A-STABLE IMPLICIT RAPID SCHEME AND SOFTWARE SOLUTION FOR ELECTROMAGNETIC WAVE PROPAGATION

By

Mathialakan Thavappiragasam

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering - Doctor of Philosophy
Computational Mathematics, Science and Engineering - Dual Major

2019

ABSTRACT

A-STABLE IMPLICIT RAPID SCHEME AND SOFTWARE SOLUTION FOR ELECTROMAGNETIC WAVE PROPAGATION

By

Mathialakan Thavappiragasam

A robust and rapid scheme to solve electromagnetics (EM) is an important requirement in the scientific computing environment in which there are several useful methods used to solve tasks in EM. Our research study is motivated by this need and is targeted to develop a fast A-stable implicit numerical scheme and scalable software solution for EM wave propagation. Our scheme is based on the Method Of Lines Transpose (MOLT) approach which discretizes time first and then solves boundary value problems. By applying the free-space Green's function, the solution is derived by decomposing particular and homogeneous solutions. The compact Simpson's quadrature based, O(N) fast convolution, a recursive algorithm, is used to solve the particular solution for N number of grid points. The homogeneous solution is obtained using a particular solution at the boundary points and the applied boundary conditions. The multi-dimensional scheme is developed using the ADI splitting approach and an arbitrary order accuracy in time is achieved by switching the time derivation to a spatial derivation using the Lax-Wendroff approach.

The focus of the work in this thesis has been to overcome the limitations in Neumann and outflow boundary conditions to get high-order accuracy by using special treatments that deal with a choice of the interpolation, finite difference stencil, and the initial conditions. In addition, we have extended these ideas to construct perfectly electrically conducting boundary conditions in 2D for the MOLT.

In addition to introducing higher-order boundary conditions, an embedded boundary method is employed to deal with complex geometries. As the method is A-stable, it does not suffer from small-time step limitations that are found in explicit finite difference time domain methods when using either embedded boundary or cut cell methods to capture geometry. Further, we are developing an open source code MOLTN (Method Of Lines Transpose, Nth

order) which is intended to be a hardware-independent, scalable software tool, using multi-node MPI, multi-core OpenMP, and GPU CUDA implementation. As a test case of the method, we implement and study the A6 magnetron with our embedded boundary method using point sources inside of the domain.

The eventual goal is to combine this method with a novel particle method for the simulations of plasma. The particle method would treat particles as point particles that generate fields that are tracked on the mesh. No density or current will be mapped to the mesh. The consistency and performance of the scheme are evaluated for EM wave propagation and scattering using different shaped objects including curved boundaries and the introduction of true point sources that demonstrate how we handle particles. Stable solutions result for a wide range of mesh sizes and potential to leverage novel computing architectures, such as GPU, have been demonstrated.

For my beloved late parents (Thavappiragasam and Pushpavathy) who motivate me beyond, to move toward my dream

ACKNOWLEDGEMENTS

My first and deepest gratitude goes to my advisor Professor Andrew Christlieb for giving me point to point guidance and support to achieve the target successfully; thank god for giving me such a mentor for my Ph.D. studies. Dr. John Luginsland is the next person to bring me through these studies. Thank you so much for your valuable guidance sir.

I will never forget your help, Professor Shanker Balasubramaniam; you established a successful path to move forward, thank you so much! I would also like to thank Professor John Verboncoeur, Professor Edward J. Rothwell, Professor Brian O'Shea for their guidance and thoughts over my Ph.D. studies. Without your support, I couldn't imagine success.

I would like to thank Dr. Matthew F. Causley, Dr. Aditya Viswanathan, and Dr. Pierson Guthrey for their support and collaborative works to accomplish every milestone, and also I am happy to thank Dr. Eric Wolf for his valuable and timely supports. The Christlieb-group, the bleeding edge research team with active researchers, is the team I have been working throughout my Ph.D. studies. Thank you colleagues, William Sands, Dr. Hana Cho, Dr. Michel Crockatt, Dr. Xiao Feng, Dr. Bosu Choi, Dr. Yan Jiang, Dr. Wei Guo, Dr. Bankim Mandal, Dr. Rouchun Zhang for your tips and tricks. I would also like to thank Dr. Bernard G. Pope, a professor emeritus of physics at the Michigan State University, for thesis proofreading and editing.

My thanks also go to professors who taught me courses at Michigan State University, and the faculty and staff that are working in the Department of Electrical and Computer Engineering and Computational Mathematics, Science, and Engineering.

Finally, I am proud to thank my family for their support and patience, especially my loving wife Kukapalini for her supports and inspirational words of encouragement to carry my studies and our family concurrently. I also want to thank friends and relatives for their motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1: Introduction

1.1 Context and Motivation

Scientists have been researching Higher Power Microwave (HPM) tubes to improve their performance in new technologies, as well as identifying applications in different areas, such as medicine, waste decomposition, radar systems, plasma screens, linear accelerators, etc. Beyond working in the Electromagnetic (EM) experimental labs, researchers also have been working on the development of HPM simulation tools using different numerical schemes. Simulations can provide insight into real experiments. Moreover, they are very flexible to implement and can test new ideas using easy, fast, and cost-efficient design cycles. In order to perform EM simulations, first, we have to solve Maxwell's equations. Several approaches have been introduced to solve Maxwell's equations. They basically fall in the broad category of Finite Difference (FD) Methods, Finite Element Methods (FEM), Method of Moments (MOM), or hybrid techniques which combine two or more methods (e.g., Finite Element Time Domain (FETD) with Finite Different Time Domain (FDTD) or FETD with MOM). Each of the aforementioned methods has pros and cons, For example, the FDTD-based explicit Yee scheme [71] has been used as a benchmark method for more than 60 years because it preserves certain physical properties very well namely, Gauss's law and the continuity condition. The drawback of the method is that it fails to handle curved boundaries in Cartesian grids due to staircasing issues. Further, explicit FD schemes can become unstable when the surface bounds a medium [52]. In contrast, the FETD method is a very prominent approach for complex geometries but this requires the solution of large systems of linear equations. Such methods are expensive and hard to scale due to the bottleneck of matrix inversions. Further, FETD schemes based on the Newmark beta method are unconditionally

stable but exhibit only second-order accuracy [29]. In the same spirit as FEM, the MOM was developed to address complex problems in EM. The method transforms the boundary-value problem into a system of linear equations [42]. Consequently, it suffers from the same performance issues as FEM, since these matrices need to be inverted. The MOLT based scheme developed by Causley et.al. [15; 13] was designed to address the issues faced by FDTD and FETD methods. It is an unconditionally stable, fast, and implicit scheme, capable of arbitrary order accuracy in time for Dirichlet and periodic boundaries on Cartesian grids. The method reduces to second order for outflow boundary condition and Neumann boundary condition on curved boundaries. My work extended the order of accuracy for these problems to arbitrary order, which was verified through refinement studies. We show fourth-order spatiotemporal accuracy for complex geometry problems including curved boundaries.

Further, a general framework was defined for problems with complex geometry using an embedded boundary approach. Simulation tools for HPM tubes, such as A6 magnetron, 12 and 18 cavity rising suns were developed using a PEC boundary condition. The scheme was derived for the electromagnetic vector potential using the Lorenz gauge which imposed the PEC boundary condition in 2D. I evaluated the simulation of A6 magnetron using a ping test and obtained six-strong resonance modes. This is identified as a challenging task because the existing tools may fail to give an accurate or a robust solution [39; 40; 51; 67], or the methods are limited to second-order accuracy [32]. The Discontinuous Galerkin PIC-based approach fails to obtain the exact six resonance modes for the simulation of an A6 magnetron due to the diffusivity of the scheme [39; 40]. The multiphysics simulation software tool VSim which is based on conformal Finite Difference Time Domain (FDTD) methods and conformal particle boundary conditions, fails to give robust solutions in the simulation of A6 magnetrons for every resolution. The user-configurable code, MAGIC which is an EM FDTD-PIC code, is limited to the second-order accuracy[32].

We will review the needs and challenges for the simulation of magnetrons, specifically A6 magnetron in the next section.

1.2 High Power Magnetrons

The Magnetron is the most tunable Higher Power Microwave (HPM) source with 30% tunable range [62]. It was invented by Arthur Hull in 1913, and magnetrons were built on his original principles in the 1920s and 1930s with power levels of about 100W. In the 1940s, the magnetrons developed by Boot and Randell achieved 10kW on their initial development. In August 1940, the cavity magnetron based new weapons development for the US defense systems during World War-II was led by Sir Henry Tizard. The magnetron driven radar had a great impact on the war and the subsequent technical development of the magnetrons was mainly carried out at the MIT radiation laboratory. The postwar development of magnetron and related technologies has been enhanced in multi-directions. Such technology is strapping, a mode selection technology which shifts the frequency spectrum by connecting alternate resonators and prevents mode hopping. The rising suns, designed by alternating between two resonators' shape to separate the mode in frequency. A new generation of high-power frequency-agile magnetrons was introduced for electronic warfare applications. The coaxial magnetron, introduced by Feinstein and Collier, allows stable tuning by use of a cavity surrounding the magnetron resonator. The Magnetron Injection Gun was introduced by French and the relativistic magnetron (high current extension of the conventional magnetron) was developed by Bekefi and Orzechowski in 1976 [5; 23; 32].

Nowadays, magnetrons, high output power (GW-class) microwave tubes, are used in numerous applications such as communication systems, radar, warfare, medical X-ray sources, and microwave ovens. To study and enhance the performance of the magnetrons, magnetron developers need a reliable and accurate simulation method. The research studies on the simulation of magnetrons have been carried out in academic research labs, national research labs as well as electrical-electronic-computer engineering industries. For this simulation, we need to derive a time-dependent solution for Maxwell's equations that is applicable for complex geometries and should have the ability to incorporate with particles. In recent years, the conventional FDTD particle-in-cell (PIC) method has been widely used to study magnetrons

as well as other microwave tubes. Even though the FDTD based Yee scheme is a well-agreed method for Maxwell's equations and preserves divergence-free quantities, because of an explicit scheme it suffers from CFL restrictions and is not suitable for curved surfaces due to the cut-cell staircasing approximation. Another approach, the conformal finite difference time domain (CFDTD) PIC method, treats cut-cells differently in order to maintain second-order accuracy for curved surfaces [51]. The CFDTD Dey-Mittra algorithm adjusts the area and lengths of cut cells in the equation of Faraday's law in Maxwell's equation, but its stability is limited by the CFL condition and it wouldn't be scalable. The ADI-FDTD is one of the most powerful schemes to solve Maxwell's equations because it relies on simple, one dimensional, tridiagonal system solvers in contrast to a single large system solver as is required by the Crank-Nicholson implicit method. However, the ADI-FDTD method is mostly used to solve non-complicated domains such as rectangular domain and is not applicable for complex geometries because of showing only first-order accuracy for stair-stepped curved boundaries. ADI-FDTD method combined with the Dey-Mittra embedded boundary method can model the curved domains associated with complex structures and time step sizes beyond the CFL limit [67]. The efficiency of this method depends on the one-dimensional tridiagonal solvers which are used underneath and that will cause a major bottle-neck which effects the scalability of the scheme. Further, the order of accuracy is limited to second-order.

Based on these studies, in order to simulate the HPM tube, we need a robust implicit scheme that applies to complex geometries. Hence, the MOLT based A-stable implicit scheme introduced in [14] is chosen to simulate HPM magnetrons and imposed embedded boundary methods to deal with complex geometries associated with these kinds of accelerator structures. We derived a time-dependent solution to Maxwell's equations in the vector potential form under the Lorenz gauge and imposed a perfect electric conductor (PEC) embedded boundary method. The scheme is unconditionally stable, shows fourth-order accuracy beyond the CFL restriction, and is freely scalable with high performance (requires $\mathcal{O}(n)$ cost for n grids). It doesn't depend on scalability restricting matrix operations but instead uses

free-space Green's function based fast convolution. We performed a cold test using our simulation study. The cold test is an experiment of classical electromagnetic/microwave studies without including charged particles and it can be done in either the frequency or the time domain. We determined the dispersion relation of the interacting structures or slow-wave structures and obtained the fundamental frequency mode using ping tests. It would be possible to carry out a follow up hot test which will include particles.

Figure 1.1 shows the 3D view of a relativistic A6 magnetron with diffraction output (MDO) [61]. It has a cylindrical cathode in the center, an anode with six vanes around the circle symmetrically, and a coupling horn which is connected with a cavity. We simulate the open cavity A6 MDO using our MOLT based scheme by imposing embedded Neumann (Chapter 4)/PEC (Chapter 5) and outflow boundary conditions for the scheme evaluation. Further, our simulation study includes a symmetrical A6 magnetron using PEC boundary conditions in 2D and Neumann boundary conditions in 3D. Further, we evaluate the A6 magnetron for the impulse response and resonance mode frequencies of it. A 2D view of an A6 magnetron without diffraction output is shown in Figure 1.2. This has a cathode of radius r_c and an anode with inner radius r_a, vane radius r_v, vane angle α_1, and cavity angle α_2.

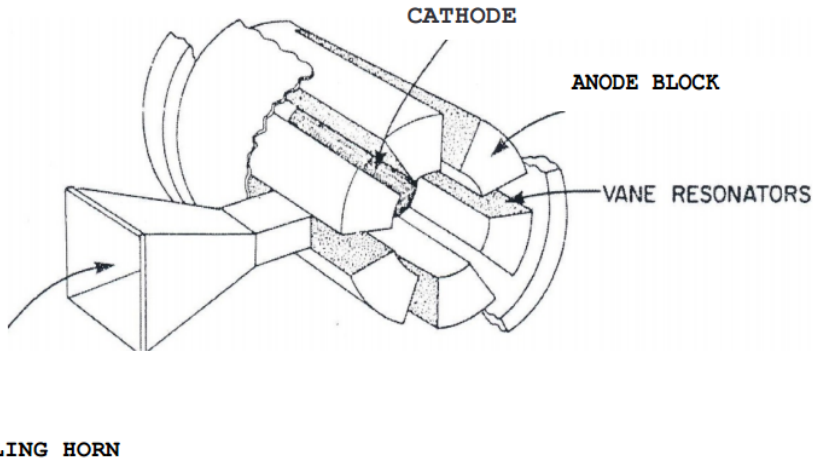


Figure 1.1: 3D Relativistic A6 magnetron with diffraction output (MDO) with a cylindrical cathode in the center [61]

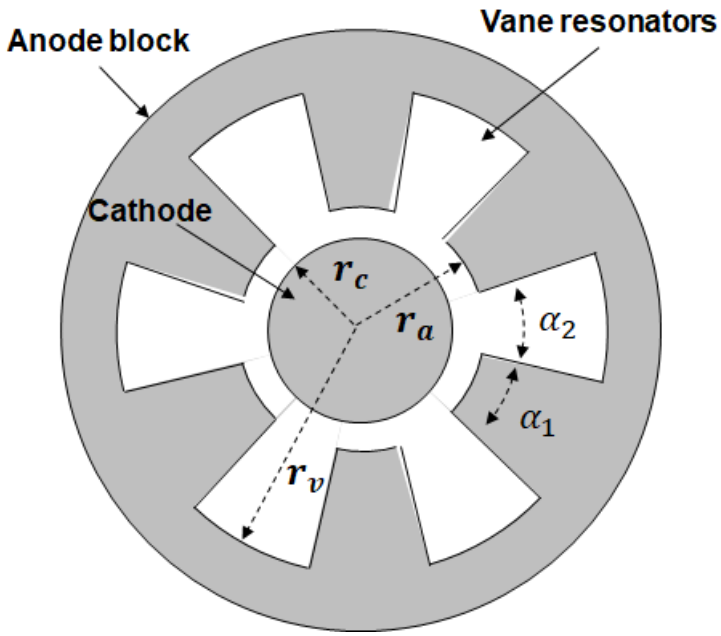


Figure 1.2: 2D view of A6 magnetron with a cathode of radius r_c and the anode with inner radius r_a, vane radius r_v, vane angle α_1, and cavity angle α_2

1.3 Research Goal and Objectives

We have been working towards developing a linear-time, fast higher-order, multi-dimensional, A-stable implicit solver that can be applicable for electromagnetic (EM) problems, specifically targeting plasma science. This approach uses MOLT formulation combined with an ADI scheme. In this way, algebraic approximations of original complex multi-physics problems can be mapped to utilize a computer's core competencies. We aim to develop several classes for leadership distributed multi-core computing platforms based on GP-GPU that is an $\mathcal{O}(N)$ direct implicit solver, and thereby enable exploration of a range of grand-challenge problems. We are motivated by the following objectives:

- The development of high-order outflow and Neumann boundary conditions in 3D and implementing a general geometric framework.

- The development of a high-order EM solver with no CFL constraint by developing

suitable boundary conditions for the vector-potential form of Maxwells equations under the Lorenz gauge. Develop a scheme for perfectly electrically conducting layers (PECs) in 2D.

- The development of high-order multi-core 2D and 3D implicit solvers on GPGPU for general geometries.
- The development of a scalable software solution using multi-core openMP and multi-node MPI.

We choose classic EM problems with complex geometries to assess our solver. These include photonic crystal waveguides, scattering of laser light on spherical objects, and simulating electron beams in an A6 relativistic magnetron.

1.4 Electromagnetic Wave Propagation

1.4.1 Electromagnetic Fields

The mathematical relationship of electromagnetic fields is governed by Maxwell's equations. The differential form of the macroscopic Maxwell's equations can be expressed as:

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0 \tag{1.1}$$

$$\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = \mathbf{J} \tag{1.2}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.3}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{1.4}$$

Where Equations 1.1 and 1.2 represent Faraday's law and Ampere's law respectively, and Equations 1.3 and 1.4 represent Gauss's law. Here $\mathbf{B}$ is the magnetic flux density (Wbm^{-2}), $\mathbf{D}$ is the electric flux density (Cm^{-2}), $\mathbf{E}$ is the electric field intensity (Vm^{-1}), $\mathbf{H}$ is the magnetic field intensity (Am^{-1}), $\mathbf{J}$ is the electric current density (Am^{-2}), and ρ is the

electric charge density (Cm^{-3}). In linear isotropic media, the electric flux density $\mathbf{D}$ and the magnetic flux density $\mathbf{B}$ have the constitutive relations with electric field intensity $\mathbf{E}$ and magnetic field intensity $\mathbf{H}$ as follows:

$$\mathbf{D} = \epsilon\mathbf{E} = \epsilon_0\epsilon_r\mathbf{E} \tag{1.5}$$

$$\mathbf{B} = \mu\mathbf{H} = \mu_0\mu_r\mathbf{H} \tag{1.6}$$

where the dielectric constant ϵ is the permittivity (Fm^{-1}), ϵ_0 is the free space permittivity and ϵ_r is the relative permittivity. Similarly, μ is the permeability, μ_0 is the free space permeability (Hm^{-1}), and μ_r is the relative permeability of the media. Upon applying the approximations (1.5 and 1.6), we get

$$\nabla \cdot \mathbf{H} = 0 \tag{1.7}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon} \tag{1.8}$$

Now consider the case where ϵ and μ do not vary with time, thus

$$\nabla \times \mathbf{E} + \mu\frac{\partial \mathbf{H}}{\partial t} = 0 \tag{1.9}$$

$$\nabla \times \mathbf{H} - \epsilon\frac{\partial \mathbf{E}}{\partial t} = \mathbf{J} \tag{1.10}$$

Since we have solvers for second-order wave equations, we convert the first order Maxwell system to second-order form.

We know the following relationship with the curl and divergence operations,

$$\nabla \times (\nabla \times \mathbf{v}) = \nabla(\nabla \cdot \mathbf{v}) - \nabla^2\mathbf{v} \tag{1.11}$$

Applying Equation 1.11 to $\mathbf{E}$, we find the wave equation for the electric field in the

presence of source,

$$\begin{aligned}
\nabla(\nabla \cdot \mathbf{E}) - \nabla^2\mathbf{E} &= \nabla \times (\nabla \times \mathbf{E}) \\
\nabla\left(\frac{\rho}{\epsilon}\right) - \nabla^2\mathbf{E} &= -\mu(\nabla \times \frac{\partial \mathbf{H}}{\partial t}) \\
&= -\mu\frac{\partial(\nabla \times \mathbf{H})}{\partial t} \\
&= -\mu\frac{\partial(\epsilon\frac{\partial \mathbf{E}}{\partial t} + \mathbf{J})}{\partial t} \\
&= -\mu\epsilon\frac{\partial^2\mathbf{E}}{\partial t^2} - \mu\frac{\partial \mathbf{J}}{\partial t} \\
\nabla^2\mathbf{E} - \frac{1}{c^2}\frac{\partial^2\mathbf{E}}{\partial t^2} &= \mu\frac{\partial \mathbf{J}}{\partial t} + \nabla\left(\frac{\rho}{\epsilon}\right)
\end{aligned} \tag{1.12}$$

where the speed $c = \frac{1}{\sqrt{\mu\epsilon}}$. Suppose there are no free charges, $\rho = 0$, and $\mathbf{J} = 0$ then we have source-free wave equation for the electric field,

$$\nabla^2\mathbf{E} - \frac{1}{c^2}\frac{\partial^2\mathbf{E}}{\partial t^2} = 0 \tag{1.13}$$

In the same way, upon applying Equation 1.11 to $\mathbf{H}$, we obtain

$$\begin{aligned}
\nabla(\nabla \cdot \mathbf{H}) - \nabla^2\mathbf{H} &= \nabla \times (\nabla \times \mathbf{H}) \\
-\nabla^2\mathbf{H} &= \nabla \times \left(\epsilon\frac{\partial \mathbf{E}}{\partial t} + \mathbf{J}\right) \\
&= \epsilon\frac{\partial(\nabla \times \mathbf{E})}{\partial t} + \nabla \times \mathbf{J} \\
&= \epsilon\frac{\partial(-\mu\frac{\partial \mathbf{H}}{\partial t})}{\partial t} + \nabla \times \mathbf{J} \\
&= -\mu\epsilon\frac{\partial^2\mathbf{H}}{\partial t^2} + \nabla \times \mathbf{J} \\
\nabla^2\mathbf{H} - \frac{1}{c^2}\frac{\partial^2\mathbf{H}}{\partial t^2} &= -\nabla \times \mathbf{J}
\end{aligned} \tag{1.14}$$

If $\mathbf{J} = 0$ then we have source-free wave equation for the magnetic field,

$$\nabla^2 \mathbf{H} - \frac{1}{c^2}\frac{\partial^2 \mathbf{H}}{\partial t^2} = 0 \tag{1.15}$$

We have derived a wave equation for the electric field (1.12) and the magnetic field (1.14) previously. However, since our targeting application domain is plasma science, we need to obtain electric scalar and magnetic vector potential formulation from the mixed potential form of Maxwell's equation which can easily deal with charged particles. We give a detailed derivation for the vector potential formulation in Chapter 5

1.4.2 Boundary Conditions for Electromagnetic Fields

The electromagnetic fields and flux densities are subject to boundary conditions; they can be derived from the integral form of Maxwell's equations. Let's consider a current and charge-free boundary s that separates the regions 1 and 2 as shown in Figure 1.3.

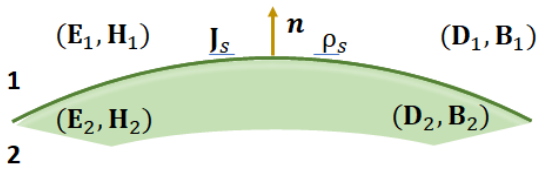


Figure 1.3: Boundary surface between two regions with electric fields $\mathbf{E}_1$ and $\mathbf{E}_2$, magnetic field $\mathbf{H}_1$ and $\mathbf{H}_2$, electric flux densities $\mathbf{D}_1$ and $\mathbf{D}_2$, and magnetic flux densities $\mathbf{B}_1$ and $\mathbf{B}_2$ respectively. Here, $\mathbf{J}_s$ is the surface current, ρ_s is the surface charge density, and $\mathbf{n}$ is the normal vector pointed out from the region two.

Faraday's and Ampere's laws predict electric and magnetic fields are continuous along the boundary, so the tangential boundary conditions can be expressed as,

$$\hat{n} \times (\mathbf{E}_1 - \mathbf{E}_2) = 0 \quad \hat{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = 0 \tag{1.16}$$

Gauss's law predicts the electric and magnetic fluxes are continuous along the normal to

the boundary surface, so the normal boundary conditions can be expressed as,

$$\hat{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = 0 \quad \hat{n} \cdot (\mathbf{B}_1 - \mathbf{B}_2) = 0 \tag{1.17}$$

Suppose the surface s has the induced surface electric current density $\mathbf{J}_s$ and the induced surface electric charge density ρ_s, the tangential and normal boundary conditions can be predicted from Faraday's, Ampere's, and Gauss's laws as follows,

$$\hat{n} \times (\mathbf{E}_1 - \mathbf{E}_2) = 0 \quad \hat{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s \tag{1.18}$$

$$\hat{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s \quad \hat{n} \cdot (\mathbf{B}_1 - \mathbf{B}_2) = 0 \tag{1.19}$$

1.5 Overview

We give a general overview of this dissertation and draw attention to the original contributions of the work. Basically, this work begins with MOLT based A-stable implicit solvers introduced by [15] and later developed to achieve higher-order schemes [14] and adopt embedded boundary methods [13]. We provide high-order outflow and Neumann boundary conditions in 3D while targeting 3D multi-core high-order implicit EM solvers on the GP-GPU platform. This work contributes to the development of high-order outflow and Neumann boundary conditions in 3D for general geometries and 3D high-order EM solver using the vector-potential form of Maxwells equations under the Lorenz gauge. Further, this work implements these solvers on a multi-core computing platform, GP-GPU in order to release a full-throttle multi-scale solver to enhance spatiotemporal performance. We now describe the structure of the dissertation section by section.

We start with an introduction of our work in Chapter 1 including motivation (in Section 1.1) and objectives (Section 1.3). In Chapter 2, we describe the available MOLT based implicit scheme (in Section 2.2.1) and explain how we extend the technique to a 3D solver

using an ADI scheme (in Section 2.3). We present numerical results for newly developed 3D solvers including variable speed test cases. Waveguides on a photonic crystal are used to assess our 2D variable speed solver.

High-order temporal schemes are explained in Chapter 3 including a novel scheme for high-order outflow boundary conditions described in Section 3.2. Further, this chapter includes numerical results for the proposed high-order 3D solver.

In Chapter 4, we begin with a description of the embedded method presented by [13] and extend it to higher-order using the method given in [15] for 2D and 3D problems. Further, we develop a general approach to deal with complex boundary problems in Section 4.3. We give an explanation of our high-order 3D scheme in Section 4.2.3 and present numerical results for complex boundary problems, electromagnetic wave scattering by PEC cylinders (4.4.1), electron beams in A6 Relativistic Magnetron in 2D (4.4.3), and scattering of Laser light on spherical objects in 3D (4.4.2) which are implemented using a fourth-order embedded boundary scheme.

In Chapter 5, we explain our versatile approach for electromagnetic vector potential and the implementation of PEC boundary conditions. We provide several test cases including frequency mode analysis for A6 magnetron in 2D. Chapter 6 explains nuts and bolts of a software solution which can be scalable using high-performance computing systems. Finally in Chapter 7 we summarize our work that was carried out throughout this study and give a list of possible directions that can be used for further advancement.

CHAPTER 2: Mathematical Model for Wave Equation Solver

2.1 Introduction

In this chapter, we review the novel impact method which we will further develop in later chapters. The new work in this chapter is the extension to 3D. We also discuss the advantages of this approach over other methods.

Maxwell's equations in free-space result in a hyperbolic wave equation with finite speed of propagation of a field quantity. We develop a numerical scheme in 3D to solve the hyperbolic wave equation by imposing different boundary conditions, specifically outflow and embedded Neumann and PEC boundaries. We are mainly focused on the development of an EM solver for problems with complex geometries, which can be easily accepted on multi-core technologies facilitated by GP-GPU computing platforms. Our main target is to derive a time-dependent solution for Maxwell's equations with higher-order accuracy in time and space.

In order to obtain the solution, we choose an implicit unconditionally stable Method Of Lines Transpose (MOLT - also known as the transverse method of lines or Rothes method) [4; 17; 38; 41; 54; 50; 31] based numerical scheme coupled with an embedded boundary approach to deal with complex geometry problems [15; 14; 13]. This was implemented in 2D and we extend the scheme to 3D by successful implementations for different applications. This approach uses a MOLT formulation combined with an Alternating Direction Implicit (ADI) scheme. The method avoids the use of matrices that typically result from the discretization of the spatial parts of a problem and thus eliminates the main bottleneck in scaling implicit methods. In this scheme, a PDE is first discretized in time, and then the resulting boundary-value problems are solved using a Green's function method. In particular, the inverse of the

resulting modified Helmholtz operator is analytically constructed and evaluated efficiently using an $O(N)$ recursive fast convolution algorithm. Extension to multi-dimensions is formed using an ADI scheme, and each line is solved independently. Next, we describe why this scheme is an optimal solution.

When we use this scheme to solve the hyperbolic wave equation, the spatial information along a line in the simulation is connecting the boundaries to all points in space along that line to advance a computational solution in time. Along this line, every point in space knows about the other points due to the nature of the Green's function used in the solution. The approximation of spatial convolution depends on the quadrature size we choose, and the accuracy of the time derivation can be extended by using the higher-order scheme which performs a set of spatial convolutions/sweeps in multiple computing levels. The hyperbolic wave equation form of Maxwell's equation is preserved by the time discretization and the spatial convolution. Further, we proved the scheme is strictly dispersing (no diffusive formulation) [53; 15], and satisfies the Gauss's divergence-free conditions. When we extend the scheme to multi-D using ADI splitting, the spatial points talk to each other by moving in each direction, and they communicate along the lines in that direction. We reduce the splitting error by doing a couple of sweeps along each direction and taking the average in multiple computing levels.

Historically, time-dependent solutions of the wave equation are most commonly constructed with the method-of-lines (MOL) approach [47; 48; 72; 59]. The MOL schemes consist of either collocation (nodal) methods or Galerkin (modal) methods [7; 12; 20]. The collocation methods (e.g.finite-difference, finite-volume, and finite-element methods) are used to find a solution at a particular point, but in Galerkin methods, the solution is projected onto a set of basis functions and the time-dependent coefficients are computed. The approach which has influenced most numerical methods in computational electromagnetics is Finite-difference time-domain (FDTD) methods [46; 65; 19], which use an explicit scheme, so these schemes are restricted by Courant-Friedrichs-Lewy (CFL) condition (concerning the

ratio of the time step to the spatial step) that limits the time step size. In 1966, Kane S. Yee derived an FDTD based time-dependent solution for Maxwell's equations [71]. The FDTD Yee scheme has been used for a large number of problems in computational electromagnetics. However, the scheme is limited to Cartesian grids and suffers from the staircase effect on curved boundaries. Further, it cannot be used to deal with discontinuities across material interfaces while maintaining a high-order of accuracy. Hence, the scheme is not suitable for curved objects or media with material interfaces [57; 19]. There have been many attempts apart from FDTD methods [46; 65; 19] to complex geometry and raise the order of these methods above two [21; 19; 11; 10; 18]. However, these methods have not made it into the mainstream because they are not robust. In addition to stability issues, these methods typically suffer from very small CFL [21; 19; 11; 18] conditions because of small cells at the boundaries.

In contrast, implicit schemes break the CFL restriction. Hence, the implicit schemes are most preferable to the problems with tightly coupled scales such as problems in plasma science. Alternating direction implicit FDTD (ADI-FDTD) is an implicit scheme that is used to develop an A-stable implicit Maxwell solver in several plasma-related problems [73; 58; 26; 27], but they are all second order in time. Further, the implicit Finite Element Time Domain (FETD) method based on the Newmark beta approach is unconditionally stable, but it gives second-order accuracy. Further, the implicit and frequency domain schemes require cost-expensive matrix inversion. Therefore, the MOLT based scheme is chosen to be an appropriate scheme for this simulation study in order to obtain high-order accuracy rapidly.

In Section 2.2, we give the implicit semi-discrete solution for the 1D wave equation, in Section 2.3 we explain how we can extend this framework to multi-dimensions. In Section 2.2.3, we derive equations for several applicable boundary conditions including outflow boundary conditions, and Section 2.4 explains details of the higher-order scheme, and finally Section 2.6 reports a set of test cases. The subsections of section 2.4 are laid out as follows.

In Section 2.4.1, we derive a family of schemes of order $2P$ for the one-dimensional case, and in Section 2.4.2, we generalize the higher-order scheme for higher spatial dimensions, producing ADI methods of order $2P$ which will be A-stable. Here P refers to the number of terms taken in the MOLT expansion.

2.2 One Dimensional Implicit Wave Equation Solver

Based on the initial boundary value problem with consistent boundary conditions, the wave equation for one spatial dimension can be written as follows,

$$\frac{1}{c^2}\partial_{tt}u - \partial_{xx}u = S(x,t), \quad x \in [a,b], \quad t < 0, \tag{2.1}$$

with initial conditions

$$u(x,0) = f(x), \quad x \in [a,b],$$
$$\partial_t u(x,0) = g(x), \quad x \in [a,b],$$

where c is the propagation speed and $S(x,t)$ denotes a source. This is well-posed once consistent boundary conditions are appended, such as:

1. Dirichlet boundary condition: $u(a,t) = U_L(t) \quad and \quad u(b,t) = U_R(t)$.
2. Neumann boundary condition: $\partial_x u(a,t) = V_L(t) \quad and \quad \partial_x u(b,t) = V_R(t)$.
3. Periodic boundary condition: $u(a,t) = u(b,t) \quad and \quad \partial_x u(a,t) = \partial_x u(b,t)$.
4. Outflow boundary condition: $\partial_t u(a,t) = c\partial_x u(a,t) \quad and \quad \partial_t u(b,t) = -c\partial_x u(b,t)$.

2.2.1 Semi-discretization

The semi-discrete form of the wave equation can be obtained as follows:
We start by discretizing u_{tt} using the second order time centered finite difference approxi-

mation,

$$\partial_{tt}u^n = \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} - \frac{\Delta t^2}{12}\partial_{tttt}u(x,\eta), \quad \eta \in [t_{n-1}, t_{n+1}]. \tag{2.2}$$

The Laplacian has to be evaluated at time t_{n+1} to obtain an implicit scheme. To perform this, we can apply a symmetric 3-point averaging for the Laplacian because we choose a centered finite difference stencil.

$$\partial_{xx}u^n = \partial_{xx}\Big(u^n + \frac{u^{n+1} - 2u^n + u^{n-1}}{\beta^2}\Big) - \frac{\Delta t^2}{\beta^2}\partial_{ttxx}u(x,\eta). \tag{2.3}$$

where $\beta > 0$.

Using (2.2) and (2.3), we obtain the second order semi-discrete equation,

$$\Big(\partial_{xx} - \frac{\beta^2}{(c\Delta t)^2}\Big)\Big(u^n + \frac{u^{n+1} - 2u^n + u^{n-1}}{\beta^2}\Big) = -\frac{\beta^2}{(c\Delta t)^2} - S(x,t_n) + O(\Delta t^2). \tag{2.4}$$

The differential operator in Equation (2.4) can be interpreted as a modified Helmholtz operator with,

$$\mathcal{L}_x[u] := \Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)u(x), \quad \alpha = \frac{\beta}{c\Delta t}, \quad x \in [a,b]. \tag{2.5}$$

On rearranging equation (2.4), we form the modified Helmholtz equation which can be written as,

$$\mathcal{L}_x[u^{n+1} - (2-\beta^2)u^n + u^{n-1}] = \beta^2\Big(u^n + \frac{1}{\alpha^2}S^n\Big). \tag{2.6}$$

We solve this modified Helmholtz equation (2.6) by inverting the Helmholtz operator $\mathcal{L}$ using the free space Greens function. In this way, we start with the solution subject to the boundary $[-\infty, \infty]$ and then map it to the actual boundary $[a,b]$ by making the boundary adjustment terms. Next, we describe the process in detail beginning with the derivation of

Green's formula. Equation (2.4) can be written as,

$$\Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)\psi(x) = s(x), \quad x \in [a, b], \tag{2.7}$$

where $\psi(x) = u^{n+1} - (2-\beta^2)u^n + u^{n-1}$ and $s(x) = \beta^2\Big(u^n + \frac{1}{\alpha^2}S^n\Big)$, subject to the homogeneous Dirichlet boundary condition

$$\psi(a) = 0, \psi(b) = 0, \tag{2.8}$$

The semi-discrete form of Equation (2.6) approximates the hyperbolic wave equation with the second-order of accuracy in time using centered finite difference, and the symmetric formulation removes the numerical dissipation. The scheme was proved as purely dispersive [53], so it is more favorable for long-time simulations and overcomes the spurious effect of numerical diffusion. Consistency of the scheme was proved for free space and with boundary conditions. Further, the scheme was proved as an unconditionally stable scheme using Von-Neumann analysis (see [53] for proof). Based on those characteristics/proofs of the scheme, the hyperbolic nature of the equation is preserved.

Upon multiplying the expression $\Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)\psi(x)$ with a test function $P(x)$ and integrate it from a to b,

$$\begin{aligned}
&\int_a^b \Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)\psi(x)P(x)\mathrm{d}x \\
&= \int_a^b \psi(x)\Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)P(x)\mathrm{d}x - \frac{1}{\alpha^2}\Big(P(x)\partial_x\psi(x)\Big|_a^b - \psi(x)\partial_x P(x)\Big|_a^b\Big)
\end{aligned} \tag{2.9}$$

Now we have the Green's formula,

$$\int_a^b \Big[\psi(x)\Big(1-\frac{1}{\alpha^2}\partial_{xx}\Big)P(x)-\Big(1-\frac{1}{\alpha^2}\partial_{xx}\Big)\psi(x)P(x)\Big]\mathrm{d}x=\frac{1}{\alpha^2}\Big(P(x)\partial_x\psi(x)\Big|_a^b-\psi(x)\partial_xP(x)\Big|_a^b\Big) \tag{2.10}$$

Suppose the function P is a free space Green's function

$$\Big(1-\frac{1}{\alpha^2}\partial_{xx}\Big)P(x,x')=\delta(x-x') \tag{2.11}$$

$$\begin{aligned}\int_a^b \psi(x)\delta(x-x')\mathrm{d}x-\int_a^b s(x)P(x,x')\mathrm{d}x&=\frac{1}{\alpha^2}\Big(P(x,x')\partial_x\psi(x,x')\Big|_a^b-\psi(x)\partial_xP(x,x')\Big|_a^b\Big)\\ \psi(x')-\int_a^b s(x)P(x,x')\mathrm{d}x&=\frac{1}{\alpha^2}\Big(P(x,x')\partial_x\psi(x)\Big|_a^b-\psi(x)\partial_xP(x,x')\Big|_a^b\Big)\end{aligned} \tag{2.12}$$

Hence,

$$\psi(x')=\int_a^b s(x)P(x,x')\mathrm{d}x+\frac{1}{\alpha^2}\Big(P(x,x')\partial_x\psi(x)\Big|_a^b-\psi(x)\partial_xP(x,x')\Big|_a^b\Big) \tag{2.13}$$

This ends up with the particular solution and boundary correction which can be obtained by the boundary condition we chose (Dirichlet, Neumann, periodic, outflow etc.). Here we chose to use the free space Greens function and enforce the boundary conditions on $u(a,t)$ and $u(b,t)$ through expressions we derive for the unknown terms. Let's apply the homogeneous Dirichlet boundary condition (2.8),

$$\psi(x')=\int_a^b s(x)P(x,x')\mathrm{d}x+\frac{1}{\alpha^2}\Big(P(b,x')\partial_x\psi(x)\Big|_{x=b}-P(a,x')\partial_x\psi(x)\Big|_{x=a}\Big) \tag{2.14}$$

Now we have two unknowns $\partial_x\psi(x)\Big|_{x=a}$ and $\partial_x\psi(x)\Big|_{x=b}$ and they can be obtained by solving the systems of linear equations that are derived by the limits on Equation (2.14) to $x' \to a$ and $x' \to b$,

$$\psi(a) = \int_a^b s(x)P(x,a)\mathrm{d}x + \frac{1}{\alpha^2}\Big(P(b,a)\partial_x\psi(x)\Big|_{x=b} - P(a,a)\partial_x\psi(x)\Big|_{x=a}\Big) \tag{2.15}$$

$$\psi(b) = \int_a^b s(x)P(x,b)\mathrm{d}x + \frac{1}{\alpha^2}\Big(P(b,b)\partial_x\psi(x)\Big|_{x=b} - P(a,b)\partial_x\psi(x)\Big|_{x=a}\Big) \tag{2.16}$$

Since here, we are imposing homogeneous boundary condition, $\psi(a) = 0$ and $\psi(b) = 0$

$$\partial_x\psi(x)\Big|_{x=a} = w_\alpha\Big(-P(a,b)\int_a^b s(x)P(x,a)\mathrm{d}x + P(a,a)\int_a^b s(x)P(x,b)\mathrm{d}x\Big) \tag{2.17}$$

$$\partial_x\psi(x)\Big|_{x=b} = w_\alpha\Big(-P(b,b)\int_a^b s(x)P(x,a)\mathrm{d}x + P(b,a)\int_a^b s(x)P(x,b)\mathrm{d}x\Big) \tag{2.18}$$

where

$$w_\alpha = \frac{\alpha^2}{P(a,a)P(b,b) - P(a,b)P(b,a)}$$

Now we compute the exact free space Green's function to substitute the test function P,

$$\Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)G(x,x') = \delta(x - x') \tag{2.19}$$

subject to $G(x,x') = 0$ when $x \to -\infty$ and $x \to \infty$. We can show that the Green's function satisfies the continuity condition $G\Big|_{x'-}^{x'+} = 0$ and jump conditions $\partial_x G\Big|_{x'-}^{x'+} = -\alpha^2$. Consider the homogeneous solution ϕ, $\mathcal{L}\phi = 0$ and decompose it to two solutions ϕ_1 and ϕ_2 for $x > x'$ and $x < x'$. So they also satisfy $\mathcal{L}\phi_1 = 0$ and $\mathcal{L}\phi_2 = 0$ and ϕ_1 and ϕ_2 can be

written in the form of,

$$\phi_1 = c_{11}sinh(\alpha x) + c_{12}cosh(\alpha x) \tag{2.20}$$

$$\phi_2 = c_{21}sinh(\alpha x) + c_{22}cosh(\alpha x) \tag{2.21}$$

Upon applying the continuity and jump conditions ,

$$c_{11}sinh(\alpha x') + c_{12}cosh(\alpha x') - c_{21}sinh(\alpha x) - c_{22}cosh(\alpha x') = 0 \tag{2.22}$$

$$c_{11}cosh(\alpha x') - c_{12}sinh(\alpha x') + -c_{21}cosh(\alpha x) + c_{22}sinh(\alpha x') = -\alpha \tag{2.23}$$

Let's consider $x' = 0$, then we get, $c_{12} = c_{22}$ and $c_{21} = c_{11} + \alpha$. Now we write the solutions ϕ_1 and ϕ_2 in the exponential form,

$$\phi_1 = c_{11}\Big(\frac{e^{\alpha x} - e^{-\alpha x}}{2}\Big) + c_{12}\Big(\frac{e^{\alpha x} + e^{-\alpha x}}{2}\Big) \tag{2.24}$$

$$\phi_2 = (c_{11} + \alpha)\Big(\frac{e^{\alpha x} - e^{-\alpha x}}{2}\Big) + c_{12}\Big(\frac{e^{\alpha x} + e^{-\alpha x}}{2}\Big) \tag{2.25}$$

For $x > x'$, $\lim_{x' \to \infty} \phi_1 = 0$, we choose $c_{11} = -c_{12}$ such that $e^{\alpha x}$ vanishes, so,

$$\phi_1 = -c_{11}e^{-\alpha x}$$

Similarly, for $x < x'$, $\lim_{x' \to -\infty} \phi_2 = 0$, we choose $c_{11} = -\alpha/2$ such that $e^{-\alpha x}$ vanishes, so,

$$\phi_2 = \frac{\alpha}{2}e^{\alpha x}$$

Hence, the Green's function $G(x, x') = \frac{\alpha}{2}e^{-\alpha|x-x'|}$, and therefore the inverted modified Helmholtz operator can be written as,

$$\mathcal{L}_x^{-1}[u] = \underbrace{I_x[u]}_{\text{Particular solution}} + \underbrace{Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}}_{\text{Homogeneous solution}}. \tag{2.26}$$

where,

$$I_x[u] := \frac{\alpha}{2} \int_a^b e^{-\alpha|x-y|} u(y) \mathrm{d}y, \quad x \in [a,b], \tag{2.27}$$

and

$$A = -\frac{\psi(a)}{2} - \frac{1}{2\alpha} \partial_x \psi(x)\Big|_{x=a}, \qquad B = -\frac{\psi(b)}{2} + \frac{1}{2\alpha} \partial_x \psi(x)\Big|_{x=b} \tag{2.28}$$

A and B also can be written in integral form as follows,

$$A = \frac{\alpha}{2} \int_{-\infty}^a e^{-\alpha(a-y)} u(y) dy, \qquad B = \frac{\alpha}{2} \int_b^\infty e^{-\alpha(y-b)} u(y) dy. \tag{2.29}$$

The coefficients A and B of the homogeneous solution are determined by applying boundary conditions. Now, equation (2.6) can be written using the inverted Helmholtz operator, $\mathcal{L}^{-1}$ as follows,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 u^n + \beta^2 \mathcal{L}_x^{-1}[u^n] + \beta^2 \mathcal{L}_x^{-1}\Big[\frac{1}{\alpha^2} S^n\Big]. \tag{2.30}$$

The definition can additionally be modified to include boundary corrections on a finite domain (see [15]).

We also introduce a new operator $\mathcal{D}$ related to equation (2.26) that we are going to use frequently in higher-order and higher dimensional solutions.

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}_x[u^n] + \beta^2 \mathcal{L}_x^{-1}\Big[\frac{1}{\alpha^2} S^n\Big], \quad \mathcal{D}_x[u] := u - \mathcal{L}_x^{-1}[u]. \tag{2.31}$$

The computational cost of evaluating $I_x[u]$ in equation (2.26) is typically $\mathcal{O}(N^2)$; however it can be computed efficiently with second order accuracy at $\mathcal{O}(N)$ cost using the fast convolution method detailed next.

2.2.2 Fast Convolution Algorithm

We are going to see how we can compute the particular solution using a fast convolution algorithm. The integration for the particular solution, $I[u](x)$ can be performed by decomposing it into left ($I^L[u](x)$) and right ($I^R[u](x)$)) oriented integrals as shown in the Figure 2.1

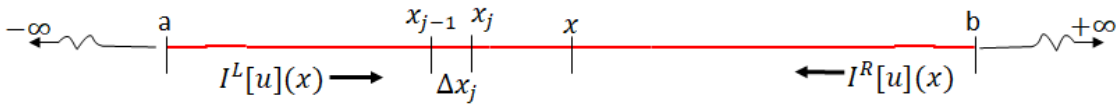


Figure 2.1: Fast convolution line integration which decomposes left $I^L[u](x)$ and right $I^R[u](x)$) oriented integrals between the domain a and b with the spatial step size, $\Delta x_j (= x_j - x_{j-1})$.

We begin by splitting the integration at $y = x$, and integrating from a to x and x to b; i.e.,

$$I[u](x) = I^L[u](x) + I^R[u](x) \tag{2.32}$$

$$I^L[u](x) = \frac{\alpha}{2}\int_a^x u(y)e^{-\alpha(x-y)}\mathrm{d}y, \quad I^R[u](x) = \frac{\alpha}{2}\int_x^b u(y)e^{-\alpha(y-x)}\mathrm{d}y,$$

so that both integrands decay exponentially away from x. Additionally, they satisfy exponential recurrence relations, which means that

$$\begin{aligned} I^L[u](x_j) &= e^{-\alpha\Delta x_j} I^L[u](x_{j-1}) + J^L[u](x_j) \quad j = 1 \quad to \quad N \\ J^L[u](x_j) &= \frac{\alpha}{2}\int_0^{\Delta x_j} e^{-\alpha y} u(x-y)\mathrm{d}y, \end{aligned} \tag{2.33}$$

$$I^R[u](x_j) = e^{-\alpha \Delta x_j} I^R[u](x_{j+1}) + J^R[u](x), \quad j = N-1 \quad to \quad 0$$
$$J^R[u](x_j) = \frac{\alpha}{2} \int_0^{\Delta x_j} e^{-\alpha y} u(x+y) \mathrm{d}y. \tag{2.34}$$

where the $\Delta x_j = x_j - x_{j-1}$ which is the spatial grid size. The recursive representation and the localization reduces the computational complexity. In particular, using a second order polynomial interpolant in place of u in the integral allows us to compute $I[u](x)$ in $\mathcal{O}(N)$ time.

The local integrals ((2.33) and (2.34)) may be evaluated using quadrature or analytically. For example, we can derive a second order quadrature using compact Simpson's rule and Lemma 2.1,

$$\begin{aligned} J_j^L &\approx P u_j + Q u_{j-1} + R\left(u_{j+1} - 2u_j + u_{j-1}\right), \\ J_j^R &\approx P u_j + Q u_{j+1} + R\left(u_{j+1} - 2u_j + u_{j-1}\right), \end{aligned} \tag{2.35}$$

with quadrature weights,

$$P = 1 - \frac{1-d}{v}, \quad Q = -d + \frac{1-d}{v}, \quad R = \frac{1-d}{v^2} - \frac{1+d}{2v},$$

where $v = \alpha \Delta x, d = e^{-v}$, and u_j denotes the solution at grid point x_j

This method can be summarized as follows (Algorithm 1),

2.2.3 Boundary Condition

We can determine the Homogeneous coefficients (A, B) from the imposed boundary conditions and the end/boundary point values of the particular solution ($I_x(x_{start})$ and $I_x(x_{end})$ where x_{start}, and x_{end} are two boundary points). Let us work with several common boundary conditions in 1D first. Then these can be expandable to higher dimension.

Algorithm 1 Fast Convolution

u_n - array of u values at time t_n,
ν - array of weighted nodes along the specific direction,
expweight - array of exponentially weighted nodes along the specific direction,
p_s - order of accuracy in space
for $j = 1$ to n **do**
    Compute $J_L(j+1)$ and $J_R(j)$ via quadrature (see equation (2.35) for second order quadrature) or analytical integration using *expweight* and p_s
end for
$d = e^{-\nu}$
for $j = 1$ to n **do**
    $I_L(j+1) = d(j)I_L(j) + J_L(j+1)$
    $I_R(n-j+1) = d(n-j+2)I_R(n-j+2) + J_L(n-j+1)$
end for
for $j = 1$ to n **do**
    $I = \frac{1}{2}\big(I_L(j) + I_R(j)\big)$
end for

2.2.3.1 Dirichlet Boundary Condition

Let us begin with Dirichlet boundary conditions. Evaluating the semi-discrete solution defined in equation (2.30) at $x = a$ and b, we get.

$$U_L(t_{n+1}) - 2U_L(t_n) + U_L(t_{n-1}) = -\beta^2 U_L(t_n) + \beta^2 \left(I\left[u^{n+1} + \frac{1}{\alpha^2} S^n\right](a) + A + Be^{-\alpha(b-a)} \right), \tag{2.36}$$

$$U_R(t_{n+1}) - 2U_R(t_n) + U_R(t_{n-1}) = -\beta^2 U_R(t_n) + \beta^2 \Big(I\Big[u^{n+1} + \frac{1}{\alpha^2} S^n\Big](b) + Ae^{-\alpha(b-a)} + B \Big), \tag{2.37}$$

We can rearrange the linear system by unknown and known values,

$$A + \mu B = -w_a^D,$$
$$\mu A + B = -w_b^D.$$

Solving the linear system for the unknowns A and B gives,

$$A = \frac{(w_a^D - \mu w_b^D)}{(\mu^2 - 1)}, \quad B = \frac{(w_b^D - \mu w_a^D)}{(\mu^2 - 1)}, \tag{2.38}$$

where,

$$w_a^D = I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - \frac{1}{\beta^2}\Big(U_L(t_{n+1}) + (\beta^2-2)U_L(t_n) + U_L(t_{n-1})\Big), \tag{2.39}$$

$$w_b^D = I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - \frac{1}{\beta^2}\Big(U_R(t_{n+1}) + (\beta^2-2)U_R(t_n) + U_R(t_{n-1})\Big), \tag{2.40}$$

and $\mu = e^{-\alpha(b-a)}$.

2.2.3.2 Neumann Boundary Condition

For Neumann boundary condition, we obtain the identities, $I^{'}(a) = \alpha I(a), \quad I^{'}(b) = -\alpha I(b)$ using the characteristics of the integral solution (equation (2.27)); specifically, all dependence on x is on the Greens function which is a simple exponential function. Differentiating the semi-discrete solution (2.30), and applying the Neumann boundary conditions at $x = a$ and b yields,

$$V_L(t_{n+1}) - 2V_L(t_n) + V_L(t_{n-1}) = -\beta^2 V_L(t_n) + \beta^2\Big(I\Big[u^{n+1} + \frac{1}{\alpha^2}S^n\Big](a) + A + Be^{-\alpha(b-a)}\Big), \tag{2.41}$$

and

$$V_R(t_{n+1}) - 2V_R(t_n) + V_R(t_{n-1}) = -\beta^2 V_R(t_n) + \beta^2\Big(I\Big[u^{n+1} + \frac{1}{\alpha^2}S^n\Big](b) + Ae^{-\alpha(b-a)} + B\Big). \tag{2.42}$$

We can rearrange the linear system by unknown and known values,

$$A - \mu B = w_a^N,$$
$$-\mu A + B = w_b^N.$$

Solving the linear system for the unknowns A and B gives,

$$A = \frac{(w_a^N + \mu w_b^N)}{(\mu^2 - 1)}, \quad B = \frac{(w_b^N + \mu w_a^N)}{(\mu^2 - 1)}, \tag{2.43}$$

where,

$$w_a^N = I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - \frac{1}{\beta^2}\Big(V_L(t_{n+1}) + (\beta^2-2)V_L(t_n) + V_L(t_{n-1})\Big), \tag{2.44}$$

$$w_b^N = I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - \frac{1}{\beta^2}\Big(V_R(t_{n+1}) + (\beta^2-2)V_R(t_n) + V_R(t_{n-1})\Big). \tag{2.45}$$

2.2.3.3 Periodic Boundary Condition

By applying the periodic boundary conditions on the semi-discrete solution (2.30), we obtain,

$$I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) + A + \mu B = I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) + \mu A + B, \tag{2.46}$$

and

$$\alpha\Big(I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - A + \mu B\alpha\Big) = \alpha\Big(-I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - \mu A + B\alpha\Big). \tag{2.47}$$

We can rearrange the linear system to obtain,

$$\begin{aligned}(\mu-1)A + (1-\mu)B &= I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) - I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](b),\\ (1-\mu)A + (1-\mu)B &= I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a) + I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](b).\end{aligned}$$

Upon solving these linear equations we get,

$$A = \frac{I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](b)}{(1-\mu)}, \quad B = \frac{I\Big[u^n + \frac{1}{\alpha^2}S^n\Big](a)}{(1-\mu)}. \tag{2.48}$$

2.2.3.4 Outflow Boundary Condition

We can extend the definition of the outflow boundary conditions to the domains exterior to $[a, b]$. Assuming that our initial condition has compact support in $[a, b]$, after time $t = t_n$, the domain of dependence of $u^n(x)$ extends to $[a - ct_n, b + ct_n]$, since the propagation speed

is c. Now the solution of $\mathcal{L}^{-1}$ ((2.26)) can be written as.

$$\mathcal{L}^{-1}[u^n(x)] = \frac{\alpha}{2}\int_{a-ct_n}^{b+ct_n} e^{-\alpha|x-y|}u^n(y)dy = I[u(x)] + A^n e^{-\alpha(x-a)} + B^n e^{-\alpha(b-x)}.$$

where

$$A^n := \frac{\alpha}{2}\int_{a-ct_n}^{a} e^{-\alpha(a-y)}u^n(y)dy, \tag{2.49}$$

$$B^n := \frac{\alpha}{2}\int_{b}^{b+ct_n} e^{-\alpha(y-b)}u^n(y)dy. \tag{2.50}$$

Now we turn these spatial integrals into time integrals which exist at precisely the end points $x = a$ and b. First we consider outflow along the right boundary, $x > b$ and assume this region contains only right traveling waves, $u(x,t) = u(x-ct)$. By tracing backward along a characteristic ray we find,

$$u(b+y,t) = u(b, t-y/c), y > 0.$$

Hence,

$$B^n = \frac{\alpha}{2}\int_0^{ct_n} e^{-\alpha(y)}u^n(b+y,t_n)dy = \frac{\alpha c}{2}\int_0^{t_n} e^{-\alpha(cs)}u^n(b,t_n-s)ds.$$

Now, we can impose outflow boundary conditions using the behavior of u at $x = b$. A temporal recurrence relation due to the exponential will be,

$$\begin{aligned} B^n &= \frac{\alpha c}{2}\int_0^{\Delta t} e^{-\alpha cs}u(b,t_n-s)ds + e^{-\alpha c\Delta t}\int_0^{t_{n-1}} e^{-\alpha cs}u(b,t_{n-1}-s)ds, \\ &= \frac{\beta}{2}\int_0^1 e^{-\beta z}u(b,t_n-z\Delta t)dz + e^{-\beta}B^{n-1}. \end{aligned} \tag{2.51}$$

where $\beta = \alpha c \Delta t$. Therefore, the boundary coefficient B^n can be computed locally in both time and space. We choose a quadratic interpolant in terms of u to have second order accuracy.

$$u(b, t_n - z\Delta t) = u^n(b) - \frac{z}{2}\Big(u^{n+1}(b) - u^{n-1}(b)\Big) + \frac{z^2}{2}\Big(u^{n+1}(b) - 2u^n(b) - u^{n-1}(b)\Big). \quad (2.52)$$

Using the Equation (2.52) would give the second order approximation for outflow. To compute the values that come from this approximation, we will make use of the following lemma which comes from integration by parts (see [15] for proof),

Lemma 2.1. *For integers $m \geq 0$ and real $v > 0$,*

$$E_m := v \int_0^1 \frac{z^m}{m!} e^{-vz} dz = \frac{1}{v^m}\Big(1 - e^{-v} P_m(v)\Big)$$

where $P_m(v) = \sum_{l=0}^{m} \frac{v^l}{l!}$ is the Taylor series expansion of order m of e^v.

Using this lemma and integrating the expression (2.52) analytically, we can get,

$$B^n = e^{-\beta} B^{n-1} + \gamma_0 u^{n+1}(b) + \gamma_1 u^n(b) + \gamma_2 u^{n-1}(b), \quad (2.53)$$

where,

$$\begin{aligned} \gamma_0 &= E_2(\beta) - \frac{1}{2} E_1(\beta), \\ \gamma_1 &= -2E_2(\beta) + E_0(\beta), \quad \textit{and} \\ \gamma_2 &= E_2(\beta) + \frac{1}{2} E_1(\beta). \end{aligned} \quad (2.54)$$

There are two unknowns in this equation (2.53), B^n and u^{n+1}, so we need one more equation,

that can be derived by evaluating equation (2.30) at $x = b$,

$$u^{n+1}(b) - 2u^n(b) + u^{n-1}(b) = -\beta^2\Big(u^n(b) - \big(I(b) + \mu A^n + B^n\big)\Big). \tag{2.55}$$

Upon solving these two linear equations (2.53) and (2.55) we can obtain,

$$-\Gamma_0\mu A^n + (1-\Gamma_0)B^n = e^{-\beta}B^{n-1} + \Gamma_0 I(b) + \Gamma_1 u^n(b) + \Gamma_2 u^{n-1}(b), \tag{2.56}$$

where

$$\Gamma_0 = \frac{\beta^2}{2}\gamma_0, \quad \Gamma_1 = \gamma_1 - \gamma_0(\beta^2 - 2), \quad \Gamma_2 = \gamma_2 - \gamma_0.$$

Similarly, by considering outflow at the other end $x < a$, we get

$$(1-\Gamma_0)A^n - \Gamma_0\mu B^n = e^{-\beta}A^{n-1} + \Gamma_0 I(a) + \Gamma_1 u^n(a) + \Gamma_2 u^{n-1}(a). \tag{2.57}$$

Solving the linear system of equations (2.56) and (2.57) gives,

$$A^n = \frac{(1-\Gamma_0)w_a^{Out} + \mu\Gamma_0 w_b^{Out}}{(1-\Gamma_0)^2 - (\mu\Gamma_0)^2}, \quad B^n = \frac{(1-\Gamma_0)w_b^{Out} + \mu\Gamma_0 w_a^{Out}}{(1-\Gamma_0)^2 - (\mu\Gamma_0)^2}, \tag{2.58}$$

where

$$w_a^{Out} = e^{-\beta}A^{n-1} + \Gamma_0 I(a) + \Gamma_1 u^n(a) + \Gamma_2 u^{n-1}(a),$$
$$w_b^{Out} = e^{-\beta}B^{n-1} + \Gamma_0 I(b) + \Gamma_1 u^n(b) + \Gamma_2 u^{n-1}(b)$$

According to the solution of the coefficients A^n and B^n, we realize that we need to know their values at the previous time step(A^{n-1} and B^{n-1}). For utilization purposes, we set $A^0 = B^0 = 0$.

2.3 Multi-Dimensional Implicit Wave Equation Solver using ADI Scheme

The extension to multi-dimension is formed using an alternating direction implicit (ADI) scheme, and each line is solved independently.

The multi-dimensional wave equation can be represented using the initial boundary value problem,

$$\frac{1}{c^2}\frac{\partial^2 u(\mathbf{x},t)}{\partial t^2} - \nabla^2 u(\mathbf{x},t) = S(\mathbf{x},t), \quad \mathbf{x}\in\Omega, \quad t>0, \tag{2.59}$$

$$u(\mathbf{x},0) = f(\mathbf{x}), \quad \mathbf{x}\in\Omega,$$
$$\partial_t u(\mathbf{x},0) = g(\mathbf{x}), \quad \mathbf{x}\in\Omega.$$

with consistent boundary conditions, $u(\mathbf{x},t) = h(\mathbf{x},t), \mathbf{x}\in\partial\Omega, t>0$, source term $S(\mathbf{x},t)$, and wave speed c

The multi-dimensional implicit scheme uses an ADI scheme [15], and each ADI line is solved independently. Consider a scheme for three spatial dimensions,

$$\begin{aligned}
1-\frac{1}{\alpha^2}\nabla^2 &= 1-\frac{1}{\alpha^2}\Big(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\Big)\\
&= \Big(1-\frac{1}{\alpha^2}\frac{\partial^2}{\partial x^2}\Big)\Big(1-\frac{1}{\alpha^2}\frac{\partial^2}{\partial y^2}\Big)\Big(1-\frac{1}{\alpha^2}\frac{\partial^2}{\partial z^2}\Big)\\
&\quad+\frac{1}{\alpha^4}\Big(\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial x^2\partial z^2}+\frac{\partial^4}{\partial y^2\partial z^2}\Big)-\frac{1}{\alpha^6}\frac{\partial^6}{\partial x^2\partial y^2\partial z^2}
\end{aligned}$$

Hence,

$$1-\frac{1}{\alpha^2}\nabla^2 = \mathcal{L}_x\mathcal{L}_y\mathcal{L}_z + \mathcal{O}\big((c\Delta t)^4\big) \tag{2.60}$$

where $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$ are one dimensional univariate modified Helmholtz operators [15]

applied in the indicated spatial variable. Now the semi-discrete equation,

$$\mathcal{L}_x\mathcal{L}_y\mathcal{L}_z\Big[\beta^2 u^n + u^{n+1} - 2u^n + u^{n-1}\Big] = \beta^2 u^n + \frac{\beta^2}{\alpha^2} S^n(\mathbf{x}, t) \tag{2.61}$$

Upon inverting the modified Helmholtz operators $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$, we have a semi-discrete solution,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}_{xyz}[u^n] + \beta^2 \mathcal{L}_z^{-1}\mathcal{L}_y^{-1}\mathcal{L}_x^{-1}\Big[\frac{1}{\alpha^2} S^n\Big](x, y, z) \tag{2.62}$$

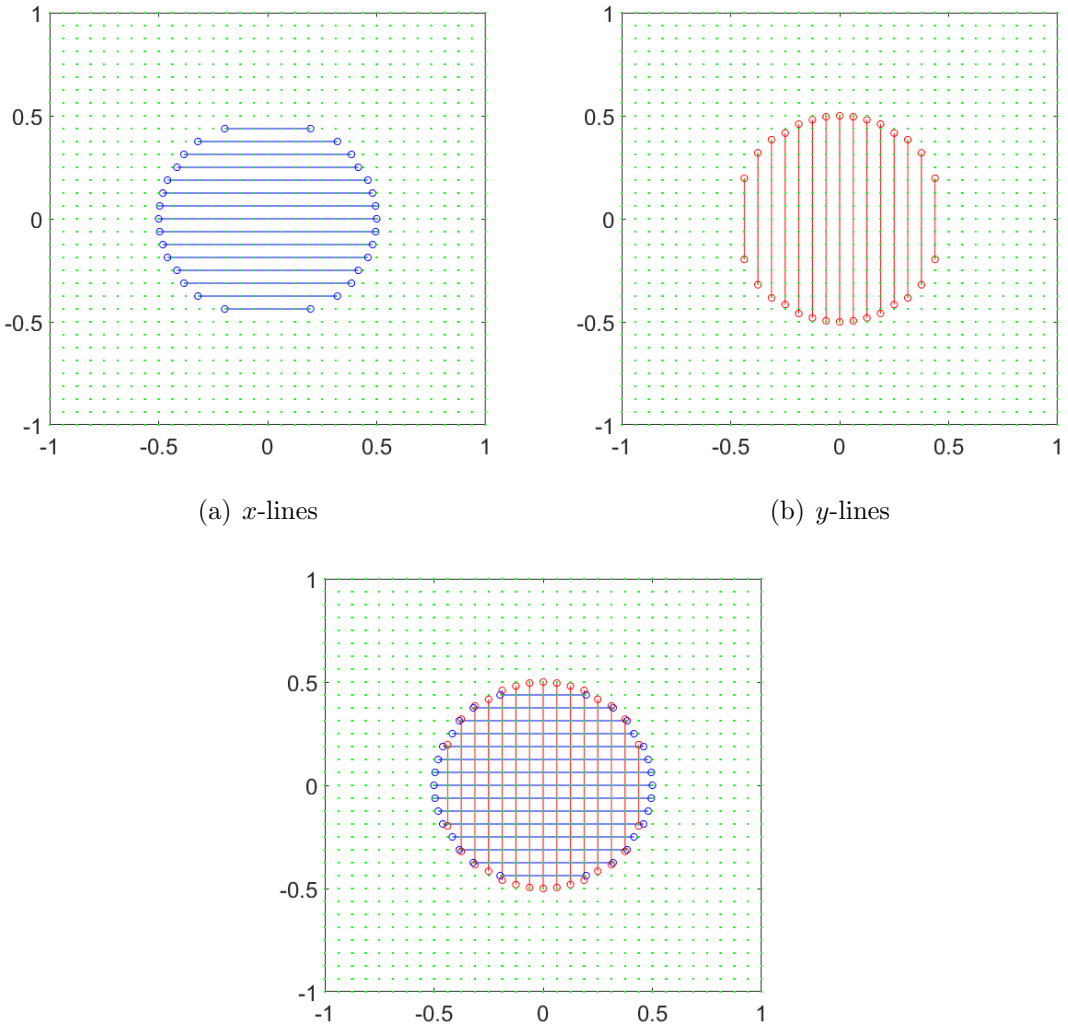
where the three dimensional operator is, $\mathcal{D}_{xyz}[u] := u - \mathcal{L}_z^{-1}\mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[u]$,

while in two dimensions it is

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}_{xy}[u^n] + \beta^2 \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}\Big[\frac{1}{\alpha^2} S^n\Big](x, y), \quad \mathcal{D}_{xy}[u] := u - \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[u]. \tag{2.63}$$

Inversion of the operator $\mathcal{L}_\gamma^{-1}$ is performed sequentially leading to the usual x, y and z "sweeps" of the ADI algorithm that is represented by a combination of the operator $\mathcal{L}_\gamma^{-1}$. Because the inverse operators are defined along lines, it is quite natural to discretize 2D and 3D regions along Cartesian lines. However, the endpoints of these lines are not restricted to residing at mesh points and can always be chosen to lie on the boundary $\partial\Omega$. For example, the x, y, boundary points for the x-sweep and y-sweep for a circle are shown in Figure 2.2. We first perform x-sweeps along with x-lines with actual boundary ends and obtain the intermediate solution and then apply y-sweeps over the intermediate solution, along with y-lines with actual boundary ends. The xy-sweeps form a complete circular boundary (Figure 2.2 - (c)). Similarly, we can perform y-sweeps first then x-sweeps next and take an average to obtain the solution

The general approach for implementation of the 3D ADI scheme follows the steps shown below: Assume the number of x, y, and z lines are n_x, n_y, and n_z respectively.
At each time step,



(a) x-lines

(b) y-lines

(c) xy-lines

Figure 2.2: (a) x-, (b) y- and (c) xy lines with exact circular boundary points on the mesh lines that are used for the ADI x (a) and y (b) sweeps.

1. Perform the x-sweep

   Operate $\mathcal{L}^{-1}$ on $u(x, y, z)$ along x, and store the result into a temporary variable $w_x(x, y_i, z_k)$ for $1 \leq k \leq n_z$ and $1 \leq i \leq n_y$. The boundary conditions are imposed at $x = x_{a(ik)}$ and $x_{b(ik)}$.

2. Perform the y-sweep

   Operate $\mathcal{L}^{-1}$ on $w_x(x, y, z)$ along y, and store the result into a temporary variable $w_{yx}(x_j, y, z_k)$ for $1 \leq k \leq n_z$ and $1 \leq j \leq n_x$. The boundary conditions are imposed

at $y = y_{a(jk)}$ and $y_{b(jk)}$.

3. Perform the z-sweep

   For $1 \leq i \leq n_y$ and $1 \leq j \leq n_x$ using $w_{yx}(x, y, z)$, solve for the equation $u^{n+1} = 2u^n + u^{n-1} - \beta^2 \mathcal{D}_{xyz}$, where $\mathcal{D}_{xyz} = u_n - \mathcal{L}_z^{-1}[w_{yx}]$. The boundary conditions are now applied at $z = z_{a(ij)}$ and $z_{b(ij)}$.

In order to improve the accuracy of the ADI solve, the inversion of the x, y and z Helmholtz operators is symmetrized, by averaging the results of x—y —z, y —z —x, and z —x —y solves. That means we need to apply the operators $\mathcal{D}_{xyz}$, $\mathcal{D}_{yzx}$, and $\mathcal{D}_{zyx}$ and then take the average.

We give the detailed algorithm for the two-dimensional implicit wave equation solver in Algorithm 2 and relevant supporting functions are given in Algorithms 3 and 4. Algorithm 3 computes solution u at time t_{n+1} using our two dimensional implicit solver that calls the function LINV given in Algorithm 4 to apply the operator $\mathcal{L}^{-1}$ to perform x and y sweeps. LINV may be written in two functions for x and y sweeps separately. The function *applyBC* called in Algorithm 4 is used to find the homogeneous coefficients A^n and B^n using appropriate boundary conditions. Those are retrieved as the first and second elements of the vector H respectively. Since we have to use the value of the coefficients at the previous time step (t_{n-1}), A^{n-1} and B^{n-1} to compute A^n and B^n for outflow boundary condition, these algorithms, however, need to be modified with minor changes by passing reference type parameters to the function *applyBC*. The parameter should be defined within the main Algorithm 2 and be called via *computeU*. Further, although Algorithm 2 would typically be used to deal with problems with rectangular boundaries, it can be extended to adopt problems with randomly placed boundary points in each grid line such as when dealing with a circular boundary using the embedded boundary method. Here, we need to use matrices x and y instead of vectors and define additional vectors to keep boundary points of each horizontal and vertical grid line.

Algorithm 2 Two Dimensional Solver

x_A, x_B and y_A , y_B - boundary points along x and y respectively,
n_x, n_y - number of grid points along x and y respectively,
Δt - time step size,
T - total time,
c - wave speed,
β - averaging parameter,
p_s -order of accuracy in space.
$x = grid\_vector(x_A, x_B, n_x)$
$y = grid\_vector(y_A, y_B, n_y)$
$set\_initialvalue(u_{n-1}, t_0)$
$set\_initialvalue(u_n, t_1)$
$\alpha = \frac{\beta}{c\Delta t}$
$n_t = \frac{T}{\Delta t}$
$\mu_x = e^{-\alpha(x(n)-x(0))}$
$\mu_y = e^{-\alpha(y(n)-y(0))}$
$expA_x = e^{-\alpha(x-x(0))}$
$expB_x = e^{-\alpha(x(n)-x)}$
$expA_y = e^{-\alpha(y-y(0))}$
$expB_y = e^{-\alpha(y(n)-y)}$
$\nu_x = \alpha(x(2 : n_x) - x(1 : n_x - 1))$
$\nu_y = \alpha(y(2 : n_y) - y(1 : n_y - 1))$
$expweight_x = exp\_weights(\nu_x, p)$
$expweight_y = exp\_weights(\nu_y, p$
for $t_n = 1$ to n_t **do**
    $u_{n+1} = compute\_u(u_{n-1}, u_n, n_x, n_y, \beta, \mu_x, \mu_y, x_A, x_B, y_A, y_B, expA_x, expB_x, expA_y, expB_y, \nu_x, \nu_y, expweight_x, expweight_y, p_s)$
    $u_{n-1} = u_n$
    $u_n = u_{n+1}$
end for

2.4 Arbitrary Temporal Order Accuracy

So far, we have derived multi-dimensional semi-discretizations of the wave equation with second-order accuracy in time. In order to take large time steps in problems, we need higher-order accuracy in time. While there is no stability restriction placed on our A-stable scheme, considerations of accuracy present themselves when the CFL number becomes large. Indeed, when large time steps are taken, the anisotropies introduced by the dimensional splitting are very pronounced.

Algorithm 3 *compute_u* : Compute u at time t_{n+1}

u_{n-1}, u_n - array of u values at time t_{n-1} and t_n respectively,
n_x, n_y - number of grid points along x and y respectively,
β - averaging parameter,
$\mu_x = e^{-\alpha(x(n)-x(0))}$,
$\mu_y = e^{-\alpha(y(n)-y(0))}$,
x_A, x_B and y_A , y_B - boundary points along x and y respectively,
$expA_x$, $expB_x$ - exponential vector of grid distance from left boundary and right boundary respectively along x,
$expA_y$, $expB_y$ - exponential vector of grid distance from bottom boundary and up boundary respectively along y,
ν_x, ν_y - array of weighted nodes along x and y respectively,
$expweight_x$, $expweight_y$ - array of exponentially weighted nodes along x and y respectively,
p_s -order of accuracy in space
$linv\_x = linv(u_1, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
$linv\_yx = linv(Linv_x, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p, 1)$
$D_{xy} = u_n - linv\_yx$
$u_{n+1} = 2u_n + u_{n-1} - \beta^2 D_{xy}$

Algorithm 4 *linv* : $\mathcal{L}^{-1}$

u_n - array of u values at time t_n,
n - number of lines that need to be sweep,
$bdry_A$, $bdry_B$ - boundary points along the specific line,
expA expB- exponential vector of grid distance from left boundary and right boundary respectively along the line,
ν - array of weighted nodes along the line,
expweight - array of exponentially weighted nodes along the line,
p -order of accuracy in space,
dir - determine which sweep is going to do, along x or y determined by 0 or 1 respectively
if dir == 0 **then**
  for $i = 1$ to n **do**
    $I = fastconvolution(u(i,:), \nu, expweight, p_s)$
    $H = apply\_bc(u(bdry_A), u(bdry_B), I(bdry_A), I(bdry_B), \beta, bdry_A, bdry_B, \mu)$
    $Linv(i,:) = I + H(1)expA + H(2)expB$
  end for
else
  for $j = 1$ to n **do**
    $I = fastconvolution(u(:,j), \nu, expweight, p_s)$
    $H = apply\_bc(bdry_A), u(bdry_B), I(bdry_A), I(bdry_B), \beta, bdry_A, bdry_B, \mu)$
    $Linv(:,j) = I + H(1)expA + H(2)expB$
  end for
end if

In this section, we discuss A-stable schemes of arbitrary order using a MOL^T formulation of the wave equation, and by implicitly including higher-order derivatives. However, we construct the derivatives using a novel approach: recursive convolution. The cornerstone of our method lies in the fact that by using recursive applications of the convolution operators introduced in [15], the inversions of higher derivatives can be performed analytically. So the resulting scheme is made explicit, even at the semi-discrete level. In constructing the analytical convolution operators, we incorporate the boundary conditions directly. In this way, without additional complexity Dirichlet and periodic boundary conditions can be implemented to higher-order. Since dealing with outflow boundary conditions depend on previous time step values at the boundaries, we need to do a little more work to implement it in higher-order. We use the recursive convolution algorithm which consumes $O(N)$ operations. So, the schemes we obtain in $d = 1, 2$ and 3 dimensions will achieve accuracy $2P$ in $O(P^d N)$ operations per time step.

2.4.1 Arbitrary Order One Dimensional Scheme

A higher-order scheme can be achieved by including more spatial derivatives in the numerical scheme. We choose a Lax-Wendroff approach to increase the temporal accuracy of our second order semi-discrete solution. This approach requires us to exchange time derivatives with spatial derivatives in the Taylor expansion. Let's start with the semi-discrete wave equation and apply the Lax-Wendroff procedure. We have

$$u^{n+1} - 2u^n + u^{n-1} = 2 \sum_{m=1}^{\infty} \frac{\Delta t^{2m}}{(2m)!} \left(\partial_{tt}\right)^m u^n \tag{2.64}$$

Hence, by using the fact,

$$\left(\partial_{tt}\right)^m u = \left(c^2 \partial_{xx}\right)^m u, \quad m \geq 1.$$

$$u^{n+1} - 2u^n + u^{n-1} = 2\sum_{m=1}^{\infty} \frac{(c\Delta t)^{2m}}{(2m)!} (\partial_{xx})^m u^n$$
$$= 2\sum_{m=1}^{\infty} \frac{\beta^{2m}}{(2m)!} \left(\frac{\partial_{xx}}{\alpha^2}\right)^m u^n. \tag{2.65}$$

Differential operators $(\partial_{xx})^m$ have to be approximated properly. To do this, we can consider the convolution operator $\mathcal{D}$ from (2.31). We begin by considering the equation,

$$\mathcal{F}\left[\left(\frac{\partial_{xx}}{\alpha^2}\right)^m\right] = (-1)^m \left(\frac{k}{\alpha}\right)^{2m}$$

where $\mathcal{F}$ denotes the Fourier transform. Now, we need to find $(k/\alpha)^2$; it can be formed using $\mathcal{D}$ as follows,

$$\hat{D} := \mathcal{F}[\mathcal{D}] = 1 - \mathcal{F}\left[\mathcal{L}^{-1}\right] = 1 - \frac{1}{1 + \left(\frac{k}{\alpha}\right)^2} = \frac{\left(\frac{k}{\alpha}\right)^2}{1 + \left(\frac{k}{\alpha}\right)^2}.$$

Solving for the quantity $(k/\alpha)^2$ in terms of $\hat{D}$, gives an expression for all even derivatives as follows,

$$\left(\frac{k}{\alpha}\right)^2 = \frac{\hat{D}}{1-\hat{D}} = \sum_{p=1}^{\infty} \hat{D}^p \quad \text{and}$$
$$\left(\frac{k}{\alpha}\right)^{2m} = \left(\frac{\hat{D}}{1-\hat{D}}\right)^m = \sum_{p=m}^{\infty} \binom{p-1}{m-1} \hat{D}^p,$$

By inserting these into the Taylor expansion (2.65), we obtain

$$u^{n+1} - 2u^n + u^{n-1} = \sum_{m=1}^{\infty} (-1)^m \frac{2\beta^{2m}}{(2m)!} \sum_{p=m}^{\infty} \binom{p-1}{m-1} \mathcal{D}^p[u^n].$$

Upon reversing the order of summation, we have

$$u^{n+1} - 2u^n + u^{n-1} = \sum_{p=1}^{\infty} A_p(\beta) \mathcal{D}^p[u^n], \tag{2.66}$$

where $A_p(\beta)$ is a polynomial of coefficients in β^2,

$$A_p(\beta) = 2\sum_{m=1}^{p}(-1)^m \frac{\beta^{2m}}{(2m)!}\binom{p-1}{m-1}. \tag{2.67}$$

From this scheme (2.4.1), 1D second (identical to the original second order scheme (2.31)) and fourth order equations are

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}[u^n] \tag{2.68}$$

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}[u^n] - \left(\beta^2 - \frac{\beta^4}{12}\right)\mathcal{D}^2[u^n] \tag{2.69}$$

Algorithm 3 can be modified as follows to be able to compute the solution with higher-order accuracy in 1D.

Algorithm 5 Compute u at time t_{n+1} with high-order accuracy

u_{n-1}, u_n - array of u values at time t_{n-1} and t_n respectively,
n - number of points along the line,
β - averaging parameter,
$\mu = e^{-\alpha(x(n)-x(0))}$,
$bdryA$, $bdryB$ - boundary points along the specific line
$expA$, $expB$ - exponential vector of grid distance from left boundary and right boundary respectively along the line,
ν - array of weighted nodes along the line,
$expweight$ - array of exponentially weighted nodes along the line,
p_s -order of accuracy in space,
p_t -order of accuracy in time
$P = \frac{p_t}{2}$
$Dterms = 0$
for $k = 1$ to P **do**
    $D = u - linv(u, n, \beta, \mu, bdryA, bdryB, expA, expB, \nu, expweight, p_s)$
    $Dterms = Dterms + poly\_highorder(\beta, P)D$
    $u = D$
end for
$u_{n+1} = 2u_n - u_{n-1} + Dterms$

Temporal cost for a scheme of order P is $O(PN)$ per time step for N spatial points. Since the local truncation errors are $O((c\Delta t/\beta)^{2P+2})$, the error constant will decrease with

Algorithm 6 *poly_highorder* : Compute the coefficients for high-order accuracy

1: β - averaging parameter,
2: k - order of accuracy at current stage
3: coef = 0;
4: **for** $m = 1$ to k **do**
5: $\quad coef = coef + (-1)^m \frac{\beta^{2m}}{(2m)!}\binom{k-1}{m-1}$
6: **end for**
7: $coef = 2coef$

increasing β. However, it should be limited with an upper value to maintain stability. The stability of this scheme was proved in [15] using Von-Neumann analysis and they obtained an upper limit of β for several even orders of temporal accuracy which is summarized in Table 2.1. We can notice that β_{max} decreases with increasing P

P	1	2	3	4	5
Order	2	4	6	8	10
β_{max}	2	1.4840	1.2345	1.0795	0.9715

Table 2.1: The maximum values β_{max} for which the P^{th} order scheme remains A-stable [14].

We need to set appropriate initial conditions to achieve the expected order. Suppose we would like to get P^{th} order accuracy, the initial value should be computed to $O(\Delta t^{2P})$. For our 3-step scheme (2.66), u^0 and u^1 have to be initialized. They may be computed analytically. However, in general, we know the exact value of u^0 $(= f(x))$, and we need to approximate the value for $u^1 = u(x, \Delta t)$. To do so, we can apply the Lax-Wendroff procedure in Taylor expansion, we obtain

$$
\begin{aligned}
u^1 &= \sum_{m=0}^{\infty} \frac{\Delta t^m}{m!} \partial_t^m u^0 \\
&= \sum_{m=0}^{\infty} \left(\frac{\Delta t^{2m}}{(2m)!} \partial_t^{2m} u^0 + \frac{\Delta t^{2m+1}}{(2m+1)!} \partial_t^{2m+1} u^0 \right) \\
&= \sum_{m=0}^{\infty} \left(\frac{(c\Delta t)^{2m}}{(2m)!} \partial_x^{2m} u^0 + \frac{(c\Delta t)^{2m+1}}{(2m+1)!} \partial_x^{2m} \frac{1}{c} \partial_t u^0 \right) \\
&= \sum_{m=0}^{\infty} \left(\frac{(c\Delta t)^{2m}}{(2m)!} \partial_x^{2m} f(x) + \frac{(c\Delta t)^{2m+1}}{(2m+1)!} \partial_x^{2m} \frac{1}{c} g(x) \right). \qquad (2.70)
\end{aligned}
$$

where $g(x) = u_t(x, 0)$. This expansion can now be truncated at $O(\Delta t^{2P})$ to have P^{th} order of accuracy.

2.4.2 Extension to Higher Dimensions for Arbitrary Order Accuracy

We derive higher-order accurate solutions for the wave equation in higher dimensions using an ADI scheme. We begin with the expansion of Equation (2.64) by introducing the Laplacian in the Lax-Wendroff procedure.

$$
u^{n+1} - 2u^n + u^{n-1} = 2 \sum_{m=1}^{\infty} \frac{\Delta t^{2m}}{(2m)!} \left(\partial_{tt}^m \right) u^n = 2 \sum_{m=1}^{\infty} \frac{\beta^{2m}}{(2m)!} \left(\frac{\nabla^2}{\alpha^2} \right)^m u^n.
$$

Before approximating higher-order powers of the Laplacian operator, consider univariate modified Helmholtz operators, and their corresponding $\mathcal{D}$ operators as given below,

$$
\mathcal{L}_\gamma := 1 - \frac{\partial_\gamma^2}{\alpha^2}, \quad \mathcal{D}_\gamma := 1 - \mathcal{L}_\gamma^{-1}, \quad \gamma = x, y, z.
$$

Since these operators satisfy identities, $\mathcal{L}_\gamma \mathcal{D}_\gamma[u] = \mathcal{L}[u] - u = -\frac{\partial_{\gamma\gamma}}{\alpha^2} u$, the Laplacian in 3D can be written by,

$$
-\frac{\nabla^2}{\alpha^2} = \mathcal{L}_x \mathcal{D}_x + \mathcal{L}_y \mathcal{D}_y + \mathcal{L}_z \mathcal{D}_z = \mathcal{L}_x \mathcal{L}_y \mathcal{L}_z [\mathcal{C}_{xyz}], \qquad (2.71)
$$

where

$$\mathcal{C}_{xyz} : \mathcal{L}_y^{-1}\mathcal{L}_z^{-1}\mathcal{D}_x + \mathcal{L}_z^{-1}\mathcal{L}_x^{-1}\mathcal{D}_y + \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}\mathcal{D}_z \tag{2.72}$$

Consider the $\mathcal{D}$ operator in three dimensions.

$$\mathcal{D}_{xyz} := 1 - \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}\mathcal{L}_z^{-1}. \tag{2.73}$$

Upon rearranging and inverting (2.73), we get an identity $\mathcal{L}_x\mathcal{L}_y\mathcal{L}_z = (1 - \mathcal{D}_{xyz})^{-1}$, Now, the Laplacian becomes,

$$-\frac{\nabla^2}{\alpha^2} = (1 - \mathcal{D}_{xyz})^{-1}\,\mathcal{C}_{xyz}.$$

By expanding $(1-\mathcal{D})^{-m}$ as a power series, we can construct even orders of the Laplacian as follows,

$$\left(\frac{\nabla^2}{\alpha^2}\right)^m = (-1)^m \mathcal{C}_{xyz}^m \sum_{p=m}^{\infty} \binom{p-1}{m-1} \mathcal{D}_{xyz}^{p-m}. \tag{2.74}$$

Upon omitting the subscripts, the semi-discrete scheme for 2D and 3D is

$$\begin{aligned} u^{n+1} - 2u^n + u^{n-1} &= \sum_{m=1}^{\infty} \frac{2\beta^{2m}}{(2m)!} \left(\frac{\nabla^2}{\alpha^2}\right)^m u^n \\ &= \sum_{m=1}^{\infty} (-1)^m \frac{2\beta^{2m}}{(2m)!} \mathcal{C}^m \sum_{p=m}^{\infty} \binom{p-1}{m-1} \mathcal{D}^{p-m}[u^n] \\ &= \sum_{p=1}^{\infty} \sum_{m=1}^{p} (-1)^m \frac{2\beta^{2m}}{(2m)!} \binom{p-1}{m-1} \mathcal{C}_{xyz}^m \mathcal{D}_{xyz}^{p-m}[u^n]. \end{aligned} \tag{2.75}$$

We give second and fourth-order schemes here,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xyz}[u^n] \tag{2.76}$$

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xyz}[u^n] - \left(\beta^2 \mathcal{D}_{xyz} - \frac{\beta^4}{12}\mathcal{C}_{xyz}\right) \mathcal{C}_{xyz}[u^n] \tag{2.77}$$

As in the 1D case, these schemes will be unconditionally stable for all Δt, and the same

range for β as shown in Table 2.1.

The algorithm for computing u in two dimensions using ADI splitting with a given order of accuracy in time is designed as below in Algorithm 7. Using this algorithm, we can retrieve the solution with accuracy $\mathcal{O}(p_t)$ in time.

Algorithm 7 *compute_u_high* :Compute u in 2D with high-order accuracy

u_{n-1}, u_n - array of u values at time t_{n-1} and t_n respectively,
n_x, n_y - number of points along x and y respectively,
β - averaging parameter,
$\mu_x = e^{-\alpha(x(n)-x(0))}$,$\mu_y = e^{-\alpha(y(n)-y(0))}$,
x_A, x_B, and y_A, y_B - boundary points in x and y directions respectively,
$expA_x$, $expB_x$ - exponential vector of grid distance from left boundary and right boundary respectively along x,
$expA_y$, $expB_y$ - exponential vector of grid distance from bottom boundary and up boundary respectively along y,
ν_x, ν_y - array of weighted nodes along x and y respectively,
$expweight_x$, $expweight_y$ - array of exponentially weighted nodes along x and y respectively,
p_s and p_t - order of accuracy in space and time respectively
$P = \frac{p_t}{2}$
$struct\_outarray(0) = compute\_c(u, n_x, n_y, \beta, \mu_x, \mu_y, x_A, x_B, y_A, y_B, expA_x, expB_x, expA_y, expB_y, \nu_x, \nu_y, expweight_x, expweight_y, p_s)$
$cd\_terms = -\beta^2 struct_outarray(0)$
for $k = 2$ to P **do**
  $struct\_inarray = struct\_outarray$
  $struct\_outarray(1) = compute\_cd(struct\_inarray(0), n_x, n_y, \beta, \mu_x, \mu_y, x_A, x_B, y_A, y_B, expA_x, expB_x, expA_y, expB_y, \nu_x, \nu_y, expweight_x, expweight_y, p_s)$
  for $i = 2$ to $k - 1$ **do**
    $struct\_outarray(i) = compute\_d(struct\_inarray(i), n_x, n_y, \beta, \mu_x, \mu_y, x_A, x_B, y_A, y_B, expA_x, expB_x, expA_y, expB_y, \nu_x, \nu_y, xpweight_x, expweight_y, p_s)$
  end for
  for $i = k$ to 1 **do**
    $cd\_terms = cd\_terms + ploy\_highorder(\beta, i, k) struct\_outarray(i - 1)$
  end for
end for
$u_{n+1} = 2u_n - u_{n-1} + cd\_terms$

Two unique functions were designed to compute solutions for the convolution operators $\mathcal{D}$ (Algorithm 9) and $\mathcal{C}$ (Algorithm 8) by leveraging the fast convolution algorithm (Algorithm 1). Even though these two algorithms can be used together to compute the operators $\mathcal{C}$ and

$\mathcal{D}$ by applying them on the same vector of values at any stage, we need a separate function as given in Algorithm 10 in order to avoid redundant calculations and follow up issues with computing certain homogeneous boundary coefficients. For this algorithm, we expand the equation

$\mathcal{C}_{xy} = L_y^{-1}D_x + L_x^{-1}D_y$ as follows using $\mathcal{D}_\gamma = 1 - L_\gamma^{-1}$,

$$\mathcal{C}_{xy} = (L_y^{-1} + L_x^{-1}) - (L_y^{-1}L_x^{-1} + L_x^{-1}L_y^{-1}) \tag{2.78}$$

The Algorithm 6 computes the coefficients of the equation (2.75) for higher dimension, higher-order accuracy in terms of β using (2.81).

Algorithm 8 *compute_c* : Compute operator $\mathcal{C}$ in 2D; $\mathcal{C}_{xy} = L_y^{-1}D_x + L_x^{-1}D_y$

1: u - array of u values at time t,
2: n_x, n_y - number of points along x and y respectively,
3: β - averaging parameter,
4: $\mu_x = e^{-\alpha(x(n)-x(0))}$,
5: $\mu_y = e^{-\alpha(y(n)-y(0))}$,
6: x_A, x_B, and y_A, y_B - boundary points along x and y respectively,
7: $expA_x$, $expB_x$ - exponential vector of grid distance from left boundary and right boundary respectively along x,
8: $expA_y$, $expB_y$ - exponential vector of grid distance from bottom boundary and up boundary respectively along y,
9: ν_x, ν_y - array of weighted nodes along x and y respectively,
10: $expweight_x$, $expweight_y$ - array of exponentially weighted nodes along x and y respectively,
11: p_s -order of accuracy in space
12: $D_x = u - Linv(u, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
13: $linv\_yD_x = linv(D_x, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p_s, 1)$
14: $D_y = u - linv(u, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p_s, 1)$
15: $linv\_xD_y = linv(D_y, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
16: $C_{xy} = linv\_yD_x + linv\_xD_y$

2.5 Treatment for Variable Speed Wave Propagation

This procedure of exchanging time derivatives with spatial derivatives in the semi-discrete equation using the Lax-Wendroff procedure will also work in the case of variable wave speeds.

Algorithm 9 *compute_d* :Compute operator $\mathcal{D}$ in 2D; $\mathcal{D}_{xy} = 1 - L_y^{-1} L_x^{-1}$

u - array of u values at time t, n_x, n_y - number of points along x and y respectively,
β - averaging parameter,
$\mu_x = e^{-\alpha(x(n)-x(0))}$,
$\mu_y = e^{-\alpha(y(n)-y(0))}$,
x_A, x_B, and y_A, y_B - boundary points along x and y directions respectively,
$expA_x$, $expB_x$ - exponential vector of grid distance from left boundary and right boundary respectively along x,
$expA_y$, $expB_y$ - exponential vector of grid distance from bottom boundary and up boundary respectively along y,
ν_x, ν_y - array of weighted nodes along x and y respectively,
$expweight_x$, $expweight_y$ - array of exponentially weighted nodes along x and y respectively,
p_s -order of accuracy in space
$linv\_x = linv(u, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
$linv\_yx = linv(linv\_x, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p_s, 1)$
$D_{xy} = u - linv\_yx$

Algorithm 10 *compute_cd* : Compute operators $\mathcal{C}$ and $\mathcal{D}$ in 2D; $\mathcal{D}_{xy}$ and $\mathcal{C}_{xy}$

u - array of u values at time t, n_x, n_y - number of points along x and y respectively,
β - averaging parameter,
$\mu_x = e^{-\alpha(x(n)-x(0))}$,
$\mu_y = e^{-\alpha(y(n)-y(0))}$,
x_A, x_B, and y_A, y_B - boundary points along x and y respectively,
$expA_x$, $expB_x$ - exponential vector of grid distance from left boundary and right boundary respectively along x,
$expA_y$, $expB_y$ - exponential vector of grid distance from bottom boundary and up boundary respectively along y,
ν_x, ν_y - array of weighted nodes along x and y respectively,
$expweight_x$, $expweight_y$ - array of exponentially weighted nodes along x and y respectively,
p_s -order of accuracy in space
$linv\_x = linv(u, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
$linv\_y = linv(u, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p_s, 1)$
$linv\_yx = linv(linv\_x, n_x, \beta, \mu_y, y_A, y_B, expA_y, expB_y, \nu_y, expweight_y, p_s, 1)$
$linv\_xy = linv(linv\_y, n_y, \beta, \mu_x, x_A, x_B, expA_x, expB_x, \nu_x, expweight_x, p_s, 0)$
$D_{xy} = u - linv\_yx$
$C_{xy} = linv\_x + linv\_y - (linv\_yx + linv\_xy)$

The wave equation with variable wave speed can be written as,

$$u_{tt} = [c(x)]^2 u_{xx} \tag{2.79}$$

Assume that the speed is finite and bounded as $c_1 \leq c(x) \leq c_2$. It can be normalized by $\bar{c}(x) = c(x)/c_2$. We choose $\alpha = \frac{\beta}{c_2 \Delta t}, \quad \beta > 0$. Thus,

$$\begin{aligned} u^{n+1} - 2u^n + u^{n-1} &= 2\sum_{m=1}^{\infty} \frac{\Delta t^{2m}}{(2m)!} (\partial_{tt})^m u^n \\ &= 2\sum_{m=1}^{\infty} \frac{\Delta t^{2m}}{(2m)!} \left(c^2(x)\partial_{xx}\right)^m u^n \\ &= 2\sum_{m=1}^{\infty} \frac{\beta^{2m}}{(2m)!} \left(\bar{c}^2(x)\frac{\partial_{xx}}{\alpha^2}\right)^m u^n \\ &= \sum_{m=1}^{\infty} (-1)^m \frac{2\beta^{2m}}{(2m)!} \bar{c}^{2m}(x) \sum_{p=m}^{\infty} \binom{p-1}{m-1} \mathcal{D}^p[u^n] \\ &= \sum_{p=1}^{\infty} A_p(\beta)\mathcal{D}^p[u^n], . \end{aligned} \tag{2.80}$$

and the polynomial of coefficients will be,

$$A_p(\beta) = 2\sum_{m=1}^{p} (-1)^m \frac{(\beta\bar{c}(x))^{2m}}{(2m)!} \binom{p-1}{m-1}. \tag{2.81}$$

2.6 Numerical Results

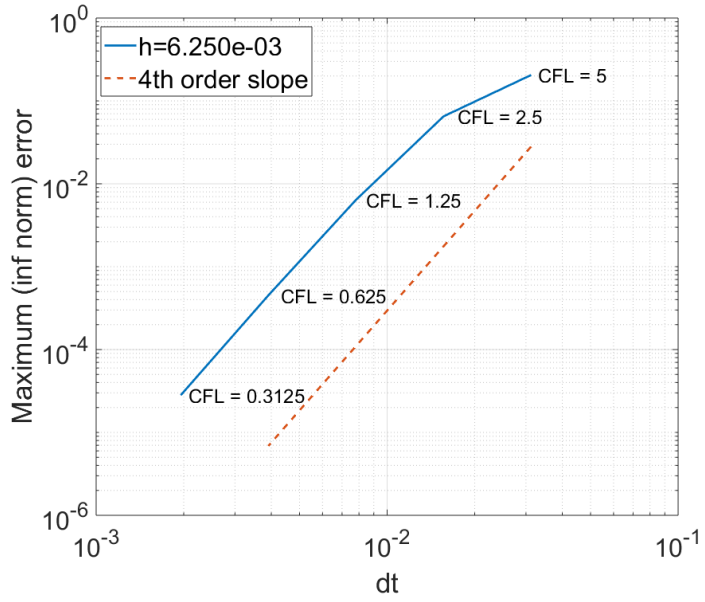
We present a set of test cases that evaluates our framework; specifically, we consider problems with variable/piecewise constant wave speed in one, two, and three dimensions, using a Gaussian pulse as an initial source or a point source. For 2D, we choose a classic EM problem - a transverse magnetic (TM) mode photonic crystal waveguide.

2.6.1 Convergence Studies

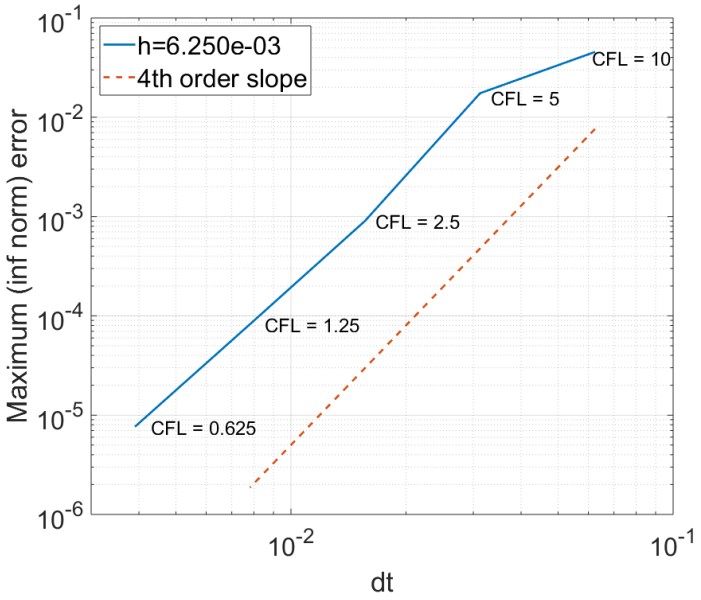
2.6.1.1 Higher Order Wave Solvers in Two Dimension

In this section, we consider a two-dimensional fourth-order temporally accurate solver. A square domain ($\Omega = [-1, 1] \times [-1, 1]$) is chosen to assess the two dimensional higher-order solver. We place a point source $cos(\omega t)$ at the center of the domain, $(0, 0)$ and apply different

boundary conditions.



(a) Dirichlet boundary condition



(b) Periodic boundary conditions

Figure 2.3: Fourth order time convergence of the 2D wave solver using Dirichlet (a) and periodic (b) boundary conditions with $\Delta x = \Delta y = 6.25 \times 10^{-3}$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 2.0$ on a square domain $\Omega = [-1, 1]^2$ with a point source $cos(\omega t), \omega = 1$ at the center of the box, $(0, 0)$. The CFL ($= \frac{c\Delta t}{\Delta x}$) value changes proportional to the time step size Δt with fixed spatial step size.

Figure 2.3 shows fourth order time convergence obtained for Dirichlet and periodic boundary conditions by performing a refinement study on a square domain $\Omega = [-1, 1] \times [-1, 1]$ with a point source $cos(\omega t)$, $\omega = 1$ at the center of the box, $(0, 0)$. The CFL $(= \frac{c\Delta t}{\Delta x})$ value changes proportional to the time step size Δt with fixed spatial step size, $\Delta x = \Delta y = 0.00625$. We consider a final time $T = 2.0$, with a fixed spatial resolution of 320×320 spatial points. The discrete L_∞ norm of the error is constructed at each time step and the maximum over all time steps is used to generate the graph in Figure 2.3.

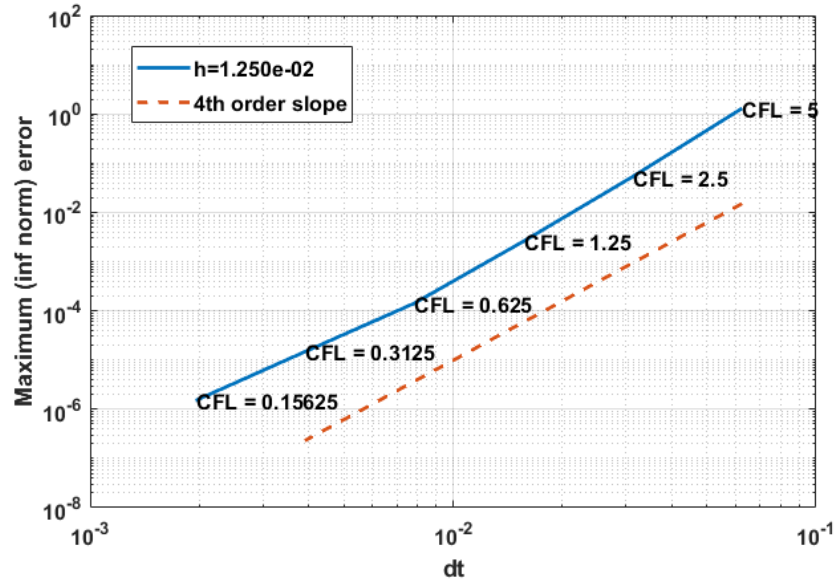
2.6.1.2 Higher Order Wave Solvers in Three Dimensions

We first show fourth order convergence for Dirichlet boundary condition by performing a time refinement study on a cubic domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$ with a point source $cos(\omega t)$ at center of the domain, $(0, 0, 0)$. This runs up to time $T = 2.0$, with a fixed spatial resolution of $160 \times 160 \times 160$ spatial points. The discrete L^∞ norm of the error is constructed at each time step and the maximum error over all time steps is used to graph Figure 2.4 for $\Delta t = \frac{625e-2}{2^k}$, $k = 1$ to 5, with Dirichlet boundary conditions.

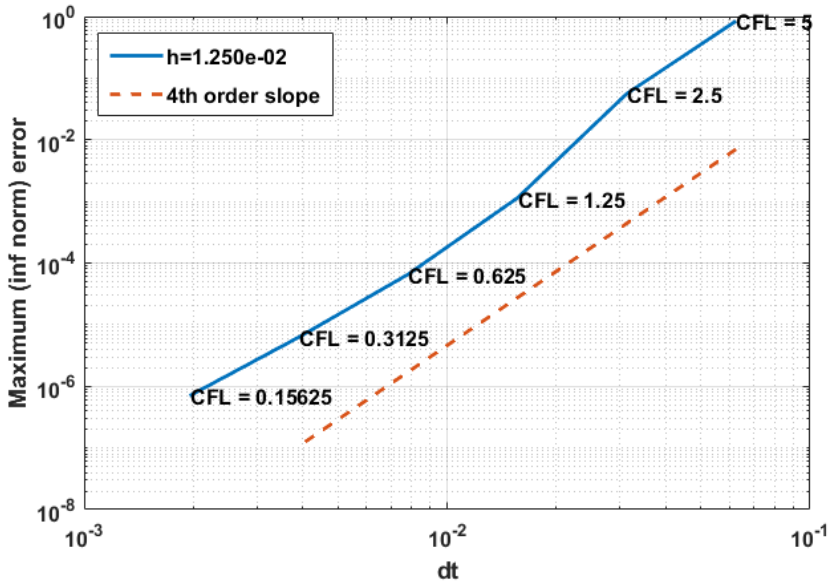
2.6.2 Three Dimensional Waves

In this section, we validate our three dimensional solver using a cubic domain ($\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$). We set the spatial grid size as $\Delta x = \Delta y = \Delta z = 0.0625$ ($32 \times 32 \times 32$ grid points) and the time step size to $\Delta t = 0.0313$. We successfully tested the solver for a point source by applying Dirichlet, periodic and outflow boundary conditions along the cubic faces. Figures 2.5, 2.6, and 2.7 show the field of a point source $cos(\omega t), \omega = 1$ that is placed at $(0, -1+3\Delta x, 0)$ using Dirichlet, periodic, and outflow boundary conditions along the six faces respectively.

In each case, a three dimensional wave generated by a point source (Figure 2.5, 2.6, and 2.7) smoothly propagates in a cube and obeys applied boundary conditions along the surfaces of the cube.



(a) $\omega = 0.1$



(b) $\omega = 1$

Figure 2.4: Fourth order convergence of 3D wave solver using Dirichlet boundary conditions with $\Delta x = \Delta y = \Delta z = 1.25 \times 10^{-2}$ $\omega = 1$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 2.0$ on a cubic domain $\Omega = [-1, 1]^3$ with a point source $cos(\omega t)$ at the center of the cube, $(0, 0, 0)$ for (a) $\omega = 0.1$ and (b) $\omega = 1$. The CFL $(= \frac{c\Delta t}{\Delta x})$ value changes proportional to the time step size Δt with fixed spatial step size.

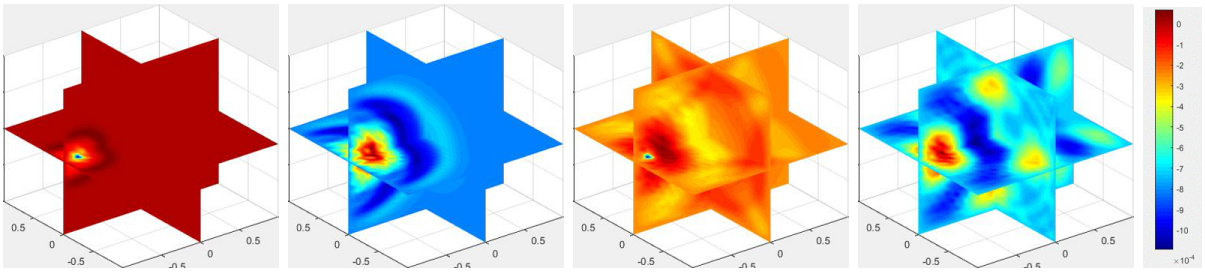


Figure 2.5: Time evolution of a point source field $cos(\omega t), \omega = 1$ within a cubic domain ($\Omega = [-1, 1]^3$) by imposing Dirichlet boundary condition along the surface with the spatial step size, $\Delta x = \Delta y = \Delta z = 0.0625$ $\omega = 1$ and the time step size, $\Delta t = 0.0313$.

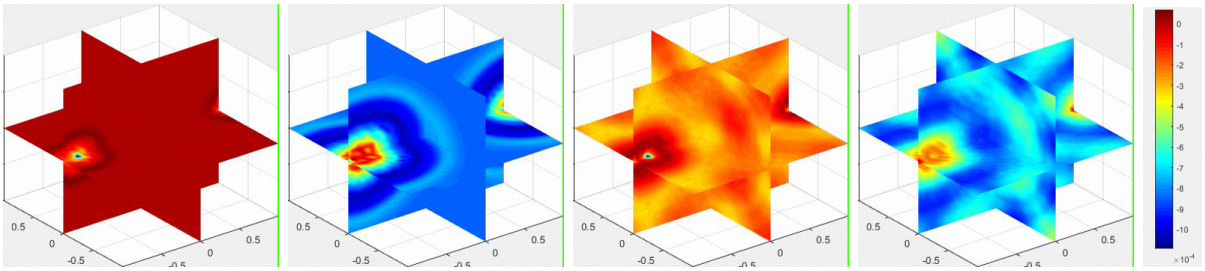


Figure 2.6: Time evolution of a point source field $cos(\omega t), \omega = 1$ within a cubic ($\Omega = [-1, 1]^3$) by imposing periodic boundary condition along the surface with the spatial step size, $\Delta x = \Delta y = \Delta z = 0.0625$ $\omega = 1$ and the time step size, $\Delta t = 0.0313$.

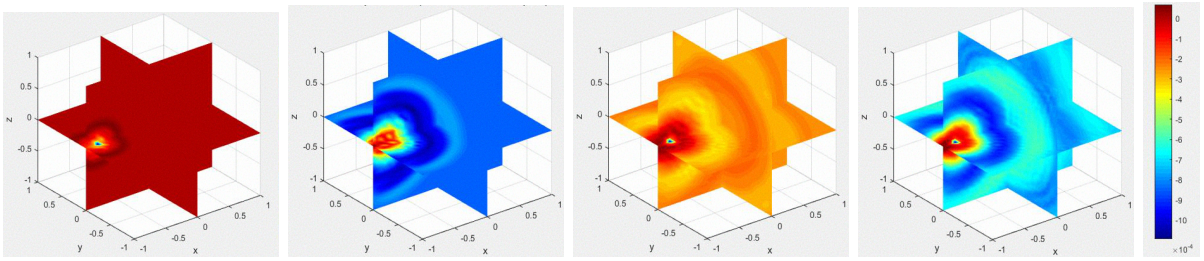


Figure 2.7: Time evolution of a point source field $cos(\omega t), \omega = 1$ within a cubic ($\Omega = [-1, 1]^3$) by imposing outflow boundary condition along the surface with the spatial step size, $\Delta x = \Delta y = \Delta z = 0.0625$ $\omega = 1$ and the time step size, $\Delta t = 0.0313$.

2.6.3 Waves with Variable Speeds

2.6.3.1 One Dimensional Case

A one dimensional wave travels through two different consecutive domains, $\Omega_1 \in [-1, 0]$ and $\Omega_2 \in [0, 1]$ with piecewise constant speed $c_1 = 1.0$ and $c_2 = 2.0$ respectively. The solution

until the wave reaches the location of discontinuity, $x = 0$ is,

$$u(x,t) = a_i e^{-\eta(x-c_1(t-t_0))^2} \tag{2.82}$$

where a_i is the initial amplitude, η is the Gaussian shape parameter, and t_0 is a time offset which delays the Gaussian pulse.

A portion of the traveling wave in domain one, Ω_1 will be reflected at the location of interface, $x = 0$, while the remaining waves are transmitted to domain two, Ω_2. Thus, we need to consider the combination of transmitted, reflected, and incident waves from the location of the interface, $x = 0$ and onward. The left domain (Ω_1) will have components of the incident and reflected waves, and the right domain (Ω_2) will have the transmitted component of the original wave.

In this way, the solution obeys the following equations,

$$\begin{aligned} u_1(t) &= a_i e^{\eta(x-c_1(t-t_0))^2} + a_r e^{\eta(x+c_1(t-t_0))^2} \quad x < 0 \\ u_2(t) &= a_t e^{\bar{\eta}(x-c_2(t-t_0))^2} \quad x \geq 0, \end{aligned}$$

where, a_r and a_t are amplitudes of the reflected and transmitted wave respectively and $\bar{\eta}$ is the shape parameter of the transmitted Gaussian pulse. Because of the zero transverse displacement at the interface, $u_1(0,t) = u_2(0,t)$, that gives us,

$$(a_i + a_r)e^{\eta c_1^2(t-t_0)^2} = a_t e^{\bar{\eta} c_2^2(t-t_0)^2}.$$

By equating the coefficients we get,

$$a_i + a_r = a_t \tag{2.83}$$

$$\eta c_1^2 = \bar{\eta} c_2^2. \tag{2.84}$$

From energy conservation, we obtain

$$\frac{a_i^2}{\sqrt{\eta}} = \frac{a_r^2}{\sqrt{\eta}} + \frac{a_t^2}{\sqrt{\bar{\eta}}}. \tag{2.85}$$

Upon solving equations (2.84), and (2.85), we obtain

$$a_t = \frac{2a_i c_2}{(c_1 + c_2)},$$

$$a_r = a_i - a_t.$$

For our first numerical example, we choose $a_i = 1$, so $a_t = \frac{2c_2}{c_1+c_2}$, and $a_r = 1 - a_t$. Further, we chose $c_1 = 1.0$ and $c_2 = 2.0$, and apply our solver along the domain $\Omega_1 \cup \Omega_2$ with 1024 uniform grid points of size $\Delta x = 0.002$, while maintaining a CFL of 2.5 by choosing time step $\Delta t = 0.005$. Figure 2.8 shows time snapshots of the moving wave using outflow boundary conditions at the left and right boundaries. The numerical results agree closely with the theoretical solutions as can be seen in the figure.

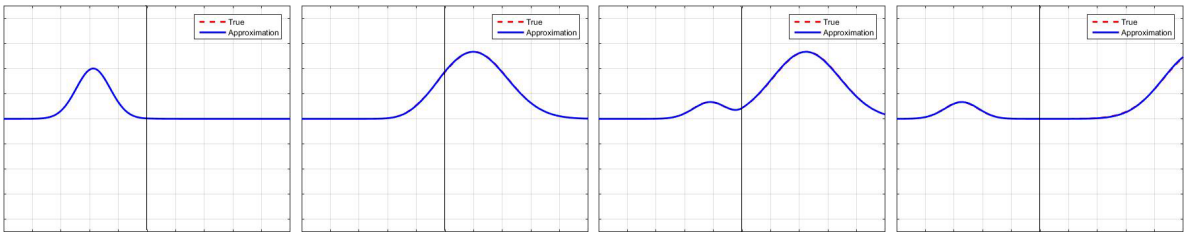


Figure 2.8: 1D wave travelling over the two domains Ω_1 and Ω_2 with piecewise constant speed $c_1 = 1.0$ and $c_2 = 2.0$. For this experiment, we choose spatial step $\Delta x = 0.002$, and time step $\Delta t = 0.005$.

A problem with multiple domains was selected as our second numerical test case with the same spatial step size, $\Delta x = 0.002$, and time step size $\Delta t = 0.005$, hence CFL remains 2.5. The entire domain has eight equal width sub-domains of size 0.25, and the wave travels at the same speed in every bi-domain. The first sub-domain is between -1 and -0.75, has waves travelling with speed c_1 (=1.0); the wave speed is c_2 (= 2.0) in the even-numbered sub-domains.

Figure 2.9 shows time snapshots of the moving wave using outflow boundary conditions at the external left and right boundaries.

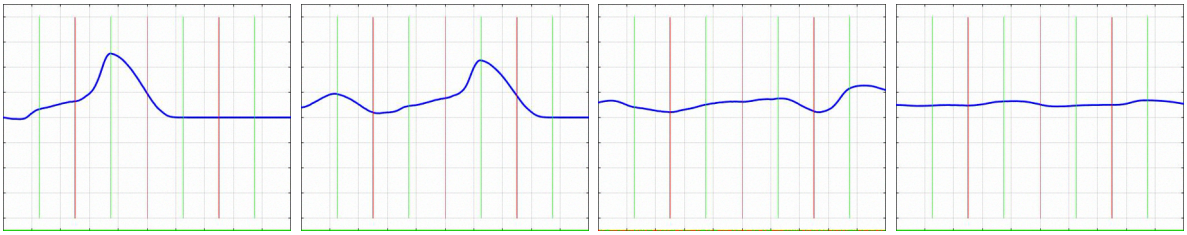


Figure 2.9: 1D wave travelling over the layered media with eight domains and the wave travels with the same speed in every bi-domain ($c_1 = 1.0$ and $c_2 = 2.0$). For this experiment, we choose spatial step $\Delta x = 0.002$, and time step $\Delta t = 0.005$ along layered media with piecewise constant wave speed.

2.6.3.2 Two Dimensional Case

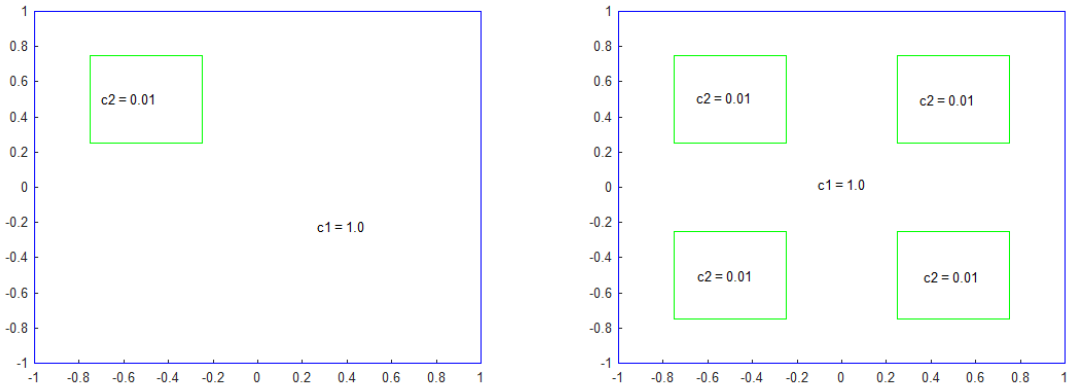
In this section, we consider the two-dimensional variable speed solver. A square domain with square patches is chosen to assess the two-dimensional variable speed solver. Due to the material property of patches, waves travel with different speeds through these patches compared with the remaining area. We choose a Gaussian pulse $e^{-25(x^2+y^2)}$ as an initial solution and apply outflow boundary condition along the border of the square domain ($\Omega = [-1, 1] \times [-1, 1]$). Since we use spatial step size $\Delta x = \Delta y = 0.02$ and time step size $\Delta t = 0.005$, applicable CFL is 0.25. We demonstrate two test cases: One and Four patches are placed for case-i and case-ii respectively as shown in Figure 2.10

Case-i - Single patch

A 0.5×0.5 square patch is placed at the left top corner centered at, (-0.5, 0.5). The wave speed is set to be $c_2(= 0.1)$ in the patch, and $c_1(= 1.0)$ in the remaining area. Figure 2.11 shows time snapshots of the solution.

Case-ii - Four patches

There are four 0.5 x 0.5 square patches placed at the four corners; i.e., centered at (-0.5, 0.5), (-0.5, -0.5), (0.5, 0.5), and (0.5, -0.5). The wave speed is set to be $c_2(= 0.1)$ in every patch, and $c_1(= 1.0)$ in the remaining area. Figure 2.12 shows time snapshots of the wave



(a) single patch (b) 4 patches

Figure 2.10: Geometrical view of 0.5×0.5 (a) one and (b) four square patches in the square domain $\Omega = [-1, 1]^2$ with $c_2(= 0.1)$ and $c_1(= 1.0)$ are the wave speeds in the patches and the remaining area.

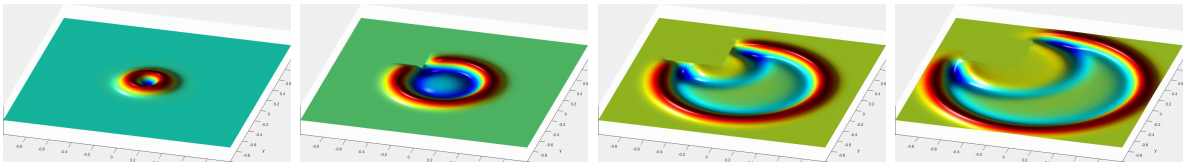


Figure 2.11: Time evolution of a Gaussian field $e^{-25(x^2+y^2)}$ in a square domain $\Omega = [-1, 1]^2$ with a 0.5^2 square patch as shown in Figure 2.10-(a). Here, the spatial step size, $\Delta x = \Delta y = 0.02$ and the time step size, $\Delta t = 0.005$.

at different time instants.

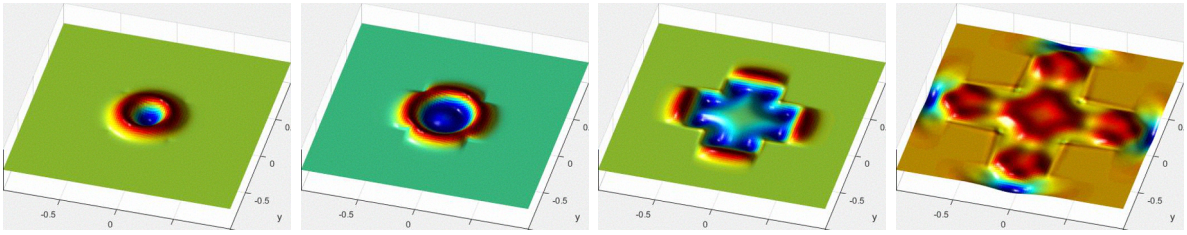
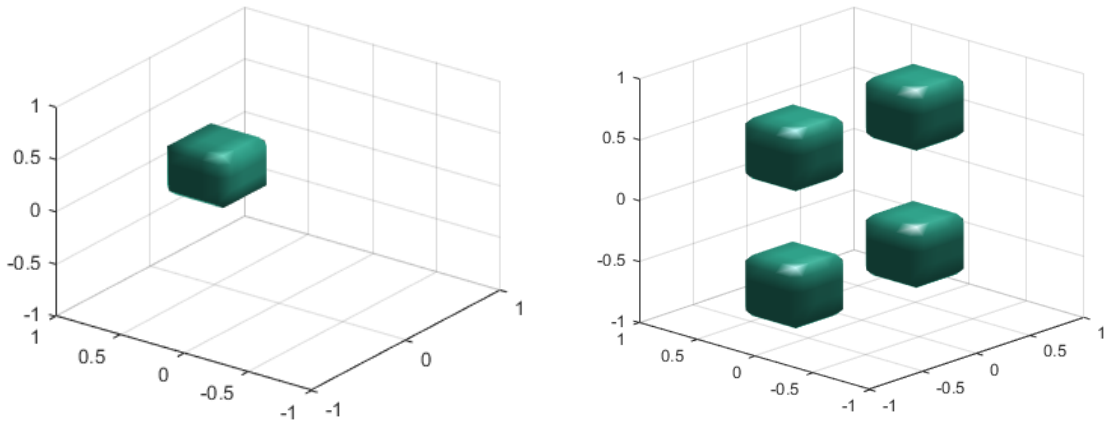


Figure 2.12: Time evolution of a Gaussian field $e^{-25(x^2+y^2)}$ in a square domain $\Omega = [-1, 1]^2$ with four 0.5^2 square patches as shown in Figure 2.10-(b). Here, the spatial step size, $\Delta x = \Delta y = 0.02$ and the time step size, $\Delta t = 0.005$.

2.6.3.3 Three Dimensional Cases

In this section, we discuss the three-dimensional variable speed wave-solver. A cubic domain with cubic patches is chosen to assess the three-dimensional variable speed solver. Due to the material property of patches, waves travel with different speeds through these patches compared with the remaining area. We place a Gaussian pulse $e^{-36(x^2+y^2+z^2)}$ as an initial solution and apply outflow boundary conditions along the border of the cube ($\Omega = [-1,1] \times [-1,1] \times [-1,1]$). Since we use a spatial grid size $\Delta x = \Delta y = \Delta z = 0.0625$ ($32 \times 32 \times 32$ grid points) and time step size $\Delta t = 0.0313$, applicable CFL is 0.5. We demonstrate two test cases: one, and four patches for case-i and case-ii respectively as shown in Figure 2.13



(a) single patch (b) 4 patches

Figure 2.13: Geometrical view of $0.5 \times 0.5 \times 0.5$ (a) one and (b) four cubic patches in the cubical domain $\Omega = [-1,1]^3$ with $c_2(= 0.1)$ and $c_1(= 1.0)$ are the wave speeds in the patches and the remaining area.

Case-i - Single patch

A $0.5 \times 0.5 \times 0.5$ cubic patch is placed at the left top corner centered at, (-0.5, 0, 0.5). The wave speed is set to be $c_2(= 0.1)$ in the patch, and $c_1(= 1.0)$ in the remaining area. Figure 2.14 shows snapshots of the wave at different time instants.

Case-ii - Four patches

There are four $0.5 \times 0.5 \times 0.5$ cubic patches placed at the four corners; i.e., centered at (-0.5,

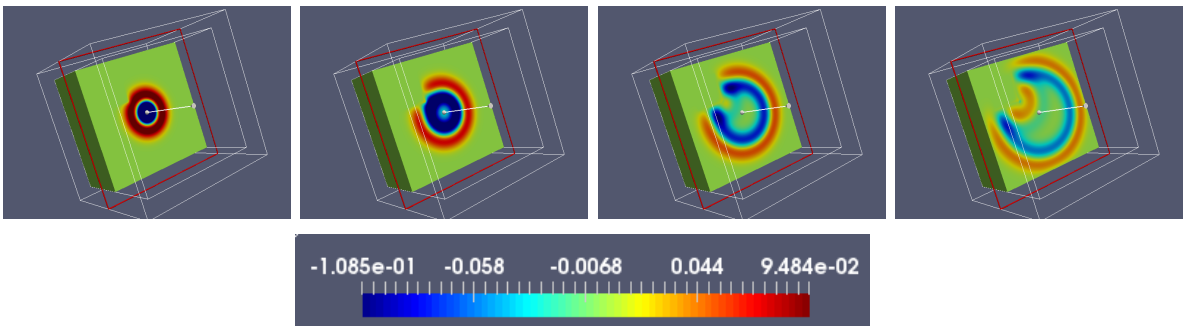


Figure 2.14: Time evolution of a Gaussian field $e^{-25(x^2+y^2+z^2)}$ in a cubical domain $\Omega = [-1, 1]^3$ with a 0.5^3 cubic patch as shown in Figure 2.13-(a). Here, the spatial step size, $\Delta x = \Delta y = \Delta z = 0.0625$ and the time step size, $\Delta t = 0.0313$.

0, 0.5), (-0.5, 0, -0.5), (0.5, 0, 0.5), and (0.5, 0, -0.5). The wave speed is set to be $c_2(= 0.1)$ in every patch, and $c_1(= 1.0)$ in the remaining area. Figure 2.15 shows snapshots of the wave at different time instants.

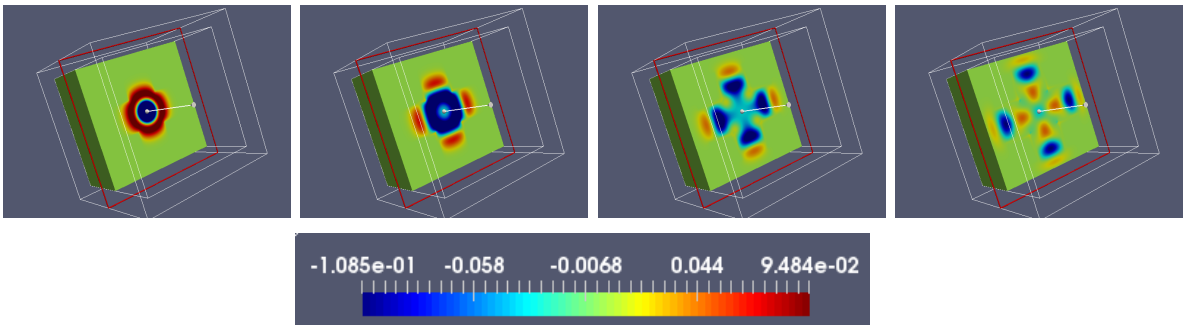


Figure 2.15: Time evolution of a Gaussian field $e^{-25(x^2+y^2+z^2)}$ in a cubical domain $\Omega = [-1, 1]^3$ with four 0.5^3 cubic patches as shown in Figure 2.13-(b). Here, the spatial step size, $\Delta x = \Delta y = \Delta z = 0.0625$ and the time step size, $\Delta t = 0.0313$.

The 3D Gaussian pulse smoothly travels through the different material interfaces provided by patches (one in Figure 2.14 and four in Figure 2.15) and the rest of the area.

2.6.4 Waveguide in a Photonic Crystal

2.6.4.1 Problem Definition

We study the propagation of light in a photonic crystal, a low-loss periodic dielectric medium, in which the atoms or molecules are replaced by macroscopic media with different

dielectric constants, and the periodic potential is replaced by periodic dielectric functions [43]. These are very useful in applications such as the design of fiber-optic cables, laser engineering, high-speed computing, and spectroscopy.

In order to study this propagation, we first cast Maxwell equations written in terms of the electric field, **E** and magnetic field **H** in the form of a second-order wave equation.

2.6.4.2 Geometry of the Problem

A two-dimensional photonic crystal which is periodic in the x, y directions and homogeneous in the z direction is chosen as the test domain. The crystal has photonic band gaps in the xy plane and it can prevent light from propagating in any direction within the plane. This system has discrete translational symmetry in the xy plane, and this symmetry property can be used to characterize its electromagnetic modes. Any modes that propagate in the xy plane are invariant under reflections through the plane. The Transverse Magnetic (TM) modes have the **H** field in the plane and, the **E** field normal to the plane. As shown in Figure 2.16, a line defect is introduced along the y direction by removing a line of rods in a set of alumina rods. This can be represented using a square lattice of cylindrical rods in the air (Figure 2.17).

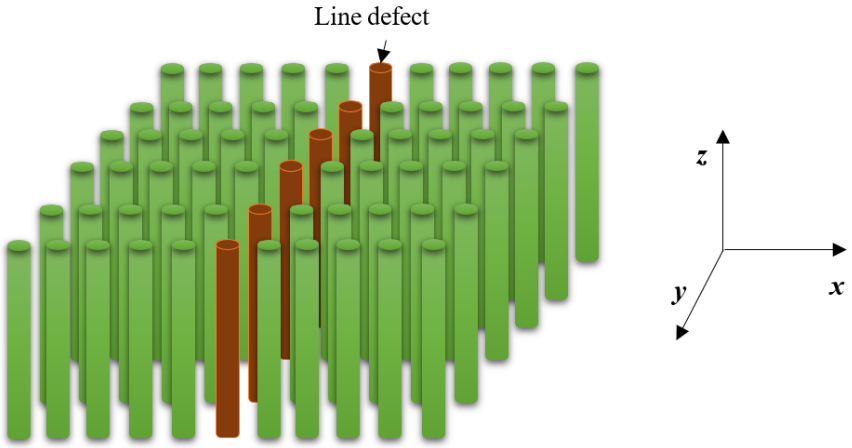


Figure 2.16: Schematic illustration of a line defect (in red) within a set of cylindrical rods (in green) on a photonic crystal.

In our model, we chose an m-by-n lattice that has dielectric rods which are periodic along x (n rods) and y (m rods) axes with the lattice constant a. The cylindrical rod with radius r and dielectric constant ϵ is homogeneous along the z direction (we imagine the cylinders are infinitely tall). A column of rods is removed to produce a cavity, so it has reflecting walls on both sides along the y direction. We consider the waveguide in TM mode using the following boundary conditions: periodic in the x direction and outflow in the y direction. We assume that there are no free charges, so only the dielectric constants are changing in space due to the rods.

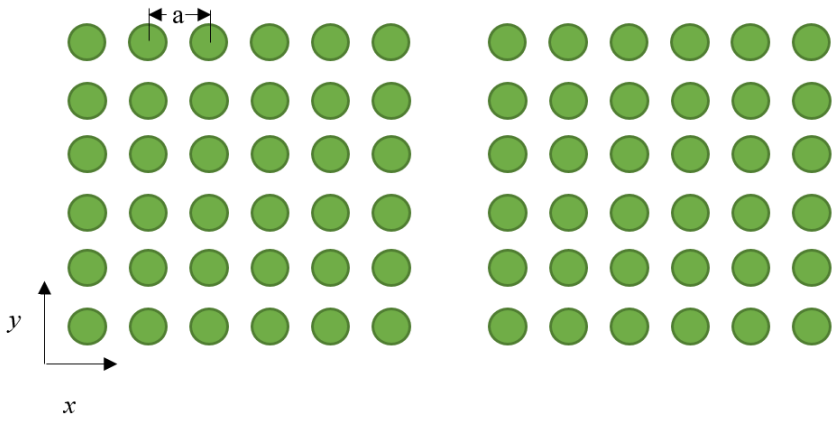


Figure 2.17: 2D Lattice of cylindrical rods which are periodic along x and y axes with lattice constant a on a photonic crystal.

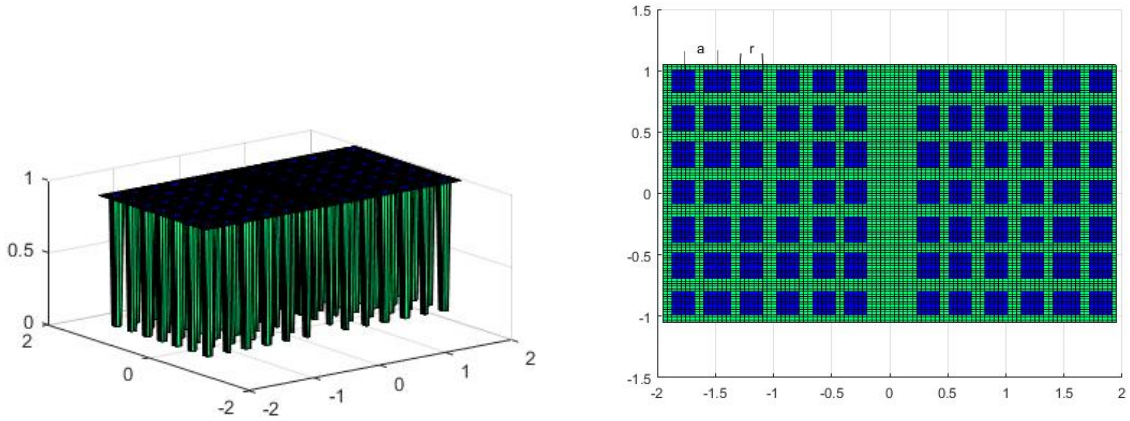
2.6.4.3 Result and Discussion

We performed numerical simulations of a waveguide on the photonic crystal using square and cylindrical rods. A rectangular (3.9×2.1) lattice of rods was chosen, the rods were placed in x, y directions with distance $a = 0.3$. The dielectric constant of rods was chosen to be $\epsilon = 8.9$. For the ADI scheme, we chose $n_x = n_y = 100$, with time step $t = 0.0105$.

Using square rods

The set of square rods of size $r = 0.38a$ is modeled as shown in Figure 2.18, and we obtained the result as shown in Figure 2.19 using the boundary conditions: periodic in the direction

x and outflow in the direction y for a point source $e^{(-i\omega t)}$, where $\omega = ck_y$, $k_y = 0.88(2a)$, $c = \frac{1}{\sqrt{(\epsilon\mu)}}$, $\epsilon = 8.9$, and $\mu = 1$.



(a) 3D (b) 2D

Figure 2.18: Square lattice of square rods of side length $0.38a$ with lattice constant a and dielectric constant $\epsilon = 8.9$ in 3D (a) and 2D (b), a linear defect is formed by removing a column of rods.

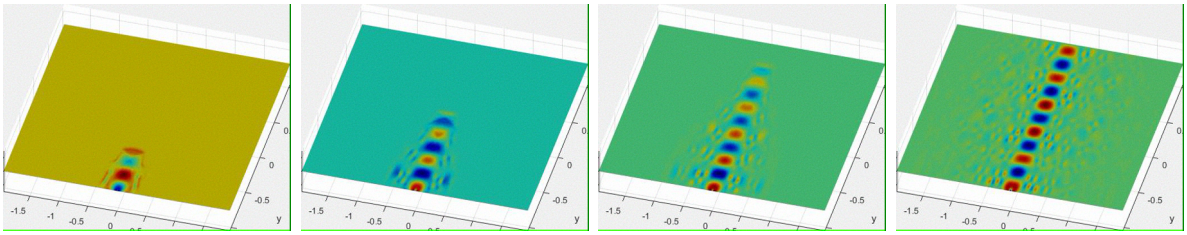


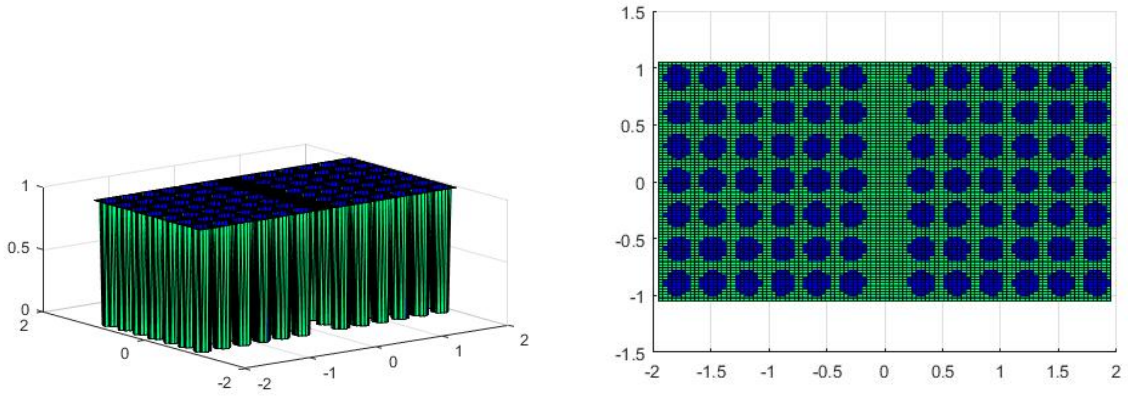
Figure 2.19: The wave generated by the point source, $e^{(-i\omega t)}$, where $\omega = ck_y$, $k_y = 0.88(2a)$, $c = \frac{1}{\sqrt{(\epsilon\mu)}}$, $\epsilon = 8.9$, and $\mu = 1$ travelling through the photonic waveguide using the model shown in Figure 2.18

Using cylindrical rods

We replaced the square rods with cylindrical rods and used the same parameter choices as in case i (Figure 2.20).

We obtained the result as shown in Figure 2.21 using the following boundary conditions: periodic in x and outflow in y

Further, we tested the model for a source-free wave equation with initial solution $e^{(-25(x^2+y^2))}$ using the same model (Figure 2.20) and obtained the result as shown in Figure 2.22. It is



(a) 3D (b) 2D

Figure 2.20: Square lattice of cylindrical rods of side length $0.38a$ with lattice constant a and dielectric constant $\epsilon = 8.9$ in 3D (a) and 2D (b), a linear defect is formed by removing a column of rods.

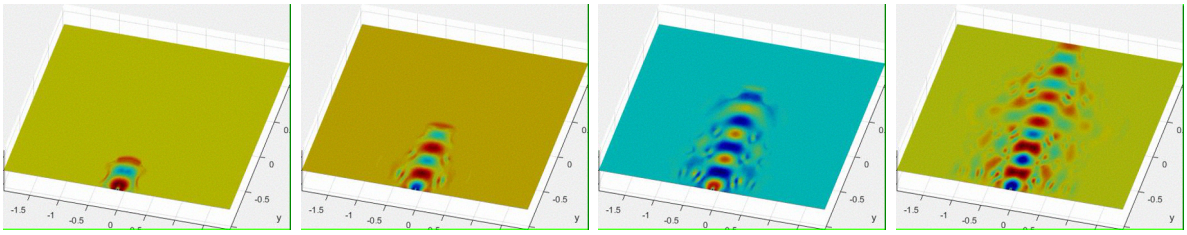


Figure 2.21: The wave generated by the point source, $e^{(-i\omega t)}$, where $\omega = ck_y$, $k_y = 0.88(2a)$, $c = \frac{1}{\sqrt{(\epsilon\mu)}}$, $\epsilon = 8.9$, and $\mu = 1$ travelling through the photonic waveguide using the model shown in Figure 2.20

important to know that with the embedded boundary method, such as we are pursuing here, it is no "harder" to introduce an embedded boundary method for a square then it is for a circle.

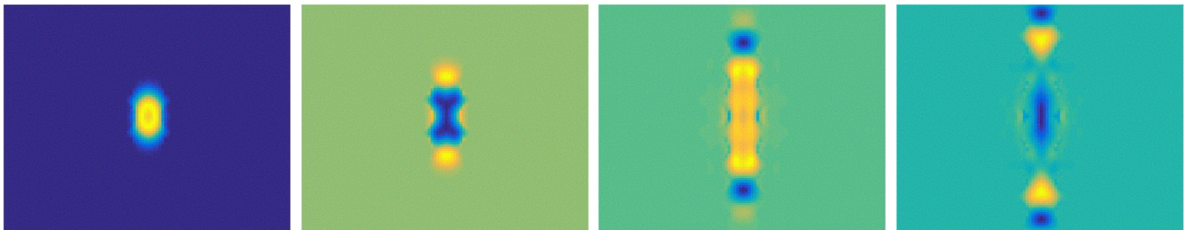


Figure 2.22: The wave, $e^{(-25(x^2+y^2))}$ traveling through the photonic waveguide using the model shown in Figure 2.20.

2.7 Summary

We derived the implicit semi-discrete solution for the multi-dimension wave equation subject to several applicable boundary conditions including Dirichlet, periodic, Neumann, and outflow boundary conditions. We extended the second-order scheme to arbitrary higher-order using Dirichlet and periodic boundaries on Cartesian grids. The Pth order scheme consumes $O(PN)$ time and $O(P^2N)$ space for the domain with N grid points, it can be reduced to ideally linear time and space complexity $O(N)$ for considerably large problems ($N >>$). The three-dimensional high-order scheme is implemented for Dirichlet and periodic boundary conditions and evaluated using different applications including a variable speed wave. In the next chapter, we describe the derivation of high-order outflow boundary conditions.

CHAPTER 3: Higher Order Non-reflecting Boundary Condition

3.1 Introduction

In this chapter, we develop a high-order non-reflecting boundary condition that can be applicable for curved boundaries based on the scheme described in Chapter 2.

A way to define a finite computational domain in the infinite physical domain is by using an artificial boundary which is called non-reflecting or open boundary condition. Since we develop simulation of A6 magnetron with diffraction output (A6 MDO), we need a scheme that successfully handles non-reflecting boundary conditions, even for curved boundaries. The approach described in Chapter 2 gives second-order accuracy for the outflow boundary condition. Basically, this approach extends the spatial domain for a specific time instance and switches spatial integration into time integration at the boundary ends. We use higher-order interpolation polynomial based on a higher-order finite-difference stencil for the time integration approximation and apply a suitable initial condition to reach high-order accuracy.

Historically, the open boundary condition was developed for outgoing wave propagation from the Sommerfeld radiation condition [66] by Zienkiewicz and Newton [74]. The wave-based interpretation of this boundary condition is more applicable for EM wave propagation [25; 60; 63]. Later, Higdon developed a sequence of increasing-order boundary conditions at various incidence angles [35]. The absorbing layer-based method, Perfectly Matched Layer (PML), is found to be the most popular method. It was developed by Berenger [6] for the 2-D Maxwell equations, and has been extended to 3D. PML imposes both losses of electric and magnetic fields, identifying the correct loss term for PML so that the outgoing waves not reflected in the domain can be problem-specific. However, it is difficult to choose an optimal conductive parameter(σ) for the best performance of the PML. Another approach,

the sponge layer method [16], requires extra grid points to make padding which implements the functions to prevent the reflection. Kirchhoff's formula-based method developed by Ting and Miksis [69] also requires more computation.

Recently, Bradley [1] introduced a high-order nonreflecting boundary condition which uses a compressed nonreflecting boundary kernel to achieve high-order for 2D and 3D cases. This boundary kernel approach is very complicated and consumes $O(MNlogN)$ work in 2D and $O(MNlog^2N)$ work in 3D, for M time steps and N spatial grid points. Without using the complicated boundary kernel approach, our higher-order scheme for outflow boundary condition works well for multi-dimensional problems with consuming $O(PMN)$ works, for the temporal order of accuracy $2P$ [14]. For arbitrarily big problems, $MN >> P$, so the complexity will be reduced to $O(MN)$.

In this chapter, we discuss 4-th order derivation for outflow boundary conditions. In Section 3.2, we derive the high-order outflow boundary condition and give a concrete definition. Section 3.3 gives the suitable initial condition for this approach, and finally we report a set of test cases in Section 3.4.

3.2 Extension to Higher Order in Accuracy

We need to apply boundary conditions appropriately to obtain high order accuracy. Dirichlet, Neumann, and periodic boundary conditions can be implemented straightforwardly (see [15] for details) using the approach shown in [14]. Since we have to know the behavior of the wave equation outside of the domain $x \leq a$ and $x \geq b$, the outflow boundary condition requires little additional computation as detailed in the following.

In order to obtain higher-order outflow boundary conditions, we consider more terms in the Taylor series. Note, equation (2.52) uses three terms for second-order accuracy. We use the same approach as explained previously in section 2.2.3.4. Besides, we perform some iterations along the boundary stencil to ensure convergence. For this procedure, we need to know the solution at time steps, t_{n+1}, t_{n+2}, and so on, if we choose the centered fi-

nite difference stencil to derive equations with higher-order accuracy (as was used to derive second-order outflow boundary conditions). A problem with this approach is that we need data at a future time, t_{n+2}, which we don't have direct access to. Instead, we prefer to use one-sided backward finite difference stencils to obtain higher-order accuracy and obtain boundary coefficients explicitly.

Let us derive outflow boundary conditions with fourth order accuracy. First, we construct a time interpolant at the right boundary ($x > b$) using a Taylor series expression of the form (Here $z = s/\Delta t$ as introduced in 2.52)

$$u(b, t_n - z\Delta t) \approx u(b, t_n) - z\Delta t u_t(b, t_n) + \frac{z^2\Delta t^2}{2} u_{tt}(b, t_n) - \frac{z^3\Delta t^3}{6} u_{ttt}(b, t_n) + \frac{z^4\Delta t^4}{24} u_{tttt}(b, t_n) \tag{3.1}$$

By truncating higher order error terms, we only need to approximate the first time derivative to fourth order accuracy, second time derivative to third order accuracy, third time derivative to second order accuracy, and fourth time derivative to first order accuracy. To perform this approximation, we use a five point backward finite difference stencil to approximate u_t, u_{tt}, u_{ttt}, and u_{tttt} to the desired order of accuracy. we obtain,

$$\begin{aligned} u(b, t_n - z\Delta t) \approx u^n(b) &- z\Big(\frac{25}{12}u^n(b) - 4u^{n-1}(b) + 3u^{n-2}(b) - \frac{4}{3}u^{n-3}(b) + \frac{1}{4}u^{n-4}(b)\Big) \\ &+ \frac{z^2}{2}\Big(\frac{35}{12}u^n(b) - \frac{26}{3}u^{n-1}(b) + \frac{19}{2}u^{n-2}(b) - \frac{14}{3}u^{n-3}(b) + \frac{11}{12}u^{n-4}(b)\Big) \\ &- \frac{z^3}{6}\Big(\frac{5}{2}u^n(b) - 9u^{n-1}(b) + 12u^{n-2}(b) - 7u^{n-3}(b) + \frac{3}{2}u^{n-4}(b)\Big) \\ &+ \frac{z^4}{24}\Big(u^n(b) - 4u^{n-1}(b) + 6u^{n-2}(b) - 4u^{n-3}(b) + u^{n-4}(b)\Big) \end{aligned} \tag{3.2}$$

Integrating this expression analytically using Lemma 2.1, we arrive at

$$B^n = e^{-\beta}B^{n-1} + \gamma_0 u^n(b) + \gamma_1 u^{n-1}(b) + \gamma_2 u^{n-2}(b) + \gamma_3 u^{n-3}(b) + \gamma_4 u^{n-4}(b) \tag{3.3}$$

where,

$$\begin{aligned}
\gamma_0 &= E_0(\beta) - \frac{25}{12}E_1(\beta) + \frac{35}{12}E_2(\beta) - \frac{5}{2}E_3(\beta) + E_4(\beta),\\
\gamma_1 &= 4E_1(\beta) - \frac{26}{3}E_2(\beta) + 9E_3(\beta) - 4E_4(\beta),\\
\gamma_2 &= -3E_1(\beta) + \frac{19}{2}E_2(\beta) - 12E_3(\beta) + 6E_4(\beta),\\
\gamma_3 &= \frac{4}{3}E_1(\beta) + \frac{14}{3}E_2(\beta) + 7E_3(\beta) - 4E_4(\beta),\\
\gamma_4 &= -\frac{1}{4}E_1(\beta) + \frac{11}{12}E_2(\beta) - \frac{3}{2}E_3(\beta) + E_4(\beta).
\end{aligned} \tag{3.4}$$

Likewise, by considering the left boundary $x < a$, we get

$$A^n = e^{-\beta}A^{n-1} + \gamma_0 u^n(a) + \gamma_1 u^{n-1}(a) + \gamma_2 u^{n-2}(a) + \gamma_3 u^{n-3}(a) + \gamma_4 u^{n-4}(a). \tag{3.5}$$

We can also compute the coefficients for second-order accuracy using the explicit approach in this form,

$$\begin{aligned}
A^n &= e^{-\beta}A^{n-1} + \gamma_0 u^n(a) + \gamma_1 u^{n-1}(a) + \gamma_2 u^{n-2}(a).\\
B^n &= e^{-\beta}B^{n-1} + \gamma_0 u^n(b) + \gamma_1 u^{n-1}(b) + \gamma_2 u^{n-2}(b).
\end{aligned} \tag{3.6}$$

where,

$$\begin{aligned}
\gamma_0 &= E_0(\beta) - \frac{3}{2}E_1(\beta) + E_2(\beta),\\
\gamma_1 &= 2E_1(\beta) - 2E_2(\beta),\\
\gamma_2 &= -\frac{1}{2}E_1(\beta) + E_2(\beta).
\end{aligned} \tag{3.7}$$

Now we have equations to compute the homogeneous boundary coefficients A^n and B^n to second and fourth-order accuracy. Note that the boundary constants corresponding to operator $D[u]$ (denoted by A_1^n, B_1^n) are independent of the boundary constants corresponding

to operator $D^2[u]$ (denoted by A_2^n, B_2^n). Therefore, our fourth-order wave solution can be constructed on two levels.

- Level 1

  Compute $D[u]$ using u; A_1^n and B_1^n are obtained by second order solution implicitly (2.58) or explicitly (3.6).

- Level 2

  Compute $D^2[u]$ using $D[u]$; A_2^n and B_2^n are obtained by fourth order solution (3.3) and (3.5) explicitly

Now, we provide algorithms for fourth-order outflow boundary conditions in one dimension. We first modify Algorithm 5 to work for outflow boundary conditions with fourth-order accuracy in time. This Algorithm 11 computes γ coefficients (see 3.7 and 3.4) using second-order centered, and fourth-order backward finite different stencils in Level 1 and Level 2 computations respectively. Initially, outflow boundary coefficients (vector H) are set to zero, meaning boundary coefficients at time step t_0 are zero, and previous solutions at boundaries are maintained (ubp_{pr}) for the computation.

During the computation of $\mathcal{L}^{-1}$ using outflow (Algorithm 12), we use an iterative method to ensure the solution converges. In each iteration, it computes boundary coefficients and updates the solutions within the boundary stencil. We perform the iteration until it satisfies a criterion $|u - u_{temp}|_\infty < tol$ where tol is a tolerance which may be chosen to be quite small. We developed an Algorithm 13 to compute outflow boundary coefficients using the previously explained approach.

A general procedure to compute the γ coefficients for any order based on a given finite difference stencil is designed in Algorithm 14. This algorithm utilizes a function $fdcoeffF(k, \bar{x}, x)$ published by [49] to compute coefficients for finite difference approximations of the derivative of order k at $\bar{x}$ based on grid values at points in the boundary stencil x. In Algorithm 15, the function E represents the expression defined in lemma 2.1

Algorithm 11 Compute u at time t_{n+1} by imposing outflow boundary conditions

u_n, u_{n-1}, and u_{n-2} - array of u values at time t_n, t_{n-1}, and t_{n-2} respectively,
$bdry_A, bdry_B$ - boundary points,
$expA$ $expB$- exponential vector of grid distance from left boundary and right boundary respectively along the line,
ν - array of weighted nodes along the line,
$expweight$ - array of exponentially weighted nodes along the line,
p_s, p_t -order of accuracy in space and time respectively,
xA, xB - left and right boundary points respectively.
$stimp_2 = 1 : -1 : 1$
$stexp_2 = 0 : -1 : -4$
$g(1,:) = gamma(\beta, 2, stimp_2)$
$g(2,:) = gamma(\beta, 4, stexp_4)$
$P = 2$
$Dterms = 0$
for $k = 1$ to P **do**
    $D = u - linv\_out(u_n, \beta, bdry_A, bdry_B, expA, expB, \nu, expweight, p_s, 2k, H, ubp_{pr}, g, maxit, tol)$

    $Dterms = Dterms + poly\_high(\beta, P)D$
    $u = D$
end for
$u^{n+1} = 2u^n - u^{n-1} + Dterms$

3.3 Appropriate Initial Condition

Our MOLT based implicit numerical scheme computes solution u^{n+1} at time t_{n+1} using the solutions u^n and u^{n-1} at previous time steps t_n, and t_{n-1}. Therefore our initial condition should be handled carefully in order to avoid order reduction of the scheme. In Causley's higher order paper [14] he introduced a technique to approximate u^1 at time t_1 using the initial solution at time t_0, u^0.

However, since our high-order outflow scheme requires solutions u^n, u^{n-1}, u^{n-2}, u^{n-3}, and u^{n-4} at the boundary points for the time levels t_n, t_{n-1}, t_{n-2}, t_{n-3}, and t_{n-4} to compute homogeneous coefficients, the above approximation is not enough to go with high-order outflow scheme. Therefore, we require the initial solution u^1, u^2, u^3, and u^4 to use our higher order scheme. In order to obtain those values, we first use the second order scheme with the initial approximation of u^1 and obtain u^2, u^3, and u^4, then afterwords we use our

Algorithm 12 *linv_out*: Compute $\mathcal{L}^{-1}$ for outflow boundary condition

u - array of u values at time t_n,
β - averaging parameter,
$bdry_A, bdry_B$ - boundary points,
expA expB- exponential vector of grid distance from left boundary and right boundary respectively along the line,
ν - array of weighted nodes along the line,
expweight - array of exponentially weighted nodes along the line,
p_s -order of accuracy in space,
p_t - order of accuracy in time.
H - boundary coefficients at time step t_n,
ubp_{pr} - solution at the boundary points at previous time steps,
g - a vector of γ coefficients for a given order of accuracy,
maxit - maximum number of iterations,
tol - tolerance number for the convergence.
Compute particular solution
$I = fastconvolution(u, \nu, expweight, p_s)$
Boundary correction
$u_{temp} = u$
$it = 0$
repeat
    $H = apply\_bc\_outflow(u_A, u_B, bdry_A, bdry_B, \beta, H, ubp_{pr}, g, p_t)$
    $u_{temp} = update\_stencil(u_{temp}, I, expA, expB)$
    $it = it + 1$
until $(|u - u_{temp}|_\infty < tol)$ OR $(it < maxit)$
Operate $\mathcal{L}^{-1}$
$k = \frac{p_t}{2}$
$linv = I + H(k,1)expA + H(k,2)expB$

fourth order scheme to compute the solution u^5.

3.4 Numerical Results

In this section, several test cases are reported for higher-order accuracy in 2D and 3D.

3.4.1 Convergence Test for High-Order Outflow

We first show fourth order convergence of outflow boundary conditions by performing a refinement study on a rectangular domain $\Omega = [-1, 1] \times [-1, 1]$ and a cubic domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$ for 2D and 3D respectively. In each case, we use a point

Algorithm 13 *apply_bc_outflow*: Apply outflow boundary conditions

I_A, I_B - solution at boundary points,
$bdry_A, bdry_B$ - boundary points,
H - boundary coefficients at time step t_n,
ubp_{pr} - solution at the boundary points at previous time steps,
g - a vector of γ coefficients for a given order of accuracy,
β - averaging parameter,
p - order of accuracy in time.
$p = 2 * k$
$H(k,1) = H(k,1)e^{-\beta} + g(k,1)I_A$
$H(k,2) = H(k,2)e^{-\beta} + g(k,1)I_B$
for j = 2 to p +1 **do**
  for bp = 0 to 1 **do**
    $H(k,bp) = H(k,bp) + g(k,j)ubp_{pr}(p+2-j,bp)$
  end for
end for
for j = 1 to p **do**
  $ubp_{pr}(j,:) = ubp_{pr}(j+1,:)$
end for
$ubp_{pr}(p+1,0) = u_{xA}$
$ubp_{pr}(p+1,1) = u_{xB}$

Algorithm 14 *gamma* : Compute the γ coefficients for a given order of accuracy

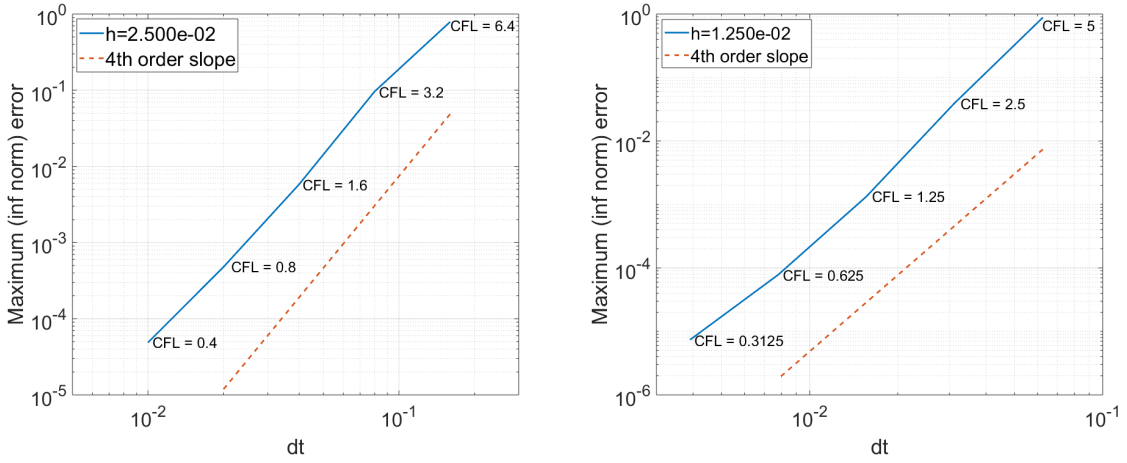
β - averaging parameter,
p - order of accuracy in time,
stencil - array of uniform grid points
for j = 1 to p **do**
  $cfs(:,j) = fdcoeffF(j,0,stencil)$
end for
for k = 1 to p+1 **do**
  for j = 1 to p **do**
    $g(k) = g(k) + (-1)^j E(j,\beta)$
  end for
end for
$g(1) = g(1) + E(0,\beta)$

source $cos(\omega t), \omega = 1$ placed at center of the domain ($(0,0)$ in 2D and $(0,0,0)$ in 3D), and run the evaluation up to dimensionless time $T = 2.0$, with a fixed spatial resolution of 160×160 spatial points in 2D and $160 \times 160 \times 160$ in 3D. The discrete L_∞ norm of the error is constructed at each time step and maximum error over all time steps is used to graph in Figure 3.1 for $\Delta t = \frac{125 \times 10^{-3}}{2^k}$, $k = 1$ to 5, with outflow boundary conditions. The

Algorithm 15 E : Compute the coefficients E based on lemma 2.1

1: m and v - integers,
2: $m \geq 0$ and $v > 0$
3: $P_{mv} = 0$
4: **for** l = 0 to m **do**
5: $\quad P_{mv} = P_{mv} + \frac{v^l}{l!}$
6: **end for**
7: $E_{mv} = \frac{1}{v^m}(1 - e^{-v})P_{mv}$

CFL ($= \frac{c\Delta t}{\Delta x}$) value changes proportional to the time step size Δt with fixed spatial step size, $\Delta x = \Delta y = \Delta z$.



(a) in 2D (b) in 3D

Figure 3.1: Fourth order convergence of (a) 2D and (b) 3D wave solver using outflow boundary conditions with the spatial step size $h = 1.25 \times 10^{-2}$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 2.0$ on a (a) square and (b) cubic domain with a point source $cos(\omega t), \omega = 1$ at the center of the domain. The CFL ($= \frac{c\Delta t}{h}$) value changes proportional to the time step size Δt with fixed spatial step size, h.

3.4.2 Rotating Gaussian Pulse

In this section, we examine the ability of our outflow scheme to handle the effect of a rotating Gaussian pulse. First, we perform refinement studies on a square domain $\Omega = [-1, 1] \times [-1, 1]$ for a rotating Gaussian pulse through different angles. We chose the Gaussian

pulse as an initial solution,

$$e^{-36\left(\left(\frac{cos(\theta)x+sin(\theta)y}{r1}\right)^2+\left(\frac{sin(\theta)x+cos(\theta)y}{r2}\right)^2\right)} \tag{3.8}$$

with $r_1 = 2$ and $r_2 = 1$ and rotate the angle θ from $\frac{\Pi}{18}$ to $\frac{\Pi}{2}$ for this refinement studies. We run the evaluation up to the dimensionless time $T = 2.0$, with a fixed spatial resolution of 100×100 spatial points. The discrete L_∞ norm of the error is constructed at each time step and maximum error over all time step is used to graph in Figure 3.2 for $\Delta t = \frac{0.2}{2^k}$, $k = 0$ to 4, with our fourth order outflow boundary conditions.

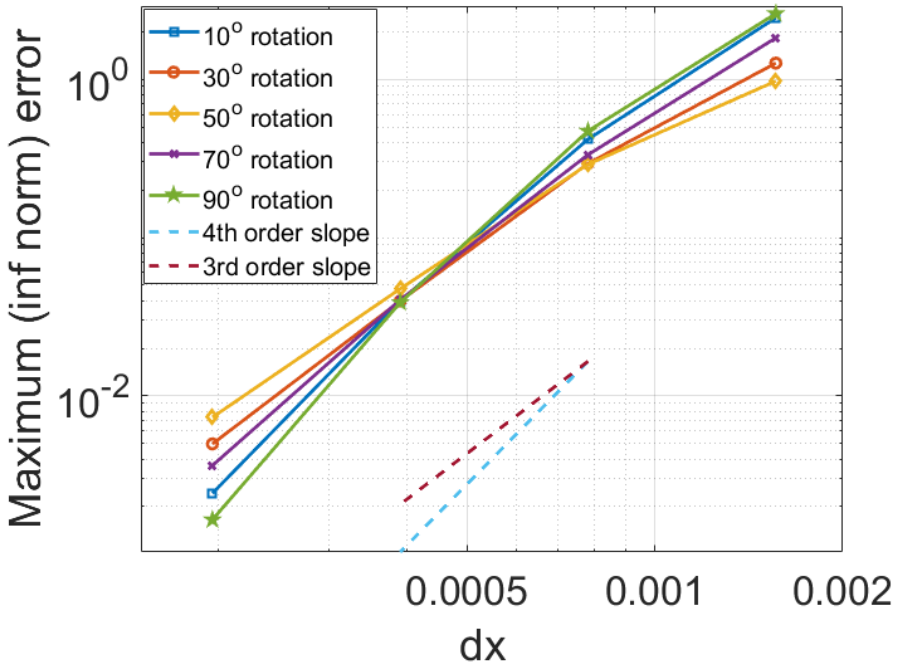
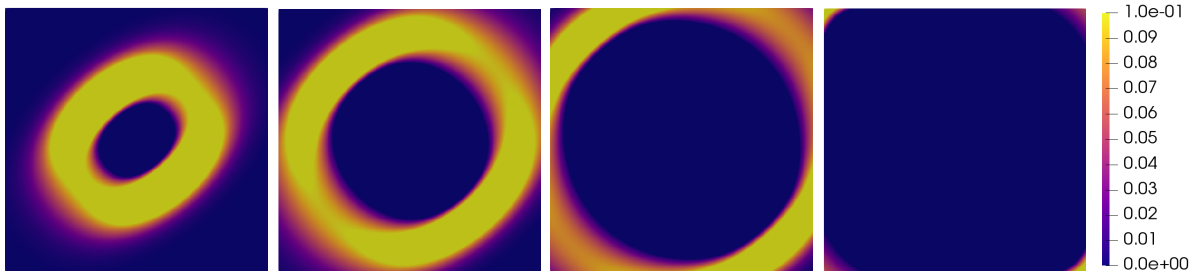


Figure 3.2: Fourth order convergence of the oval shape Gaussian pulse (however it reduces to the above third order near to $\theta = \frac{\Pi}{4}$ due to the ADI splitting error) given by the Equation 3.8 rotated through $\frac{\Pi}{18}$ to $\frac{\Pi}{2}$ on square domain $\Omega = [-1, 1]^2$ by imposing Outflow boundary conditions along the boundaries. This self-refinement study measures L_∞ norm of the error at time $T = 2.0$ with $h = 0.02$.

The convergence plot shows that the order of the accuracy reduces from fourth-order to above third-order while the angle goes to $\frac{\Pi}{4}$. This is acceptable behavior because the highest splitting error should be obtained for the angle $\frac{\Pi}{4}$ due to the ADI splitting.

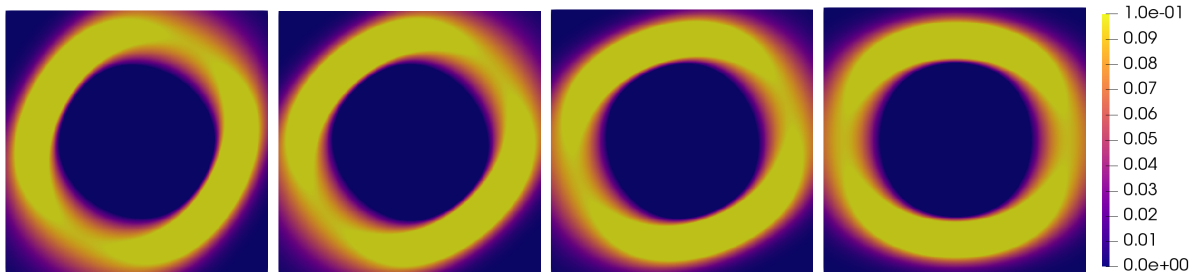
3.4.2.1 Time Evaluation of 50^o angled Gaussian pulse

In this section, we present a time evolution of the same Gaussian pulse defined in equation 3.8 with $r_1 = 2, r_2 = 1$ and angle $\theta = 50^o$ using outflow boundary conditions. We chose a rectangular domain $\Omega = [-1, 1] \times [-1, 1]$ with spatial resolution of 300×300 spatial points, and set the time step size $\Delta t = 3.3 \times 10^{-3}$ to maintain CFL value 0.5. Figure 3.3 shows snapshots of the wave at different time instants



(a) t = 0.3597 (b) t = 0.6864 (c) t = 1.0131 (d) t = 1.3695

Figure 3.3: Time evolution of a Gaussian pulse given by the Equation 3.8 with angle $\theta = 50^o$ on square domain $\Omega = [-1, 1]^2$ by imposing outflow boundary conditions along the boundaries. Here the time step size $\Delta t = 3.3 \times 10^{-3}$ and CFL value is 0.5. The wave is leaving completely as expected.



(a) $\theta = 30^o$ (b) $\theta = 50^o$ (c) $\theta = 70^o$ (d) $\theta = 90^o$

Figure 3.4: Evolution of a Gaussian field given by the Equation 3.8 at time t = 0.6864, rotated through different angles (a) $\theta = 30^o$, (b) $\theta = 50^o$, (c) $\theta = 70^o$, and (d) $\theta = 90^o$ on square domain $\Omega = [-1, 1]^2$ by imposing outflow boundary conditions along the boundaries, Here the time step size $\Delta t = 3.3 \times 10^{-3}$ and CFL value is 0.5 at time t = 0.1287 using outflow boundary conditions. We observe the same behaviour of the wave for rotated Gaussian pulse through different angles. The waves leave completely through the curved boundary.

3.4.3 Outflow Boundary Condition Along Curve Boundaries

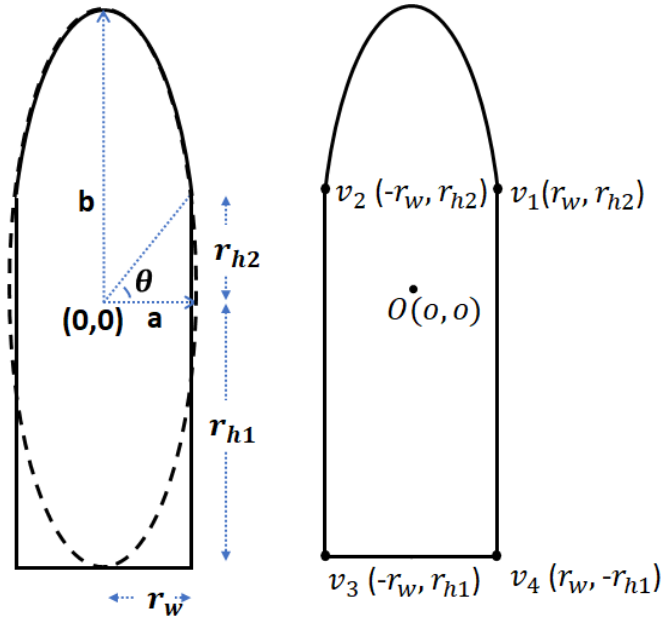
We performed an experiment to examine the ability of our outflow boundary treatment along the curved boundaries. For this experiment, we built a 2D geometry with a curved boundary as shown in Figure 3.5 (a). The curve is a portion of an ellipse which has axes a and b and centred at the origin $(0, 0)$.

$$r_w = \frac{ab}{\sqrt{b^2 + a^2 \tan^2 \theta}}$$

$$r_{h2} = r_w \tan \theta$$

$$r_{h1} = b$$

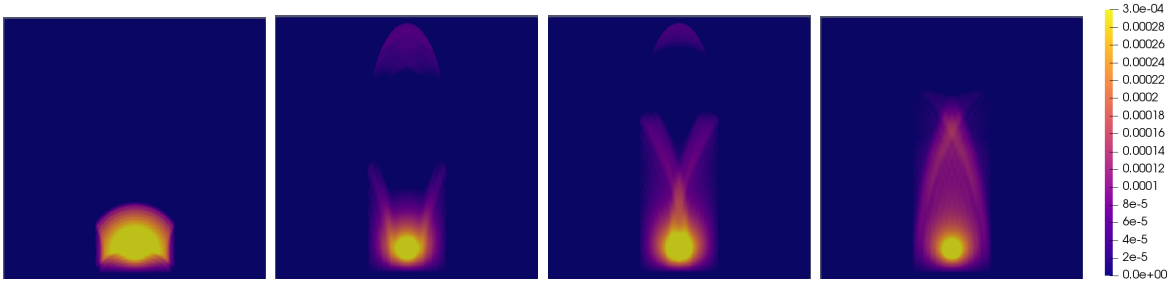
This 2D object can be represented as a 2D graph (Figure 3.5), $G(V, E(E_S, E_A))$ with a set of vertices $V = (v_1, v_2, v_3, v_4)$, straight edges $E_S = [e_{s1}(\equiv (v_2, v_3)), e_{s2}(\equiv (v_3, v_4)), e_{s3}(\equiv (v_3, v_4)), e_{s4}(\equiv (v_4, v_1))]$, and an arch edge $E_A = [e_{a1}(\equiv (v_1, v_2, O))]$.



(a) Geometry (b) Graph

Figure 3.5: (a) Geometry and (b) graph representation of the 2D object constructed by using an oval of width $2a$ and height $2b$ and a rectangle of $2r_w \times (r_{h1} + r_{h2})$. Here $r_{h1} = b$ and v_1, v_2, v_3, and v_4 are the vertices of the object.

We made an instant of this object with $a = 1.6$, $b = 4.8$, and $\theta = 50^o$, place a point source $\cos \omega t$ with $\omega = 0.5$ at $O(0, 100\Delta x)$, apply outflow boundary condition along the elliptical curved edge and Dirichlet along the remaining straight edges. Time evaluation of the source field for the spatial grid width $\Delta x = \Delta y = 0.039$ and time step size $\Delta t = 0.0195$ is shown in Figure 3.6



(a) $t = 1.8135s$ (b) $t = 11.778s$ (c) $t = 12.6945s$ (d) $t = 13.7085s$

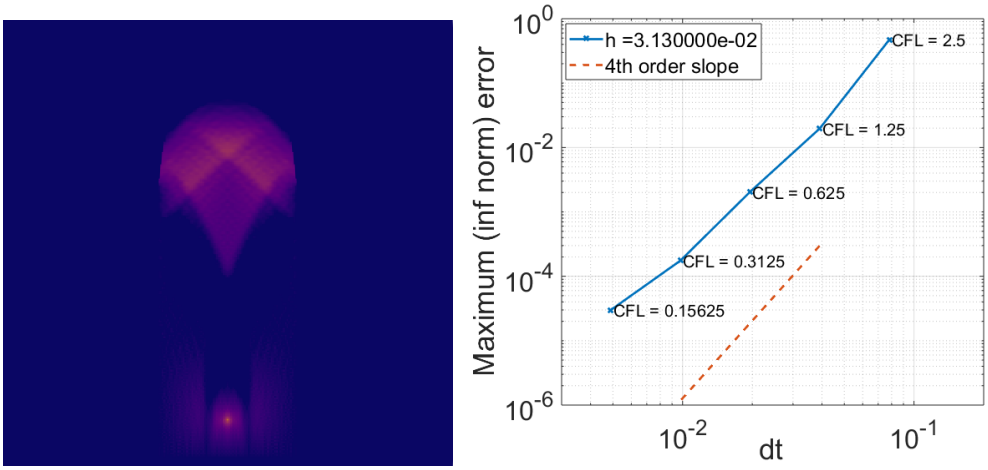
Figure 3.6: Time evolution of the point source field $\cos \omega t$ with $\omega = 0.5$ at $(0, 100\Delta x)$ in the object as shown in Figure 3.5 by imposing outflow boundary conditions along the curved boundary and homogeneous Dirichlet along the straight boundaries. Here the spatial step size, $\Delta x = \Delta y = 0.039$ and the time step size, $\Delta t = 0.0195$

Further, we performed refinement studies for the same structure and obtained the fourth-order accuracy in time by computing the L_∞ error for the final time $T = 5.0$ as shown in the plot 3.7.

3.4.4 Convergence Studies Using Analytical Solution

We perform this experiment to confirm the order of accuracy of our scheme using an analytical solution. We use the fourth order scheme (see [14] for the proof) as given in Equation 3.9 which includes the source term s,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xyz}[u^n] - \left(\beta^2 \mathcal{D}_{xyz} - \frac{\beta^4}{12}\mathcal{C}_{xyz}\right)\mathcal{C}_{xyz}[u^n] + \frac{\beta^2}{12\alpha^2}\left(s^{n+1} + 10s^n + s^{n-1}\right) + \frac{\beta^2}{12\alpha^4}\mathcal{C}_{xyz}[s^n] \tag{3.9}$$



(a) Field at $t = 5.0$ (b) Convergence plot

Figure 3.7: (a) Snapshot of the point source field at time $T = 5.0$ and (b) Fourth order convergence of 2D wave solver using Outflow boundary conditions along the curve boundary with the spatial grid width $\Delta x = \Delta y = 0.078$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 5.0$ on the object shown in Figure 3.5 with a point source $cos(\omega t), \omega = 0.5$ at the center of the box, $(0, 100\Delta x)$

Here we use three dimensional integration operator $\mathcal{C}_{xyz}$ acting on the source term s for a high-order correction. We derive an analytical solution for a point source $sin(4\pi t)$ placed at (x_0, y_0, z_0) as follows,

$$\frac{1}{4\pi} \frac{e^{-36(t/15-1).^2} sin(4\pi t)}{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 + \delta^2}} \tag{3.10}$$

where $\delta \in \mathcal{R}^+$, $0 < \delta^2 < \Delta x$

We choose a cubic domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$, place a point source $sin(4\pi t)$ at the center $(0, 0, 0)$, impose outflow boundary conditions along the all six boundary surfaces, and run the evaluation up to dimensionless time $T = 3.7$, with a fixed spatial resolution of $80 \times 80 \times 80$. The discrete L_∞ norm of the error is constructed at each time step and maximum error over all time steps is used to graph in Figure 3.8 for $\Delta t = \frac{2\times 10^{-1}}{2^k}$, $k = 0$ to 3.

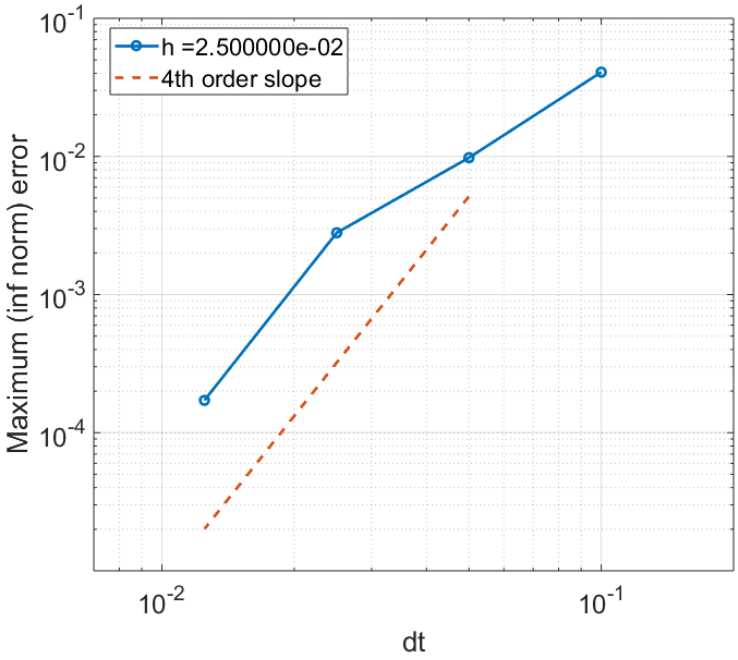


Figure 3.8: Fourth order convergence of 3D wave solver using outflow boundary conditions with the spatial step size $h = 2.5 \times 10^{-2}$. This measures L_∞ norm of the error which compares with analytical solution using a point source $sin(4\pi t)$ placed at the center of the domain.

3.5 Summary

We derived a solution for high-order outflow boundary conditions using our MOLT based scheme which uses the extended backward finite difference time-stencil along the boundaries and applies a suitable initial condition. The computation begins with the second-order scheme followed by the 4th-order scheme after three time-steps. We evaluated the scheme using several test cases including curved boundaries. In the next chapter, we describe the derivation of the high-order embedded Neumann boundary method for multi-dimensions.

CHAPTER 4: Higher Order Embedded Neumann Boundary Method

4.1 Introduction

In this chapter, we review the second-order embedded Neumann boundary method for the scheme described in Chapter 2 and extend the order of spatiotemporal accuracy for this scheme to arbitrary high-order. Further, we develop a general framework for complex geometries in 3D.

We choose an embedded boundary method to deal with complicated geometries. As an initial test for the simulation of magnetrons, we impose the embedded Neumann boundary condition (later, in Chapter 5 we introduce our embedded PEC boundary condition). The MOLT based scheme described in Chapter 2 gives second-order accuracy for embedded Neumann boundary condition, therefore we want to improve the accuracy for arbitrary higher-order in time and space. In this chapter, we describe a high-order 3D embedded Neumann boundary method and generalize it for complex geometries in 3D. The high-order accuracy in time is achieved using the Lax-Wendroff approach as explained previously in Chapter 2. Our treatment for embedded Neumann boundary condition includes the ghost points and boundary corrections. First, the solution at the ghost points are computed using Hermite-Birkhoff interpolation, and then interior points were updated using a boundary correction iteration along with boundary stencils. Hence, we extend the interpolation stencils to achieve high-order accuracy in space.

The embedded boundary method can deal with curved boundaries and material interfaces by superimposing boundaries on Cartesian grids and it also can handle off-grid points. Thus, the embedded boundary method is the most suitable approach for complicated geometries in different application areas such as fluid dynamics in aerospace engineering [55; 44],

electromagnetics [3], and acoustics [45; 2]. Further, several material interface problems are dealt with embedded boundary methods such as dielectric materials. Here, the question is how to handle wave propagation along with material/domain interfaces. Researchers propose appropriate jump conditions [11; 3; 46; 22] which were derived from the domain's material properties and apply the suitable boundary conditions along the interfaces using interpolation techniques. In [46], the embedded method was used for numerical modeling of electromagnetic scattering of an incident plane wave by a dielectric circular cylinder. They dealt with discontinuous wave propagation speed using appropriate jump conditions, but the scheme is in second-order temporal and spatial accuracy for 2D problems. In [11] and [18] second and fourth-order accuracy was achieved using an upwinding embedded boundary method for Maxwell's equations based on Runge-Kutta type time discretization. However, the approach is very cost expensive and is not as fast as our MOLT based scheme. Another method proposed in [3] uses interior boundary points instead of exterior ghost points to apply Neumann or Dirichlet boundary conditions with the embedded boundary method and gives fourth-order accuracy for the wave equation in 2D. The method is based on a compact Pade-type discretization of spatial derivatives together with the Taylor series method in time and it can remove small-cell stiffness problems for both Neumann or Dirichlet boundary conditions. However, it also deals with cost expensive and slow matrix operations to obtain the solution.

An approach similar to our scheme has been used in [9; 8]. They provide an implicit ADI based numerical scheme for solving the heat and wave equations in a general embedded-boundary domain with high-order spatial accuracy using Fourier continuation spatial approximations and second-order temporal accuracy. Thus, the big difference from our approach is that they use a Fourier-based method to invert the semi-discrete differential operator instead of constructing and inverting the modified Helmholtz operator as we do. Besides, the Fourier continuation method allows for minimal dispersion errors for wave propagation problems. However, the schemes are only second-order accurate in time and higher-order

accuracy requires resorting to Richardson extrapolations.

In the following sections, we describe the higher-order Neumann boundary conditions in Section 4.2, detailing its implementation in 1D, 2D, and 3D. In Section 4.3, we describe the generalized embedded method suitable for any complex geometry and finally provide numerical results.

4.2 Derivation of The Scheme

Neumann boundary conditions for boundary geometries along grid lines can be directly imposed using a two-point boundary correction by utilizing surface grids in 2D and volumetric grids in 3D. The method can be applied to polygonal domains by the use of multiple overset grids, each aligned with a boundary segment, which communicates with the interior grid through interpolation on a ghost cell region. We chose the embedded boundary methods explained in [13] as the basis to impose Neumann boundary conditions for curved boundary surfaces in 3D. Here we determine the corresponding Dirichlet boundary values at the endpoints of each $x-$, $y-$ and $z-$ sweep lines that result in the approximate satisfaction of the Neumann boundary condition.

The Neumann boundary condition is unconditionally unstable even when space is set continuously [46; 22]. This means all embedded boundary methods need a touch of diffusion to stabilize it. In our earlier work, we developed a $\mathcal{O}(\Delta t^2)$ with one higher-order diffusion term to stabilize it. Here we extend the diffusion term to the fourth-order accuracy.

4.2.1 One Dimensional Scheme

Let us consider 1D domain $\Omega \equiv [x_a, x_b]$ with a left boundary x_B, and we choose a ghost point x_G, which is the exterior-point to the simulation domain (it should have at least one interior neighbor). We give the formulation of the left domain in detail next,

As shown in Figure 4.1, x_I and x_{II} are two interior points, their distance from boundary

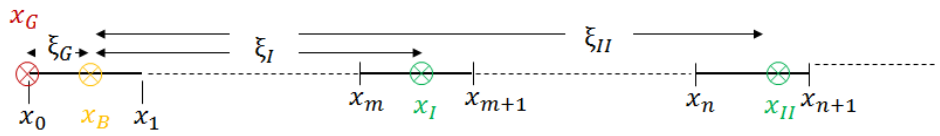


Figure 4.1: Geometry of 1D embedded Neumann boundary domain $\Omega \equiv [x_a, x_b]$ with left boundary x_B, ghost point x_G, and interpolation points x_I and x_{II}, where the distance $\xi_I = |x_I - x_a|$, $\xi_{II} = |x_{II} - x_a|$, and $\xi_G = |x_G - x_a|$

point are $\xi_I = |x_I - x_a|$ and $\xi_{II} = |x_{II} - x_a|$ along the normal. We define ξ_G as the distance between ghost and boundary point ($\xi_G = |x_G - x_a|$). We apply Hermite-Birkhoff interpolate, $P(\xi)$ through the points, x_G, x_a, x_I, and x_{II} by imposing the conditions $P'(0) = 0$, $P(\xi_I) = u_I$ and $P(\xi_{II}) = u_{II}$, and explicitly solving for u_G gives:

$$u_G^{l+1} = \left(\frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2}\right) u_I^l + \left(\frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2}\right) u_{II}^l + O(\Delta x^2),$$

We have introduced additional notation of $l+1$ and l, which designate how we will set up a fixed point iteration using the Hermite-Birkhoff interpolate. The iteration itself is a simple first order fixed point method. Let $\mathbf{w}^{(l)}$ and $\mathbf{w}^{(l+1)}$ be the solutions at the l^{th} and $(l+1)^{th}$ iterations. We choose a tolerance tol, and maximum number of allowed iterations mit, and define a stopping criterion as $|\mathbf{w}^{(l+1)} - \mathbf{w}^{(l)}|_\infty < tol$ OR $nit > mit$ where nit is the current iteration number. The detailed steps for this iterative approach are given in Algorithm 16,

As detailed in [13] we need to include an artificial dissipation term into the solution to maintain stability for such an embedded boundary method. Thus the second-order solution $u^{n+1}(x)$ at the next time step looks like,

$$u^{n+1}(x) - 2u^n(x) + u^{n-1}(x) = -\beta^2 \mathcal{D}_x^{(1)}[u^n] - \epsilon \mathcal{D}_x^{(2)}[u^{n-1}]. \tag{4.1}$$

where ϵ is an artificial dissipation coefficient, that satisfies $0 < \epsilon < 1$. The Value of $\mathcal{D}_x^{(2)}[u^n]$ at the previous time step can be used as $\mathcal{D}_x^{(2)}[u^{n-1}]$ at the current time step, and we need to go with one more computing-level to obtain $\mathcal{D}_x^{(2)}[u^n]$. According to [13], β has to be reduced

Algorithm 16 Compute L^{-1} for 1D scheme

1: Compute particular solution $I^{n+1} = I[u^n]$
2: Initial guess $\mathbf{w}^{n+1(0)} \approx 3u^n - 3u^{n-1} + u^{n-2}$
3: Initialize the iteration counter $nit = 0$
4: **repeat**
5: Compute $\mathbf{w}_I^{n+1(l)}$ and $\mathbf{w}_{II}^{n+1(l)}$, at the interpolation points x_I, and x_{II} and as well as at the interpolation points along the right boundary
6: Compute solution at the ghost points using the following Hermite-Birkhoff interpolant $\mathbf{w}_G^{n+1(l)} = \left(\frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2}\right) \mathbf{w}_I^{n+1(l)} + \left(\frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2}\right) \mathbf{w}_{II}^{n+1(l)}$
7: Compute homogeneous coefficients a_1 and b_1 using the fact (Here, x_{aG} and x_{bG} are ghost points), $\mathbf{w}^{n+1(l)} = I^{n+1} + a_1 e^{-\alpha(x - x_{aG})} + b_1 e^{-\alpha(x_{bG} - x)}$.
8: Compute solution and update boundary stencil $\mathbf{w}^{n+1(l+1)} = I^{n+1} + a_1 e^{-\alpha(x - x_{aG})} + b_1 e^{-\alpha(x_{bG} - x)}$
9: $nit = nit + 1$
10: **until** $(|\mathbf{w}^{n+1(l+1)} - \mathbf{w}^{n+1(l)}| < tol$ OR $nit > mit)$

by a small amount from the case published in [14].

4.2.2 Two Dimensional Scheme

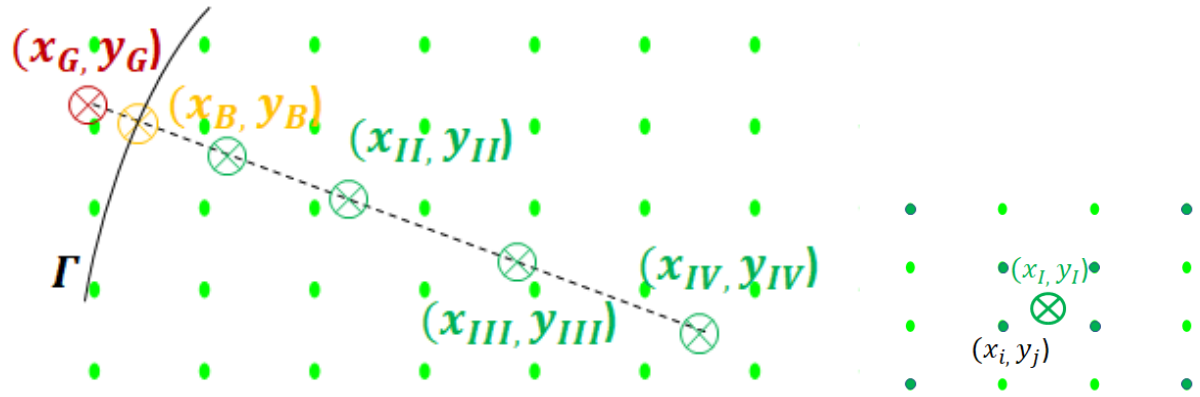
When extending the embedded Neumann boundary method to 2D cases, we need to consider two important changes: every spatial point will be treated in x and y Cartesian coordinates, and the ghost point approximation will be made by using bi-linear interpolation. To achieve high-order accuracy, we need more computing-levels (perform P computing-levels for $2P$-th order of accuracy in time) and extend the interpolation stencils by using 4-points along the normal for the Hermite-Birkhoff interpolation and an eight-point stencil for the bi-linear interpolation. We can derive second-order and fourth-order 2D solution with artificial dissipation ([13]) as given below,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xy}^{(1)}[u^n] + \epsilon \mathcal{C}_{xy}^{(2)}[u^{n-1}], \tag{4.2}$$

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xy}^{(1)}[u^n] - \left(\beta^2 \mathcal{D}_{xy}^{(2)} - \frac{\beta^4}{12} \mathcal{C}_{xy}^{(2)}\right) \mathcal{C}_{xy}^{(1)}[u^n] + \epsilon \mathcal{C}_{xy}^{(3)}[u^{n-1}]. \tag{4.3}$$

We now consider the situation of a two-dimensional domain with a curved boundary Γ

displayed in Figure 4.2. In the 2D case, we define a ghost point (x_G, y_G) which is an exterior point (x_i, y_j), but at least one of the neighboring points $(x_{i\pm1}, y_j)$ or $(x_i, y_{j\pm1})$ should be an interior point. In order to compute the value of unknown u_G at the ghost point location (x_{aG}, y_{aG}), we construct a quadratic Hermite-Birkhoff boundary interpolant $P(\xi)$ along the direction normal to the boundary, which intersects the boundary curve Γ at location (x_a, y_a). We choose four interior points selected along the normal of the surface with distances, $\xi_I = |(x_I, y_I) - (x_a, y_a)| = \Delta s_I$, $\xi_{II} = |(x_{II}, y_{II}) - (x_a, y_a)| = 2\Delta s_I$, $\xi_{III} = |(x_{III}, y_{III}) - (x_a, y_a)| = 3\Delta s_I$ and $\xi_{IV} = |(x_{IV}, y_{IV}) - (x_a, y_a)| = 4\Delta s_I$, where we will typically take $\Delta s_I = \sqrt{2}\Delta x$, and compute values u_I, u_{II}, u_{III}, and u_{IV} at points (x_I, y_I), (x_{II}, y_{II}), (x_{III}, y_{III}), and (x_{IV}, y_{IV}) respectively by interpolating from interior grid points, and use these interpolation values u_I, u_{II}, u_{III}, and u_{IV} to perform the Hermite-Birkhoff interpolant $P(\xi)$. Further, the distance from the boundary to the ghost point is defined as $\xi_G = |(x_G, y_G) - (x_a, y_a)|$.



(a) Geometry (b) 8-points stencil

Figure 4.2: (a)Geometry of 2D embedded Neumann boundary method with a boundary point (x_B, y_B), ghost point (x_G, y_G), and interpolation points (x_I, y_I), (x_{II}, y_{II}), (x_{III}, y_{III}) and (x_{IV}, y_{IV}) and (b) symmetric 8-points stencil to interpolate the solution at the point (x_I, y_I).

The solution at the ghost point, u_G can be computed using the Hermite-Birkhoff interpolant $P(\xi)$ by imposing the conditions $P'(0) = 0$, $P(\xi_I) = u_I$, $P(\xi_{II}) = u_{II}$, $P(\xi_{III}) = u_{III}$,

and $P(\xi_{IV}) = u_{IV}$ given by,

$$u_G^{l+1} = \left(\frac{\xi_{III}^2 - \xi_G^2}{\xi_{III}^2 - \xi_I^2}\right) u_I^l + \left(\frac{\xi_{IV}^2 - \xi_G^2}{\xi_{IV}^2 - \xi_{II}^2}\right) u_{II}^l + \left(\frac{\xi_G^2 - \xi_I^2}{\xi_{III}^2 - \xi_I^2}\right) u_{III}^l + \left(\frac{\xi_G^2 - \xi_{II}^2}{\xi_{IV}^2 - \xi_{II}^2}\right) u_{IV}^l + O(\Delta x^2),$$

We obtain u_I and u_{II} by the approximations using the symmetric 8-point stencil interpolation. Suppose the interpolation point u_I lies in a cell with corners (x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), and (x_i, y_{j+1}), we have the following approximation for u_I

$$\begin{aligned} u_I = {} & w_1 u_{i,j} + w_2 u_{i+1,j} + w_3 u_{i+1,j+1} + w_4 u_{i,j+1} \\ & + w_5 u_{i-1,j-1} + w_6 u_{i+2,j-1} + w_7 u_{i+2,j+2} + w_8 u_{i-1,j+2} \end{aligned} \tag{4.4}$$

where

$$\begin{aligned} w_1 &= \frac{(x_{i+2} - x_I)(y_{j+2} - y_I)}{4\Delta x \Delta y} & w_2 &= \frac{(x_I - x_{i-1})(y_{j+2} - y_I)}{4\Delta x \Delta y} \\ w_3 &= \frac{(x_I - x_{i-1})(y_I - y_{j-1})}{4\Delta x \Delta y} & w_4 &= \frac{(x_{i+2} - x_I)(y_I - y_{j-1})}{4\Delta x \Delta y} \\ w_5 &= \frac{(x_{i+1} - x_I)(y_{j+1} - y_I)}{4\Delta x \Delta y} & w_6 &= \frac{(x_I - x_i)(y_{j+1} - y_I)}{4\Delta x \Delta y} \\ w_7 &= \frac{(x_I - x_i)(y_I - y_j)}{4\Delta x \Delta y} & w_8 &= \frac{(x_{i+1} - x_I)(y_I - y_j)}{4\Delta x \Delta y} \end{aligned}$$

As described in 1D (4.2.1), we use an iterative scheme to improve the accuracy of the solution. For each term in the operator on the right hand side of the second order solution given by equation 4.2, or fourth order in time solution given by equation 4.3, we must identify the correct ghost point that allows that term to guarantee that the normal derivative is zero along the boundary. Taking the $\mathcal{C}^{(1)}[u^n]$ operator and expanding the operator out we have $\mathcal{C}^{(1)}[u^n] = \mathcal{L}_x^{-1}[u^n] + \mathcal{L}_y^{-1}[u^n] - \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[u^n] - \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}[u^n]$. The fixed point iteration identifies the ghost point for $\mathbf{w_x} = \mathcal{L}_x^{-1}[u^n]$, $\mathbf{w_y} = \mathcal{L}_y^{-1}[u^n]$, $\mathbf{w_{xy}} = \mathcal{L}_x^{-1}[\mathbf{w_y}]$, and $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$ such that when solving for the boundary correction terms the operator satisfies $\partial_n \mathbf{w_x}|_{\partial\Omega} = 0$, $\partial_n \mathbf{w_y}|_{\partial\Omega} = 0$, $\partial_n \mathbf{w_{xy}}|_{\partial\Omega} = 0$, $\partial_n \mathbf{w_{yx}}|_{\partial\Omega} = 0$.

Let us consider $\mathbf{w_x}$ and $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$, knowing that $\mathbf{w_y}$ and $\mathbf{w_{xy}}$ are similar. The iterative process for $\mathbf{w_x}$ starts by making an initial guess at the direct boundary values $\mathbf{w_x}|_{\partial\Omega}$, using an extrapolate in time ${\mathbf{w_x}}^{n+1,(0)} \approx 3{\mathbf{w_x}}^{n} - 3{\mathbf{w_x}}^{n-1} + {\mathbf{w_x}}^{n-2}$ at each boundary point (Algorithm 17).

Algorithm 17 Compute $\mathbf{w_x}(= L_x^{-1}[u])$

1: Compute the interior sweep (particular solution) $I_x^{n+1} = I_x[u^n]$
2: Initial guess ${\mathbf{w_x}}^{n+1(0)} \approx 3{\mathbf{w_x}}^{n} - 3{\mathbf{w_x}}^{n-1} + {\mathbf{w_x}}^{n-2}$
3: Initialize the iteration counter $nit = 0$
4: **repeat**
5: **for** $k = 1$ to n_y **do**
6: Compute $\mathbf{w_x}_{I(k)}$, and $\mathbf{w_x}_{II(k)}$, at the interpolation points $(x_{I(k)}, y_{(k)})$, and $(x_{II(k)}, y_{(k)})$ and as well as at the interpolation points along the right boundary using bilinear interpolation
7: Compute solution at the ghost points $(x_{aG(k)}, y_{(k)})$ and $(x_{bG(k)}, y_{(k)})$ using the Hermite-Birkhoff interpolant
8: Compute homogeneous coefficients $a_{1(k)}$ and $b_{1(k)}$ using the fact ${\mathbf{w_x}}^{n+1(l)} = I^{n+1} + a_{x(k)}e^{-\alpha(x-x_{aG(k)})} + b_{x(k)}e^{-\alpha(x_{bG(k)}-x)}$
9: Compute solution and update boundary stencil ${\mathbf{w_x}}^{n+1(l+1)} = I^{n+1} + a_{x(k)}e^{-\alpha(x-x_{aG(k)})} + b_{x(k)}e^{-\alpha(x_{bG(k)}-x)}$
10: **end for**
11: $nit = nit + 1$
12: **until** $(|{\mathbf{w_x}}^{n+1(l+1)} - {\mathbf{w_x}}^{n+1(l)}| < tol$ OR $nit > mit)$

Given the update on $\mathbf{w_x}$, the next step is to consider $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$. The process starts with the initial guess ${\mathbf{w_x}}^{n+1,(0)} \approx 3u^n - 3u^{n-1} + u^{n-2}$ at the boundary (Algorithm 18).

The same process is done for $\mathbf{w_y}$ and $\mathbf{w_{xy}}$. The Hermite-Birkhoff interpolate enforces that the normal derivative is zero for $\mathbf{w_x}$, $\mathbf{w_y}$, $\mathbf{w_{xy}}$ and $\mathbf{w_{yx}}$ such that $\mathcal{C}^{(1)}[u^n]$ satisfies the normal derivative condition on $\partial\Omega$ to within tolerance. For the fourth order formulation, we compute the boundary conditions for $\mathcal{C}^{(1)}[u^n]$ and then we repeat the process for $\mathcal{C}^{(2)}$ and $\mathcal{D}^{(2)}$ acting on $\mathcal{C}^{(1)}[u^n]$.

Next, we will eamine the higher order embedded Neumann boundary method for 3D in detail,

Algorithm 18 Compute $\mathbf{w_{yx}}(= L_y^{-1}[\mathbf{w_x}])$

1: Compute the interior sweep (particular solution) $I_y^{n+1} = I_y[\mathbf{w_x}^n]$
2: Initial guess $\mathbf{w_{yx}}^{n+1(0)} \approx 3u^n - 3u^{n-1} + u^{n-2}$
3: Initialize the iteration counter $nit = 0$
4: **repeat**
5: **for** $k = 1$ to n_x **do**
6: Compute $\mathbf{w_{yx}}_{I(k)}$, and $\mathbf{w_{yx}}_{II(k)}$, at the interpolation points $(x_{(k)}, y_{I(k)}$, and $(x_{(k)}, y_{II(k)})$ and as well as at the interpolation points along the right boundary using bilinear interpolation
7: Compute solution at the ghost points $(x_{(k)}, y_{aG(k)})$ and $(x_{(k)}, y_{bG(k)})$ using the Hermite-Birkhoff interpolant
8: Compute homogeneous coefficients $a_{1(k)}$ and $b_{1(k)}$ using the fact $\mathbf{w_{yx}}^{n+1(l)} = I^{n+1} + a_{y(k)}e^{-\alpha(y-y_{aG(k)})} + b_{y(k)}e^{-\alpha(y_{bG(k)}-y)}$
9: Compute solution and update boundary stencil $\mathbf{w_{yx}}^{n+1(l+1)} = I^{n+1} + a_{y(k)}e^{-\alpha(y-y_{aG(k)})} + b_{y(k)}e^{-\alpha(y_{bG(k)}-y)}$
10: **end for**
11: $nit = nit + 1$
12: **until** $(|\mathbf{w_{yx}}^{n+1(l+1)} - \mathbf{w_{yx}}^{n+1(l)}| < tol$ OR $nit > mit)$

4.2.3 Three Dimensional Scheme

We now consider the situation of a three-dimensional domain with a curved-surface boundary Γ displayed in Figure 4.3. In the 3D case, we define a ghost point (x_{aG}, y_{aG}, z_{aG}) which is an exterior point (x_i, y_j, z_k), but at least one of the neighbouring points $(x_{i\pm1}, y_j, z_k)$ or $(x_i, y_{j\pm1}, z_k)$ or $(x_i, y_j, zk \pm 1)$ of it should be an interior point. As described in the one and two dimensional cases, to compute u_G, we construct a quadratic Hermite-Birkhoff boundary interpolant $P(\xi)$ along the direction normal to the boundary surface, which intersects the boundary curved-surface Γ at location (x_a, y_a, z_a). We choose two interior points selected along the normal of the surface with distances, $\xi_I = |(x_I, y_I, z_I) - (x_a, y_a, z_a) = \Delta s_I|$ and $\xi_{II} = |(x_{II}, y_{II}, z_{II}) - (x_a, y_a, z_a) = 2\Delta s_I|$, where we will typically take $\Delta s_I = \sqrt{3}\Delta x$, and compute values u_I and u_{II}, at points (x_I, y_I, z_I) and (x_{II}, y_{II}, z_{II}) respectively by interpolating from interior grid points, and use these interpolation values u_I and u_{II} to perform the Hermite-Birkhoff interpolant $P(\xi)$. Further, the distance from the boundary to the ghost point is defined as $\xi_G = |((x_{aG}, y_{aG}, z_{aG})) - (x_a, y_a, z_a)|$.

The solution at ghost point, u_G can be computed using the Hermite-Birkhoff interpolant

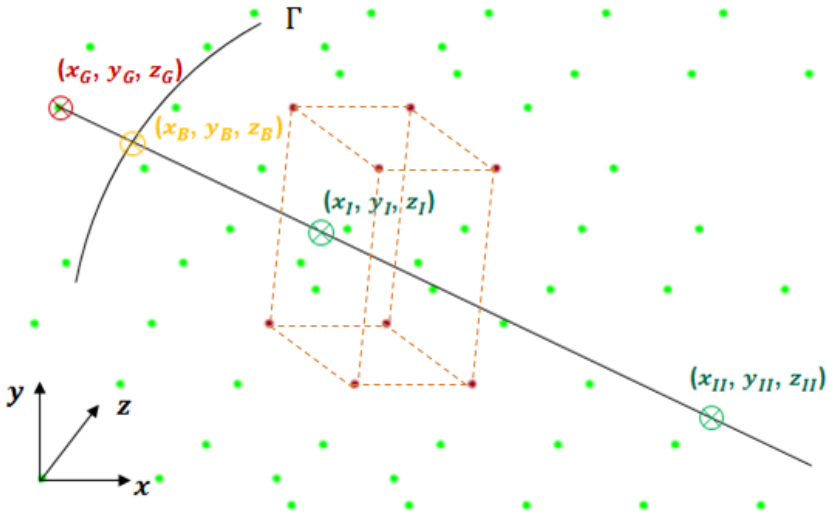


Figure 4.3: Geometry of 3D embedded Neumann boundary method with a boundary point (x_B, y_B, z_B), ghost point (x_G, y_G, z_G), and interpolation points (x_I, y_I, z_I), (x_{II}, y_{II}, z_{II}), $(x_{III}, y_{III}, z_{III})$ and (x_{IV}, y_{IV}, z_{IV})

$P(\xi)$ by imposing the conditions $P'(0) = 0$, $P(\xi_I) = u_I$, and $P(\xi_{II}) = u_{II}$, given by,

$$u_G^{l+1} = \left(\frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2}\right) u_I^l + \left(\frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2}\right) u_{II}^l + O(\Delta x^2),$$

We obtain u_I and u_{II} by the approximations using the tri-linear interpolation. Suppose the interpolation point u_I lies in a cell of rectangular prism with corners (x_i, y_j, z_k), (x_{i+1}, y_j, z_k), (x_{i+1}, y_{j+1}, z_k) , (x_i, y_{j+1}, z_k), (x_i, y_j, z_{k+1}), (x_{i+1}, y_j, z_{k+1}), $(x_{i+1}, y_{j+1}, z_{k+1})$, and (x_i, y_{j+1}, z_{k+1}), we have the following approximation for u_I

$$\begin{aligned} u_I = {} & w_1 u_{i,j,k} + w_2 u_{i+1,j,k} + w_3 u_{i+1,j+1,k} + w_4 u_{i,j+1,k} \\ & + w_5 u_{i,j,k+1} + w_6 u_{i+1,j,k+1} + w_7 u_{i+1,j+1,k+1} + w_8 u_{i,j+1,k+1} \end{aligned} \tag{4.5}$$

where

$$w_1 = \frac{(x_{i+1} - x_I)(y_{j+1} - y_I)(z_{k+1} - z_I)}{\Delta x \Delta y \Delta z} \qquad w_2 = \frac{(x_I - x_i)(y_{j+1} - y_I)(z_{k+1} - z_I)}{\Delta x \Delta y \Delta z}$$
$$w_3 = \frac{(x_I - x_i)(y_I - y_j)(z_{k+1} - z_I)}{\Delta x \Delta y \Delta z} \qquad w_4 = \frac{(x_{i+1} - x_I)(y_I - y_j)(z_{k+1} - z_I)}{\Delta x \Delta y \Delta z}$$
$$w_5 = \frac{(x_{i+1} - x_I)(y_{j+1} - y_I)(z_I - z_k)}{\Delta x \Delta y \Delta z} \qquad w_6 = \frac{(x_I - x_i)(y_{j+1} - y_I)(z_I - z_k)}{\Delta x \Delta y \Delta z}$$
$$w_7 = \frac{(x_I - x_i)(y_I - y_j)(z_I - z_k)}{\Delta x \Delta y \Delta z} \qquad w_8 = \frac{(x_{i+1} - x_I)(y_I - y_j)(z_I - z_k)}{\Delta x \Delta y \Delta z}$$

Three dimensional 2-nd and 4-th order schemes can be defined as follows,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}^{(1)}_{xyz}[u^n] + \epsilon \mathcal{C}^{(2)}_{xyz}[u^{n-1}], \tag{4.6}$$

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}^{(1)}_{xyz}[u^n] - \left(\beta^2 \mathcal{D}^{(2)}_{xyz} - \frac{\beta^4}{12}\mathcal{C}^{(2)}_{xyz}\right)\mathcal{C}^{(1)}_{xyz}[u^n] + \epsilon \mathcal{C}^{(3)}_{xyz}[u^{n-1}]. \tag{4.7}$$

We use the operators $\mathcal{C}$ and $\mathcal{D}$ in three dimensions, and need to operate $\mathcal{L}_z^{-1}$ in the $z-$ direction in addition to $\mathcal{L}_x^{-1}$ and $\mathcal{L}_y^{-1}$ to perform $z-$ sweeps. Taking the $\mathcal{C}^{(1)}[u^n]$ operator and expanding the operator out we have $\mathcal{C}^{(1)}[u^n] = \mathcal{L}_z^{-1}\mathcal{L}_y^{-1}[u^n] + \mathcal{L}_x^{-1}\mathcal{L}_z^{-1}[u^n] + \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[u^n] - \mathcal{L}_z^{-1}\mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[u^n] - \mathcal{L}_x^{-1}\mathcal{L}_z^{-1}\mathcal{L}_y^{-1}[u^n] - \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}\mathcal{L}_z^{-1}[u^n]$. The fixed point iteration identifies the ghost point for $\mathbf{w_x} = \mathcal{L}_x^{-1}[u^n]$, $\mathbf{w_y} = \mathcal{L}_y^{-1}[u^n]$, $\mathbf{w_z} = \mathcal{L}_z^{-1}[u^n]$, $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$, $\mathbf{w_{zy}} = \mathcal{L}_z^{-1}[\mathbf{w_y}]$, $\mathbf{w_{xz}} = \mathcal{L}_x^{-1}[\mathbf{w_z}]$, $\mathbf{w_{zyx}} = \mathcal{L}_z^{-1}[\mathbf{w_{yx}}]$, $\mathbf{w_{xzy}} = \mathcal{L}_x^{-1}[\mathbf{w_{zy}}]$, and $\mathbf{w_{yxz}} = \mathcal{L}_y^{-1}[\mathbf{w_{xz}}]$, and each of these operators satisfies $\partial_n(.)|_{\partial\Omega} = 0$ when solving for the boundary correction terms.

Let us consider $\mathbf{w_x}$, $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$ and $\mathbf{w_{zyx}} = \mathcal{L}_z^{-1}[\mathbf{w_{yx}}]$, knowing that computing of ($\mathbf{w_y}$, $\mathbf{w_{zy}}$, $\mathbf{w_{xzy}}$), and ($\mathbf{w_z}$, $\mathbf{w_{xz}}$, $\mathbf{w_{yxz}}$) are similar. The iterative process for $\mathbf{w_x}$ starts by making an initial guess at the direct boundary values $\mathbf{w_x}|_{\partial\Omega}$, using an extrapolation in time ${\mathbf{w_x}}^{n+1,(0)} \approx 3{\mathbf{w_x}}^n - 3{\mathbf{w_x}}^{n-1} + {\mathbf{w_x}}^{n-2}$ at each boundary point. Given the update on $\mathbf{w_x}$, the next step is to consider $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$. The process starts with the initial guess ${\mathbf{w_{yx}}}^{n+1,(0)} \approx 3{\mathbf{w_{yx}}}^n - 3{\mathbf{w_{yx}}}^{n-1} + {\mathbf{w_{yx}}}^{n-2}$ at the boundary, and then using the update on $\mathbf{w_{yx}}$, the following step is to consider $\mathbf{w_{zyx}} = \mathcal{L}_z^{-1}[\mathbf{w_{yx}}]$. The process starts with the initial guess

$\mathbf{w_{zyx}}^{n+1,(0)} \approx 3u^n - 3u^{n-1} + u^{n-2}$ at the boundary. We use three separate algorithms for each sweep $\mathcal{L}_x^{-1}$, $\mathcal{L}_y^{-1}$, and $\mathcal{L}_z^{-1}$, and each algorithm processes three major tasks: computation of a particular solution, a boundary correction, and then obtaining the value of actual $\mathcal{L}^{-1}$. We give the detailed steps for $\mathcal{L}_z^{-1}$ in Algorithm 19.

Algorithm 19 Compute $\mathbf{w_{zyx}}(= L_z^{-1}[\mathbf{w_{yx}}])$

Compute the interior sweep (particular solution) $I_z^{n+1} = I_z[\mathbf{w_{yx}}^n]$
Initial guess $\mathbf{w_{zyx}}^{n+1(0)} \approx 3u^n - 3u^{n-1} + u^{n-2}$
Initialize the iteration counter $nit = 0$
repeat
  for $i = 1$ to n_y **do**
    for $j = 1$ to n_x **do**
      Compute $\mathbf{w_{zyx}}_{I(i,j)}$, and $\mathbf{w_{zyx}}_{II(i,j)}$, at the interpolation points $(x_{(j)}, y_{(i)}, z_{I(i,j)})$, and $(x_{(j)}, y_{(i)}, (z_{II(i,j)})$ and as well as at the interpolation points along the right boundary using bi-linear interpolation
      Compute solution at the ghost points $(x_{(j)}, y_{(i)}, z_{aG(i,j)})$ and $(x_{(j)}, y_{(i)}, z_{bG(i,j)})$ using the Hermite-Birkhoff interpolant
      Compute homogeneous coefficients $a_{z(i,j)}$ and $b_{z(i,j)}$ using the fact $\mathbf{w_{zyx}}^{n+1(l)} = I^{n+1} + a_{z(i,j)}e^{-\alpha(z-z_{aG(i,j)})} + b_{z(i,j)}e^{-\alpha(z_{bG(i,j)}-z)}$
      Compute solution and update boundary stencil $\mathbf{w_{zyx}}^{n+1(l+1)} = I^{n+1} + a_{z(i,j)}e^{-\alpha(z-z_{aG(i,j)})} + b_{z(i,j)}e^{-\alpha(z_{bG(i,j)}-z)}$
    end for
  end for
  $nit = nit + 1$
until $(|\mathbf{w_{zyx}}^{n+1(l+1)} - \mathbf{w_{zyx}}^{n+1(l)}| < tol$ OR $nit > mit)$

Next, we are going to develop a generalized higher dimensional, higher-order scheme for problems with complex geometries.

4.3 Treatment for Complex Geometries

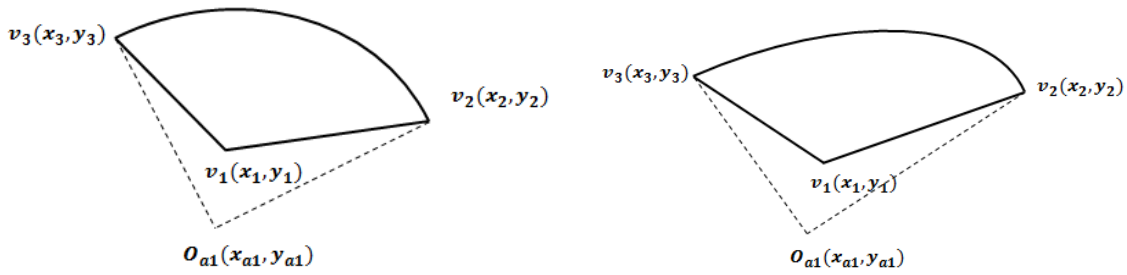
We now consider domains with complex geometries such as a model for A6 magnetron that has a set of arch areas joined together in 2D. The A6 magnetron has a complex geometry (Figure 1.2), and so it serves as a nice numerical test case for this boundary condition as opposed to a physical accurate boundary condition in a magnetron. We chose an embedded boundary method to solve such complex problems. The complex geometries in

higher-dimensional problems may be even harder. In our approach, however, since the higher dimensional problems are solved with ADI schemes, we need to worry about one-dimensional lines. Thus, we need to know all the relevant properties of each line to solve it. However, these lines are going to be broken into several segments due to complex boundaries. First, we compute boundary intersection points (which may be in off-grids), where the grid lines intersect with boundaries.

We give a detailed explanation for 3D complex geometries in this section. In our approach, first, we decompose higher-dimensional geometry to 2D slices and then represent it using 2D graphs with a set of vertices and edges. The edges can be classified as straight-line edges and curves or arches. The graph G can be given as,

$$G(V, E(E_S, E_A)), \quad E_S \equiv (v_{si}, v_{sj}), \quad and \quad E_A \equiv (O_{ij}, v_{ai}, v_{aj}) \tag{4.8}$$

where v_{si}, v_{sj}, v_{ai}, and v_{aj} are the vertices and O_{ij} is center of the arch (v_{ai}, v_{aj}) which may be a circular arch (4.4 (a)) or an elliptical arch (4.4 (b)). In Figure 4.4, each object has three vertices (v_1, v_2, v_3), two straight edges $e_{s1}(\equiv (v_1, v_2))$, $e_{s2}(\equiv (v_3, v_1))$, and one arch edge $e_{a1}(\equiv (v_2, v_3, O_{a1}))$ that can be represented as a graph $G([v_1, v_2, v_3], [[e_{s1}, e_{s2}], [e_{a1})]])$



(a) with circular arch (b) with elliptical arc

Figure 4.4: 2D graph with 3 vertices $(v_1(x_1, y_1)$, $v_2(x_2, y_2)$, and $v_3(x_3, y_3)$, 2 straight edges $e_{s1}(\equiv (v_1, v_2))$, $e_{s2}(\equiv (v_3, v_1))$ and 1 circular arch (a) or elliptical arch (b) edge $e_{a1}(\equiv (v_2, v_3, O_{a1}))$ centered at $O_{a1}(x_{a1}, ya1)$.

4.3.1 Pre-computing

Before beginning the PDE evolution, we need to perform a pre-computation to identify key characteristics of the geometry such as boundary and relevant parameter values along the grid lines in each direction. The flow diagram in Figure 4.5 shows major tasks performed in pre-computation and results obtained at the end of each task.

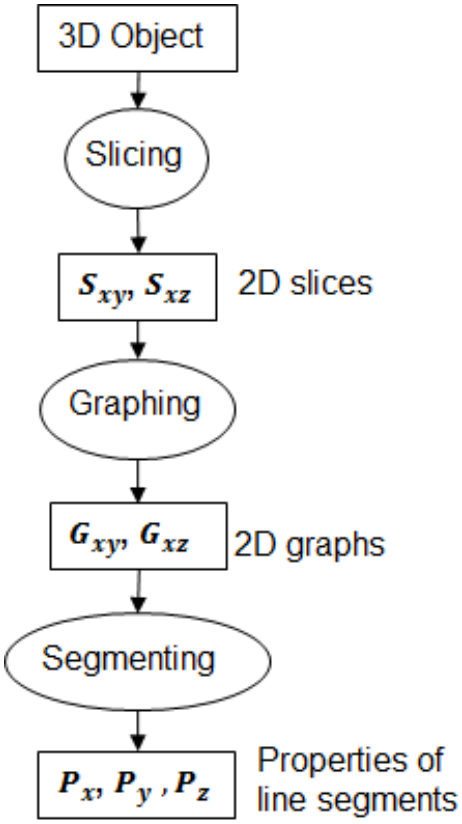


Figure 4.5: Flow diagram of pre-computation for 3D problems with complex geometries; it cuts a 3D object into 2D slices S_{xy}, S_{xz}, makes 2D graphs G_{xy}, G_{xz}, and identifies the properties of the line segments P_x, P_y, and P_z.

1. **Slicing**

   This task decomposes a given 3D object into a set of 2D objects by slicing along the grid lines. We can form three sets of 2D slices S_{yz}, S_{xz}, and S_{xy} in each direction x, y, and z respectively, but two sets are sufficient for our computation (if the object is uniform along any direction, we need to process slicing in that direction only - yielding

a set of slices). Suppose we apply the slicing along z and y directions, we will have a set of xy slices, S_{xy} placed along z with distance Δz apart and a set of xz slices, S_{xz} placed along y with distance Δy.

2. **Graphing**

   This task defines the geometry of each 2D slice using 2D graphs as expressed in equation 4.8. We have two sets of 2D slices, S_{xy} of size n_z and S_{xz} of size n_y where n_z and n_y are the number of spatial grid points along the z and y directions respectively. Each slice should have at least one 2D object. Suppose we assume a 3D object has only convex surfaces along with primary directions x, y, and z, then 2D objects in the slices will be defined by

$$G_{xy}(k), \quad k = 1, 2, ...n_z$$
$$G_{xz}(j), \quad j = 1, 2, ...n_y$$

   Therefore, we will obtain, $n_z + n_y$ graphs.

3. **Segmentation**

   This task generates line segments along each grid line in each direction x, y, and z using intersections between the grid lines and the surface of the object. The segments along x and y can be computed using the graph G_{xy} and along x and z can be computed using the graph G_{xz}, but we should avoid unneeded duplicate computations for x in order to reduce computing cost. In the final stage of the pre-computation all relevant parameters/properties P_x of size $n_x \times n_{sx}$, P_y of size $n_y \times n_{sy}$ and P_z of size $n_z \times n_{sz}$ for each line segment will be computed. Here n_{sx}, n_{sy} and n_{sz} are the number of line segments along x, y, and z respectively. Since, however, each line in the same direction does not need to have the same amount of segments, n_{sx}, n_{sy} and n_{sz} are not single variables, they are vectors/arrays to keep track of the number of segments in each line.

Now, the sizes of P_x, P_y, and P_z are $\sum_{i=1}^{n_x} n_{sx}(i)$, $\sum_{j=1}^{n_y} n_{sy}(j)$, and $\sum_{k=1}^{n_z} n_{sz}(k)$ respectively.

After we are done with the pre-computation, we will have three sets of line segments with their properties (P_x, P_y, and P_z) for each direction. These segments will be utilized by our higher-order solver to obtain a solution with higher-order accuracy. Actually, during each sweep, each line segment is treated individually with its own piecewise constant wave speed, and then the 3D higher-order equation 2.77 is applied to compute the solution u^{n+1} at the next time step t_{n+1}.

4.3.2 Geometry for A6 MDO in 2D

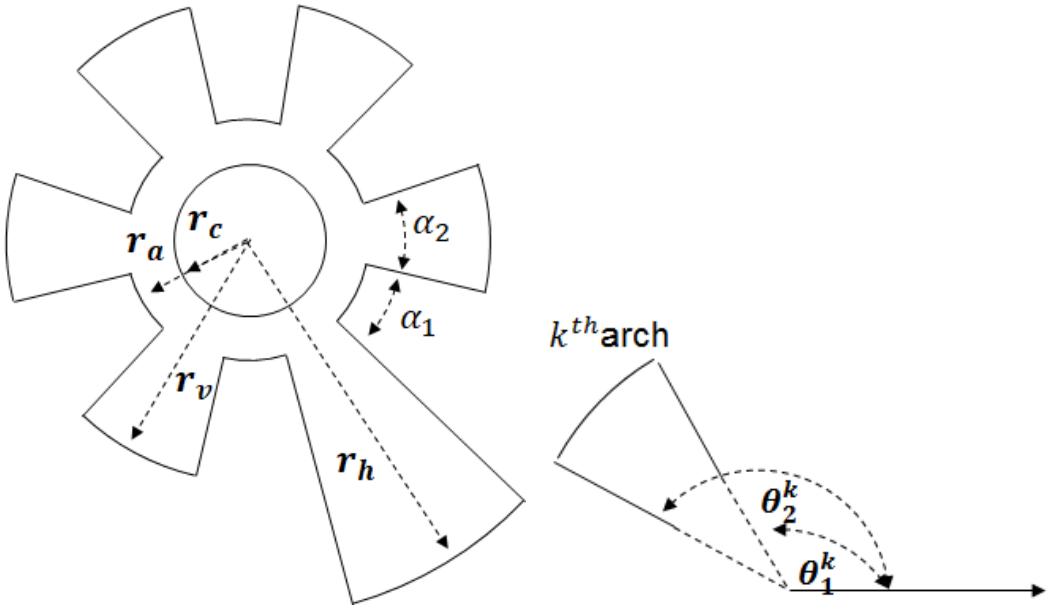
A model of A6M is a useful device to study high power microwave sources. This device has a solid cathode and a solid anode with six cavities as shown in Figure 4.6-(a)). In our modeling, we consider both categories solid and transparent cathodes ([28; 56]) and design the coupling horn by removing the top portion of a rectangular pyramid and fixing it with a cavity of the anode. If we install the whole pyramid in the same place, the top will be at the center of the anode. In order to model the A6 magnetron with diffraction output (MDO) in 2D, we assume that the cathode and anode are infinitely long in the z-direction. Figure 4.6-(a) shows the axial view of the MDO in 2D. Suppose α_1 and α_2 are the angular widths of the vane and cavity respectively, we can compute the angles θ_1 and θ_2 for each cavity of arch in terms of α_1 and α_2 (Figure 4.6-(b)). The angle of the k^{th} arch is given by,

$$\theta_1^k = \theta_1^{k-1} + \alpha_1 + \alpha_2 = \theta_1^1 + (k-1)(\alpha_1 + \alpha_2) \tag{4.9}$$

$$\theta_2^k = \theta_1^{k-1} + 2\alpha_1 + \alpha_2 = \theta_1^1 + k\alpha_1 + (k-1)\alpha_2 \tag{4.10}$$

where, since this is a symmetric uniform model $\alpha_1 + \alpha_2 = \frac{\pi}{3}$. The model has two closed objects, a six wings object of anode with an object of circle inside which represents the cathode. The geometry of a six wings object can be represented with a graph which has 24 vertices, 12 straight line edges and 12 arches [68].

During the pre-computation, for each horizontal and vertical lines, we find the intersection



(a) Axial view (b) Cavity angle

Figure 4.6: A6 MDO (a) Axial view with a solid cathode of radius r_c in the center, the anode with inner radius r_a, vane radius r_v, vane angle α_1, cavity angle α_2, and a coupling horn of radius r_h (b) the angle of k^{th} cavity which is represented by the angles θ_1^k and θ_2^k, $\theta_2^k = \theta_1^k + \alpha_1$.

points with edges of graphs and compute ghost, boundary, and interior points((x_I, y_I) and (x_{II}, y_{II})) Figure 4.7 and relevant parameters which are needed to apply bi-linear interpolation. The pre-computation task forms two sets of line segments for each direction which have to be treated during x and y sweeps with our numerical scheme.

4.4 Numerical Results

In order to evaluate our numerical scheme, we consider several test cases in 2D and 3D using classic EM problems such as electromagnetic scattering of a plane wave by metal circular cylinders (which can be extended to dielectric circular cylinders and applicable to photonic crystal applications), scattering of laser light by spherical objects, and electron beams in A6M.

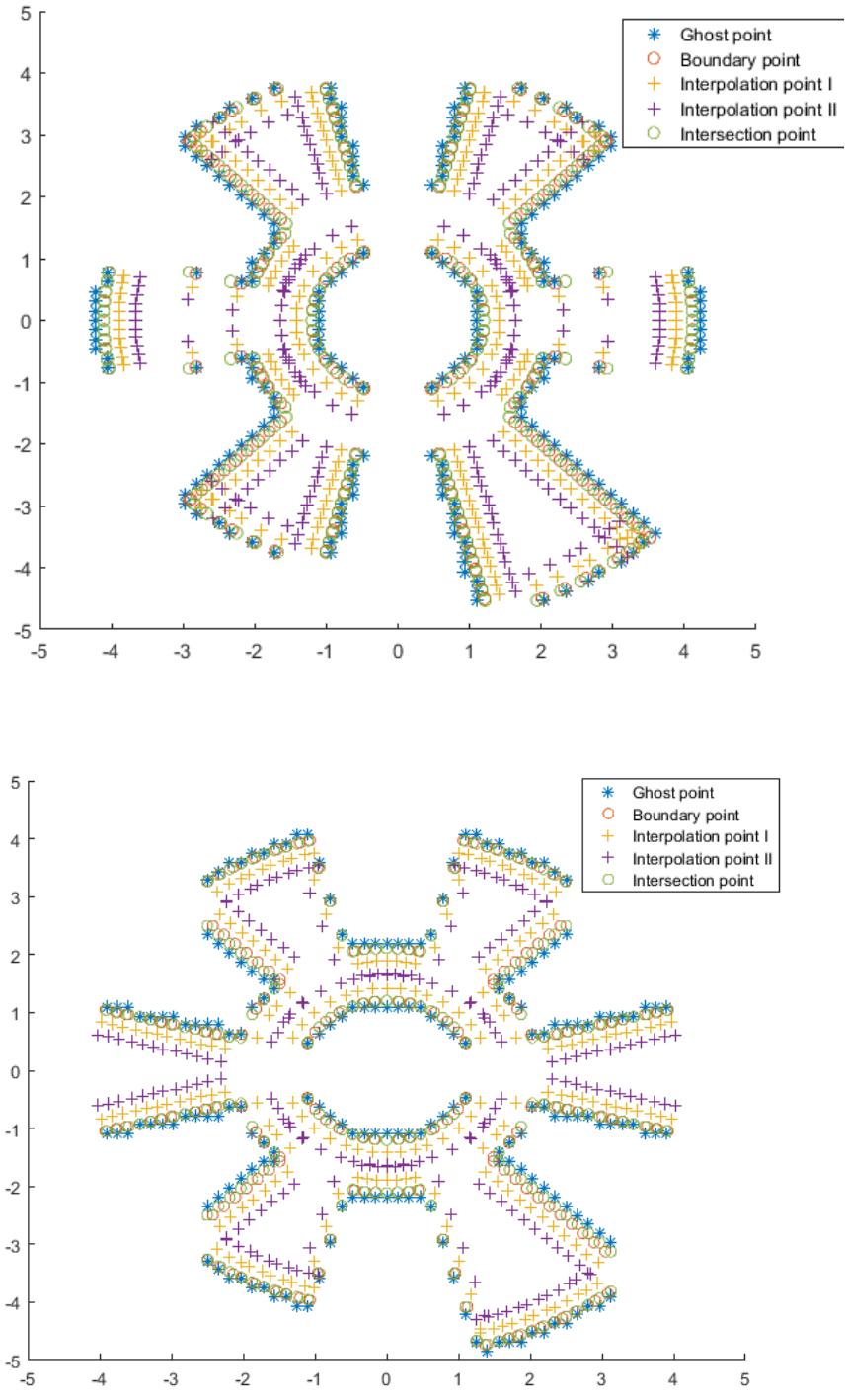
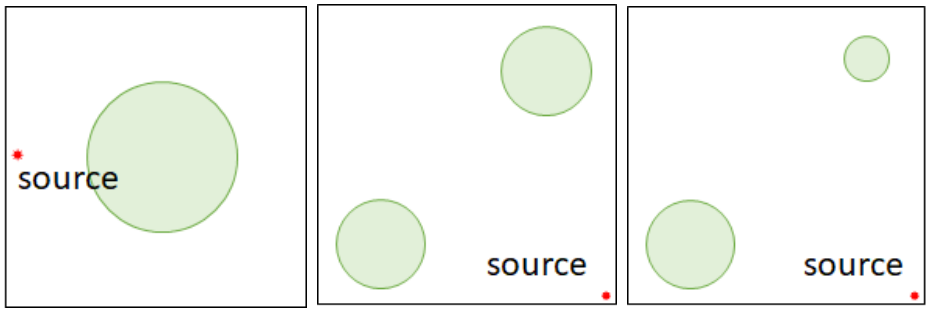


Figure 4.7: Geometrical structure of A6 MDO with key points; intersection, boundary, ghost, and interpolation points required to obtain (a) $x-$ and (b) $y-$ sweeps.

4.4.1 Cylindrical Scattering

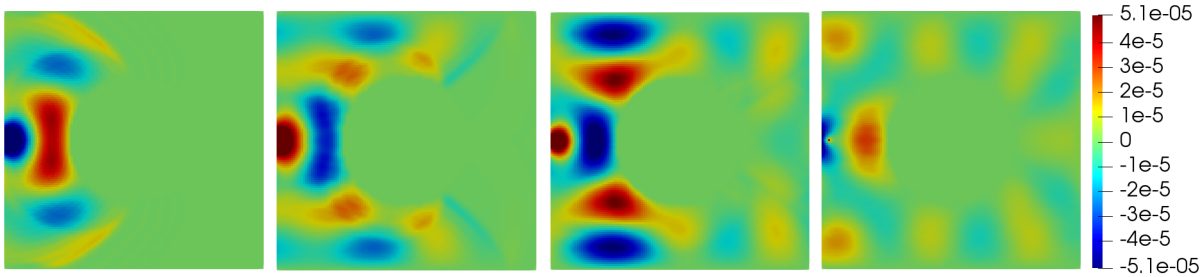
We begin with electromagnetic wave scattering of a plane wave by a metal cylinder of radius r which is homogeneous along the z direction. By assuming the cylinder is infinitely tall, we can represent this case in a 2D model. We choose a rectangular domain $\Omega = [-1,1]\times[-1,1]$ with a circle of radius $r = 0.5$ with center at $(0,0)$, and analyse a point source field ($cos(\omega t)$, $\omega = 10$) placed at $(-1+3\Delta x, 0), \Delta x = 0.0156$ (Figure 4.8 - (a)) by applying Neumann, outflow and Dirichlet boundary conditions along the circular boundary, vertical rectangular borders (in y direction), and horizontal rectangular borders (in x direction) respectively. We set the grid size to be 128×128, time step size $\Delta t = 0.0078$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. Figure 4.9 shows snapshots of the scattered wave at different time instants.



(a) A cylinder at the center (b) Two symmetrical cylinders (c) Two asymmetrical cylinders

Figure 4.8: 2D model for a single cylindrical scatter at the (a) center and two (b) symmetric and (c) asymmetric cylindrical scatters in the corners. Here the red mark indicates the point source.

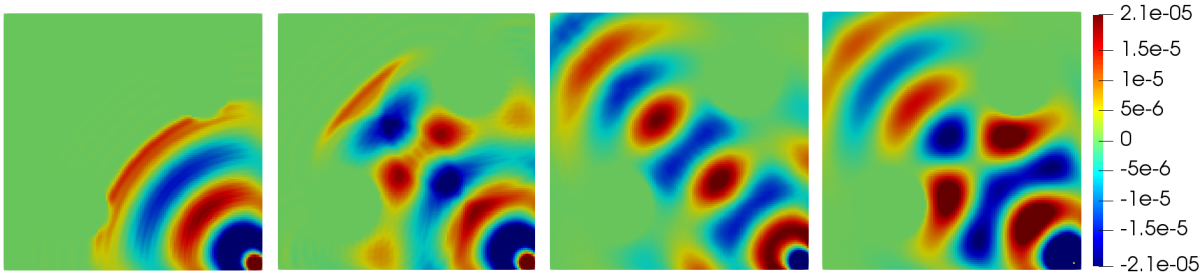
For the following two experiments, we use two circular scatters instead of one and consider symmetric and asymmetric geometry with a rectangular domain $\Omega = [-2,2]\times[-2,2]$. We move the point source to the bottom right corner and place another circular scatter with the same radius $r_1 = 0.3$ for the symmetric case (Figure 4.8 - (c)) and with radius $r_2 = 0.15$ for the asymmetric case (Figure 4.8 - (d)) at the top right corner $(1.5, 1.5)$. We apply Neumann and outflow boundary conditions along the circular and the four rectangular boundaries respectively and retain the same values for parameters $\Delta x, \Delta t, \beta$, and ϵ. Figures 4.10 and



(a) $t = 1.0608$ (b) $t = 2.0436$ (c) $t = 3.1668$ (d) $t = 4.329$

Figure 4.9: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with a circular scatter of radius $r = 0.5$ in the center of the square domain $\Omega = [-1, 1]^2$ with a spatial step size $\Delta x = \Delta y = 0.0156$ and time step size $\Delta t = 0.0078$.

4.11 show snapshots of the wave at different time instants for the symmetric and asymmetric cases respectively.

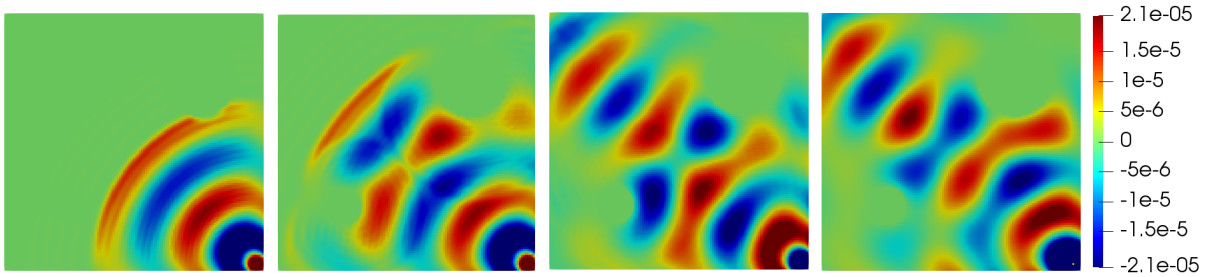


(a) $t = 1.2714$ (b) $t = 1.9032$ (c) $t = 2.886$ (d) $t = 4.29$

Figure 4.10: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with two symmetric circular scatters of radii $r_1 = r_2 = 0.3$ in the bottom-left and top-right corners of the square domain $\Omega = [-2, 2]^2$ with a spatial step size $\Delta x = \Delta y = 0.0156$ and time step size $\Delta t = 0.0078$.

4.4.1.1 Convergence studies

We demonstrate fourth order convergence by performing a refinement study on a 2D square domain $\Omega = [-1, 1] \times [-1, 1]$ with a circular scatter of radius $r = 0.5$ at the center, and a point source $cos(\omega t)$, $\omega = 1$ placed at $(0, -1+3\Delta x)$, $\Delta x = 0.0156$. This runs up to the final time $T = 2.0$, with a fixed spatial resolution of 160×160 spatial points. The discrete L_∞ norm of the error is constructed at each time step and maximum error over all time steps is used to graph 4.12 for $\Delta t = \frac{6.25}{2^k}$, $k = 1$ to 5, with Neumann and outflow boundary



(a) $t = 1.2714$ (b) $t = 1.9032$ (c) $t = 2.886$ (d) $t = 4.29$

Figure 4.11: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with two asymmetrical circular scatters of radii $r_1 = 0.3$ and $r_2 = 0.15$ in the bottom-left and top-right corners of the square domain $\Omega = [-2, 2]^2$ with a spatial step size $\Delta x = \Delta y = 0.0156$ and time step size $\Delta t = 0.0078$.

conditions along the circular and rectangular boundaries respectively.

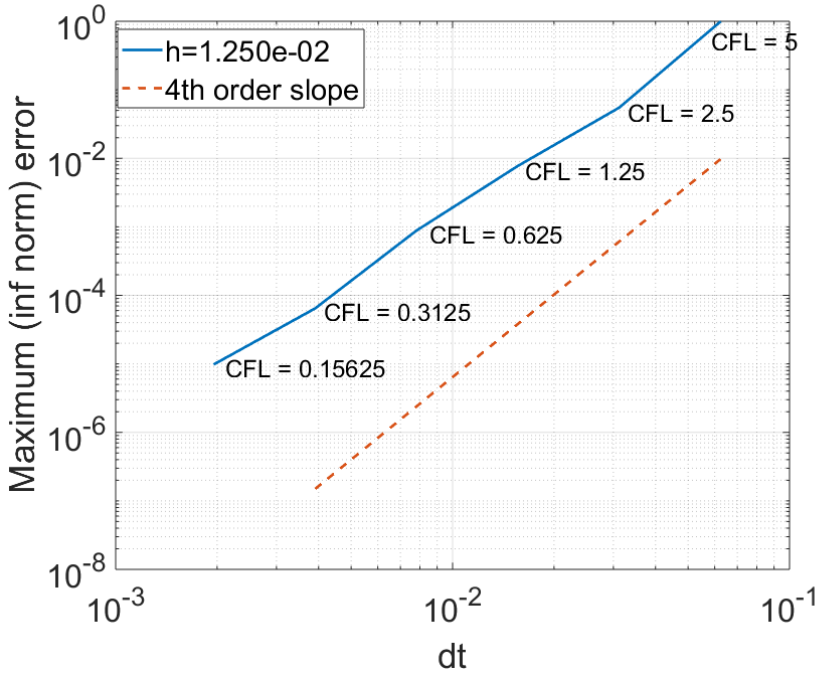
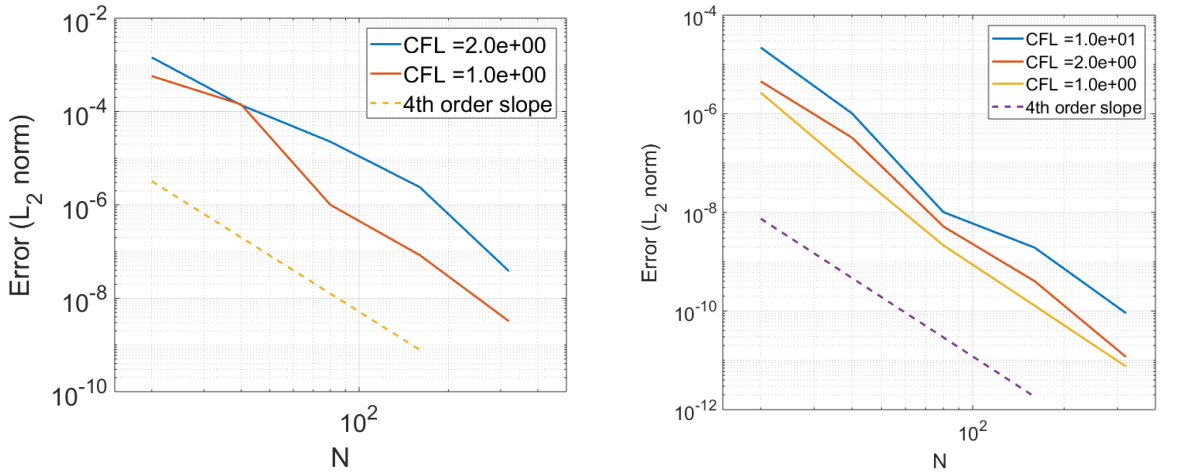


Figure 4.12: Fourth order convergence of 2D wave solver using a point source $cos(\omega t)$, $\omega = 1$ placed at $(0, -1 + 3\Delta x), \Delta x = 0.0156$ on a square domain $\Omega = [-1, 1]^2$ with a circular scatter of radius $r = 0.5$ at the center by imposing Neumann and outflow along the circular and square boundaries. This is a self-refinement study which measures L_∞ norm of the error at the final time $T = 2.0$.

For the space convergence study, a point source $cos(\omega t)$, $\omega = 1$ is placed at $(-1+3\Delta x, 0)$, and runs up to time $T = 2.0$, with fixed CFL values 2 and 1, where the wave speed c is

chosen to be 1. Our L_∞ norm error plots of the solution at time $T = 2.0$ with varying resolutions show fourth order accuracy (Figure 4.13 - (a)). We also note that the normal derivative along the boundary converges with fourth order accuracy (Figure 4.13 - (b)).



(a) Space convergence of solution

(b) Space convergence of boundary derivative

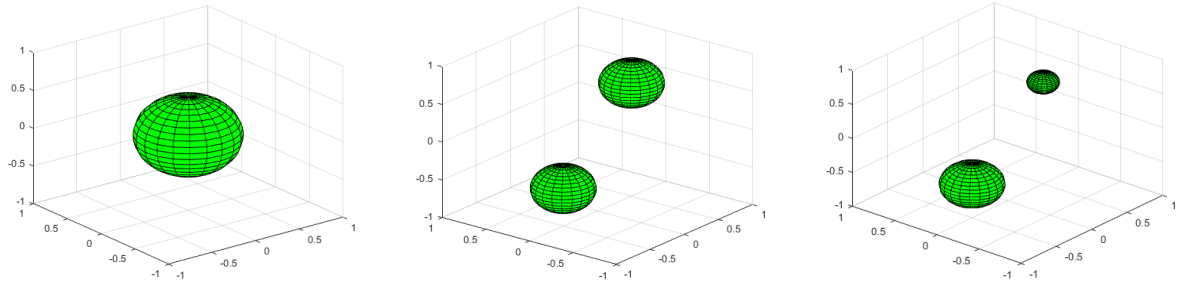
Figure 4.13: Convergence plots for a) space using entire solution and b) space using boundary derivative on a square domain $\Omega = [-1, 1]^2$ with a circular scatter of radius $r = 0.5$ at the center by imposing Neumann and outflow along the circular and square boundaries. This is a self-refinement study which measures L_2 norm of the error at the final time $T = 2.0$.

4.4.2 Spherical Scattering

In this section, we evaluate our 3D scheme using the scattering of light by a spherical object.

For the first test case, we chose a cubic domain ($\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$) with a spherical scatter of radius $r = 0.5$ at the center of the cube $(0, 0, 0)$ (Figure 4.14, (a)). We place a source field ($cos(\omega t)$, $\omega = 10$) placed $3\Delta z$ above the center of the bottom surface(xy plane), at $(0, 0, -1 + 3\Delta z), \Delta z = 0.0156$ by applying Neumann and outflow boundary conditions along the surface of the sphere and six surfaces of the cube respectively. We set the grid size to be $128 \times 128 \times 128$, time step size $\Delta t = 0.0078$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. Figure 4.15 shows snapshots of the wave at different time

instants.



(a) Single sphere (b) Two symmetric spheres (c) Two asymmetrical spheres

Figure 4.14: 3D model for a single spherical scatter at the center of a cube, two b) symmetric and c) asymmetric spherical scatters in the corners.

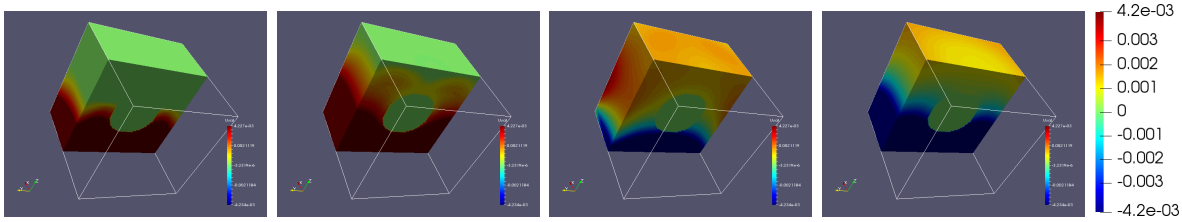


Figure 4.15: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with a spherical scatter of radii $r = 0.5$ in the center of the cubic domain $\Omega = [-1, 1]^3$ with a spatial step size $\Delta x = \Delta y = \Delta z = 0.0156$ and time step size $\Delta t = 0.0078$.

For the next two experiments we placed two equal sized spheres of radii $r_1 = r_2 = 0.3$ for the first case and two spheres with different radii $r_1 = 0.3$ and $r_2 = 0.15$ for the second case at the corners of the cubical domain as shown in Figure 4.14 - (b), (c). We move the point source to $(0, 0, -1 + 3\Delta z), \Delta z = 0.0156$ and simulate the wave propagation by applying Neumann and outflow boundary conditions along the surface of the two spheres and six surfaces of the cube respectively. We retain the same values of parameters Δx, Δt, β, and ϵ as before. Figure 4.16 and 4.17 show snapshots of the wave at different time instants for the symmetric and asymmetric cases respectively.

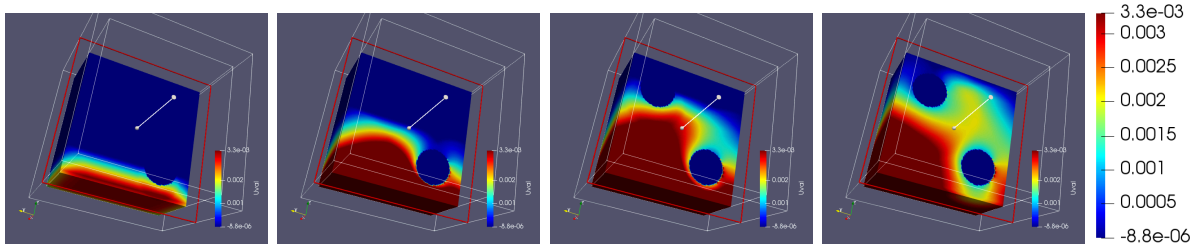


Figure 4.16: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with two symmetric spherical scatters of radii $r_1 = r_2 = 0.3$ in the bottom-left and top-right corners of the cubic domain $\Omega = [-1, 1]^3$ with a spatial step size $\Delta x = \Delta y = \Delta z = 0.0156$ and time step size $\Delta t = 0.0078$.

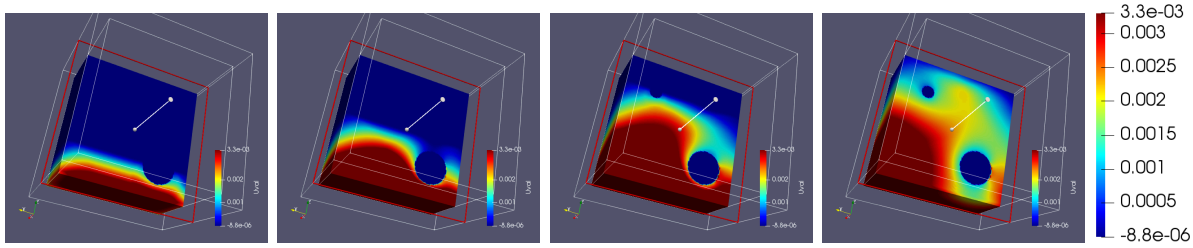


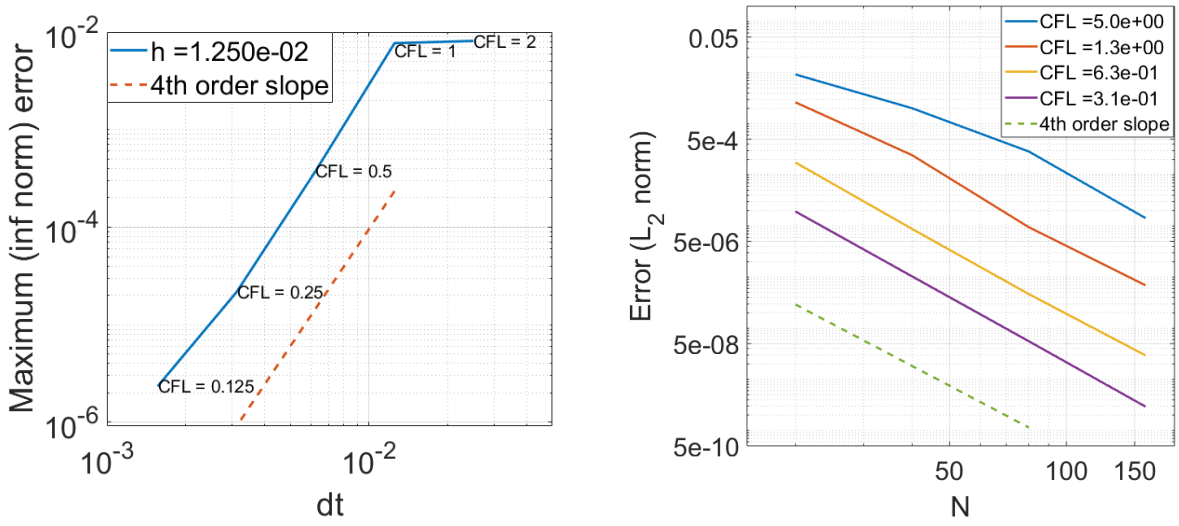
Figure 4.17: Time evolution of a point source field $cos(\omega t)$, $\omega = 10$ with two asymmetric spherical scatters of radii $r_1 = 0.3$ and $r_2 = 0.15$ in the bottom-left and top-right corners of the cubic domain $\Omega = [-1, 1]^3$ with a spatial step size $\Delta x = \Delta y = \Delta z = 0.0156$ and time step size $\Delta t = 0.0078$.

4.4.2.1 Convergence Studies

We demonstrate fourth order convergence in time and space by performing self-refinement studies on a rectangular domain $\Omega = [-1, 1]^3$ with a spherical scatter of radius $r = 0.5$ at the center.

For the time convergence study, a point source $cos(\omega t)$, $\omega = 1$ is placed at $(0, 0, -1 + 3\Delta z), \Delta z = 0.0156$, and runs up to dimensionless time $T = 2.0$, with a fixed spatial resolution of $160 \times 160 \times 160$ spatial points. The discrete L_∞ norm of the error is constructed at each time step and maximum error over all time steps is used to graph Figure 4.18 -a) for $\Delta t = \frac{0.025}{2^k}$, $k = 1$ to 5, with Neumann and outflow boundary conditions along the sphere surface and cube surfaces respectively.

For the space convergence study, a point source $cos(\omega t)$, $\omega = 1$ is placed at $(0, 0, -1 + 3\Delta z)$, and evolves to dimensionless time $T = 2.0$, with fixed CFL values 2, 1, and 0.5 where



(a) Time convergence

(b) Space convergence

Figure 4.18: Fourth order (a) time and (b) space convergence plots using a spherical scatter of radius $r = 0.5$ at the center of a cubic domain $\Omega = [-1, 1]^3$ by imposing Neumann and outflow along the surface of the sphere and cubic boundaries. This is a self-refinement study which measures (a) L_∞ and (b) L_2 norm of the error at the final time $T = 2.0$.

the wave speed c is chosen to be 1. Our L_2 norm error plots of the solution at dimensionless time $T = 2.0$ with varying resolutions show fourth order accuracy (Figure 4.18 -b)).

4.4.3 Electron Beam in Magnetrons

4.4.3.1 A6 Magnetron with Diffraction Output (MDO) in 2D

In this section, we test our solver using a modified model for the A6M in 2D. We use a point source instead of voltage source and made two experiments using solid and transparent cathodes with a coupling horn of radius $r_h = 4.9$, radius of the vane resonators $r_v = 4.11$, the anode radius $r_a = 2.11$, the cathode radius $r_c = 1.58$, the angular width of the vane, $\alpha_1 = \frac{\pi}{6}$, and the cavity, $\alpha_2 = \frac{\pi}{6}$. For the first test case, we simulate the scattered wave of a circular source, $cos(\omega t)(|x^2 + y^2 - r_c^2| < 2\Delta x), \omega = 10$ placed along with the cathode. 4.19 shows snapshots of the wave captured at different time instants.

In the second test case, we use six point-sources that are placed around the circle of the

cathode each nearby a cavity (we maintain 15^o from left edge-radius of each cavity) and simulate the waves propagated by these sources. 4.20 shows snapshots of the wave captured at different time instants. We applied Neumann boundary conditions along with the model for both cases.

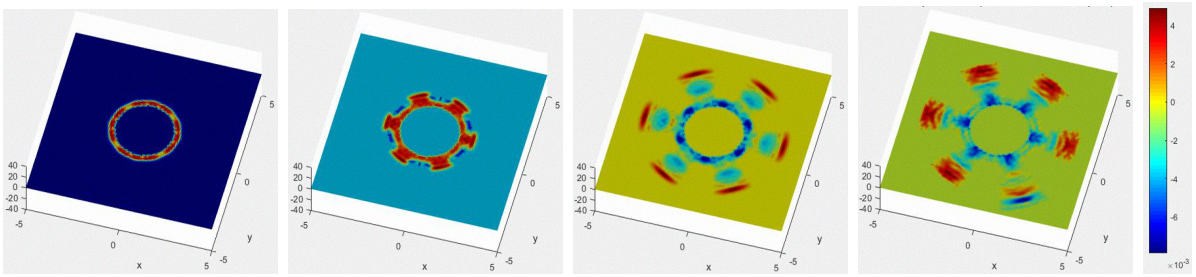


Figure 4.19: Time evolution of a circular source field $cos(\omega t)(|x^2+y^2-r_c^2| < 2\Delta x), \omega = 10$ in the model of A6RM with a solid cathode of radius $r_c = 1.58$ and anode of inner radius $r_a = 2.11$, vane radius $r_v = 4.11$ and the coupling horn radius $r_h = 4.9$ and the angular width of the vane, $\alpha_1 = \frac{\pi}{6}$, and the cavity, $\alpha_2 = \frac{\pi}{6}$.

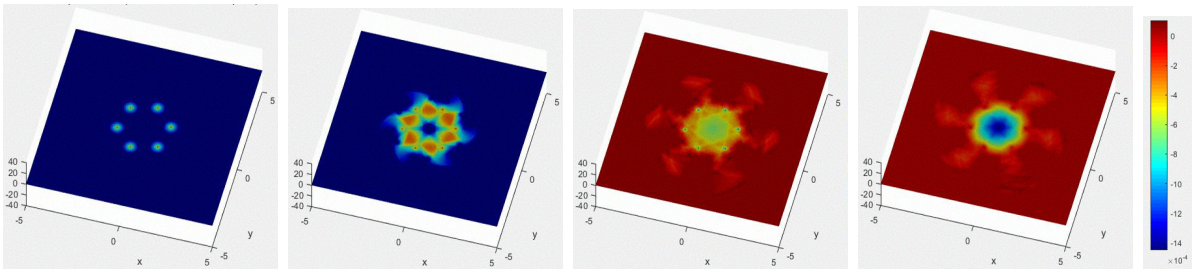
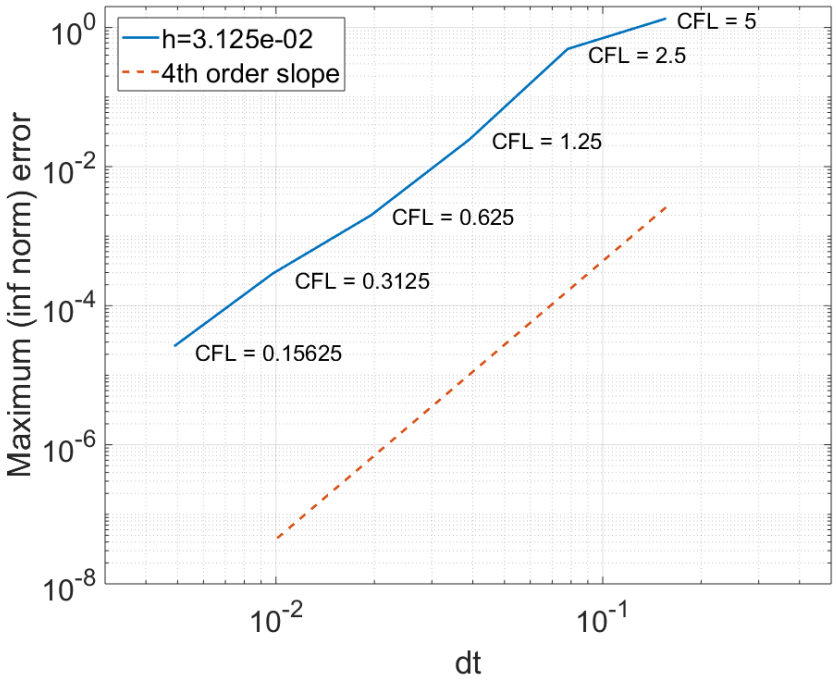


Figure 4.20: Time evolution of fields provided by six point sources in the model of A6M with a transparent cathode of radius $r_c = 1.58$ and anode of inner radius $r_a = 2.11$, vane radius $r_v = 4.11$ and the coupling horn radius $r_h = 4.9$ and the angular width of the vane, $\alpha_1 = \frac{\pi}{6}$, and the cavity, $\alpha_2 = \frac{\pi}{6}$.

Further, we perform a convergence test by a refinement study on a model for A6M without cathodes. We set the radii for coupling horn, vane resonators, and anode as $r_h = 4.9$ $r_v = 4.11$, and $r_a = 2.11$ respectively. The angular width of the vane, $\alpha_1 = \frac{\pi}{6}$ and the cavity angle, $\alpha_2 = \frac{\pi}{6}$. We place a point source $cos(\omega t)$, $\omega = 1$ at the center $(0, 0)$, and evolve to the final dimensionless time $T = 2.0$, with a fixed spatial resolution of 160×160 spatial points. The discrete L_∞ norm of the error is constructed at each time step and maximum error

over all time steps is used to graph Figure 4.21 for $\Delta t = \frac{0.3125}{2^k}$, $k = 1$ to 6, with Neumann boundary conditions along the model. It shows fourth order convergence.



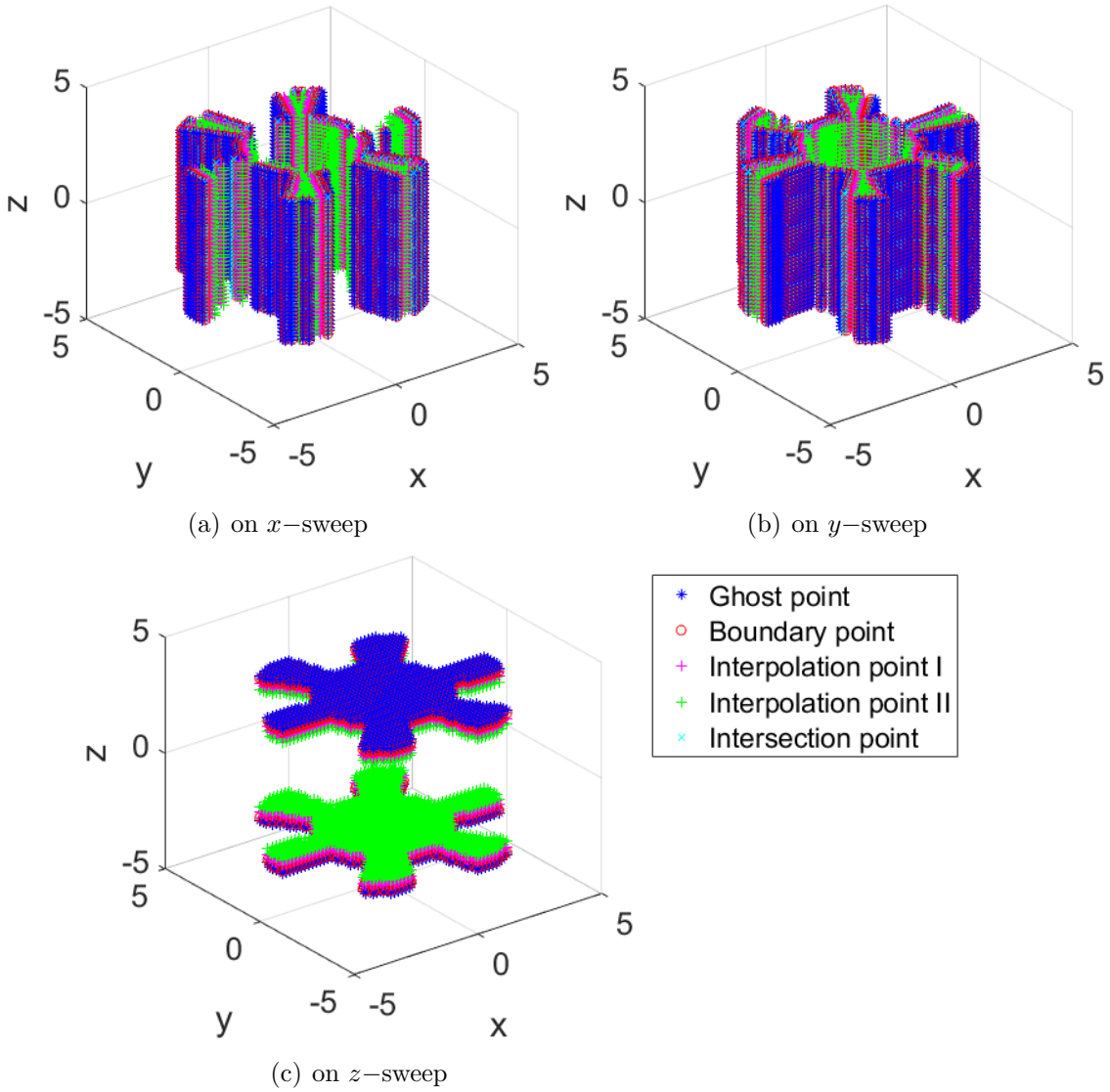
(a) A6M

Figure 4.21: Fourth order convergence of fields provided by a point source $cos(\omega t)$, $\omega = 1$ at the center $(0, 0)$, of A6MDO with the anode of radii $r_h = 4.9$ $r_v = 4.11$, and $r_a = 2.11$ and angular width of the vane, $\alpha_1 = \frac{\pi}{6}$ and the cavity, $\alpha_2 = \frac{\pi}{6}$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 2.0$ for fixed spatial step size $\Delta x = \Delta y = 3.13 \times 10^{-2}$

4.4.3.2 A6 Magnetron in 3D

For this experiment, a 3D domain of $\Omega = [-5, 5]^3$ is chosen with radius of the vane resonators $r_v = 4.11$, anode radius $r_a = 2.11$, cathode radius $r_c = 1.58$, angular width of vane 20^o, cavity angle 40^o, and thick $h = 6$ along z. We set the grid size to be 128^3, time step size $\Delta t = 0.039$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. In this simulation of A6 magnetron with a transparent cathode which is mimicked by using six emitters/line sources, we calculated the time evolution by imposing embedded Neumann boundary conditions over the boundary stencil during the $x-$, $y-$, and $z-$ sweeps (Figure

4.23). Figure 4.22 shows the geometrical setup of the 3D A6 magnetron, indicating key on/off mesh grid points used by the scheme.



(a) on $x-$sweep

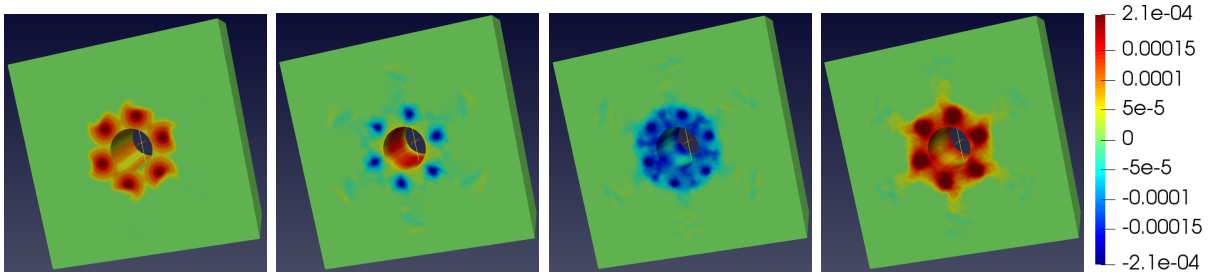
(b) on $y-$sweep

(c) on $z-$sweep

Figure 4.22: Key points used for the simulation of 3D A6 magnetron; intersection, boundary, ghost, and interpolation points required to obtain (a) $x-$, (b) $y-$, and (c) $z-$ sweeps.

4.4.3.3 Convergence Studies

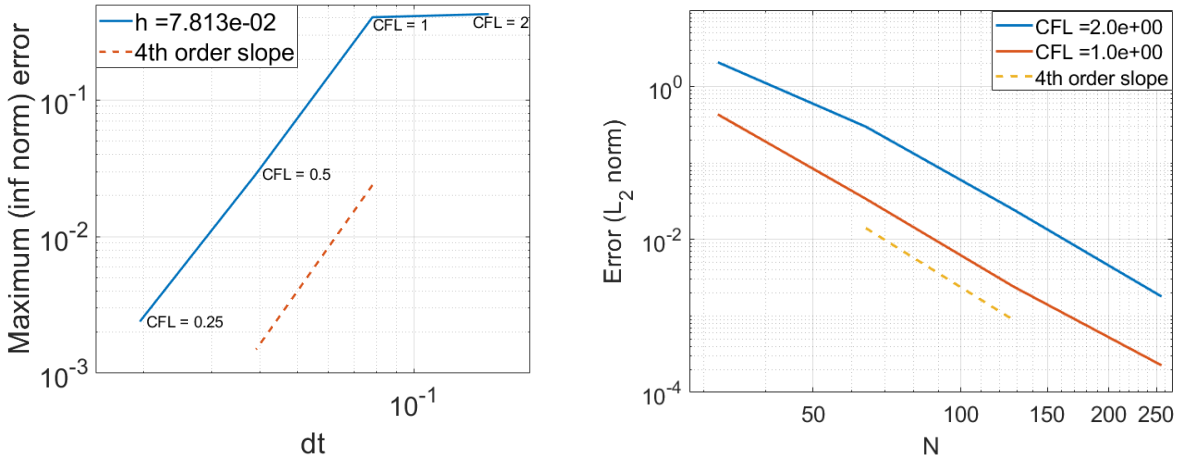
Further, we perform self-refinement studies to evaluate the consistency of our scheme for A6 magnetron with a transparent cathode. We retain the same values of geometrical parameters (r_h, r_v, r_a, r_c, α_1, α_2, and h) and emitters. The simulation evolves to the final



(a) $t = 1.5625$ (b) $t = 4.6875$ (c) $t = 6.25$ (d) $t = 11.7187$

Figure 4.23: Time evolution of fields provided by six point sources in the model of 3D A6 magnetron with a transparent cathode of radius $r_c = 1.58$ and anode of inner radius $r_a = 2.11$, vane radius $r_v = 4.11$ and the coupling horn radius $r_h = 4.9$, the angular width of the vane, $\alpha_1 = \frac{\pi}{9}$, cavity angle, $\alpha_2 = \frac{2\pi}{9}$, and thickness $h = 6$.

time $T = 2.0$ by imposing Neumann boundary conditions along the model. For the time-convergence test, we set a fixed spatial resolution of 128^3 spatial points, the discrete L_∞ norm of the error is constructed at each time step and maximum error over all time steps is used to graph Figure 4.24 -a) for $\Delta t = \frac{0.3125}{2^k}$, $k = 1$ to 4. It shows fourth order convergence.



(a) time convergence (b) space convergence

Figure 4.24: Fourth order (a) time convergence and (b) space convergence of fields provided by six point sources in the model of 3D A6 magnetron with a transparent cathode and the anode of radii $r_h = 4.9$ $r_v = 4.11$, and $r_a = 2.11$ and angular width of the vane, $\alpha_1 = \frac{\pi}{6}$ and the cavity, $\alpha_2 = \frac{\pi}{6}$. This is a self-refinement study which measures L_∞ norm of the error at time $T = 2.0$ for a fixed spatial step size of $\Delta x = \Delta y = 3.13 \times 10^{-2}$

For space-convergence test, we use fixed CFL values 2, 1, and 0.5 where the wave speed c is chosen to be 1. Our L_∞ norm error plots of the solution at dimensionless time $T = 2.0$ with varying resolutions show fourth order accuracy (Figure 4.24 -b)).

4.5 Summary

We proposed a fourth-order 3D embedded boundary method and developed a generalized framework for complex geometry domains in 3D. We use the extended stencil for both computations of the solution at the interpolation points (8-point stencil) and the ghost points (4-point stencil) to achieve 4th-order convergence in space. And we use the same Lax-Wendroff approach (exchange the time derivative to spatial derivative) to obtain higher-order accuracy in time. We gave a detailed description of the high-order Neumann boundary conditions and its implementation in 1D, 2D, and 3D. We evaluated our scheme for several problems with complex geometries including A6 magnetron in 3D. Since the magnetrons originally designed using metal, our best choice to represent its boundaries should be PEC. Therefore, we discuss the PEC boundary condition in the next chapter. We give descriptions of the derivation of PEC boundary conditions for our MOLT based scheme and its evaluation.

CHAPTER 5: Second-Order PEC Boundary Condition

5.1 Introduction

In this chapter, we develop an embedded PEC boundary condition for the scheme described in Chapter 2 and develop an exact simulation for magnetrons by solving the vector potential form of Maxwell's equations.

Since magnetrons are physically designed by metal boundaries and have complex geometries, we need PEC boundary conditions to represent the exact metal boundaries and the embedded boundary method to deal with complex geometries. Hence, we derive an embedded PEC boundary condition based on the electromagnetic vector potential for the simulation of magnetrons in this chapter. We introduce a fast and geometrically flexible approach to calculate the solution to Maxwell's equations in vector potential form under the Lorenz gauge. As presented, the method is 4th order in time and second-order in space, but the A-stable formulation could be extended to higher order in both time and space. While there is no conceptual limitation to develop this in 3D, our initial work has centered on 2D. The eventual goal is to combine this method with particle methods for the simulations of plasma. In the current work, the scheme is evaluated for EM wave propagation within an object that is bounded by PEC.

Vector potential formulations of electromagnetism are widely used in classical and quantum physics. Maxwell's equations describe the time-evolution of four fields: the magnetic flux density ($\mathbf{B}$), the electric field intensity ($\mathbf{E}$), the electric flux density ($\mathbf{D}$), and the magnetic field intensity ($\mathbf{H}$). It is often convenient to employ a formulation based on the magnetic vector potential $\mathbf{A}$ and electric scalar potential ϕ. Maxwell's equations then reduce to uncoupled wave equations for the vector potential $\mathbf{A}$ and scalar potential ϕ under the Lorenz

gauge condition [37]. We use our MOLT based implicit A-stable scheme to solve these wave equations.

A Perfect Electric Conductor (PEC) boundary condition is derived based on the continuity of the magnetic flux normal to the boundary and is applied to the magnetic vector potential. Our formulation avoids the additional compatibility conditions used in [34]. This embedded PEC boundary condition is a slightly modified embedded Neumann boundary condition. The fundamental difference is that one of the partial spatial derivatives in the PDE is negative. As presented, the current embedded boundary formulation is second-order in space but could be extended to higher-order.

In this chapter, we first derive wave equations for vector and scalar potentials $\mathbf{A}$ and ϕ under the Lorenz gauge condition discussed in section 5.2. Then we describe our two-dimensional high-order implicit scheme using ADI splitting and the higher-order embedded PEC boundary conditions in section 5.3, and finally we give numerical results for several test cases in Section 5.5.

5.2 2D Electric Scalar and Magnetic Vector Potential

The macroscopic Maxwell's equations are:

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E}, \tag{5.1}$$

$$\partial_t \mathbf{D} = \nabla \times \mathbf{H} - \mathbf{J}, \tag{5.2}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{5.3}$$

$$\nabla \cdot \mathbf{D} = \rho. \tag{5.4}$$

where $\mathbf{J}$ is the electric current density, and ρ is the electric charge density. In a linear isotropic medium, $\mathbf{D} = \epsilon\mathbf{E}$ and $\mathbf{B} = \mu\mathbf{H}$. The dielectric constant $\epsilon = \epsilon_0\epsilon_r$ with ϵ_0 and ϵ_r being the free space and relative permittivity respectively, and the permeability $\mu = \mu_0\mu_r$ with μ_0 and μ_r being the free space and relative permeability respectively.

The electric scalar potential ϕ and magnetic vector potential $\mathbf{A}$ are related to $\mathbf{E}$ and $\mathbf{B}$ by

$$\mathbf{E} = -\nabla\phi - \partial_t \mathbf{A}, \tag{5.5}$$

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{5.6}$$

For free space, Ampere's law (Equation 5.2) is,

$$\frac{1}{c^2}\partial_t \mathbf{E} = \nabla \times \mathbf{B} - \mu_0 \mathbf{J}. \tag{5.7}$$

where the free space phase velocity $c = \frac{1}{\sqrt{(\mu_0 \epsilon_0)}}$.

Substituting $\mathbf{E}$ and $\mathbf{B}$ using equations 5.5 and 5.6 and imposing the Lorenz gauge condition

$$\frac{1}{c^2}\partial_t \phi = -\nabla \mathbf{A}.$$

a wave equation in terms of the magnetic vector potential $\mathbf{A}$ results:

$$\frac{1}{c^2}\partial_t^2 \mathbf{A} - \nabla^2 \mathbf{A} = \mu_0 \mathbf{J}. \tag{5.8}$$

Similarly by imposing the Lorenz gauge condition, a wave equation for the scalar potential ϕ in free space is obtained:

$$\frac{1}{c^2}\partial_t^2 \phi - \nabla^2 \phi = \frac{\rho}{\epsilon_0}. \tag{5.9}$$

5.3 Perfect Electric Conductor Boundary

The electric field ($\mathbf{E}$) is continuous along the boundary and the magnetic flux ($\mathbf{B}$) is continuous along the normal to the boundary. Since the Perfectly Conducting Boundary (PEC) has infinite electrical conductivity ($\sigma = \infty$), there will be no interior electric field ($\mathbf{E_2} = 0$) in the perfect conductor. It also follows that there is no magnetic field ($\mathbf{H_2} = 0$).

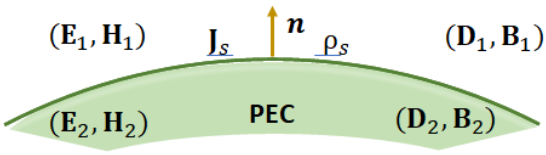


Figure 5.1: Boundary surface between two regions with electric fields $\mathbf{E}_1$ and $\mathbf{E}_2$, magnetic field $\mathbf{H}_1$ and $\mathbf{H}_2$, electric flux densities $\mathbf{D}_1$ and $\mathbf{D}_2$, and magnetic flux densities $\mathbf{B}_1$ and $\mathbf{B}_2$ respectively. Here, $\mathbf{J}_s$ is the surface current, ρ_s is the surface charge density, and $\mathbf{n}$ is the normal vector pointed out from the region two (Perfect Electric Conductor).

The boundary conditions for the PEC become:

$$\mathbf{n} \times \mathbf{E_1} = 0, \tag{5.10}$$

$$\mathbf{n} \times \mathbf{H_1} = \mathbf{J_s}, \tag{5.11}$$

$$\mathbf{n} \cdot \mathbf{B_1} = 0, \tag{5.12}$$

$$\mathbf{n} \cdot \mathbf{D_1} = \rho_\mathbf{s}. \tag{5.13}$$

where $\mathbf{n}$ is the unit normal vector to the boundary and $\mathbf{J_s}$ and $\rho_\mathbf{s}$ are the surface current density and charge density respectively.

Consider a two-dimensional vector/scalar potential (we are solving 2D problems in this section). If we choose to restrict B to the x-y plane, the vector potential $\mathbf{A}$ only has a z component,

$$\mathbf{A} = \mathbf{A}_z(x, y)\mathbf{z}. \tag{5.14}$$

Using equation 5.6 and the boundary condition represented by equation 5.12 we get

$$\mathbf{n} \cdot (\partial_y \mathbf{A}_z - \partial_x \mathbf{A}_z) = 0. \tag{5.15}$$

5.4 Numerical Approximation for PEC Boundary Conditions

In this section we discuss enforcing boundary conditions with the above method. For PEC boundary conditions, we extend our previous work on Neumann boundary conditions to PEC (Chapter 4, [13]). Before we move with boundary conditions, let us express the second and fourth order two dimensional schemes,

$$\mathbf{A}^{n+1} - 2\mathbf{A}^n + \mathbf{A}^{n-1} = -\beta^2 \mathcal{C}_{xy}^{(1)}[\mathbf{A}^n]. \tag{5.16}$$

$$\mathbf{A}^{n+1} - 2\mathbf{A}^n + \mathbf{A}^{n-1} = -\beta^2 \mathcal{C}_{xy}^{(1)}[\mathbf{A}^n] - \left(\beta^2 \mathcal{D}_{xy}^{(2)} - \frac{\beta^4}{12}\mathcal{C}_{xy}^{(2)}\right) \mathcal{C}_{xy}^{(1)}[\mathbf{A}^n]. \tag{5.17}$$

where superscripts on the operators $\mathcal{C}_{xy}$ and $\mathcal{D}_{xy}$ denote level numbers.

On a domain $\Omega = [x_a, x_b] \times [y_a, y_b]$ outflow boundary and PEC conditions can be defined by,

1. PEC:

   $\partial_y \mathbf{A}_z(x_a, t) = 0, \quad -\partial_y \mathbf{A}_z(x_b, t) = 0,$

   $-\partial_x \mathbf{A}_z(y_a, t) = 0, \quad \partial_x \mathbf{A}_z(y_b, t) = 0.$

2. Outflow boundary condition:

   $\partial_t \mathbf{A}(x_a, t) = c\partial_x \mathbf{A}(x_a, t)$, $\partial_t \mathbf{A}(x_b, t) = -c\partial_x \mathbf{A}(x_b, t),$

   $\partial_t \mathbf{A}(y_a, t) = c\partial_y \mathbf{A}(y_a, t)$, $\partial_t \mathbf{A}(y_b, t) = -c\partial_y \mathbf{A}(y_b, t).$

Unlike Neumann boundary conditions, the challenge here is that PEC needs y derivatives during the $x-$ sweeps and x derivatives during the $y-$ sweeps. This is challenging for an ADI method because of the separation of directional information.

For a rotated square, the PEC boundary condition can use a similar approach as used in the original method for Neumann boundaries in [13] because the directions are coupled at the boundary. Rotating the domain through any angle, say θ, as shown in figure 5.2, we have,

1. Left: $cos(\theta)\partial_y \mathbf{A}_z - sin(\theta)\partial_x A\mathbf{A}_z = 0$,

2. Right: $-cos(\theta)\partial_y \mathbf{A}_z + sin(\theta)\partial_x \mathbf{A}_z = 0$,

3. Up: $sin(\theta)\partial_y \mathbf{A}_z - cos(\theta)\partial_x \mathbf{A}_z = 0$,

4. Down: $-sin(\theta)\partial_y \mathbf{A}_z + cos(\theta)\partial_x \mathbf{A}_z = 0$.

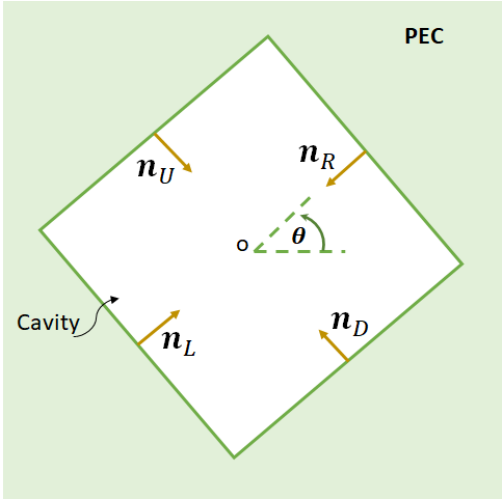


Figure 5.2: A PEC square cavity rotated by an angle θ with normal vectors n_L, n_D, n_R, and n_U along the left, down, right, and up boundaries.

In the next sections, we demonstrate how to construct a solution for **A** given the Dirichlet boundaries, followed by an extension of the fixed point map for Neumann boundaries in [13] to the case of PEC to generate the Dirichlet boundary conditions (including an adaptation for the mesh aligned case).

5.4.1 Using Dirichlet Boundary Conditions for A to Capture PEC Boundary

In our embedded boundary method for Neumann, we constructed an efficient convergent Neumann to Dirichlet Map, converting a boundary condition on the outward normal derivative of a function into a condition that constructs ghost points outside the domain. The method is designed such that the generated ghost points enforce the desired Neumann

condition on the boundary. In the next two sections, we adapt this idea to PEC, which is a condition on the tangential derivative.

We now show how to carry out the sweeps with given values at the end of each line of the 4th order ADI method. In our 2D scenario, to construct the operator $\mathcal{L}_x^{-1}$ and $\mathcal{L}_y^{-1}$ to build $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$ and $\mathcal{D}^{(2)}$, each line is treated as a 1D problem, where the lines are evaluated with $x-$ sweeps and $y-$ sweeps respectively [13]. We note that the operators that make up $\mathcal{D}^{(k)}$ and $\mathcal{C}^{(k)}$ have their own boundary conditions to enforce at level k.

The objective is given the boundary condition, solve for each a_x and b_x along the $x-$ direction and a_y and b_y along the $y-$ direction. We will denote $\mathbf{A}(t_i, x_a) = A_L(t_i)$, $\mathbf{A}(t_i, x_b) = A_R(t_i)$, $\mathbf{A}(t_i, y_a) = A_D(t_i)$ and $\mathbf{A}(t_i, y_b) = A_U(t_i)$. Here, $[x_a, x_b]$ and $[y_a, y_b]$ are horizontal and vertical lines that we make sweeps along. These permit the boundary conditions to be incorporated into the method.

As we did for wave solvers in our early work, [15], taking $A_L(t_i)$ and $A_R(t_i)$ as fixed, for the $\mathcal{L}_x^{-1}$ component of the second order term $\mathcal{C}^{(1)}$ along the line $\partial\Omega = [x_a, x_b]$, we arrive at two equations in two unknowns:

$$
\begin{aligned}
&\mathbf{A}_L(t_{n+1}) + (\beta^2 - 2)\mathbf{A}_L(t_n) + \mathbf{A}_L(t_{n-1}) \\
&= \beta^2\Big(I\Big[\mathbf{A}_L(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](x_a) + a_x - M_x b_x\Big), \\
&\mathbf{A}_R(t_{n+1})(\beta^2 - 2)\mathbf{A}_R(t_n) + \mathbf{A}_R(t_{n-1}) \\
&= \beta^2\Big(I\Big[\mathbf{A}_R(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](x_b) + M_x a_x + b_x\Big).
\end{aligned}
$$

where $M_x = e^{-\alpha(x_b - x_a)}$. We can rearrange the linear system into unknown and known values,

$$
\begin{aligned}
a_x + M_x b_x &= w_a^P, \\
M_x a_x + b_x &= w_b^P.
\end{aligned}
$$

Solving the linear system for the unknowns a_x and b_x gives

$$a_x = \frac{(w_a^P - M_x w_b^P)}{(1 - M_x^2)}, \quad b_x = \frac{(w_b^P - M_x w_a^P)}{(1 - M_x^2)}, \tag{5.18}$$

where

$$\begin{aligned} w_a^P =& \frac{1}{\beta^2}\Big(\mathbf{A}_L(t_{n+1}) + (\beta^2 - 2)\mathbf{A}_L(t_n) + \mathbf{A}_L(t_{n-1})\Big) - I\Big[\mathbf{A}_L(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](x_a), \\ w_b^P =& \frac{1}{\beta^2}\Big(\mathbf{A}_R(t_{n+1}) + (\beta^2 - 2)\mathbf{A}_R(t_n) + \mathbf{A}_R(t_{n-1})\Big) - I\Big[\mathbf{A}_R(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](x_b). \end{aligned}$$

Taking $A_D(t_i)$ and $A_U(t_i)$ as fixed for $\mathcal{L}_y^{-1}$ of the second order term $\mathcal{C}^{(1)}$ along the line $\partial\Omega = [y_a, y_b]$, we arrive at two equations in two unknowns for a_y and b_y:

$$\begin{aligned} \mathbf{A}_D(t_{n+1})(\beta^2 - 2)\mathbf{A}_D(t_n) + \mathbf{A}_D(t_{n-1}) &= \beta^2\Big(I\Big[\mathbf{A}_D + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](y_a) + a_y + M_y b_y\Big), \\ \mathbf{A}_U(t_{n+1})(\beta^2 - 2)\mathbf{A}_U(t_n) + \mathbf{A}_U(t_{n-1}) &= \beta^2\Big(I\Big[\mathbf{A}_U + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](y_b) + M_y a_y + b_y\Big). \end{aligned}$$

where $M_y = e^{-\alpha(y_b - y_a)}$. Solving the linear system for the unknowns a_y and b_y gives

$$a_y = \frac{(w_a^P - M_y w_b^P)}{(1 - M_y^2)}, \quad b_y = \frac{(w_b^P - M_y w_a^P)}{(1 - M_y^2)}, \tag{5.19}$$

where

$$\begin{aligned} w_a^P =& \frac{1}{\beta^2}\Big(\mathbf{A}_D(t_{n+1}) + (\beta^2 - 2)\mathbf{A}_D(t_n) + \mathbf{A}_D(t_{n-1})\Big) - I\Big[\mathbf{A}_D(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](y_a) \\ w_b^P =& \frac{1}{\beta^2}\Big(\mathbf{A}_U(t_{n+1}) + (\beta^2 - 2)\mathbf{A}_U(t_n) + \mathbf{A}_U(t_{n-1})\Big) - I\Big[\mathbf{A}_U(t_n) + \frac{\mu_0}{\alpha^2}\mathbf{J}^n\Big](y_b). \end{aligned} \tag{5.20}$$

For the operator $\mathcal{C}^{(2)} + \mathcal{D}^{(2)}$ acting on $\mathcal{C}^{(1)}[A^n]$, which is the higher order correction, one can obtain similar equations, except that the values of $A_L(t_i)$, $A_R(t_i)$, $A_D(t_i)$ and $A_U(t_i)$ at the boundaries are explicitly zero. This is due to linearity and the leading order operator satisfies the boundary condition exactly.

5.4.2 Embedded Boundary Method, An effective Neumann to Dirichlet Map for Creating Ghost Points

Given that we know how to solve for the Dirichlet boundary conditions for equations 5.16 and 5.17, as in [14], we would like to extend this process to the PEC case. As discussed in [13], the reason to employ an embedded boundary approach for a differential boundary condition has to do with stability. The difference here is we need to develop an iterative map to find the homogeneous coefficients a and b for each piece of the operator instead of directly computing these coefficients.

Our embedded boundary method is based on our initial work in [13]. In this paper, we developed a second-order embedded boundary method for Neumann boundary conditions which was second-order accurate in time and space. Using the same ideas, we extended this method to PEC boundary conditions for the vector potential in 2D. However, in [13], direct application through the ADI of the operators is more direct. Extending this work to fourth-order involves developing an approach to solving for the boundary conditions for the operators $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$, and $\mathcal{D}^{(2)}$.

We use an iterative method to obtain accurate values at the boundaries using an initial approximation and correct it by imposing our PEC boundary condition through a fixed point iteration. The fixed point iteration is a multi-step process that converts the derivative condition along the boundary to a Dirichlet boundary condition, to be set at the ghost point. These Dirichlet boundary values force the solution to satisfy the PEC condition at the boundary to within the tolerance of iteration of the fixed point method. **It should be noted that the iteration is local at the boundary and does not involve re-computing the interior sweeps, making this update cost-effective.**

To understand the method, we start by considering a collection of uniform points with a boundary passing through, see figure 5.3. We define the ghost points to be the collection of points that are greater than one half of a grid spacing away, but less than two and a half grid spacings away, from the boundary. As in our work in [13], starting from the

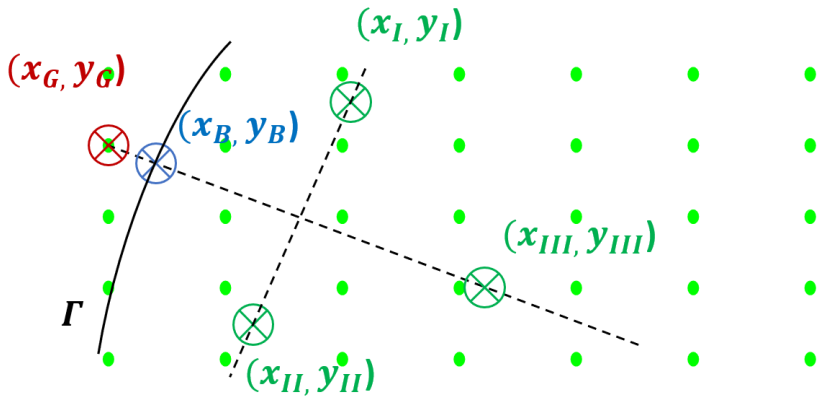


Figure 5.3: Geometry of 2D embedded PEC boundary method with a boundary point (x_B, y_B), ghost point (x_G, y_G), and interpolation points (x_I, y_I), (x_{II}, y_{II}), and (x_{III}, y_{III}).

ghost point we define a normal to the surface, along which we will enforce the boundary condition. We define the fixed point method along this normal. In figure 5.3, the ghost point being considered is the red grid point labeled (x_G, y_G), the blue point labeled (x_B, y_B) is the boundary point and there are three interior points needed that are related to the normal, at (x_I, y_I), (x_{II}, y_{II}), and (x_{III}, y_{III}). Their distances from the boundary point (x_B, y_B) are $\xi_I = |(x_I, y_I) - (x_B, y_B)| = \sqrt{2}\Delta s_I$, $\xi_{II} = |(x_{II}, y_{II}) - (x_B, y_B)| = \sqrt{2}\Delta s_I$, and $\xi_{III} = |(x_{III}, y_{III}) - (x_B, y_B)| = 2\Delta s_I$, where we will typically take $\Delta s_I = \sqrt{2}\Delta x$. The fixed point method starts by assuming we know $\mathbf{A}_I$, $\mathbf{A}_{II}$, $\mathbf{A}_{III}$, and $\partial_T \mathbf{A}_B$, which is taken as a solution at $\{(x_I, y_I)$, (x_{II}, y_{II}) and $(x_{III}, y_{III})\}$ and the tangential derivative of a solution at (x_B, y_B). Here we enforce $\partial_T \mathbf{A}_B = 0$ by making $\partial_T P|_{(x_B, y_B)} = 0$ one of the conditions we use to solve for the coefficients of the Hermite-Birkhoff interpolating polynomial, $P(x, y) = c_0 + c_1 x + c_2 y + c_3 xy$. Enforcing that the Hermite-Birkhoff interpolates these points and explicitly solving for $\mathbf{A}_G$ gives:

$$\mathbf{A}_G^{l+1} = \Gamma_I \mathbf{A}_I^l + \Gamma_{II} \mathbf{A}_{II}^l + \Gamma_{III} \mathbf{A}_{III}^l + O(\Delta x^2), \tag{5.21}$$

where,

$$\begin{aligned}
\Gamma_I &= 1 + \frac{\gamma_3(\gamma_4+\gamma_5) - \gamma_6(\gamma_1+\gamma_2)}{(\gamma_4\gamma_2 - \gamma_5\gamma_1)}, \\
\Gamma_{II} &= \frac{(\gamma_6\gamma_2 + \gamma_3\gamma_5)}{(\gamma_4\gamma_2 - \gamma_5\gamma_1)}, \\
\Gamma_{III} &= \frac{(\gamma_6\gamma_1 + \gamma_3\gamma_4)}{(\gamma_4\gamma_2 - \gamma_5\gamma_1)},
\end{aligned} \tag{5.22}$$

where,

$$\begin{aligned}
\gamma_1 &= x_{II} - x_I + y_{II} - y_I, \\
\gamma_2 &= x_I - x_{III} + y_I - y_{III}, \\
\gamma_3 &= x_G - x_I + y_G - y_I, \\
\gamma_4 &= (x_{II} - x_I)(x_B - y_B) + x_I y_I, \\
\gamma_5 &= (x_I - x_{III})(x_B - y_B) + x_{III} y_{III}, \\
\gamma_6 &= (x_B - y_B)(x_G - x_1) + x_G(y_G - y_1).
\end{aligned} \tag{5.23}$$

We have introduced additional notation of $l+1$ and l, which designate how we will set up a fixed point iteration using the Hermite-Birkhoff interpolant. The iteration itself is a simple first order fixed point method. Let $\mathbf{w}^{(l)}$ and $\mathbf{w}^{(l+1)}$ be the solutions at the l^{th} and $(l+1)^{th}$ iterations. We choose a tolerance tol, and maximum number of allowed iterations mit, and define a stopping criterion as $|\mathbf{w}^{(l+1)} - \mathbf{w}^{(l)}|_\infty < tol$ OR $nit > mit$ where nit is the current iteration number. We will now discuss the quasi local process for obtaining $\mathbf{w}^{(l+1)}$ given the rest of the information.

For each term in the operator on the right hand side of the second order solution given by equation 5.16, or fourth order in time solution given by equation 5.17, we must identify the correct ghost point that allows that term to guarantee that the tangential derivative is zero along the boundary. Taking the $\mathcal{C}^{(1)}[\mathbf{A}^n]$ operator and expanding the operator out we

have $\mathcal{C}^{(1)}[\mathbf{A}^n] = \mathcal{L}_x^{-1}[\mathbf{A}^n] + \mathcal{L}_y^{-1}[\mathbf{A}^n] - \mathcal{L}_y^{-1}\mathcal{L}_x^{-1}[\mathbf{A}^n] - \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}[\mathbf{A}^n]$. The fixed point iteration identifies the ghost point for $\mathbf{w_x} = \mathcal{L}_x^{-1}[\mathbf{A}^n]$, $\mathbf{w_y} = \mathcal{L}_y^{-1}[\mathbf{A}^n]$, $\mathbf{w_{xy}} = \mathcal{L}_x^{-1}[\mathbf{w_y}]$, and $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$ such that when solving for the boundary correction terms the operator satisfies $\partial_T \mathbf{w_x}|_{\partial\Omega} = 0$, $\partial_T \mathbf{w_y}|_{\partial\Omega} = 0$, $\partial_T \mathbf{w_{xy}}|_{\partial\Omega} = 0$, $\partial_T \mathbf{w_{yx}}|_{\partial\Omega} = 0$.

Let us consider $\mathbf{w_x}$ and $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$, knowing that $\mathbf{w_y}$ and $\mathbf{w_{xy}}$ are similar. The iterative process for $\mathbf{w_x}$ starts by making an initial guess at the direct boundary values $\mathbf{w_x}|_{\partial\Omega}$, using an extrapolate in time ${\mathbf{w_x}}^{n+1,(0)} \approx 3{\mathbf{w_x}}^{n} - 3{\mathbf{w_x}}^{n-1} + {\mathbf{w_x}}^{n-2}$ at each boundary point (Algorithm 20).

Algorithm 20 Compute $\mathbf{w_x}(= L_x^{-1}[\mathbf{A}])$

1: Compute the interior sweep (particular solution) $I_x^{n+1} = I_x[\mathbf{A}^n]$
2: Initial guess ${\mathbf{w_x}}^{n+1(0)} \approx 3{\mathbf{w_x}}^{n} - 3{\mathbf{w_x}}^{n-1} + {\mathbf{w_x}}^{n-2}$
3: Initialize the iteration counter $nit = 0$
4: **repeat**
5: **for** $k = 1$ to n_y **do**
6: Compute $\mathbf{w_x}_{I(k)}$, and $\mathbf{w_x}_{II(k)}$, at the interpolation points $(x_{I(k)}, y_{(k)})$, and $(x_{II(k)}, y_{(k)})$ and as well as at the interpolation points along the right boundary using bilinear interpolation
7: Compute solution at the ghost points $(x_{aG(k)}, y_{(k)})$ and $(x_{bG(k)}, y_{(k)})$ using the Hermite-Birkhoff interpolant
8: Compute homogeneous coefficients $a_{1(k)}$ and $b_{1(k)}$ using the fact ${\mathbf{w_x}}^{n+1(l)} = I^{n+1} + a_{x(k)}e^{-\alpha(x - x_{aG(k)})} + b_{x(k)}e^{-\alpha(x_{bG(k)} - x)}$
9: Compute solution and update boundary stencil ${\mathbf{w_x}}^{n+1(l+1)} = I^{n+1} + a_{x(k)}e^{-\alpha(x - x_{aG(k)})} + b_{x(k)}e^{-\alpha(x_{bG(k)} - x)}$
10: **end for**
11: $nit = nit + 1$
12: **until** $(|{\mathbf{w_x}}^{n+1(l+1)} - {\mathbf{w_x}}^{n+1(l)}| < tol$ OR $nit > mit)$

Given the update on $\mathbf{w_x}$, the next step is to consider $\mathbf{w_{yx}} = \mathcal{L}_y^{-1}[\mathbf{w_x}]$. The process starts with the initial guess ${\mathbf{w_x}}^{n+1,(0)} \approx 3\mathbf{A}^n - 3\mathbf{A}^{n-1} + \mathbf{A}^{n-2}$ at the boundary (Algorithm 21).

The same process is done for $\mathbf{w_y}$ and $\mathbf{w_{xy}}$. The Hermite-Birkhoff interpolate enforces that the tangential derivative is zero for $\mathbf{w_x}$, $\mathbf{w_y}$, $\mathbf{w_{xy}}$ and $\mathbf{w_{yx}}$ such that $\mathcal{C}^{(1)}[\mathbf{A}^n]$ satisfies the tangential derivative condition on $\partial\Omega$ to within tolerance.

For the fourth order formulation, we compute the boundary conditions for $\mathcal{C}^{(1)}[\mathbf{A}^n]$ and then we repeat the process for $\mathcal{C}^{(2)}$ and $\mathcal{D}^{(2)}$ acting on $\mathcal{C}^{(1)}[\mathbf{A}^n]$.

Algorithm 21 Compute $\mathbf{w_{yx}}(= L_y^{-1}[\mathbf{w_x}])$

1: Compute the interior sweep (particular solution) $I_y^{n+1} = I_y[\mathbf{w_x}^n]$
2: Initial guess $\mathbf{w_{yx}}^{n+1(0)} \approx 3\mathbf{A}^n - 3\mathbf{A}^{n-1} + \mathbf{A}^{n-2}$
3: Initialize the iteration counter $nit = 0$
4: **repeat**
5: **for** $k = 1$ to n_x **do**
6: Compute $\mathbf{w_{yx}}_{I(k)}$, and $\mathbf{w_{yx}}_{II(k)}$, at the interpolation points $(x_{(k)}, y_{I(k)}$, and $(x_{(k)}, y_{II(k)})$ and as well as at the interpolation points along the right boundary using bilinear interpolation
7: Compute solution at the ghost points $(x_{(k)}, y_{aG(k)})$ and $(x_{(k)}, y_{bG(k)})$ using the Hermite-Birkhoff interpolant
8: Compute homogeneous coefficients $a_{1(k)}$ and $b_{1(k)}$ using the fact $\mathbf{w_{yx}}^{n+1(l)} = I^{n+1} + a_{y(k)}e^{-\alpha(y-y_{aG(k)})} + b_{y(k)}e^{-\alpha(y_{bG(k)}-y)}$
9: Compute solution and update boundary stencil $\mathbf{w_{yx}}^{n+1(l+1)} = I^{n+1} + a_{y(k)}e^{-\alpha(y-y_{aG(k)})} + b_{y(k)}e^{-\alpha(y_{bG(k)}-y)}$
10: **end for**
11: $nit = nit + 1$
12: **until** $(|\mathbf{w_{yx}}^{n+1(l+1)} - \mathbf{w_{yx}}^{n+1(l)}| < tol$ OR $nit > mit)$

Further as detailed in [13] we need to include an artificial dissipation term to maintain stability for these embedded boundary domains. Thus we have

$$\mathbf{A}^{n+1} - 2\mathbf{A}^n + \mathbf{A}^{n-1} = -\beta^2\mathcal{C}_{xy}^{(1)}[\mathbf{A}^n] - \left(\beta^2\mathcal{D}_{xy}^{(2)} - \frac{\beta^4}{12}\mathcal{C}_{xy}^{(2)}\right)\mathcal{C}_{xy}^{(1)}[\mathbf{A}^n] + \epsilon\mathcal{C}_{xy}^{(3)}[\mathbf{A}^{n-1}].$$

where ϵ is an artificial dissipation coefficient that satisfies $0 < \epsilon < 1$. The value of $\mathcal{C}_{xy}^{(3)}[\mathbf{A}^n]$ at the previous time step can be used in place of $\mathcal{C}_{xy}^{(3)}[\mathbf{A}^{n-1}]$ at the current time step, and we need to go to one more computing level to obtain $\mathcal{C}_{xy}^{(3)}[\mathbf{A}^n]$ [68]. We obtain $\mathbf{A}_I$, $\mathbf{A}_{II}$, and $\mathbf{A}_{III}$ approximately, using the bilinear interpolation. Suppose the interpolation point $\mathbf{A}_I$ lies in a rectangular cell with corners (x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), and (x_i, y_{j+1}). Then we have the following approximation for $\mathbf{A}_I$

$$\mathbf{A}_I = w_1\mathbf{A}_{i,j} + w_2\mathbf{A}_{i+1,j} + w_3\mathbf{A}_{i+1,j+1} + w_4\mathbf{A}_{i,j+1}, \tag{5.24}$$

where

$$w_1 = \frac{(x_{i+1} - x_I)(y_{j+1} - y_I)}{\Delta x \Delta y}, \qquad w_2 = \frac{(x_I - x_i)(y_{j+1} - y_I)}{\Delta x \Delta y},$$
$$w_3 = \frac{(x_I - x_i)(y_I - y_j)}{\Delta x \Delta y}, \qquad w_4 = \frac{(x_{i+1} - x_I)(y_I - y_j)}{\Delta x \Delta y}. \tag{5.25}$$

5.5 Experimental Results

5.5.1 Square Cavity Rotated Through Different Angles

For this experiment, we chose a square cavity bounded by a PEC, placed a point source at the center of the cavity, and performed a ping test and mode analysis. First, we computed the vector potential **A** at a point inside the box using our scheme, then computed the derived frequency by taking FFT over the time history of the measured **A**, and finally analyzed the frequency modes for varying CFL value, step size, and rotation angle. Further, we have transformed into physical units here in contrast to the normalized units we have used up to this point.

We chose a square box $21cm \times 21cm$ with PEC boundaries in a square domain ($\Omega = [-21cm, 21cm]^2$), placed a point source 1 at the center of the box,$(0, 0)$, and turned it on for a single time step $t = \Delta tns$. The vector potential is measured at the point $(3.36cm, 3.36cm)$ for the time period $t = [\Delta tns, Tns]$. The derived frequencies for different CFL values (0.5, 1.0, 2.0), rotation angles (0^o, 31.42^o, 45^o), and resolutions are summarized in tables 5.5.1 and 5.5.1. Table 5.5.1 consists of frequencies obtained for the CFL value 1. Here, we set the wave speed c = 30 Gcms^{-1}, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. Figure 5.4 shows the frequency distribution for $\theta = 31.42^o$ which consists the 1-1 strong mode fundamental frequency 1GHz as expected and Figure 5.5 shows a frequency mode computed for different resolutions with CFL value 1. We see clear convergence to the analytically computed 1-1 mode fundamental frequency 1GHz. For the fixed CFL and wave speed, time step size Δt reduces with reducing spatial step size Δx. Hence, the amplitude

is reducing by the decreasing spatial step size.

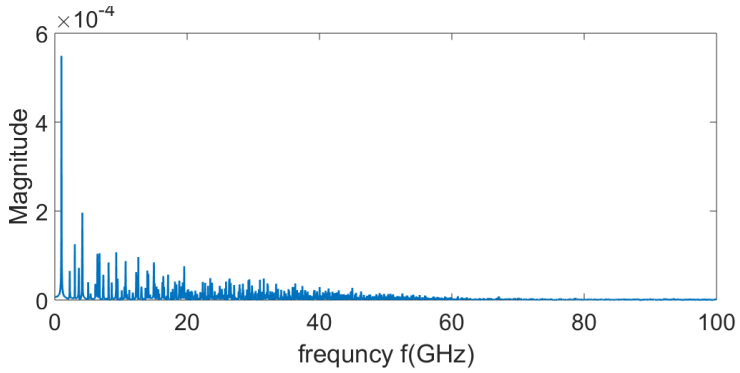


Figure 5.4: Frequency distribution for 31.42^o rotated $21cm \times 21cm$ square cavity computed using the measured vector potential at the point $(3.36cm, 3.36cm)$ for the impulse response, h = 0.084 cm

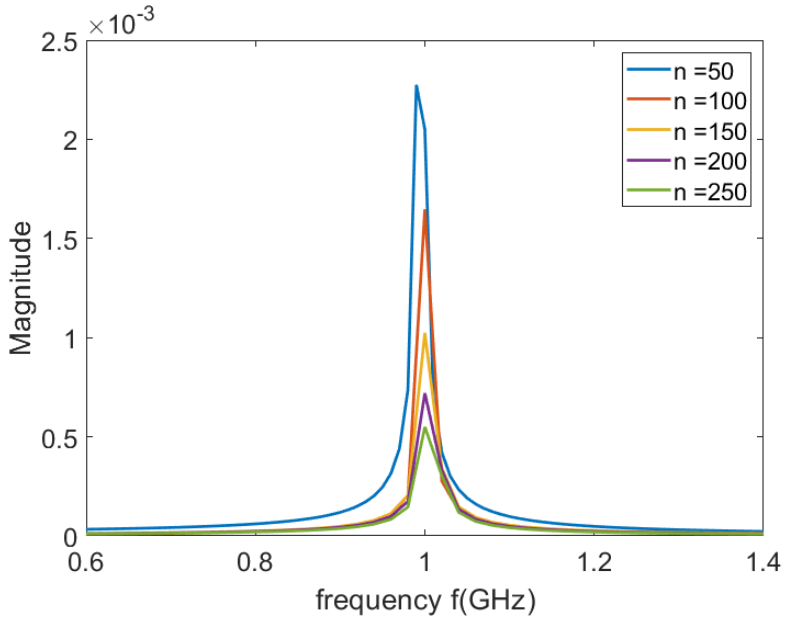
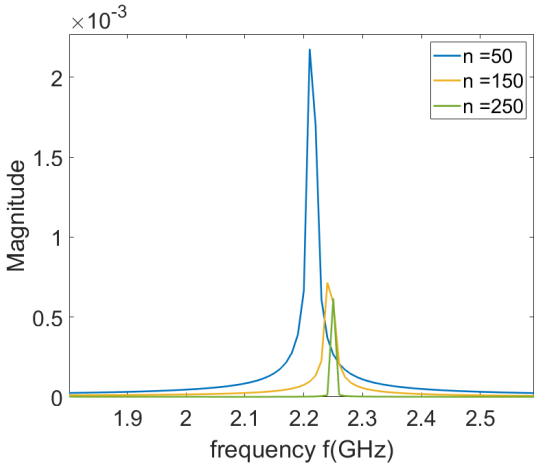
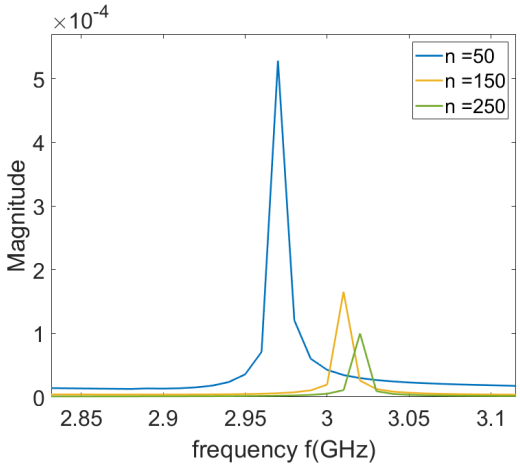


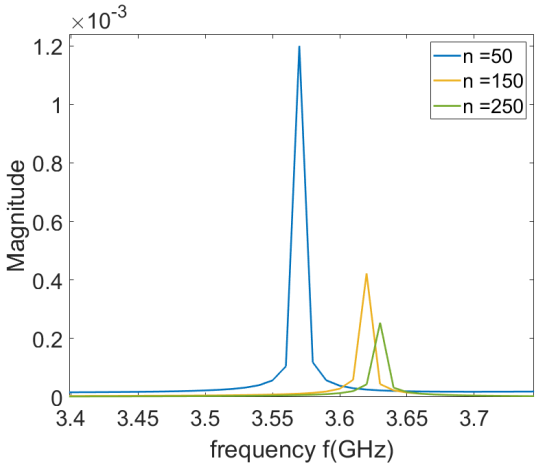
Figure 5.5: A strong fundamental mode for 31.42^o rotated $21cm \times 21cm$ square cavity computed for different resolutions.



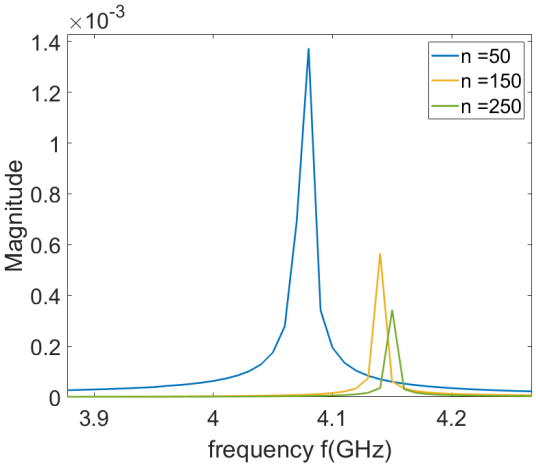
(a) 1-3 mode, $f = 2.2588GHz$



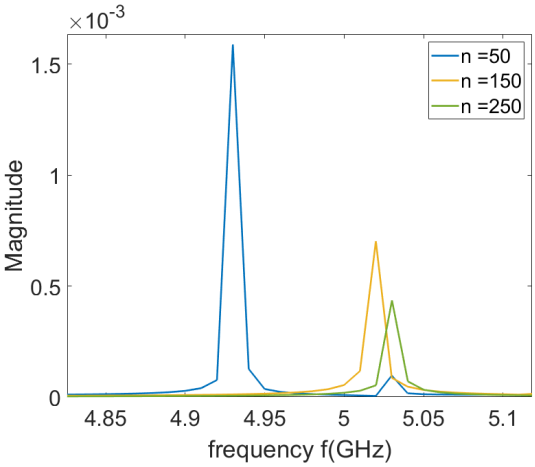
(b) 3-3 mode, $f = 3.0305GHz$



(c) 1-5 mode, $f = 3.6422GHz$



(d) 3-5 mode, $f = 4.1650GHz$



(e) 5-5 mode, $f = 5.0508GHz$

Figure 5.6: Convergence plots for the natural modes (a) 1-3, (b) 3-3, (c) 1-5, (d) 3-5, and (e) 5-5 with analytically computed frequencies (a) $f = 2.2588GHz$, (b) $f = 3.0305GHz$, (c) $f = 3.6422GHz$, (d) $f = 4.1650GHz$, and (e) $f = 5.0508GHz$ for 31.42^o rotated $21cm \times 21cm$ square cavity computed for different resolutions.

	$CFL = 0.5$, $T = 50ns$			$CFL = 2$, $T = 200ns$		
Rotated angle, θ	45^o	31.42^o	0^o	45^o	31.42^o	0^o
Grid size, N						
50^2	0.98	0.99	0.96	0.98	0.98	0.96
100^2	1.00	1.00	0.98	1.00	0.99	0.98
150^2	1.00	1.00	0.98	0.99	1.00	0.98
200^2	1.00	1.00	1.00	1.00	1.00	0.99
250^2	1.00	1.00	1.00	1.00	1.00	1.00

Table 5.1: Frequency (in GHz) obtained at the point $(3.36cm, 3.36cm)$ using the ping test.

Rotated angle, θ	0^o	31.42^o	45^o
Grid size, N			
50^2	9.599802E-1	9.899800E-1	9.899800E-1
100^2	9.899800E-1	9.999800E-1	9.999800E-1
150^2	9.899800E-1	9.999800E-1	9.999800E-1
200^2	1.000015E+0	1.000115E+0	1.000015E+0
250^2	1.000008E+0	1.000108E+0	1.000008E+0

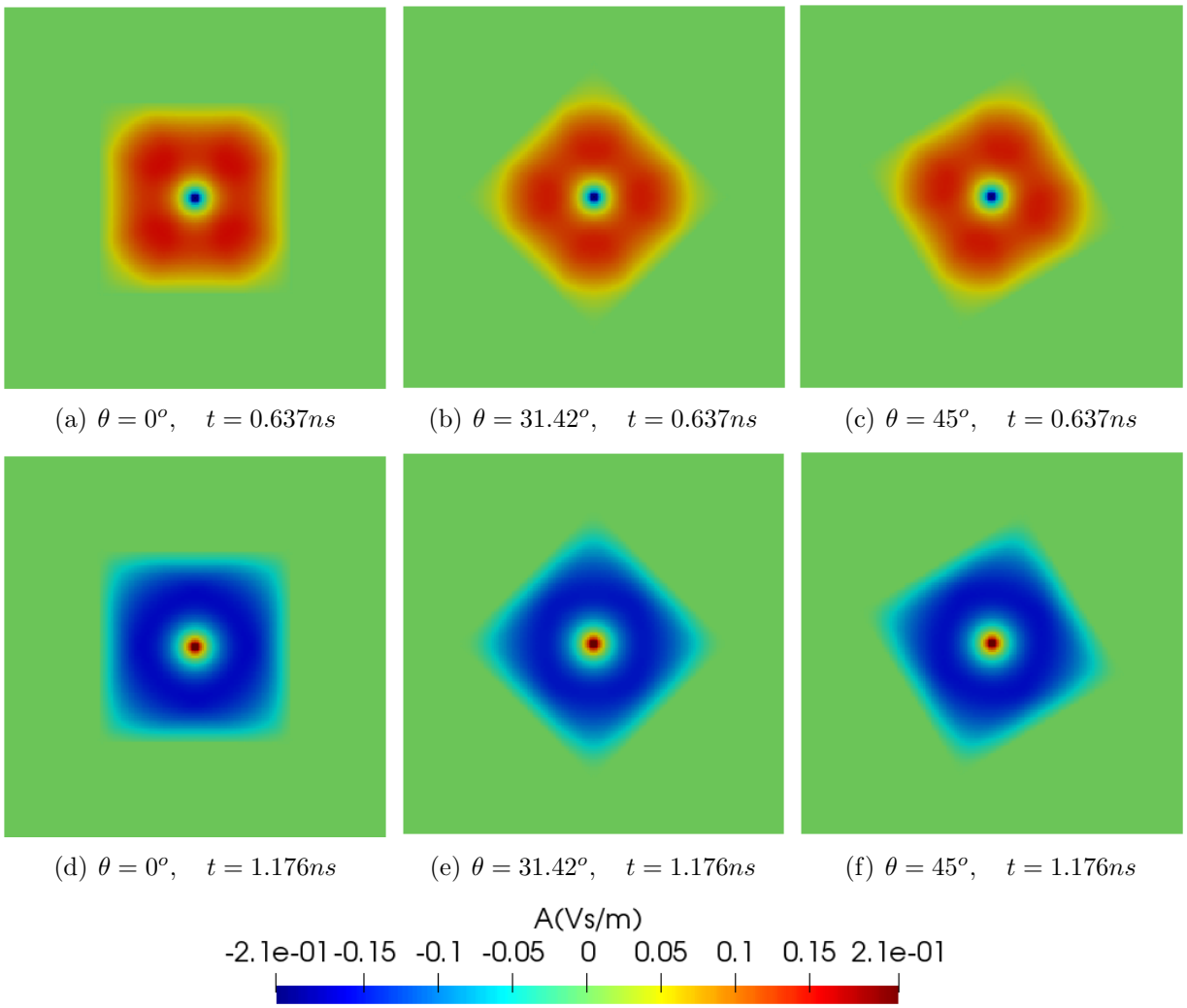
Table 5.2: Frequency (in GHz) obtained at the point $(3.36cm, 3.36cm)$ using the ping test for CFL 1.

Further, we evaluated the frequencies of the normal modes 1-3, 3-3, 1-5, 3-5, and 5-5. Our convergence plots (Figure 5.6) show the clear convergence to the analytically computed frequencies $2.2588GHz$, $3.0305GHz$, $3.6422GHz$, $4.1650GHz$, and $5.0508GHz$ for the modes 1-3, 3-3, 1-5, 3-5, and 5-5 respectively.

Figure 5.7 shows the time evolution of the potential $\mathbf{A}$ generated by a point source $sin(2\pi ft)$ with $f = 1GHz$ placed at the center of the box. We chose the same domain as used in the previous experiment and set the grid size to be 100×100, time step size $\Delta t = 7ps$.

5.5.1.1 Convergence Studies and Error Analysis

We evaluate the consistency of our scheme via (1) the ping test and (2) space-time convergence studies for the magnetic vector potential $\mathbf{A}$ in the PEC square cavity.
First, we perform a convergence study given the initial condition $sin\left(\frac{m\pi x}{L}\right) sin\left(\frac{n\pi y}{H}\right)$ with frequency mode 3-2 $(m = 3, n = 2)$ over a square domain $[0cm, 21cm]^2$ and compare our



(a) $\theta = 0^o, \quad t = 0.637ns$ (b) $\theta = 31.42^o, \quad t = 0.637ns$ (c) $\theta = 45^o, \quad t = 0.637ns$

(d) $\theta = 0^o, \quad t = 1.176ns$ (e) $\theta = 31.42^o, \quad t = 1.176ns$ (f) $\theta = 45^o, \quad t = 1.176ns$

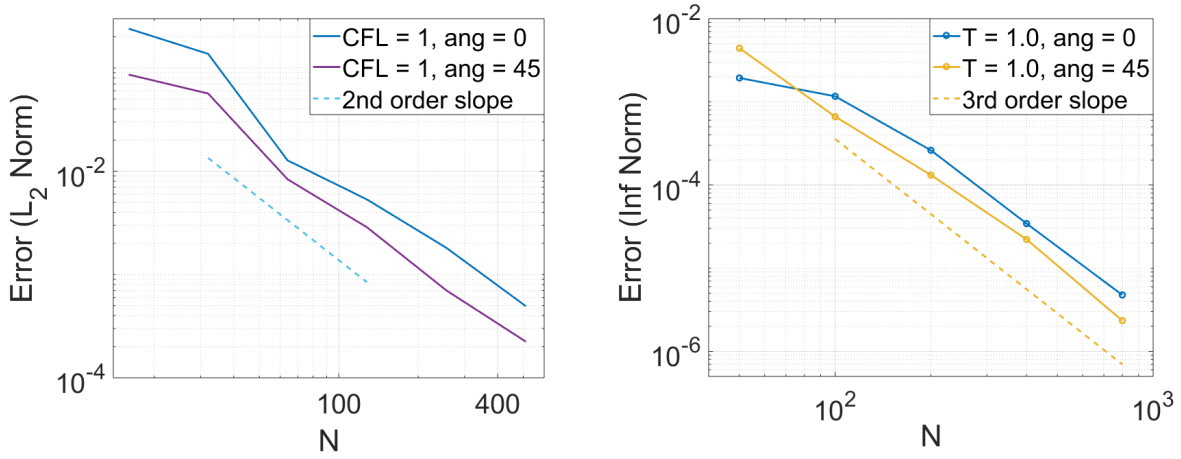
Figure 5.7: Time evolution of a point source field $sin(2\pi ft)$ with $f = 1GHz$ placed at the center of a PEC square box of grid size 100×100 rotated by the angles 0^o, 31.42^o, 45^o, time step size $\Delta t = 7ps$.

numerical solution to the exact solution given by,

$$\mathbf{A}(t, x, y) = \cos\left(c\sqrt{(\frac{m\pi}{L})^2 + (\frac{n\pi}{H})^2}t\right) \sin\left(\frac{m\pi x}{L}\right) \sin\left(\frac{n\pi y}{H}\right)$$

, where the wave speed c is chosen to be $30Gcms^{-1}$. Our L_2 norm error plots of the solution at time $T = 1.0ns$ with varying resolutions and a fixed CFL($= 1$) for the cases of mesh aligned and nonaligned (rotated by 45^o) square cavities show second order accuracy (Figure 5.8 - (a)). We also note that the tangential derivative along the boundary converges with

third order accuracy (Figure 5.8 - (b)).



(a) Space convergence of solution

(b) Space convergence of boundary derivative

Figure 5.8: Convergence plots for a) space using entire solution and b) space using boundary derivative on a PEC square domain $[0cm, 21cm]^2$. This study measures L_2 norm of the error at time $T = 1.0ns$ compared with the analytical solution, for the cases of mesh aligned and nonaligned (rotated by 45^o).

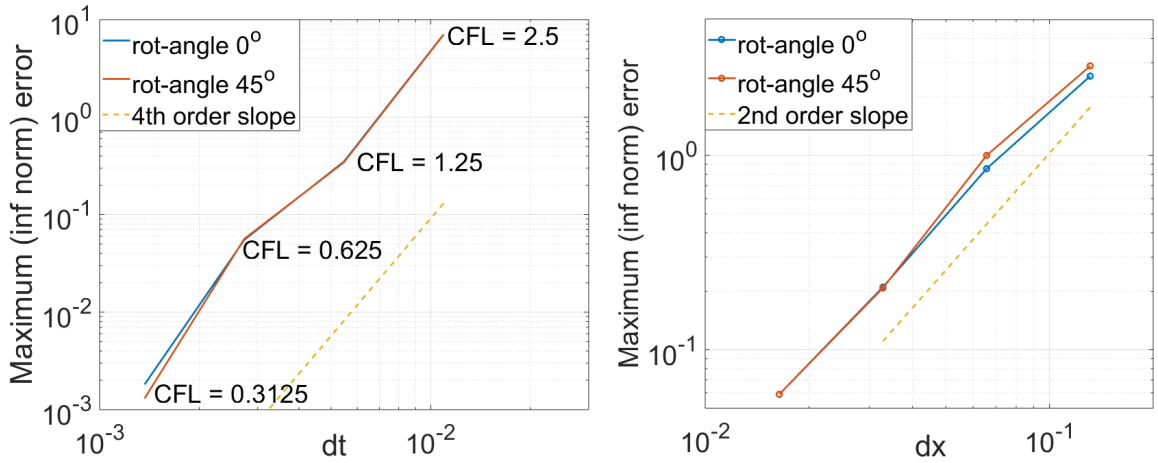
For the second evaluation, we compute the error with our fundamental frequency computation using the ping test explained above and summarize it in Table 5.5.1.1. The error is the difference between the frequencies (1-1 mode) obtained analytically ($= 1GHz$) and numerically using our scheme.

	$CFL = 0.5$, $T = 50ns$		$CFL = 2$, $T = 200ns$	
Rotated angle θ	31.42^o	0^o	31.42^o	0^o
Grid size, N				
50^2	1.001980E-2	4.501910E-2	1.501970E-2	4.501910E-2
100^2	1.999960E-5	2.001960E-2	5.019900E-3	2.001960E-2
150^2	1.999960E-5	2.001960E-2	1.999960E-3	1.501970E-2
200^2	1.999960E-5	1.500023E-5	5.015075E-4	4.985075E-4
250^2	8.0000064E-6	8.000064E-6	5.0080040E-5	8.000064E-6

Table 5.3: Numerical error in the frequency computation using the ping test at the point $(3.36cm, 3.36cm)$.

Finally, we perform a self-refinement study for time and space convergence using a point source. Figure 5.9 shows the time and space convergence plots for the cases of mesh aligned and nonaligned (rotated by 45^o) square cavities with the same configuration as used in

the above experiment done for the time evolution of a point source field. For the time convergence test, we maintain the resolution at 320×320 and reduce the time step size $\Delta t = 10.9ps \quad to \quad 0.3418ps$. For the space convergence test, we maintain the CFL value as 1 and reduce the spatial step size $\Delta x = 1.313cm$ to $0.082cm$. We obtained fourth-order convergence in time and second-order in space.



(a) Time convergence (b) Space convergence

Figure 5.9: Convergence plots for (a) time and (b) space on a PEC a square domain $[0cm, 21cm]^2$. This study measures L_∞ norm of the error at time $T = 1.0ns$ compared with the analytical solution, for the cases of mesh aligned and non-aligned (rotated by 45^o).

5.5.2 Square Cavity with a leak (diffraction Q)

The Q factor (quality factor) of a resonator is a measure of the damping, with a lower Q indicating higher damping [33]. It measures the harmonic losses of the oscillations. The diffraction Q factor is the external Q factor which characterized the energy loss due to radiation. The value of Q is infinite for a closed cavity and finite for an open cavity and it can be used to determine the amount of energy loss. The diffraction Q can be expressed as,

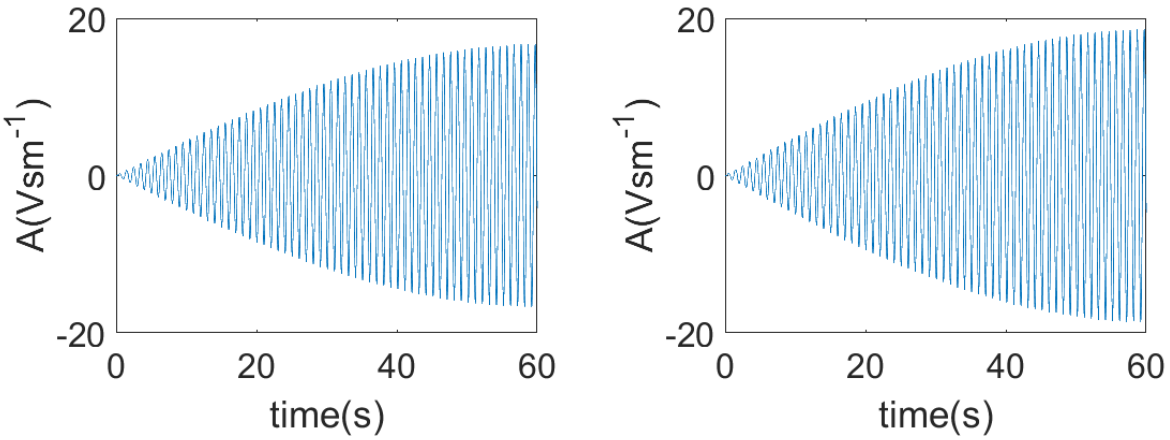
$$Q = -\frac{\pi f(t_2 - t_1)}{ln(\frac{A_1 t_1}{A_2 t_2})}$$

where A_1 and A_2 are the vector potentials at time t_1 and t_2.

In the second experiment, an open boundary is placed along the center of the right edge of the square cavity, and the diffraction quality factor Q is obtained for an oscillating point source ($sin(2\pi ft)$, $f = 1GHz$) placed at the center.

Every configuration was the same as above, except for imposing an outflow boundary condition along the open boundary and keeping the Gaussian pulse running for the entire time period $[\Delta tns, Tns]$. The vector potential **A** is measured at the point $(3.36cm, 3.36cm)$ and the computation of Q is done for aligned and nonaligned mesh cases. For non aligned meshes, we rotate the square by an angle $\theta = 45^o$.

Analytic calculation of the quality factor treating an open cavity like a transmission line with a load simulating free space (377Ω) gives a value of Q=24 [36]. The Q values obtained from our scheme (Q = 26.5768 for $\theta = 0^o$ and Q = 25.7875 for $\theta = 31.42^0$) are very close to the analytically obtained value. Figure 5.10 shows the time evaluation of vector potential **A** at the point $(3.36cm, 3.36cm)$ for both cases.



(a) $\theta = 0^o$, Q = 26.5768 (b) $\theta = 31.42^0$, Q = 25.7875

Figure 5.10: Time evaluation of **A** in (a) mesh aligned and (b) $\theta = 31.42^0$ rotated square cavities with a leak imposed by outflow boundary condition and diffraction Q which is obtained for an oscillating point source ($sin(2\pi ft)$, $f = 1GHz$)

5.5.3 A6 Magnetron

We chose a 2D A6 magnetron to test the applicability of our PEC scheme to complicated geometry. For this experiment, a 2D domain of $\Omega = [0cm, 5cm]^2$ is chosen with radius of the vane resonators $r_v = 4.11cm$, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$, angular width of vane, 20^o, and cavity angle, 40^o. We set the grid size to be 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. The embedded PEC boundary condition is imposed over the boundary stencil during the $x-$ and $y-$ sweeps.

This experiment was conducted for frequency mode analysis. A point source was placed in the center of the A6M and the frequency distribution examined using a ping test at the point $(1.4063cm, 0.8203cm)$. We obtained six strong frequency modes as shown in figure 5.11, associated with the first two passbands, clearly showing the effects of all six symmetric resonances.

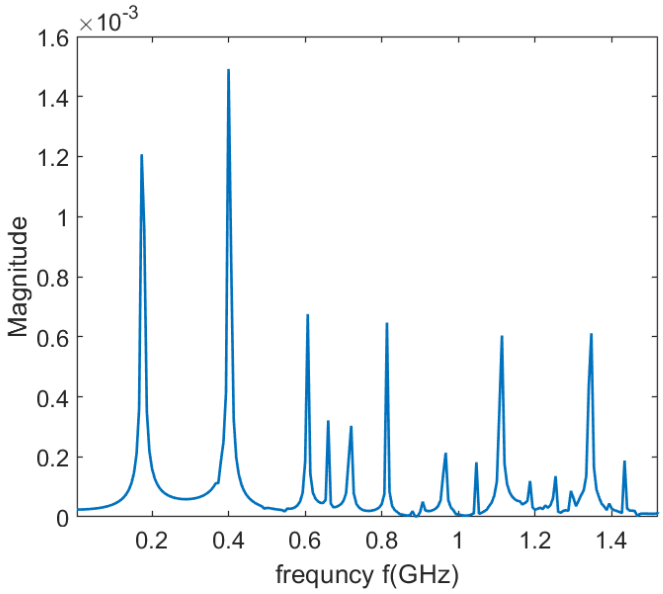
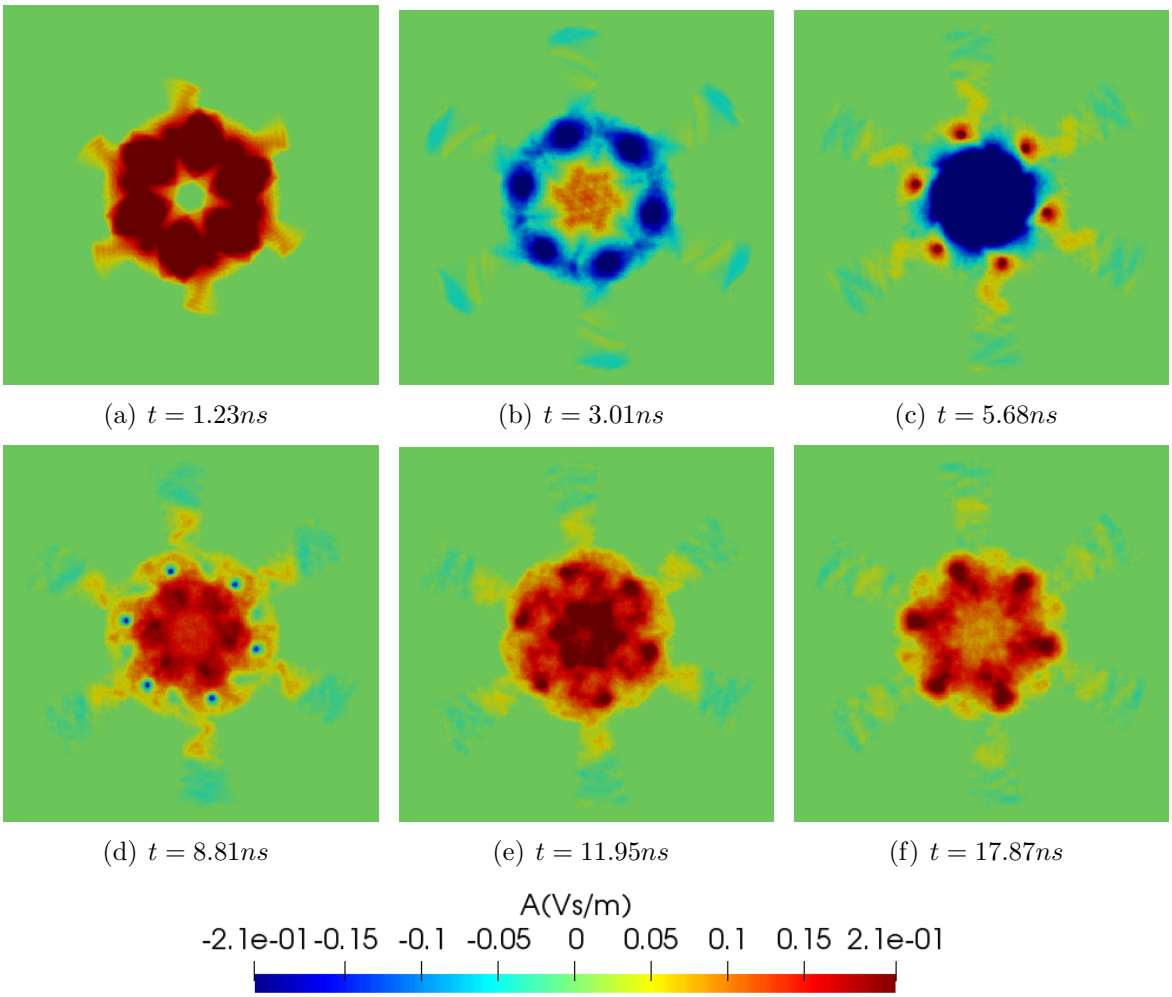


Figure 5.11: Frequency spectrum of 2D A6M with vane resonators $r_v = 4.11cm$, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$, angular width of vane, 20^o, and cavity angle, 40^o, the grid size 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$.

We further simulated the A6 magnetron with a transparent cathode (5.12) which is

mimicked by using six emitters/point sources and calculated the time evolution for the same configuration as above. The formulation maintains symmetry and high accuracy for relatively coarse-grained solutions. For example, in the A6 magnetron results, there are only 12 points across the neck of the magnetron vanes.



(a) $t = 1.23ns$ (b) $t = 3.01ns$ (c) $t = 5.68ns$

(d) $t = 8.81ns$ (e) $t = 11.95ns$ (f) $t = 17.87ns$

Figure 5.12: Evolution of the transparent cathode A6M with vane resonators $r_v = 4.11cm$, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$, angular width of vane, 20^o, and cavity angle, 40^o, the grid size 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$.

5.5.4 Rising Sun Magnetrons

We chose Rising Sun magnetrons with 12 and 18 cavities for further evaluation of our scheme to complicated geometry. Figure shows the 2D view of the 18 cavity rising sun

magnetron (AX9) with the cathode of radius r_c and the anode with inner radius r_a, short vane radius r_{vs} and long vane radius r_{vl}. For this experiment, a 2D domain of $\Omega = [0cm, 5cm]^2$ is chosen with radius of the wider and shorter vane resonators $r_{vs} = 4.11cm$, and $r_{vl} = 4.91cm$ respectively, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$. For the 12 cavity rising sun magnetron , we chose the angular width of vane, 12^o, and cavity angle, 18^o. For the 18 cavity rising sun magnetron , we chose the angular width of vane, 12^o, and cavity angle, 8^o. We set the grid size to be 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$. The embedded PEC boundary condition is imposed over the boundary stencil during the $x-$ and $y-$ sweeps.

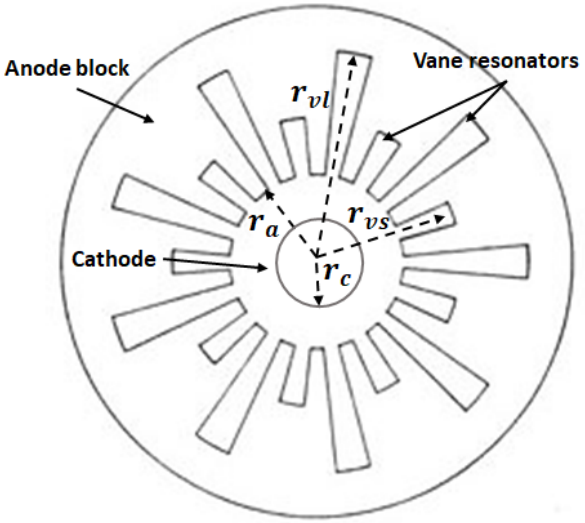
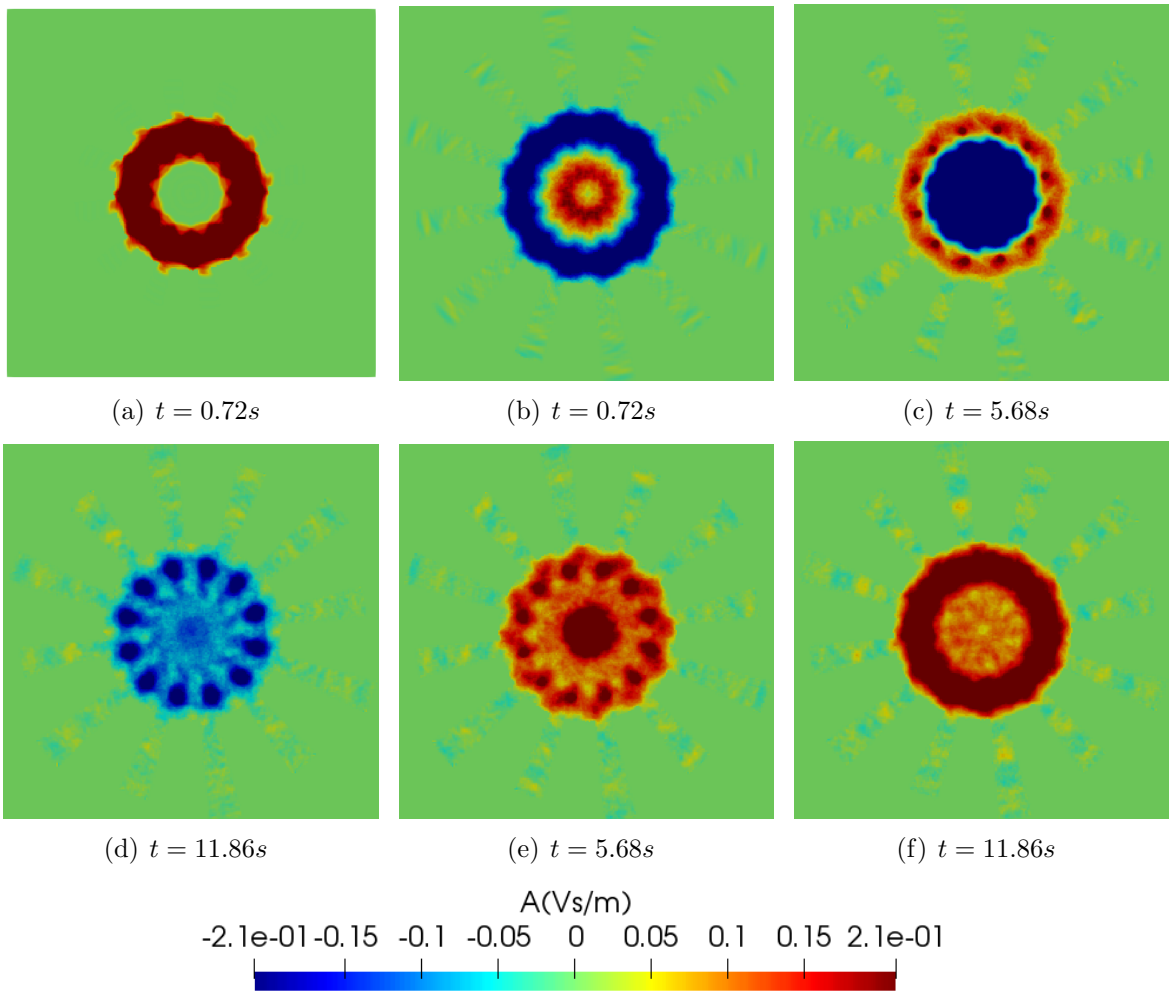


Figure 5.13: 2D view of the 18 cavity rising sun magnetron (AX9) with a cathode of radius r_c and anode of inner radius r_a, short and long cavity radii r_{vh} and r_{vl}.

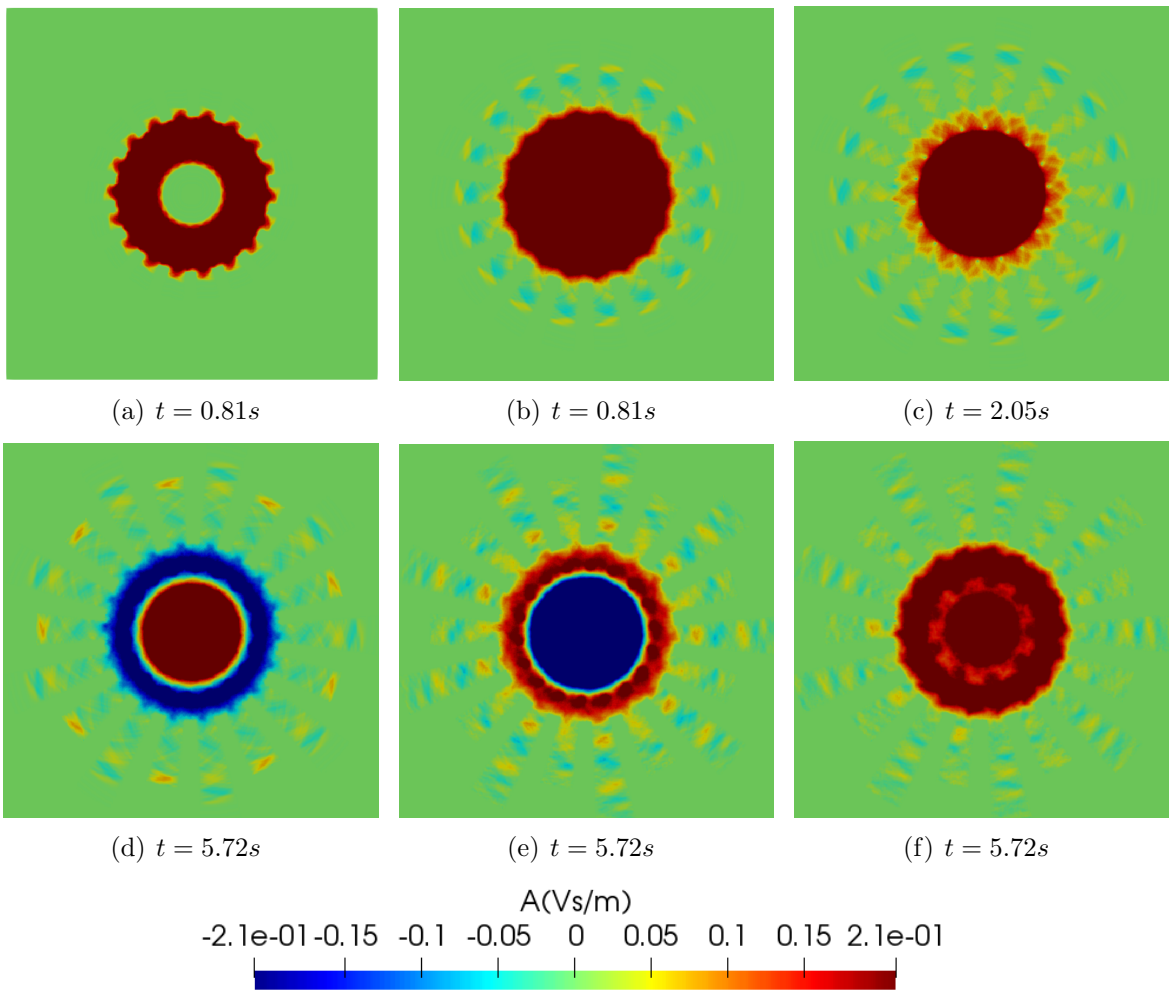
We simulated the rising sun magnetrons with transparent cathodes (Figures 5.14 and 5.15) which are mimicked by using 12 and 18 emitters/point sources for 12 and 18 cavity rising sun magnetrons respectively and calculated the time evolution for the same configuration as above. The formulation maintains symmetry and high accuracy for relatively coarse-grained solutions.



(a) $t = 0.72s$ (b) $t = 0.72s$ (c) $t = 5.68s$

(d) $t = 11.86s$ (e) $t = 5.68s$ (f) $t = 11.86s$

Figure 5.14: Evolution of the transparent cathode 12 cavity rising sun magnetron with vane resonators $r_{vs} = 4.11cm$ and $r_{vl} = 4.91cm$, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$, angular width of vane, 12^o, and cavity angle, 18^o, the grid size 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$.



(a) $t = 0.81s$ (b) $t = 0.81s$ (c) $t = 2.05s$

(d) $t = 5.72s$ (e) $t = 5.72s$ (f) $t = 5.72s$

Figure 5.15: Evolution of the transparent cathode 18 cavity rising sun magnetron (AX9) with vane resonators $r_{vs} = 4.11cm$ and $r_{vl} = 4.91cm$, anode radius $r_a = 2.11cm$, cathode radius $r_c = 1.58cm$, angular width of vane, 8^o, and cavity angle, 12^o, the grid size 128×128, time step size $\Delta t = 39ps$, averaging parameter $\beta = 1.4$ and dissipation coefficient $\epsilon = 0.1$.

5.6 Summary

We derived wave equations for vector and scalar potentials $\mathbf{A}$ and ϕ under the Lorenz gauge condition and obtained the solution for the vector potentials based wave equation subject to PEC boundary condition. We explained the embedded PEC boundary conditions in 2D and the consistency and performance of the scheme are confirmed using the ping test and frequency mode analysis for rotated square cavities. We then demonstrate the

diffraction Q value test and the use of this method for simulating an A6 magnetron. We obtained strong six resonance modes for the impulse response of A6 magnetron as expected. In the next chapter, we describe our scalable software solution MOLTN in detail.

CHAPTER 6: Software Solution and Code Acceleration

6.1 Introduction

In this chapter, we describe our new open-source code development work and give scalability studies for the OpenMP and GP-GPU CUDA versions.

The exciting EM software solutions in plasma science are developed based on the FDTD-PIC method [51; 67; 32; 70] and they have limitations as described previously in section 1.1. We have proved that our MOLT based scheme can overcome those limitations. Hence, we are motivated to develop a software solution to bring our scheme to the scientific community. We are developing an open source code **MOLTN** (Method Of Line Transpose based Nth arbitrary order scheme) which is intended to be an architecture-independent, scalable software tool, using MPI, OpenMP, and as well as GPU CUDA implementation. We are working with the templated C++ library Kokkos [24] at the lowest levels of the code to achieve this goal.

In this chapter, we give the design of MOLTN software and scalability studies for the OpenMP version in section 6.2 and the parallel design and scalability studies for the GP-GPU CUDA version in section 6.3.

6.2 Multi-core OpenMP, Multi-node MPI version

As an initial work, we developed the code in C++ using multi-core, shared memory OpenMP and multi-node, distributed memory MPI. For this implementation, we first decompose the domain as a set of subdomains and assign each sub-domain per node. We use OpenMP *pragmas* for the data parallelization within the nodes and each node communicates

with each other using the MPI library. In our scheme, internal sweeps are carried out on each node and boundary information is passed between the adjacent nodes during the left and right oriented sweeps for the fast convolution integration (as described in the section 2.2.2). We give an overview of the structure of the MOLTN in the following section,

6.2.1 Data Structures and Data Flows

As we have seen, we know the kernel of this scheme is the $\mathcal{O}(n)$ one-dimensional recursive fast convolution to obtain the particular solution and the homogeneous solution (section 2.2). Thus, we defined the MOLT KERNEL using a C++ object to implement this algorithm with subroutines to build quadrature weights, perform left and right oriented sweeps, and apply boundary conditions. We defined another major object, MOLT MESH, to represent the domain with every relevant parameter. The MOLT MESH acts as the main storage place of our implementation, which consists of a set of one-dimensional C++ pointer arrays. The controller object defined in MOLTN is the MOLT TIME-STEP which receives the user-defined object and parameters, builds MOLT MESH and performs needed computation and communication. We use node-core hybrid parallelism using MPI and OpenMP, thus initially, the whole domain has to be partitioned and distributed over the HPC cluster nodes. Then computing tasks have to be performed and the resultant relevant data must be transferred between the nodes. The MOLT TIME-STEP performs this task by invoking appropriate routines defined in the MOLTN library API that includes MOLT KERNEL, MOLT MESH, and MOLT TIME-STEP. So the application class can include these library files to carry out its work. The application of these libraries can be extended beyond the wave solver such as advection-diffusion, magnetohydrodynamic (MHD), and fluid dynamics.

6.2.2 Domain Decomposition

Here we present our domain decomposition strategy ([64]). Consider the $\mathcal{L}^{-1}(u)$ operator:

$$M(u) = \int_a^b u(y)e^{-\alpha|x-y|}dy + A_0e^{-\alpha(x-a)} + B_0e^{-\alpha(b-x)}.$$

We seek to partition this expression for subdomains, so we define c_i such that $c_0 < ... < c_i < ... < c_N$ for some N, where $c_0 = a$ and $c_N = b$, and thus for $c_i < x < c_{i+1}$,

$$M_i[u](x) = \int_{c_i}^{c_{i+1}} u(y)e^{-\alpha|x-y|}dy + A_ie^{-\alpha(x-c_i)} + B_ie^{-\alpha(c_{i+1}-x)}.$$

This expression can be rewritten in terms of the left and right sweep operators

$$\begin{aligned} M_i[u](x) =& e^{-\alpha(x-c_i)} \int_{c_i}^{x} u(y)e^{\alpha(y-c_i)}dy + e^{-\alpha(c_{i+1}-x)} \int_{x}^{c_{i+1}} u(y)e^{\alpha(c_{i+1}-y)}dy \\ &+ A_ie^{-\alpha(x-c_i)} + B_ie^{-\alpha(c_{i+1}-x)}. \end{aligned} \tag{6.1}$$

We evaluate this expression at $x = c_i$:

$$M_i[u](c_i) = e^{-\alpha(c_{i+1}-c_i)} \int_{c_i}^{c_{i+1}} u(y)e^{\alpha(c_{i+1}-y)}dy + A_i + B_ie^{-\alpha(c_{i+1}-c_i)}, \tag{6.2}$$

This is computed in the following way: we define

$$I^L[u](x) = \int_a^x u(y)e^{-\alpha|x-y|}dy = e^{-\alpha x} \int_a^x u(y)e^{\alpha y}dy$$

$$I^R[u](x) = \int_x^b u(y)e^{-\alpha|x-y|}dy = e^{\alpha x} \int_x^b u(y)e^{-\alpha y}dy$$

So then

$$M(u) = I^L[u](x) + I^R[u](x) + Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}$$

Suppose we decompose the domain $[a, b]$ to three sub domains 1,2, and 3 with pseudo boundaries c_0, c_1, c_2, and c_3 as shown in Figure 6.1, we can now apply the above expression for the each sub domain,

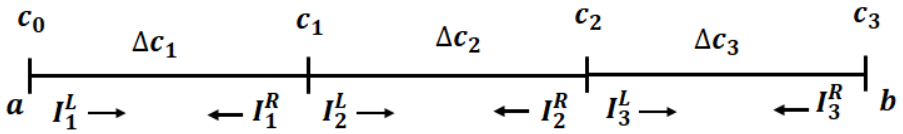


Figure 6.1: Domain Decomposition of the domain $[a, b]$ to three 1D sub domains $[c_0, c_1]$, $[c_1, c_2]$,and $[c_2, c_3]$, each evolves own internal left (I_j^L) and right (I_j^R) sweeps over the domain of length Δc_j.

$$\begin{aligned} A_1e^{-\alpha(x-a)} + B_1e^{-\alpha(c_1-x)} - A_0e^{-\alpha(x-a)} - B_0e^{-\alpha(b-x)} \\ = e^{-\alpha(c_1-x)}I_2^R[u](c_1) + e^{-\alpha(c_2-x)}I_3^R[u](c_2) \end{aligned}$$

$$\begin{aligned} A_2e^{-\alpha(x-c_1)} + B_2e^{-\alpha(c_2-x)} - A_0e^{-\alpha(x-a)} - B_0e^{-\alpha(b-x)} \\ = e^{-\alpha(x-c_1)}I_2^L[u](c_1) + e^{-\alpha(c_2-x)}I_3^R[u](c_2) \end{aligned}$$

$$\begin{aligned} A_3e^{-\alpha(x-c_2)} + B_3e^{-\alpha(b-x)} - A_0e^{-\alpha(x-a)} - B_0e^{-\alpha(b-x)} \\ = e^{-\alpha(x-c_1)}I_1^L[u](c_1) + e^{-\alpha(x-c_2)}I_2^L[u](c_2) \end{aligned}$$

We define $\Delta c_1 = a - c_1$, $\Delta c_2 = c_2 - c_1$, and $\Delta c_3 = b - c_2$. We now write:

$$A_1 e^{-\alpha(x-a)} + B_1 e^{-\alpha(c_1-x)} - A_0 e^{-\alpha(x-a)} - B_0 e^{-\alpha(c_1-x)} e^{-\alpha\Delta c_2} e^{-\alpha\Delta c_3}$$
$$= e^{-\alpha(c_1-x)} I_2^R[u](c_1) + e^{-\alpha(c_1-x)} e^{-\alpha\Delta c_2} I_3^R[u](c_2) \tag{6.3}$$

$$A_2 e^{-\alpha(x-c_1)} + B_2 e^{-\alpha(c_2-x)} - A_0 e^{-\alpha\Delta c_1} e^{-\alpha(x-c_1)} - B_0 e^{-\alpha\Delta c_3} e^{-\alpha(c_2-x)}$$
$$= e^{-\alpha(x-c_1)} I_2^L[u](c_1) + e^{-\alpha(c_2-x)} I_3^R[u](c_2) \tag{6.4}$$

$$A_3 e^{-\alpha(x-c_2)} + B_3 e^{-\alpha(b-x)} - A_0 e^{-\alpha\Delta c_1} e^{-\alpha\Delta c_2} e^{-\alpha(x-c_2)} - B_0 e^{-\alpha(b-x)}$$
$$= e^{-\alpha(x-c_2)} e^{-\alpha\Delta c_2} I_1^L[u](c_1) + e^{-\alpha(x-c_2)} I_2^L[u](c_2) \tag{6.5}$$

From the equations 6.3, 6.4, and 6.5 we can discern

$$A_1 = A_0$$
$$B_1 = B_0 e^{-\alpha\Delta c_2} e^{-\alpha\Delta c_3} + I_2^R[u](c_1) + e^{-\alpha\Delta c_2} I_3^R[u](c_2)$$
$$A_2 = A_0 e^{-\alpha\Delta c_1} + I_2^L[u](c_1)$$
$$B_2 = B_0 e^{-\alpha\Delta c_3} + I_3^R[u](c_2)$$
$$A_3 = A_0 e^{-\alpha\Delta c_1} e^{-\alpha\Delta c_2} + e^{-\alpha\Delta c_2} I_1^L[u](c_1) + I_2^L[u](c_2)$$
$$B_3 = B_0$$

We choose Δt such that $e^{-\alpha\Delta c_i} = 0$ for $i = 1...N$, hence,

$$A_1 = A_0, \quad B_1 = I_2^R[u](c_1),$$
$$A_2 = I_2^L[u](c_1), \quad B_2 = I_3^R[u](c_2),$$
$$A_3 = I_2^L[u](c_2), \quad B_3 = B_0.$$

6.2.3 Performance Analysis

As an initial evaluation of MOLTN, we tested it using a single node multi-core platform. We chose a node with 28 cores which facilitated with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor and 492 GB memory to examine our OpenMP parallel MOLTN code without using any accelerators. This strong scaling study shows nearly linear speedup (in Figure 6.3) and approximately 85% efficiency (in Figure 6.4). We observe less speedup/efficiency for a small amount of work due to the leading effect of communication cost including communication overhead. Speedup and efficiency reduce with increasing core counts due to memory bounds.Hence, it shows positive signs for the scalability of our scheme, but we need to prove the scalability of the hybrid version and also have to work with the performance enhancement using the accelerators such as NVIDIA GPU and Intel SIMD vectorization. We believe that designing an algorithm based on the architecture is the best practice to achieve high performance, and focus to work deeply with lower-level code optimization technologies, NVIDIA warp primitive-level programming and SIMD vectorization intrinsics.

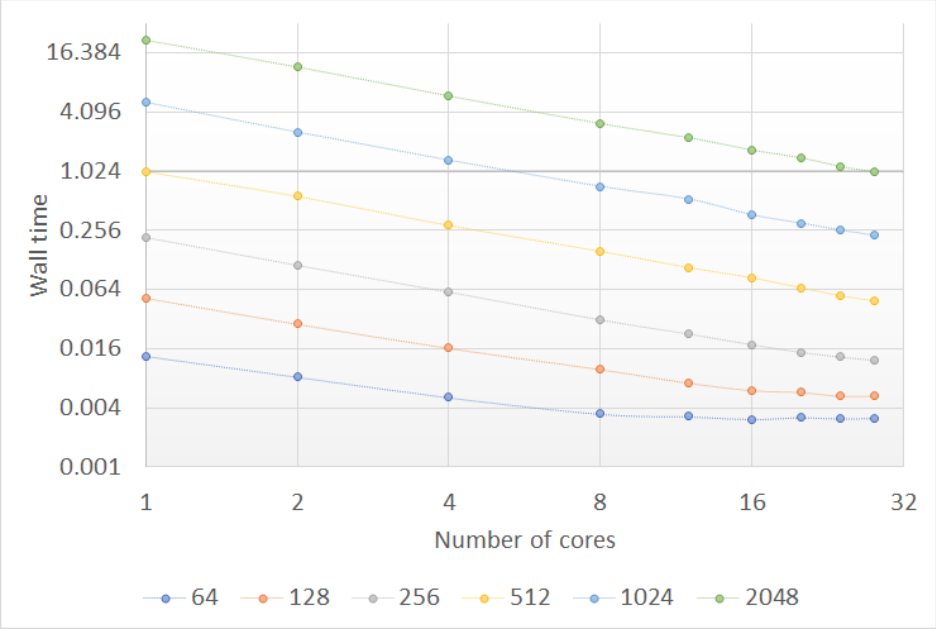


Figure 6.2: Log-log plot for wall time using 28 core Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor, for the grid size $N^2, N = 64, 128, 256, 1024, 2048$.

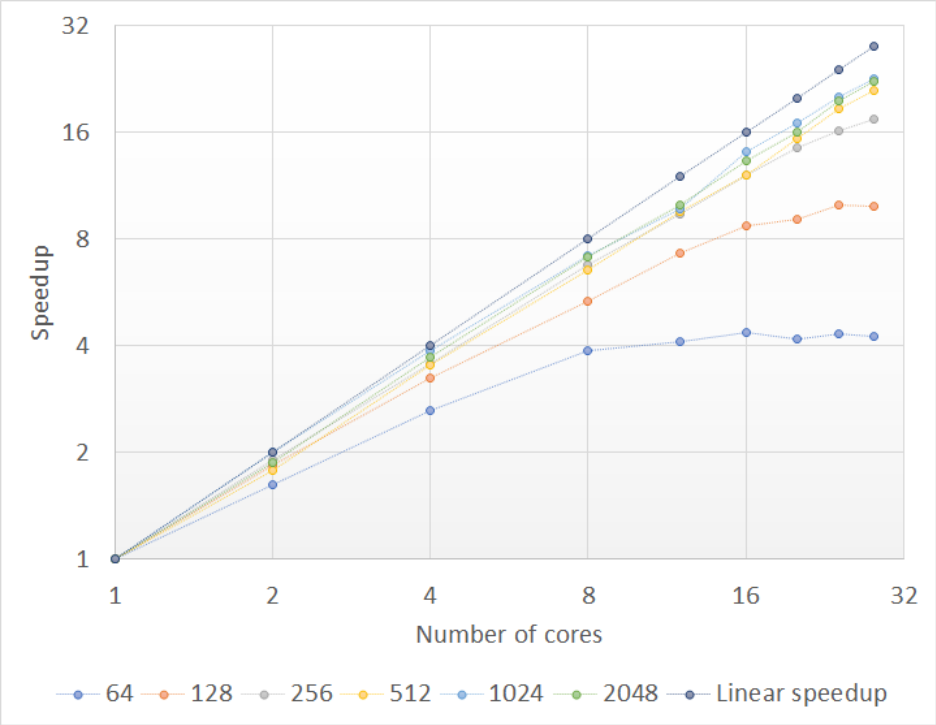


Figure 6.3: Log-log plot for speedup in using 28 core Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor for the grid size $N^2, N = 64, 128, 256, 512, 1024, 2048$.

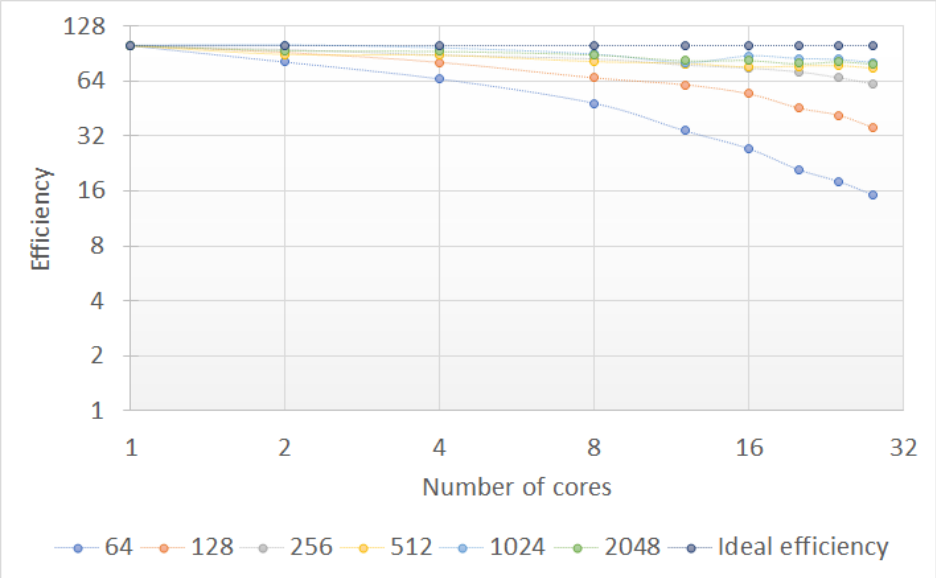


Figure 6.4: Log-log plot for efficiency using 28 core Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor for the grid size $N^2, N = 64, 128, 256, 512, 1024, 2048$.

6.3 GPGPU CUDA version

In this section, we describe our initial work of GPGPU CUDA implementation of our scheme and performance evaluation.

6.3.1 Task Organization

For this implementation, we chose the two-dimensional second-order scheme,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}_{xy}^{(1)}[u^n]$$

where,

$$\mathcal{C}_{xy} = (L_y^{-1} + L_x^{-1}) - (L_y^{-1}L_x^{-1} + L_x^{-1}L_y^{-1}) \tag{6.6}$$

Based on the structure of the scheme, the computation of the solution u^{n+1} at the time step t_{n+1} can be obtained through four parallel stages using task paralleling,

1. Perform the first set of x- and y- sweeps:

   Compute $\mathbf{w_x} = L_x^{-1}[u]$ and $\mathbf{w_y} = L_y^{-1}[u]$ concurrently,

2. Perform the second set of x- and y- sweeps:

   Compute $\mathbf{w_{yx}} = L_x^{-1}[w_x]$ and $\mathbf{w_{xy}} = L_y^{-1}[w_y]$ concurrently,

3. Obtain C_{xy}:

   Compute $C_{xy1} = \mathbf{w_x} - \mathbf{w_{yx}}$ and $C_{xy2} = \mathbf{w_y} - \mathbf{w_{xy}}$ concurrently,

4. Obtain u^{n+1}:

   Compute $u_1^{n+1} = 2u^n - \beta^2 C_{xy1}$ and $u_2^{n+1} = -u^{n-1} - \beta^2 C_{xy2}$ concurrently,

and finally obtain $u^{n+1} = u_1^{n+1} + u_2^{n+1}$. We choose two CUDA streams to go through these stages, Figure 6.5 shows the processing flow of the two streams.

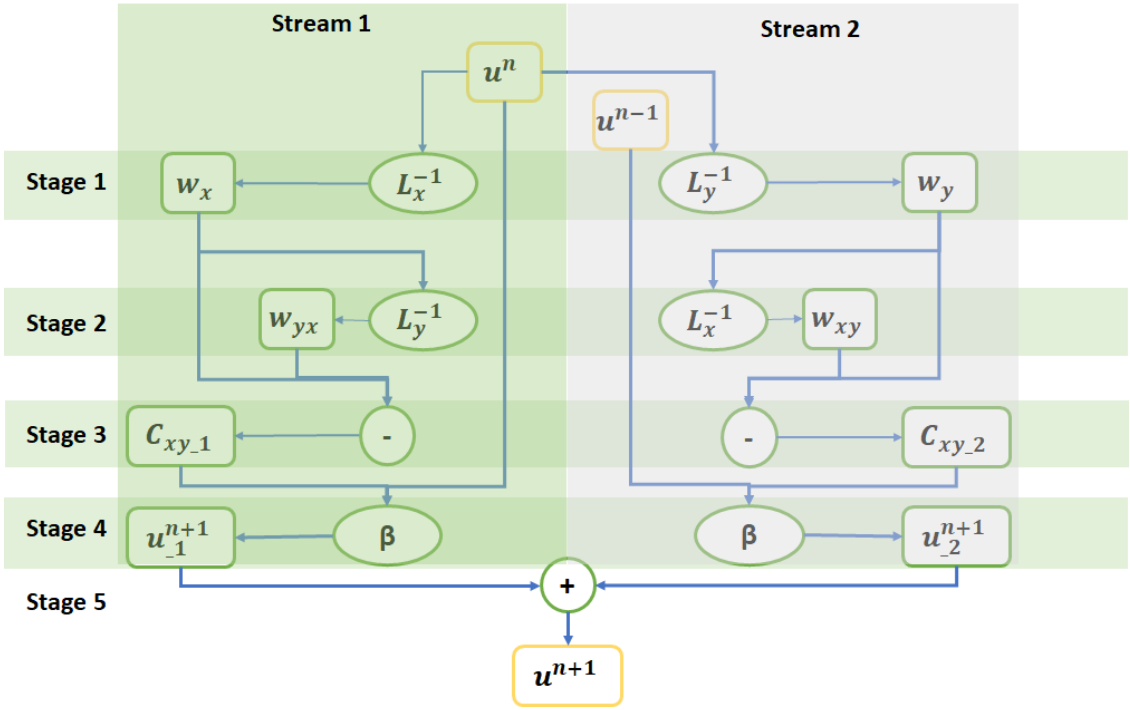


Figure 6.5: Task parallelism for the computation of u^{n+1} based on C_{xy} and parallel stages executing through CUDA streams.

6.3.2 Performance Analysis

We evaluated the scalability and performance of the scheme using NVIDIA Tesla K20 and K80 GPU accelerators. Tesla K20 (2880 CUDA cores) and K80 (4992 CUDA cores) GPUs were compared with a CPU, Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz processor. This study shows the nearly 45X and 35X speedup for Tesla K80 and K20 respectively with full utilized CUDA cores(in Figure 6.7) and linear time complexity for both CPU and GPU architectures (Figure 6.6). Even though the evaluation result shows reasonable improvement in time and cost, we have to include the MPI library to communicate between nodes for the distributed multi-node system and also consider the lower-level code optimization technologies, NVIDIA warp primitive-level programming.

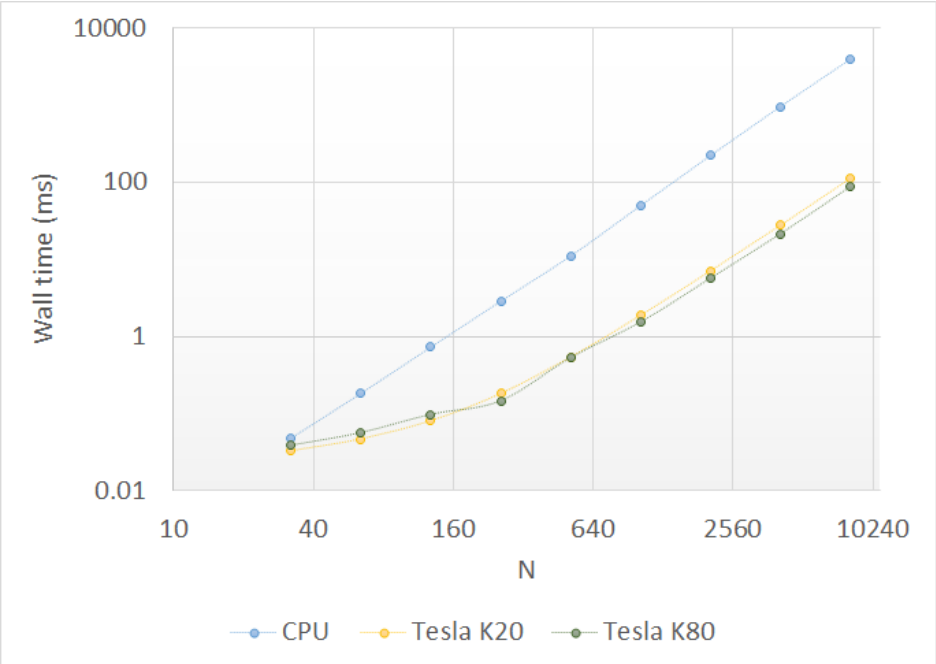


Figure 6.6: Log-log plot for wall time using Intel(R) Xeon(R) CPU, Tesla K20, and K80 GPUs for the grid size$N^2, N = 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192$.

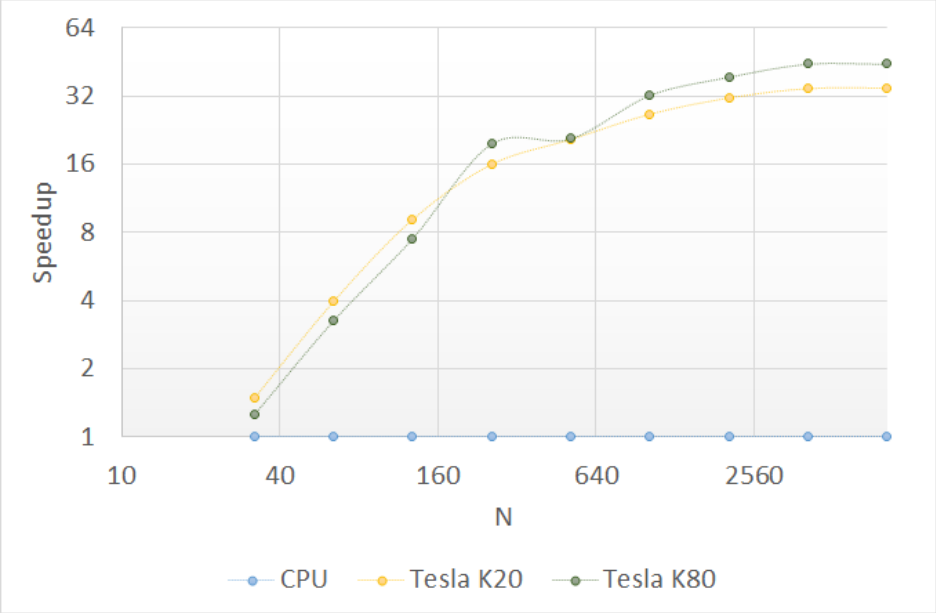


Figure 6.7: Intel(R) Xeon(R) CPU vs Tesla K20 and K80 GPU speedup (log-log plot) for the grid size $N^2, N = 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192$.

6.4 Summary

We have been developing an architecture-independent, scalable software tool **MOLTN** using MPI, OpenMP, and as well as GPU CUDA implementation. In this chapter, we have described the overview of the design of MOLTN, parallel implementation of GP-GPU CUDA version, and scalability studies for both OpenMP and CUDA versions. In the next chapter, we summarize the work carried out through these studies and give useful ideas for future work.

CHAPTER 7: Summary and Potential Future Directions

7.1 Introduction

In this chapter, I conclude the work carried out through this study, emphasize my contributions to this research work, and identify potential future directions.

7.2 Conclusion

Based on my literature review, I identified the MOLT based scheme developed by Christlieb team [15] as an unconditionally stable, fast, and implicit scheme and arbitrary order accuracy for Dirichlet and periodic boundaries on a Cartesian grid. Hence, I chose the scheme for my simulation study and overcame the identified limitations on it. The major limitation is that the scheme was second order for outflow and Neumann boundary conditions. So, I worked to extend the order of accuracy for Neumann and outflow for arbitrary order and achieved a scheme with fourth-order spatiotemporal accuracy for complex geometry problems including curved boundaries. Further, I defined a general framework for complex geometry problems and simulated such problems using an embedded boundary method. Specifically, I developed a simulation tool for HPM tubes, such as A6 magnetron, 12 and 18 cavity rising suns using PEC boundary condition. For this simulation, I derived the scheme for electromagnetic vector potential using the Lorenz gauge and imposed PEC boundary condition in 2D. I evaluated the simulation of A6 magnetron using a ping test and obtained six strong resonance modes. This is one of the challenging tasks that I accomplished.

I extended the implementation of the scheme to 3D and obtained multiple applications such as 3D variable speed waves, spherical scattering, 3D point source, and A6 magnetron in

3D. The simulation of three-dimensional A6 magnetron is an additional important milestone.

I developed codes in C++ for these simulations and improved those performances by code optimization and acceleration using parallel technologies in different architectures including NVIDIA GPU. Further, I have been working with the scalable open source MOLTN development and contributed to the development of multi-core shared memory OpenMP version and wave kernel development using kokkos.

7.3 Future Directions

7.3.1 Code Acceleration for MOLTN

As described in the Chapter 6, we are developing an open source scalable software to make visible our MOLT based scheme to the scientific community. We are on the initial stage of this development and will have to optimize the code in different levels; i) low-level architecture based ii) compiler related iii) high-level code based optimizations. To obtain accelerated code in Intel architectures, we can use the SIMD vectorization using Intel intrinsic instructions, which are C style functions that provide access to many Intel instructions, including Intel SSE (128 bits Streaming SIMD Extension), AVX (256 bits Advanced Vector Extension), and AVX512 (512 bits Advanced Vector Extension) etc. without the need to write assembly code. The SIMD vectorization techniques can increase processor throughput by performing multiple computations in a single instruction. We can use Arithmetic and Elementary Math functions such as the function $\_\_m256d \quad \_mm256\_add\_pd(\_\_m256d, \quad \_\_m256d)$ to add packed double-precision (64-bit) floating-point elements using AVX. We can also use the compiler flags for the auto-vectorization (e.g. $-O2$ or higher in intel). Another well-known accelerator is NVIDIA GPUs, which execute warps of 32 parallel threads using SIMT (Single Instruction Multiple Threads - multiple threads issue common instructions to arbitrary data) technologies. The explicit warp-level programming using warp-level primitives which were introduced in CUDA 9.0, is safe, effective and will give high-performance. We can use

NVIDIA's intrinsic functions to make the code even faster with reduced accuracy, but they can be used in device codes only.

7.3.2 Complex Geometry in MOLTN

We have already implemented simple rectangular-shaped boundary problems on the Cartesian grids for MOLTN, so we have to define a general framework for complex geometry problems on MOLTN in the future. To obtain this, i) convert MOLT KERNEL to able to operate on line-segments instead of lines because line-segments are unit elements for complex geometry cases. ii) we have to deal with the work load-balancing and data locality carefully in order to make sure better performance is obtained. I can suggest a max-fit policy which takes care of load balancing and locality at the same time.

7.3.3 Further HPM tube simulations

Even though the concept of the magnetrons is the same, every magnetron is not designed by the same structure. They may differ in the number of cavities (12 or 18 cavity rising suns), the shape of the cavities (triangular or circular cavity magnetrons), and radius of the cathode/anode, etc. In these studies, my simulations are developed based on triangular cavities. In the future, we can consider the circular cavity-anode structure as shown in Figure 7.3.3 which is a typical magnetron for microwave ovens in cutaway view. The cylindrical anode structure contains a number of equally spaced cavity resonators with the anode surface adjacent to the cylindrical cathode. Permanent magnets are used to provide the necessary magnetic field, which is perpendicular to the electric field between the cathode and the anode. The power output is coupled through an antenna that runs from one of the cavities to a waveguide that channels the microwave radiation to the cooking chamber.

Further, beyond the magnetrons, we can develop simulations for other kinds of HPM tubes such as the accelerator klystron. Figure 7.3.3 describes the two-cavity klystron [30]

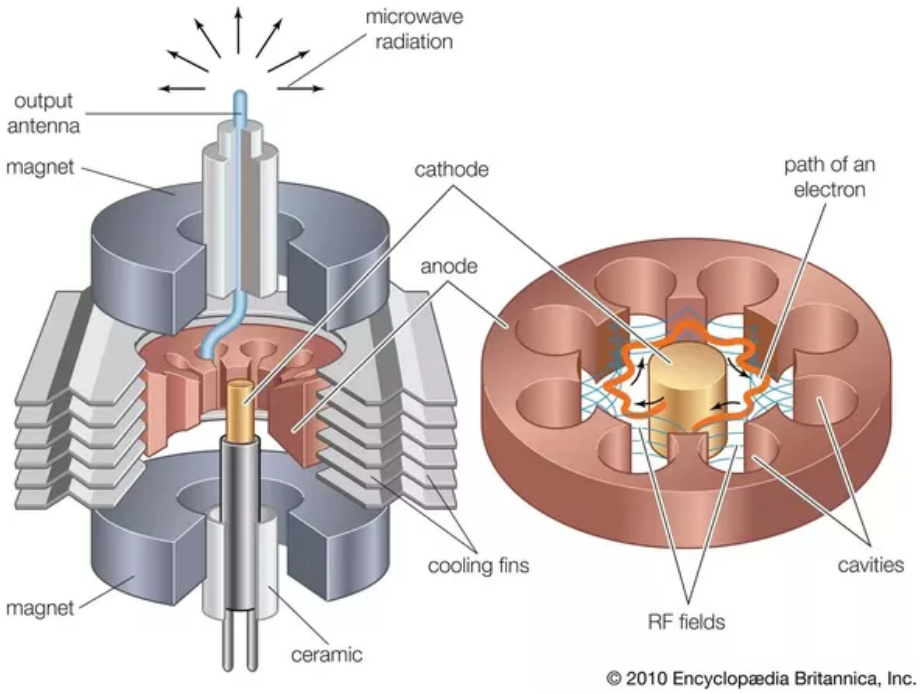


Figure 7.1: Magnetron with eight circular hole cavity anode (this figure is from Encyclopedia Britannica, Inc.)

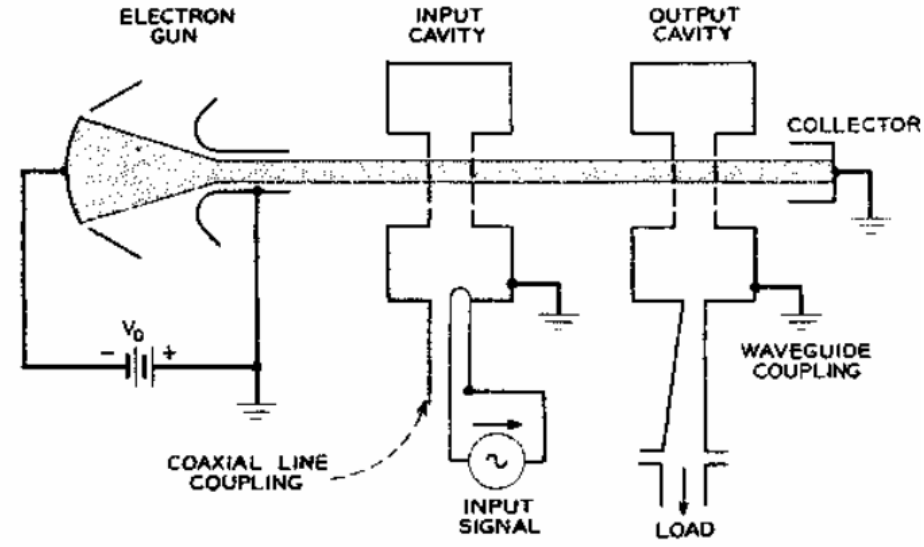


Figure 7.2: Two-cavity klystron (this figure is from Reference [30])

7.3.4 Incorporate with Particles

We obtained a cold test (in Chapter 5) using the electromagnetic waves without including particles successfully. Now, we are planning to perform hot tests by including particles in

the simulation. For this purpose, we tested for a point source as a delta function in three dimensions. Next we will see how to derive the scheme which includes the point source.

7.3.4.1 Derive the scheme for 3D point source

Let's rewrite the Helmholtz operator in 1D,

$$\mathcal{L}_x(.) = \Big(1 - \frac{1}{\alpha^2}\partial_{xx}\Big)(.)$$

We use the free space Green's function to solve the L^{-1} as previously described. The one dimensional Green's function can be given by,

$$G(x|x_0) = \frac{1}{4\pi}\frac{e^{-\alpha|x-x_0|_2}}{|x-x_0|_2} \tag{7.1}$$

Upon using the Green's function,

$$\mathcal{L}_x^{-1}(.) = \int_{\Omega_{x_0}} G(x|x_0)(.)dx_0 + \int_{\Omega_{x_0}} G(x|x_0)(.)dx_0 \tag{7.2}$$

Now, we break the solution into two pieces, the particular solution and the homogeneous solutions:

$$\mathcal{L}_x^{-1}(.) := \mathcal{I}_x(.) + \mathcal{H}_x \tag{7.3}$$

where,

$$\mathcal{I}_x(.) = \frac{\alpha}{2}\int_a^b e^{-\alpha|x-y|}(.)dy, \quad \mathcal{H}_x = a_x e^{-\alpha(x-x_a)} + b_x e^{-\alpha(x_b-x)}.$$

For three dimensional inverse operator $\mathcal{L}^{-1}(.)$ for the ADI splitting is performed one line

at a time. Recall,

$$\mathcal{L}_k^{-1}(.) := \mathcal{I}_k(.) + \mathcal{H}_k \tag{7.4}$$

where,

$$\mathcal{I}_k(.) = \frac{\alpha}{2}\int_a^b e^{-\alpha|k-k_0|}(.)dk_0, \quad \mathcal{H}_k = a_k e^{-\alpha(k-k_a)} + b_k e^{-\alpha(k_b-k)}.$$

So the three dimensional operator $\mathcal{L}^{-1}(.)$ can be expressed as,

$$\mathcal{L}^{-1}(.) = \mathcal{L}_x^{-1}(\mathcal{L}_y^{-1}(\mathcal{L}_z^{-1}(.))) + O(\Delta t^2)$$

or

$$\mathcal{L}^{-1}(.) = \mathcal{I}_{x,y,z}(.) + \mathcal{H}_{x,y,z}(.) + O(\Delta t^2)$$

or

$$\mathcal{L}^{-1}(.) = \mathcal{I}_x(\mathcal{I}_y(\mathcal{I}_z(.) + \mathcal{H}_z) + \mathcal{H}_y) + \mathcal{H}_x + O(\Delta t^2)$$

By plugging into the second order scheme,

$$u^{n+1} = (2-\beta^2)u^n - u^{n-1} + \beta^2 \bar{\mathcal{L}}_{3D}^{-1}[u^n] \tag{7.5}$$

$$u^{n+1} = (2-\beta^2)u^n - u^{n-1} + \beta^2 \mathcal{L}_x^{-1}\left[\mathcal{L}_y^{-1}\left[\mathcal{L}_z^{-1}[u^n]\right]\right]. \tag{7.6}$$

and including a point source S,

$$u^{n+1} = (2-\beta^2)u^n - u^{n-1} + \beta^2 \mathcal{L}_x^{-1}\left[\mathcal{L}_y^{-1}\left[\mathcal{L}_z^{-1}[u^n]\right]\right] + \frac{\beta^2}{\alpha^2} S^n . G_{3D}^{\delta}. \tag{7.7}$$

where,

$$G_{3D}^{\delta} = \frac{1}{4\pi}\frac{e^{-\alpha\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}}}{\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2+\delta^2}}$$

$$G^{\delta}_{3D}(x/x_0) = \frac{1}{4\pi} \frac{e^{-\alpha||x-x_0||_2}}{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 + \delta^2}}$$

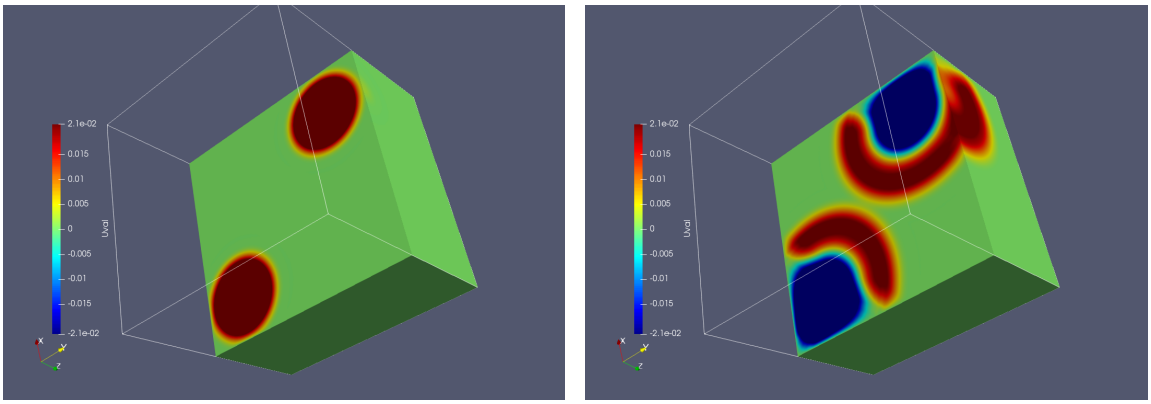
where $\delta \in \mathcal{R}^+$, $0 < \delta^2 < \Delta x$

7.3.4.2 Numerical Example

For the next two experiments we placed two point sources $sin(2\pi ft)$ with equal frequencies ($f = 1$) for the first case and two point sources $sin(2\pi ft)$ with different frequencies $f = 1$ and $f = 10$ for the second case at the corners $(-0.613, -0.613, -0.613)$ and $(1.613, 1.613, 1.613)$ of the cubical domain $([-1, 1]^3)$. We simulate the wave propagation by applying outflow and Dirichlet boundary conditions along the right surface of the cube and the other five surfaces of the cube respectively. We set the values of parameters $\Delta x = \Delta y = \Delta z = 0.013$, $\Delta t = 0.0067$, and $\beta = 2$. Figure 7.3 and 7.5 show snapshots of the wave at different time instants for the symmetric and asymmetric cases.

Finally, we perform a self-refinement study for time and space convergence using a point source $sin(2\pi ft)$ with frequencies $f = 1$ at the center of the cubical domain $([-1, 1]^3)$. Figure 7.4 shows the time and space convergence plots for the time evolution of a point source field. For the time convergence test, we maintain the resolution at $40 \times 40 \times 40$ and reduce the time step size $\Delta t = 0.1 \quad to \quad 0.0125$. For the space convergence test, we maintain the CFL value as 1.3 and reduce the spatial step size $\Delta x = 1.313$ to 0.082. We obtained second-order convergence in time and space

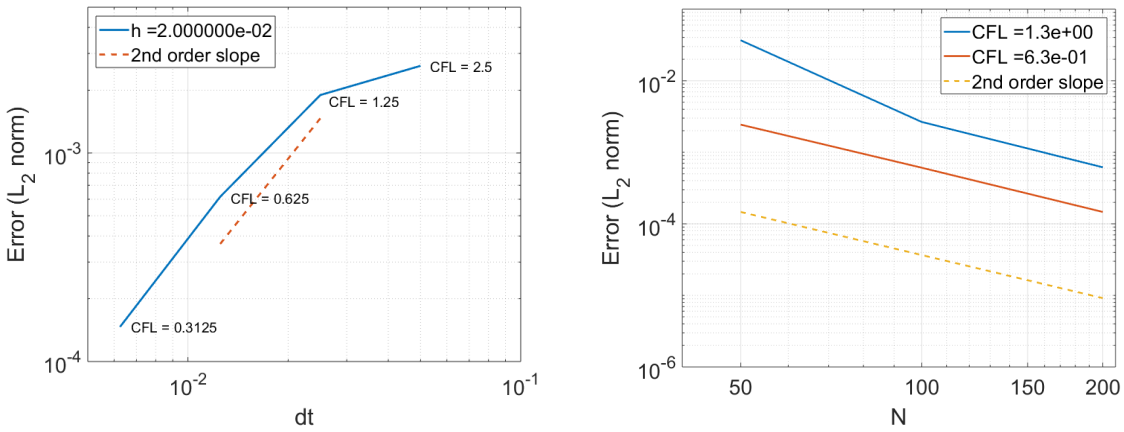
This shows a positive result to deal with particles properly, and the approach can be extended to high-order by using the Taylor series expansion in terms of the source function.



(a) $t = 1.23ns$ (b) $t = 3.01ns$

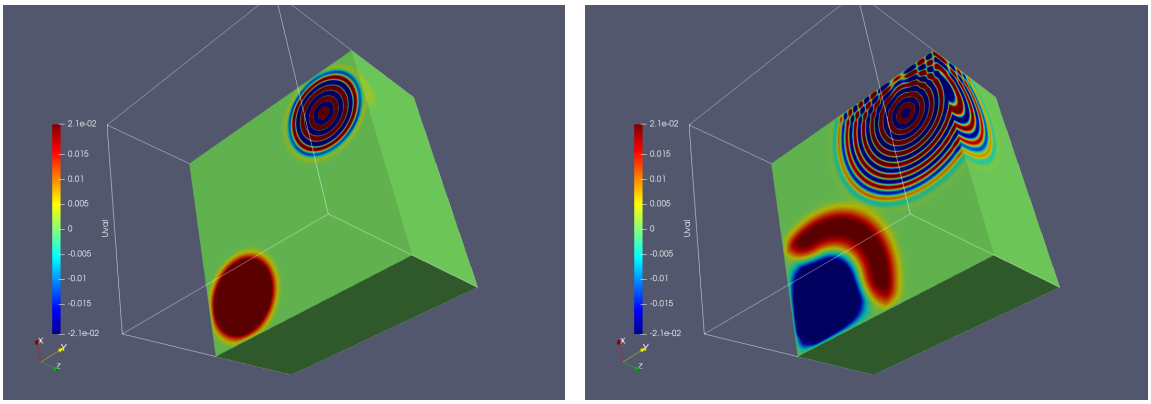
(c) $t = 5.68ns$ (d) $t = 8.81ns$

Figure 7.3: Evolution of the two 3D point sources $sin(2\pi ft)$ with the same frequency $f = 1$ at the corners of the cubical domain $([-1, 1]^3)$, spatial step size $\Delta x = \Delta y = \Delta z = 0.013$, and time step size $\Delta t = 0.0067$.



(a) time convergent (b) space convergent

Figure 7.4: Time (a) and space (b) convergence studies using 3D point source $sin(2\pi ft)$ with frequencies $f = 1$ at the center of the cubical domain $([-1, 1]^3)$.



(a) $t = 1.23ns$ (b) $t = 3.01ns$

(c) $t = 5.68ns$ (d) $t = 8.81ns$

Figure 7.5: Evolution of the two 3D point sources $sin(2\pi ft)$ with frequencies 1 and 10 at the corners of the cubical domain $([-1, 1]^3)$, spatial step size $\Delta x = \Delta y = \Delta z = 0.013$, and time step size $\Delta t = 0.0067$.

7.4 Summary

I gave the summary and contributions of this research work and gave potential future directions in this chapter. I strengthen the MOLT based scheme by including high-order Neumann, outflow and PEC boundary conditions, general geometry framework, 3D implementations, and accelerated codes and wave kernel for MOLTN development. We are proposing to work with particles, several HPM tube simulations, and optimization and development of MOLTN in the future.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Alpert, B., L. Greengard, and T. Hagstrom (2000), Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation, *SIAM Journal on Numerical Analysis*, *37*(4), 1138–1164, doi:10.1137/S0036142998336916.

[2] AlSalem, H., P. Petrov, G. Newman, and J. Rector (2018), Embedded boundary methods for modeling 3d finite-difference laplace-fourier domain acoustic-wave equation with free-surface topography, *GEOPHYSICS*, *83*(5), T291–T300, doi:10.1190/geo2017-0828.1.

[3] Appelo, D., and N. A. Petersson (2012), A Fourth-Order Accurate Embedded Boundary Method for the Wave Equation, *SIAM J. Sci. Comput.*, *34*(6), A2982–A3008.

[4] Ascher, U., R. Mattheij, and R. Russell (1995), *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Society for Industrial and Applied Mathematics, doi:10.1137/1.9781611971231.

[5] Benford, J. (2010), History and future of the relativistic magnetron, in *2010 International Conference on the Origins and Evolution of the Cavity Magnetron*, pp. 40–45, doi:10.1109/CAVMAG.2010.5565566.

[6] Berenger, J.-P. (1994), A perfectly matched layer for the absorption of electromagnetic waves, *Journal of Computational Physics*, *114*(2), 185 – 200, doi: https://doi.org/10.1006/jcph.1994.1159.

[7] Bernacki, M., S. Lanteri, and S. Piperno (2006), Time-Domain Parallel Simulation of Heterogeneous Wave Propagation on Unstructured Grids Using Explicit, Nondiffusive, Discontinuous Galerkin Methods, *Journal of Computational Acoustics*, *14*(01), 57–81, doi:10.1142/S0218396X06002937.

[8] Bruno, O. P., and M. Lyon (2010), High-order unconditionally stable FC-AD solvers for general smooth domains I . Basic elements, *Journal of Computational Physics*, *229*(6), 2009–2033.

[9] Bruno, O. P., and M. Lyon (2010), High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic elements , *Journal of Computational Physics*, *229*(6), 2009 – 2033, doi:http://dx.doi.org/10.1016/j.jcp.2009.11.020.

[10] Cai, W. (2004), Numerical methods for maxwell's equations in inhomogeneous media with material interfaces, *Journal of Computational Mathematics*, *22*(2), 156–167.

[11] Cai, W., and S. Deng (2003), An upwinding embedded boundary method for Maxwells equations in media with material interfaces: 2D case, *Journal of Computational Physics*, *190*(1), 159 – 183, doi:http://dx.doi.org/10.1016/S0021-9991(03)00269-9.

[12] Catella, A., V. Dolean, and S. Lanteri (2008), An Unconditionally Stable Discontinuous Galerkin Method for Solving the 2-D Time-Domain Maxwell Equations on Unstructured Triangular Meshes, *IEEE Transactions on Magnetics*, *44*(6), 1250–1253, doi:10.1109/TMAG.2007.916578.

[13] Causley, M., A. Christlieb, and E. Wolf (2017), Method of Lines Transpose: An Efficient Unconditionally Stable Solver for Wave Propagation, *Journal of Scientific Computing*, *70*(2), 896–921, doi:10.1007/s10915-016-0268-8.

[14] Causley, M. F., and A. J. Christlieb (2014), Higher Order A-Stable Schemes for the Wave Equation Using a Successive Convolution Approach, *SIAM Journal on Numerical Analysis*, *52*(1), 220–235, doi:10.1137/130932685.

[15] Causley, M. F., A. Christlieb, Y. Güçlü, and E. Wolf (2013), Method of Lines Transpose: A Fast Implicit Wave Propagator, *Mathematics of Computation*, submitted.

[16] Chini, G. P., and S. Leibovich (2003), An analysis of the klemp and durran radiation boundary condition as applied to dissipative internal waves, *Journal of Physical Oceanography*, *33*(11), 2394–2407, doi:10.1175/1520-0485(2003)033¡2394:AAOTKA¿2.0.CO;2.

[17] Coifman, R., V. Rokhlin, and S. Wandzura (1993), The fast multipole method for the wave equation: a pedestrian prescription, *IEEE Antennas and Propagation Magazine*, *35*(3), 7–12, doi:10.1109/74.250128.

[18] Deng, S., and W. Cai (2006), A Fourth-Order Upwinding Embedded Boundary Method (UEBM) for Maxwells Equations in Media with Material Interfaces: Part I.

[19] Ditkowski, A., K. Dridi, and J. Hesthaven (2001), Convergent Cartesian Grid Methods for Maxwell's Equations in Complex Geometries, *Journal of Computational Physics*, *170*(1), 39 – 80, doi:http://dx.doi.org/10.1006/jcph.2001.6719.

[20] Dolean, V., H. Fahs, L. Fezoui, and S. Lanteri (2010), Locally implicit discontinuous Galerkin method for time domain electromagnetics, *Journal of Computational Physics*, *229*(2), 512 – 526, doi:http://dx.doi.org/10.1016/j.jcp.2009.09.038.

[21] Dridi, K. H., J. S. Hesthaven, and A. Ditkowski (2001), Staircase-free finite-difference time-domain formulation for general materials in complex geometries, *IEEE Transactions on Antennas and Propagation*, *49*(5), 749–756, doi:10.1109/8.929629.

[22] Duru, K., and G. Kreiss (2012), A Well-Posed and Discretely Stable Perfectly Matched Layer for Elastic Wave Equations in Second Order Formulation, *Communications in Computational Physics*, pp. 1–35.

[23] Dwivedi, S., and P. K. Jain (2013), Magnetically Insulated Line Oscillator (MILO) Performance Study and its Parameter Optimization, *IEEE Transactions on Plasma Science*, *41*(9), 2532–2538, doi:10.1109/TPS.2013.2277859.

[24] Edwards, H. C., C. R. Trott, and D. Sunderland (2014), Kokkos: Enabling manycore performance portability through polymorphic memory access patterns, *Journal of Parallel and Distributed Computing*, *74*(12), 3202 – 3216, doi: https://doi.org/10.1016/j.jpdc.2014.07.003, domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[25] Engquist, B., and A. Majda (1977), Absorbing boundary conditions for the numerical simulation of waves, *Mathematics of Computation*, *31*(139), 629–651, doi: 10.1090/S0025-5718-1977-0436612-4.

[26] Fornberg, B. (2001), A short proof of the unconditional stability of the ADI-FDTD scheme, *Tech. rep.*, University of Colorado.

[27] Fornberg, B., J. Zuev, and J. Lee (2007), Stability and accuracy of time-extrapolated ADI-FDTD methods for solving wave equations, *Journal of Computational and Applied Mathematics*, *200*(1), 178 – 192.

[28] Fuks, M., and E. Schamiloglu (2005), Rapid start of oscillations in a magnetron with a "transparent" cathode, *Phys. Rev. Lett.*, *95*, 205,101, doi: 10.1103/PhysRevLett.95.205101.

[29] Gedney, S. D., and U. Navsariwala (1995), An unconditionally stable finite element time-domain solution of the vector wave equation, *IEEE Microwave and Guided Wave Letters*, *5*(10), 332–334, doi:10.1109/75.465046.

[30] Gewartowski, H. A. W., J. W. (1965), Principles of electron tubes, *D. Van Nostrand Company, Inc.*

[31] Gimbutas, Z., and V. Rokhlin (2003), A Generalized Fast Multipole Method for Nonoscillatory Kernels, *SIAM Journal on Scientific Computing*, *24*(3), 796–817, doi: 10.1137/S1064827500381148.

[32] Goplen, B., L. Ludeking, D. Smith, and G. Warren (1995), User-configurable MAGIC for electromagnetic PIC calculations, *Computer Physics Communications*, *87*(1), 54 – 86, doi:https://doi.org/10.1016/0010-4655(95)00010-D, particle Simulation Methods.

[33] Harlow, J. (2003), *Electric Power Transformer Engineering*, The Electric Power Engineering Hbk, Second Edition, CRC Press.

[34] Henshaw, W. D. (2006), A High-Order Accurate Parallel Solver for Maxwell's Equations on Overlapping Grids, *SIAM J. Scientific Computing*, *28*, 1730–1765.

[35] Higdon, R. L. (1987), Numerical absorbing boundary conditions for the wave equation, *Mathematics of Computation*, *49*(179), 65–90.

[36] Huang, Y. J., L. H. Yeh, and K. R. Chu (2014), An analytical study on the diffraction quality factor of open cavities, *Physics of Plasmas*, *21*(10), 103,112, doi: 10.1063/1.4900415.

[37] Jackson, J. D. (1999), *Classical Electrodynamics*, 3rd ed. ed., Wiley, New York, NY.

[38] Jackson, S. D., and P. H. Muir (2001), Theory and numerical simulation of nth-order cascaded Raman fiber lasers, *J. Opt. Soc. Am. B*, *18*(9), 1297–1306, doi: 10.1364/JOSAB.18.001297.

[39] Jacobs, G., and J. Hesthaven (2009), Implicit explicit time integration of a high-order particle-in-cell method with hyperbolic divergence cleaning, *Computer Physics Communications*, *180*(10), 1760 – 1767, doi:https://doi.org/10.1016/j.cpc.2009.05.020.

[40] Jacobs, G. B., J. S. Hesthaven, and G. Lapenta (2006), Simulations of the Weibel instability with a high-order discontinuous galerkin particle-in-cell solver, *Collection of Technical Papers - 44th AIAA Aerospace Sciences Meeting*, *19*, 14,189–14,199.

[41] Jia, J., and J. Huang (2008), Krylov deferred correction accelerated method of lines transpose for parabolic problems, *Journal of Computational Physics*, *227*(3), 1739–1753.

[42] Jin, J. (2010), *Theory and Computation of Electromagnetic Fields*, John Wiley & Sons, Inc, doi:10.1002/9780470874257.

[43] Joannopoulos, J., S. Johnson, J. Winn, and R. Meade (2011), *Photonic Crystals: Molding the Flow of Light (Second Edition)*, Princeton University Press.

[44] Johansen, H., and P. Colella (1998), A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains, *J. Computational. Phys.*, *147*(1), 60–85, doi:10.1006/jcph.1998.5965.

[45] Kreiss, H., and N. Petersson (2006), An embedded boundary method for the wave equation with discontinuous coefficients, *SIAM Journal on Scientific Computing*, *28*(6), 2054–2074, doi:10.1137/050641399.

[46] Kreiss, H., and N. A. Petersson (2006), An Embedded Boundary Method for the Wave Equation with Discontinuous Coefficients, *SIAM Journal on Scientific Computing*, *28*(6), 2054–2074, doi:10.1137/050641399.

[47] Lambers, J. V. (2016), Solution of time-dependent PDE through component-wise approximation of matrix functions, *IAENG Journal of Applied Mathematics*, *41*(1), 1–26.

[48] Lapidus, L., and G. F. Pinder (1999), *Numerical Solution of Partial Differential Equations in Science and Engineering*, 677 pp., John Wiley and Sons Inc.

[49] LeVeque, R. (2007), *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Society for Industrial and Applied Mathematics.

[50] Li, P., H. Johnston, and R. Krasny (2009), A Cartesian treecode for screened Coulomb interactions , *Journal of Computational Physics*, *228*(10), 3858 – 3868, doi: http://dx.doi.org/10.1016/j.jcp.2009.02.022.

[51] M. C. Lin, C. Nieter, P. H. Stoltz, D. N. Smithe (2010), Accurately and Efficiently Studying the RF Structures Using a Conformal Finite-Difference Time-Domain Particle-in-Cell Method, *The Open Plasma Physics Journal*, *3*, 48–52, doi: 10.2174/1876534301003010048.

[52] Marfurt, K. J. (1984), Accuracy of finitedifference and finiteelement modeling of the scalar and elastic wave equations, *GEOPHYSICS*, *49*(5), 533–549, doi: 10.1190/1.1441689.

[53] Matthew Causley, B. O., Andrew Christlieb, and L. V. Groningen (2014), Method of lines transpose: An implicit solution to the wave equation, *Mathematics of Computation*, *83*, 2763–2786, doi:https://doi.org/10.1090/S0025-5718-2014-02834-2.

[54] Mazzia, A., and F. Mazzia (1997), High-order transverse schemes for the numerical solution of PDEs, *Journal of Computational and Applied Mathematics*, *82*(1), 299 – 311, doi:http://dx.doi.org/10.1016/S0377-0427(97)00090-3.

[55] McCorquodale, P., P. Colella, and H. Johansen (2001), A Cartesian grid embedded boundary method for the heat equation on irregular domains.

[56] Mendonca, C., S. Prasad, E. Schamiloglu, and T. Fleming (2011), 3d icepic simulations of a pulsed relativistic magnetron with transparent cathode: A comparative study, in *2011 IEEE Pulsed Power Conference*, pp. 823–828, doi:10.1109/PPC.2011.6191521.

[57] Monk, P., and E. Suli (1994), Error estimates for Yee's method on non-uniform grids, *IEEE Transactions on Magnetics*, *30*(5), 3200–3203, doi:10.1109/20.312618.

[58] Namiki, T. (1999), A new FDTD algorithm based on alternating-direction implicit method, *IEEE Transactions on Microwave Theory and Techniques*, *47*(10), 2003–2007, doi:10.1109/22.795075.

[59] Newmark, N. (1959), *A Method of Computation for Structural Dynamics*, no. nos. 179-181 in A Method of Computation for Structural Dynamics, American Society of Civil Engineers.

[60] Orlanski, I. (1976), A simple boundary condition for unbounded hyperbolic flows, *Journal of Computational Physics*, *21*(3), 251 – 269, doi:https://doi.org/10.1016/0021-9991(76)90023-1.

[61] Palevsky, A. (1980), *Generation of Intense Microwave Radiation by the Relativistic E-beam Magnetron (experiment and Numerical Simulation)*, Massachusetts Institute of Technology, Department of Physics.

[62] Palevsky, A., and G. Bekefi (1979), Microwave emission from pulsed, relativistic ebeam diodes. II. The multiresonator magnetron, *The Physics of Fluids*, *22*(5), 986–996, doi: 10.1063/1.862663.

[63] Pearson, R. A. (1974), Consistent boundary conditions for numerical models of systems that admit dispersive waves, *Journal of the Atmospheric Sciences*, *31*(6), 1481–1489, doi:10.1175/1520-0469(1974)031¡1481:CBCFNM¿2.0.CO;2.

[64] Pierson Guthrey, W. S. A. C., Thavappiragasam Mathialakan (2019), Domain Decomposition Strategy for Method of Line Transpose Based Scheme.

[65] Ricci, P., G. Lapenta, and J. Brackbill (2002), A Simplified Implicit Maxwell Solver, *Journal of Computational Physics*, *183*(1), 117 – 141, doi: http://dx.doi.org/10.1006/jcph.2002.7170.

[66] Sommerfeld, A. (1949), p. iv, doi:https://doi.org/10.1016/B978-0-12-654658-3.50002-1.

[67] T. Austin, D. N. S., J. R. Cary, and C. Nieter (2010), Alternating Direction Implicit Methods for FDTD Using the Dey-Mittra Embedded Boundary Method, *The Open Plasma Physics Journal*, *3*, 29–35, doi:10.2174/1876534301003010029.

[68] Thavappiragasam, M., A. Viswanathan, and A. Christlieb (2017), MOLT based fast high-order three dimensional A-stable scheme for wave propagation, *Journal of Coupled Systems and Multiscale Dynamics*, *5*(2), 151–163, doi:doi:10.1166/jcsmd.2017.1137.

[69] Ting, L., and M. J. Miksis (1986), Exact boundary conditions for scattering problems, *The Journal of the Acoustical Society of America*, *80*(6), 1825–1827, doi: 10.1121/1.394297.

[70] Wang, Y., and S. Langdon (2010), Design of wave ports in fdtd and its application to microwave circuits and antennas, in *2010 IEEE Antennas and Propagation Society International Symposium*, pp. 1–4, doi:10.1109/APS.2010.5562023.

[71] Yee, K. (1966), Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Transactions on Antennas and Propagation*, *14*(3), 302–307, doi:10.1109/TAP.1966.1138693.

[72] Zafarullah, A. (1970), Application of the Method of Lines to Parabolic Partial Differential Equations With Error Estimates, *J. ACM*, *17*(2), 294–302, doi: 10.1145/321574.321583.

[73] Zheng, F., Z. Chen, and J. Zhang (1999), A finite-difference time-domain method without the Courant stability conditions, *IEEE Microwave and Guided Wave Letters*, *9*(11), 441–443, doi:10.1109/75.808026.

[74] Zienkiewicz, O., and R. Newton (1969), *Coupled Vibrations of a Structure Submerged in a Compressible Fluid*.