.*

Proof. $B_1B_2\mathbf{v} = B_1(B_2\mathbf{v}) = B_1(\lambda_2\mathbf{v}) = \lambda_2 B_1\mathbf{v} = \lambda_2\lambda_1\mathbf{v}$ $\blacksquare$

Therefore, by the preceding lemmas and theorems, the vectors $\mathbf{v}_{i,j}$ as defined in Lemma 5.1 and

$$\lambda_{i,j,\nu} = 4d_\nu \cos\left(\frac{i\pi}{N_x+1}\right)\cos\left(\frac{j\pi}{N_y+1}\right) + 2b_\nu \cos\left(\frac{i\pi}{N_x+1}\right) + 2c_\nu \cos\left(\frac{j\pi}{N_y+1}\right) + a_\nu$$

where $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $\nu = l-1, l, l+1$, form eigenpairs for the coefficient matrices $C_{m,l}$, C_l and $C_{p,l}$. Note that any nonzero vector that is a scalar multiple of $\mathbf{v}_{i,j}$ also forms eigenpairs with the eigenvalues above.

5.3.2 Diagonalization

The direct FFT algorithm requires the diagonalization of the coefficient matrices via multiplication of an orthogonal matrix, specifically the matrix of eigenvectors $\mathbf{v}_{i,j}$ after scaling. The following lemmas show the necessary scaling of these vectors and the orthogonality of this matrix.

Lemma 5.3. *Let* $\beta_r^\iota = \sin\left(r\pi\iota/(N_x+1)\right)$ *where* $r, \iota \in \{1, \ldots, N_x\}$. *Define*

$$\mathbf{v}_\iota = \begin{bmatrix}\beta_1^\iota & \beta_2^\iota & \cdots & \beta_{N_x}^\iota\end{bmatrix}^T.$$

Then $||\mathbf{v}_\iota||_2^2 = \langle \mathbf{v}_\iota, \mathbf{v}_\iota \rangle = (N_x+1)/2$.

Proof. Recall the trigonometric identity $\sin^2(\beta) = (1-\cos(2\beta))/2$ [25]. Note that $\exp\{i\alpha\} = \cos\alpha + i\sin\alpha$ where $i = \sqrt{-1}$. Also, consider the geometric series $\sum_{k=1}^{n} z^k = \dfrac{1-z^{n+1}}{1-z} - 1$ for $z \in \mathbb{C}$ [26]. Then

$$\begin{aligned}\langle \mathbf{v}_\iota, \mathbf{v}_\iota \rangle &= \sum_{r=1}^{N_x} \sin^2\left(\frac{r\pi\iota}{N_x+1}\right) \\ &= \frac{N_x}{2} - \frac{1}{2}\sum_{r=1}^{N_x} \cos\left(\frac{2r\pi\iota}{N_x+1}\right)\end{aligned}$$

$$= \frac{N_x}{2} - \frac{1}{2}\sum_{r=1}^{N_x} Re\left[\exp\left\{\frac{2r\iota\pi i}{N_x+1}\right\}\right]$$

$$= \frac{N_x+1}{2} - \frac{1}{2}Re\left[\frac{1-\exp\{2\iota\pi i\}}{1-\exp\left\{\frac{2\iota\pi i}{N_x+1}\right\}}\right].$$

Now let $\zeta = 1-\exp\{2\iota\pi i\}$ and $\eta = 1-\exp\left\{\frac{2\iota\pi i}{N_x+1}\right\}$. Note that

$$\frac{\zeta}{\eta} = \frac{Re(\zeta)Re(\eta)+Im(\zeta)Im(\eta)}{Re(\eta)^2+Im(\eta)^2} + i\frac{Im(\zeta)Re(\eta)-Re(\zeta)Im(\eta)}{Re(\eta)^2+Im(\eta)^2}$$

by division of complex numbers. Then

$$Re\left[\frac{1-\exp\{2\iota\pi i\}}{1-\exp\left\{\frac{2\iota\pi i}{N_x+1}\right\}}\right] = \frac{Re(\zeta)Re(\eta)+Im(\zeta)Im(\eta)}{Re(\eta)^2+Im(\eta)^2}$$

$$= \frac{(1-\cos(2\iota\pi))\cdot Re(\eta) - (\sin(2\iota\pi))\cdot Im(\eta)}{Re(\eta)^2+Im(\eta)^2} = 0$$

since $\iota \in \mathbb{Z}$. ■

Lemma 5.4. *Let* $\mathbf{v}_{i,j}$ *be as in Lemma 5.1. Then* $||\mathbf{v}_{i,j}||_2^2 = (N_x+1)(N_y+1)/4$ *for* $i = 1,\ldots,N_x$ *and* $j = 1,\ldots,N_y$.

Proof. Lemma 5.3 gives

$$||\mathbf{v}_{i,j}||_2^2 = \sum_{r=1}^{N_x}\sum_{s=1}^{N_y}\left[\sin\left(\frac{r\pi i}{N_x+1}\right)\sin\left(\frac{s\pi j}{N_y+1}\right)\right]^2$$

$$= \sum_{r=1}^{N_x}\sin^2\left(\frac{r\pi i}{N_x+1}\right)\left[\sum_{s=1}^{N_y}\sin^2\left(\frac{s\pi j}{N_y+1}\right)\right]$$

$$= \sum_{n=1}^{N_x}\sin^2\left(\frac{r\pi i}{N_x+1}\right)\left[\frac{N_y+1}{2}\right]$$

$$= \frac{N_y+1}{2}\sum_{r=1}^{N_x}\sin^2\left(\frac{r\pi i}{N_x+1}\right)$$

$$= \left(\frac{N_y+1}{2}\right)\left(\frac{N_x+1}{2}\right). \qquad \blacksquare$$

Lemma 5.5. *Let $\mathbf{v}_\iota$ be as in Lemma 5.3. If $\iota \neq \iota'$ then $\mathbf{v}_\iota$ and $\mathbf{v}_{\iota'}$ are orthogonal.*

Proof. First, note the trigonometric identity $2\sin(\alpha)\sin(\beta) = \cos(\alpha-\beta) - \cos(\alpha+\beta)$ [25]. Then suppose $\iota \neq \iota'$. It follows

$$\begin{aligned}
\langle \mathbf{v}_\iota, \mathbf{v}_{\iota'} \rangle &= \sum_{n=1}^{N_x} \sin\left(\frac{\iota n\pi}{N_x+1}\right)\sin\left(\frac{\iota' n\pi}{N_x+1}\right) \\
&= \frac{1}{2}\sum_{n=1}^{N_x}\left[\cos\left(\frac{(\iota-\iota')n\pi}{N_x+1}\right) - \cos\left(\frac{(\iota+\iota')n\pi}{N_x+1}\right)\right] \\
&= \frac{1}{2}\sum_{n=1}^{N_x}\left[Re\left(\exp\left\{\frac{i(\iota-\iota')n\pi}{N_x+1}\right\}\right) - Re\left(\exp\left\{\frac{i(\iota+\iota')n\pi}{N_x+1}\right\}\right)\right] \\
&= \frac{1}{2}Re\left(\sum_{n=1}^{N_x}\exp\left\{\frac{i(\iota-\iota')n\pi}{N_x+1}\right\} - \sum_{n=1}^{N_x}\exp\left\{\frac{i(\iota+\iota')n\pi}{N_x+1}\right\}\right) \\
&= \frac{1}{2}Re\left(\frac{1-\exp\{i(\iota-\iota')\pi\}}{1-\exp\left\{\frac{i(\iota-\iota')n\pi}{N_x+1}\right\}} - \frac{1-\exp\{i(\iota+\iota')\pi\}}{1-\exp\left\{\frac{i(\iota+\iota')n\pi}{N_x+1}\right\}}\right).
\end{aligned}$$

Note that

$$\sigma := 1-\exp\{i(\iota-\iota')\pi\} = 1-\cos((\iota-\iota')\pi) - \sin((\iota-\iota')\pi) = 1-\cos((\iota-\iota')\pi)$$

$$\gamma := 1-\exp\{i(\iota+\iota')\pi\} = 1-\cos((\iota+\iota')\pi) - \sin((\iota+\iota')\pi) = 1-\cos((\iota+\iota')\pi).$$

and that $\iota-\iota'$ is even if and only if $\iota+\iota'$ is even. Therefore, if $\iota-\iota'$ is even then $\sigma=\gamma=0$ and $\langle \mathbf{v}_\iota, \mathbf{v}_{\iota'}\rangle = 0$. Similarly, if $\iota-\iota'$ is odd, then $\sigma=\gamma=2$. Then, let $\zeta = \dfrac{i\pi(\iota-\iota')}{2(N_x+1)}$ and $\eta = \dfrac{i\pi(\iota+\iota')}{2(N_x+1)}$. It follows,

$$\langle \mathbf{v}_\iota, \mathbf{v}_{\iota'}\rangle = Re\left[\frac{1}{1-\exp\left\{\frac{i(\iota-\iota')n\pi}{N_x+1}\right\}} - \frac{1}{1-\exp\left\{\frac{i(\iota+\iota')n\pi}{N_x+1}\right\}}\right]$$

$$= Re\left[\frac{1-\exp\left\{\frac{i\pi(\iota+\iota')}{N_x+1}\right\}-\left(1-\exp\left\{\frac{i\pi(\iota-\iota')}{N_x+1}\right\}\right)}{\left(1-\exp\left\{\frac{i\pi(\iota-\iota')}{N_x+1}\right\}\right)\left(1-\exp\left\{\frac{i\pi(\iota+\iota')}{N_x+1}\right\}\right)}\right]$$

$$= Re\left[\frac{\exp\left\{\frac{i\pi\iota}{N_x+1}\right\}\left(\exp\left\{\frac{-i\pi\iota'}{N_x+1}\right\}-\exp\left\{\frac{i\pi\iota'}{N_x+1}\right\}\right)}{\exp\{\zeta\}\left(\exp\{-\zeta\}-\exp\{\zeta\}\right)\exp\{\eta\}\left(\exp\{-\eta\}-\exp\{\eta\}\right)}\right]$$

$$= Re\left[\frac{\exp\left\{\frac{-i\pi\iota'}{N_x+1}\right\}-\exp\left\{\frac{i\pi\iota'}{N_x+1}\right\}}{\left(\exp\{-\zeta\}-\exp\{\zeta\}\right)\left(\exp\{-\eta\}-\exp\{\eta\}\right)}\right]$$

$$= Re\left[\frac{-2i\sin\left(\frac{\pi\iota'}{N_x+1}\right)}{2i\sin\left(\frac{\pi(\iota-\iota')}{2(N_x+1)}\right)\cdot 2i\sin\left(\frac{\pi(\iota+\iota')}{2(N_x+1)}\right)}\right]=0$$

since $2i\sin(\theta)=\exp\{i\theta\}-\exp\{-i\theta\}$. ■

Lemma 5.6. *Let $\mathbf{v}_{i,j}$ be defined as in Lemma 5.1. If $i\neq i'$ or $j\neq j'$ then $\mathbf{v}_{i,j}$ and $\mathbf{v}_{i',j'}$ are orthogonal.*

Proof. Let $\alpha_r^{l,l'}=\sin\left(ir\pi/(N_x+1)\right)\sin\left(i'r\pi/(N_x+1)\right)$. Suppose $j\neq j'$. Then

$$\sum_{s=1}^{N_y}\sin\left(\frac{js\pi}{N_y+1}\right)\sin\left(\frac{j's\pi}{N_y+1}\right)=0$$

by Lemma 5.5. It follows

$$\langle\mathbf{v}_{i,j},\mathbf{v}_{i',j'}\rangle=\sum_{r=1}^{N_x}\sum_{s=1}^{N_y}\alpha_r^{i,i'}\sin\left(\frac{js\pi}{N_y+1}\right)\sin\left(\frac{j's\pi}{N_y+1}\right)$$

$$=\sum_{n=1}^{N_x}\left[\alpha_r^{i,i'}\sum_{s=1}^{N_y}\sin\left(\frac{js\pi}{N_y+1}\right)\sin\left(\frac{j's\pi}{N_y+1}\right)\right]=0.$$

The same can be shown for $l\neq l'$. ■

Lemma 5.7. *Let $\mathbf{v}_{i,j}$ be as in Lemma 5.1. Define $\mathbf{w}_{i,j}=||\mathbf{v}_{i,j}||^{-1}\mathbf{v}_{i,j}$. Let*

$$V=\begin{bmatrix}\mathbf{w}_{1,1} & \mathbf{w}_{2,1} & \cdots & \mathbf{w}_{N_x,1} & \mathbf{w}_{1,2} & \cdots & \mathbf{w}_{N_x,N_y}\end{bmatrix}\in\mathbb{R}^{N_x\cdot N_y\times N_x\cdot N_y}.$$

Then V is an orthogonal matrix.

Proof. Let V be as defined above. Then

$$V^TV = V^T\begin{bmatrix}\mathbf{w}_{1,1} & \mathbf{w}_{2,1} & \cdots & \mathbf{w}_{N_x,1} & \mathbf{w}_{1,2} & \cdots & \mathbf{w}_{N_x,N_y}\end{bmatrix}$$

$$= \begin{bmatrix}V^T\mathbf{w}_{1,1} & V^T\mathbf{w}_{2,1} & \cdots & V^T\mathbf{w}_{N_x,1} & V^T\mathbf{w}_{1,2} & \cdots & V^T\mathbf{w}_{N_x,N_y}\end{bmatrix}$$

$$= \begin{bmatrix} \langle \mathbf{w}_{1,1}, \mathbf{w}_{1,1}\rangle & \langle \mathbf{w}_{1,1}, \mathbf{w}_{2,1}\rangle & \cdots & \langle \mathbf{w}_{1,1}, \mathbf{w}_{N_x,N_y}\rangle \\ \langle \mathbf{w}_{2,1}, \mathbf{w}_{1,1}\rangle & \langle \mathbf{w}_{2,1}, \mathbf{w}_{2,1}\rangle & & \\ \vdots & & \ddots & \vdots \\ \langle \mathbf{w}_{N_x,N_y}, \mathbf{w}_{1,1}\rangle & \langle \mathbf{w}_{N_x,N_y}, \mathbf{w}_{2,1}\rangle & \cdots & \langle \mathbf{w}_{N_x,N_y}, \mathbf{w}_{N_x,N_y}\rangle \end{bmatrix} = I$$

by Lemmas 5.5 and 5.6. ■

Theorem 5.3. *Let λ_i and $\mathbf{v}_i$ for $i = 1, \ldots, n$ be eigenpairs of $B \in \mathbb{R}^{n\times n}$ where $n \in \mathbb{N}$. If $V = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_n\end{bmatrix}$ is orthogonal then $V^TBV = \Lambda$ is the diagonal matrix $\begin{bmatrix}\lambda_1 & \cdots & \lambda_n\end{bmatrix} I$.*

Proof. $BV = \begin{bmatrix}B\mathbf{v}_1 & \cdots & B\mathbf{v}_n\end{bmatrix} = \begin{bmatrix}\lambda_1\mathbf{v}_1 & \cdots & \lambda_n\mathbf{v}_n\end{bmatrix} = V\Lambda$ ■

Let V be the orthogonal matrix from in Lemma 5.7. Also, define $\mathbf{w}_l = V^T\mathbf{u}_l$, $\Lambda_{m,l} = V^TC_{m,l}V$, $\Lambda_l = V^TC_lV$ and $\Lambda_{p,l} = V^TC_{p,l}V$. Then

$$C_{m,l}\mathbf{u}_{l-1} + C_l\mathbf{u}_l + C_{p,l}\mathbf{u}_{l+1} = \mathbf{f}_l,$$

$$V^TC_{m,l}VV^T\mathbf{u}_{l-1} + V^TC_lVV^T\mathbf{u}_l + V^TC_{p,l}VV^T\mathbf{u}_{l+1} = V^T\mathbf{f}_l,$$

$$\Lambda_{m,l}\mathbf{w}_{l-1} + \Lambda_l\mathbf{w}_l + \Lambda_{p,l}\mathbf{w}_{l+1} = \widehat{\mathbf{f}}_l \tag{5.4}$$

where $l = 2, \ldots, N_z - 1$ and $\widehat{\mathbf{f}}_l = V^T\mathbf{f}_l$. The cases for $l = 1$ and $l = N_z$ are very similar. Note that $\Lambda_{m,l}$, Λ_l and $\Lambda_{p,l}$ are diagonal matrices of eigenvalues by Theorem 5.3, which yields $N_x{\cdot}N_y$ independent linear systems. The systems can be solved using the LU decomposition of the generated tridiagonal matrix with $\mathcal{O}(N_z)$ computational complexity. The computations in the solution are independent with respect to both the x and y directions of the computational

domain. Therefore, the computations can be parallelized with respect to the either direction. Prior to solving this system, $\widehat{\mathbf{f}}_l = V^T \mathbf{f}_l$ for $l = 1, \ldots, N_z$ must be found.

5.3.3 Finding the DST with FFT

Subsection 5.3.3 discusses the computation of the transformed right-hand side $V^T \mathbf{f}_l$. Note that $V^T \in \mathbb{R}^{N_x \cdot N_y \times N_x \cdot N_y}$ and $\mathbf{f}_l \in \mathbb{R}^{N_x \cdot N_y}$. Therefore, the computational complexity of the matrix vector multiplication of $V^T \mathbf{f}_l$ is $\mathcal{O}\left({N_x}^2 \cdot {N_y}^2\right)$. The calculation is required for each $\mathbf{f}_l$ where $l = 1, \ldots, N_z$. Transforming the right-hand side in this manner is not ideal even on modern computers. The following definitions provide tools to reduce the computational complexity.

Definition 5.1. *The **discrete sine transform** of the vector* $\mathbf{x} = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T \in \mathbb{C}^n$ *is given by* $DST(\mathbf{x}) = \begin{bmatrix} \widehat{x}_1 & \ldots & \widehat{x}_n \end{bmatrix}^T$ *where*

$$\widehat{x}_k = \sum_{l=1}^{n} \sin\left(\frac{kl\pi}{n+1}\right) x_l$$

for $k \in \{1, \ldots, n\}$ *[27].*

Thus $\widehat{\mathbf{f}}_l = V^T \mathbf{f}_l$, is simply the DST of $\mathbf{f}_l$ in both the x and y directions. The subsequent definition utilizes $i = \sqrt{-1}$ and i is not an index.

Definition 5.2. *The **discrete Fourier transform** of the vector* $\mathbf{y} = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix}^T \in \mathbb{C}^n$ *is given by* $\widehat{\mathbf{y}} = \begin{bmatrix} \widehat{y}_1 & \ldots & \widehat{y}_n \end{bmatrix}^T$ *where*

$$\begin{aligned} \widehat{y}_k &= \sum_{l=1}^{n} \exp\left\{\frac{-2\pi i(l-1)(k-1)}{n}\right\} y_l \\ &= \sum_{l=1}^{n} \left[\cos\left(\frac{2\pi(l-1)(k-1)}{n}\right) - i\sin\left(\frac{2\pi(l-1)(k-1)}{n}\right)\right] y_l \end{aligned}$$

for $k \in \{1, \ldots, n\}$ [27].

Lemma 5.8. *Let $\mathbf{x} \in \mathbb{C}^n$ and $\mathbf{y} \in \mathbb{C}^{2n+2}$. If $\mathbf{y} = \begin{bmatrix} 0 & \mathbf{x}^T & 0 & \cdots & 0 \end{bmatrix}^T$ then $\widehat{\mathbf{x}} = -Im\left(\widehat{\mathbf{y}}_{2:n+1}\right)$ where $\widehat{\mathbf{x}}$ is the DST of $\mathbf{x}$, $\widehat{\mathbf{y}}$ is the DFT of $\mathbf{y}$ and $-Im\left(\widehat{\mathbf{y}}_{2:n+1}\right)$ is the negative of the complex part of the vector elements in $\widehat{\mathbf{y}}$ from the second entry to the $(n+1)$-st entry.*

Proof. Let $N = 2n+2$, then $2/N = 1/(n+1)$. Define $y_1 = 0$, $y_{n+2} = y_{n+3} = \cdots = y_{2n+2} = 0$ and $y_l = x_{l-1}$ for $l = 2, \ldots, n+1$. Then for $k = 1, \ldots, n$,

$$\begin{aligned}
\widehat{x}_k &= \sum_{l=1}^{n} \sin\left(\frac{kl\pi}{n+1}\right) x_l \\
&= \sum_{l=1}^{n} \sin\left(\frac{kl\pi}{n+1}\right) y_{l+1} \\
&= \sum_{l=1}^{n+1} \sin\left(\frac{k(l-1)\pi}{n+1}\right) y_l \\
&= \sum_{l=1}^{N} \sin\left(\frac{2k(l-1)\pi}{N}\right) y_l \\
&= -Im\left(\sum_{l=1}^{N} \left[\cos\left(\frac{2\pi(l-1)k}{N}\right) - i\sin\left(\frac{2\pi(l-1)k}{N}\right)\right] y_l\right) \\
&= -Im(\widehat{y}_{k+1}).
\end{aligned}$$

■

Lemma 5.8 shows that computing the DST of a vector is equivalent to computing the DFT of a particular extension of that vector. The DFT can be computed using the FFT. Let FFT_x and FFT_y be the FFTs in the x and y directions respectively. Then $\widehat{\mathbf{f}}_l = V^T \mathbf{f}_l = FFT_y\left(FFT_x\left(\tilde{\mathbf{f}}_l\right)\right)$ where $\tilde{\mathbf{f}}_l$ is the appropriate extension of $\mathbf{f}_l$. The computational complexity of finding $\widehat{\mathbf{f}}_l$ is reduced to $\mathcal{O}\left(N_x \cdot N_y \log\left(N\right)\right)$ where $N = \max\{N_x, N_y\}$.

5.3.4 Block Tridiagonal Solver

To illustrate the process of the block tridiagonal solver, the LU decomposition of the diagonalized system is required. Let $\Lambda_l, \Lambda_{m,l}, \Lambda_{p,l} \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ be the diagonal matrices of eigenvalues as in (5.4). The decomposed matrix is then of the form $\Lambda = LU \in \mathbb{C}^{N_x \cdot N_y \cdot N_z \times N_x \cdot N_y \cdot N_z}$, given explicitly as

$$\begin{bmatrix} \Lambda_1 & \Lambda_{p,1} & & \\ \Lambda_{m,2} & \Lambda_2 & \ddots & \\ & \ddots & \ddots & \Lambda_{p,N_z-1} \\ & & \Lambda_{m,N_z} & \Lambda_{N_z} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ L_2 & 1 & & \\ & \ddots & \ddots & \\ & & L_{N_z} & 1 \end{bmatrix} \begin{bmatrix} U_1 & \Lambda_{p,1} & & \\ & U_2 & \ddots & \\ & & \ddots & \Lambda_{p,N_z-1} \\ & & & U_{N_z} \end{bmatrix}$$

where $U_1 = \Lambda_1$, $L_{l+1} = \Lambda_{m,l+1} \cdot U_l^{-1}$ and $U_{l+1} = \Lambda_{l+1} - L_{l+1}\Lambda_{p,l}$ for $l = 1, \ldots, N_z - 1$. As a diagonal matrix with nonzero entries, U_l^{-1} is easily found.

After transformation, the system (5.3) can be written $\widehat{\mathbf{f}} = \Lambda \mathbf{w} = LU\mathbf{w}$ where $\mathbf{w}$ is the transformed solution vector $\mathbf{u}$ as in (5.4). The tridiagonal linear system can be solved by first finding $\mathbf{y}$ in $L\mathbf{y} = \widehat{\mathbf{f}}$ and then solving for $\mathbf{w}$ in $U\mathbf{w} = \mathbf{y}$. The vector $\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 & \ldots & \mathbf{y}_{N_z} \end{bmatrix}^T$ is such that $\mathbf{y}_1 = \mathbf{f}_1$ and $\mathbf{y}_l = \mathbf{f}_l - L_l \mathbf{y}_{l-1}$ for $l = 2, \ldots, N_z$. The computations are clearly dependent in the z direction but there is no dependency in the x and y directions. The same is true for computing $\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 & \ldots & \mathbf{w}_{N_z} \end{bmatrix}^T$ were $\mathbf{w}_{N_z} = U_{N_z}^{-1}\mathbf{y}_{N_z}$ and $\mathbf{w}_l = U_l^{-1}\left(\mathbf{y}_l - \Lambda_{p,l}\mathbf{w}_{l+1}\right)$ for $l = N_z - 1, \ldots, 1$. The method is therefore highly parallel.

The solution vector $\mathbf{u}$ is simply the reverse transform of $\mathbf{w}$, that is $\mathbf{u}_l = V\mathbf{w}_l$ for $l = 1, \ldots, N_z$. As seen in the previous subsection, this matrix vector multiplication is equivalent to calculating $\mathbf{u}_l = V\mathbf{w}_l = FFT_y\left(FFT_x\left(\tilde{\mathbf{w}}_l\right)\right)$ where $\tilde{\mathbf{w}}_l$ is the appropriate extension of $\mathbf{w}_l$. The computational complexity of calculating $\mathbf{u}_l$ from $\mathbf{w}_l$ is $\mathcal{O}\left(N_x \cdot N_y \log\left(N\right)\right)$ where $N = \max\{N_x, N_y\}$.

5.3.5 Sequential Algorithm

Subsection 5.3.5 gives the algorithm for approximating the solution to (1.1,1.2) under the assumptions of Chapter 5. As shown in the preceding subsections, the process involves transforming the right-hand side, solving the resulting block tridiagonal system, and then reversing the transformation of the solution. Algorithm 1 details the solver.

Algorithm 1 FFT Solver

1: **for** $l = 1, \ldots, N_z$ **do**
2: 2D forward DST of the RHS in x and y directions
3: **end for**
4: **for** $j = 1, \ldots, N_y$; $i = 1, \ldots, N_x$ **do**
5: Solve the tridiagonal system using LU decomposition
6: **end for**
7: **for** $l = 1, \ldots, N_z$ **do**
8: 2D inverse DST of the solution in x and y directions
9: **end for**

In the C implementation of this algorithm and all those that follow, MIT's standard open-source C library FFTW [15] is used to compute the required fast Fourier transforms. The most computationally expensive sections in the FFT solver are these forward and reverse transformations with computational complexity $\mathcal{O}\left(N_x \cdot N_y \log\left(N\right)\right)$. Consequently, the transforms are the primary concerns for reducing the solver's wall-time in the parallel implementations.

5.4 Parallel Implementations

Section 5.4 provides a detailed examination of the parallelization of the developed direct FFT solver. The three-dimensional algorithm is provided and implies the necessary steps for the two-dimensional case. As stated in the introduction, the objective of computations in parallel is to reduce wall-time through evenly-distributed simultaneous processes. When im-

plemented correctly, wall-time is inversely related to the number of processes. OpenMP and MPI are the preferred libraries that facilitate computations in parallel and implementations of each are described below.

5.4.1 OpenMP

OpenMP provides a relatively simple implementation and excellent reduction in computation time, although speed-up is limited by the restriction to a single computer, or node. Algorithm 2 shows how OpenMP integrated into the direct FFT solver parallelizes the computations. Recall that the required forward and reverse transforms are independent with respect to z. OpenMP divides these DST computations as evenly as possible among the available processes. The solution of the tridiagonal system is independent in both the x and y directions and therefore the iterations of both loops are evenly distributed among processes.

Algorithm 2 OpenMP Parallel FFT Solver

```
#pragma omp parallel for
for l = 1, ..., N_z do
    2D forward DST of the RHS in x and y directions
end for
#pragma omp parallel for collapse(2)
for j = 1, ..., N_y; i = 1, ..., N_x do
    Solve the tridiagonal system using LU decomposition
end for
#pragma omp parallel for
for l = 1, ..., N_z do
    2D inverse DST of the solution in x and y directions
end for
```

A benefit of OpenMP is that it automatically distributes the work evenly among the processes, although this default operation consumes additional time. Manual division of iterations among processes is also an option. While desirable results were achieved with

the automatic division, a manual division was tested to improve the speed-up. After several experiments with various grid sizes and number of processes, no significant improvement was achieved. Therefore, all the results provided utilize the default division.

Another strategy was considered to improve performance. As seen in Algorithm 2, parallelization of the two-dimensional DST is possible. The transform is computed by finding the DST of all rows in the x direction and then all rows in the y direction as illustrated in Subsection 5.3.3. This permits the computations to be divided in either direction. A function in the FFTW library that accomplishes this with OpenMP is FFTW multi-threading [15]. The method was compared to the method of Algorithm 2, but no performance improvement was found. FFTW multi-threading is used in the hybrid implementation.

Despite the lack of benefit through the manual division of process iterations or parallelizing the two-dimensional DST, the OpenMP implementation demonstrates excellent, near-linear speed-up in wall-time. The speed-up was observed on both a multi-core personal computer and the single node of a cluster [16]. The solver is the ideal parallel tool in cases where the computational grid size does not overload the RAM of the machine.

5.4.2 MPI

MPI, the standard for communication between processes, has the capacity to utilize RAM across multiple nodes required for large-scale computational grids. The MPI implementation allocates the minimal memory required on each process's available memory. This allocation method enables the ability to run much larger computational grids as the program is no longer limited by the memory of a single node. Each process independently runs its own copy of the program on its assigned portion of the computations. The process passes mes-

sages, or transfers data, when necessary. Algorithm 3 outlines this procedure. However, communication time grows with the number of processes and can shrink the reduction in wall-time gained by utilizing more nodes, preventing a linear speed-up.

Algorithm 3 MPI Parallel FFT Solver

```
1: Find start_y, start_z, end_y and end_z using the process rank
2: for l = start_z, ..., end_z do
3:     2D forward DST of the RHS in x and y directions
4: end for
5: Transfer data via MPI to the appropriate processes
6: for j = start_y, ..., end_y; i = 1, ..., N_x do
7:     Solve the tridiagonal system using LU decomposition
8: end for
9: Transfer data via MPI to the appropriate processes
10: for l = start_z, ..., end_z do
11:     2D inverse DST of the solution in x and y directions
12: end for
```

A major difference from the OpenMP implementation is the entire MPI program must be parallelized, including the LU decomposition. Parallelization is required as the L and U arrays are the length $N_x \cdot N_y \cdot N_z$, which requires a large amount of memory for large grid sizes. To successfully run on multiple nodes, each node needs only the necessary parts of the arrays.

Figure 5.3 gives a graphical example of the transfer of data via MPI. For ease of illustration, the figure assumes the use of three processes. The first step shows how the domain is divided as evenly as possible among the three processes with respect to the vertical, z direction. The forward transform is computed here since the calculations do not depend on z. Once the DST of the right-hand side is computed, certain subsets of the domain need to be sent to different processes as the tridiagonal solver is dependent with respect to the z direction. The second step illustrates the subdomains that need to be sent and received

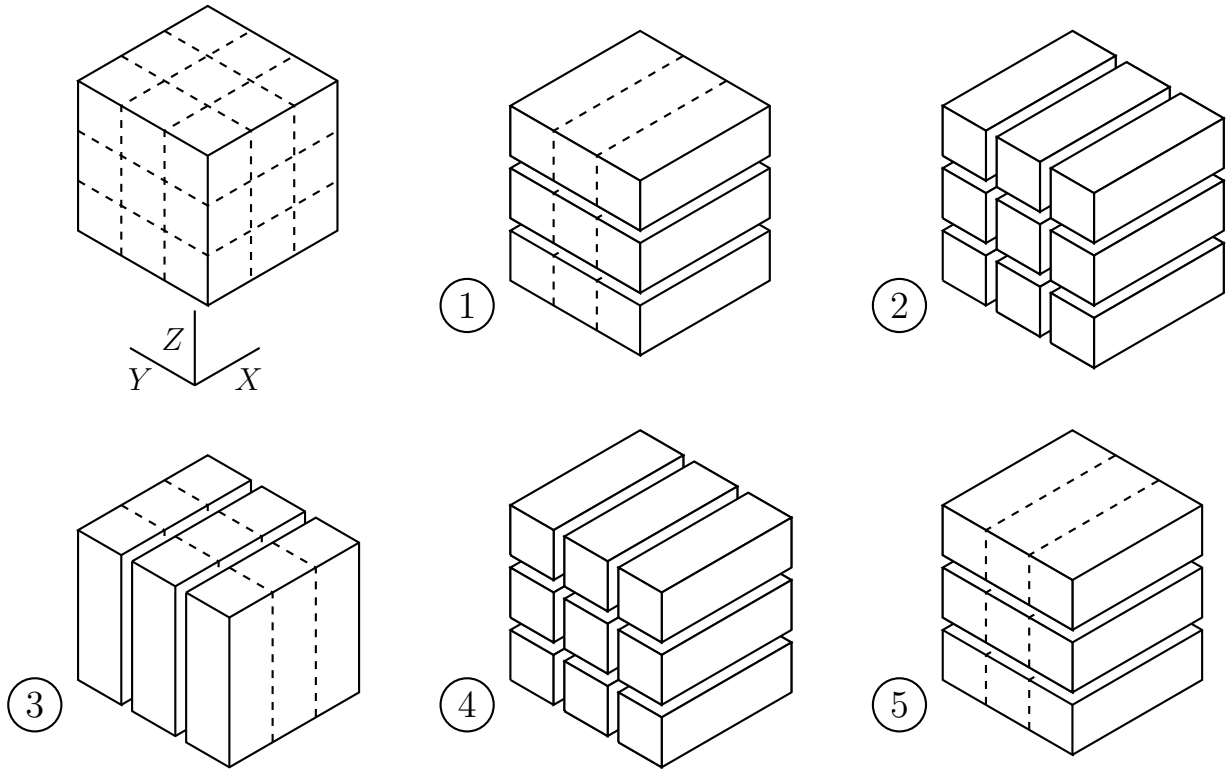
among processes. Note that the subdomains along the diagonal will not be sent as they currently reside on the appropriate process. The third step shows the sections of the domain assembled on the appropriate process after receiving the messages sent in the second step. Now the domain is divided as evenly as possible among the processes with respect to the y direction so that the tridiagonal solver can be calculated in parallel. The fourth step is simply the reverse of step two. The fifth and final step has the domain divided with respect to the z direction and can now calculate the reverse transform providing the numerical solution.

The MPI implementation demonstrates excellent, near linear wall-time speed-up on a multi-core personal computer and across multiple nodes in a large cluster [16]. This implementation's capabilities are immense, however it does have a limit. The number of MPI processes can not exceed $\min\{N_z, N_y\}$, otherwise there would be processes with no work to do. This can be seen in the division of the for-loops in Algorithm 3. This bound on the number of processes is increased with the method described in the following subsection.

5.4.3 Hybrid

Subsection 5.4.3 presents a hybrid implementation of both MPI and OpenMP. The hybrid approach is not limited to a single node like the OpenMP implementation, nor is it limited to $\min\{N_z, N_y\}$ processes as in the strictly MPI version. The hybrid method uses MPI to access different nodes on a cluster, then OpenMP to utilize the processors on each node. Algorithm 4 outlines the implementation of this method. MPI is used in the same way as before, but now an MPI process is considered as an entire node. On the forward and reverse transform steps OpenMP is used by FFTW multi-threading as described earlier in this section. The tridiagonal solver utilizes OpenMP by dividing the work in the x direction.

Figure 5.3: 3D Transfer of Data Between MPI Processes



Algorithm 4 Hybrid Parallel FFT Solver

1: Set up multi-threaded FFTW
2: Find $start_y$, $start_z$, end_y and end_z using the process rank
3: **for** $l = start_z, \ldots, end_z$ **do**
4: 2D forward DST of the RHS in x and y directions with multi-threading
5: **end for**
6: Transfer data via MPI to the appropriate processes
7: #pragma omp parallel for
8: **for** $j = start_y, \ldots, end_y$; $i = 1, \ldots, N_x$ **do**
9: Solve the tridiagonal system using LU decomposition
10: **end for**
11: Transfer data via MPI to the appropriate processes
12: **for** $l = start_z, \ldots, end_z$ **do**
13: 2D inverse DST of the solution in x and y directions with multi-threading
14: **end for**

The hybrid implementation demonstrates wall-time speed-up across multiple nodes in a large cluster. Its reduction in computation time is further from linear than the strictly MPI implementation. The lack of linear speed-up, however, is overshadowed by the ability of the hybrid approach to scale much farther by use of significantly more processes and memory. The largest successful test was run on an impressively large computational grid of 4096^3. The hybrid implementation completed this in just 27.52 seconds while the strictly MPI version required 445.80 seconds [16]. Further numerical tests with more detail are forthcoming.

5.5 Summary

Chapter 5 presented a highly parallel FFT solver and the required restrictions for such a solver. Parallel implementations in OpenMP, MPI and a hybrid structure were described in detail. Chapter 6 presents a novel generalized eigenvalue solver capable of solving the Helmholtz equation with Sommerfeld-like conditions on all boundaries.

6 Direct Generalized Eigenvalue Solver

To solve the inclusion problem with Sommerfeld-like conditions on all boundaries, a more versatile solver is needed. Chapter 6 presents such a solver that is direct and parallel. The boundary assumptions of the previous chapter are lifted, and Sommerfeld-like conditions are now considered on all boundaries. Three cases are considered starting with constant k. The cases where k varies only vertically and in all spatial directions will follow. Chapter 6 focuses on the three-dimensional case, but the two-dimensional case is nearly identical.

6.1 Uniform Medium

Assume k is constant, then the discretized system can be written as

$$\begin{aligned}
(A_a + \alpha_a B_a)\,\mathbf{u}_1 + (1+\beta_a)\,B_a\mathbf{u}_2 &= \mathbf{f}_1 \\
B_a\mathbf{u}_{l-1} + A_a\mathbf{u}_l + B_a\mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = 2,\ldots,N_z-1 \\
(1+\beta_a)\,B_a\mathbf{u}_{N_z-1} + (A_a + \alpha_a B_a)\,\mathbf{u}_{N_z} &= \mathbf{f}_{N_z}
\end{aligned}$$

where $A_a, B_a \in \mathbb{C}^{N_x\cdot N_y \times N_x\cdot N_y}$ are generated from the 27-point stencil given in Chapter 3. Also, α_a and β_a are the appropriate coefficients from (4.5), (4.7) or (4.8) depending on the order of approximation. For the development of this algorithm and those that follow, the matrix B_a is assumed nonsingular. Left multiplying by the inverse of B_a gives

$$\begin{aligned}
\left(B_a^{-1}A_a + \alpha_a I\right)\mathbf{u}_1 + (1+\beta_a)\,\mathbf{u}_2 &= B_a^{-1}\mathbf{f}_1 \\
\mathbf{u}_{l-1} + B_a^{-1}A_a\mathbf{u}_l + \mathbf{u}_{l+1} &= B_a^{-1}\mathbf{f}_l \qquad \text{for } l = 2,\ldots,N_z-1 \\
(1+\beta_a)\,\mathbf{u}_{N_z-1} + \left(B_a^{-1}A_a + \alpha_a I\right)\mathbf{u}_{N_z} &= B_a^{-1}\mathbf{f}_{N_z}.
\end{aligned}$$

To diagonalize this system suppose $A_a S = B_a S \Lambda_a$ where $\Lambda_a \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ is the diagonal matrix of eigenvalues of $B_a^{-1} A_a$ and $S \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ is the matrix of eigenvectors. It follows,

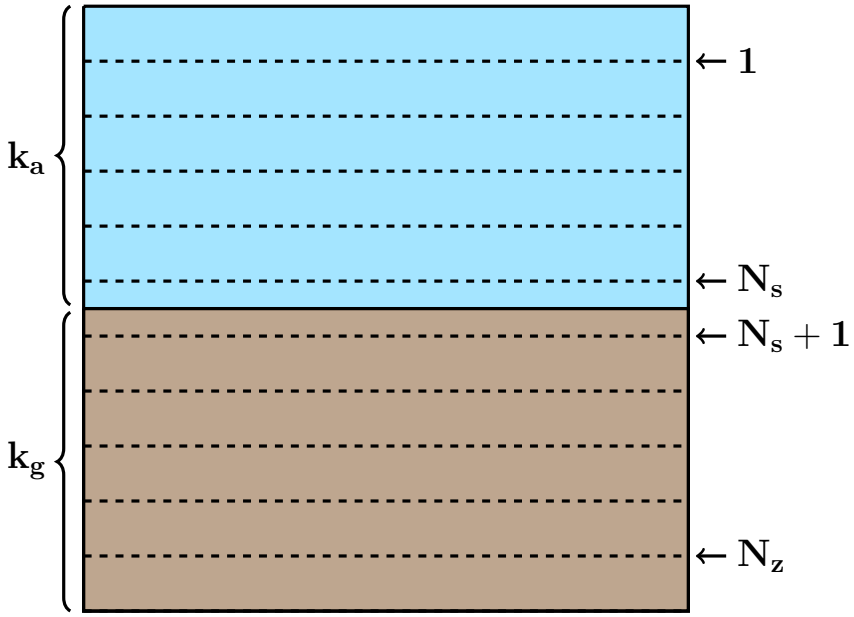
$$\begin{aligned}
(\Lambda_a + \alpha_a I)\, \mathbf{w}_1 + (1 + \beta_a)\, \mathbf{w}_2 &= \widehat{\mathbf{f}}_1 \\
\mathbf{w}_{l-1} + \Lambda_a \mathbf{w}_l + \mathbf{w}_{l+1} &= \widehat{\mathbf{f}}_l \qquad \text{for } l = 2, \ldots, N_z - 1 \\
(1 + \beta_a)\, \mathbf{w}_{N_z - 1} + (\Lambda_a + \alpha_a I)\, \mathbf{w}_{N_z} &= \widehat{\mathbf{f}}_{N_z}
\end{aligned}$$

where $\mathbf{w}_l = S^{-1} \mathbf{u}_l$ and $\widehat{\mathbf{f}}_l = S^{-1} B_a^{-1} \mathbf{f}_l$ for $l = 1, \ldots, N_z$. This diagonalization yields $N_x \cdot N_y$ independent linear systems. These systems can be solved using the LU decomposition of the generated tridiagonal matrix with $\mathcal{O}(N_z)$ computational complexity. Therefore, this solver is structured similar to the FFT Solver. The calculations required for the forward transform of the right-hand side, $\mathbf{f}$ for $l = 1, \ldots, N_z$, are independent with respect to the z direction and hence can be parallelized. The same is true for the reverse transform of $\mathbf{w}_l$. The calculations required to find $\mathbf{w}_l$ via LU decomposition are independent with respect to the x and y directions and can be parallelized.

6.2 Air and Soil Medium

The previous section's solver is highly parallel, but it is limited to constant k applications. Section 6.2 loosens this assumption and considers the case of k varying in the vertical direction. In three-dimensions this gives $k(x, y, z) = k(z)$. More specifically, k varies according to a domain with air and soil. Let k_a be the coefficient in the air and k_g be the coefficient in the soil, which are constant. Define N_s as the vertical index that is the last to contain the air coefficient k_a. Air and soil coefficients and the vertical index are illustrated in Figure 6.1.

Figure 6.1: Vertical Cross-Section of Air and Soil Domain



The resulting discretized system is written as

$$
\begin{aligned}
(A_a + \alpha_a B_a)\,\mathbf{u}_1 + (1+\beta_a)\,B_a\mathbf{u}_2 &= \mathbf{f}_1 \\
B_a\mathbf{u}_{l-1} + A_a\mathbf{u}_l + B_g\mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = 2,\ldots,N_s \\
B_a\mathbf{u}_{l-1} + A_g\mathbf{u}_l + B_g\mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = N_s+1,\ldots,N_z-1 \\
(1+\beta_g)\,B_g\mathbf{u}_{N_z-1} + (A_g + \alpha_g B_g)\,\mathbf{u}_{N_z} &= \mathbf{f}_{N_z}
\end{aligned}
$$

where $A_a, B_a, A_g, B_g \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ are generated from the 27-point stencil as before, but differ by the coefficients k_a and k_g. Similarly, α_a, β_a, α_g and β_g are the appropriate coefficients from the boundary conditions and only differ by k_a and k_g. Left multiplication of B_a^{-1} for $l = 1,\ldots,N_s$ and B_g^{-1} for $l = N_s+1,\ldots,N_z$ gives

$$
\left(B_a^{-1}A_a + \alpha_a I\right)\mathbf{u}_1 + (1+\beta_a)\,\mathbf{u}_2 = B_a^{-1}\mathbf{f}_1
$$

$$\mathbf{u}_{l-1} + B_a^{-1}A_a\mathbf{u}_l + B_a^{-1}B_g\mathbf{u}_{l+1} = B_a^{-1}\mathbf{f}_l \qquad \text{for } l = 2, \ldots, N_s$$

$$B_g^{-1}B_a\mathbf{u}_{l-1} + B_g^{-1}A_g\mathbf{u}_l + \mathbf{u}_{l+1} = B_g^{-1}\mathbf{f}_l \qquad \text{for } l = N_s+1, \ldots, N_z-1$$

$$(1+\beta_g)\,\mathbf{u}_{N_z-1} + \left(B_g^{-1}A_g + \alpha_g I\right)\mathbf{u}_{N_z} = B_g^{-1}\mathbf{f}_{N_z}.$$

Now suppose $A_aS = B_aS\Lambda_a$ where $\Lambda_a \in \mathbb{C}^{N_x\cdot N_y \times N_x\cdot N_y}$ is the diagonal matrix of eigenvalues and $S \in \mathbb{C}^{N_x\cdot N_y \times N_x\cdot N_y}$ is the matrix of eigenvectors. Similarly, suppose $A_gR = B_gR\Lambda_g$. Then, define

$$\mathbf{w}_l = S^{-1}\mathbf{u}_l \qquad \text{for } l = 1, \ldots, N_s$$

$$\mathbf{w}_l = R^{-1}\mathbf{u}_l \qquad \text{for } l = N_s+1, \ldots, N_z$$

$$\widehat{\mathbf{f}}_l = S^{-1}B_a^{-1}\mathbf{f}_l \qquad \text{for } l = 1, \ldots, N_s$$

$$\widehat{\mathbf{f}}_l = R^{-1}B_g^{-1}\mathbf{f}_l \qquad \text{for } l = N_s+1, \ldots, N_z.$$

Left multiplication of S^{-1} for $l = 1, \ldots, N_s$ and R^{-1} for $l = N_s+1, \ldots, N_z$ gives

$$(\Lambda_a + \alpha_a I)\,\mathbf{w}_1 + (1+\beta_a)\,\mathbf{w}_2 = \widehat{\mathbf{f}}_1$$

$$\mathbf{w}_{l-1} + \Lambda_a\mathbf{w}_l + \mathbf{w}_{l+1} = \widehat{\mathbf{f}}_l \qquad \text{for } l = 2, \ldots, N_s-1$$

$$\mathbf{w}_{N_s-1} + \Lambda_a\mathbf{w}_{N_s} + M\mathbf{w}_{N_s+1} = \widehat{\mathbf{f}}_{N_s}$$

$$M^{-1}\mathbf{w}_{N_s} + \Lambda_g\mathbf{w}_{N_s+1} + \mathbf{w}_{N_s+2} = \widehat{\mathbf{f}}_{N_s+1}$$

$$\mathbf{w}_{l-1} + \Lambda_g\mathbf{w}_l + \mathbf{w}_{l+1} = \widehat{\mathbf{f}}_l \qquad \text{for } l = N_s+2, \ldots, N_z-1$$

$$(1+\beta_g)\,\mathbf{w}_{N_z-1} + (\Lambda_g + \alpha_g I)\,\mathbf{w}_{N_z} = \widehat{\mathbf{f}}_{N_z}$$

where $M = S^{-1}B_a^{-1}B_gR$. To solve this transformed system, assume that

$$\mathbf{w}_l = D_{a,l}\mathbf{w}_{l+1} + P_{a,l} \qquad \text{for } l = 1, \ldots, N_s-1$$

$$\mathbf{w}_l = D_{g,l}\mathbf{w}_{l-1} + P_{g,l} \qquad \text{for } l = N_z, \ldots, N_s + 2.$$

Consider first $l = 1, \ldots, N_s$. When $l = 1$ the boundary conditions give the following

$$\widehat{\mathbf{f}}_1 = (\Lambda_a + \alpha_a I)\,\mathbf{w}_1 + (1 + \beta_a)\,\mathbf{w}_2$$

$$\mathbf{w}_1 = -(\Lambda_a + \alpha_a I)^{-1}(1 + \beta_a)\,\mathbf{w}_2 + (\Lambda_a + \alpha_a I)^{-1}\widehat{\mathbf{f}}_1$$

$$\mathbf{w}_1 = D_{a,1}\mathbf{w}_2 + P_{a,1}$$

where $D_{a,1} = -(1 + \beta_a)(\Lambda_a + \alpha_a I)^{-1}$ and $P_{a,1} = (\Lambda_a + \alpha_a I)^{-1}\widehat{\mathbf{f}}_1$. For the remaining layers in the air, that is $l = 2, \ldots, N_s - 1$, it follows

$$\widehat{\mathbf{f}}_l = \mathbf{w}_{l-1} + \Lambda_a \mathbf{w}_l + \mathbf{w}_{l+1}$$

$$\widehat{\mathbf{f}}_l = (\Lambda_a \ + D_{a,l-1})\,\mathbf{w}_l + P_{a,l-1} + \mathbf{w}_{l+1}$$

$$\mathbf{w}_l = -(\Lambda_a \ + D_{a,l-1})^{-1}\mathbf{w}_{l+1} + (\Lambda_a \ + D_{a,l-1})^{-1}\left(\widehat{\mathbf{f}}_l - P_{a,l-1}\right)$$

$$\mathbf{w}_l = D_{a,l}\mathbf{w}_{l+1} + P_{a,l}$$

where $D_{a,l} = -(\Lambda_a \ + D_{a,l-1})^{-1}$ and $P_{a,l} = (\Lambda_a \ + D_{a,l-1})^{-1}\left(\widehat{\mathbf{f}}_l - P_{a,l-1}\right)$. Similarly, in the opposite direction the lower boundary, $l = N_z$, gives

$$\widehat{\mathbf{f}}_{N_z} = (1 + \beta_g)\,\mathbf{w}_{N_z - 1} + (\Lambda_g + \alpha_g I)\,\mathbf{w}_{N_z}$$

$$\mathbf{w}_{N_z} = D_{g,N_z}\mathbf{w}_{N_z - 1} + P_{g,N_z}$$

where $D_{g,N_z} = -(1 + \beta_g)(\Lambda_g + \alpha_g I)^{-1}$ and $P_{g,N_z} = (\Lambda_g + \alpha_g I)^{-1}\widehat{\mathbf{f}}_{N_z}$. Finally, for the remaining layers in the soil, that is $l = N_z - 1, \ldots, N_s + 2$,

$$\widehat{\mathbf{f}}_l = \mathbf{w}_{l-1} + \Lambda_g \mathbf{w}_l + \mathbf{w}_{l+1}$$

$$\mathbf{w}_l = D_{g,l}\mathbf{w}_{l-1} + P_{g,l}$$

where $D_{g,l} = -\left(\Lambda_g + D_{g,l+1}\right)^{-1}$ and $P_{g,l} = \left(\Lambda_g + D_{g,l+1}\right)^{-1}\left(\widehat{\mathbf{f}}_l - P_{g,l+1}\right)$. The only two layers remaining are

$$\mathbf{w}_{N_s-1} + \Lambda_a \mathbf{w}_{N_s} + M\mathbf{w}_{N_s+1} = \widehat{\mathbf{f}}_{N_s}$$

$$M^{-1}\mathbf{w}_{N_s} + \Lambda_g \mathbf{w}_{N_s+1} + \mathbf{w}_{N_s+2} = \widehat{\mathbf{f}}_{N_s+1}.$$

A similar substitution with these layers gives

$$D_{a,N_s-1}\mathbf{w}_{N_s} + P_{a,N_s-1} + \Lambda_a \mathbf{w}_{N_s} + M\mathbf{w}_{N_s+1} = \widehat{\mathbf{f}}_{N_s}$$

$$M^{-1}\mathbf{w}_{N_s} + \Lambda_g \mathbf{w}_{N_s+1} + D_{a,N_s+2}\mathbf{w}_{N_s+1} + P_{a,N_s+2} = \widehat{\mathbf{f}}_{N_s+1}.$$

producing the following two by two block system

$$\begin{bmatrix} (D_{a,N_s-1} + \Lambda_a) & M \\ M^{-1} & (D_{g,N_s+2} + \Lambda_g) \end{bmatrix} \begin{bmatrix} \mathbf{w}_{N_s} \\ \mathbf{w}_{N_s+1} \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{f}}_{N_s} - P_{a,N_s-1} \\ \widehat{\mathbf{f}}_{N_s+1} - P_{a,N_s+2} \end{bmatrix}$$

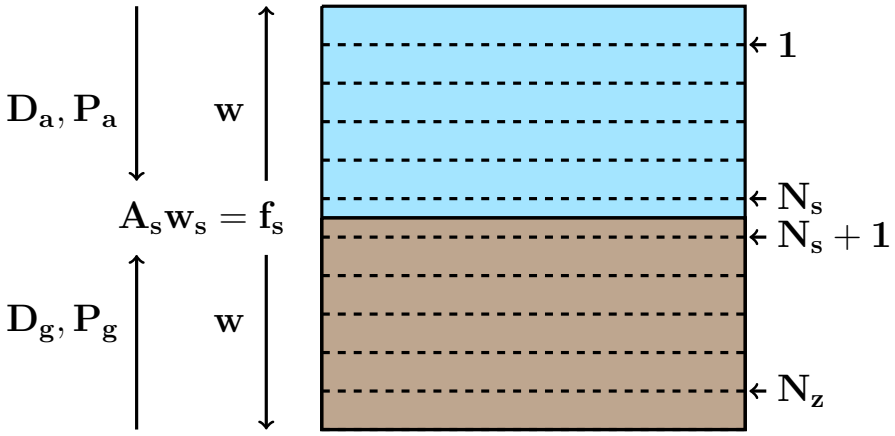
$$A_s \mathbf{w}_s = \mathbf{f}_s \tag{6.1}$$

where $A_s \in \mathbb{C}^{2N_x \cdot N_y \times 2N_x \cdot N_y}$ and $\mathbf{w}_s,\ \mathbf{f}_s \in \mathbb{C}^{2N_x \cdot N_y}$. The system (6.1) can be solved with LU decomposition giving $\mathbf{w}_{N_s}$ and $\mathbf{w}_{N_s+1}$. Note this is not the block LU factorization described for the FFT solver or the generalized eigenvalue solver for constant k. The remaining transformed solution, $\mathbf{w}_l$ for $l = 1, \ldots, N_s - 1$ and $l = N_s + 2, \ldots, N_z$, is found by computing

$$\mathbf{w}_l = D_{a,l}\mathbf{w}_{l+1} + P_{a,l} \quad \text{for } l = N_s - 1, \ldots, 1$$

$$\mathbf{w}_l = D_{g,l}\mathbf{w}_{l-1} + P_{g,l} \quad \text{for } l = N_s + 2, \ldots, N_z.$$

Figure 6.2: Generalized Eigenvalue Solver Design



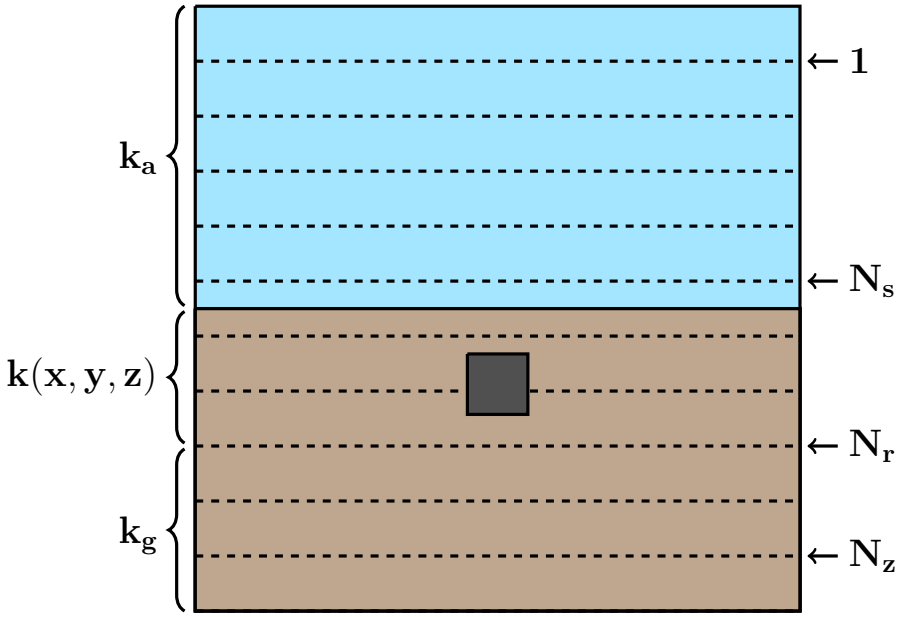
This process is illustrated in Figure 6.2. Note the necessary calculations are independent with respect to the vertical direction and hence can be computed in parallel with the exception of the solution of $A_s \mathbf{w}_s = \mathbf{f}_s$.

6.3 Air and Soil Medium with Subsurface Inclusion

In Section 6.3 the assumptions of Chapter 5 are completely lifted. That is, Sommerfeld-like conditions are now considered on all sides, and k can vary in all spatial directions. Specifically, k will vary as illustrated by Figure 6.3 with air, soil and a subsurface inclusion. Again, N_s is the vertical index that is the last to contain the air coefficient. Define N_r as the first vertical index after the inclusion. Then the system is written

$$
\begin{aligned}
(A_a + \alpha_a B_a)\,\mathbf{u}_1 + (1+\beta_a)\,B_a \mathbf{u}_2 &= \mathbf{f}_1 \\
B_a \mathbf{u}_{l-1} + A_a \mathbf{u}_l + B_a \mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = 2, \ldots, N_s - 1 \\
B_a \mathbf{u}_{N_s-1} + A_a \mathbf{u}_{N_s} + B_{N_s} \mathbf{u}_{N_s+1} &= \mathbf{f}_{N_s} \\
C_l \mathbf{u}_{l-1} + A_l \mathbf{u}_l + B_l \mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = N_s + 1, \ldots, N_r - 1
\end{aligned}
$$

Figure 6.3: Vertical Cross-Section of Air and Soil Domain with Subsurface Inclusion



$$C_{N_r}\mathbf{u}_{N_r-1} + A_g\mathbf{u}_{N_r} + B_g\mathbf{u}_{N_r+1} = \mathbf{f}_{N_r}$$

$$B_g\mathbf{u}_{l-1} + A_g\mathbf{u}_l + B_g\mathbf{u}_{l+1} = \mathbf{f}_l \qquad \text{for } l = N_r+1, \ldots, N_z-1$$

$$(1+\beta_g)\, B_g\mathbf{u}_{N_z-1} + (A_g + \alpha_g B_g)\, \mathbf{u}_{N_z} = \mathbf{f}_{N_z}$$

where A_a, B_a, A_g and B_g are as defined in the previous section. The matrices C_l, A_l, $B_l \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ are generated from the general 27-point stencil. Suppose $A_a S = B_a S \Lambda_a$ where $\Lambda_a \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ is the diagonal matrix of eigenvalues and $S \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ is the matrix of eigenvectors. Similarly, suppose $A_g R = B_g R \Lambda_g$. Define

$$\mathbf{w}_l = S^{-1}\mathbf{u}_l \qquad \text{for } l = 1, \ldots, N_s$$

$$\mathbf{w}_l = R^{-1}\mathbf{u}_l \qquad \text{for } l = N_r, \ldots, N_z$$

$$\widehat{\mathbf{f}}_l = S^{-1}B_a^{-1}\mathbf{f}_l \qquad \text{for } l = 1, \ldots, N_s$$

$$\widehat{\mathbf{f}}_l = R^{-1}B_g^{-1}\mathbf{f}_l \qquad \text{for } l = N_r, \ldots, N_z.$$

The system above can now be partially diagonalized. Multiplying this system by B_a^{-1} for the air layers and B_g^{-1} for the sub-inclusion layers gives

$$\begin{aligned}
(\Lambda_a + \alpha_a I)\,\mathbf{w}_1 + (1+\beta_a)\,\mathbf{w}_2 &= \widehat{\mathbf{f}}_1 \\
\mathbf{w}_{l-1} + \Lambda_a \mathbf{w}_l + \mathbf{w}_{l+1} &= \widehat{\mathbf{f}}_l \qquad \text{for } l = 2, \ldots, N_s - 1 \\
\mathbf{w}_{N_s-1} + \Lambda_a \mathbf{w}_{N_s} + M_a \mathbf{u}_{N_s+1} &= \widehat{\mathbf{f}}_{N_s} \\
C_l \mathbf{u}_{l-1} + A_l \mathbf{u}_l + B_l \mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = N_s + 1, \ldots, N_r - 1 \\
M_g \mathbf{u}_{N_r-1} + \Lambda_g \mathbf{w}_{N_r} + \mathbf{w}_{N_r+1} &= \widehat{\mathbf{f}}_{N_r} \\
\mathbf{w}_{l-1} + \Lambda_g \mathbf{w}_l + \mathbf{w}_{l+1} &= \widehat{\mathbf{f}}_l \qquad \text{for } l = N_r + 1, \ldots, N_z - 1 \\
(1+\beta_g)\,\mathbf{w}_{N_z-1} + (\Lambda_g + \alpha_g I)\,\mathbf{w}_{N_z} &= \widehat{\mathbf{f}}_{N_z}
\end{aligned}$$

where $M_a = S^{-1}B_a^{-1}B_{N_s}$ and $M_g = R^{-1}B_g^{-1}C_{N_r}$. All layers excluding those from $l = N_s, \ldots, N_r$ are fully diagonalized. To solve this partially transformed system assume that

$$\begin{aligned}
\mathbf{w}_l &= D_{a,l}\mathbf{w}_{l+1} + P_{a,l} \qquad \text{for } l = 1, \ldots, N_s - 1 \\
\mathbf{w}_l &= D_{g,l}\mathbf{w}_{l-1} + P_{g,l} \qquad \text{for } l = N_z, \ldots, N_r + 1.
\end{aligned}$$

Then the transformed solution is found the same way as in Section 6.2, but only for those layers in the air and soil beneath the inclusion. The matrices $D_{a,l}$ and $P_{a,l}$ are defined identically to those in Section 6.2. The only layers remaining are

$$\begin{aligned}
\mathbf{w}_{N_s-1} + \Lambda_a \mathbf{w}_{N_s} + M_a \mathbf{u}_{N_s+1} &= \widehat{\mathbf{f}}_{N_s} \\
C_l \mathbf{u}_{l-1} + A_l \mathbf{u}_l + B_l \mathbf{u}_{l+1} &= \mathbf{f}_l \qquad \text{for } l = N_s + 1, \ldots, N_r - 1
\end{aligned}$$

$$M_g \mathbf{u}_{N_r-1} + \Lambda_g \mathbf{w}_{N_r} + \mathbf{w}_{N_r+1} = \widehat{\mathbf{f}}_{N_r}.$$

These layers form the block tridiagonal system $A_s \mathbf{w}_s = \mathbf{f}_s$, given explicitly by

$$\begin{bmatrix} D_{a,N_s-1} + \Lambda_a & M_a & & & & & & \\ C_{N_s+1} S & A_{N_s+1} & B_{N_s+1} & & & & & \\ & C_{N_s+2} & A_{N_s+2} & B_{N_s+2} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & C_{N_r-2} & A_{N_r-2} & B_{N_r-2} & & \\ & & & & C_{N_r-1} & A_{N_r-1} & B_{N_r-1} R & \\ & & & & & M_g & D_{g,N_r+1} + \Lambda_g \end{bmatrix} \begin{bmatrix} \mathbf{w}_{N_s} \\ \mathbf{u}_{N_s+1} \\ \mathbf{u}_{N_s+2} \\ \mathbf{u}_{N_s+3} \\ \vdots \\ \mathbf{u}_{N_r-3} \\ \mathbf{u}_{N_r-2} \\ \mathbf{u}_{N_r-1} \\ \mathbf{w}_{N_r} \end{bmatrix}$$

$$= \begin{bmatrix} \widehat{\mathbf{f}}_{N_s} - P_{a,N_s-1} \\ \mathbf{f}_{N_s+1} \\ \mathbf{f}_{N_s+2} \\ \vdots \\ \mathbf{f}_{N_r-2} \\ \mathbf{f}_{N_r-1} \\ \widehat{\mathbf{f}}_{N_r} - P_{g,N_r+1} \end{bmatrix} \tag{6.2}$$

where $A_s \in \mathbb{C}^{N \cdot N_x \cdot N_y \times N \cdot N_x \cdot N_y}$ and $\mathbf{w}_s$, $\mathbf{f}_s \in \mathbb{C}^{N \cdot N_x \cdot N_y}$ with $N = N_r - N_s + 1$. Once the system (6.2) is solved via LU factorization, the remaining transformed solution, $\mathbf{w}_l$ for $l = 1, \ldots, N_s - 1$ and $l = N_r + 1, \ldots, N_z$, is found by calculating

$$\mathbf{w}_l = D_{a,l} \mathbf{w}_{l+1} + P_{a,l} \qquad \text{for } l = N_s - 1, \ldots, 1$$

$$\mathbf{w}_l = D_{g,l} \mathbf{w}_{l-1} + P_{g,l} \qquad \text{for } l = N_r + 1, \ldots, N_z.$$

This process is similar to that illustrated in Figure 6.2, except that A_s is larger, including up to the first layer past the inclusion. The necessary calculations are independent with respect to the vertical direction and can be computed in parallel except for the solution of $A_s \mathbf{w}_s = \mathbf{f}_s$.

6.4 Parallel Implementation

Section 6.4 provides the detailed parallelization of the developed direct generalized eigenvalue solvers presented in Chapter 6. The focus is the three-dimensional algorithm, but bears similarities to the two-dimensional algorithm. As in Section 5.4, the goal of implementing parallel algorithms is to observe a reduction in wall-time by dividing the computations as evenly as possible among the available processes.

First, consider the case given in Section 6.1, where k is constant and Sommerfeld-like conditions are considered on all boundaries. The structure of this solver is nearly identical to that of the FFT solver and hence can be parallelized in the same way with OpenMP and MPI as described in Section 5.4. The OpenMP parallelization of the process is shown in Algorithm 5. The transforms of the right-hand side $\mathbf{f}$ and transformed solution $\mathbf{w}$ are independent with respect to the vertical direction, the z direction in three-dimensions. Consequently, these calculations are divided among the processes with respect to the vertical layers. The solution of the tridiagonal system is dependent in the vertical direction and independent otherwise. Therefore, the computations can be divided with respect to the x and y directions. In a distributed memory environment, the appropriate data transfer needs to take place between the transform steps and the tridiagonal solver, as described in Subsection 5.4.2.

Algorithm 5 Parallel Generalized Eigenvalue Solver for Constant k

```
#pragma omp parallel for
for l = 1, ..., N_z do
    f̂_l = S^{-1} B_a^{-1} f_l
end for
#pragma omp parallel for collapse(2)
for j = 1, ..., N_y; i = 1, ..., N_x do
    Solve the tridiagonal system using LU decomposition
end for
#pragma omp parallel for
for l = 1, ..., N_z do
    u_l = S w_l
end for
```

The algorithms presented in sections 6.2 and 6.3 are similar. The key difference is the number of layers included in $A_s\mathbf{w}_s = \mathbf{f}_s$. As a result, only the parallelization for the solver in the case with an inclusion is described here. The case for a strictly air and soil medium is given by the Algorithm 6 by assuming $N_r = N_s + 1$.

The matrices $D_{a,l}$ and $D_{g,l}$ for $l \in \{1, \ldots, N_s - 1, N_r + 1, \ldots, N_z\}$ are computed as preliminary steps to the direct solver. Their computations are parallelizable as there is no dependency with respect to the vertical direction. Additionally, the matrices M_a, M_b and the LU factorization of A_s are computed preliminarily. The sequential solver is shown in Algorithm 6. However, it is easily parallelized with OpenMP by including the directive,"#pragma omp parallel for" prior to each loop. This OpenMP implementation is highly parallel and a powerful alternative to iterative methods. MPI can be used to parallelize the loops with data transfers before and after the solution of $A_s\mathbf{w}_s = \mathbf{f}$.

Algorithm 6 Parallel Generalized Eigenvalue Solver for Air and Soil with Inclusion

for $l = 1, \ldots, N_s$ **do**
  $\widehat{\mathbf{f}}_l = S^{-1} B_a^{-1} \mathbf{f}_l$
end for
for $l = N_r, \ldots, N_z$ **do**
  $\widehat{\mathbf{f}}_l = R^{-1} B_g^{-1} \mathbf{f}_l$
end for
$P_{a,1} = (\Lambda_a + \alpha_a I)^{-1} \widehat{\mathbf{f}}_1$ and $P_{g,N_z} = (\Lambda_g + \alpha_g I)^{-1} \widehat{\mathbf{f}}_{N_z}$
for $l = 2, \ldots, N_s - 1$ **do**
  $P_{a,l} = (\Lambda_a + D_{a,l-1})^{-1} \left(\widehat{\mathbf{f}}_l - P_{a,l-1} \right)$
end for
for $l = N_z - 1, \ldots, N_r + 1$ **do**
  $P_{g,l} = (\Lambda_g + D_{g,l-1})^{-1} \left(\widehat{\mathbf{f}}_l - P_{g,l+1} \right)$
end for
Solve the $A_s \mathbf{w}_s = \mathbf{f}_s$ with previously computed LU decomposition
for $l = N_s - 1, \ldots, 1$ **do**
  $\mathbf{w}_l = D_{a,l} \mathbf{w}_{l+1} + P_{a,l}$
end for
for $l = N_r + 1, \ldots, N_z$ **do**
  $\mathbf{w}_l = D_{g,l} \mathbf{w}_{l-1} + P_{g,l}$
end for
for $l = 1, \ldots, N_s$ **do**
  $\mathbf{u}_l = S \mathbf{w}_l$
end for
for $l = N_r, \ldots, N_z$ **do**
  $\mathbf{u}_l = R \mathbf{w}_l$
end for

6.5 Summary

Chapter 6 provided a detailed explanation of the generalized eigenvalue solver. Parallel algorithms for both constant and variable wave numbers were developed. Chapter 7 presents a detailed explanation of the three steps of the partial FFT solver.

7 Direct Partial FFT Solver

Chapter 7 presents an approach that combines the high parallel efficiency of the FFT solver and the versatility of the generalized eigenvalue solver. While both are independently impressive, the combination of the two into a novel partial FFT solver provides even faster solution time than the generalized eigenvalue solver. A similar partial FFT approach was introduced for the simplified case of constant k by Toivanen and Wolfmayr [10]. As before, the focus remains on the three-dimensional case, but the two-dimensional case can be inferred.

Consider the matrices $A, B \in \mathbb{C}^{N_x \cdot N_y \cdot N_z \times N_x \cdot N_y \cdot N_z}$. Let A be the matrix formed by the discretization described in Chapter 3 with Sommerfeld-like conditions on all boundaries. Also, let $\mathbf{u}, \mathbf{f} \in \mathbb{C}^{N_x \cdot N_y \cdot N_z}$ be such that $\mathbf{f}$ is the right-hand side formed by the discretization and $\mathbf{u}$ is the numerical solution. The problem at hand becomes $A\mathbf{u} = \mathbf{f}$. Let B be the matrix formed by the same discretization but restrict k to exclude the inclusion and only included Sommerfeld-like conditions on the top and bottom boundaries with respect to the vertical direction.

To present these matrices explicitly assume k is constant. If $n \in \mathbb{Z}$, let $I(n)$ and $\mathcal{I}(n)$ be defined as in Chapter 5. Then define T as the mapping such that $T(n)$ is the $n \times n$ matrix with ones at $(1,1)$ and (n,n) and zero otherwise. Similarly, let $\mathcal{T}(n)$ be the mapping such that $\mathcal{T}(n)$ is the $n \times n$ matrix with ones at $(1,2)$ and $(n, n-1)$ zero otherwise. For example:

$$T(4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{T}(4) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Consider the matrices $D_{x,a,b}, E_{x,a,b} \in \mathbb{C}^{N_x \times N_x}$ such that

$$D_{x,a,b} = aI(N_x) + b\mathcal{I}(N_x)$$

$$E_{x,b} = \alpha_x bT(N_x) + \beta_x b\mathcal{T}(N_x).$$

where $a,\ b \in \mathbb{C}$ and $\alpha_x, \beta_x \in \mathbb{C}$ are the boundary coefficients from Section 4.2 in the x direction. Then let $D_y, \mathcal{D}_y \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ be such that

$$D_y = I(N_y) \otimes D_{x,a_1,b_1} + \mathcal{I}(N_y) \otimes D_{x,c_1,d_1}$$

$$\mathcal{D}_y = I(N_y) \otimes D_{x,a_2,b_2} + \mathcal{I}(N_y) \otimes D_{x,c_2,d_2}$$

where $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2 \in \mathbb{C}$ are coefficients from Figure 5.2 at layer $l = 1$ and $\alpha_y, \beta_y \in \mathbb{C}$ are the boundary coefficients from Section 4.2 in the y direction. To include the Sommerfeld-like conditions in the x and y directions consider $D_{xy}, \mathcal{D}_{xy} \in \mathbb{C}^{N_x \cdot N_y \times N_x \cdot N_y}$ where

$$\begin{aligned} D_{xy} = {} & I(N_y) \otimes [D_{x,a_1,b_1} + E_{x,b_1}] + \mathcal{I}(N_y) \otimes [D_{x,c_1,d_1} + E_{x,d_1}] \\ & + \alpha_y T(N_y) \otimes [D_{x,c_1,d_1} + E_{x,d_1}] + \beta_y \mathcal{T}(N_y) \otimes [D_{x,c_1,d_1} + E_{x,d_1}] \\ \mathcal{D}_{xy} = {} & I(N_y) \otimes [D_{x,a_2,b_2} + E_{x,b_2}] + \mathcal{I}(N_y) \otimes [D_{x,c_2,d_2} + E_{x,d_2}] \\ & + \alpha_y T(N_y) \otimes [D_{x,c_2,d_2} + E_{x,d_2}] + \beta_y \mathcal{T}(N_y) \otimes [D_{x,c_2,d_2} + E_{x,d_2}]. \end{aligned}$$

Then the matrices A and B are given by

$$A = I(N_z) \otimes D_{xy} + \mathcal{I}(N_z) \otimes \mathcal{D}_{xy} + \alpha_z T(N_z) \otimes \mathcal{D}_{xy} + \beta_z \mathcal{T}(N_z) \otimes \mathcal{D}_{xy}$$

$$B = I(N_z) \otimes D_y + \mathcal{I}(N_z) \otimes \mathcal{D}_y + \alpha_z T(N_z) \otimes \mathcal{D}_y + \beta_z \mathcal{T}(N_z) \otimes \mathcal{D}_y$$

Therefore, in the case of constant k the matrix $C = B - A \in \mathbb{C}^{N_x \cdot N_y \cdot N_z \times N_x \cdot N_y \cdot N_z}$ is vastly sparse. Differing only at the positions necessary for the approximation of the Sommerfeld-like

boundary conditions. This difference also exists in the case of k varying in only the vertical direction. If k contains inclusions, C will also contain nonzero elements corresponding to the location of the inclusion. The sparse nature of C is the defining characteristic that will be taken advantage of in the partial FFT solver. Any matrix equation of the form $B\mathbf{x} = \mathbf{y}$ can be solved utilizing the FFT solver given in Chapter 5, while $A\mathbf{x} = \mathbf{y}$ requires the generalized eigenvalue solver from Chapter 6. Algorithm 7 outlines the general process of the direct partial FFT solver.

Algorithm 7 Partial FFT Solver

1: Solve $B\mathbf{x} = \mathbf{f}$ via FFT solver for only the necessary components of $\mathbf{x}$
2: Solve $A\mathbf{y} = C\mathbf{x}$ via generalized eigenvalue solver for only the necessary components of $\mathbf{y}$, where $\mathbf{y} = \mathbf{u} - \mathbf{x}$
3: Solve $B\mathbf{u} = \mathbf{f} + C(\mathbf{x} + \mathbf{y})$ for the entire numerical solution $\mathbf{u}$

The first step in Algorithm 7 could be solved with the FFT solver to find the full vector $\mathbf{x}$; however, only a small portion is needed as $C\mathbf{x}$ is vastly sparse. Only the elements of $\mathbf{x}$ that are necessary for the matrix vector multiplication, $C\mathbf{x}$, are transformed back using the DST. Step two is similar; the number of calculations necessary for the transform of the right-hand side, $C\mathbf{x}$, is drastically reduced, and only the necessary elements of $\mathbf{y}$ are calculated. Additionally, the final step has a reduced number of computations. Since the DST of $\mathbf{f}$ was already computed in step one, only the nonzero elements of $C(\mathbf{x} + \mathbf{y})$ need be transformed. This procedure solves the entire system but can utilize the FFT solver making it highly parallel. These steps are examined in detail in Section 7.1.

7.1 Computation Reduction

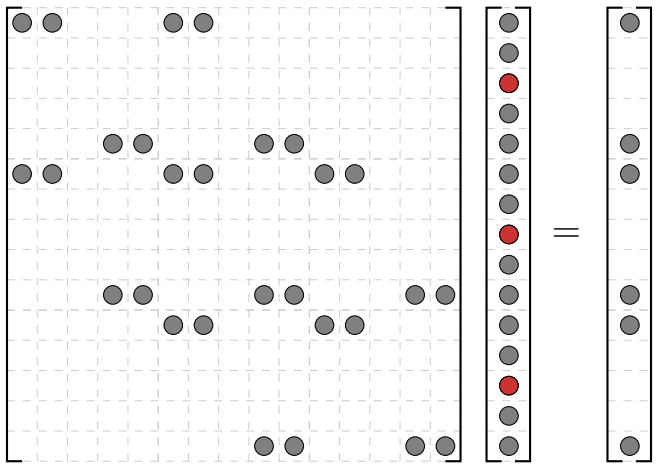
Section 7.1 provides detail in the reduction of computations achieved in the Partial FFT Solver. Each step in Algorithm 7 will be examined illustrating how the use of the FFT solver, in combination with, the generalized eigenvalue solver can reduce the total number of required computations after minor modification. The generalized eigenvalue solver has a computational complexity of $\mathcal{O}\left({N_x}^2 \cdot {N_y}^2 \cdot N_z\right)$ while this partial FFT solver is only $\mathcal{O}\left(N_x \cdot N_y \cdot N_z \log\left(N\right)\right)$ where $N = \max\{N_x, N_y\}$. These operation counts are based on the case for constant k.

The two solvers presented in Chapters 5 and 6 have a similar structure. The right-hand side must first be transformed, but the FFT solver transformation is by a DST calculated via FFT rather than multiplication by the matrices of eigenvectors as in the generalized eigenvalue solver. In the remainder of Section 7.1 both these right-hand side transformations will be referred to as the forward transform. After the forward transform, both the FFT and generalized eigenvalue solvers compute the transformed solution. The solution is then transformed back to the original space. This will be referred to as the reverse transform. The algorithms provided in Section 7.1 are reductions of the algorithms given in Chapters 5 and 6. No calculation dependencies are introduced, and therefore, the parallel implementations provided in Sections 5.4 and 6.4 still apply.

7.1.1 Step One

The first step in the partial FFT solver is to solve the auxiliary equation $B\mathbf{x} = \mathbf{f}$. However, looking ahead at step two shows that the entire vector $\mathbf{x}$ is not needed since $\mathbf{x}$ is multiplied by the vastly sparse matrix C. Only the elements of $\mathbf{x}$ necessary to compute $C\mathbf{x}$ are found.

Figure 7.1: 2D Matrix Sparsity Example



To illustrate this concept, consider the simple case in two-dimensions where $N_x = 5$ and $N_y = 3$. Then the matrix $C \in \mathbb{C}^{15\times 15}$ contains only $12N_y - 8 = 28$ nonzero elements with only four in the first and last rows of each N_x layer. Figure 7.1 illustrates this matrix and the product $C\mathbf{x}$. The elements shown in red in the vector $\mathbf{x}$ are not utilized in the multiplication and can be omitted. In application, N_x and N_y are significantly larger and the number of omitted elements is given by $(N_x - 4)N_y$.

The three-dimensional case further benefits from the sparsity of C. The matrix C contains only $(12N_y - 8)N_z - 2(12N_y - 8)$ nonzero elements and $\mathbf{x} \in \mathbb{C}^{N_x \cdot N_y \cdot N_z}$ needs only $4N_y \cdot N_z$ elements to conduct the multiplication $C\mathbf{x}$, that is $(N_x - 4)N_y \cdot N_z$ can be omitted. Therefore, in step one the necessary computations can be reduced in the FFT Solver as shown in Algorithm 8. Calculating the partial solution of $B\mathbf{x} = \mathbf{f}$ the entire transformed right-hand side, $\widehat{\mathbf{f}}$, must be computed. The transformed solution is then computed, but only the $4N_y \cdot N_z$ elements necessary to compute $C\mathbf{x}$ are transformed back via DST. This reduces the computational complexity of the reverse transform to only $\mathcal{O}\left(N_y \cdot N_z \log\left(N\right)\right)$ from

$\mathcal{O}\left(N_x \cdot N_y \cdot N_z \log\left(N\right)\right)$ where $N = \max\{N_x, N_y\}$ as the DST is calculated via FFT. These calculations are still independent with respect to the vertical direction and can therefore be parallelized.

Algorithm 8 FFT Solver Reverse Transform Reduced

```
for l = 1, ..., N_z do
    2D forward DST of the RHS in x and y directions
end for
for j = 1, ..., N_y; i = 1, ..., N_x do
    Solve the tridiagonal system using LU decomposition
end for
for l = 1, ..., N_z do
    1D DST of the solution in the x direction, but only for the first and last two rows
    1D DST of the solution in the y direction
end for
```

7.1.2 Step Two

The second step in the Partial FFT Solver is to solve the equation $A\mathbf{y} = C\mathbf{x}$ where $\mathbf{y} = \mathbf{u} - \mathbf{x}$. The solving is accomplished with a modified generalized eigenvalue solver that takes advantage of the sparsity in the right-hand side, $C\mathbf{x}$. Due to sparsity fewer computations are required to transform the right-hand side verses the original generalized eigenvalue solver. Additionally, looking ahead at step three, the solution of step two, $\mathbf{y}$, will be multiplied by C. The resulting reverse transform has the same reduction as in step one.

Returning to the two-dimensional example with $N_x = 5$ and $N_y = 3$, consider the vector $C\mathbf{x}$ in Figure 7.1. Because the vector only contains nonzero elements at the first and last row in each N_x block, the transform of the right-hand side in the generalized eigenvalue solver, $S^{-1}B_a^{-1}\left(C\mathbf{x}\right)_j$ for $j = 1, \ldots, N_y$, is reduced from ${N_x}^2 \cdot N_y$ multiplications to only $2N_x \cdot N_y$. Similar to step one, the full solution, $\mathbf{y}$, is not needed. Hence, the reverse transformation to $\mathbf{y}$ is reduced by $(N_x - 4)N_y$ elements.

Algorithm 9 Reduced Generalized Eigenvalue Solver

for $l = 1, \ldots, N_z$; $j = 1, \ldots, N_y$; $i = 1, \ldots, N_x$ **do**
  $\widehat{\mathbf{f}}_{i,j,l} = (B_a S)^{-1}_{i,1} (C\mathbf{x})_{1,j,l} + (B_a S)^{-1}_{i,N_x} (C\mathbf{x})_{N_x,j,l}$
end for
for $j = 1, \ldots, N_y$; $i = 1, \ldots, N_x$ **do**
  Solve the tridiagonal system using LU decomposition
end for
for $l = 1, \ldots, N_z$; $j = 1, \ldots, N_y$; **do**
  $\mathbf{y}_{1,j,l} = S_{1,1:N_x \cdot N_y} \cdot \mathbf{w}_l$
  $\mathbf{y}_{2,j,l} = S_{2,1:N_x \cdot N_y} \cdot \mathbf{w}_l$
  $\mathbf{y}_{N_x-1,j,l} = S_{N_x-1,1:N_x \cdot N_y} \cdot \mathbf{w}_l$
  $\mathbf{y}_{N_x,j,l} = S_{N_x,1:N_x \cdot N_y} \cdot \mathbf{w}_l$
end for

In the three-dimensional case the computational complexity of the forward transform is reduced to $\mathcal{O}(N_x \cdot N_y \cdot N_z)$ from $\mathcal{O}\left({N_x}^2 \cdot {N_y}^2 \cdot N_z\right)$. The reverse transformation to $\mathbf{y}$ is reduced as in step one. In the full generalized eigenvalue solver, the transforms are computed by matrix vector multiplication for each vertical layer requiring ${N_x}^2 \cdot {N_y}^2 \cdot N_z$ scalar multiplications. However, in this case the work has been reduced to four dot-products for each vertical layer requiring $4N_x \cdot N_y \cdot N_z$ scalar multiplications. Algorithm 9 shows the reduced generalized eigenvalue solver for constant k. The algorithm uses vector notation, $\mathbf{w}_l$, introduced in Chapter 5 by (5.1). The cases for k varying only vertically or with an inclusion take advantage of the sparsity in the same manner.

7.1.3 Step Three

The third and final step of the partial solver is finding $\mathbf{u}$ in $B\mathbf{u} = \mathbf{f} + C(\mathbf{x}+\mathbf{y})$ via FFT solver, which modifies the FFT solver differently than step one. Here the reduction is achieved in the forward transform. The two-dimensional DST required in the forward step is reduced in the x directions. There are only two nonzero elements in each N_x block. Therefore, the number of operations in the forward transform is reduced to $\mathcal{O}(N_y \cdot N_z \log(N_y))$ from

$\mathcal{O}\left(N_z \cdot N_x \cdot N_y \log(N)\right)$. This reduction is shown in Algorithm 10. In the two-dimensional example with $N_x = 5$ and $N_y = 3$, only the DST in the x direction is needed and hence the multiplications required are only $2N_y = 6$ for the forward transform.

Algorithm 10 FFT Solver Forward Transform Reduced

1: **for** $l = 1, \ldots, N_z$ **do**
2: $\quad \widehat{\mathbf{r}}_l = \widehat{\mathbf{f}}_l + DST_y(DST_x(C(\mathbf{x} + \mathbf{y})_l))$
3: **end for**
4: **for** $j = 1, \ldots, N_y$; $i = 1, \ldots, N_x$ **do**
5: $\quad$ Solve the tridiagonal system using LU decomposition with $\widehat{\mathbf{r}}$ as the transformed RHS
6: **end for**
7: **for** $l = 1, \ldots, N_z$ **do**
8: $\quad$ 2D inverse DST of the solution in x and y directions
9: **end for**

7.2 Summary

Chapter 7 provided a detailed explanation of the three steps of the partial FFT solver. Steps one, two and three demonstrated the reduction in computations gained by combining the generalized eigenvalue and FFT solvers. Chapter 8 presents detailed examinations of the accuracy and parallel performance of the implementations of all three solvers considered by the research.

8 Numerical Results

Chapter 8 presents the results of numerical experiments implemented in the programming language C to demonstrate the power and versatility of the given parallel numerical methods. Some results are from runs on Lawrence Berkeley National Laboratory's Cori cluster. The tests utilize Haswell nodes on Cori, which contain Intel Xeon 2.3 GHz processors with 32 physical cores and 128 GB 2133 MHz DDR4 memory. Additional numerical experiments were conducted on an iMac desktop, MacBook Pro laptop or server with a Xeon X5690 3.47 GHz processor and 144 GB of memory. The MackBook Pro contains an Intel Quad-Core i7 2.7 GHz processor and 16 GB 2133 MHz LPDDR3 memory. The iMac contains an Intel Quad-Core i7 2.93 GHz processor and 16 GB 2133 MHz LPDDR3 memory. Results are divided into two sections, one for the FFT solver and the other for the generalized eigenvalue and partial FFT solvers. All times provided in Chapter 8 are wall-times measured in seconds.

8.1 FFT Solver

Building on Chapter 5's presentation of three different parallelization strategies for the FFT solver, Section 8.1 compares the performance of the three techniques for test problems with Dirichlet boundary conditions. The Helmholtz and convection-diffusion equations are considered in three-dimensions. A more detailed examination of this solver is provided in the author's previous publication [16].

8.1.1 Helmholtz

OpenMP, MPI and hybrid implementations of the developed direct FFT solver were used to obtain approximate solutions of the problem (1.1,1.2) with Dirichlet boundary conditions.

In the following numerical experiments, the coefficient k is considered to be constant or vary only in the vertical direction. For either case k is defined by

$$k(z) = a - b\sin(cz) \tag{8.1}$$

with $a > b \geq 0$. Note that k is constant when $b = 0$. To test the accuracy, consider the analytic solution

$$u(\mathbf{x}) = \sin(\beta x)\sin(\gamma y)\, e^{-k(z)/c}$$

where $\beta^2 + \gamma^2 = a^2 + b^2$. This gives the right-hand side

$$f(\mathbf{x}) = -b(2a + c)\sin(cz)e^{-k(z)/c}\sin(\beta x)\sin(\gamma y).$$

The domain under consideration is $\Omega = [0, \pi] \times [0, \pi] \times [0, \pi]$. Turkel considered this test problem in [5], with results from runs on a Supermicro cluster consisting of 12 nodes. Each node contained two Intel Xeon E5520 2.27 GHz quad processors that shared 8 GB of RAM.

Table 8.1 presents a comparison of two solvers on the problem above utilizing variable k with $a = 10$, $b = 9$, $c = 10$, $\gamma = 9$ and $\beta = 10$. The first two rows of Table 8.1 show results from the iterative solver used in [5], while the remaining results are from the second-, fourth- and sixth-order implementations of the FFT solver provided in Chapter 5. Define the relative L_2 error as $||\mathbf{u} - \mathbf{u}_a||_2/||\mathbf{u}||_2$ where $\mathbf{u}$ is the numerical solution and $\mathbf{u}_a$ is the analytic solution at the same grid points. The tolerance for the iterative method was set at .001 for the L_2 error. The grid sizes for the direct FFT solver runs were chosen to produce an L_2 error below this tolerance. The iterative solver was run on the Supermicro cluster while the direct FFT solver was run on both the Xeon X5690 server and iMac described earlier.

Table 8.1: Comparison of Direct and Iterative Solvers

Machine	Scheme	Type	N	Iterations	Seconds
Supermicro	2nd	iteration	333	1970	703
Supermicro	6th	iteration	45	350	1.01
X5690	2nd	direct	353	NA	15.18
X5690	4th	direct	62	NA	0.078
X5690	6th	direct	50	NA	0.055
iMac	2nd	direct	353	NA	19.8
iMac	4th	direct	62	NA	0.097
iMac	6th	direct	50	NA	0.08

In this case $N = N_x = N_y = N_z$.

Table 8.1 shows the direct FFT solver on the X5690 and iMac i7 was approximately 46 and 36 times faster than the iterative solver respectively in the second-order case. With the sixth-order scheme, the direct FFT solver was approximately 18 and 13 times faster on the X5690 and iMac i7 than the iterative solver, respectively. The direct solver results are from sequential runs where only one process was utilized. These results demonstrate the immense advantage the direct FFT solver has in this problem even without parallel computations. The same accuracy, or resolution, is achieved in a fraction of the time.

The sequential and parallel implementations of the FFT solver give consistent results on all considered grids. Table 8.2 shows the convergence of the second-, fourth- and sixth-order implementations, respectively. These tests utilize variable k with $a = 10$, $b = 9$, $c = 10$, $\gamma = 9$ and $\beta = 10$. The table shows the maximum error between the numerical solution, $\mathbf{u}$, and the analytic solution $\mathbf{u}_a$. If the grid size is doubled in each direction the error decreases by roughly four, 16 and 64 times for second-, fourth- and sixth-order, respectively. This confirms the accuracy of the numerical methods. These values for the error are the same

Table 8.2: FFT Solver Convergence

	Second-Order		Fourth-Order		Sixth-Order	
Grid	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$
125^3	5.757e-03	4.727e-13	3.449e-05	3.630e-13	2.188e-06	3.158e-13
250^3	1.485e-03	2.585e-12	2.178e-06	1.986e-12	3.494e-08	2.054e-12
500^3	3.745e-04	6.588e-12	1.372e-07	5.083e-12	5.521e-10	5.215e-12

The numerical and analytic solutions are given by $\mathbf{u}$ and $\mathbf{u}_a$ respectively.

regardless of the number of processes used. In addition, Table 8.2 shows that the residual for each order's implementation is approximately zero with respect to machine precision.

Next the parallel performance of the direct FFT solver is examined. Figures 8.1 and 8.2 show the scaled wall-times required to numerically solve the system when k is constant and $k(\mathbf{x}) = k(z)$ respectively. In the case of constant k the values $a = 20$, $b = 0$, $c = 10$, $\gamma = 16$ and $\beta = 12$ are chosen in (8.1). For variable k the same values are used as in the previous experiment, that is $a = 10$, $b = 9$, $c = 10$, $\gamma = 9$ and $\beta = 10$. The results given in these figures were recorded from runs on Cori. They show the comparison of the strictly OpenMP implementation verses the strictly MPI implementation on a single node. Near optimal speed-up is observed, that is the doubling of computational processes cuts the solution time by nearly half.

Tables 8.3 and 8.4 show parallelization results of the FFT solver's hybrid implementation. These results require the use of a cluster and hence were recorded from runs on Cori. To efficiently access as many computational processes as possible, the hybrid implementation utilizes MPI to access different nodes in the cluster while OpenMP give access to each processor on the node. The efficiency of this hybrid version is displayed in Table 8.3. The number of processors utilized on each node by OpenMP is shown in the columns while the

rows represent the number of nodes. With this implementation the solution time can be reduced to just over half a second on a grid of 512^3.

Figure 8.1: Constant k FFT Solver OpenMP vs MPI on 512^3

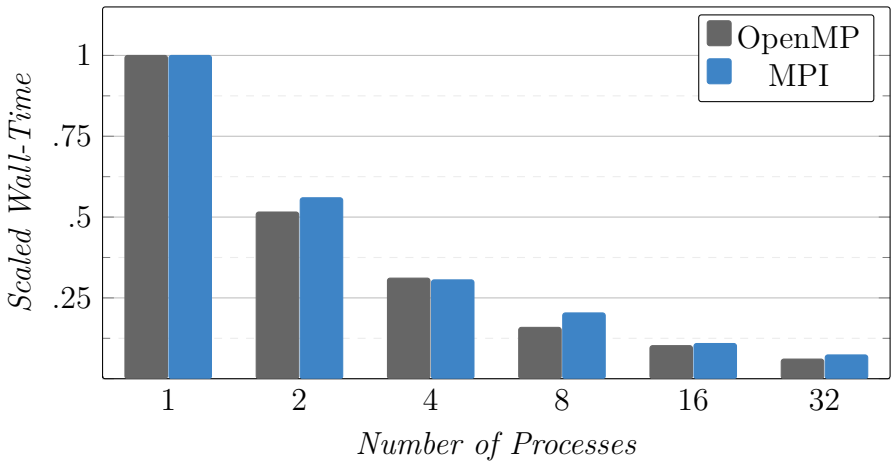
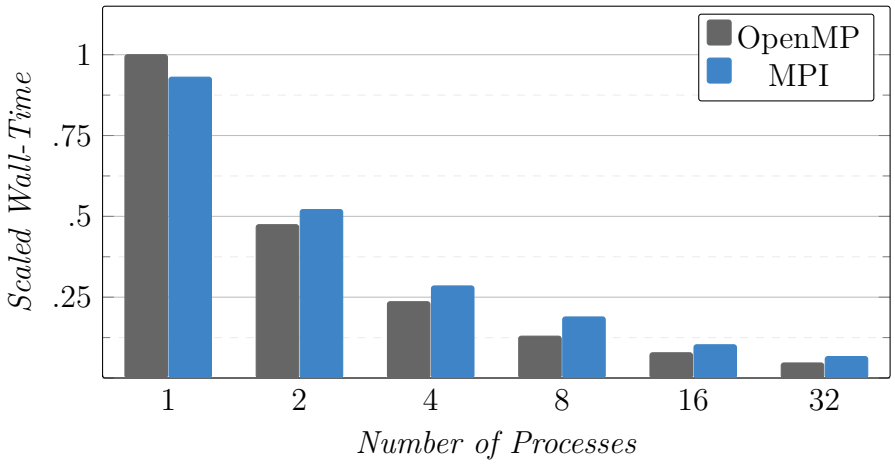


Figure 8.2: Variable k FFT Solver OpenMP vs MPI on 512^3



The true power of the hybrid implementation is shown in Table 8.4. Results of the tests run with very large grid sizes, up to 4096^3, creating an extremely high-resolution solution are presented. The numerical solution on this large grid was achieved in only 27.522 seconds.

Table 8.3: FFT Solver Hybrid on 512^3

Nodes	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads	32 Threads
1	39.61	21.73	12.93	8.83	6.84	5.62
2	19.87	10.94	6.48	4.56	3.48	2.92
4	9.99	5.66	3.48	2.52	2.10	1.82
8	5.27	3.11	1.99	1.55	1.37	1.27
16	2.49	1.42	0.85	0.64	0.58	0.58
32	1.85	1.33	1.09	0.93	0.76	0.80

Table 8.4: FFT Solver Hybrid on Large Grids

Grid	Nodes	Processors	Seconds
512^3	1	32	7.794
1024^3	4	128	16.911
2048^3	32	1024	19.418
4096^3	256	8192	27.522

This reduction in wall-time was accomplished by utilizing a total of 8192 processors in a very efficient manner. Finding such a high-resolution solution is only possible through parallel computing.

8.1.2 Convection-Diffusion

To demonstrate the versatility of the FFT solver, an example using convection-diffusion is presented. The discretization for the general convection-diffusion equation given in Section 3.4 does not fit the pattern necessary to be solved by the FFT solver, however it does in the common case of dominate vertical convection. This allows the convection coefficients to be defined as $a_1 = 0$, $a_2 = 0$ and $a_3 \in \mathbb{R}$, which reveals the necessary stencil pattern. Only the fourth-order scheme and OpenMP are considered. The test problem under consideration

Table 8.5: FFT Solver Convergence for Convection-Diffusion

Grid	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$
64^3	3.261e-03	1.544e-15
128^3	2.058e-04	4.610e-15
256^3	1.294e-05	9.900e-15
512^3	8.198e-07	3.260e-14

The numerical and analytic solutions are given by $\mathbf{u}$ and $\mathbf{u}_a$ respectively.

can be presented as

$$\Delta u - a_3 \frac{\partial}{\partial z} u = 0$$

with $a_3 = 100$. The rectangular domain is defined by

$$\Omega = \left[0, \sqrt{2}\right] \times \left[0, \sqrt{2}\right] \times [0, 1]$$

with Dirichlet boundary conditions

$$u(x, y, 0) = \sin\left(\frac{\pi x}{\sqrt{2}}\right) \sin\left(\frac{\pi y}{\sqrt{2}}\right)$$

$$u(x, y, 1) = 2\sin\left(\frac{\pi x}{\sqrt{2}}\right) \sin\left(\frac{\pi y}{\sqrt{2}}\right)$$

$$u(0, y, z) = u\left(\sqrt{2}, y, z\right) = u\left(x, 0, z\right) = u\left(x, \sqrt{2}, z\right) = 0.$$

The analytic solution is given by

$$u(\mathbf{x}) = e^{a_3 z/2} \sin\left(\frac{\pi x}{\sqrt{2}}\right) \sin\left(\frac{\pi y}{\sqrt{2}}\right) \frac{\sinh(\sigma(1-z)) + 2e^{-a_3/2}\sinh(\sigma)}{\sinh(\sigma)}$$

where $\sigma = \sqrt{\pi^2 + a_3^2/4}$.

The fourth-order convergence of the numerical solution to the analytic solution on a

sequence of grids is presented in Table 8.5. The table shows the maximum error between the numerical solution, $\mathbf{u}$, and the analytic solution $\mathbf{u}_a$. The error decreases by approximately 16 times as the number of grid points doubles in each spatial direction, and the consistency is true regardless of the number of processes used. The convergence of the error confirms the accuracy of the numerical method. In addition, Table 8.5 shows the residual is approximately zero with respect to machine precision.

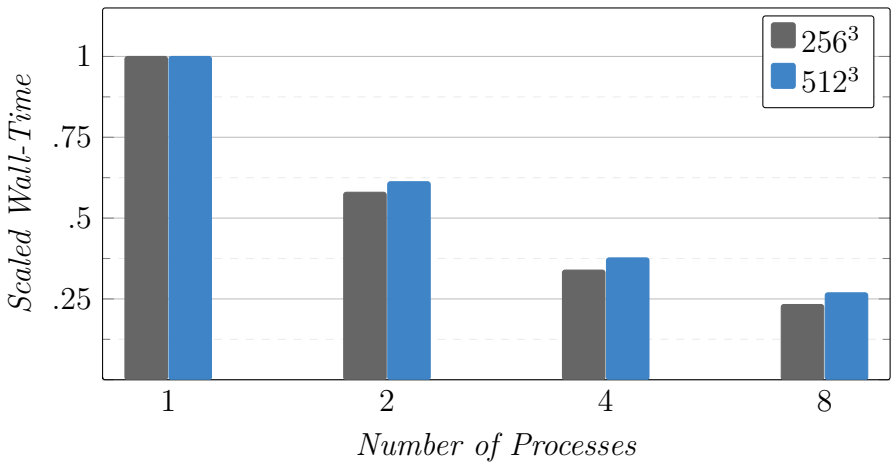
Table 8.6 provides the results of computation time in seconds on a single Cori node. The number of OpenMP processes in the test varies from one to eight. The table shows the wall-time of the direct solver for the grid sizes 256^3 and 512^3. The parallel algorithm gives approximately a four times speed-up from one process, run sequentially, to eight processes. Figure 8.3 shows the parallel computation times from Table 8.6 scaled by the sequential time. The decrease in computation time is approximately the same of the FFT solver for the Helmholtz tests provided in Table 8.5. Section 8.1 results have shown that not only is the FFT solver highly parallel and efficient, but versatile in its application.

Table 8.6: FFT Solver OpenMP for Convection-Diffusion

Processes	256^3 Grid	512^3 Grid
1	7.859 s	40.088 s
2	4.558 s	24.553 s
4	2.666 s	15.110 s
8	1.829 s	10.878 s

The wall-times are given in seconds.

Figure 8.3: Convection-Diffusion FFT Solver OpenMP



8.2 Generalized Eigenvalue and Partial FFT Solvers

The tests presented in Section 8.2 find numerical solutions of the problem (1.1,1.2) with Sommerfeld-like conditions on all boundaries. This problem requires the use of the generalized eigenvalue solver or the partial FFT solver. First, test problems with constant k demonstrate the accuracy of the numerical schemes. The findings are compared to results given by Toivanen and Wolfmayr in [10]. The tests are then expanded to include k varying in the vertical direction as shown in Figure 6.1 with realistic values for air and soil [13]. Finally, the accuracy of the algorithms' implementations are given on test problems with subsurface inclusions.

8.2.1 Constant k

Subsection 8.2.1 provides the OpenMP implementations of the developed direct generalized eigenvalue and partial FFT solvers are used to obtain approximate solutions of the problem (1.1,1.2) with Sommerfeld-like boundary conditions. The wave number k is assumed to be

Table 8.7: Generalized Eigenvalue Solver Convergence

	Second-Order		Fourth-Order		Sixth-Order	
Grid	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$
33^2	1.63e+01	3.07e-12	1.18e+00	7.17e-11	3.36e-02	6.88e-13
65^2	3.58e+00	3.37e-12	7.07e-02	1.18e-10	4.38e-04	1.41e-12
129^2	8.47e-01	5.49e-12	4.41e-03	3.44e-10	6.57e-06	4.85e-12
257^2	2.09e-01	2.79e-11	2.76e-04	6.87e-10	1.02e-07	1.69e-11
513^2	5.22e-02	9.06e-11	1.72e-05	2.34e-09	1.63e-09	3.89e-11

The numerical and analytic solutions are given by $\mathbf{u}$ and $\mathbf{u}_a$ respectively.

constant. The two-dimensional case is investigated.

The first test's purpose is to examine the accuracy of the numerical schemes. Both the generalized eigenvalue and partial FFT solvers are tested on the same problem. Let $k = \sqrt{439.2}$. Consider the function

$$\phi(x) = \exp\left(ik(x+1)\right) + \exp\left(-ik(x-1)\right) - 2.$$

In the two-dimensional case the analytic solution on the domain $\Omega = [-1,1]\times[-1,1]$ is given by $u(x,y) = \phi(x)\phi(y)$. The right-hand side follows

$$f(\mathbf{x}) = -k^2\left[\phi(x)\phi(y) - 2\phi(x) - 2\phi(y)\right]. \tag{8.2}$$

Tables 8.7 and 8.8 show the maximum error between the numerical solution, $\mathbf{u}$, and the analytic solution $\mathbf{u}_a$ in the generalized eigenvalue and the partial FFT solvers, respectively. The error decreases by roughly four, 16 and 64 times for second-, fourth- and sixth-order, respectively, confirming the accuracy of these solver's implementations. The maximum errors are the same regardless of the number of processes used. In addition, the tables show

Table 8.8: Partial FFT Solver Convergence

	Second-Order		Fourth-Order		Sixth-Order	
Grid	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$	$\|\|\mathbf{u}-\mathbf{u}_a\|\|_\infty$	$\|\|A\mathbf{u}-\mathbf{f}\|\|_2$
33^2	1.63e+01	7.29e-11	1.18e+00	4.11e-11	3.36e-02	2.77e-12
65^2	3.58e+00	9.87e-12	7.07e-02	1.41e-11	4.38e-04	6.75e-12
129^2	8.47e-01	1.20e-11	4.41e-03	1.04e-09	6.57e-06	3.23e-11
257^2	2.09e-01	4.24e-10	2.76e-04	1.53e-09	1.02e-07	5.98e-11
513^2	5.22e-02	6.88e-10	1.72e-05	5.34e-09	1.63e-09	3.59e-09

The numerical and analytic solutions are given by $\mathbf{u}$ and $\mathbf{u}_a$ respectively.

the residual for each order's implementation is approximately zero with respect to machine precision.

The next test assumes that $k = 2\pi$ on the domain $\Omega = [0, 1] \times [0, 1]$. The parallel results of this test are shown in Tables 8.9 and 8.10. The choice of the grid size in this case is to compare the results provided by Toivanen and Wolfmayr in [10]. Toivanen and Wolfmayr's solver, namely fast solver, is implemented via Matlab utilizing a similar method to the partial FFT solver described in Chapter 7. Unlike the partial FFT solver, the fast solver's results are restricted to constant k. Results for the fast solver were found by runs on a laptop with an Intel(R) Core(TM) i5-6267U CPU 2.90 GHz processor and 16 GB 2133 MHz LPDDR3 memory. All other computation times were found by runs on the MackBook Pro described in the beginning of Chapter 8 with four OpenMP processes.

Table 8.9: Solver Timing Comparison on Uniform Grids

$N_x = N_y$	65	129	257	513	1025	2049	4097
Matlab's backslash	0.028	0.064	0.334	1.611	8.198	65.490	1421.4
Fast solver	0.01	0.02	0.06	0.23	1.05	4.58	-
Initialization	0.012	0.066	0.266	1.308	5.340	28.407	211.74
Generalize eigenvalue solver	0.001	0.003	0.014	0.080	0.315	1.850	13.749
Partial FFT solver	0.001	0.004	0.020	0.088	0.251	1.256	5.696

The wall-times are given in seconds.

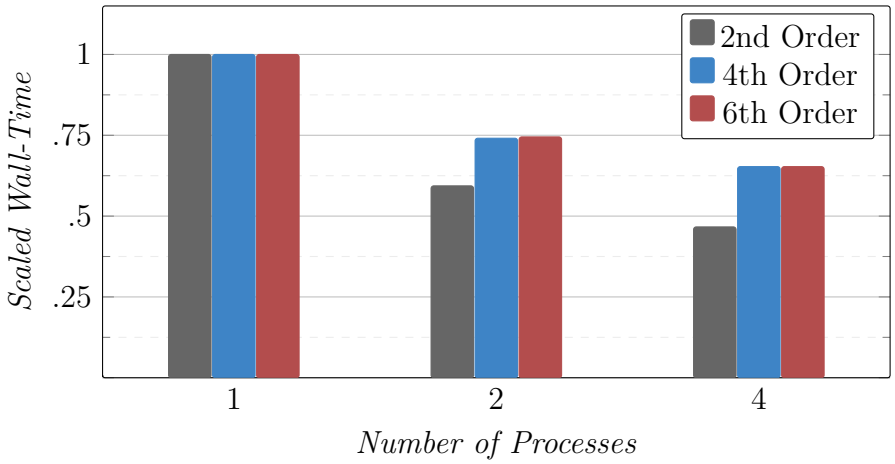
Table 8.10: Solver Timing Comparison on Nonuniform Grids

N_x	65	65	2049	2049	2049	4097	4097
N_y	65	2049	65	2049	4097	2049	4097
Matlab's backslash	0.028	0.655	0.640	65.490	506.62	436.67	1421.4
Fast solver	0.01	0.10	0.12	4.58	-	-	-
Initialization	0.012	0.017	28.899	28.407	28.962	215.57	211.74
Generalize eigenvalue solver	0.001	0.007	0.620	1.850	3.371	7.489	13.749
Partial FFT solver	0.001	0.015	0.571	1.256	2.446	3.780	5.697

The wall-times are given in seconds.

Table 8.9 provides a comparison with a uniform grid, while Table 8.10 shows experiments with nonuniform grid. These comparisons are between the second-order partial FFT solver in C and the second-order fast solver in Matlab. Toivanen and Wolfmayr state that a C implementation of the fast solver would be more efficient with respect to solution time; however, such an implementation's results were not provided in [10]. The initialization rows in Tables 8.9 and 8.10 represent the time required to calculate the eigenvalues, eigenvectors and LU factorizations needed in the partial FFT solver. The partial FFT solver bears no advantage over the generalized eigenvalue solver until relatively larger grid sizes, an expected

Figure 8.4: Constant k Partial FFT Solver OpenMP on 1025^2

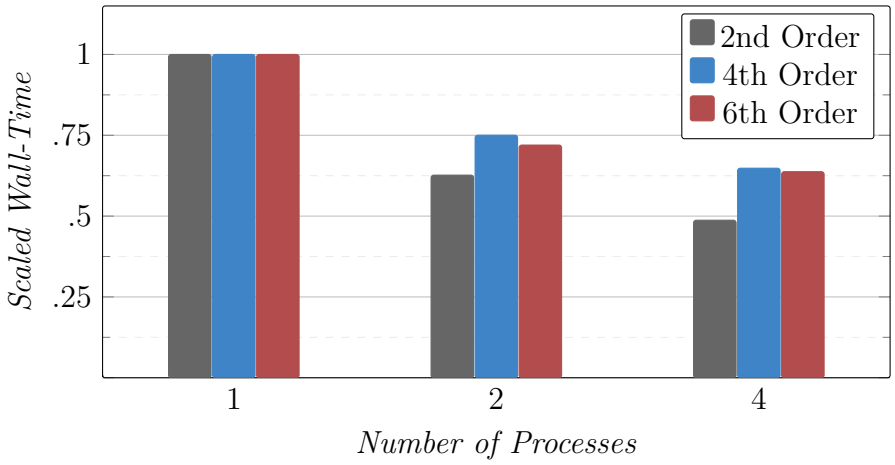


trait as the partial FFT solver combines two versions of the FFT solver and the generalized eigenvalue solver. Both the partial FFT and generalized eigenvalue solvers are quicker than the fast solver. When comparing the solvers to Matlab's backslash there is no contest. The partial FFT solver is up to 20 times faster in wall-time than backslash in the case where $N_x = 65$ and $N_y = 2049$, but only slightly faster when $N_x = 2049$ and $N_y = 65$. The drastic difference of the wall-time in these two cases is expected because the eigenvalues and eigenvectors are computed for an $N_x \times N_x$ matrix and the parallelism is based on the y direction.

Figures 8.4 and 8.5 give the scaled wall-times of the second-, fourth- and sixth-order implementations on two different grids. These times are found using one, two and four OpenMP threads. Each order of approximation shows a reduction in time. The second-order case performs the best. The better performance is due to the reduced number of computations because the matrix B_a from Chapter 6 is simply the identity matrix.

The results in Subsection 8.2.1 demonstrate the efficiency of both the partial FFT and

Figure 8.5: Constant k Partial FFT Solver OpenMP on 2049^2



the generalized eigenvalue solvers. The comparison to the second-order solver developed by Toivanen and Wolfmayr in [10] shows an increased ability to utilize larger grids with a faster solution time. Both the solvers show an immense improvement from Matlab's backlash solver.

8.2.2 Air and Soil Medium

Subsection 8.2.1 focused on the case with constant k for a comparison with the fast solver [10]. Subsection 8.2.2 examines the parallelization of the generalized eigenvalue and partial FFT solvers implementations on test problems that push beyond the capabilities of the fast solver given by Toivanen and Wolfmayr in [10]. All tests were run on the MacBook Pro laptop. The domain considered is $\Omega = [-1, 1] \times [-1, 1]$. Here k varies in the vertical direction representing an air and soil medium as shown in Figure 6.1. If $y = 0$ is the location of the

Table 8.11: Partial FFT Solver Residuals for Air and Soil Problem

Grid	Second-Order	Fourth-Order
129^2	9.57E-12	3.64E-12
257^2	2.41E-11	1.07E-11
513^2	4.82E-11	3.33E-11

interface between the air and soil, then

$$k(\mathbf{x}) = \begin{cases} \sqrt{439.2} & \text{for } y < 0 \\ \sqrt{1273 + 31i} & \text{for } y \geq 0 \end{cases}$$

where $\mathbf{x} \in \mathbb{R}^2$. The values in k are chosen for realistic representations of air and soil [13]. The right-hand side for this test is (8.2). Table 8.11 demonstrates the accuracy of the partial FFT solver by showing that the residual is nearly zero in each case tested.

Tables 8.12 and 8.13 give the results for the second-order case with computational grids $N_x = N_y = 2049$ and $N_x = N_y = 4097$ respectively. The generalized eigenvalue solver's initialization includes the time required to find the eigenvalues and eigenvectors of the matrices B_a and B_g. Additionally, the matrices D_a, D_g and LU decomposition for the matrix A_s are computed. The definitions of these matrices are given in Chapter 6. The initialization required for the FFT solver is the LU decomposition of the diagonalized system described in Chapter 5. When the grid size is 2049^2 it is seen that the partial FFT solver can find the numerical solution in roughly half the time it takes the generalized eigenvalue solver. The partial FFT solver becomes more advantageous with larger grid sizes as shown in Table 8.13 with a grid size of 4097^2 where the partial FFT solver reduces the wall-time of the generalized eigenvalue solver by more than half.

Table 8.12: Generalized Eigenvalue and Partial FFT Solvers OpenMP on 2049^2

		1 Thread	**2 Threads**	**4 Threads**
Initialization	FFT	0.39213	0.241561	0.204826
	EIG	53.996724	40.764139	32.553124
Generalized Eigenvalue Solver		4.981626	3.131687	1.902369
Partial FFT Solver	FFT	0.756528	0.442337	0.253204
	EIG	1.12378	0.816346	0.775145
	FFT	0.604237	0.384812	0.188181
	Total	2.484601	1.643549	1.216581

EIG refers to the generalized eigenvalue solver.

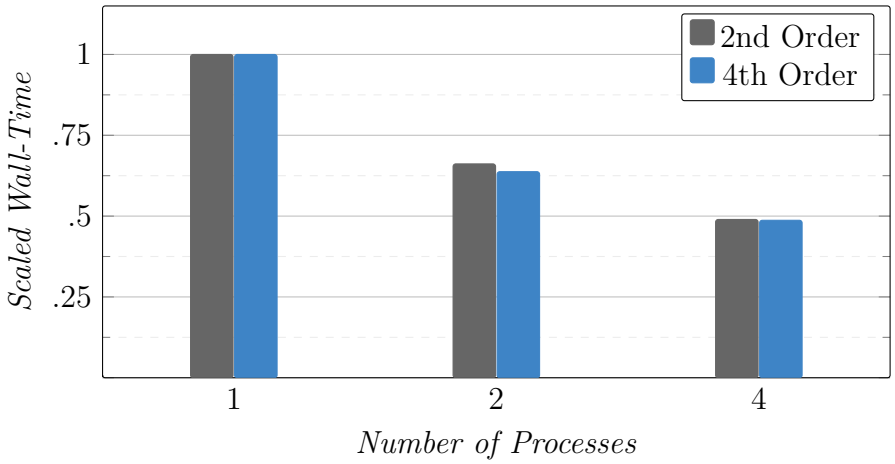
Table 8.13: Generalized Eigenvalue and Partial FFT Solvers OpenMP on 4097^2

		1 Thread	**2 Threads**	**4 Threads**
Initialization	FFT	1.771211	1.12604	0.837973
	EIG	389.701499	289.154195	242.122615
Generalized Eigenvalue Solver		34.754707	20.48377	13.261538
Partial FFT Solver	FFT	3.215323	1.749894	1.09352
	EIG	4.713536	3.52659	3.246609
	FFT	2.691277	1.493174	0.833005
	Total	10.620197	6.769714	5.173189

EIG refers to the generalized eigenvalue solver.

Figure 8.6 shows the wall-times for second- and fourth-order on a grid size of 2049^2. The times shown are from running the partial FFT solver where favorable wall-time speed-up is observed. The results shown in this subsection demonstrate the partial FFT solver's improvement in solution time over the generalized eigenvalue solver. Furthermore, strong parallelism is shown by the reduction in wall-time with the addition of processes. The most significant improvement the partial FFT solver has over the fast solver is the ability to solve

Figure 8.6: Variable k Partial FFT Solver OpenMP on 2049^2



problems with k varying in the vertical direction [10]. The highly parallel nature of the partial FFT solver with k varying in the vertical direction shows that it can provide an efficient preconditioner in an iterative solver for the inclusion problem.

8.2.3 Air and Soil Medium with Subsurface Inclusions

Subsection 8.2.3 considers the subsurface inclusion test problem. The following presents a mathematical model for the propagation of an electromagnetic field. Let $\mathbf{x} = (x, y) \in \mathbb{R}^2$ and the air-soil interface be at $y = 0$. Air is given by $y < 0$ and the soil by $y \geq 0$ as shown in Figure 6.3. Let E_0 be an electrical field originated by a ground-penetrating radar. Consider E_0 to be a linearly polarized plane wave with the direction of propagation parallel to the positive direction of the y axis: $E_0 = (0, e^{i\omega y})$, where ω is the angular frequency [13].

Maxwell's system implies that $E = (0, u(\mathbf{x}))$ where the function $u(\mathbf{x})$ satisfies the following Helmholtz equation

$$0 = \Delta u(\mathbf{x}) + k^2(\mathbf{x})u(\mathbf{x}) \tag{8.3}$$

$$k^2(\mathbf{x}) = \omega^2\mu\varepsilon(\mathbf{x}) + i\omega\mu\sigma(\mathbf{x}). \tag{8.4}$$

Consider the so called loss tangent $\tan(\delta) = \sigma/(\omega\varepsilon)$. The substitution of (8.3) into (8.4) gives $k^2 = \omega^2\mu\varepsilon(1 + i\tan(\delta))$. Assume that $\mu \equiv \mu_0$ and $\varepsilon \equiv \varepsilon_0\varepsilon_r$, where μ_0 and ε_0 are the magnetic permeability and the dielectric permittivity of a vacuum and ε_r is the relative dielectric constant. In addition, assume $\varepsilon = \varepsilon_1$ and $\tan(\delta) = 0$ in air, when $y < 0$. In the soil

$$\varepsilon_r = \varepsilon_r(\mathbf{x}) = \varepsilon_{r1} + \Delta\varepsilon_{r1}(\mathbf{x}) > 0,$$

$$\tan(\delta(\mathbf{x})) = \tan(\delta_1) + \Delta\tan(\delta_1(\mathbf{x}))$$

where ε_{r1} and $\tan(\delta_1)$ are positive constants and the perturbations $\Delta\varepsilon_{r1}(\mathbf{x})$ and $\Delta\tan(\delta_1(\mathbf{x}))$ are due to the presence of small mine-like inclusions. Assume that the perturbations $\Delta\varepsilon_{r1}(\mathbf{x})$ and $\Delta\tan(\delta_1(\mathbf{x}))$ are only supported within these inclusions. The method under consideration can be generalized to a layered medium. In this case ε_{r1} and $\tan(\delta_1)$ are functions depending on only the y direction. Let $k_0(\mathbf{x})$ be the function $k(\mathbf{x})$ without inclusions present, as

$$k_0^2(\mathbf{x}) = \begin{cases} \omega^2\mu_0\varepsilon_0, & \text{if } y < 0 \\ \omega^2\mu_0\varepsilon_0\varepsilon_{r1}\left[1 + i\tan(\delta_1)\right], & \text{if } y \geq 0. \end{cases}$$

Therefore, the function $k_0(\mathbf{x})$ has constant values both in air and soil with a discontinuity at the air-soil interface. Further, let $u_0(\mathbf{x})$ be the solution of equation (8.3) which corresponds to the initializing plane wave e^{ik_0y} where $y < 0$. Consider the function $u(\mathbf{x})$ in the form $u(\mathbf{x}) = u_0(\mathbf{x}) + v(\mathbf{x})$ where the function $v(\mathbf{x})$ represents the wave scattered by the mine-like

Table 8.14: Electrical Parameters

	ε_r	$\tan(\delta)$	$k^2(\omega = 1GHz)$
Air	1	0	439.2
Soil	2.9	0.025	$1273 + 31i$
Inclusion	2.86	0.0018	$1050 + 2.26i$

inclusions. This function satisfies the radiation boundary conditions (1.3). Then

$$u_0(\mathbf{x}) = \begin{cases} e^{iyk_0(\mathbf{x})} + ae^{-iyk_0(\mathbf{x})}, & \text{if } y < 0 \\ \\ be^{iyk_0(\mathbf{x})}, & \text{if } y \geq 0 \end{cases}$$

where a and b are the reflection and transmission coefficients defined as

$$a = \frac{k_a - k_g}{k_a + k_g}, \qquad b = \frac{2k_a}{k_a + k_g}.$$

Here k_a and k_g are the values of $k_0(\mathbf{x})$ for $y < 0$ and $y \geq 0$, respectively, as in Figure 6.1. The presence of these coefficients ensures the continuity of the function $u_0(\mathbf{x})$ together with its first derivatives at the air-soil interface. Substituting $u(\mathbf{x}) = u_0(\mathbf{x}) + v(\mathbf{x})$ into (8.3), gives the equation in the form (1.1) with

$$f(\mathbf{x}) = \begin{cases} 0, & \text{outside inclusions,} \\ \\ (k_g^2 - k^2(\mathbf{x}))u_0(\mathbf{x}), & \text{inside inclusions.} \end{cases}$$

The model described above is now implemented on a problem with a mine-like target in wet soil [13]. The ranges of parameters $k(\mathbf{x})$ are given in the Table 8.14. The domain

Table 8.15: Partial FFT Solver Residuals for Inclusion Problem

Grid	Second-Order	Fourth-Order
129^2	5.64e-14	2.47e-14
257^2	1.63e-13	4.69e-14
513^2	3.83e-13	1.19e-13

considered is $\Omega = [-1, 1] \times [-1, 1]$. The function $k(\mathbf{x})$ is defined as

$$k(\mathbf{x}) = \begin{cases} 439.2 & \text{if } y < 0 \\ 1273 + 31i & \text{if } y \geq 0 \text{ and } y \notin S \\ 1050 + 2.26i & \text{if } y \in S \end{cases}$$

where $S = \{\mathbf{x} \in \Omega \mid |x - .3| \leq .15 \text{ and } |y - .2| < .04\}$ is the set of grid points within the rectangular inclusion with width .15, height .04 and center (.03,.2).

The partial FFT solver was tested on the problem described above. Table 8.15 shows the residual of multiple runs on the MacBook Pro. All residuals are approximately zero with respect to machine error, an accuracy rarely produced by iterative methods [5, 12]. Figure 8.7 illustrates the domain with air, soil and a rectangular mine-like inclusion. Figure 8.8 shows the numerical solution from the second-order scheme on a grid size of 129^2. Figure 8.9 shows the speed-up in wall-time from one to four OpenMP processes for the calculation of the numerical solutions with grid sizes of 257^2 and 513^2. The wall-time is reduced by approximately one half, from one to four process.

8.3 Summary

Results shown in Chapter 8 demonstrate the highly parallel FFT solver's ability to reduce wall-time with the addition of more processes. The FFT solver is versatile in its application as

Figure 8.7: 2D Domain with Air, Soil and Rectangular Inclusion

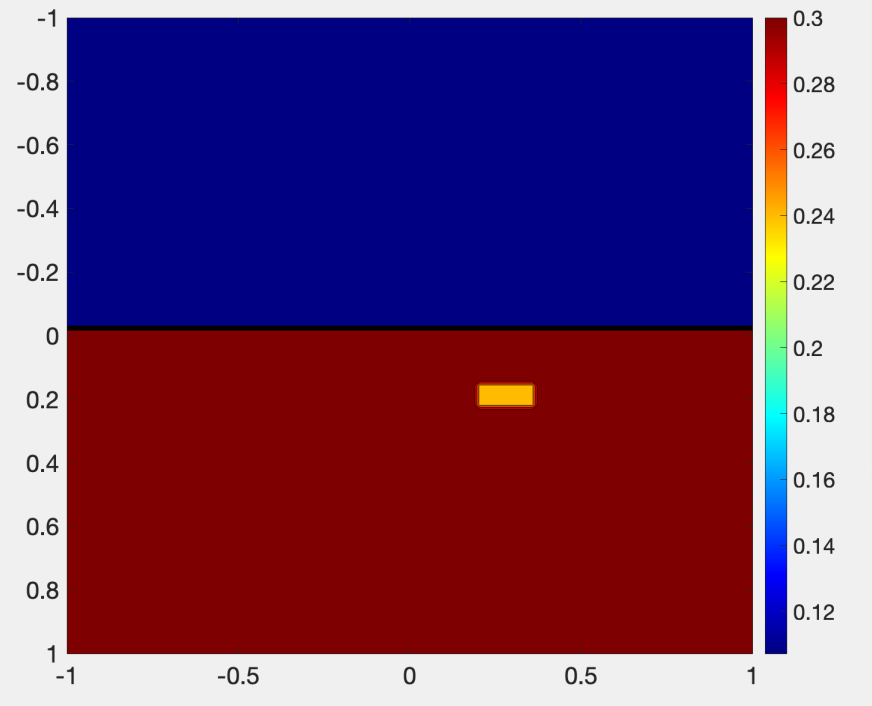


Figure 8.8: Real Part of 2D Solution with Rectangular Inclusion

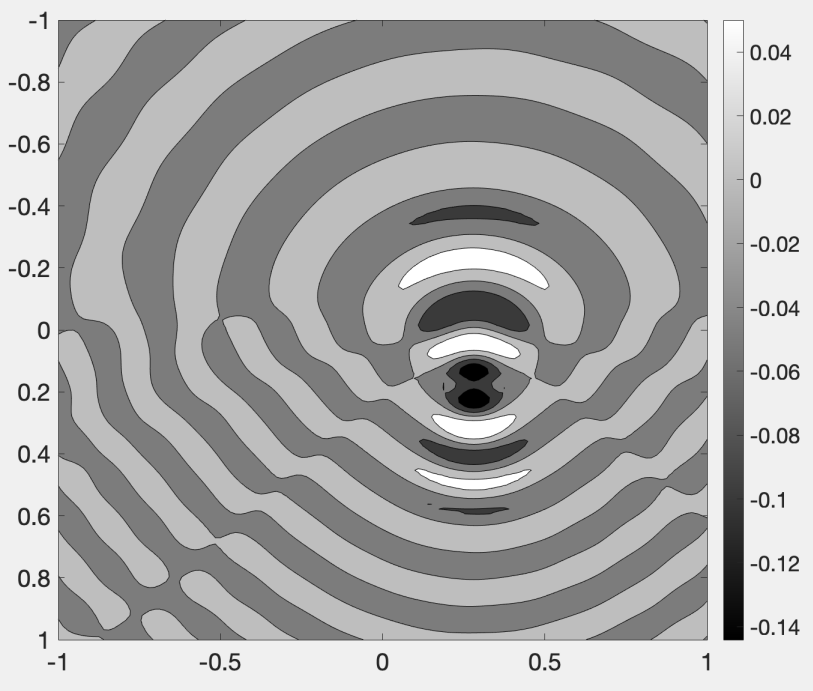
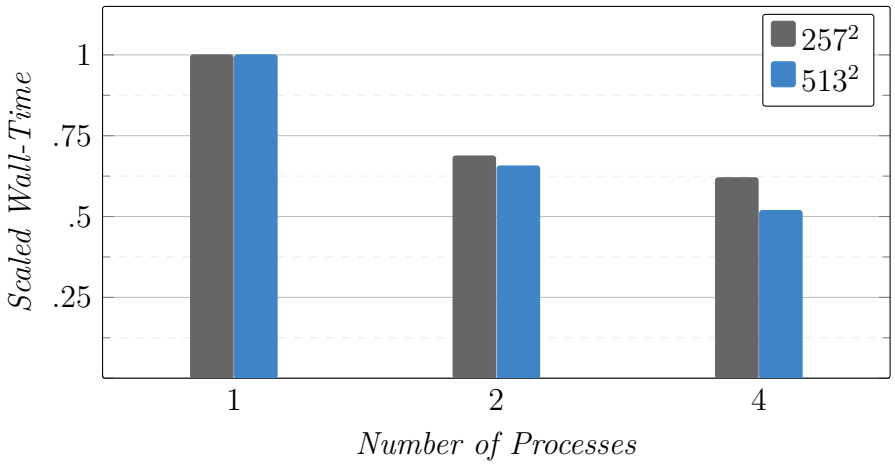


Figure 8.9: Partial FFT Solver OpenMP with Rectangular Inclusion



shown in the convection-diffusion example. The partial FFT solver demonstrated the ability to directly solve the subsurface scattering problem with high-accuracy. This accuracy is rarely produced by iterative methods. Furthermore, the partial FFT solver demonstrates a significant level of parallelism by the reduction of wall-time with the addition of OpenMP processes in the case of k varying vertically. Chapter 9 reviews the conclusions of the research study and proposes directions for future work.

9 Conclusion and Future Work

Increasing the resolution of existing numerical solvers is the focus of research in many areas of science and engineering. The research study added to this body of knowledge by utilizing high-order approximations combined with highly parallel solvers and showing the highly parallel solvers can provide high-resolution solutions for both the two- and three-dimensional Helmholtz equations. The versatility of these numerical methods in a diversity of applications was demonstrated. These methods provide an effective alternative approach to iterative methods when solving discretized systems that are neither positive definite nor Hermitian. The highly parallel nature of these methods provides the ability to produce extremely high-resolution solutions on problems for which traditional sequential methods fail. The parallel implementations reduce the computation time by taking advantage of both shared and distributed memory environments by use of OpenMP and MPI respectively.

Chapters 2 and 3 provided the second-, fourth- and sixth-order finite difference approximations of the Helmholtz equations in two- and three-dimensions respectively. These approximations were developed through closely following, and extending on previous work from [2, 3, 4, 5, 6]. Chapter 3 developed the fourth-order discretization for the convection-diffusion equation as an example of the versatility of the numerical methods. This discretization, under specific restrictions, falls into the class of linear systems that are effectively handled by the presented FFT solver. The Dirichlet and Sommerfeld-like boundary conditions were stated in Chapter 4 with a simple two-dimensional example on their implementations. Additionally, the Sommerfeld-like conditions and their approximates were described. The FFT solver's derivation and parallel implementation was provided in Chapter 5. Details on the solver's

parallelism along with the OpenMP, MPI and hybrid implementations were presented. The novel generalized eigenvalue and partial FFT solvers were developed in Chapters 6 and 7 respectively. The generalized eigenvalue solver's development is described in detail. An explanation on the improvements provided by the partial FFT solver are shown along with a description of the parallel implementation.

Chapter 8 presented numerical results of multiple test problems. The FFT solver's parallel efficiency is tested and demonstrates the impressive capabilities. An example of the diverse applicability of this solver is provided in the form of a generalization to the approximated solution of the convection-diffusion equation. Thorough examples of all the solvers' accuracy are shown. Results of tests were given that demonstrate the efficiency of the generalized eigenvalue and partial FFT solvers' parallel implementations. Tests with subsurface inclusions exemplify the accuracy of these numerical methods.

9.1 Future Work

The solvers presented in the research study are efficient and have a wide array of applications. The parallel implementations provide excellent speed-up in wall-time. However, improvements can be made. The solutions of the transformed systems in the generalized eigenvalue, and partial FFT solvers, use the Linear Algebra PACKage (LAPACK) for the LU decomposition and solution [28]. There is potential for improvement in parallelization, and therefore, computation time using a software package called SuperLU developed by in part by Sherry Li of Lawrence Berkeley National Laboratory [29].

Memory utilization can be reduced. The implementations require several arrays, but it may be possible for optimization by reusing allocated memory. Within the partial FFT

solver only portions of the matrices R, S, B_a and B_g are needed; the memory storing these matrices could be reduced to only the necessary portions.

Another potential improvement is in the calculation of the DST in the forward and reverse transforms in the FFT solver. The FFTW libraries contain optimized functions for computing the DST directly. It may be possible to reduce memory and computation time by utilizing these functions rather than finding the DST via DFT of extended arrays.

The forward solution of high-frequency electromagnetic wave propagation presented in the research provides a vital piece in the imaging of subsurface mine-like targets. The next step is using the partial FFT solver to generate many solutions to the forward problem with mine-like inclusions. Multiple variations on the inclusions would be considered, including different locations, sizes, and shapes of the inclusion. These solutions can provide images at the surface that may be used to train convolutional neural networks to give an estimated location, size and shape of the inclusion. The estimate can be used as an initial estimate in the reverse imaging algorithm.

10 Bibliography

[1] R. Robey and Y. Zamora, *Parallel and High Performance Computing*. Manning Publications Company, 2021.

[2] M. Nabavi, M. Siddiqui, and J. Dargahi, "A new 9-point sixth-order accurate compact finite-difference method for the Helmholtz equation," *Journal of Sound and Vibration*, vol. 307, no. 3, pp. 972 – 982, 2007.

[3] G. Sutmann, "Compact finite difference schemes of sixth order for the Helmholtz equation," *Journal of Computational and Applied Mathematics*, vol. 203, no. 1, pp. 15 – 31, 2007.

[4] S. Lele, "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics*, vol. 103, 1992.

[5] E. Turkel, D. Gordon, R. Gordon, and S. Tsynkov, "Compact 2D and 3D Sixth Order Schemes for the Helmholtz Equation with Variable Wave Number," *Journal of Computational Physics*, vol. 232, pp. 272 – 287, 2013.

[6] Y. Gryazin, "High-order approximation compact schemes for forward subsurface scattering problems," in *Radar Sensor Technology XVIII* (K. I. Ranney and A. Doerry, eds.), vol. 9077, pp. 131 – 139, International Society for Optics and Photonics, SPIE, 2014.

[7] L. Evans, *Partial Differential Equations*. American Mathematical Society, second edition ed., 2015.

[8] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer, fourth edition ed., 2019.

[9] I. Graham and S. Sauter, "Stability and finite element error analysis for the Helmholtz equation with variable coefficients," *Mathematics of Computation*, vol. 89, no. 321, pp. 105 – 138, 2020.

[10] J. Toivanen and M. Wolfmayr, "A fast Fourier transform based direct solver for the Helmholtz problem," *Numerical Linear Algebra with Applications*, vol. 27, 2020.

[11] I. Graham, U. Langer, J. Melenk, and M. Sini, eds., *Direct and Inverse Problems in Wave Propagation and Applications*. De Gruyter, 2013.

[12] Y. Gryazin, Y. Lee, and R. Gonzales, "Scalable high-resolution algorithms for landmine imaging problem," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIV* (S. S. Bishop and J. C. Isaacs, eds.), vol. 11012, pp. 241 – 250, International Society for Optics and Photonics, SPIE, 2019.

[13] Y. Gryazin, "Application of multigrid approach to the subsurface imaging problem," in *Detection and Remediation Technologies for Mines and Minelike Targets VIII* (R. S. Harmon, J. H. H. Jr., and J. T. Broach, eds.), vol. 5089, pp. 1271 – 1278, International Society for Optics and Photonics, SPIE, 2003.

[14] D. Chopp, *Introduction to High Performance Scientific Computing*. SIAM, 2019.

[15] M. Frigo and S. Johnson, "FFTW Documentation." Massachusetts Institute of Technology, 2020.

[16] R. Gonzales, Y. Gryazin, and Y. Lee, "Parallel FFT algorithms for high-order approximations on three-dimensional compact stencils," *Parallel Computing*, vol. 103, p. 102757, 2021.

[17] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, first edition ed., 1990.

[18] I. Harari, "A survey of finite element methods for time-harmonic acoustics," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 13, pp. 1594 – 1607, 2006. A Tribute to Thomas J.R. Hughes on the Occasion of his 60th Birthday.

[19] E. Turkel, "Numerical difficulties solving time harmonic systems," *Nato Science Series Sub Series III Computer and Systems Sciences*, pp. 319 – 337, 2001.

[20] Y. Gryazin, M. Klibanov, and T. Lucas, "GMRES Computation of High Frequency Electrical Field Propagation in Land Mine Detection," *Journal of Computational Physics*, vol. 158, no. 1, pp. 98 – 115, 2000.

[21] H. Elman and D. O'Leary, "Efficient iterative solution of the three-dimensional helmholtz equation," *Journal of Computational Physics*, vol. 142, pp. 163 – 181, 1998.

[22] H. Elman and D. O'Leary, "Eigenanalysis of Some Preconditioned Helmholtz Problems," *Numerische Mathematik*, vol. 83, 03 2001.

[23] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*. Oxford University Press, 2014.

[24] R. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, first edition ed., 2007.

[25] R. Hostetler and R. Larson, *Trigonometry*. Houghton Mifflin Company, fifth edition ed., 2000.

[26] L. Ahlfors, *Complex Analysis*. McGraw-Hill, third edition ed., 1979.

[27] R. Kress, *Numerical Analysis*. Springer, 1998.

[28] E. Anderson, Z. Bai, C. Bischof, L. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, third edition ed., 1999.

[29] X. Li, "An Overview of SuperLU: Algorithms, Implementation, and User Interface," *toms*, vol. 31, no. 3, pp. 302 – 325, 2005.