

Louisiana Tech University

Louisiana Tech Digital Commons

Doctoral Dissertations

Graduate School

Spring 5-2020

**ABC Method and Fractional Momentum Layer for the FDTD
Method to Solve the Schrödinger Equation on Unbounded
Domains**

Joshua Paul Wilson

Follow this and additional works at: <https://digitalcommons.latech.edu/dissertations>

**ABC METHOD AND FRACTIONAL MOMENTUM LAYER FOR THE FDTD
METHOD TO SOLVE THE SCHRÖDINGER EQUATION
ON UNBOUNDED DOMAINS**

by

Joshua Wilson, B.S., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2020

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

March 17, 2020

Date of dissertation defense

We hereby recommend that the dissertation prepared by

Joshua Paul Wilson, B.S., M.S.

entitled **ABC METHOD AND FRACTIONAL MOMENTUM LAYER FOR
THE FDTD METHOD TO SOLVE THE SCHRÖDINGER EQUATION
ON UNBOUNDED DOMAINS**

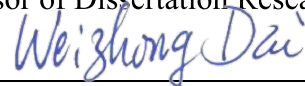
be accepted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computational Analysis & Modeling



Weizhong Dai

Supervisor of Dissertation Research



Weizhong Dai

Head of Computational Analysis & Modeling

Doctoral Committee Members:

Dentcho Genov

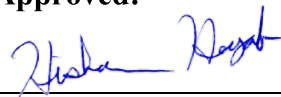
Songming Hou

Lee Sawyer

Box Leangsuksun

Neven Simicevic

Approved:



Hisham Hegab

Dean of Engineering & Science

Approved:



Ramu Ramachandran

Dean of the Graduate School

ABSTRACT

The finite-difference time-domain (FDTD) method and its generalized variant (G-FDTD) are efficient numerical tools for solving the linear and nonlinear Schrödinger equations because not only are they explicit, allowing parallelization, but they also provide high-order accuracy with relatively inexpensive computational costs. In addition, the G-FDTD method has a relaxed stability condition when compared to the original FDTD method. It is important to note that the existing simulations of the G-FDTD scheme employed analytical solutions to obtain function values at the points along the boundary; however, in simulations for which the analytical solution is unknown, theoretical approximations for values at points along the boundary are desperately needed. Hence, the objective of this dissertation research is to develop absorbing boundary conditions (ABCs) so that the G-FDTD method can be used to solve the nonlinear Schrödinger equation when the analytical solution is unknown.

To create the ABCs for the nonlinear Schrödinger equation, we initially determine the associated Engquist-Majda one-way wave equations and then proceed to develop a finite difference scheme for them. These ABCs are made to be adaptive using a windowed Fourier transform to estimate a value of the wavenumber of the carrier wave. These ABCs were tested using the nonlinear Schrödinger equation for 1D and 2D soliton propagation as well as Gaussian packet collision and dipole radiation. Results show that these ABCs perform well, but they have three key limitations. First, there are inherent reflections at the interface of the

interior and boundary domains due to the different schemes used the two regions; second, to use the ABCs, one needs to estimate a value for the carrier wavenumber and poor estimates can cause even more reflection at the interface; and finally, the ABCs require different schemes in different regions of the boundary, and this domain decomposition makes the ABCs tedious both to develop and to implement.

To address these limitations for the FDTD method, we employ the fractional-order derivative concept to unify the Schrödinger equation with its one-way wave equation over an interval where the fractional order is allowed to vary. Through careful construction of a variable-order fractional momentum operator, outgoing waves may enter the fractional-order region with little to no reflection and, inside this region, any reflected portions of the wave will decay exponentially with time. The fractional momentum operator is then used to create a fractional-order FDTD scheme. Importantly, this single scheme can be used for the entire computational domain, and the scheme smooths the abrupt transition between the FDTD method and the ABCs. Furthermore, the fractional FDTD scheme relaxes the precision needed for the estimated carrier wavenumber. This fractional FDTD scheme is tested for both the linear and nonlinear Schrödinger equations. Example cases include a 1D Gaussian packet scattering off of a potential, a 1D soliton propagating to the right, as well as 2D soliton propagation, and the collision of Gaussian packets. Results show that the fractional FDTD method outperforms the FDTD method with ABCs.

Replace this page with the approval for scholarly dissemination form.

DEDICATION

To those I have lost, and those I have found.

TABLE OF CONTENTS

ABSTRACT.....	iii
DEDICATION	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
ACKNOWLEDGMENTS	xiv
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 BACKGROUND OVERVIEW	7
2.1 Schrödinger Equations.....	7
2.2 FDTD and G-FDTD Methods	10
2.2.1 1D Case for the FDTD Method	10
2.2.2 1D Case for the G-FDTD Method	11
2.2.3 2D Case for the G-FDTD Method	14
2.2.4 Methods to Solve the Schrödinger Equation on Unbounded Physical Domains	15
2.3 Fractional-Order Derivatives and Approximations Thereof.....	19
2.3.1 Fractional-Order Derivatives	20
2.3.2 Lagrange Approximations	24
2.4 GPU Computing	30
2.5 Chapter Summary	31

CHAPTER 3	ABSORBING BOUNDARY CONDITIONS	33
3.1	One-Way Wave Equations	33
3.2	One-Way Wave Equations for 1D Case	34
3.3	One-Way Wave Equations for 2D Case	36
3.4	Absorbing Boundary Conditions	37
3.4.1	1D Case for Absorbing Boundary Conditions	37
3.4.2	2D Case for Absorbing Boundary Conditions	39
3.5	Near-Boundary Treatment	41
3.6	ABC Parameter Choice	44
3.7	Adaptive Parameter Selection	45
3.7.1	Computational Procedure	46
3.8	Chapter Summary	51
CHAPTER 4	NUMERICAL RESULTS FOR ABC METHOD	52
4.1	Testing the ABC	52
4.2	Testing the Parallel Algorithm	57
4.3	Numerical Examples	60
4.4	Chapter Summary	67
CHAPTER 5	FRACTIONAL MOMENTUM OPERATOR AND THE FRACTIONAL FDTD SCHEME	71
5.1	Fractional Momentum Operator	71
5.2	1D Fractional FDTD Method	77
5.3	2D Fractional FDTD Method	80
5.4	Discussion	81

5.5 Chapter Summary	82
CHAPTER 6 NUMERICAL RESULTS FOR FRACTIONAL FDTD METHOD	83
6.1 Parameter Selection	83
6.2 Numerical Examples.....	92
6.3 Examples in 1D.....	95
6.4 Examples in 2D.....	98
6.5 Discussion.....	106
6.6 Chapter Summary	108
CHAPTER 7 CONCLUSION.....	109
REFERENCES	111

LIST OF TABLES

Table 4.1:	Parameters for Power Law Fitting for GPU/CPU Comparisons.	60
------------	--	----

LIST OF FIGURES

Figure 2.1:	Diagram of 1D implementation of the FML. The SE is used in the domain of physical interest; the TBC is used on the computational boundaries; and the FML is sandwiched between the two.	19
Figure 2.2:	Plot of the gamma function $\Gamma(x)$	20
Figure 3.1:	(a) Illustration (to scale) of the 1D G-FDTD implementation with ABC for arbitrary M_{\max} in the main computational region. For brevity, $B = 4M_{\max} + 1$. (b) Illustration (not to scale) of the 2D G-FDTD implementation with ABC where $M_{\max} = 1$. (c) Illustration (to scale) of the northeast corner of the 2D implementation.	42
Figure 3.2:	High-level flowchart of the algorithm used for the 2-D G-FDTD with ABC. Here, D_1 is the main computation region where $M = 1$ and D_0 is the square annular region where $M = 0$. Each domain with a cardinal direction as a subscript refers to the side boundary. Refer to Fig. 3.1 on page 42 for an illustration of D_0 , D_1 , D_N , D_E , D_S , and D_W	48
Figure 3.3:	Pseudocode for an algorithm for implementing the G-FDTD method with ABC in parallel.	49
Figure 3.4:	Example code for adding two vectors in parallel using PyOpenCL.	50
Figure 4.1:	Reflection coefficient as a function of the wavenumber k of the incoming soliton as (a) the total number of points used in the simulation varies, (b) the ABC was preconditioned with the various wavenumbers k_0 , (c) the window width \tilde{b} of the Gabor transform was varied for the adaptive case, and (d) the ABC was compared with implicit scheme developed by Zhang <i>et al.</i> [29].	55
Figure 4.2:	<i>Top:</i> Average total computation time needed to calculate one unit of time for the 2-D NLSE using the G-FDTD method with ABC plotted versus the linear resolution used. <i>Bottom:</i> The speed-up (CPU Timing/GPU Timing) versus linear resolution.	59

Figure 4.3:	(Example 4.1) Space-time plot of a simulation of the propagation of a 1D bright soliton $ \psi(x, t) $ with $p = 3$, $\lambda = -2$ where (a) the analytical solution was calculated, (b) the G-FDTD method was used with boundary points set to zero, (c) the G-FDTD method was used with boundary values given by the analytical solution, and (d) the G-FDTD method was used with ABC on the boundary.	62
Figure 4.4:	(Example 4.2, Case 1). Simulation of soliton propagating diagonally toward the boundary using G-FDTD method with $p = 3$, $\lambda = -2$, $\Delta t = (\Delta x)^2/32$ where (a) the analytical solution was used on boundary, (b) the boundary values were set to zero, and (c) the boundaries were calculated using ABCs.	64
Figure 4.5:	(Example 4.2, Case 2) Simulation soliton propagating with different wavenumbers on each boundary toward the boundary using G-FDTD method with $p = 3$, $\lambda = -2$, $\Delta t = (\Delta x)^2/32$ where (a) the analytical solution was used on boundary, (b) the boundary values were set to zero, and (c) the boundaries were calculated using ABCs.	66
Figure 4.6:	(Example 4.3). Simulations of Gaussian packet collision using G-FDTD with $p = 3$, $\lambda = -2$ where (a) the boundary points were set to zero, and (b) ABC was used to calculate boundary points.	68
Figure 4.7:	(Example 4.4). Simulations of in-phase dipole radiation with dipole distance $d = 2$ and angular frequency $\omega = 6$ using G-FDTD with $p = 3$, $\lambda = -2$ where (a) the boundary points were set to zero, and (b) the ABC was used on the boundary.	69
Figure 6.1:	Plot of the sigmoid functions $\sigma_1(u)$, $\sigma_2(u)$, and $\sigma_3(u)$. The linear function that satisfies the continuity conditions, but not the piecewise smooth conditions, has been plotted as a black dashed line.	86
Figure 6.2:	Plot of the sigmoid functions $\sigma_1(u)$, $\sigma_2(u)$, and $\sigma_3(u)$ where the function $f(u) = 1 - u$ has been subtracted from each sigmoid.	86
Figure 6.3:	Plot of the sigmoid function $\sigma_3(u; q)$, for various parameters q . The inflection point for each sigmoid is given by the point $(q, 1 - q)$ and is labeled as a black dot. For each sigmoid, the parameter q is simply the u -coordinate of the inflection point.	88
Figure 6.4:	Plot of the reflection coefficient $R(k; L, \kappa)$ where the maximum was taken over the width parameter $1 \leq L \leq 8$	90
Figure 6.5:	Contour plot of the reflection coefficient $R(k; L, \kappa)$ where the maximum was taken over velocity parameter $1 \leq \kappa \leq 10$	91

Figure 6.6:	Contour plot of the reflection coefficient $R(k; L, \kappa)$ where the data was limited to the slice $\kappa = k$.	93
Figure 6.7:	Contour plot of the reflection coefficient $R(k; L, k)$ where the maximum was taken over $k \in [0, 10]$.	94
Figure 6.8:	(Example 6.1). <i>Top</i> : Simulation of a 1D soliton traveling toward the right boundary using the fractional FDTD method with FML. <i>Bottom</i> : Plot of fractional order $\alpha(x)$.	97
Figure 6.9:	(Example 6.2). <i>Top</i> : Simulation of a 1D particle traveling to the right using fractional FDTD method with FML, where there is a step potential at $x = 0$. <i>Bottom</i> : Plot of fractional order $\alpha(x)$.	99
Figure 6.10:	(Example 6.3, Case 1). Surface plot of $ \psi ^2$ of a 2D soliton using where ψ is determined by (a) the analytical solution, (b) the FDTD method with ABC, (c) the fractional FDTD method with FML. The interior of each dotted square represents the physical domain.	101
Figure 6.11:	(Example 6.3, Case 1). Close up comparison of ABC method and fractional FDTD method with FML at $t = 3.0$. Notice the high-frequency distortions present when using the ABC method, but not present when using the FML method.	102
Figure 6.12:	(Example 6.3, Case 1). The simulation where the FML was used was allowed to run longer to show that the wave was, in fact, absorbed.	103
Figure 6.13:	(Example 6.3, Case 2). Surface plot of $ \psi ^2$ of a 2D soliton using where ψ is determined by (a) the analytical solution, (b) FDTD method with ABC, (c) fractional FDTD method with FML. The interior of each black-dotted square represents the physical domain.	105
Figure 6.14:	(Example 6.4). Plot of $ \psi ^2$ for two 2D Gaussian packets colliding where ψ is determined by (a) the FDTD method where function values outside the domain of physical interest are set to be zero, (b) FDTD method with ABC, (c) fractional FDTD method with FML. The interior of each black-dotted square represents the physical domain.	107

ACKNOWLEDGMENTS

This research was supported by a research fellowship grant from the Louisiana Board of Regents [LEQSF(2015-2020)-GF-06]. The author would like to thank his advisor, Weizhong Dai, and his advisory committee members, Dentcho Genov, Songming Hou, Lee Sawyer, Box Leangsuksun, and Neven Simicevic, for their valuable inputs and suggestions pertaining to the research presented in this dissertation. The author would also like to thank David M. Merchant and Wanda F. Kaiser for their constructive criticism and help during the copy-editing process.

CHAPTER 1

INTRODUCTION

It is well-known that the time-dependent Schrödinger equation (SE) models the propagation of waves, particles, as well as solitons in scientific fields such as nonlinear optical and electromagnetic media, quantum superconductors, fluid mechanics, Bose-Einstein condensates, polariton fluids, plasma physics, DNA energy transport, and deep water rogue waves [1–13]. The SE and nonlinear SE (NSE) can be expressed jointly as

$$i\partial_t\psi - \nabla^2\psi + V\psi + \lambda|\psi|^{q-1}\psi = 0, \quad (1)$$

where $i = \sqrt{-1}$, $\psi = \psi(t, \mathbf{x})$ is a complex-valued function of the time variable t and the n -dimensional position vector \mathbf{x} , and ∇^2 is the Laplace operator, and V is a potential [2]. For $\lambda = 0$, we have the SE, and when $V = 0$ we have the NSE. The integer $q \geq 3$ determines the order of the nonlinear coupling, while the real constant λ can be either positive or negative corresponding to repulsive or attractive behavior, respectively [14, 15]. For $V = 0$, well-known soliton solutions exist for both positive and negative values of λ . For $\lambda < 0$, wave disturbances, known as bright solitons, exist as excitations on top of a zero density background medium. For $\lambda > 0$, wave disturbances, known as dark solitons, exist as areas of lower excitation in a constant density background. In this study, we consider only the attractive case ($\lambda < 0$) as the repulsive case requires entirely different boundary considerations due to the constant density background [2, 14, 16]. For $\lambda < 0$, the NSE

permits bright hyperbolic secant solutions. In one dimension (1D), the following is a well-known soliton solution to the NSE:

$$\psi(x, t) = a|2/\lambda|^{1/2} \text{sech}(a(x - 2kt)) e^{ikx} e^{-i(a^2 - k^2)t} \quad (1.1)$$

where $a > 0$ and k is real. Later on, we will use these soliton solutions (and similar 2D solitons) to test the validity of our numerical methods.

The G-FDTD method is an extension of the FDTD method which provides a way to relax the stability condition of the FDTD method, and it provides higher-order temporal accuracy while not drastically increasing the computational cost [17, 18]. Finally, both the FDTD and G-FDTD methods are explicit schemes which allow for easy parallelization. It is important to note that the existing simulation of the G-FDTD scheme employed analytical solutions to obtain function values at the points along the boundary; however, in simulations for which the analytical solution is unknown, theoretical approximations for values at points along the boundary are desperately needed. Hence, it requires us to develop absorbing boundary conditions (ABCs) so that the G-FDTD method can be used to solve the nonlinear Schrödinger equation when the analytical solution is unknown.

There are many sophisticated ABCs proposed for the SE [19–28]. In particular, Zhang et al. [29, 30] used the operator splitting method to design the ABCs for the 1D and 2D SE on unbounded domains. Notably, Antoine et al. used time-fractional operators with fractional order of 1/2 to derive the artificial boundary condition for 1D cubic nonlinear SE on unbounded domains [31]. Other approaches to design the ABCs for SE on unbounded domains can be found in [16, 32–45] and the references therein.

It should be pointed out that many ABCs attempt to approximate the solution along the boundary using some assumed functional form. However, if the outgoing wave does not sufficiently satisfy the assumed form, there can be substantial reflection at the interface of the interior and boundary domains. To overcome this troublesome issue, our intuition is to somehow transform from the SE into the one-way wave equation (that has transparent boundary conditions; abbreviated TBCs) by introducing an idealized variable-order fractional momentum operator, which allows the order of the fractional momentum to decrease *gradually* from the SE to the one-way wave equation through a fractional momentum layer (FML). In so doing, we aim to smooth the abrupt transition. As such, through careful construction of the FML, we hope that the reflected waves will decay rapidly and the total accumulated error inside the physical region is thereby reduced.

Objective. The objective of this dissertation research is to develop two boundary methods so that the FDTD and G-FDTD method can be used to solve the linear and nonlinear SE when the analytical solution is unknown. The first method is an ABC that is based on the solely on the Engquist-Majda one-way wave equations. The second method is a novel technique that unifies the Schrödinger equation with its one-way wave equation through the use of the fractional-order derivative concept.

For the first method, to develop the ABCs, we discretize the Engquist-Majda one-way wave equations to create a finite difference scheme. The ABCs are made to be adaptive using a windowed Fourier transform to estimate a value of the wavenumber of the carrier wave, a parameter that is needed for the ABC. While the obtained ABCs provide good results when computing with the FDTD method, they have three key limitations. First, there is inherent reflection at the interface of the interior and boundary domains due to

the different schemes used in region. Second, to use the ABCs, one needs to estimate a value for the carrier wavenumber and poor estimates can cause even more reflection at the interface. Finally, the ABCs require different schemes in different regions of the boundary, and this domain decomposition makes the ABCs tedious both to develop and to implement; moreover, as the dimensionality of the SE increases, the number of edges – hence the number required of ABCs – increases rapidly.

To address these limitations for the ABCs, we employ the fractional-order derivative concept to unify the Schrödinger equation with its one-way wave equation over an interval where the fractional order is allowed to vary. Through careful construction of a variable-order fractional momentum operator, outgoing waves may enter the fractional-order region with little to no reflection and, inside this region, any reflected portions of the wave will decay exponentially with time. The fractional momentum operator is then used to create a fractional-order FDTD scheme. Importantly, this single scheme can be used for the entire computational domain, and the scheme smooths the abrupt transition between the FDTD method and the ABCs. Furthermore, the fractional FDTD scheme relaxes the precision needed for the estimated carrier wavenumber.

Organization. The organization of this dissertation is structured as follows:

Chapter 1 provides a brief overview and motivation for this research by (1) describing areas of applications of the Schrödinger equation, (2) discussing the existing numerical solutions and their limitations, (3) introducing the need for wave-absorption methods when

numerically solving the Schrödinger equation on physically unbounded domains in particular absorbing boundary conditions; and, (4) explaining the limitations of absorbing boundary conditions and (5) giving a road map for using fractional-order techniques for wave absorption. This chapter also provides an overview for the structure of this dissertation.

Chapter 2 serves as an introduction to the linear and nonlinear Schrödinger equations. It also discusses the main numerical methods that are used throughout this study, and it provides an overview of relevant fractional derivative techniques used in the study. Additionally, the advantages of GPU computing over CPU computing are discussed.

Chapter 3 develops the one-way wave equations for the 1D and 2D cases. Next, we discuss the near-boundary treatment so that the ABC can be used with the Generalized Finite-Difference Time-Domain method for solving the nonlinear Schrödinger equation. Then, since the ABCs require certain parameters, we discuss the heuristic we use for determining those parameters and a method for adaptively choosing parameters. Finally, we develop the scheme for the absorbing boundary conditions by using finite difference techniques.

Chapter 4 gives the tests and numerical examples of the ABC method. These experiments include simulations of nonlinear SE for 1D and 2D soliton propagation as well as Gaussian packet collision.

Chapter 5 introduces a symbolically idealized fractional-order momentum operator that unifies the SE with its one-way wave equation. The idealized operator is then developed through careful consideration of the behavior of fractional momentum operators along the left and right boundaries. Finally, the fractional FDTD method is developed by determining the discrete version of the fractional SE using finite difference techniques.

Chapter 6 gives the tests and numerical examples for the fractional FDTD scheme. These experiments include 1D soliton propagation, 1D particle interacting with a potential, as well as 2D soliton propagation and Gaussian packet collision.

Chapter 7 concludes this dissertation and provides future direction in research.

CHAPTER 2

BACKGROUND OVERVIEW

This chapter provides an overview of the linear and nonlinear Schrödinger equations as well as the G-FDTD method which can be used for accurate simulation. We also discuss boundary methods such as absorbing boundary conditions and provide motivation for the new fractional FDTD method and the fractional momentum layer. Furthermore, the relevant topics from fractional calculus are discussed and several fractional derivatives are introduced. Finally, we discuss the advantages of GPU computing for scientific applications.

2.1 Schrödinger Equations

The linear Schrödinger equation (SE) is an important equation in physics and chemistry as well as other science and engineering disciplines. The linear SE describes the time evolution of non-relativistic massive particles interacting with potentials [46]. With $\hbar = 1$ and $m = \frac{1}{2}$, linear SE is given by

$$i\partial_t\psi = \hat{p}^2\psi + V\psi, \quad (2.1)$$

where $i = \sqrt{-1}$, $\psi = \psi(\mathbf{x}, t)$ is a complex wave function of the position vector $\mathbf{x} = \langle x_1, \dots, x_n \rangle$ and the time coordinate t . The momentum operator is given by $\hat{\mathbf{p}} = -i\hbar\nabla$ where ∇ is the gradient, $\hat{p}^2 = \hat{\mathbf{p}} \cdot \hat{\mathbf{p}}$, and $V = V(\mathbf{x}, t)$ describes a potential.

The nonlinear Schrödinger equation (NSE) characterizes quasi-monochromatic wave phenomena that appears in weakly nonlinear, dispersive media where dissipative processes are negligible [2]. The NSE can be expressed as

$$i\partial_t\psi = \hat{p}^2\psi - \lambda|\psi|^{q-1}\psi. \quad (2.2)$$

The NSE appears in the literature of mathematical physics when describing the behavior of Bose-Einstein condensates, superconductors, nonlinear optical media, plasmas, DNA energy transport, as well as deep-water rogue waves [2–11]. The integer $q \geq 3$ determines order of the nonlinear coupling, and the proportionality constant λ can be either positive or negative corresponding to defocusing or focusing behavior, respectively [14, 15]. For $q = 3$, soliton solutions exist for both positive and negative values of λ . For $\lambda < 0$, wave disturbances, known as bright solitons, exist as excitations on top of a zero density background. For $\lambda > 0$, wave disturbances, known as dark solitons, exist as areas of lower excitation on top of a constant, non-zero background. In this study, we consider only the focusing case ($\lambda < 0$) as the defocusing case requires different considerations [2, 14, 16, 47]. For $\lambda < 0$, the NLSE permits bright hyperbolic secant solutions. The following soliton is a known solution for the 1D case:

$$\psi(x, t) = a|2/\lambda|^{1/2} \text{sech}(a(x - 2kt)) e^{ikx} e^{-i(a^2 - k^2)t} \quad (2.3a)$$

where $a > 0$ and k is real. Additionally, for the 2D case, the following soliton solution is permitted:

$$\psi(\mathbf{x}, t) = a|2/\lambda|^{1/2} \text{sech}(\mathbf{a} \cdot (\mathbf{x} - 2t\mathbf{k})) e^{i\mathbf{k} \cdot \mathbf{x}} e^{-i(k^2 - a^2)t} \quad (2.3b)$$

where $\mathbf{x} = \langle x, y \rangle$, $\mathbf{a} = a\hat{\mathbf{u}}_a$, $\mathbf{k} = k\hat{\mathbf{u}}_k$, for some unit vectors $\hat{\mathbf{u}}_a$ and $\hat{\mathbf{u}}_k$. Letting $\mathcal{V} = V + \lambda|\psi|^{q-1}$, then Eqs. (2.1) and (2.2) can be unified with the following equation

$$i\partial_t\psi = \hat{p}^2\psi + \mathcal{V}\psi, \quad (2.4)$$

where $\lambda = 0$ corresponds to the linear case and $V = 0$ corresponds to the nonlinear case.

Continuity Equation and Conservation Law. We will show that any function that satisfies Eq. (2.4), will also satisfy conservation of probability. Specifically, any solution to Eq. (2.4) also satisfies a continuity equation. Thereby, we can prove that total probability is conserved, and for any volume Σ , the total probability can change if and only if probability flows through the boundary of that region, $\partial\Sigma$.

By multiplying Eq. (2.4) by the conjugate function $\bar{\psi}$, multiplying the conjugate of Eq. (2.4) by ψ , then taking the difference of the results, one may obtain that solution to the SE satisfies the following continuity equation:

$$\partial_t|\psi|^2 + \nabla \cdot \mathbf{j} = 0, \quad (2.5)$$

where the probability current is defined as $\mathbf{j} = -i(\bar{\psi}\nabla\psi - \psi\nabla\bar{\psi})$ [46]. Now, if we assume $\psi(x, y, t) = \psi_0(x, y, t)e^{\pm i\mathbf{k}\cdot\mathbf{x}}$ where \mathbf{k} is a constant vector, then we obtain $|\psi(x, y, t)| = |\psi_0(x, y, t)|$ and $\mathbf{j} = 2|\psi|^2\mathbf{k}$. Hence, the continuity equation simplifies to be

$$\partial_t|\psi|^2 + 2\mathbf{k} \cdot \nabla|\psi|^2 = 0. \quad (2.6)$$

In the 1D case, the above equation further simplifies to $\partial_t|\psi|^2 + 2k\partial_x|\psi|^2 = 0$.

To obtain the conservation law, we integrate over some volume Σ and use the divergence theorem to find

$$\frac{d}{dt} \int_{\Sigma} |\psi|^2 dV + \oint_{\partial\Sigma} \mathbf{j} \cdot d\mathbf{s} = 0. \quad (2.7)$$

For unbounded domains such as \mathbb{R}^2 , $\oint_{\partial\Sigma} \mathbf{j} \cdot \mathbf{ds} = 0$ and

$$\int_{\Sigma} |\psi|^2 dV = \text{constant}. \quad (2.8)$$

2.2 FDTD and G-FDTD Methods

As stated previously, there are many explicit numerical methods for solving the NSE [17, 18, 27, 48–81]. Namely, these include the spectral and pseudospectral methods [55–57], finite difference methods [58–70], space-time finite-element methods [71], quadrature discretization methods [72–75], and finite-difference time-domain (FDTD) methods [17, 18, 27, 76–81].

In this study, we use the Generalized Finite-Difference Time-Domain (G-FDTD) method developed by Dai and Moxley *et al.* [18, 27, 77–80]. We choose to use the G-FDTD method not only because it is explicit and thus allows parallelization, but also because it provides high-order accuracy with relatively inexpensive computation. Furthermore, the G-FDTD method has a relaxed stability condition when compared to the original FDTD method [17, 18, 27].

2.2.1 1D Case for the FDTD Method

The one-dimensional FDTD method for the *linear* SE is presented by Sullivan in [17].

Recall that the 1D SE is given by

$$i\hbar\partial_t\psi = -\frac{\hbar^2}{2m}\partial_x^2\psi + \mathcal{V}(x,t)\psi. \quad (2.9)$$

The SE is then split into real valued functions ψ_{R} and ψ_{I} where $\psi = \psi_{\text{R}} + i\psi_{\text{I}}$ giving the coupled equations

$$\partial_t\psi_{\text{R}} = +\left(\frac{\hbar}{2m}\partial_x^2 - \hbar^{-1}\mathcal{V}\right)\psi_{\text{I}}, \quad (2.10a)$$

$$\partial_t \psi_{\text{I}} = - \left(\frac{\hbar}{2m} \partial_x^2 - \hbar^{-1} \mathcal{V} \right) \psi_{\text{R}}. \quad (2.10b)$$

To obtain the FDTD method, we evaluate the real and imaginary parts of ψ at the point $(x_i, t_{n+1/2})$ and let $\psi(x_i, t_n)$ be approximated by ψ_i^n . Using the finite difference approximations $\partial_t \psi|_{x_i}^{t_{n+1/2}} \approx (\psi_i^{n+1} - \psi_i^n)/\Delta t$ and $\partial_x^2 \psi|_{x_i}^{t_{n+1/2}} \approx (\psi_{i+1}^{n+1/2} - 2\psi_i^{n+1/2} + \psi_{i-1}^{n+1/2})/\Delta x^2$, we obtain the following numerical scheme:

$$[\psi_{\text{R}}]_i^{n+1} - [\psi_{\text{R}}]_i^n = + \left(\tilde{\mu} \delta_x^2 - \hbar^{-1} \mathcal{V}_i^{n+1/2} \right) [\psi_{\text{I}}]_i^{n+1/2}, \quad (2.11a)$$

$$[\psi_{\text{I}}]_i^{n+1} - [\psi_{\text{I}}]_i^n = - \left(\tilde{\mu} \delta_x^2 - \frac{1}{\Delta t \hbar} \mathcal{V}_i^{n+1/2} \right) [\psi_{\text{R}}]_i^{n+1/2}, \quad (2.11b)$$

where $\tilde{\mu} = \Delta t / \Delta x^2 \mu$, $\mu = \hbar / 2m$, and δ_x^2 is the second-order central difference operator defined by

$$\delta_x^2 f_i = f_{i+1} - 2f_i + f_{i-1}. \quad (2.12)$$

The FDTD method is the inspiration for and a special case of the G-FDTD method that follows.

2.2.2 1D Case for the G-FDTD Method

The derivation of the G-FDTD method is similar to the development of the Lax-Wendroff scheme for hyperbolic equations [82]. To start, consider a sufficiently smooth function $f = f(t)$ and some small time step Δt , then we have following Taylor series expansion

$$f(t \pm \Delta t/2) = \sum_{n=0}^{\infty} (\pm 1)^n \frac{\Delta t^n}{2^n n!} f^{(n)}(t). \quad (2.13)$$

The above expansion may be used to obtain the following sum where the even terms vanish,

$$f(t + \Delta t/2) - f(t - \Delta t/2) = 2 \sum_{n=0}^{\infty} \frac{\Delta t^{2n+1}}{2^{2n+1} (2n+1)!} f^{(2n+1)}(t) \quad (2.14)$$

which simplifies as

$$f(t + \Delta t/2) - f(t - \Delta t/2) = \sum_{n=0}^{\infty} \frac{\Delta t^{2n+1}}{4^n (2n+1)!} f^{(2n+1)}(t). \quad (2.15)$$

Letting $t \rightarrow t - \Delta t$ yields

$$f(t) - f(t - \Delta t) = \sum_{n=0}^{\infty} \frac{\Delta t^{2n+1}}{4^n (2n+1)!} f^{(2n+1)}(t - \Delta t/2). \quad (2.16)$$

From Eq. (2.16), it is clear that we need to determine expressions for the odd-order temporal derivatives in order to have increased accuracy in determining how the system will evolve at each time step. Truncating the series at the term $m = M$, then we have the finite sum that follows:

$$f(t) - f(t - \Delta t) = \sum_{m=0}^M \frac{\Delta t^{2m+1}}{4^m (2m+1)!} f^{(2m+1)}(t - \Delta t/2) + O(\Delta t^{2M+3}). \quad (2.17)$$

At onset, determining expressions for these time derivatives is not trivial; however, by making some approximations, the expressions will become easier to handle. Now, consider the one dimensional (1D) SE given by

$$i\hbar \partial_t \psi = -\frac{\hbar^2}{2m} \partial_x^2 \psi + \mathcal{V}(x, t) \psi. \quad (2.18)$$

Solving for $\partial_t \psi$ yields

$$\partial_t \psi = -\frac{i}{\hbar} H \psi. \quad (2.19)$$

where $H = -\frac{\hbar^2}{2m} \partial_x^2 + \mathcal{V}(x, t)$ is a (possibly nonlinear) Hamiltonian. This gives our first temporal derivative. To find the third-order temporal derivative, we first need to find the second-order one. The second-order temporal derivative can be obtained by differentiating Eq. (2.19) with respect to time. For now, we will take special care to consider the commutation relation of the Hamiltonian and the time derivative operator $\partial_t H = H \partial_t + \mathcal{V}_t$ where

$\mathcal{V}_t = \frac{\partial \mathcal{V}}{\partial t}$. Then, the second temporal derivative is given by

$$\partial_t^2 \psi = \partial_t (\partial_t \psi) = -\frac{i}{\hbar} \partial_t H \psi = -\frac{i}{\hbar} \left(H \partial_t + \frac{i}{\hbar} \mathcal{V}_t \right) \psi = -\frac{1}{\hbar^2} (H^2 + i\hbar \mathcal{V}_t) \psi. \quad (2.20)$$

Furthermore, the third temporal derivative is given by

$$\partial_t^3 \psi = -\frac{1}{\hbar^2} \partial_t (H^2 + i\hbar \mathcal{V}_t) \psi = \frac{i}{\hbar^3} (H^3 - \hbar^2 \mathcal{V}_{tt} + i\hbar (H \mathcal{V}_t + 2\mathcal{V}_t H)) \psi, \quad (2.21)$$

where $\mathcal{V}_{tt} = \frac{\partial^2 \mathcal{V}}{\partial t^2}$. At this point, it is clear that finding higher and higher temporal derivatives can quickly become intractable without some assumptions. However, for $M = 1$, we have

$$\begin{aligned} \psi(t) - \psi(t - \Delta t) &= -\frac{i\Delta t}{\hbar} H \psi(t - \Delta t/2) + \frac{i\Delta t^3}{24\hbar^3} H^3 \psi(t - \Delta t/2) \\ &\quad + \frac{\Delta t^3}{24\hbar} (H \mathcal{V}_t + 2\mathcal{V}_t H - \mathcal{V}_{tt}) \psi(t - \Delta t/2). \end{aligned} \quad (2.22)$$

In [18], through a process known as linearization, the authors assume that \mathcal{V} is independent of t , that is $\frac{\partial \mathcal{V}}{\partial t} = 0$ (or equivalently the Hamiltonian commutes with the time-derivative operator). In this way, the mathematics becomes much more simple, but the G-FDTD method still provides increased accuracy. In particular, the expressions for the temporal derivative are given as $\partial_t^m \psi = (\frac{-i}{\hbar})^m H^m \psi$, which yields the update equation

$$\psi(t) - \psi(t - \Delta t) = i \sum_{m=0}^M \frac{(-1)^{m+1} \Delta t^{2m+1}}{4^m (2m+1)!} H^{2m+1} \psi(t - \Delta t/2). \quad (2.23)$$

Letting $\psi^n = \psi(t_n)$ where $t_n = n\Delta t$, we have

$$\psi^n - \psi^{n-1} = i \sum_{m=0}^M \frac{(-1)^{m+1} \Delta t^{2m+1}}{4^m (2m+1)!} H^{2m+1} \psi^{n-1/2}, \quad (2.24)$$

which we may separate into real and imaginary components using $\psi^n = [\psi_{\text{R}}]^n + i[\psi_{\text{I}}]^n$ to obtain the coupled equations that follow:

$$[\psi_{\text{R}}]^n = [\psi_{\text{R}}]^{n-1} + \sum_{m=0}^M \frac{(-1)^m \Delta t^{2m+1}}{4^m (2m+1)!} H^{2m+1} [\psi_{\text{I}}]^{n-1/2}, \quad (2.25)$$

$$[\psi_{\text{I}}]^n = [\psi_{\text{I}}]^{n-1} - \sum_{m=0}^M \frac{(-1)^m \Delta t^{2m+1}}{4^m (2m+1)!} H^{2m+1} [\psi_{\text{R}}]^{n-1/2}. \quad (2.26)$$

What remains now is to find appropriate finite difference approximations for the Hamiltonian operator. Letting $H|_{x=x_i}^{t=t_{n-1/2}} = \partial_x^2 - \mathcal{V}_i^{n+1/2}$ where $x_i = a + i\Delta x$, $i = 0, \dots, N$, $x_N = b$, the 1D G-FDTD method is given as follows:

$$[\psi_{\text{R}}]_i^{n+1/2} = [\psi_{\text{R}}]_i^{n-1/2} + \sum_{m=0}^M \frac{(-1)^m}{4^m (2m+1)!} \left(\mu_x D_x^2 - \frac{\Delta t}{\hbar} \mathcal{V}_i^{n+1} \right)^{2m+1} [\psi_{\text{I}}]_i^n, \quad (2.27a)$$

$$[\psi_{\text{I}}]_i^{n+1/2} = [\psi_{\text{I}}]_i^{n-1/2} + \sum_{m=0}^M \frac{(-1)^{m+1}}{4^m (2m+1)!} \left(\mu_x D_x^2 - \frac{\Delta t}{\hbar} \mathcal{V}_i^{n+1} \right)^{2m+1} [\psi_{\text{R}}]_i^n, \quad (2.27b)$$

$$[\psi_{\text{R}}]_i^{n+1} = [\psi_{\text{R}}]_i^n + \sum_{m=0}^M \frac{(-1)^m}{4^m (2m+1)!} \left(\mu_x D_x^2 - \frac{\Delta t}{\hbar} \mathcal{V}_i^{n+1/2} \right)^{2m+1} [\psi_{\text{I}}]_i^{n+1/2}, \quad (2.27c)$$

$$[\psi_{\text{I}}]_i^{n+1} = [\psi_{\text{I}}]_i^n + \sum_{m=0}^M \frac{(-1)^{m+1}}{4^m (2m+1)!} \left(\mu_x D_x^2 - \frac{\Delta t}{\hbar} \mathcal{V}_i^{n+1/2} \right)^{2m+1} [\psi_{\text{R}}]_i^{n+1/2}, \quad (2.27d)$$

for $n = 0, 1, 2, 3, \dots$, where $\mu_x = \Delta t / \hbar \Delta x^2$ and D_x^2 is a finite-difference approximation for the 1D Laplacian. In the original paper [18], D_x^2 was the fourth-order central difference operator given as

$$D_x^2 \psi_j = -\frac{1}{12} (\psi_{j+2} - 16\psi_{j+1} + 30\psi_j - 16\psi_{j-1} + \psi_{j-2}). \quad (2.28)$$

Notice that for $M = 0$ the G-FDTD method reduces to the FDTD method. The truncation error is $\mathcal{O}(\Delta t^{2M+2} + \Delta x^4)$, and the stability condition for this method is given as

$$\left| \sum_{m=0}^M \frac{(-1)^m}{(2m+1)!} \left(\frac{8}{3} \mu_x + \frac{V^{p-1} \Delta t}{2} \right)^{2m+1} \right| \leq c < 1, \quad (2.29)$$

where c, V, T are real constants such that $c \in [0, 1)$ and $\max_{(x_j, t_n) \in S} |\psi_j^n| < V$ where $S = R_1 \times (0, T)$ [18].

2.2.3 2D Case for the G-FDTD Method

Similarly, for the two dimensional (2D) case, we consider a bounded rectangular region $R_2 \subset \mathbb{R}^2$ and assume that $\psi = \psi(x, y, t)$ is integrable over R_2 where $\psi = \psi_{\text{R}} + i\psi_{\text{I}}$ for real

valued functions $\psi_R = \psi_R(x, y, t)$ and $\psi_I = \psi_I(x, y, t)$. Then, with $\psi_R(x_j, y_k, t_n) \approx [\psi_R]_{jk}^n$ and $\psi_I(x_j, y_k, t_n) \approx [\psi_I]_{jk}^n$, the 2D G-FDTD scheme is given as follows [18, 27]:

$$[\psi_R]_{jk}^{n+1/2} = [\psi_R]_{jk}^{n-1/2} + \sum_{m=0}^M \frac{(-1)^m}{4^m(2m+1)!} (D^2 - \Delta t \mathcal{V}_{jk}^n)^{2m+1} [\psi_I]_{jk}^n, \quad (2.30a)$$

$$[\psi_I]_{jk}^{n+1/2} = [\psi_I]_{jk}^{n-1/2} + \sum_{m=0}^M \frac{(-1)^{m+1}}{4^m(2m+1)!} (D^2 - \Delta t \mathcal{V}_{jk}^n)^{2m+1} [\psi_R]_{jk}^n, \quad (2.30b)$$

$$[\psi_R]_{jk}^{n+1} = [\psi_R]_{jk}^n + \sum_{m=0}^M \frac{(-1)^m}{4^m(2m+1)!} (D^2 - \Delta t \mathcal{V}_{jk}^{n+1/2})^{2m+1} [\psi_I]_{jk}^{n+1/2}, \quad (2.30c)$$

$$[\psi_I]_{jk}^{n+1} = [\psi_I]_{jk}^n + \sum_{m=0}^M \frac{(-1)^{m+1}}{4^m(2m+1)!} (D^2 - \Delta t \mathcal{V}_{jk}^{n+1/2})^{2m+1} [\psi_R]_{jk}^{n+1/2}, \quad (2.30d)$$

where $D^2 = \sigma_x D_x^2 + \sigma_y D_y^2$, $\sigma_x = \Delta t / \Delta x^2$, $\sigma_y = \Delta t / \Delta y^2$, and D_x^2 and D_y^2 are finite-difference approximations for the second derivative of ψ with respect to x and y , respectively. In the original paper, D_x^2 and D_y^2 were the fourth-order central difference operator.

That is,

$$D_x^2 \psi_{jk}^n = -\frac{1}{12} (\psi_{j+2k}^n - 16\psi_{j+1k}^n + 30\psi_{jk}^n - 16\psi_{j-1k}^n + \psi_{j-2k}^n), \quad (2.31a)$$

$$D_y^2 \psi_{jk}^n = -\frac{1}{12} (\psi_{jk+2}^n - 16\psi_{jk+1}^n + 30\psi_{jk}^n - 16\psi_{jk-1}^n + \psi_{jk-2}^n). \quad (2.31b)$$

The G-FDTD method has had promising results [18, 27, 77–80]; therefore, we would like to extend the method to include cases where the solution along the boundary is unknown; in particular, the case where the equation is not physically bounded.

2.2.4 Methods to Solve the Schrödinger Equation on Unbounded Physical Domains

Due to the unavoidable, inherent limitations of computing, especially when simulating unbounded physical domains, it is necessary to truncate the physical domain and impose a computational boundary. The computational boundary is imposed purely for the simulation;

therefore, it would be transparent to waves in the analytical solution. At this stage, there are a few options as follows:

- (i) **Lose simulation accuracy.** If no further action is taken, the domain of the *accurate* unbounded simulation will repeatedly shrink after each application of the Laplacian operator. This is because the discrete Laplacian operator is a kernel convolution, and, in general, kernel convolutions need points outside the image (or domain) to calculate points within the image.
- (ii) **Use wrapping.** One may use wrapping to “glue” the boundary edges of the simulation to one another. This turns our domain into a topological quotient space, and specific wrapping done will affect the topology of the simulation [83]. By using wrapping, it is possible to remove the imposed boundary; however, the new imposed topology does not necessarily represent the physical domain. However, this is useful for simulations on domains such as infinite crystal lattices [84].
- (iii) **Use an endpoint boundary method.** One may use a separate numerical scheme where the Laplacian is calculated not using a midpoint method but rather an endpoint method. Critically, however, one must ensure that linear system is diagonally dominant. Additionally, the difference between the interior scheme and the boundary schemes may cause reflections at the interface of the two separate methods.
- (iv) **Use a separate equation.** One may use a separate boundary equation such as a one-way wave equation to calculate the time update; however, the analytical solution would generally have reflections due at the interface; moreover, there will likely be reflections at the interface due to the difference in the numerical schemes as well.

(v) **Use padding.** One may pad the simulation before each application of the Laplacian; however, one must be careful in choosing a particular padding scheme. The padding method can be created using the boundary conditions given in the problem such as the standard Dirichlet, Neumann, or Robin boundary conditions; however, because the boundary that we consider is not physical, but, rather, it is imposed by the computational method, the aforementioned standard boundary conditions are insufficient. One popular method for padding the boundary is the use of absorbing boundary conditions (or artificial boundary conditions; both abbreviated ABCs). For any particular (two-way) wave equation, ABCs can be created from the associated Engquist-Majda one-way wave equations. Crucially, it should be noted that may ABCs pad the boundary after the time step has been calculated.

Absorbing Boundary Conditions. Currently, one of the most powerful methods to overcome the challenge of an unbounded physical domain is the use of the artificial boundary method which allows for simulation on truncated domains. The artificial boundary method truncates the unbounded domain and partitions the truncated domain into the following two parts: (1) the bounded interior domain which contains the region of physical interest, and (2) the bounded exterior domain which encompasses the interior domain. Suitable artificial boundary conditions (also called absorbing boundary conditions; both abbreviated as ABCs) are developed based on the properties of outgoing waves in the boundary region. Instead of the SE, the ABCs are imposed on the exterior domain and one may obtain a numerical solution to the computationally bounded problem (with ABCs) that is a good approximation to that of the original unbounded problem. Thus, this artificial boundary method allows for the computational domain to be finite while limiting the reflection of

waves at the boundary. Hence, accumulated error due to reflected waves can be mitigated even for extended simulation times.

There are many sophisticated ABCs proposed for the SE [19–28]. In particular, Zhang et al. [29, 30] used the operator splitting method to design the ABCs for the 1D and 2D SE on an unbounded domain. Notably, Antoine et al. used time-fractional operators with fractional order of $1/2$ to derive the artificial boundary condition to solve the 1D cubic nonlinear SE on unbounded domains [31]. Other approaches to design the ABCs for SE on unbounded domains can be found in [16, 32–45] and the references therein.

Many ABCs attempt to approximate the solution along the boundary using some assumed functional form. However, if the outgoing wave does not sufficiently satisfy the assumed form, there can be substantial reflection at the interface of the interior and boundary domains. Moreover, each for each simulation edge, one needs to develop a unique ABC. Therefore, one must decompose the domain into several pieces, which quickly becomes tedious for larger dimensions.

Fractional Momentum Layer. To overcome the issue of reflection caused by the interface of the interior and boundary domains, we transform the SE into its associated one-way wave equation (with transparent boundary conditions; abbreviated TBCs) by introducing an idealized variable-order fractional momentum operator, which allows the order of the fractional momentum to decrease *gradually* from the SE to the one-way wave equation through a fractional momentum layer (FML) as shown in Fig. 2.1 below. In so doing, we aim to smooth the abrupt transition. As such, through careful construction of the FML and variable-order fractional operators, the reflected portions of outgoing waves will decay rapidly. Thereby, the total accumulated error inside the physical region is reduced.

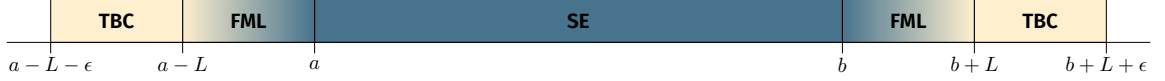


Figure 2.1: Diagram of 1D implementation of the FML. The SE is used in the domain of physical interest; the TBC is used on the computational boundaries; and the FML is sandwiched between the two.

2.3 Fractional-Order Derivatives and Approximations Thereof

So that the ideas behind the fractional momentum layer can be more easily understood, we will now discuss relevant concepts from fractional calculus. While there are a plethora of fractional derivatives, only the Caputo fractional derivative (and its numerical approximation) will be utilized in the latter portions of this manuscript. Even so, in this section, we will define several other derivatives to put our choice of the Caputo derivative into context.

Gamma Function. The gamma function allows for a natural extension of the factorial function to non-integer and complex numbers. The gamma function for positive integers is defined as $\Gamma(n) = (n - 1)!$ and, for positive real numbers, it can be expressed as

$$\Gamma(\alpha) = \int_0^{\infty} u^{\alpha-1} e^{-u} du. \quad (2.32)$$

It is well known that the gamma function over the complex plane is then defined as the analytic continuation of the equation above. The gamma function is holomorphic over the complex plane except on the set of non-positive integers $\{0\} \cup \mathbb{Z}^-$ where the function has simple poles. These poles are easy to see from Euler's reflection formula expressed as

$$\Gamma(z)\Gamma(1 - z) = \frac{\pi}{\sin(\pi z)}, \quad (2.33)$$

which allows for the easy evaluation of the gamma function at negative values. A plot of the gamma function is given in Fig. 2.2.

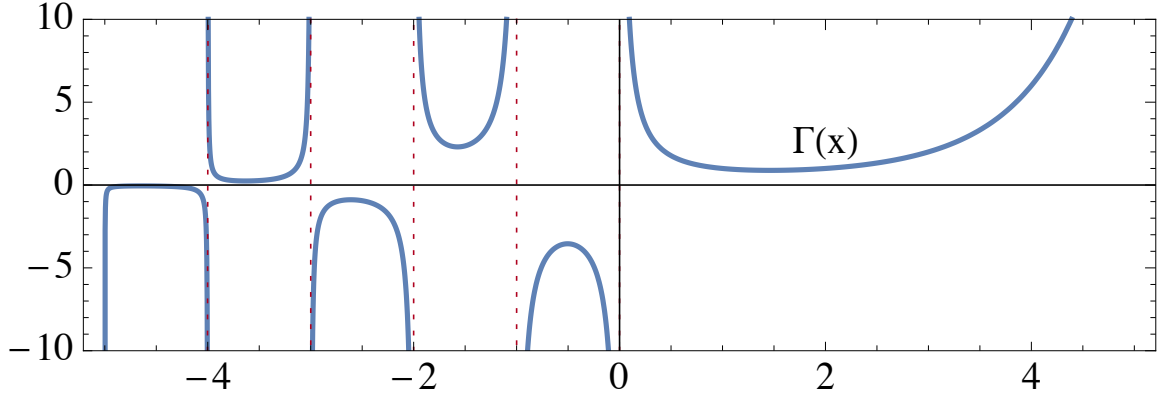


Figure 2.2: Plot of the gamma function $\Gamma(x)$.

2.3.1 Fractional-Order Derivatives

For this dissertation research, the computation will use what is known as the L2 approximation of the Caputo fractional-order derivative. However, we will take time to discuss several other fractional-order derivatives so that this choice may be contextualized later. The other derivatives of note are the Grünwald-Letnikov derivative, the Fourier derivative, the Liouville derivative, the Riemann derivative, and the Riesz derivative.

Grünwald-Letnikov Derivative. The Grünwald-Letnikov (G-L) derivative is one of the more easily understood approximations for a fractional-order derivative. It is a generalization of the limit definition of the derivative. Recall that

$$\frac{d}{dx}f(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}, \quad (2.34)$$

and

$$\frac{d^2}{dx^2}f(x) = \lim_{h \rightarrow 0} \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}. \quad (2.35)$$

Indeed, the forward difference definition of the n th order derivative of f is expressed as

$$\frac{d^n}{dx^n}f(x) = \lim_{h \rightarrow 0} h^{-n} \sum_{k=0}^n (-1)^k \binom{n}{k} f(x + (n-k)h). \quad (2.36)$$

Now, to extend this into the fractional-order case we let $n \rightarrow \alpha$ which yields

$$\frac{d^\alpha}{dx^\alpha} f(x) = \lim_{h \rightarrow 0} h^{-\alpha} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(x + (\alpha - k)h), \quad (2.37)$$

where, for non-integer α , the binomial coefficient is defined via the gamma function as

$\binom{\alpha}{k} = \frac{\Gamma(1+\alpha)}{\Gamma(1+k)\Gamma(1+\alpha-k)}$. The above equation can be expanded about αh to obtain the G-L derivative as

$$\frac{d^\alpha}{dx^\alpha} f(x) = \lim_{h \rightarrow 0} h^{-\alpha} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(x - hk) + O(h^{1-\alpha}). \quad (2.38)$$

For $0 \leq \alpha < 1$, the error term vanishes in the limit leaving the G-L derivative as

$$\frac{d^\alpha}{dx^\alpha} f(x) = \lim_{h \rightarrow 0} h^{-\alpha} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(x - hk). \quad (2.39)$$

Notice that for any particular evaluation point, x , the G-L derivative cannot be expressed exactly using only points in a small neighborhood about x . For computation, one may truncate the sum whenever the coefficient $h^{-\alpha} \binom{\alpha}{k}$ is within an error tolerance. For example, $|\binom{\alpha}{k}| < \sigma h$ may be a good choice for some σ such that $0 < \sigma \ll 1$. Using the reflection formula given in Eq. (2.33)

$$\begin{aligned} \binom{\alpha}{k} &= \frac{\Gamma(1+\alpha)}{\Gamma(1+k)\Gamma(1+\alpha-k)} = \frac{\Gamma(1+\alpha)\Gamma(k-\alpha)}{\pi\Gamma(1+k)} \sin(\pi(k-\alpha)) \\ &= -(-1)^k \frac{\Gamma(1+\alpha)\Gamma(k-\alpha)}{\pi\Gamma(1+k)} \sin(\pi\alpha). \end{aligned} \quad (2.40)$$

Hence, we must choose k so that $\left| \frac{\Gamma(1+\alpha)\Gamma(k-\alpha)}{\pi\Gamma(1+k)} \sin(\pi\alpha) \right| < \sigma h$, or equivalently,

$$\frac{\Gamma(k-\alpha)}{\Gamma(1+k)} < \frac{\sigma\pi h}{\Gamma(1+\alpha)|\sin(\pi\alpha)|}. \quad (2.41)$$

Using Stirling's approximation, bounds for the gamma function are given as $\sqrt{2\pi n} n^n e^{-n} <$

$\Gamma(1+n) < n^{n+1/2} e^{1-n}$, and it follows that for $k > 1 + \alpha$

$$\frac{\Gamma(k-\alpha)}{\Gamma(1+k)} \geq \frac{\sqrt{k-\alpha-1}(k-\alpha-1)^{k-\alpha-1} e^{-k+\alpha+1}}{\sqrt{2\pi k} k^k e^{-k}}$$

$$\begin{aligned}
&= \frac{e^{1+\alpha}}{\sqrt{2\pi}} \left(1 - \frac{1+\alpha}{k}\right)^{k+1/2} (k - \alpha - 1)^{-1-\alpha} \\
&\geq \frac{e^{1+\alpha}}{\sqrt{2\pi}} (k - \alpha - 1)^{-1-\alpha}
\end{aligned} \tag{2.42}$$

Hence, we may choose to truncate the sum in the G-L derivative at the term N such that

$$\frac{e^{1+\alpha}}{\sqrt{2\pi}} (N - \alpha - 1)^{-1-\alpha} < \frac{\sigma\pi h}{\Gamma(1+\alpha)|\sin(\pi\alpha)|}. \tag{2.43}$$

Isolating N , we find

$$N > 1 + \alpha + e \left(\frac{\Gamma(1+\alpha)|\sin(\pi\alpha)|}{\sigma h \sqrt{2\pi^3}} \right)^{\frac{1}{1+\alpha}}. \tag{2.44}$$

Fourier Derivative. The Fourier derivative is another easily understood fractional order derivatives. It is derived by generalizing the properties of the integer-order derivatives under the Fourier transform. That is, let $f(x)$ be a function and let $\hat{f}(k)$ be the Fourier transform of f defined by

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx, \tag{2.45}$$

then we may represent f as the inverse Fourier transform of \hat{f} defined as

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dk. \tag{2.46}$$

Then, the Fourier representation of the first derivative of f is given by

$$\frac{d}{dx} f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} ik \hat{f}(k) e^{ikx} dk. \tag{2.47}$$

Indeed, if n is an integer, then the n th derivative of f has a Fourier representation as

$$\frac{d^n}{dx^n} f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (ik)^n \hat{f}(k) e^{ikx} dk. \tag{2.48}$$

Now, let α be a real constant, and allow $n \rightarrow \alpha$, then we define the fractional-order operator

$$\frac{d^\alpha}{dx^\alpha} f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (ik)^\alpha \hat{f}(k) e^{ikx} dk. \tag{2.49}$$

Liouville Derivative. The Liouville derivatives are the first-order derivative of the Liouville integrals and are given by

$${}_{-\infty}^{\text{L}}\text{D}_x^{\alpha+n}f(x) = \frac{\text{d}}{\text{d}x} \frac{1}{\Gamma(1-\alpha)} \int_{-\infty}^x \frac{f(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.50a)$$

$${}_x^{\text{L}}\text{D}_\infty^{\alpha+n}f(x) = \frac{\text{d}}{\text{d}x} \frac{1}{\Gamma(1-\alpha)} \int_x^{-\infty} \frac{f(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.50b)$$

where $0 < \alpha < 1$.

Riemann Derivative. The Riemann derivatives are the first-order derivative of the Liouville integrals and are given by

$${}_0^{\text{R}}\text{D}_x^\alpha f(x) = \frac{\text{d}}{\text{d}x} \frac{1}{\Gamma(1-\alpha)} \int_0^x \frac{f(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.51a)$$

$${}_x^{\text{R}}\text{D}_0^\alpha f(x) = \frac{\text{d}}{\text{d}x} \frac{1}{\Gamma(1-\alpha)} \int_x^0 \frac{f(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.51b)$$

where $0 < \alpha < 1$. To define higher-order fractional derivatives in both the Liouville and Riemann sense, techniques introduced by Caputo or Riesz are needed.

Caputo Derivative. The Liouville-Caputo derivative, herein referred to only as the Caputo derivative, is defined for all fractional orders $\alpha + n$ where $0 \leq \alpha \leq 1$ and $n \in \mathbb{N}$. The left- and right-handed Caputo derivatives are given by

$${}_{-\infty}^{\text{C}}\text{D}_x^{\alpha+n}f(x) = \frac{1}{\Gamma(1-\alpha)} \int_{-\infty}^x \frac{f^{(n+1)}(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.52a)$$

$${}_x^{\text{C}}\text{D}_\infty^{\alpha+n}f(x) = \frac{1}{\Gamma(1-\alpha)} \int_x^\infty \frac{f^{(n+1)}(\xi)}{(x-\xi)^\alpha} \text{d}\xi \quad (2.52b)$$

Riesz Derivative. The Riesz derivative uses both the left and right-handed Liouville derivatives to obtain an operator which approaches the second-order derivative. The Riesz derivative is given as follows:

$${}^{\text{RZ}}\text{D}_x^{2\alpha} = -\frac{{}_{-\infty}^{\text{L}}\text{D}_x^\alpha + {}_x^{\text{L}}\text{D}_\infty^\alpha}{2 \cos(\pi\alpha/2)}. \quad (2.53)$$

It should be pointed out that as $\alpha \rightarrow 1$ we have ${}^{\text{RZ}}D_x^{2\alpha} \rightarrow \partial_x^2$; however, a similar result does not hold true for $\alpha \rightarrow 1/2$. Explicitly, as $\alpha \rightarrow 1/2$, ${}^{\text{RZ}}D_x^{2\alpha} \not\rightarrow \partial_x$.

2.3.2 Lagrange Approximations

The Lagrange approximations provide accurate approximations of the fractional derivative for fractional orders within interval $[0, 2]$. In particular, L1 approximation is used for fractional orders within the interval $[0, 1]$, and the L2 is used for fractional orders within the interval $[1, 2]$.

L1 Approximation. The L1 approximation of the Caputo derivative uses the Lagrange polynomial of order 1 to approximate the function f over an small interval of size h [85]. This method is reminiscent of compound trapezoidal rule for calculating an integral. However, rather than integrating the Lagrange polynomial over each interval, one may approximate the first-order derivative as a constant over each interval; therefore, only a fractional kernel is left inside the integrand. The fractional kernel is of the form $\int_a^b (\xi - x)^{-\alpha} d\xi$. For, $0 < \alpha < 1$ and $x \leq a \leq b$, then the following result follows directly from the anti-derivative:

$$\int_a^b (\xi - x)^{-\alpha} d\xi = \frac{(b - x)^{1-\alpha} - (a - x)^{1-\alpha}}{1 - \alpha}. \quad (2.54)$$

Now, we will begin to develop an approximation for the function f over a small interval $[x_j, x_{j+1}]$ where $x_j = jh$. The first-order Lagrange approximation of the function f is given by the polynomial

$$\begin{aligned} P(x) &= \frac{x - x_j}{x_{j+1} - x_j} f_{j+1} + \frac{x - x_{j+1}}{x_j - x_{j+1}} f_j \\ &= \frac{x - x_j}{h} f_{j+1} - \frac{x - x_{j+1}}{h} f_j = \frac{f_{j+1} - f_j}{h} x + \frac{x_{j+1} + x_j}{h} f_j \end{aligned} \quad (2.55)$$

where $f_j = f(x_j)$. Since $P(x)$ is a first-order polynomial, $P'(x) = \frac{f_{j+1} - f_j}{h}$ is constant; hence, first-order derivative inside the integral of the Caputo derivative can be approximated as a constant over a small interval.

To determine the error, from this approximation, let $R_j(x) = f(x) - P(x)$ be the error (or remainder) term. Then, $R_j(x_j) = R_j(x_{j+1}) = 0$, and $R_j''(x) = f''(x)$. Define a function new function g such that

$$g(u) := R_j(u) - \frac{(u - x_j)(u - x_{j+1})}{(x - x_j)(x - x_{j+1})} R_j(x). \quad (2.56)$$

Then, $g(x_j) = g(x_{j+1}) = g(x) = 0$, implying that, on the interval $[x_j, x_{j+1}]$, $g(u)$ has at least 3 zeros in the interval. By Rolle's theorem, $g'(u)$ has at least 2 zeros on the interval (x_j, x_{j+1}) , and $g''(u)$ has at least one zero on the interval. For any particular value of $x \in (x_j, x_{j+1})$, let $v_j \in (x_j, x_{j+1})$ be such that $g''(v_j) = 0$, then

$$0 = g''(v_j) = f''(v_j) - \frac{2}{(x - x_j)(x - x_{j+1})} R_j(x), \quad (2.57)$$

therefore, for some value of v_j in the interval

$$R_j(x) = \frac{1}{2}(x - x_j)(x - x_{j+1})f''(v_j) \quad (2.58)$$

Now, we consider the magnitude of the error over the interval.

$$\begin{aligned} |R_j(x)| &= \frac{1}{2} |(x - x_j)(x - x_{j+1})f''(v_j)| \\ &\leq \frac{1}{2} \left| \left(\frac{x_{j+1} + x_j}{2} - x_j \right) \left(\frac{x_{j+1} + x_j}{2} - x_{j+1} \right) f''(v_j) \right| \\ &\leq \frac{1}{8} (x_{j+1} - x_j)^2 |f''(v_j)| \\ &= \frac{1}{8} h^2 |f''(v_j)| \end{aligned} \quad (2.59)$$

Therefore, the magnitude of the error due to this approximation is $O(h^2)$. In [85], Sun *et al.* derive an expression for the integral over some finite interval. We will take a similar

approach, but we will instead derive an approximation for the infinite interval, and then truncate the infinite sum later. This difference in approach will lead to slightly different numerical approximations. Let $h > 0$ be small, then $x_j = jh$ and $x_j - x_k = x_{j-k}$ for all $k \in \mathbb{Z}$. We will now replace the single integral in the Caputo derivative by the sum of integrals over the small intervals. Letting $f(x_j) = f_j$, $a_0^\alpha = 1$, and $a_k^\alpha = (k+1)^{1-\alpha} - k^{1-\alpha}$ for $k \geq 1$, we obtain the approximation for the Caputo derivative that follows:

$$\begin{aligned}
{}_{-\infty}^{\text{C}}\mathbf{D}_{x_j}^\alpha f(x_j) &= \frac{1}{\Gamma(1-\alpha)} \int_{-\infty}^{x_j} \frac{f'(\xi)}{(x_j - \xi)^\alpha} d\xi \\
&= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{\infty} \int_{x_{j-k-1}}^{x_{j-k}} \frac{f'(\xi)}{(x_j - \xi)^\alpha} d\xi \\
&= \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \frac{f_{j-k} - f_{j-k-1}}{h} \int_{x_{j-k-1}}^{x_{j-k}} \frac{1}{(x_j - \xi)^\alpha} d\xi \\
&\quad + \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{\infty} \int_{x_{j-k-1}}^{x_{j-k}} \frac{R'_{j-k-1}(\xi)}{(x_j - \xi)^\alpha} d\xi \\
&= \frac{-1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \frac{f_{j-k} - f_{j-k-1}}{h} [x_k^{1-\alpha} - x_{k+1}^{1-\alpha}] + O(h^{2-\alpha}) \\
&\approx \frac{-1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^{\infty} (f_{j-k} - f_{j-k-1}) [(k)^{1-\alpha} - (k+1)^{1-\alpha}] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^{\infty} \tilde{a}_k^{1-\alpha} (f_{j-k} - f_{j-k-1}) \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[a_0^\alpha f_j + \sum_{k=1}^{\infty} \tilde{a}_k^\alpha f_{j-k} - \sum_{k=0}^{\infty} \tilde{a}_k f_{j-k-1} \right] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[f_j + \sum_{k=1}^{\infty} \tilde{a}_k^\alpha f_{j-k} - \sum_{k=1}^{\infty} \tilde{a}_{k-1}^\alpha f_{j-k} \right] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[f_j + \sum_{k=1}^{\infty} (\tilde{a}_k^\alpha - \tilde{a}_{k-1}^\alpha) f_{j-k} \right] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^{\infty} \tilde{b}_k^\alpha f_{j-k}, \tag{2.60}
\end{aligned}$$

where $\tilde{b}_0^\alpha = 1$, $\tilde{b}_1^\alpha = 2^{1-\alpha} - 2$, and $\tilde{b}_k^\alpha = (k+1)^{1-\alpha} - 2k^{1-\alpha} + (k-1)^{1-\alpha}$ for $k \geq 2$.

Similarly, the forward Caputo fractional derivative is given by

$$\begin{aligned}
{}^C D_{x_j}^\alpha f(x_j) &= \frac{1}{\Gamma(1-\alpha)} \int_{x_j}^\infty \frac{f'(\xi)}{(\xi - x_j)^\alpha} d\xi \\
&= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^\infty \int_{x_j+x_k}^{x_j+x_{k+1}} \frac{f'(\xi)}{(\xi - x_j)^\alpha} d\xi \\
&= \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^\infty \frac{f_{j+k+1} - f_{j+k}}{h} \int_{x_j+k}^{x_j+k+1} \frac{1}{(x_j - \xi)^\alpha} d\xi \\
&\quad + \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^\infty \int_{x_j+k}^{x_j+k+1} \frac{R'_{j+k}(\xi)}{(x_j - \xi)^\alpha} d\xi \\
&= \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^\infty \frac{f_{j+k+1} - f_{j+k}}{h} [x_{k+1}^{1-\alpha} - x_k^{1-\alpha}] + O(h^{2-\alpha}) \\
&\approx \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^\infty (f_{j+k+1} - f_{j+k}) [(k+1)^{1-\alpha} - k^{1-\alpha}] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^\infty \tilde{a}_k^{1-\alpha} (f_{j+k+1} - f_{j+k}) \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[-\tilde{a}_0^\alpha f_j + \sum_{k=0}^\infty \tilde{a}_k^\alpha f_{j+k+1} - \sum_{k=1}^\infty \tilde{a}_k^\alpha f_{j+k} \right] \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[-f_j + \sum_{k=1}^\infty \tilde{a}_{k-1}^\alpha f_{j+k} - \sum_{k=1}^\infty \tilde{a}_k^\alpha f_{j+k} \right] \\
&= \frac{-1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \left[f_j + \sum_{k=1}^\infty (\tilde{a}_k^\alpha - \tilde{a}_{k-1}^\alpha) f_{j+k} \right] \\
&= \frac{-1}{\Gamma(2-\alpha)} \frac{1}{h^\alpha} \sum_{k=0}^\infty \tilde{b}_k^\alpha f_{j-k}. \tag{2.61}
\end{aligned}$$

From this, we define the finite-difference operators

$$\nabla_{\bar{x}}^\alpha f_j := \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^\infty \tilde{b}_k^\alpha f_{j-k} \tag{2.62a}$$

$$\nabla_x^\alpha f_j := \frac{-1}{\Gamma(2-\alpha)} \sum_{k=0}^\infty \tilde{b}_k^\alpha f_{j+k}. \tag{2.62b}$$

Recall that $\tilde{b}_0^\alpha = 1$, $\tilde{b}_1^\alpha = 2^{1-\alpha} - 2$, and $\tilde{b}_k^\alpha = (k+1)^{1-\alpha} - 2k^{1-\alpha} + (k-1)^{1-\alpha}$ for $k \geq 2$,

and notice that for $k \geq 1$ we have $\lim_{\alpha \rightarrow 0} \tilde{b}_k^\alpha = 0$. Additionally, $\lim_{\alpha \rightarrow 1} \tilde{b}_1^\alpha = -1$, and, for

$k \geq 2$, $\lim_{\alpha \rightarrow 1} \tilde{b}_k^\alpha = 0$. Therefore, $\lim_{\alpha \rightarrow 1} \nabla_{\bar{x}}^\alpha f_j = \nabla_{\bar{x}} f_j = f_j - f_{j-1}$ and $\lim_{\alpha \rightarrow 1} \nabla_x^\alpha f_j = \nabla_x f_j = f_{j+1} - f_j$. Also, $\lim_{\alpha \rightarrow 0} \nabla_{\bar{x}}^\alpha f_j = f_j$ and $\lim_{\alpha \rightarrow 0} \nabla_x^\alpha f_j = -f_j$. With the above definitions, the L_1 finite difference approximations may be compactly written as follows:

$${}_{-\infty}^C D_{x_j}^\alpha f(x_j) = \frac{1}{h^\alpha} \nabla_{\bar{x}}^\alpha f(x_j) + O(h^{2-\alpha}), \quad (2.63a)$$

$${}_{x_j}^C D_\infty^\alpha f(x_j) = \frac{1}{h^\alpha} \nabla_x^\alpha f(x_j) + O(h^{2-\alpha}), \quad (2.63b)$$

The error of $O(h^{2-\alpha})$ for Eq. (2.63), was shown by Sun *et al.* in [86].

L2 Approximation. The L2 approximation is used to calculate the Caputo derivative for orders within the interval $[1,2]$. Let $x_j = jh$, then, on the interval $[x_{j-1}, x_{j+1}]$, the first-order Lagrange approximation of the function f is given by the polynomial

$$P(x) = \frac{x - x_j}{x_{j+1} - x_j} \frac{x - x_{j-1}}{x_{j+1} - x_{j-1}} f_{j+1} + \frac{x - x_{j+1}}{x_j - x_{j+1}} \frac{x - x_{j-1}}{x_j - x_{j-1}} f_j + \frac{x - x_j}{x_{j-1} - x_j} \frac{x - x_{j+1}}{x_{j-1} - x_{j+1}} f_{j-1} \quad (2.64)$$

which has the second derivative $P''(x) = \frac{1}{h^2}(f_{j+1} - 2f_j + f_{j-1})$. Following our procedure

for the L1 approximation, we have

$$\begin{aligned} {}_{-\infty}^C D_{x_j}^{\alpha+1} f(x_j) &= \frac{1}{\Gamma(1-\alpha)} \int_{-\infty}^{x_j} \frac{f''(\xi)}{(x_j - \xi)^\alpha} d\xi \\ &= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{\infty} \int_{x_j - x_{k+1}}^{x_j - x_k} \frac{f''(\xi)}{(x_j - \xi)^\alpha} d\xi \\ &= \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \frac{f_{j-k+1} - 2f_{j-k} + f_{j-k-1}}{h^2} \int_{x_j - x_{k+1}}^{x_j - x_k} \frac{1}{(x_j - \xi)^\alpha} d\xi \\ &\quad + \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \int_{x_j - x_{k+1}}^{x_j - x_k} \frac{R_{j-k-1}''(\xi)}{(x_j - \xi)^\alpha} d\xi \\ &= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{1+\alpha}} \sum_{k=0}^{\infty} \tilde{a}_k \frac{f_{j-k+1} - 2f_{j-k} + f_{j-k-1}}{h^2} + O(h^{3-\alpha}) \\ &\approx \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^\alpha (f_{j-k+1} - 2f_{j-k} + f_{j-k-1}) \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} (f_{j-k+1} - f_{j-k}) \\
&\quad - \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} (f_{j-k} - f_{j-k-1}) \\
&= \frac{1}{h^{1+\alpha}} (\nabla_{\bar{x}}^{\alpha} f_{j+1} - \nabla_{\bar{x}}^{\alpha} f_j) \\
&= \frac{1}{h^{\alpha+1}} \nabla_{\bar{x}}^{\alpha} (f_{j+1} - f_j) \\
&= \frac{1}{h^{\alpha+1}} \nabla_{\bar{x}}^{\alpha} \nabla_x f_j
\end{aligned} \tag{2.65}$$

Similarly, for the right-handed Caputo derivative, we have

$$\begin{aligned}
{}^C_{x_j} \mathbf{D}_{\infty}^{\alpha+1} f(x_j) &= \frac{1}{\Gamma(1-\alpha)} \int_{x_j}^{\infty} \frac{f''(\xi)}{(\xi - x_j)^{\alpha}} d\xi \\
&= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{\infty} \int_{x_j+x_k}^{x_j+x_{k+1}} \frac{f''(\xi)}{(\xi - x)^{\alpha}} d\xi \\
&= \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \frac{f_{j-k+1} - 2f_{j-k} + f_{j-k-1}}{h^2} \int_{x_j+k}^{x_j+k+1} \frac{1}{(\xi - x_j)^{\alpha}} d\xi \\
&\quad + \frac{1}{\Gamma(2-\alpha)} \sum_{k=0}^{\infty} \int_{x_j+k}^{x_j+k+1} \frac{R''_{j+k}(\xi)}{(x_j - \xi)^{\alpha}} d\xi \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} \frac{f_{j+k+1} - 2f_{j+k} + f_{j+k-1}}{h^2} + O(h^{3-\alpha}) \\
&\approx \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} (f_{j+k+1} - 2f_{j+k} + f_{j+k-1}) \\
&= \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{\alpha+1}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} (f_{j+k+1} - f_{j+k}) \\
&\quad - \frac{1}{\Gamma(2-\alpha)} \frac{1}{h^{1+\alpha}} \sum_{k=0}^{\infty} \tilde{a}_k^{\alpha} (f_{j+k} - f_{j+k-1}) \\
&= \frac{1}{h^{1+\alpha}} (\nabla_x^{\alpha} f_{j+1} - \nabla_x^{\alpha} f_j) \\
&= \frac{1}{h^{\alpha+1}} \nabla_x^{\alpha} (f_{j+1} - f_j) \\
&= \frac{1}{h^{\alpha+1}} \nabla_x^{\alpha} \nabla_{\bar{x}} f_j
\end{aligned} \tag{2.66}$$

Now we define the new finite-difference operators for $1 \leq \alpha \leq 2$ as follows:

$$\nabla_{\bar{x}x}^\alpha = \nabla_{\bar{x}}^{\alpha-1} \nabla_x, \quad \text{and} \quad \nabla_{x\bar{x}}^\alpha = \nabla_x^{\alpha-1} \nabla_{\bar{x}} \quad (2.67)$$

Notice these limiting cases for operators defined in Eq. (2.67),

$$\lim_{\alpha \rightarrow 2} \nabla_{\bar{x}x}^\alpha = \delta_x^2, \quad \text{and} \quad \lim_{\alpha \rightarrow 1} \nabla_{\bar{x}x}^\alpha = +\nabla_x, \quad (2.68a)$$

$$\lim_{\alpha \rightarrow 2} \nabla_{x\bar{x}}^\alpha = \delta_x^2, \quad \text{and} \quad \lim_{\alpha \rightarrow 1} \nabla_{x\bar{x}}^\alpha = -\nabla_{\bar{x}}, \quad (2.68b)$$

where $\delta_x^2 f_j = f_{j+1} - 2f_j + f_{j-1}$ is the second-order central difference operator. Importantly,

$$\lim_{\alpha \rightarrow 2} \frac{\nabla_{\bar{x}x}^\alpha + \nabla_{x\bar{x}}^\alpha}{2} = \delta_x^2. \quad (2.69)$$

Interestingly, we also have the result

$$\lim_{\alpha \rightarrow 1} \nabla_{\bar{x}x}^\alpha + \nabla_{x\bar{x}}^\alpha = \nabla_x - \nabla_{\bar{x}} = \delta_x^2. \quad (2.70)$$

Non-Locality and Computational Complexity. The non-locality of the fractional derivative means that the L1 and L2 approximations are fairly computationally expensive. However, there is a new paradigm of using graphical processing units (GPUs) for scientific applications. Utilizing GPUs to compute these derivatives (in bulk) offers a speedup in performance over CPU calculations. Moreover, finite difference methods, in general, have seen great speedups due to the development of GPU computing.

2.4 GPU Computing

The use of GPUs for scientific computing is rising because they allow for parallel computing using many computational cores as compared with a central processing unit (CPU). Exemplified by the fact that GPU clock-speeds are reported in MHz while CPU clock-speeds are reported in GHz, CPU clock-speeds are significantly faster than GPU clock-speeds;

however, GPUs have many, *many* more computational cores than CPUs. Hence, GPUs can perform many simultaneous parallel computations in bulk which can reduce the total amount of computation time [41, 87, 88]. Since explicit numerical methods for solving SEs are inherently parallel, it is convenient to utilize GPU computing to greatly decrease computation time. The decreased computation times allow for more rapid testing during the development of numerical schemes, as well as more rapid simulation in general once a scheme has been finalized.

GPU/CPU Comparisons. It is notoriously difficult to fairly compare the speedup that a GPU will provide over a CPU. While there have been many articles claiming GPU-speedups on the order of 10x-1000x, these reports have been shown to be inflated for various reasons. In 2010, Lee et al. showed that the average GPU-speedup was 2.5x for certain algorithms by taking into account factors such as using contemporary processors, overhead, and optimizations [87]. Nevertheless, GPU-enabled scientific computing has had a strong impact on the field of computational physics [89]. In a more modern analysis, Buber et al. showed that, for machine learning applications, the speedups up to 5x could be obtained [88]. Fortunately, in 2018, Chen *et al.* showed that certain machine learning neural network models (in particular, convolutional neural networks) are analogous to explicit methods for solving differential equations [90]. Therefore, the GPU speedups obtained through the data parallelism of neural networks can be expected in kind for explicit finite difference methods.

2.5 Chapter Summary

In this section, we have discussed the Schrödinger equation, the G-FDTD method for solving the Schrödinger equation, as well as several existing boundary methods. In particular,

we discussed absorbing boundary conditions, and we laid the framework for the fractional momentum layer by introducing the concept of fractional derivatives. Finally, we have discussed the advantages of GPU computing for scientific applications, specifically the application of explicit finite difference methods.

CHAPTER 3

ABSORBING BOUNDARY CONDITIONS

In this chapter, explicit absorbing boundary conditions (ABCs) are presented for the recently developed Generalized Finite-Difference Time-Domain (G-FDTD) method for solving the nonlinear Schrödinger equation so that the method can be used on unbounded domains when the analytical solution along the boundary is unknown. The ABC scheme results from the Box-like discretization of Engquist-Majda one-way wave equations. Using the energy-weighted wave-number parameter selection method, the ABCs are made to be adaptive. By simulating solitons onto the computational boundary, the reflection coefficient is numerically shown to be a function of the incoming soliton's wavenumber and other simulation parameters. Furthermore, a parallelized algorithm is developed for implementing the G-FDTD method with ABCs. The algorithm, when implemented on a GPU, is shown to give up to a 200-times speedup for large simulations as compared with using a CPU. Examples are given to show the applicability of the algorithm¹.

3.1 One-Way Wave Equations

The derivation for the ABCs begins similarly to the explicit methods in [25, 27]. Additionally, the scheme is similar to the Mur ABC [33, 39] and relies on the discretization techniques from the Box scheme. We choose to use the Box scheme techniques over a

¹The derivation in this chapter has been published in the journal *Computer Physics Communications* [44].

more naïve approach to ensure that, in the 1D case, the ABCs are defined for all input wavenumbers k , especially zero, and, in the 2D case, the east/west and north/south ABCs are defined regardless of the angle subtended by the input wave vector \mathbf{k} and the x -axis. It should be noted that while our derivation technique is different from the one used by Zhang *et al.*, our continuous boundary conditions are equivalent to the ones of second-order developed in [29, 30]. The main difference lies in the discrete boundary scheme. Since Zhang *et al.* developed their scheme to be incorporated with an implicit linearized Crank-Nicolson interior scheme, the boundary scheme they developed is also implicit. We develop our ABCs to be explicit since they are to be incorporated with the G-FDTD method which is explicit. Another difference lies in the near-boundary treatment since the interior scheme used by Zhang *et al.* only needs to approximate two grid points of the boundary, whereas the G-FDTD scheme requires $(4M + 2)$ grid points where M is from the expressions for the 1D and 2D G-FDTD methods given by Eqs. (2.27) and Eq. (2.30), respectively. Hence, we need a method to approximate the solutions at the other $4M$ points.

3.2 One-Way Wave Equations for 1D Case

To develop the 1D case, we first construct an Engquist-Majda (EM) one-way wave equation [32, 34]. Recall that the 1D NSE ² is given by

$$i\partial_t\psi - \partial_x^2\psi + \lambda|\psi|^2\psi = 0. \quad (3.1)$$

²Indeed, the NSE presented here is slightly different than the one originally stated in Eq. (2.2) as sign on the kinetic energy term ∂_x^2 is reversed. The NSE presented here is the NSE that is used in the original G-FDTD paper [18]. This sign reversal is equivalent to the transformation composite transformation $t \rightarrow -t$ and $\lambda \rightarrow -\lambda$. In any case, this does not significantly affect the derivation.

Assuming that, on the boundary, the wave packet is approximately an outgoing sinusoidal wave of the following form:

$$\psi(x, t) = \psi_0(t)e^{\pm ik_x x}. \quad (3.2)$$

Using the above ansatz, the canonical momentum along the x -direction is given by

$$\hat{p}_x \psi = -i\partial_x \psi = \pm k_x \psi. \quad (3.3)$$

Recasting the above equation as a single operator, we find $(i\partial_x \pm k_x)\psi = 0$. Generally, wave functions will have more than one wavenumber [25]; hence, for a wave function with two wavenumbers κ_1 and κ_2 that are aligned ($\kappa_1 \kappa_2 > 0$), we have

$$(i\partial_x \pm \kappa_1)(i\partial_x \pm \kappa_2)\psi = 0. \quad (3.4)$$

If the two wavenumbers are different for a particular outgoing wave, then, by parameterizing for those two wavenumbers can allow each wave component to be absorbed; however, if the outgoing waves is monochromatic, then one may set $\kappa_1 = \kappa_2$ and the wave can be absorbed to second order [27]. Expanding out the operator product in Eq. (3.4), we find

$$-\partial_x^2 \psi \pm ic_x \partial_x \psi + V_x \psi = 0. \quad (3.5)$$

where $c_x = |\kappa_1 + \kappa_2|$ and $V_x = \kappa_1 \kappa_2$. Here, c_x is related to the group velocity of the wave, and V_x is an effective potential. Solving Eq. (3.5) for $\partial_x^2 \psi$ and substituting the result into the NSE in Eq. (3.1) yields the EM one-way wave equation as

$$\partial_t \psi \mp c_x \partial_x \psi = \lambda |\psi|^{p-1} \psi - V_x \psi. \quad (3.6)$$

It should be pointed out that Eq. (3.6) is equivalent to the second-order continuous boundary conditions developed in [29]. The EM one-way wave equation can be easily extended to the 2D case.

3.3 One-Way Wave Equations for 2D Case

To determine the EM one-way wave equations for the 2D case, recall that the 2D NSE³ is given by

$$i\partial_t\psi - \partial_x^2\psi - \partial_y^2\psi + \lambda|\psi|^{p-1}\psi = 0. \quad (3.7)$$

It is important to note that when obtaining the one-way wave equations, only the portions of the Laplacian that represent coordinates normal to the boundary need to be substituted for using equations similar to Eq. (3.5) on page 35. To develop the one-way wave equations for the x -direction, we substitute Eq. (3.5) into Eq. (3.7) to find

$$(\partial_t \mp c_x \partial_x)\psi = i \left(\lambda|\psi|^{p-1} - V_x \right) \psi - i\partial_y^2\psi, \quad (3.8)$$

which is the 2D analog of equation Eq. (3.6). It should be pointed out that Eq. (3.8) is equivalent to the second-order continuous boundary conditions developed in [30]. Now, switching the roles of x and y in the above equation, we obtain

$$(\partial_t \mp c_y \partial_y)\psi = i \left(\lambda|\psi|^{p-1} - V_y \right) \psi - i\partial_x^2\psi. \quad (3.9)$$

For the 1D case, it should be pointed out that the G-FDTD scheme in Eq. (2.27) requires information about the value of ψ at $(4M + 2)$ many points on both the left- and right-side boundaries of the computational domain. These values must be known before using the G-FDTD scheme. Similarly, for the 2-D case implemented on an $J \times K$ grid, Eq. (2.30)

³As in the previous section, the NSE presented here is the one from the original paper that develops the G-FDTD method [18].

requires information about the value of ψ at $J \cdot (4M + 2)$ many points on both the north and south boundaries, and $K \cdot (4M + 2)$ many points on both the east and west boundaries. This requires developing absorbing boundary conditions for providing the values at those boundary points if the analytical solution is unknown. In the present study, we choose $M = 1$ in Eq. (2.27) and Eq. (2.30) so that the truncation error of the G-FDTD scheme is $O(\Delta t^4 + \Delta x^4)$, and, in the next section, we will develop absorbing boundary conditions for this case.

3.4 Absorbing Boundary Conditions

Using the EM one-way wave equations from the previous section, we will develop discrete absorbing boundary conditions for the 1D and 2D cases, respectively. Our derivation is similar to the one presented by Cole *et al.* in [39].

3.4.1 1D Case for Absorbing Boundary Conditions

We will now derive the absorbing boundary conditions for the 1D case by proceeding in a similar fashion to that of the ABCs developed in [39]. Let $t_n = n\Delta t$, $x_j = j\Delta x$, and define the operators d_x and A_x such that $d_x f(x) = f(x + \Delta x/2) - f(x - \Delta x/2)$ and $A_x f(x) = f(x + \Delta x/2) + f(x - \Delta x/2)$. Then, $f'(x) \approx \frac{1}{\Delta x} d_x f(x)$, $f'(t) \approx \frac{1}{\Delta t} d_t f(t)$, and $f(x) \approx \frac{1}{2} A_x f(x)$. Hence, we have the following discrete absorbing boundary condition

$$\left(d_t \mp \frac{c_x \Delta t}{\Delta x} d_x \right) \psi(x, t) = i \frac{\Delta t}{2} \left(\frac{\lambda}{2^{p-1}} |A_x \psi(x, t)|^{p-1} - V_x \right) \cdot A_x \psi(x, t). \quad (3.10)$$

To ease notation, let b be an index such that the point x_b lies inside the boundary region. And let $a = b \pm 1$ be the index such that the point x_a is just inside the interior region. Now, let m be the index such that $x_m = \frac{x_b + x_a}{2}$, then, $\psi_m \approx \frac{\psi_b + \psi_a}{2} = \frac{1}{2} A_x \psi_m$. Additionally, note that $\psi^{n+1/2} \approx \frac{\psi^{n+1} + \psi^n}{2}$. If we evaluate Eq. (3.10) at the point $(x_m, t_{n+1/2})$ and let $\tilde{c}_x = \frac{c_x \Delta t}{\Delta x}$,

we have the following:

$$(d_t A_x \psi_m^{n+1/2} \mp \tilde{c}_x A_t d_x \psi_m^{n+1/2}) = i \Delta t \left(\frac{\lambda}{2^{p-1}} |A_x \psi_m^{n+1/2}|^{p-1} - V_x \right) \cdot A_x \psi_m^{n+1/2}. \quad (3.11)$$

Expanding the operators on the left-hand side of the above equation, we obtain the following

$$\begin{aligned} (\psi_b^{n+1} + \psi_a^{n+1} - \psi_b^n - \psi_a^n) + \tilde{c}_x (\psi_b^{n+1} - \psi_a^{n+1} + \psi_b^n - \psi_a^n) \\ = i \Delta t \left(\frac{\lambda}{2^{p-1}} |A_x \psi_m^{n+1/2}|^{p-1} - V_x \right) \cdot A_x \psi_m^{n+1/2} \end{aligned} \quad (3.12)$$

Since $d_x \psi_m$ has opposite signs on opposite boundaries, Eq. (3.12) holds true on both boundaries [39]. Further simplifying and solving for ψ_b^{n+1} , we obtain the 1D absorbing boundary condition as

$$\begin{aligned} \psi_b^{n+1} = \psi_a^n + \frac{1 - \tilde{c}_x}{1 + \tilde{c}_x} (\psi_b^n - \psi_a^{n+1}) \\ + i \frac{\Delta t}{1 + \tilde{c}_x} \left(\frac{\lambda}{2^{p-1}} |A_x \psi_m^{n+1/2}|^{p-1} - V_x \right) \cdot A_x \psi_m^{n+1/2}. \end{aligned} \quad (3.13)$$

In order to use the boundary condition with the G-FDTD, we must separate the real and imaginary components by and substituting $\psi_j^n = [\psi_{\mathbf{R}}]_i^n + i[\psi_{\mathbf{I}}]_i^n$ back into the ABC. Doing this, we obtain the 1D discrete absorbing boundary conditions compatible with the G-FDTD scheme as

$$\begin{aligned} [\psi_{\mathbf{R}}]_b^n = [\psi_{\mathbf{R}}]_a^{n-1} + \sigma_x ([\psi_{\mathbf{R}}]_b^{n-1} - [\psi_{\mathbf{R}}]_a^n) \\ - \frac{\Delta t}{1 + \tilde{c}_x} \mathcal{N}_x [A_x \psi_m^{n-1/2}] \cdot A_x [\psi_{\mathbf{I}}]_m^{n-1/2}, \end{aligned} \quad (3.14a)$$

$$\begin{aligned} [\psi_{\mathbf{I}}]_b^n = [\psi_{\mathbf{I}}]_a^{n-1} + \sigma_x ([\psi_{\mathbf{I}}]_b^{n-1} - [\psi_{\mathbf{I}}]_a^n) \\ + \frac{\Delta t}{1 + \tilde{c}_x} \mathcal{N}_x [A_x \psi_m^{n-1/2}] \cdot A_x [\psi_{\mathbf{R}}]_m^{n-1/2}, \end{aligned} \quad (3.14b)$$

$$[\psi_{\mathbf{R}}]_b^{n+1/2} = [\psi_{\mathbf{R}}]_a^{n-1/2} + \sigma_x ([\psi_{\mathbf{R}}]_b^{n-1/2} - [\psi_{\mathbf{R}}]_a^{n+1/2})$$

$$- \frac{\Delta t}{1 + \tilde{c}_x} x^2 \mathcal{N}_x [A_x \psi_m^n] \cdot A_x [\psi_I]_m^n, \quad (3.14c)$$

$$\begin{aligned} [\psi_I]_b^{n+1/2} &= [\psi_I]_a^{n-1/2} + \sigma_x ([\psi_I]_b^{n-1/2} - [\psi_I]_a^{n+1/2}) \\ &\quad + \frac{\Delta t}{1 + \tilde{c}_x} \mathcal{N}_x [A_x \psi_m^n] \cdot A_x [\psi_R]_m^n, \end{aligned} \quad (3.14d)$$

where $\sigma_x = \frac{1 - \tilde{c}_x}{1 + \tilde{c}_x}$ and $\mathcal{N}_x [\psi] = \frac{\lambda}{2^{p-1}} |\psi|^{p-1} - V_x$. As shown in the next section, one may easily extend this ABC scheme to the 2D case.

3.4.2 2D Case for Absorbing Boundary Conditions

The 2D case for the ABCs proceed in a similar method to the 1D case, but we now include the y-portion of the Laplacian. To this end, we define the second-order central difference operator $\delta_y^2 f(y) = f(y + \Delta y) - 2f(y) + f(y - \Delta y)$. Then $f''(y) \approx \frac{1}{\Delta y^2} \delta_y^2 f(y)$ and

$$\begin{aligned} \left(d_t \mp \frac{c_x \Delta t}{\Delta x} d_x \right) \psi(x, y, t) \\ = i \frac{\Delta t}{2} \left(\frac{\lambda}{2} |A_x \psi(x, y, t)|^{p-1} - V_x - \frac{1}{\Delta y^2} \delta_y^2 \right) \cdot A_x \psi(x, y, t). \end{aligned} \quad (3.15)$$

Evaluating at the point $(x_m, y_k, t_{n+1/2})$ and using the same conventions to ease notation as in the 1D case, we find

$$\begin{aligned} (d_t A_x \mp \tilde{c}_x A_t d_x) \psi_{mk}^{n+1/2} \\ = i \Delta t \left(\frac{\lambda}{2^{p-1}} |A_x \psi_{mk}^{n+1/2}|^{p-1} - V_x - \frac{1}{\Delta y^2} \delta_y^2 \right) \cdot A_x \psi_{mk}^{n+1/2} \end{aligned} \quad (3.16)$$

where $\tilde{c}_x = \frac{c_x \Delta t}{\Delta x}$. Solving for ψ_{bk}^{n+1} , we find the 2D ABC for the east- and west-side boundaries which are given by

$$\begin{aligned} \psi_{bk}^{n+1} &= \psi_{ak}^n + \frac{1 - \tilde{c}_x}{1 + \tilde{c}_x} (\psi_{bk}^n - \psi_{ak}^{n+1}) \\ &\quad + i \frac{\Delta t}{1 + \tilde{c}_x} \left(\frac{\lambda}{2^{p-1}} |A_x \psi_{mk}^{n+1/2}|^{p-1} - V_x - \frac{1}{\Delta y^2} \delta_y^2 \right) \cdot A_x \psi_{mk}^{n+1/2}. \end{aligned} \quad (3.17)$$

Similarly, one may find the 2D ABC for the north- and south-side boundaries. Now, we proceed by separating ψ into its real and imaginary components, and then follow a similar procedure as before. Then, the east and west absorbing boundary conditions given in Eq (3.18), and the north and south boundary conditions are given in Eq (3.19). With $\mu_x = \frac{\Delta t}{\Delta x^2}$, $\mu_y = \frac{\Delta t}{\Delta y^2}$, $\sigma_x = \frac{1-\tilde{c}_x}{1+\tilde{c}_x}$, and $\sigma_y = \frac{1-\tilde{c}_y}{1+\tilde{c}_y}$, the ABCs are given as follows:

$$[\psi_{\mathbf{R}}]_{bj}^n = [\psi_{\mathbf{R}}]_{aj}^{n-1} + \sigma_x([\psi_{\mathbf{R}}]_{bj}^{n-1} - [\psi_{\mathbf{R}}]_{aj}^n) - \frac{1}{1+\tilde{c}_x} \left(\Delta t \mathcal{N}_x [A_x \psi_{mj}^{n-1/2}] - \mu_y \delta_y^2 \right) \cdot A_x [\psi_{\mathbf{I}}]_{mj}^{n-1/2}, \quad (3.18a)$$

$$[\psi_{\mathbf{I}}]_{bj}^n = [\psi_{\mathbf{I}}]_{aj}^{n-1} + \sigma_x([\psi_{\mathbf{I}}]_{bj}^{n-1} - [\psi_{\mathbf{I}}]_{aj}^n) + \frac{1}{1+\tilde{c}_x} \left(\Delta t \mathcal{N}_x [A_x \psi_{mj}^{n-1/2}] - \mu_y \delta_y^2 \right) \cdot A_x [\psi_{\mathbf{R}}]_{mj}^{n-1/2}, \quad (3.18b)$$

$$[\psi_{\mathbf{R}}]_{bj}^{n+1/2} = [\psi_{\mathbf{R}}]_{aj}^{n-1/2} + \sigma_x([\psi_{\mathbf{R}}]_{bj}^{n-1/2} - [\psi_{\mathbf{R}}]_{aj}^{n+1/2}) - \frac{1}{1+\tilde{c}_x} \left(\Delta t \mathcal{N}_x [A_x \psi_{mk}^n] - \mu_y \delta_y^2 \right) \cdot A_x [\psi_{\mathbf{I}}]_{mj}^n, \quad (3.18c)$$

$$[\psi_{\mathbf{I}}]_{bj}^{n+1/2} = [\psi_{\mathbf{I}}]_{aj}^{n-1/2} + \sigma_x([\psi_{\mathbf{I}}]_{bj}^{n-1/2} - [\psi_{\mathbf{I}}]_{aj}^{n+1/2}) + \frac{1}{1+\tilde{c}_x} \left(\Delta t \mathcal{N}_x [A_x \psi_{mj}^n] - \mu_y \delta_y^2 \right) \cdot A_x [\psi_{\mathbf{R}}]_{mj}^n, \quad (3.18d)$$

and

$$[\psi_{\mathbf{R}}]_{ib}^n = [\psi_{\mathbf{R}}]_{ia}^{n-1} + \sigma_y([\psi_{\mathbf{R}}]_{ib}^{n-1} - [\psi_{\mathbf{R}}]_{ia}^n) - \frac{1}{1+\tilde{c}_y} \left(\Delta t \mathcal{N}_y [A_y \psi_{im}^{n-1/2}] - \sigma_x \delta_x^2 \right) \cdot A_y [\psi_{\mathbf{I}}]_{im}^{n-1/2}, \quad (3.19a)$$

$$[\psi_{\mathbf{I}}]_{ib}^n = [\psi_{\mathbf{I}}]_{ia}^{n-1} + \sigma_y([\psi_{\mathbf{I}}]_{ib}^{n-1} - [\psi_{\mathbf{I}}]_{ia}^n) + \frac{1}{1+\tilde{c}_y} \left(\Delta t \mathcal{N}_y [A_y \psi_{im}^{n-1/2}] - \sigma_x \delta_x^2 \right) \cdot A_y [\psi_{\mathbf{R}}]_{im}^{n-1/2}, \quad (3.19b)$$

$$\begin{aligned}
[\psi_{\mathbf{R}}]_{ib}^{n+1/2} &= [\psi_{\mathbf{R}}]_{ia}^{n-1/2} + \sigma_y([\psi_{\mathbf{R}}]_{ib}^{n-1/2} - [\psi_{\mathbf{R}}]_{ia}^{n+1/2}) \\
&\quad - \frac{1}{1 + \tilde{c}_y} (\Delta t \mathcal{N}_y [A_y \psi_{im}^n] - \sigma_x \delta_x^2) \cdot A_y [\psi_{\mathbf{I}}]_{im}^n, \quad (3.19c)
\end{aligned}$$

$$\begin{aligned}
[\psi_{\mathbf{I}}]_{ib}^{n+1/2} &= [\psi_{\mathbf{I}}]_{ia}^{n-1/2} + \sigma_y([\psi_{\mathbf{I}}]_{ib}^{n-1/2} - [\psi_{\mathbf{I}}]_{ia}^{n+1/2}) \\
&\quad + \frac{1}{1 + \tilde{c}_y} (\Delta t \mathcal{N}_y [A_y \psi_{im}^n] - \sigma_x \delta_x^2) \cdot A_y [\psi_{\mathbf{R}}]_{im}^n. \quad (3.19d)
\end{aligned}$$

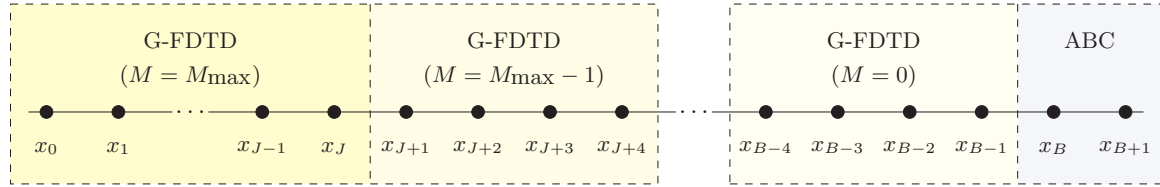
It should be pointed out that the appropriate ABC for each boundary edge needs be applied sequentially at each of the $(4M + 2)$ many points. Applying these boundary conditions sequentially is not ideal when using a GPU. To further speed up the calculation, we employ a parallelizable near-boundary treatment described in the following section.

3.5 Near-Boundary Treatment

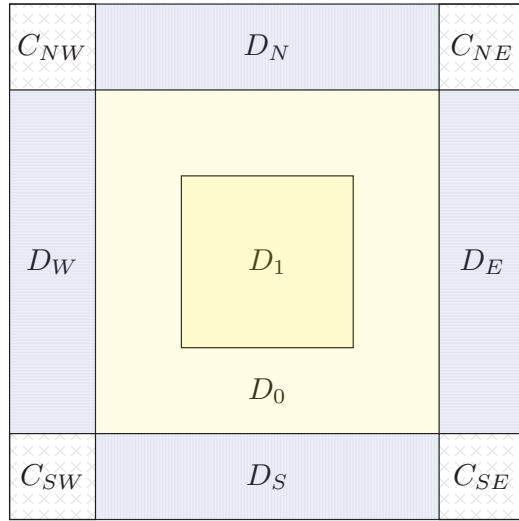
The absorbing boundary conditions must be applied sequentially; hence, it is beneficial to determine the value of ψ at as many points as possible using a parallelizable method. Since the G-FDTD method is parallelizable, we developed a method to implement the G-FDTD method near the boundary by decreasing the order of the temporal truncation error.

As mentioned previously, the G-FDTD scheme requires the value of ψ at $(4M + 2)$ points on each of the left- and right-side boundaries for the 1D case. To overcome this, we develop near-boundary treatments for the 1D case involving domain decomposition, and, similarly, we develop near-boundary treatments for the 2D case. In this way, the ABC only needs to be used to determine the value of the outermost two points for each boundary. The domain decomposition used for 1D and 2D are shown in Fig. 3.1.

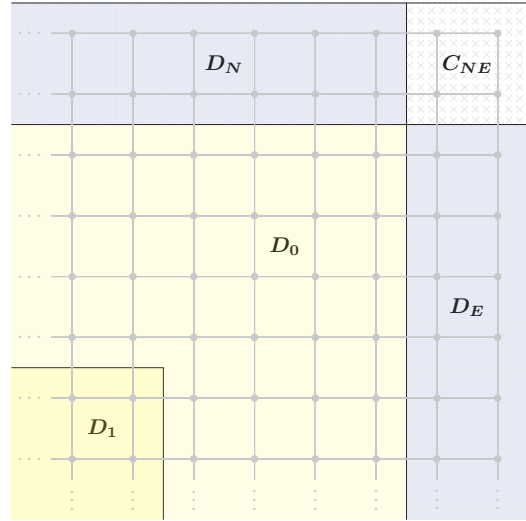
For the right-side boundary of the 1D case, to use the G-FDTD method on a grid containing $(J + 1)$ points with $M = M_{\max}$ in Eq. (2.27), we pad the computational domain



(a)



(b)



(c)

Figure 3.1: (a) Illustration (to scale) of the 1D G-FDTD implementation with ABC for arbitrary M_{\max} in the main computational region. For brevity, $B = 4M_{\max} + 1$. (b) Illustration (not to scale) of the 2D G-FDTD implementation with ABC where $M_{\max} = 1$. (c) Illustration (to scale) of the northeast corner of the 2D implementation.

with $(4M+2)$ points on the left and decompose the new computational domain into $(M_{\max} + 1)$ regions. Shown in Fig. 3.1a, the main computational domain consists of $(J + 1)$ points labeled x_0, \dots, x_J and M in Eq. (2.27) is set to be M_{\max} for all calculations in this region. The first subdomain of the padding region consists of the four points x_{J+1}, \dots, x_{J+4} and uses the G-FDTD method with M in Eq. (2.27) set to be $(M_{\max} - 1)$ for all calculations in this region. Similarly, the second subdomain of the padding region consists of the four points x_{J+5}, \dots, x_{J+8} and uses the G-FDTD method with M in Eq. (2.27) set to be $(M_{\max} - 2)$. This continues until there are only two points left in the total padding region to which one can apply the absorbing boundary conditions. A similar process can be applied to the boundary on the left side.

Shown in Fig. 3.1b and Fig. 3.1c is a similar padding and decomposition process for the 2D case with $M = 1$ in the main computational region. The main difference from the 1D case is that, in the 2D case, all padding subregions are connected except for the regions in which the ABC is applied. The disconnectedness of the ABC is due to the inherent directionality of the ABC which depends on the normal direction of the boundary edge. The inherent directionality implies that each boundary edge requires its own tailored ABC. This decomposition can easily be extended to cases where $M > 1$ as in the 1D case described previously.

Now that the near boundary treatment is established, it is necessary to determine the values of the ABC parameters c_x, c_y, V_x, V_y .

3.6 ABC Parameter Choice

It is necessary to determine the appropriate values for c_x and V_x in the one-way wave equation, Eq. (3.6), so that solutions to our boundary conditions also satisfy the 1D version of the continuity equation Eq. (2.5). To this end, multiplying Eq. (3.6) by the conjugate function $\bar{\psi}$ and multiplying the conjugate equation of Eq. (3.6) by ψ then adding the results yields the following differential equation

$$\partial_t |\psi|^2 + c_x \partial_x |\psi|^2 = 0. \quad (3.20)$$

Notice that the above equation has the same form as the continuity equation Eq. (2.6). In particular, if we choose $c_x = 2k_x$ where k_x is the wavenumber of the carrier wave, then the continuous boundary condition will satisfy the continuity equation. We still have freedom to choose κ_1 and κ_2 so long as they have the same sign and satisfy the condition

$$\kappa_1 + \kappa_2 = 2k_x. \quad (3.21)$$

Now, consider the 1D NSE and Engquist-Majda (EM) equations. Construct operators Q and Q' to be the NSE and EM operators, respectively, such that $Q\psi = (i\partial_t - \nabla^2 + \lambda|\psi|^{p-1})\psi$ and $Q'\psi = (i\partial_t \mp ic_x\partial_x + \lambda|\psi|^{p-1} - V_x + (\partial_x^2 - \nabla^2))\psi$. Suppose that the wave function is a plane wave then $\psi(x, y, t) = f(y, t)e^{\pm ikx}$. Suppose that we choose κ_1 and κ_2 such that $\kappa_1 = \kappa - \Delta\kappa$ and $\kappa_2 = \kappa + \Delta\kappa$, then $c_x = 2\kappa$ and $V_x = \kappa^2 - \Delta\kappa^2$. By calculating the difference between the operators Q and Q' , we obtained an estimate of the error (and possible reflection) due to an incorrect choice of wavenumber. Below, this difference is given as follows:

$$(Q - Q')\psi = -(\partial_x^2 \mp ic_x\partial_x\psi - V_x)\psi$$

$$\begin{aligned}
&= (k^2 - c_x k + V_x) \psi \\
&= (k^2 - 2k\kappa + \kappa^2 - \Delta\kappa^2) \psi \\
&= ((k - \kappa)^2 - \Delta\kappa^2) \psi.
\end{aligned} \tag{3.22}$$

By Eq. (3.21) and Eq. (3.22), to absorb an incoming sinusoidal wave with wavenumber k , we should choose $c_x = 2k$ and $V_x = k^2$ (equivalent to $\kappa = k$ and $\Delta\kappa = 0$) to satisfy $Q\psi = Q'\psi$.

Now, suppose that we have preemptively parameterized the boundary condition before the velocity of the incoming wave is known. With $\kappa_1 = \kappa_2 = k_0$, we have $c_x = 2k_0$ and $V_x = k_0^2$. Now, consider a wave of the form $\psi(x, y, t) = f(y, t)e^{\pm ikx}$ impacting the boundary. By a similar procedure as for Eq. (30), we find that $(Q - Q')\psi = (k - k_0)^2\psi$. Hence, for sinusoidal waves, the smaller the difference between the exact and prescribed wavenumbers $\Delta k = (k - k_0)$, the more accurate the boundary conditions should be.

Notice that the prescribed parameters $c_x = 2k_0$ and $V_x = k_0^2$ are not affected by any transverse component of the velocity. Therefore, we should be able to choose the pairs (c_x, V_x) and (c_y, V_y) independently.

3.7 Adaptive Parameter Selection

If the wavenumber of the impacting wave is not known, one may use the energy-weighted wave-number parameter selection method proposed by Xu *et al.* in [38] and used by Zhang *et al.* in [29, 30]. This approximation begins with a windowed Fourier transform also known as a Gabor transform. The Gabor transform is defined as follows:

$$\hat{\psi}(k) = \int_{x_\ell}^{x_r} w(x) \psi(x) e^{-ikx} dx = \int_{x_r-b}^{x_r} \psi(x) e^{-ikx} dx, \tag{3.23}$$

where the window function is given by

$$w(x) = \begin{cases} 1, & x \in [x_r - b, x_r], \\ 0, & x \notin [x_r - b, x_r]. \end{cases} \quad (3.24)$$

Then, the approximate value for k is given by

$$k_0 = \frac{\int_0^\infty \xi |\hat{\psi}(\xi)|^m d\xi}{\int_0^\infty |\hat{\psi}(\xi)|^m d\xi}. \quad (3.25)$$

Based on their experiments, Xu *et al.* recommended using $m = 4$.

3.7.1 Computational Procedure

Usually, finite difference methods calculations are performed sequentially on a Central Processing Unit (CPU) that is present in every computer [91]. There has been a recent shift in scientific computing to perform calculations in parallel on a GPU. This shift is brought on due to the ability of a GPU to perform large volumes of calculations in parallel given that each of the calculations is sufficiently similar [92]. Whereas a CPU is designed to do many sequential calculations very quickly, a GPU aims to reduce the collective time taken to do many parallel calculations [93]. As a consequence, even though a GPU will have slower clock speeds and smaller memory bandwidth than that of a contemporary CPU, the GPU can outperform the CPU for parallel computation on large data sets.

To employ the computational power from the GPU, we use PyOpenCL. PyOpenCL is a Python interface to OpenCL, which is written in C. Additionally, OpenCL is an open-source, cross-platform parallel programming API compatible with both CPUs and GPUs [91, 94–97]. By contrast, there is another GPU programming paradigm, CUDA, which is a proprietary software package developed to be used specifically with NVIDIA GPUs. While PyOpenCL allows access to OpenCL, it still requires some knowledge of C inasmuch as

all GPU kernels must be written in C or Fortran; however, one can use Python function wrappers to call compiled C kernels which will allow simple customized access to GPU functionality through Python. In this way, we are able to take advantage of the clarity and ease-of-use of the Python programming language as well as quick computation provided by a compiled programming language.

We developed an algorithm that utilizes PyOpenCL to implement the G-FDTD and absorbing boundary conditions on the GPU. Fig. 3.3 and Fig. 3.2, respectively, show the pseudocode and flowchart of the parallel algorithm for implementing the G-FDTD method with ABC with $M_{\max} = 1$ in the 2D case. The algorithm represented by the pseudocode and flowchart is described as follows: First, use the G-FDTD scheme with $M = 1$ to calculate the values inside the interior region D_1 , and use the G-FDTD scheme with $M = 0$ to calculate the annular region D_0 . Then, use the ABC to calculate the values in each boundary region, D_N, D_S, D_E, D_W . Since the ABC must be calculated sequentially, one may iterate over the ABC twice to update the values of the outermost points. Now, we replace the values in the initialized arrays with the newly calculated values in the correct temporal order and repeat for the desired number of time steps. This is outlined in Fig. 3.3, where $GFDTD_w$ computes the G-FDTD method with $M = w$, and ABC computes the appropriate ABC for each boundary.

Before the algorithm can be implemented, there are several steps necessary for setup. In particular, one must prepare the CPU (host) to send instructions to the GPU (device), and load values into the device memory. A basic setup for adding two vectors in parallel is included in Fig. 3.4.

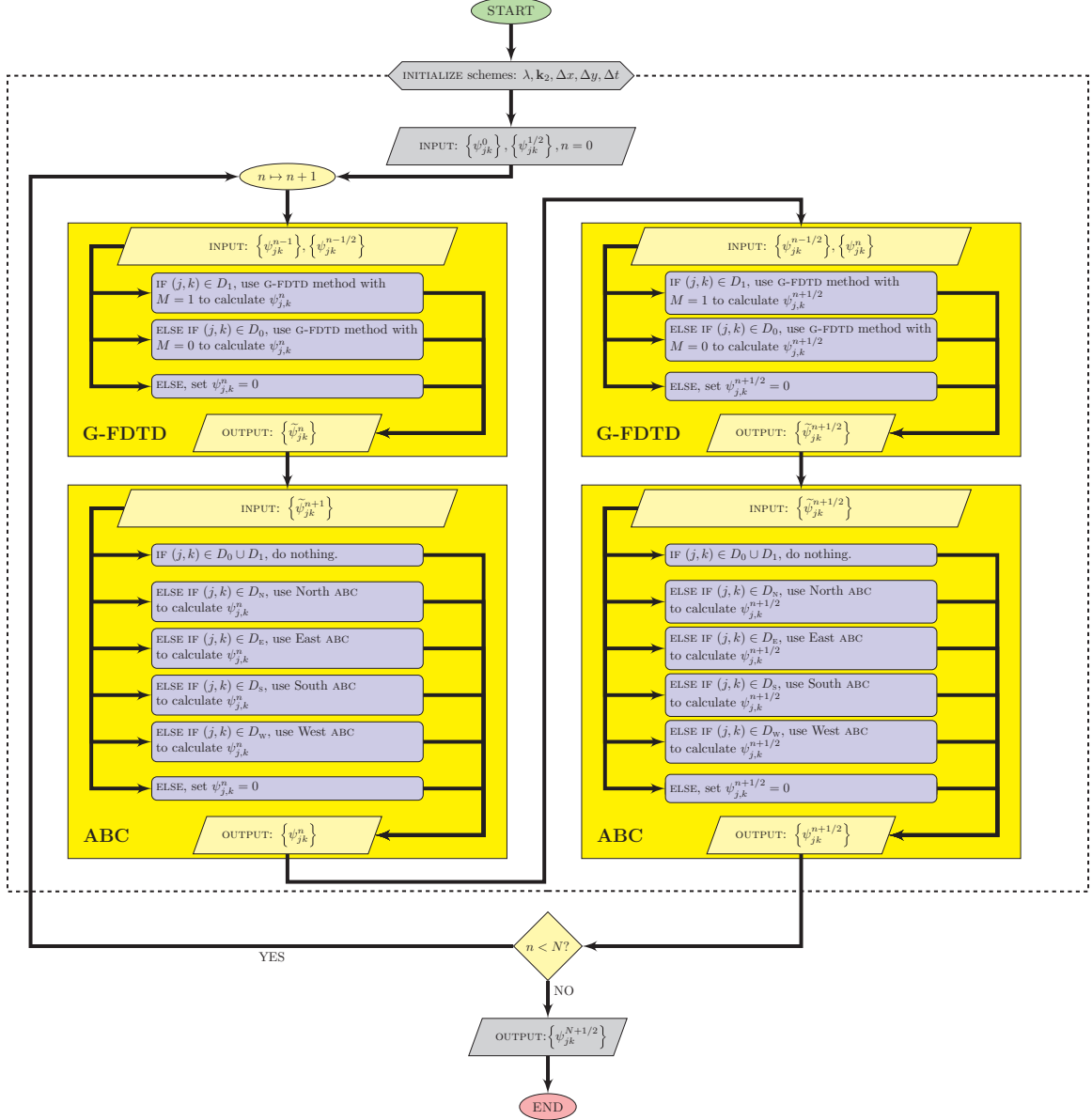


Figure 3.2: High-level flowchart of the algorithm used for the 2-D G-FDTD with ABC. Here, D_1 is the main computation region where $M = 1$ and D_0 is the square annular region where $M = 0$. Each domain with a cardinal direction as a subscript refers to the side boundary. Refer to Fig. 3.1 on page 42 for an illustration of D_0 , D_1 , D_N , D_E , D_S , and D_W .

Algorithm 1 : How to Implement G-FDTD Method with ABC	
	input : $\psi^0, \psi^{1/2}, N$
	output : ψ^N
	parameters $\lambda, \mathbf{k}_2, \Delta x, \Delta y, \Delta t$
	:
1	initialization
2	$n \leftarrow 0$
3	$d = \text{dimension}(\psi^0)$
4	$\psi_{jk}^1 \leftarrow \text{zero_array}(d)$
5	$\psi_{jk}^{3/2} \leftarrow \text{zero_array}(d)$
6	computation
7	while $n \leq N$ do
8	for $\nu = 1, 3/2$ do
9	G-FDTD
10	forall (j, k) do
11	if $(j, k) \in D_1$ then
12	$\psi_{jk}^\nu \leftarrow \text{GFDTD}_1(\psi^{\nu-1/2}, \psi^{\nu-1}, j, k)$
13	else if $(j, k) \in D_0$ then
14	$\psi_{jk}^\nu \leftarrow \text{GFDTD}_0(\psi^{\nu-1/2}, \psi^{\nu-1}, j, k)$
15	else
16	$\psi_{jk}^\nu \leftarrow 0$
17	end
18	ABC
19	forall (j, k) do
20	$\psi_{jk}^\nu \leftarrow \text{ABC}(\psi^\nu, \psi_{jk}^{\nu-1/2}, \psi^{\nu-1}, j, k)$
21	end
22	end
23	update
24	$n \leftarrow n + 1$
25	for $\nu = 1, 3/2$ do
26	forall (j, k) do
27	$\psi_{jk}^{\nu-1} \leftarrow \psi_{jk}^\nu$
28	end
29	end
30	end

Figure 3.3: Pseudocode for an algorithm for implementing the G-FDTD method with ABC in parallel.


```

1  import numpy as np
2  import pyopencl as cl
3
4  # On run, this find a host (CPU) and device (GPU) on which to perform the
5  # computation this will prompt the user to choose a device from a list of
6  # devices detected.
7  ctx = cl.create_some_context()
8  # Set up the queue for the computation
9  queue = cl.CommandQueue(ctx)
10
11 # Set up host values
12 u0 = np.random.random(1000000)
13 u1 = np.random.random(1000000)
14 # Set up resultant array filled with zeros
15 u2 = np.zeros(1000000)
16
17 mflag = cl.mem_flags
18 # Load arrays into device memory
19 U0 = cl.Buffer(ctx,mflag.READ | mflag.COPY_HOST_PTR, hostbuf=u0)
20 U1 = cl.Buffer(ctx,mflag.READ | mflag.COPY_HOST_PTR, hostbuf=u1)
21 # Prepare resultant array on device
22 U2 = cl.Buffer(ctx,mflag.READ_WRITE | mflag.COPY_HOST_PTR, hostbuf=u2)
23
24 #Create a string containing the desired program
25 program_string = """
26 /* This is a string of code written in C */
27
28 /* Enable double precision */
29 #pragma OPENCL EXTENSION cl_khr_fp64 : enable
30
31 /* function that adds two vectors */
32 __kernel void add(
33     __global double *U0,
34     __global double *U1,
35     __global double *U2)
36 {
37     int j = get_global_id(1);
38     U2[j] = U0[j] + U1[j];
39 }
40 """
41
42 # Build the program
43 prg = cl.Program(ctx,program_string).build()
44 # Array shape is required for CL functions
45 SHAPE = u0.shape
46 # Perform add
47 prg.add(queue, SHAPE, None, U0, U1, U2)
48 # Copy result from device memory to host memory
49 prg.add(queue, SHAPE, None, U0, U1, U2)
50

```

Figure 3.4: Example code for adding two vectors in parallel using PyOpenCL.

3.8 Chapter Summary

This section developed absorbing boundary for the G-FDTD method for solving the NSE were for both the 1D and 2D cases as well as near boundary treatments. The ABCs, however, require parameters c_x and V_x , and the reasonable choices of values for those parameters were presented which depend on the wavenumber of the impinging wave. A method was discussed to make the parameter selection adaptive. Moreover, this chapter discusses the computational procedure for the GPU implementation of the G-FDTD method with ABCs.

CHAPTER 4

NUMERICAL RESULTS FOR ABC METHOD

In this section, we first analyze the reflection coefficient of solitons impacting the boundary as a function of wavenumber, and as several simulation variables are varied. These variables include the spacial resolution of the simulation, the wavenumber used to calculate c_x and V_x , and the width of the window in the Gabor transform. We also compare our ABC with the implicit ABC developed by Zhang *et al.* [29] by measuring the reflection coefficients of each. We then analyze the timing of the parallel algorithm when implemented on a GPU and multicore CPU, and a serialized version of the algorithm implemented on a CPU ¹. Finally, we provide several numerical examples including 1D and 2D soliton propagation, 2D Gaussian packet collision, and 2D dipole radiation.

4.1 Testing the ABC

To analyze the performance of the ABCs, we define the reflection coefficient R as

$$R = \frac{\int_a^b |\psi(x, \tau)|^2 dx}{\int_a^b |\psi(x, 0)|^2 dx}, \quad (4.1)$$

where τ is some time step after initial reflection of the soliton off the boundary, but before it has impacted the opposite boundary. Without computational boundaries imposed by truncated domains, solitons would propagate freely through the boundary; therefore, the reflection coefficient is a measure of the absolute global error introduced into the system

¹The results in this chapter have been published in the journal *Computer Physics Communications* [44]

by the imposed boundaries. A perfectly absorbing boundary would have a reflection coefficient of $R = 0$, and $R = 1$ is consistent with Dirichlet and Neumann boundary conditions. Hence, the smaller the value of R , the greater the performance of the boundary conditions.

Four cases were proposed for analyzing the performance of the ABCs. For all cases, we utilized the reflection coefficient given in Eq. (4.1). For each case, we used the nonlinear SE given by

$$i\partial_t\psi - \partial_x^2\psi - 2|\psi|^2\psi = 0, \quad (4.2)$$

and the time step was chosen to be $\Delta t = (\Delta x)^2/10$. We considered initial conditions such that the exact solution is given by

$$\psi(x, t; k) = \text{sech}(x - 15 - 2kt)e^{-ik(x-15)}e^{i(k^2-1)t}, \quad (35)$$

where ψ is parameterized by the wavenumber k and is propagating to the right on the domain $[-20, 20]$. The ABC was implemented on the right-side boundary. Fifty values were chosen from the range $[1, 10]$ as values for k , and those k -values were used to parameterized solitons used for the initial conditions of 50 different simulations. We chose the range $[1, 10]$ since it represents an order of magnitude increase in k without having solitons so slow that they took too long to approach the boundary or so fast that they quickly accrued numerical error.

Test Case 1. We tested how the reflection coefficient changes with the number of grid points N used in the simulation. We did this by choosing a value of N and calculating the reflection coefficients for each of the aforementioned 50 simulations. The value of the wavenumber k_0 used to precondition the ABC was chosen to be exact ($k_0 = k$ for each soliton) and we chose $c_x = 2k_0$ and $V_x = k_0^2$. This was done for $N = 200, 300, 400, 500$.

From Fig. 4.1a, we can see that, for small wavenumber k , the reflection coefficient is not greatly affected by the choice of N as all plots are clustered for small k . We can also see that there is a minimum to each of the reflection coefficient curves and the location of the minimum is dependent on the number of points used in the simulation. In particular, for larger values of N , the minimum of R obtains a smaller value and occurs at a larger wavenumber. Hence, while the number of grid points does not significantly change the reflection of solitons with smaller values of k , a finer spacial grid is needed in order to absorb solitons with larger values of k . This is consistent with the fact that larger frequencies require more sampling points for accurate reconstruction.

Test Case 2. We varied the wavenumber used to precondition the ABC. In this way, we can test the performance of the ABC if the velocity is not chosen particularly well. For this case, we set $N = 400$, and we chose to use wavenumbers of $k_0 = 1, 2, 5, 10$ to precondition the ABC with $c_x = 2k_0$ and $V_x = k_0^2$. For each value of k_0 , we calculated the reflection coefficient for each of the 50 different initial conditions.

It can be seen in Fig. 4.1b that choosing a good approximation k_0 is important. The black dashed line represents the baseline reflection coefficient for when the wavenumber is chosen exactly ($k_0 = k$). Large differences from the exact wavenumber k and prescribed wavenumber k_0 are still able to absorb the majority of the soliton, but 30-40% of the wave may be reflected back into the computational domain. That being said, small variations from the exact value k do not significantly change the order of the reflection coefficient.

Test Case 3. We considered the adaptive case, and in particular, how the width of the window used in the Gabor transform affects the reflection coefficient. This test will allow for an appropriate window width to be determined. Here, we kept $N = 400$, and hence

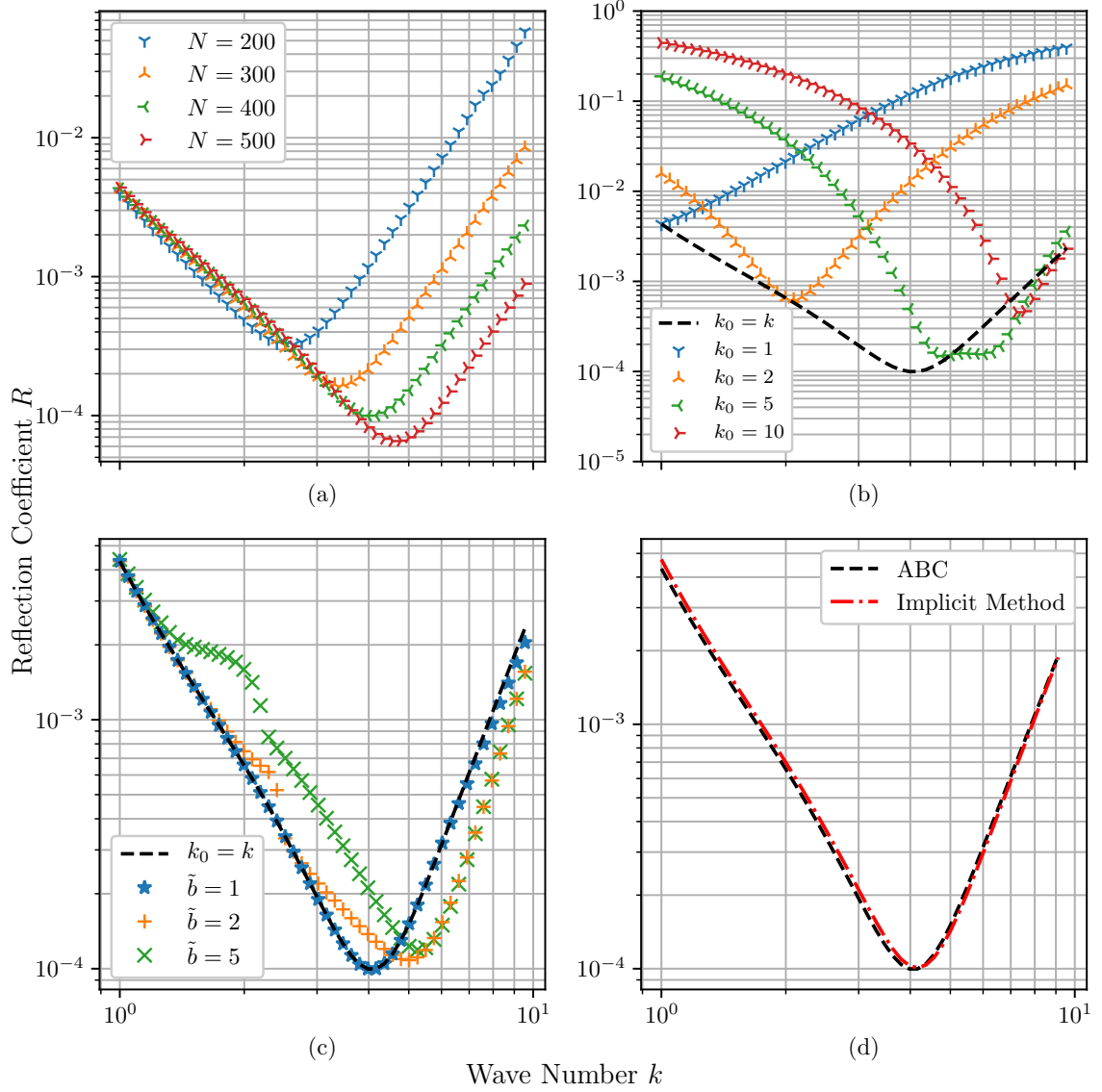


Figure 4.1: Reflection coefficient as a function of the wavenumber k of the incoming soliton as (a) the total number of points used in the simulation varies, (b) the ABC was preconditioned with the various wavenumbers k_0 , (c) the window width \tilde{b} of the Gabor transform was varied for the adaptive case, and (d) the ABC was compared with implicit scheme developed by Zhang *et al.*[29].

$\Delta x = 0.1$. When implementing the Gabor transform numerically, we chose values of $b = 0.1, 0.2, 0.5$ where b is the width of the window in the Gabor transform on page 45. We define $\tilde{b} = b/\Delta x$ to be the number points that were taken to be nonzero in the discrete Gabor transform. Then, for each of the 50 values of k , we tested the case where $\tilde{b} = 1, 2, 5$.

From Fig. 4.1c, we can see how the reflection coefficient varies as the width of the window used in the Gabor transform varies. The black dashed line in Fig. 4.1c is the baseline reflection coefficient for when the wavenumber is approximated exactly ($k_0 = k$). Remarkably, using the adaptive method with the window width $\tilde{b} = 1$ is entirely sufficient to approximate k ! In fact, there is little difference between the case where $\tilde{b} = 1$ and the case where the exact boundary conditions were prescribed at onset. This may be because solitons are nearly monochromatic [18]. While using a window width of one is nearly as good as prescribing the exact boundary conditions, solitons with larger values of k can be absorbed more readily with larger window widths. On the other hand, solitons with smaller values of k are reflected more readily. Hence, for simulations where smaller wavenumber are not a concern, a larger window width could be used to decrease the reflection coefficient of solitons with larger k .

Test Case 4. We compared our method with the second-order implicit ABC developed by Zhang *et al.* [29]. For each value of k , we preconditioned our ABC and the implicit ABC with the exact wavenumber $k_0 = k$ and $N = 400$. We then ran the G-FDTD scheme with each boundary condition. It should be pointed out that, since the ABC by Zhang *et al.* is implicit, this calculation was not performed on the GPU since separating the real and imaginary parts of the scheme would have been fairly complicated. This means that in programming language without easily implemented support for complex floats, this

computation is lengthy and difficult to program. For this reason, this calculation was done using standard scientific Python libraries.

From Fig. 4.1d, we can see that there is little difference between the ABC scheme developed in this study and the second-order ABC scheme developed by Zhang *et al.* [29]. Importantly, even though the presented ABC is explicit, the difference between their reflection coefficients is negligible in the regimes that we tested.

4.2 Testing the Parallel Algorithm

To determine the performance of the algorithm on the GPU with respect to the algorithms on the CPU, we compare the total real-world time it takes to compute one unit of simulation time on the processor. We perform the G-FDTD method with ABCs using three different implementations: 1) a serialized version of the algorithm implemented in Python using standard optimizations for NumPy (that is the standard Python linear algebra library) performed on a CPU (Py-CPU), 2) the algorithm implemented in Python and parallelized with PyOpenCL that utilizes the multicore parallel processing on the CPU (CL-CPU), and 3) the algorithm implemented in Python and parallelized with PyOpenCL that utilizes the parallel processing on the GPU (CL-GPU). Both the CL-GPU and CL-CPU are implementations of the parallelized algorithm described previously, but Py-CPU only uses standard scientific Python libraries.

Each method was timed by implementing a simulation using the G-FDTD with ABC for several step sizes, Δx . The simulation was 2-D, so there were roughly $1/(\Delta x)^2$ operations per time step. Additionally, the time step chosen for each simulation was $\Delta t = (\Delta x)^2/10$. Therefore, the number of individual float calculations to simulate a single unit

of simulation time should be proportional to $1/(\Delta x)^4$. Since the Py-CPU implementation is non-parallel, we expected that the total amount of real-world time necessary for the simulation complete is proportional to $1/(\Delta x)^4$. Both the CL-CPU and CL-GPU implementations can perform more calculations at once, therefore, we should see smaller exponents for $1/\Delta x$.

The algorithm was tested for performance on an Intel Core i7 CPU with four cores and a clock speed of 2.8 GHz. The GPU used was an AMD Radeon R9 M370X, which has 640 cores and a clock speed of 800 MHz. The initial condition for each simulation was set to be zero everywhere since the time to complete any floating-point operation was roughly independent of the actual value of the float. We used various values of Δx and $\Delta t = (\Delta x)^2/10$. The simulation was run for integer part of $1/\Delta t$ time steps. These simulations were performed several times for each value of Δx .

The top of Fig. 4.2 shows the plots of average total time-performance data collected while using our computational procedure for the 2-D case. The datasets were produced by measuring the real-world time necessary to compute one unit of time in the simulation. Since we chose $\Delta t \propto (\Delta x)^2$, and the simulation is two-dimensional, we should expect that the total real-world time necessary to calculate a unit of time T_{total} is proportional to $(\frac{1}{\Delta x})^4$ for sequential processing.

We found that each of the data sets roughly followed a power law with respect to the linear resolution $1/\Delta x$. Table 1 shows the parameters for the power law fits assuming the form $T_{total} = 10^b (\frac{1}{\Delta x})^m$ for each data set. For sequential processing, the time needed scales as expected, while for parallel processing the total time does not grow as quickly with respect to $1/\Delta x$ as the sequential CPU.

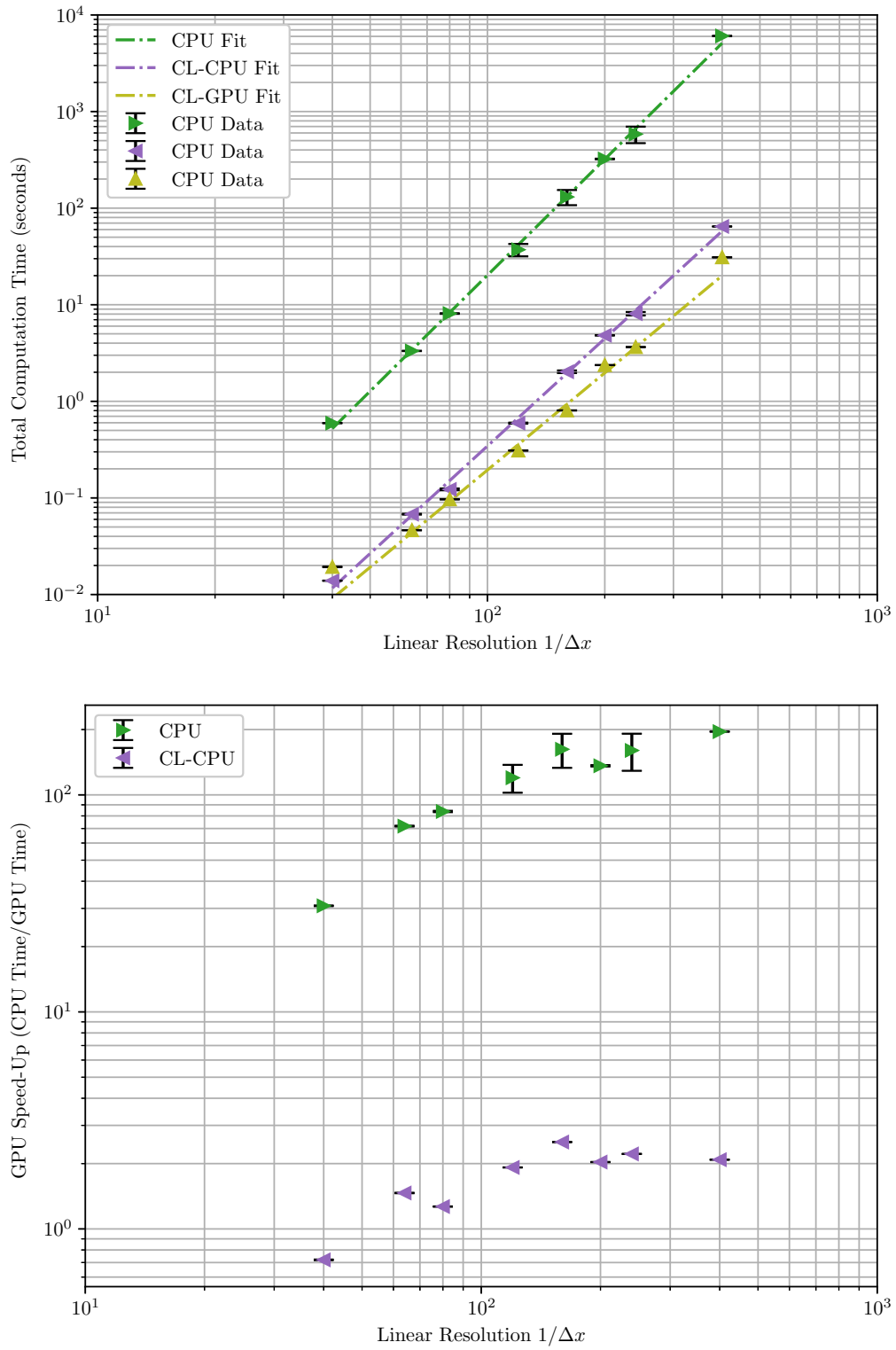


Figure 4.2: *Top:* Average total computation time needed to calculate one unit of time for the 2-D NLSE using the G-FDTD method with ABC plotted versus the linear resolution used. *Bottom:* The speed-up (CPU Timing/GPU Timing) versus linear resolution.

Table 4.1: Parameters for Power Law Fitting for GPU/CPU Comparisons.

Method	Power m	Constant b
Py-CPU	3.990	-7.395
CL-CPU	3.696	-7.855
CL-GPU	3.342	-6.674

The bottom of Fig. 4.2 shows the relative speedup in the calculation when using the GPU over both sequential and parallelized CPU implementations. The GPU for small data sets calculates more slowly than the parallelized CPU, however, it outperforms the parallelized CPU for large data sets leveling off near a speedup factor of two. Compared with the non-parallelized CPU implementation, the GPU provides substantial speedup. In particular, for large data sets, the speedup factor is nearly 200!

4.3 Numerical Examples

To test the applicability of our new computational procedure, five examples were tested. These include a 1D bright soliton, two 2-D bright solitons, a collision of Gaussian wave packets, and dipole radiation. For all cases, we chose the ABC parameters c_x , c_y , V_x , and V_y by examining the wavenumber or wavevector of the carrier wave of the wave function that is to be absorbed.

Example 4.1 (1D NSE, Soliton Propagation). We implemented the ABC using with the 1D bright soliton solution presented in [18]. The G-FDTD was used to calculate the NSE with $\lambda = -2$ and $p = 3$, where the analytical solution can be expressed as $\psi(x, t) = \text{sech}(x + 10 - 4t)e^{-i(k_2(x+10)-3t)}$ with $k_2 = 2$. In our computation, 400 grid points were used to evenly span the domain $-20 \leq x \leq 20$, and Δt was insured to satisfy the stability

condition for the G-FDTD method. Fig. 4.3 shows different solutions to the NSE with the initial conditions given by the analytical solution at $t = 0$ and $t = \Delta t/2$.

From Fig. 4.3a, we can see that, for the analytical solution, the wave passes directly through the boundary without reflection near $t = 7.5$. In Fig. 4.3b, where the boundaries are set equal to zero, the wave reflects off the boundaries at $t = 7.5$ and $t = 17.5$, but has little distortion when it is away from the boundaries. For Fig. 4.3c, the G-FDTD scheme with domain decomposition in the interior, and the analytical solution on the boundaries, one can see that the wave is almost completely absorbed near $t = 7.5$, and the reflection part of the wave begins to disperse. In Fig. 4.3d, we implemented the G-FDTD scheme with absorbing boundary conditions. In this example, we found that the wavenumber of the carrier wave is 2; therefore, we chose $c_x = 2k_2 = 4$ and $V_x = k_2^2 = 4$. In the ABC, we used these parameters on the right-side boundary and set the left-side boundary to be zero at all times. One can see that the wave is mostly absorbed at $t = 7.5$; however, there is still some visible reflection. The amount of reflection is greater than that of Fig. 4.3c, but is still very small compared to the initial magnitude of the wave.

Example 4.2 (2D NSE, Soliton Propagation). For the example shown in Fig. 4.4 and Fig. 4.5 on pages 64 and 66 respectively, we considered the 2D NSE given as

$$i\partial_t\psi - \partial_x^2\psi - \partial_y^2\psi - 2|\psi|^2\psi = 0, \quad (4.3)$$

and a bright soliton traveling diagonally towards the northeast corner. In our computation, the domain was chosen to be $-20 \leq x, y \leq 20$ with a 400×400 point grid evenly spaced across the domain to give $\Delta x = \Delta y = 0.1$ and a time step of $\Delta t = (\Delta x)^2/32$.

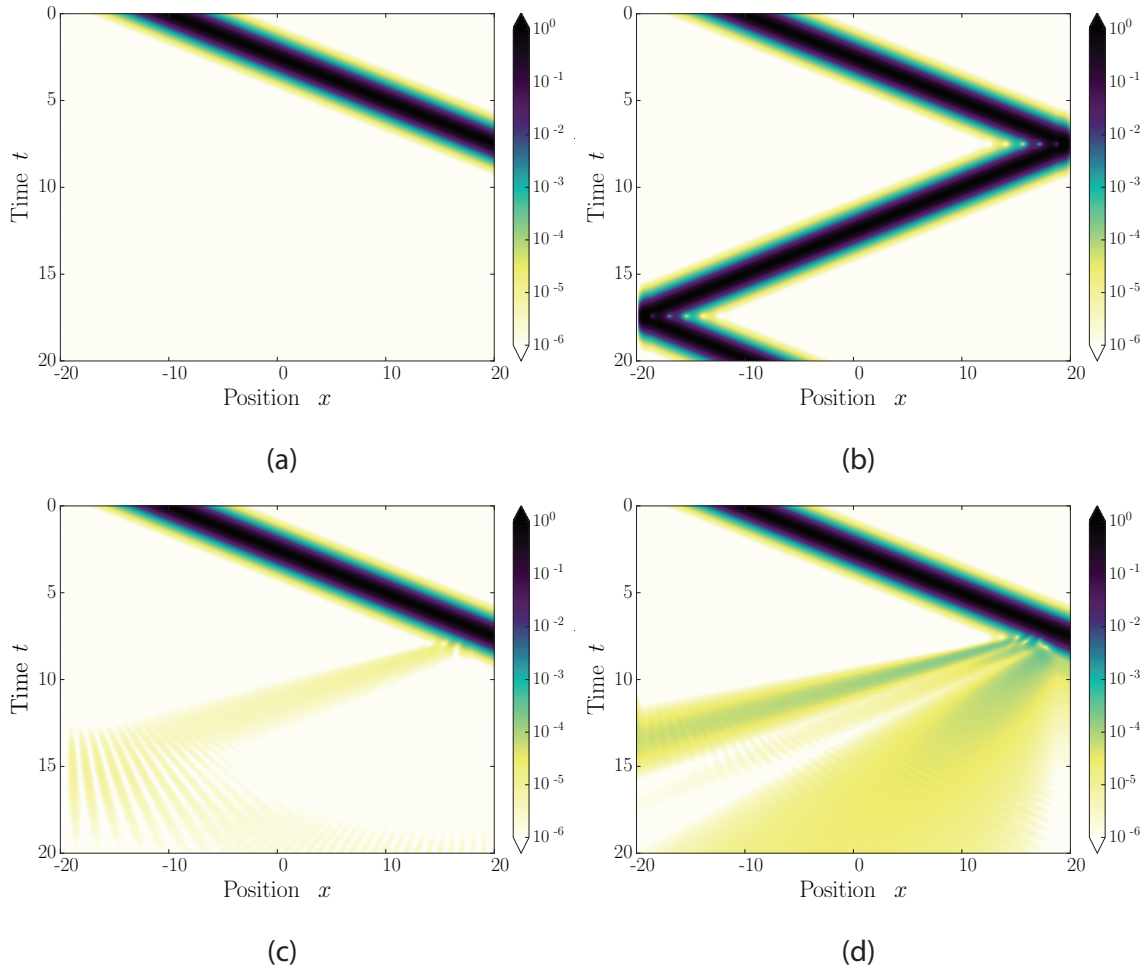


Figure 4.3: (Example 4.1) Space-time plot of a simulation of the propagation of a 1D bright soliton $|\psi(x, t)|$ with $p = 3$, $\lambda = -2$ where (a) the analytical solution was calculated, (b) the G-FDTD method was used with boundary points set to zero, (c) the G-FDTD method was used with boundary values given by the analytical solution, and (d) the G-FDTD method was used with ABC on the boundary.

Case 1. For this case, we chose the initial conditions corresponding to the soliton solution $\psi(x, y, t) = \text{sech}(\mathbf{k}_1 \cdot \mathbf{x} - 8t)e^{-i(\mathbf{k}_2 \cdot \mathbf{x} - 6t)}$ where $\mathbf{k}_1 = \langle 1, 1 \rangle$ and $\mathbf{k}_2 = \langle 2, 2 \rangle$. Notice that the x - and y -components for each of \mathbf{k}_1 and \mathbf{k}_2 are the same. In Fig. 4.4a, the G-FDTD method was used and all values on the boundary were calculated using the analytical solution. The simulation in Fig. 4.4b has all boundary values are all set to zero. To produce Fig. 4.4c, we used absorbing boundary conditions on all boundaries with $c_x = c_y = V_x = V_y = 4$. The value of these parameters was calculated exactly as in the 1D example. One can see from Fig. 4.4c that analytical solution on the boundary produces very small distortions beginning at $t = 0.6$ which accumulate as the wave propagates. When the boundary values are set to zero, one can see from Fig. 4.4b that a large amount of error that forms around $t = 0.6$ as a high-frequency reflection. This is because points which were nonzero initially were set to zero after the first time step. At $t = 1.2$, one can see the circular ripples near $\langle x, y \rangle = \langle 18, 2 \rangle$ and $\langle x, y \rangle = \langle 2, 18 \rangle$. This may be due to the solution undergoing wave-collapse caused by the wave interfering with its reflection. The wave continues reflect off the boundary at $t = 1.8$ and by $t = 2.4$ the errors from reflection have nearly obscured the soliton from view. At $t = 3.0$, nearly the entire wave has been reflected back into the computational domain. When the absorbing boundaries are used, as seen in Fig. 4.4c, similar small distortions appear as compared with the case that used the analytical boundary. At $t = 3.0$, the distortions are roughly an order of magnitude larger, locally, than that of the simulation using the analytical solution; however, they are several orders of magnitude smaller than in the simulation using values of zero along the boundary.

Case 2. Now we consider the case where $k_{1,x} \neq k_{1,y}$ and $k_{2,x} \neq k_{2,y}$. This time, we chose $\mathbf{k}_1 = \langle \frac{\sqrt{6}}{2}, \frac{\sqrt{2}}{2} \rangle$ and $\mathbf{k}_2 = \langle \sqrt{2}, \sqrt{6} \rangle$, then we chose the initial conditions such

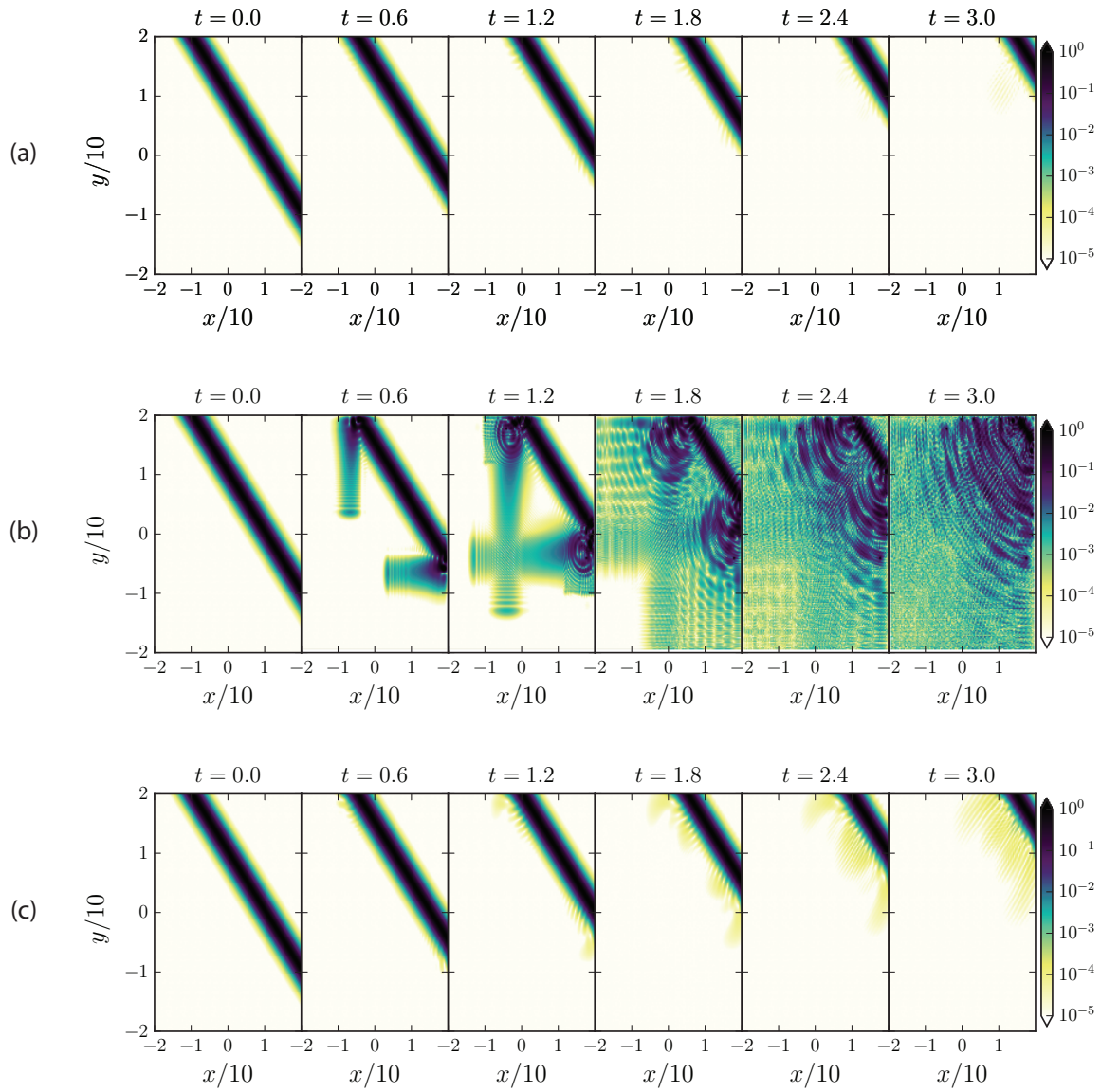


Figure 4.4: (Example 4.2, Case 1). Simulation of soliton propagating diagonally toward the boundary using G-FDTD method with $p = 3$, $\lambda = -2$, $\Delta t = (\Delta x)^2/32$ where (a) the analytical solution was used on boundary, (b) the boundary values were set to zero, and (c) the boundaries were calculated using ABCs.

that exact solution was given by $\psi(x, y, t) = \text{sech}(\mathbf{k}_1 \cdot \mathbf{x} - 4\sqrt{3}t)e^{-i(\mathbf{k}_2 \cdot \mathbf{x} - 6t)}$. We used the G-FDTD scheme with ABCs where the parameters for the ABCs were chosen to be $c_x = 2k_{2,x} = 2\sqrt{2}$, $c_y = 2k_{2,y} = 2\sqrt{6}$, $V_x = k_{2,x}^2 = 2$, $V_y = k_{2,y}^2 = 6$. In Fig. 4.5a, the boundary values were set to be zero, and the simulation became unstable just before $t = 1.8$. This instability is likely due to the errors caused by the soliton reflecting back into the computational domain. In Fig. 4.5b, the soliton is able to freely pass through the boundary with little reflection. This example is particularly important since it shows that a soliton can still be absorbed even when the group and phase velocities are not aligned and have different x - and y -components.

Example 4.3 (2D NSE, Gaussian Packet Collision). Illustrated in Fig. 4.6 on page 68, we used the same grid as in the second and third examples but used $\Delta t = (\Delta x)^2/16$ instead. We again used the G-FDTD method with the ABC to solve the NSE given by Eq. (4.3). We collided two Gaussian packets, initially separated by a distance of 20 units, each with a standard deviation of 2 units and a wave vectors of $\mathbf{k} = \langle \pm 2, 0 \rangle$ so that they would travel in opposite directions toward one another. As a consequence, we chose $c_x = c_y = V_x = V_y = 4$ similar to the previous two examples. Fig. 4.6a is a simulation using the G-FDTD method with points on the boundary set to zero, and for Fig. 4.6b ABC was used to compute points on the boundary. At $t = 0$ and $t = 3.0$ the wave is significantly far from the boundary that little reflection has occurred. However, by $t = 6.0$ the simulation with boundary points set to zero has already formed a high-frequency interference pattern due to reflection while the simulation using the ABC only has an interference pattern due to the packets themselves interfering. At $t = 12$, the simulation in Fig. 4.6b shows that the wave still is near the boundary; however, it is impossible to determine this in Fig. 4.6a. At $t = 15$, the wave is

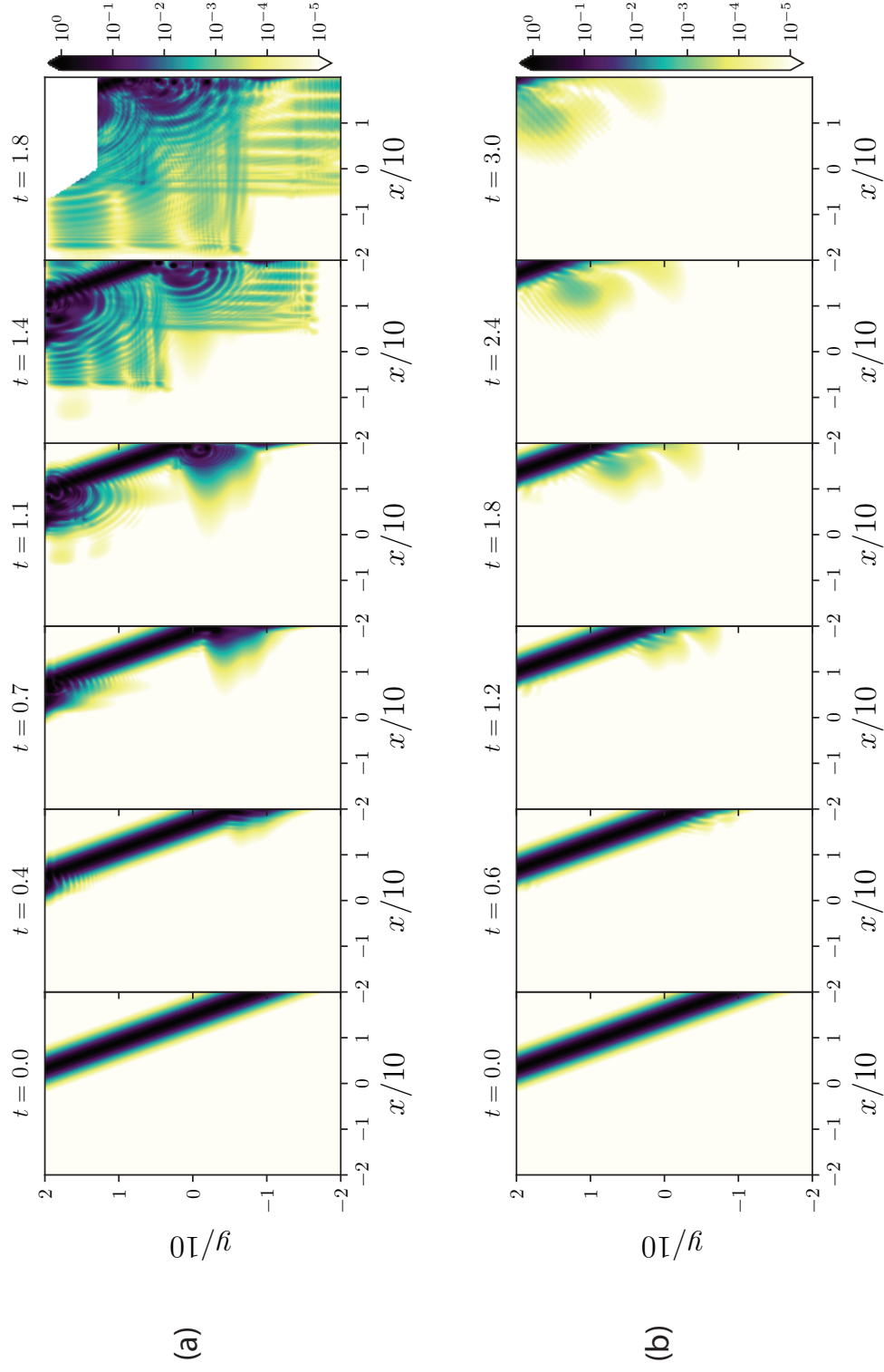


Figure 4.5: (Example 4.2, Case 2) Simulation soliton propagating with different wavenumbers on each boundary toward the boundary using G-FDTD method with $p = 3$, $\lambda = -2$, $\Delta t = (\Delta x)^2/32$ where (a) the analytical solution was used on boundary, (b) the boundary values were set to zero, and (c) the boundaries were calculated using ABCs.

nearly completely absorbed in Fig. 4.6b, but in Fig. 4.6a nearly the entire wave has been reflected into the computational domain as error.

Example 4.4 (2D NSE, Dipole Radiation). As shown in Fig. 4.7, we examined the behavior of the NSE given by Eq. (4.3) with in-phase dipole disturbances. We chose $\Delta t = 0.01$, and used the same grid as before. We chose the dipole distance to be 2 and chose the angular frequency $\omega = 6$. We then chose $c_x = c_y = V_x = V_y = 4$ as before. Fig. 4.7a shows a simulation of the described dipole radiation using G-FDTD without ABCs and Fig. 4.7b shows the same simulation with ABCs. At $t = 1.2$ and $t = 2.4$, the simulations are fairly similar since both of the waves have yet to interact with the boundary. At $t = 3.6$, one can see that the ABC allows the waves to pass through the boundary, while the simulation with the boundaries set to zero has developed interference artifacts due to reflection. At $t = 4.8$ and $t = 6.0$ the ABC allows the wave to pass through the boundary and the dipole interference pattern remains visible throughout the simulation, but in the simulation with boundaries set to zero the dipole interference structure is obscured due to reflection from the boundaries.

4.4 Chapter Summary

We have shown that the G-FDTD method along with the ABCs presented in Chapter 3 allows for the simulation of the NSE on unbounded domains. Using several test cases, we have determined appropriate parameters to use with ABC depending on the wavenumber of the impinging wave. We have also tested a method by which one can make the ABC adaptive by determining the value of the wavenumber by way of a Gabor transform. Furthermore, we have tested the method using several numerical examples including 1D and

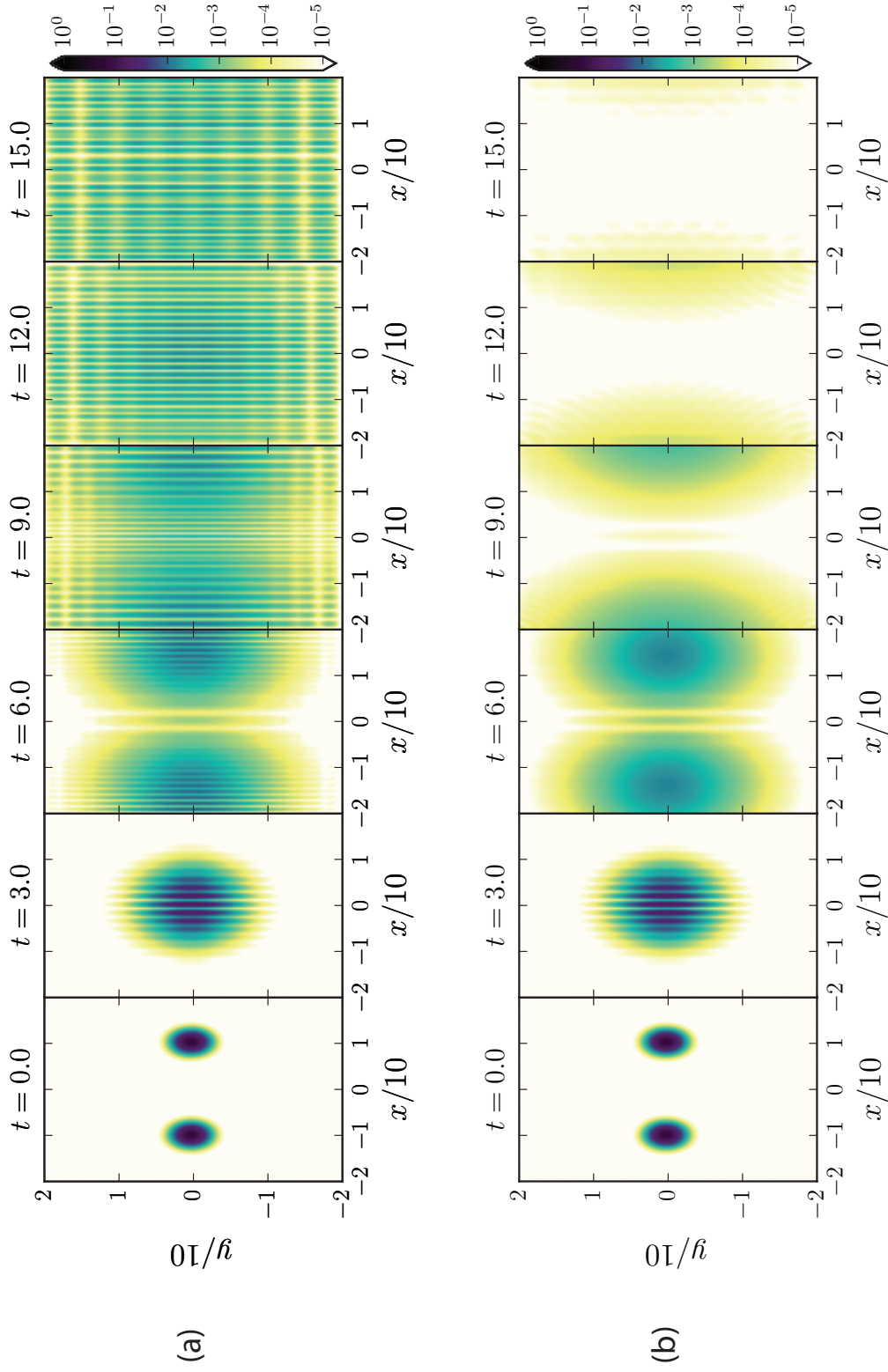


Figure 4.6: (Example 4.3). Simulations of Gaussian packet collision using G-FDTD with $p = 3$, $\lambda = -2$ where (a) the boundary points were set to zero, and (b) ABC was used to calculate boundary points.

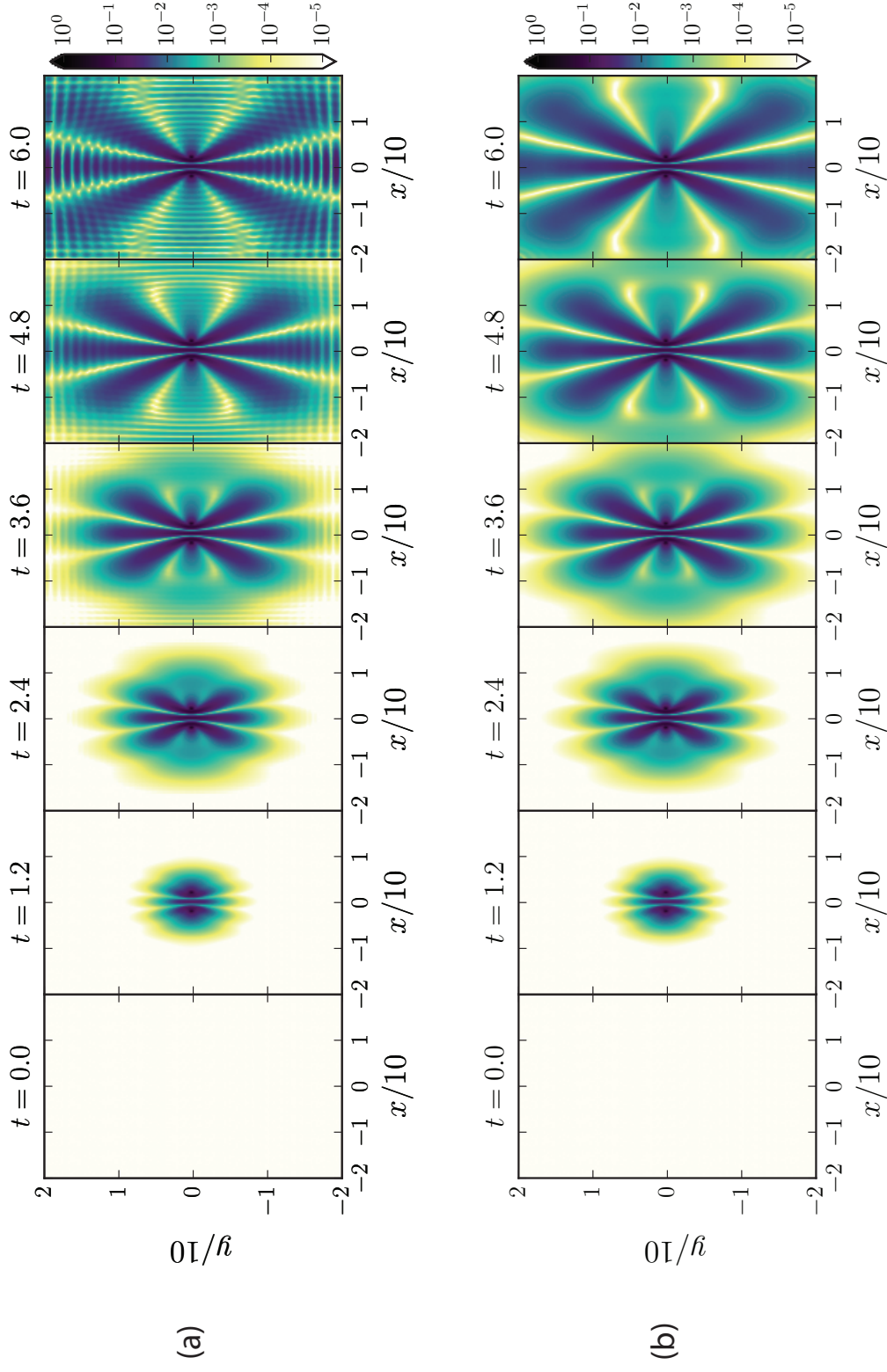


Figure 4.7: (Example 4.4). Simulations of in-phase dipole radiation with dipole distance $d = 2$ and angular frequency $\omega = 6$ using G-FDTD with $p = 3$, $\lambda = -2$ where (a) the boundary points were set to zero, and (b) the ABC was used on the boundary.

2D soliton propagation, as well as the collision of 2D Gaussian packets, and 2D dipole radiation.

CHAPTER 5

FRACTIONAL MOMENTUM OPERATOR AND THE FRACTIONAL FDTD SCHEME

The ABC method was developed specifically for the G-FDTD implementation of the NSE because ABC already exists for the G-FDTD method for the linear SE [27]. That being said, there are three key disadvantages to the using the ABC method: (1) there is inherent reflection due to the interface of two differential equations, (2) the ABC requires tedious domain decomposition, (3) one must guess the wave number of the incoming wave fairly accurately, or there can be substantial reflections at the boundary.

We will attempt to remedy these three issues through the introduction of a fractional momentum operator which smooths the transition between the interior and exterior schemes for both the linear and nonlinear cases. Since this method is novel and will be applied to both the linear *and* nonlinear cases, we will use the more simple FDTD scheme as the interior scheme rather than the more complicated G-FDTD scheme. We do this to show that the method works for a base case, and more higher-order accuracy methods based on this approach can be created later.

5.1 Fractional Momentum Operator

We first write the SE and NSE into a single expression as

$$i\hbar\partial_t\psi = \frac{1}{2m}\hat{p}_x^2\psi + \mathcal{V}\psi, \quad (5.1)$$

where $\psi = \psi(\mathbf{x}, t)$ is a complex wave function, $\hat{p}_x = -i\hbar\nabla$ is the momentum operator, and $\mathcal{V} = \lambda|\psi|^2 + V(\mathbf{x}, t)$. Here, $\lambda = 0$ corresponds to the linear SE and $V = 0$ corresponds to the NSE. While, under a standard discretization scheme, the SE does not have transparent boundary conditions, there are two corresponding TBCs in the form of one-way wave equations with imaginary restoring terms that can be written as [33, 34, 39]

$$i\hbar\partial_t\psi = c_x\hat{p}_x\psi + \mathcal{N}\psi, \quad (5.2)$$

where $c_x = \pm\hbar\kappa/2m$ for some real wave number κ . The goal is to gradually transform Eq. (5.1) into Eq. (5.2) over a small region, so that outgoing waves can exit the domain of physical interest (or computed physical domain) without reflection. For this reason, we unify Eq. (5.1) and Eq. (5.2) by introducing a symbolically ideal fractional SE as

$$i\hbar\partial_t\psi = \frac{c_x}{(2mc_x)^\alpha}\hat{p}_x^\alpha\hat{p}_x\psi + \mathcal{N}\psi, \quad (5.3)$$

where \hat{p}_x^α is a complex-rotated, two-sided fractional differential operator [50, 98] with $0 \leq \alpha \leq 1$ which will be defined explicitly later. We will use the left- and right-handed variable order fractional Caputo derivatives proposed by Almeida et al. in [99], which may be expressed concisely as follows:

$$D_{x^\pm}^{\alpha(x)}f(x) = \frac{\pm 1}{\Gamma(1 - \alpha(x))} \int_0^\infty \frac{f'(x \mp \xi)}{\xi^{\alpha(x)}} d\xi. \quad (5.4)$$

Note that Eq. (5.4) reduces to the standard Caputo derivatives if α has no dependence on x , and that the Caputo derivative of a constant is zero [99–101]. For ease of notation, we define $\partial_{x^\pm}^\alpha = D_{x^\pm}^{\alpha(x)}$. The space-fractional derivative normally used for the fractional SE is the Riesz derivative [100, 102–104]; however, for the Riesz derivative, $\lim_{\alpha \rightarrow 1} {}^{\text{RZ}}D_x^{2\alpha} \neq \partial_x$ which makes it insufficient for our purposes.

To see how Eq. (5.3) works, let $\psi(x, t) = e^{i(kx - \omega t)}$ where k and ω may be real or complex. Suppose for now that α does not depend on x , then $\partial_{x^\pm}^\alpha \psi = (\pm ik)^\alpha \psi$ [100], where $z^\alpha = |z|^\alpha e^{i\alpha \arg(z)}$ and $\arg(z)$ represents the argument of the complex number z . Due to the existence of left- and right-handed fractional derivatives, there are left- and right-handed fractional momentum operators $\hat{p}_{x^\pm}^\alpha$, which we define as

$$\hat{p}_{x^\pm}^\alpha = (\mp i\hbar)^\alpha \partial_{x^\pm}^\alpha. \quad (5.5)$$

Under this definition, it holds that $\lim_{\alpha \rightarrow 1} \hat{p}_{x^\pm}^\alpha \psi = \hat{p}_x \psi$ and $\lim_{\alpha \rightarrow 2} \hat{p}_{x^\pm}^\alpha \psi = \hat{p}_x^2 \psi$. Also, it can be seen that $\hat{p}_{x^\pm}^\alpha \hat{p}_{x^\pm}^\beta \psi = \hat{p}_{x^\pm}^{\alpha+\beta} \psi$. In particular, we have the factorizations $\hat{p}_x^2 \psi = \hat{p}_{x^\pm}^{\alpha+1} \hat{p}_{x^\pm}^{1-\alpha} \psi = \hat{p}_{x^\pm}^\alpha \hat{p}_x \hat{p}_{x^\pm}^{1-\alpha} \psi$.

Approximate Analytical Solution. Suppose that, *a priori*, we know that $k_0 \approx k$, then it follows that $\hat{p}_{x^\pm}^{1-\alpha} \psi = (\mp i\hbar)^{1-\alpha} (\pm ik_0)^{1-\alpha} \psi$ and

$$\hat{p}_x^2 \psi \approx (\mp i\hbar)^{1-\alpha} (\pm ik_0)^{1-\alpha} \hat{p}_{x^\pm}^{\alpha+1} \psi, \quad (5.6)$$

where $\hat{p}_{x^\pm}^{\alpha+1} = \hat{p}_{x^\pm}^\alpha \hat{p}_x$. Now, by substituting the above equation into Eq. (5.1) where we let $\mathcal{V} = 0$, we have the fractional-order equation for a free particle given as follows:

$$i\hbar \partial_t \psi = \frac{1}{2m} (\mp i\hbar)^{1-\alpha} (\pm ik_0)^{1-\alpha} \hat{p}_{x^\pm}^\alpha \hat{p}_x \psi. \quad (5.7)$$

With the ansatz $\psi(x, t) = e^{i(kx - \omega t)}$, we solve Eq. (5.7) by obtaining the dispersion relation.

$$\begin{aligned} \hbar\omega\psi &= \frac{1}{2m} (\mp i\hbar)^{1-\alpha} (\pm ik_0)^{1-\alpha} \hat{p}_{x^\pm}^\alpha \hat{p}_x \psi \\ &= \frac{1}{2m} (\mp i\hbar)^2 (\pm ik_0)^{1-\alpha} \partial_{x^\pm}^\alpha \partial_x \psi \\ &= \frac{1}{2m} (\mp i\hbar)^{1-\alpha} (\pm ik_0)^{1-\alpha} \cdot (\mp i\hbar)^\alpha (\pm ik)^\alpha \cdot (-i\hbar)(ik) \psi \\ &= \mp \frac{\hbar^2}{2m} (\pm ik_0)^{1-\alpha} (\pm ik)^\alpha (ik) \psi \\ &= -\frac{\hbar^2}{2m} (\pm ik_0)^{1-\alpha} (\pm ik)^\alpha (\pm ik) \psi \end{aligned}$$

$$= -\frac{\hbar^2}{2m}(\pm i k_0)^{1-\alpha}(\pm i k)^{\alpha+1}\psi. \quad (5.8)$$

Assuming that ψ is nontrivial ($\psi(x, t) \neq 0$) we find the dispersion relations for each FMO.

$$\omega = -\frac{\hbar}{2m}(\pm i k_0)^{1-\alpha}(\pm i k)^{\alpha+1}. \quad (5.9)$$

Refining the dispersion relation, we obtain

$$\begin{aligned} \omega &= -\frac{\hbar}{2m}(\pm i k_0)^{1-\alpha}(\pm i k)^{\alpha+1} \\ &= -\frac{\hbar}{2m}|k_0|^{1-\alpha}e^{i(1-\alpha)\arg(\pm i k_0)}|k|^{\alpha+1}e^{i(\alpha+1)\arg(\pm i k)} \\ &= -\frac{\hbar}{2m}|k_0|^{1-\alpha}|k|^{\alpha+1}e^{i(\arg(\pm i k)+\arg(\pm i k_0))}e^{i\alpha(\arg(\pm i k)-\arg(\pm i k_0))}. \end{aligned} \quad (5.10)$$

This dispersion relation expands as

$$\omega = -\frac{\hbar}{2m}|k_0|^{1-\alpha}|k|^{\alpha+1}e^{i(\arg(\pm i k_0)+\arg(\pm i k))}e^{-i\alpha(\arg(\pm i k_0)-\arg(\pm i k))}. \quad (5.11)$$

Provided that k and k_0 are real, the previous equation splits into the following two cases:

$$\omega = \begin{cases} +\frac{\hbar}{2m}|k_0|^{1-\alpha}|k|^{\alpha+1}, & k k_0 \geq 0, \\ -\frac{\hbar}{2m}|k_0|^{1-\alpha}|k|^{\alpha+1}e^{\pm i\pi\alpha \operatorname{sign}(k)}, & k k_0 < 0, \end{cases} \quad (5.12)$$

where $\operatorname{sign}(\cdot)$ represents the signum function. This gives the solutions

$$\psi(x, t) = \begin{cases} e^{i(kx-\omega_0 t)}, & k k_0 \geq 0, \\ e^{i(kx+\omega_\alpha t)}e^{-rt}, & k k_0 < 0, \end{cases} \quad (5.13)$$

where $\omega_0 = \frac{\hbar}{2m}|k_0|^{1-\alpha}|k|^{\alpha+1}$, $\omega_\alpha = \omega_0 \cos(\pi\alpha)$ and $r = \pm\omega_0 \sin(\pi\alpha) \operatorname{sign}(k)$. Each of the momentum operators has an inherent directionality; therefore, in order to have the fractional momentum point in the correct direction, we need to ensure that k_0 is in the outward pointing direction normal to the boundary. This implies that we should choose $k_0 > 0$ on the right boundary and $k_0 < 0$ on the left boundary.

We will now consider the stability of the analytical solution. For now, we will only consider the right boundary. For reflected waves on the right boundary, $k < 0$; however, we have already chosen $k_0 > 0$, therefore, $kk_0 < 0$. Hence, the solution is only stable if $r \geq 0$. Since $\sin(\pi\alpha) \geq 0$ for $0 \leq \alpha \leq 1$, and $\text{sign}(k) = -1$ for reflected waves, we must choose the lower sign on r which implies that we should use the momentum operator \hat{p}_{x-} . Using a similar argument, we must choose \hat{p}_{x+} on the left boundary. Surprisingly, while we set out to find a stable method to gradually transition between Eq. (5.1) and Eq. (5.2) over a small region, it follows that any waves that are reflected inside the region are adsorbed exponentially with time.

Idealized Operator. To implement the unified equation Eq. (5.3) on a computed physical domain of $[a, b]$, we pad each side with a boundary region of length L in which $\alpha(x)$ is allowed to gradually vary which we refer to as a fractional momentum layer (FML). Another small padding region of size ε is required in which ψ follows the TBC (one-way wave equation). Fig. 2.1 is a diagram of this padding. We define the weight function $w^+(x)$ as follows: $w^- = 0$ for $x < a$, $w^- = 1/2$ for $a \leq x \leq b$, and $w^- = 1$ for $x > b$. Additionally, we define the weight function $w^+(x) = |1 - w^-(x)|$. For convenience, we define the kinetic energy operator $\hat{T} = \frac{1}{2m}\hat{p}_x^2$ and use Eq. (5.6) (with appropriate choices of k_0) to find a fractional approximation. To this end, let k_0^- and k_0^+ be choices of k_0 from Eq. (5.6) for the left and right boundaries, respectively, where we assume that $k_0^+ = -k_0^- = \kappa$ with $\kappa > 0$. We approximate the kinetic energy operator as

$$\begin{aligned}\hat{T}\psi &= \frac{1}{2m}\hat{p}_x^2\psi \\ &= \frac{1}{2m}(w^+\hat{p}_x^2 + w^-\hat{p}_x^2)\psi\end{aligned}$$

$$\begin{aligned}
&\approx \frac{1}{2m} (w^+ (-i\hbar)^{1-\alpha} (+ik_0^-)^{1-\alpha} \hat{p}_{x^+}^{\alpha+1} + w^- (i\hbar)^{1-\alpha} (-ik_0^+)^{1-\alpha} \hat{p}_{x^-}^{\alpha+1}) \psi \\
&= \frac{1}{2m} (w^+ (-i\hbar)^{1-\alpha} (-i\kappa)^{1-\alpha} \hat{p}_{x^+}^\alpha \hat{p}_{x^+} + w^- (i\hbar)^{1-\alpha} (-i\kappa)^{1-\alpha} \hat{p}_{x^-}^\alpha \hat{p}_{x^-}) \psi \\
&= \frac{1}{2m} (-i\kappa)^{1-\alpha} (w^+ (-i\hbar)^{1-\alpha} \hat{p}_{x^+}^\alpha + w^- (i\hbar)^{1-\alpha} \hat{p}_{x^-}^\alpha) \hat{p}_x \psi \\
&= -\frac{i\hbar}{2m} (-i\kappa)^{1-\alpha} (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha) \hat{p}_x \psi \\
&= -\frac{i\hbar \kappa^{1-\alpha}}{2m} e^{-i\pi(1-\alpha)/2} (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha) \hat{p}_x \psi \\
&= -\frac{i(\hbar \kappa)^{1-\alpha}}{2m} e^{-i\pi(1-\alpha)/2} \hbar^\alpha (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha) \hat{p}_x \psi \\
&= -\frac{(\hbar \kappa)^{1-\alpha}}{2m} e^{i\pi\alpha/2} \hbar^\alpha (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha) \hat{p}_x \psi
\end{aligned} \tag{5.14}$$

From above, a convenient definition for our idealized fractional momentum operator is

$$\hat{p}_x^\alpha = -\hbar^\alpha e^{i\pi\alpha/2} (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha), \tag{5.15}$$

with the limits $\lim_{\alpha \rightarrow 1} \hat{p}_x^\alpha = \hat{p}_x$ and $\lim_{\alpha \rightarrow 0} \hat{p}_x^\alpha = -(w^+ - w^-)$. Moreover, the kinetic energy operator can be written as $\hat{T} \approx \frac{c_x}{(2m c_x)^\alpha} \hat{p}_x^\alpha \hat{p}_x$ where $c_x = \hbar \kappa / 2m$. From this, we obtain the following variable-order fractional SE in Eq. (5.3) which can be expanded as

$$i\hbar \partial_t \psi = i \frac{\hbar^2}{2m} \kappa^{1-\alpha} e^{i\pi\alpha/2} (w^+ \partial_{x^+}^\alpha - w^- \partial_{x^-}^\alpha) \partial_x \psi + \mathcal{N} \psi, \tag{5.16}$$

where $\alpha = \alpha(x)$. This can be extended into 2D by considering weight functions w_x^\pm, w_y^\pm , wave numbers κ_x, κ_y , and fractional-orders α_x, α_y along each direction to obtain

$$i\hbar \partial_t \psi = i \frac{\hbar^2}{2m} (\kappa_x^{1-\alpha^x} e^{i\theta^x} \partial_x^{\alpha^x} \partial_x + \kappa_y^{1-\alpha^y} e^{i\theta^y} \partial_y^{\alpha^y} \partial_y) \psi + \mathcal{N} \psi, \tag{5.17}$$

where $\partial_x^{\alpha^x} = w_x^+ \partial_{x^+}^{\alpha^x} - w_x^- \partial_{x^-}^{\alpha^x}$, $\alpha^x = \alpha^x(x)$, $\theta^x = \pi \alpha^x / 2$, and so on. Note that superscript x on α and θ are not powers, but rather just indicate the x -direction¹. Similarly, one may extend the unified equation to 3D and higher dimensions as well as systems of SEs.

¹While the superscript notation seems strange at the moment—indeed denoting these as α_x and θ_x would be more pleasing to the eye—the chosen notation will help to prevent diacritical overload once we transition to the discrete case.

It is important to note that the ultimate goal of this research is not simply to solve the fractional SE as that (as well as similar research) has been done many times before [49–54]. Rather, the goal is to utilize variable-order fractional calculus to create purely computational techniques that more accurately and more efficiently solve the *standard* SE and NSE on unbounded domains. Moreover, this method may provide a way to develop novel boundary conditions for simulating other differential equations on unbounded domains.

5.2 1D Fractional FDTD Method

There are various iteratively explicit numerical methods for solving the SE [48–54], namely, spectral and pseudospectral methods [55–57], finite difference methods [58–70], space-time finite-element methods [71], quadrature discretization methods [72–75], and finite-difference time-domain (FDTD) methods [17, 18, 27, 75–81]. We choose to use the FDTD method because it is simple and explicit allowing for easy parallelization.

To develop the FDTD scheme with FML for solving Eq. (5.16), we begin by separating the equation into the real and imaginary components. We define the diffusion coefficient, and the fractional diffusion coefficient to be $\mu = \hbar/2m$ and $\mu_\alpha = \mu\kappa^{1-\alpha}$, respectively. Then we let $\psi = \psi_{\text{R}} + i\psi_{\text{I}}$ where ψ_{R} and ψ_{I} are real valued, and we let $\theta = \pi\alpha/2$. Finally, by defining the fractional operator $\mathbb{D}_x^{\alpha+1} = (w^+\partial_{x^+}^\alpha - w^-\partial_{x^-}^\alpha)\partial_x$ we obtain a compact expression for the real-valued coupled differential equations as follows:

$$(\partial_t - \mu_\alpha \cos(\theta)\mathbb{D}_x^{\alpha+1})\psi_{\text{R}} = -(\mu_\alpha \sin(\theta)\mathbb{D}_x^{\alpha+1} - \hbar^{-1}\mathcal{N})\psi_{\text{I}}, \quad (5.18a)$$

$$(\partial_t - \mu_\alpha \cos(\theta)\mathbb{D}_x^{\alpha+1})\psi_{\text{I}} = +(\mu_\alpha \sin(\theta)\mathbb{D}_x^{\alpha+1} - \hbar^{-1}\mathcal{N})\psi_{\text{R}}. \quad (5.18b)$$

Now, to obtain the discrete version of in Eq. (5.16), we let N be a natural number and let $x_0 = a - L - \varepsilon$, $h = \frac{b-a+2L+2\varepsilon}{N}$, and $x_i = a - L - \varepsilon + ih$, $i = 0, 1, \dots, N$. Additionally,

denote the number of points used in the physical domain as $N_0 = \frac{b-a}{h}$. We now use the L2 approximation for the Caputo fractional derivative given in [85]. To avoid Richardson-like instabilities in the numerical scheme, we must be careful in choosing our temporal evaluation points as we proceed [82]. We evaluate the left-hand sides at the points (x_i, t_n) and the right-hand sides at the points $(x_i, t_{n+1/2})$. This evaluation is given as

$$(\partial_t - \mu_\alpha \cos(\theta) \mathbb{D}_x^{\alpha+1}) \psi_{\text{R}}|_{x_i, t_n} = - (\mu_\alpha \sin(\theta) \mathbb{D}_x^{\alpha+1} - \hbar^{-1} \mathcal{N}) \psi_{\text{I}}|_{x_i, t_{n+1/2}}, \quad (5.19a)$$

$$(\partial_t - \mu_\alpha \cos(\theta) \mathbb{D}_x^{\alpha+1}) \psi_{\text{I}}|_{x_i, t_n} = + (\mu_\alpha \sin(\theta) \mathbb{D}_x^{\alpha+1} - \hbar^{-1} \mathcal{N}) \psi_{\text{R}}|_{x_i, t_{n+1/2}}. \quad (5.19b)$$

Using the first order forward difference approximation for the time derivative, and the L2 approximation for the fractional derivative we have. Then, the FDTD method for solving the 1D fractional-order SE is given by

$$(\nabla_t - \mu_\alpha \cos(\theta) \delta_x^{\alpha_i+1}) [\psi_{\text{R}}]_i^n = - (\mu_\alpha \sin(\theta) \delta_x^{\alpha_i+1} - \hbar^{-1} \mathcal{N}_i^{n+1/2}) [\psi_{\text{I}}]_i^{n+1/2}, \quad (5.20a)$$

$$(\nabla_t - \mu_\alpha \cos(\theta) \delta_x^{\alpha_i+1}) [\psi_{\text{I}}]_i^n = + (\mu_\alpha \sin(\theta) \delta_x^{\alpha_i+1} - \hbar^{-1} \mathcal{N}_i^{n+1/2}) [\psi_{\text{R}}]_i^{n+1/2}, \quad (5.20b)$$

where $[\psi_{\text{R}}]_n^i = \psi_{\text{R}}(x_i, t_n)$, $[\psi_{\text{I}}]_n^i = \psi_{\text{I}}(x_i, t_n)$, and $\delta_x^{\alpha_i+1}$ is fractional Laplacian operator is defined as

$$\delta_x^{\alpha_i+1} = w^+ \nabla_{\bar{x}}^{\alpha_i} \nabla_x + w^- \nabla_x^{\alpha_i} \nabla_{\bar{x}} \quad (5.21)$$

where $\nabla_x f_i = f_{i+1} - f_i$ is the first-order forward difference operator, $\nabla_{\bar{x}} f_i = f_{i+1} - f_i$ is the first-order backward difference operator, and the fractional operators are defined as

$$\nabla_{\bar{x}}^{\alpha_i} f_i := \frac{1}{\Gamma(2 - \alpha_i)} \sum_{k=0}^{\infty} \tilde{b}_k^{\alpha_i} f_{i-k} \quad (5.22)$$

$$\nabla_x^{\alpha_i} f_i := \frac{-1}{\Gamma(2 - \alpha_i)} \sum_{k=0}^{\infty} \tilde{b}_k^{\alpha_i} f_{i+k}. \quad (5.23)$$

where $\tilde{b}_0^{\alpha_i} = 1$, $\tilde{b}_1^{\alpha_i} = 2^{1-\alpha_i} - 2$, and $\tilde{b}_k^{\alpha_i} = (k+1)^{1-\alpha_i} - 2k^{1-\alpha_i} + (k-1)^{1-\alpha_i}$ for $k \geq 2$.

Now, Eq. (5.20) can be expanded out as

$$[\psi_R]_i^{n+1} - [\psi_R]_i^n = \tilde{\mu}_{\alpha_i}^c \delta_x^{\alpha_i+1} [\psi_R]_i^n - \tilde{\mu}_{\alpha_i}^s \delta_x^{\alpha_i+1} [\psi_I]_i^{n+1/2} + \tilde{\mathcal{N}}_i^{n+1/2} [\psi_I]_i^{n+1/2}, \quad (5.24a)$$

$$[\psi_I]_i^{n+1} - [\psi_I]_i^n = \tilde{\mu}_{\alpha_i}^c \delta_x^{\alpha_i+1} [\psi_I]_i^n + \tilde{\mu}_{\alpha_i}^s \delta_x^{\alpha_i+1} [\psi_R]_i^{n+1/2} - \tilde{\mathcal{N}}_i^{n+1/2} [\psi_R]_i^{n+1/2}, \quad (5.24b)$$

where $\tilde{\mu}_{\alpha_i}^c = \mu_{\alpha_i} \Delta t \cos(\pi\alpha_i/2)/h^{\alpha_i+1}$, $\tilde{\mu}_{\alpha_i}^s = \mu_{\alpha_i} \Delta t \sin(\pi\alpha_i/2)/h^{\alpha_i+1}$ and $\tilde{\mathcal{N}}_i = \Delta t \hbar^{-1} \mathcal{N}_i$.

Limiting Cases. Notice that when $\alpha_i \rightarrow 1$, we have $\delta_x^{\alpha_i+1} \rightarrow \delta_x^2$, $\tilde{\mu}_{\alpha_i}^c \rightarrow 0$ and $\tilde{\mu}_{\alpha_i}^s \rightarrow \tilde{\mu}$

where $\tilde{\mu} = \Delta t \mu / h^2$. Then, the fractional FDTD method becomes the standard FDTD method given as

$$[\psi_R]_i^{n+1} - [\psi_R]_i^n = - \left(\tilde{\mu} \delta_x^2 - \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_I]_i^{n+1/2}, \quad (5.25a)$$

$$[\psi_I]_i^{n+1} - [\psi_I]_i^n = + \left(\tilde{\mu} \delta_x^2 - \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_R]_i^{n+1/2}. \quad (5.25b)$$

Notice also that when $\alpha_i \rightarrow 0$, we have $\delta_x^{\alpha_i+1} \rightarrow \nabla_x$ on the right-side boundary, and $\delta_x^{\alpha_i+1} \rightarrow -\nabla_{\bar{x}}$ on the left-side boundary, and $\tilde{\mu}_{\alpha_i}^c \rightarrow 0$ and $\tilde{\mu}_{\alpha_i}^s \rightarrow \tilde{\mu} \kappa$. On the left-hand boundary, the fractional FDTD method becomes a downwind solution to a hyperbolic equation (Engquist-Majda one-way wave equation) given as

$$[\psi_R]_i^{n+1} - [\psi_R]_i^n = \left(-\tilde{\mu} \kappa \nabla_x - \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_I]_i^{n+1/2}, \quad (5.26a)$$

$$[\psi_I]_i^{n+1} - [\psi_I]_i^n = \left(-\tilde{\mu} \kappa \nabla_x - \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_R]_i^{n+1/2}. \quad (5.26b)$$

On the right-hand boundary, the fractional FDTD method becomes an upwind solution to a hyperbolic equation given as

$$[\psi_R]_i^{n+1} - [\psi_R]_i^n = \left(+\tilde{\mu} \kappa \nabla_{\bar{x}} + \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_I]_i^{n+1/2}, \quad (5.27a)$$

$$[\psi_I]_i^{n+1} - [\psi_I]_i^n = \left(+\tilde{\mu} \kappa \nabla_{\bar{x}} - \tilde{\mathcal{N}}_i^{n+1/2} \right) [\psi_R]_i^{n+1/2}. \quad (5.27b)$$

Using a similar argument, we may obtain a fractional FDTD scheme for the 2D unified equation in Eq. (5.17).

5.3 2D Fractional FDTD Method

In a similar fashion to the 1D case, we may obtain a fractional FDTD scheme for the 2D unified equation in Eq. (5.17) by splitting ψ into real and imaginary parts. In so doing, we obtain the following coupled equations

$$\begin{aligned} & \left(\partial_t - \mu_{\alpha^x} \cos(\theta_x) \mathbb{D}_x^{\alpha_x+1} - \mu_{\alpha^y} \cos(\theta_y) \mathbb{D}_y^{\alpha_y+1} \right) \psi_{\text{R}} \\ &= - \left(\mu_{\alpha^x} \sin(\theta_x) \mathbb{D}_x^{\alpha_x+1} + \mu_{\alpha^y} \sin(\theta_y) \mathbb{D}_y^{\alpha_y+1} - \hbar^{-1} \mathcal{N} \right) \psi_{\text{I}}, \end{aligned} \quad (5.28a)$$

$$\begin{aligned} & \left(\partial_t - \mu_{\alpha^x} \cos(\theta_x) \mathbb{D}_x^{\alpha_x+1} - \mu_{\alpha^y} \cos(\theta_y) \mathbb{D}_y^{\alpha_y+1} \right) \psi_{\text{I}} \\ &= - \left(\mu_{\alpha^x} \sin(\theta_x) \mathbb{D}_x^{\alpha_x+1} + \mu_{\alpha^y} \sin(\theta_y) \mathbb{D}_y^{\alpha_y+1} - \hbar^{-1} \mathcal{N} \right) \psi_{\text{R}}, \end{aligned} \quad (5.28b)$$

Similar to the 1D case, we obtain the 2D fractional FDTD method:

$$\begin{aligned} & \left(\nabla_t - \tilde{\mu}_{\alpha_i^x}^c \delta_x^{\alpha_i^x+1} - \tilde{\mu}_{\alpha_j^y}^c \delta_y^{\alpha_j^y+1} \right) [\psi_{\text{R}}]_{ij}^n \\ &= - \left(\tilde{\mu}_{\alpha_i^x}^s \delta_x^{\alpha_i^x+1} + \tilde{\mu}_{\alpha_j^y}^s \delta_y^{\alpha_j^y+1} - \mathcal{N}_{ij}^{n+1/2} \right) [\psi_{\text{I}}]_{ij}^{n+1/2}, \end{aligned} \quad (5.29a)$$

$$\begin{aligned} & \left(\nabla_t - \tilde{\mu}_{\alpha_i^x}^c \delta_x^{\alpha_i^x+1} - \tilde{\mu}_{\alpha_j^y}^c \delta_y^{\alpha_j^y+1} \right) [\psi_{\text{I}}]_{ij}^n \\ &= + \left(\tilde{\mu}_{\alpha_i^x}^s \delta_x^{\alpha_i^x+1} + \tilde{\mu}_{\alpha_j^y}^s \delta_y^{\alpha_j^y+1} - \mathcal{N}_{ij}^{n+1/2} \right) [\psi_{\text{R}}]_{ij}^{n+1/2}, \end{aligned} \quad (5.29b)$$

where $\tilde{\mu}_{\alpha_i^x}^c = \kappa_x^{1-\alpha_i^x} \mu \Delta t \cos(\theta^x) / h^{\alpha_i^x+1}$, $\alpha_i^x = \alpha^x(x_i)$, and so on ².

²While Eq (5.29) may seem a bit overwhelming, in practice, the 2D case is easy to implement after having already created numerical fractional derivative operators for the 1D case.

5.4 Discussion

We introduce this fractional FDTD method to address the three key disadvantages of the ABC method. By developing the fractional FDTD method, we have addressed the first disadvantages but introduced some others. Firstly, there is inherent reflection due to the interface of the G-FDTD and ABC methods. Not only does the fractional FDTD method smooth the transition between the two interior and boundary schemes by way of a fractional momentum layer (FML), but also we showed that any reflected waves inside the FML will decay rapidly with time. Secondly, the ABC requires tedious domain decomposition to implement. For the fractional FDTD method, all domain decomposition is handled by the fractional momentum operator. We have yet to address the choice of wave number needed as a parameter for the FML, nor how well it needs to be known for the method to be effective. This will be addressed through numerical experiments in the next chapter.

The disadvantages of the fractional FDTD scheme are as follows: (1) The FDTD scheme is nonlocal, therefore the computation times are longer for sequential computing; however, this can be mitigated through parallel computing. (2) Both stable fractional momentum operators and stable fractional FDTD methods are tedious to create; however, they are easy to implement. This is unlike the FDTD method with ABCs which are both tedious to create and tedious to implement. (3) There is not currently a way to implement the fractional FML with the higher-order Laplacian operator used in the G-FDTD method; however, that is not to say that such an operator is impossible.

5.5 Chapter Summary

In this chapter, we have derived an expression for a fractional momentum operator in 1D and 2D and we developed a fractional FDTD scheme for solving the linear and nonlinear Schrödinger equation on an unbounded domain. We also discussed the disadvantages of the ABC method and how the fractional FDTD method resolves some of those issues while introducing other disadvantages. Those disadvantages have been addressed, and remedies have been provided where applicable.

CHAPTER 6

NUMERICAL RESULTS FOR FRACTIONAL FDTD METHOD

In this chapter, we employ the FDTD scheme with FML to simulate the propagation of both solitons and particles. Before we can begin examples, there are several parameters for the FML that must be determined. Having determined those parameters we proceed with examples of soliton and particle propagation, and Gaussian packet collision.

6.1 Parameter Selection

Several parameters must be provided in order to use the fractional FDTD scheme. These parameters include the functional form of α , the value of the velocity parameter κ , and the size of the padding layers L and ε .

Choice of Function for α . As stated previously, there is no preferred choice for how α varies in the fractional region; however, we need to ensure that α is piecewise smooth, that is, α is at least once continuously differentiable. Let $\varepsilon, L > 0$, and consider the computational domain $[a - L - \varepsilon, b + L + \varepsilon]$ where $[a, b]$ is the domain of physical interest, $[a - L, a]$ and $[b, b + L]$ are the fractional regions, and $[a - L - \varepsilon, a - L]$ and $[b + L, b + L + \varepsilon]$ are regions where transparent boundary conditions are used. Then, the following function is piecewise smooth:

$$\alpha(x) = \begin{cases} 0, & x \leq a - L, \\ \sigma\left(\frac{a-x}{L}\right), & a - L < x < a, \\ 1, & a \leq x \leq b, \\ \sigma\left(\frac{x-b}{L}\right), & b < x < b + L, \\ 0, & x \geq b + L, \end{cases} \quad (6.1)$$

where $\sigma(x)$ is a smooth function satisfying $\sigma(0) = 1$ and $\sigma(1) = \sigma'(0) = \sigma'(1) = 0$. A visual breakdown of the different regions is shown in Fig. 2.1. While a linear function would be nice to use, the α generated from that linear function would not satisfy the piecewise smooth conditions. Instead, it is convenient to choose σ to be a monotonically decreasing sigmoid function. A few appropriate choices for σ are listed as follows:

1. (Cubic Spline). A cubic spline is uniquely determined by the piecewise smooth conditions ($\sigma_1(0) = 1, \sigma_1(1) = \sigma'_1(0) = \sigma'_1(1) = 0$) and is given as

$$\sigma(u) = 2u^3 - 3u^2 + 1. \quad (6.2)$$

This spline was primarily used for all of our simulations; however, there are more simple and sophisticated functional forms.

2. (Sinusoid). A sinusoidal curve can easily satisfy the piecewise smooth conditions, however, care should be taken to ensure the sinusoid is monotonic on the interval; hence, we use only half of the period. The sinusoidal sigmoid is as follow:

$$\sigma_2(u) = \cos^2\left(\frac{\pi x}{2}\right). \quad (6.3)$$

There is little difference between the cubic spline and the above sinusoid. In particular, their root-mean-squared difference is given by

$$\left(\int_0^1 (\sigma_1(u) - \sigma_2(u))^2 du\right)^{1/2} = \sqrt{\frac{69}{280} - \frac{24}{\pi^4}} \approx 0.006708$$

Moreover, both $\sigma_1(1/2) = \sigma_2(1/2) = 1/2$. If one wishes to change the half-way point for the spline, a more sophisticated function is required.

3. (Piecewise Quadratic Spline). Let q be a real number in the interval $(0, 1)$. Then, there is a family of two-region piecewise quadratic functions satisfying the piecewise smooth conditions needed for α . These functions are parameterized by q and are given as

$$\sigma_3(u; q) = \begin{cases} 1 - \frac{1}{q}x^2, & 0 \leq x \leq q, \\ \frac{1}{1-q}(x-1)^2, & q \leq x \leq 1. \end{cases} \quad (6.4)$$

The halfway point for this sigmoid is given by u_q , where

$$u_q = \begin{cases} 1 - \sqrt{(1-q)/2}, & q \leq 1/2, \\ \sqrt{q/2}, & q \geq 1/2, \end{cases} \quad (6.5)$$

so that $\sigma_3(u_q; q) = 1/2$. Notice that, if $q = 1/2$, then, $u_q = 1/2$ also. Looking at the limiting cases of the above equation where $q \rightarrow 0$ or $q \rightarrow 1$, it is clear to see that $|u_q - \frac{1}{2}| < \frac{\sqrt{2}-1}{2}$. Furthermore, the inflection point of the sigmoid occurs at $u = q$, therefore, the maximum $\max_{0 \leq u \leq 1} |\sigma'_3(u; q)| = |\sigma'_3(q; q)| = 2$.

Shown in Fig. 6.1 is a plot of the sigmoid functions $\sigma_1(u)$, $\sigma_2(u)$ and $\sigma_3(u; 1/2)$; however, it is not easy to distinguish the different sigmoids. To see the differences in the sigmoids, Fig. 6.2 shows the same plot, but the function $f(u) = 1 - u$ is subtracted from each sigmoid.

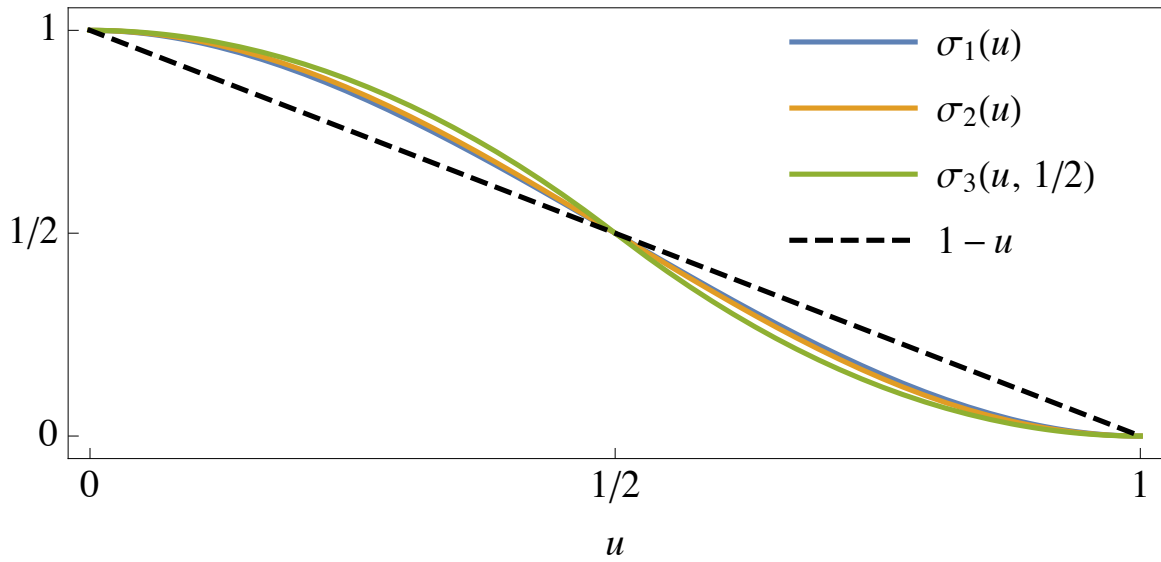


Figure 6.1: Plot of the sigmoid functions $\sigma_1(u)$, $\sigma_2(u)$, and $\sigma_3(u)$. The linear function that satisfies the continuity conditions, but not the piecewise smooth conditions, has been plotted as a black dashed line.

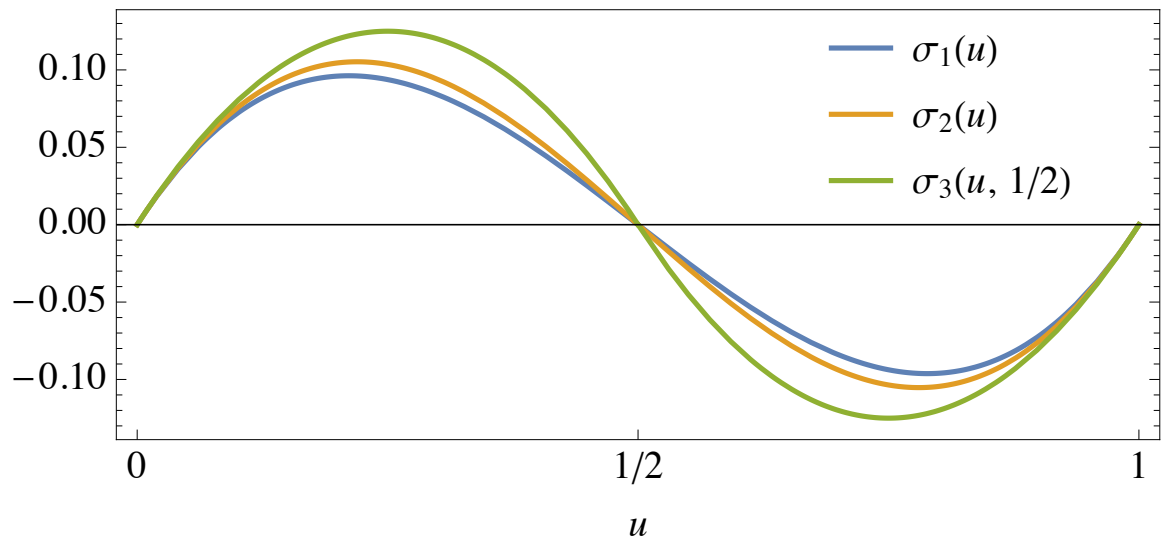


Figure 6.2: Plot of the sigmoid functions $\sigma_1(u)$, $\sigma_2(u)$, and $\sigma_3(u)$ where the function $f(u) = 1 - u$ has been subtracted from each sigmoid.

To show how the parameter q affects the sigmoid $\sigma_3(u; q)$, we have included Fig. 6.3 on page 88 which shows $\sigma_3(u; q)$ for various values of q . The limiting cases where $q = 1$ and $q = 0$ have also been plotted as red dashed lines.

While we have discussed three different choices of sigmoid, for simplicity, we used the cubic spline on both the left and right boundaries.

Value for ε . For the outermost padding regions $[a - L - \varepsilon, a - L]$ and $[b + L, b + L + \varepsilon]$, the value of ε only needs to be large enough so that the fractional operator has a sufficient number of function-values to be accurate. In our experience, $\varepsilon \approx 10h$ is sufficient.

Parameter Sweep. To determine appropriate choices for κ and L , we used the fractional FDTD scheme with FML to simulate 1D solitons of various momenta impacting FMLs of various widths. Specifically, we ran a sweep over the wavenumbers $1 \leq k \leq 10$, the velocity parameters $1 \leq \kappa \leq 10$, and the width parameters $1 \leq L \leq 8$. The grid spacing was chosen to be $h = 0.1$, and we chose $\varepsilon = 10h$. As our effectiveness metric, we used the reflection coefficient defined by

$$R_*(k; L, \kappa) = \frac{\int_a^b |\psi_*(x, t)|^2 dx}{\int_a^b |\psi_0(x, t)|^2 dx}, \quad (6.6)$$

where ψ_* is a stand-in for ψ_0 , ψ_{ABC} , or ψ_{FML} . Each function was calculated using the same initial conditions ψ_0 , ψ_{ABC} , or ψ_{FML} , but the methods were different. The wave function ψ_0 obtained by using the same initial conditions, but setting the boundary values to be zero, ψ_{ABC} was obtained by using the ABC method on the boundary, and ψ_{FML} was obtained by using the FML on the boundary. Similarly, R_* is a stand-in for R_0 , R_{ABC} , and R_{FML} . Notice that $R_0 = 1$ by definition.

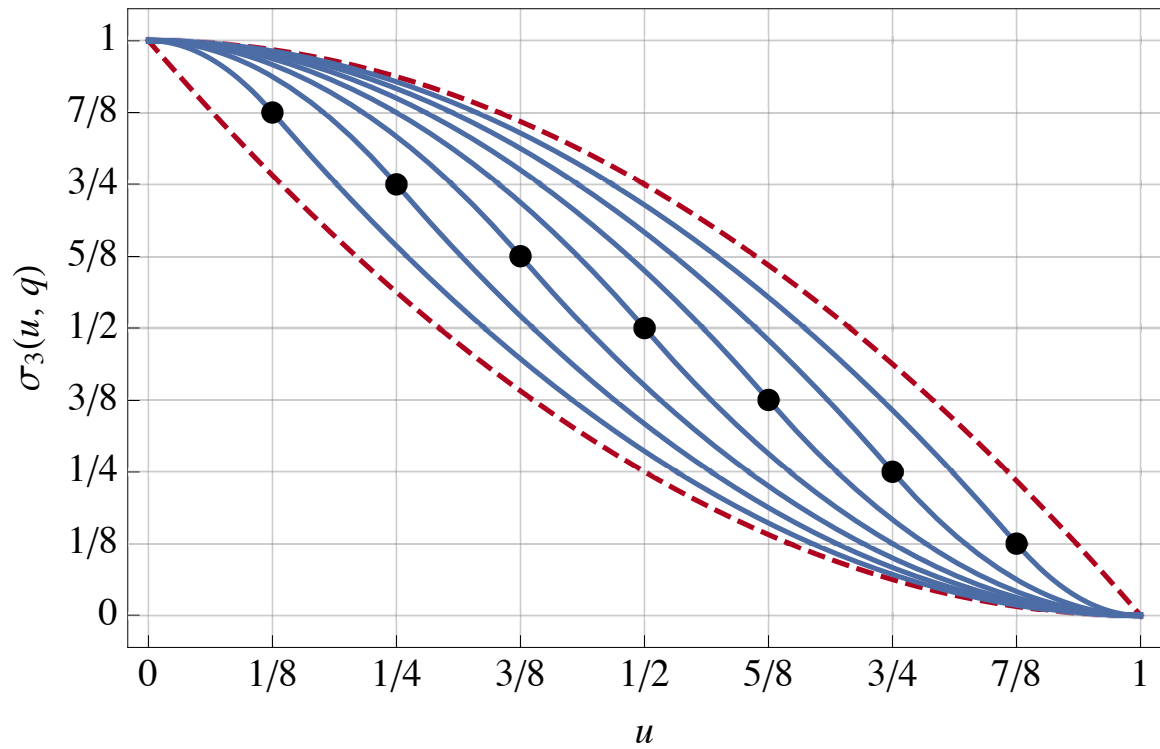


Figure 6.3: Plot of the sigmoid function $\sigma_3(u; q)$, for various parameters q . The inflection point for each sigmoid is given by the point $(q, 1 - q)$ and is labeled as a black dot. For each sigmoid, the parameter q is simply the u -coordinate of the inflection point.

For the sweep of numerical experiments described below, ψ_{FML} obtained by using the initial condition describes a soliton with a wavenumber of k and the simulation parameters were chosen to be κ and L . We performed a sweep over the intervals $k \in [0, 10]$, $\kappa \in [0, 10]$, and $L \in [0, 8]$. Of course, $R_*(k; L, \kappa)$ has a three-dimensional input, so it is difficult to visualize; therefore, for further analysis, we reduced the dimensionality to two by (1) taking the maximum over the parameter L , (2) taking the maximum over the parameter κ , (3) restricting the data to $\kappa = k$. After applying the restriction $\kappa = k$, the data no longer seemed to depend on the value of k ; therefore, we took a maximum over all values of k to obtain a function for the reflection coefficient R which only depends on L .

Test Case 1. To see how the parameter κ affects the absorption, we vary k and κ and take the maximum of the reflection coefficients produced by all lengths L that we considered. Fig. 6.4 shows the plot of $\max_{1 \leq L \leq 8} R(k; L, \kappa)$, where one can see that there is little effect on the absorption by κ . However, roughly, the best absorption coefficients can be gotten by setting $\kappa = k$. Additionally, the choice of $\kappa = k$ has physical significance for an incident wave of the form $\psi(x) = e^{ikx}$. Notice that the kinetic energy $k^{1-\alpha} \hat{p}_{x^\pm}^{\alpha+1} \psi = \pm k^2 \psi$ is conserved for all $0 \leq \alpha \leq 1$.

Test Case 2. To see how the parameter L affects the absorption, Fig. 6.5 shows the plot of the maximum $\max_{1 \leq \kappa \leq 10} R(k; L, \kappa)$. While there is no obvious choice of L , it is clear that increasing L will decrease the reflection coefficient. Considering that larger widths correspond to more gradual transitions from the Schrödinger equation to the one-way wave equation, this result is expected.

Test Case 3. If we limit the sweep data only to the slice where $\kappa = k$, shown in Fig. 6.6, it is easy to see that, for this choice, the reflection coefficient is nearly independent of the

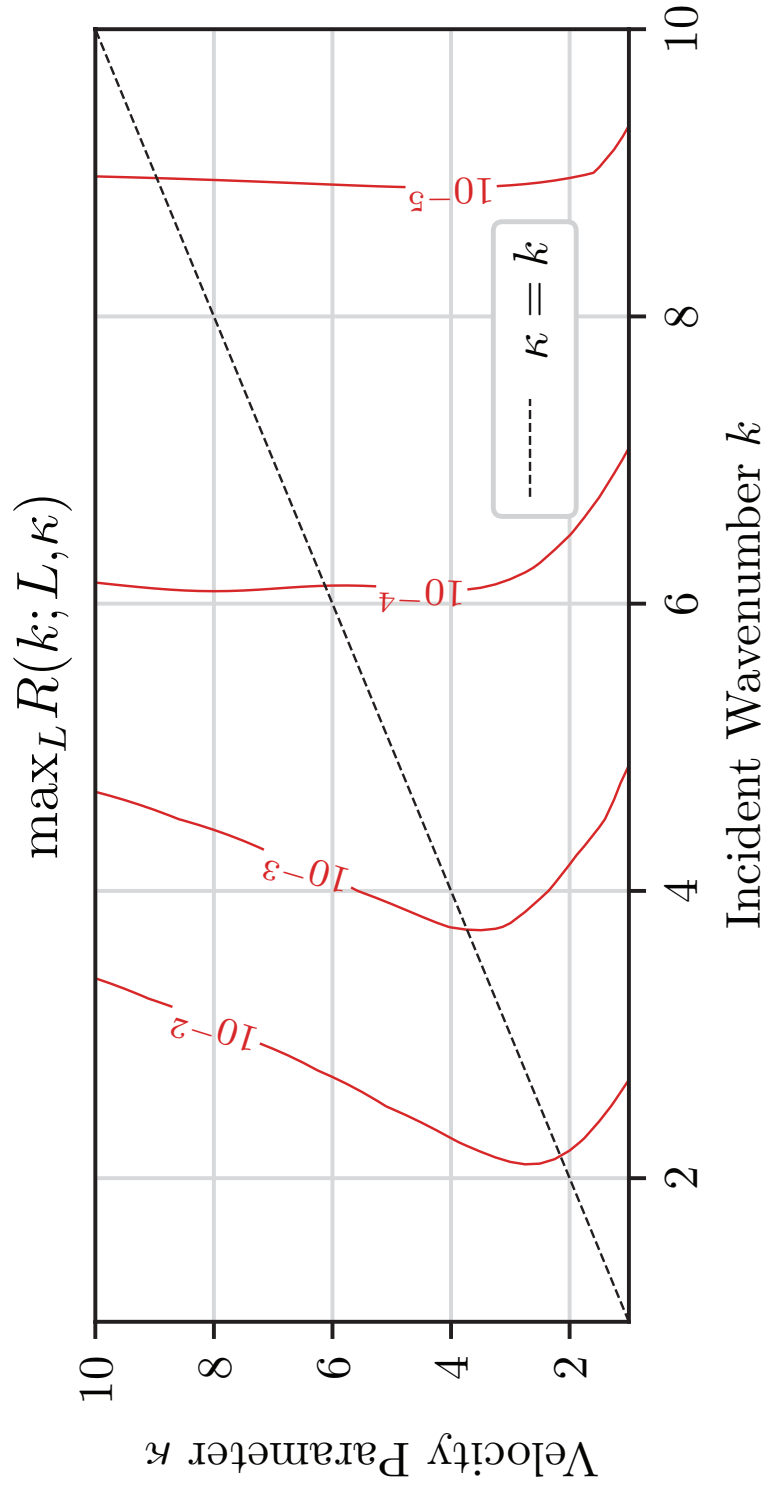


Figure 6.4: Plot of the reflection coefficient $R(k; L, \kappa)$ where the maximum was taken over the width parameter $1 \leq L \leq 8$.

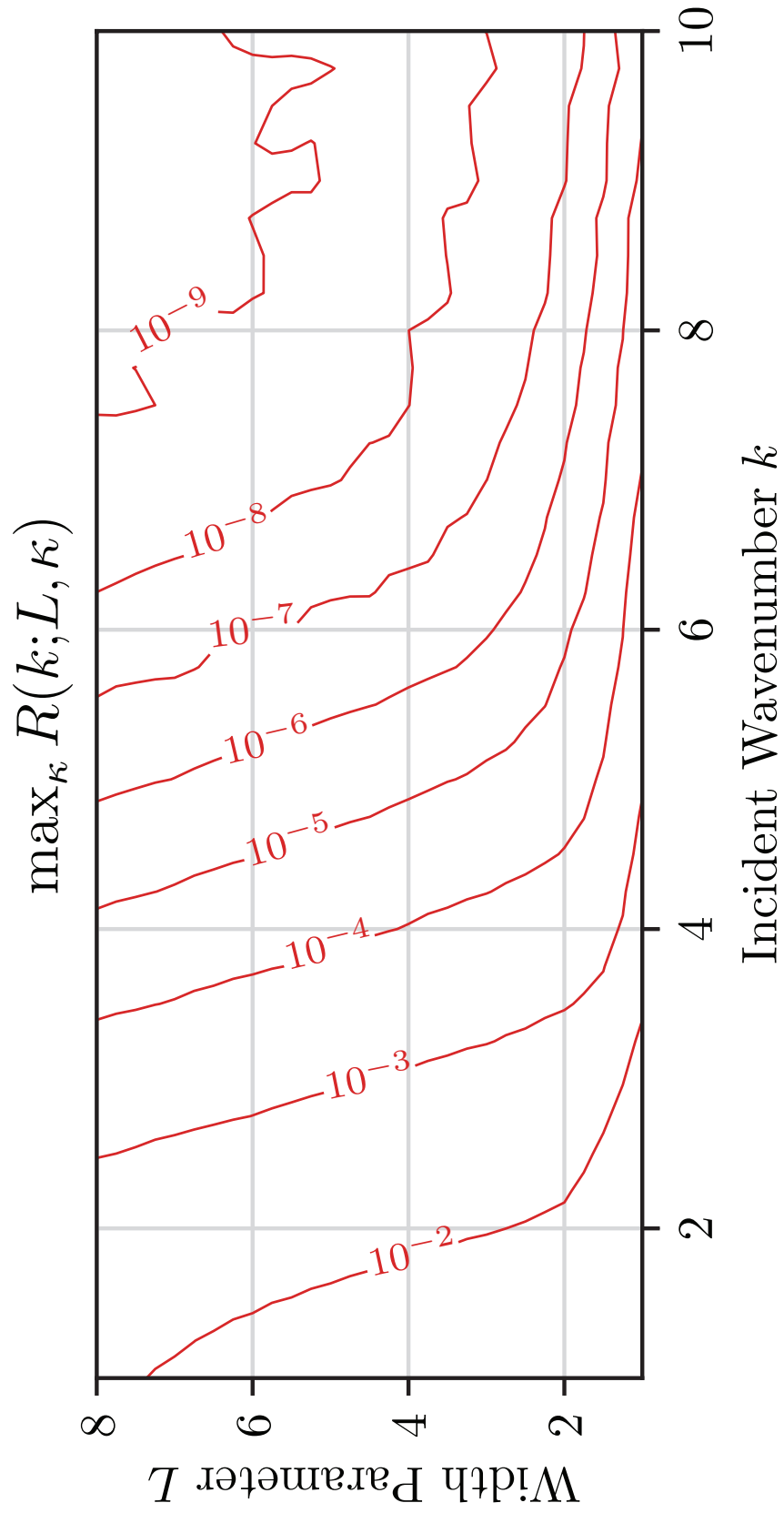


Figure 6.5: Contour plot of the reflection coefficient $R(k; L, \kappa)$ where the maximum was taken over velocity parameter $1 \leq \kappa \leq 10$.

wavenumber of the incident soliton. Fig. 6.7 shows the semi-log plot $\max_{1 \leq k \leq 10} R(k; L, \kappa)$ as a function of L . From this, we were able to see that there is an approximately linear dependence between the logarithm of the reflection coefficient and L .

Determining L . Using linear regression on the log-transformed data from Test Case 3, the fit roughly follows the equation $R = e^{2.045 - 3.003L}$ which we may solve for L to find a function for L in terms of R which can be expressed as

$$L_1(R) = 0.6812 - 0.7667 \cdot \log(R), \quad (6.7)$$

and if $R = \Delta x^2$. Hence, if the tolerance for global error due to reflection is R_{tol} , then L should be chosen such that $L = L_1(R_{\text{tol}})$. While the widths prescribed by Eq. (4.5) may be a relatively large percentage of the width physical, the FDTD scheme with FML is implemented on a GPU in our computation; therefore, the large width does not significantly increase the computational time.

6.2 Numerical Examples

This section presents 1D and 2D examples of the fractional FDTD method and compares the results to the exact solution (when possible), and the FDTD method with the ABCs presented in Chapter 3.

The 1D simulations were performed using Python with the standard linear algebra module Numpy. For the 2D simulations, we compare the FML with the recently developed ABC [44]. Each of the 2D simulations was performed on a GPU. For the simulations using the fractional FDTD method with FML, we used TensorFlow, a GPU-enabled tensor framework originally developed for machine learning [105]. However, due to the necessity

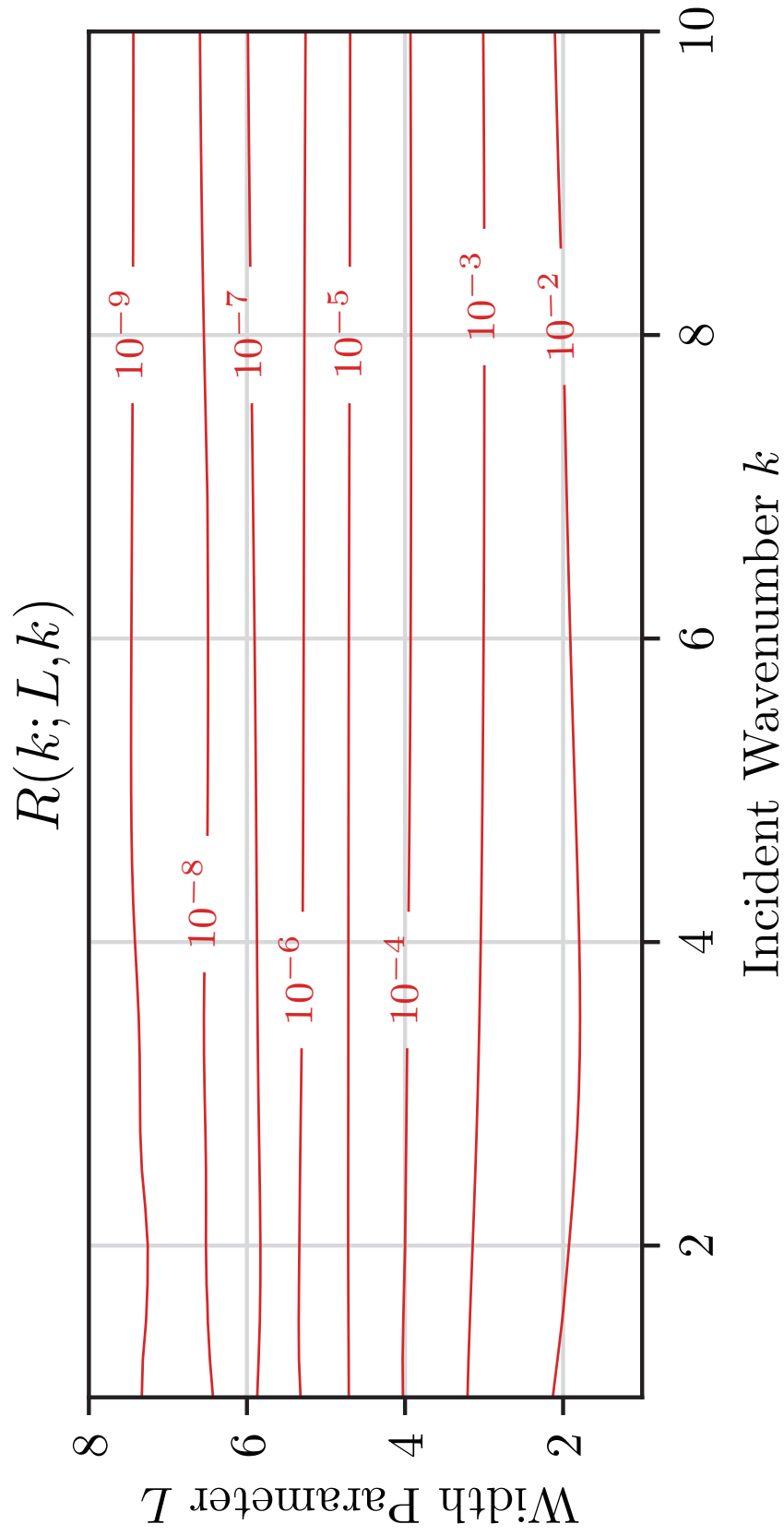


Figure 6.6: Contour plot of the reflection coefficient $R(k; L, \kappa)$ where the data was limited to the slice $\kappa = k$.

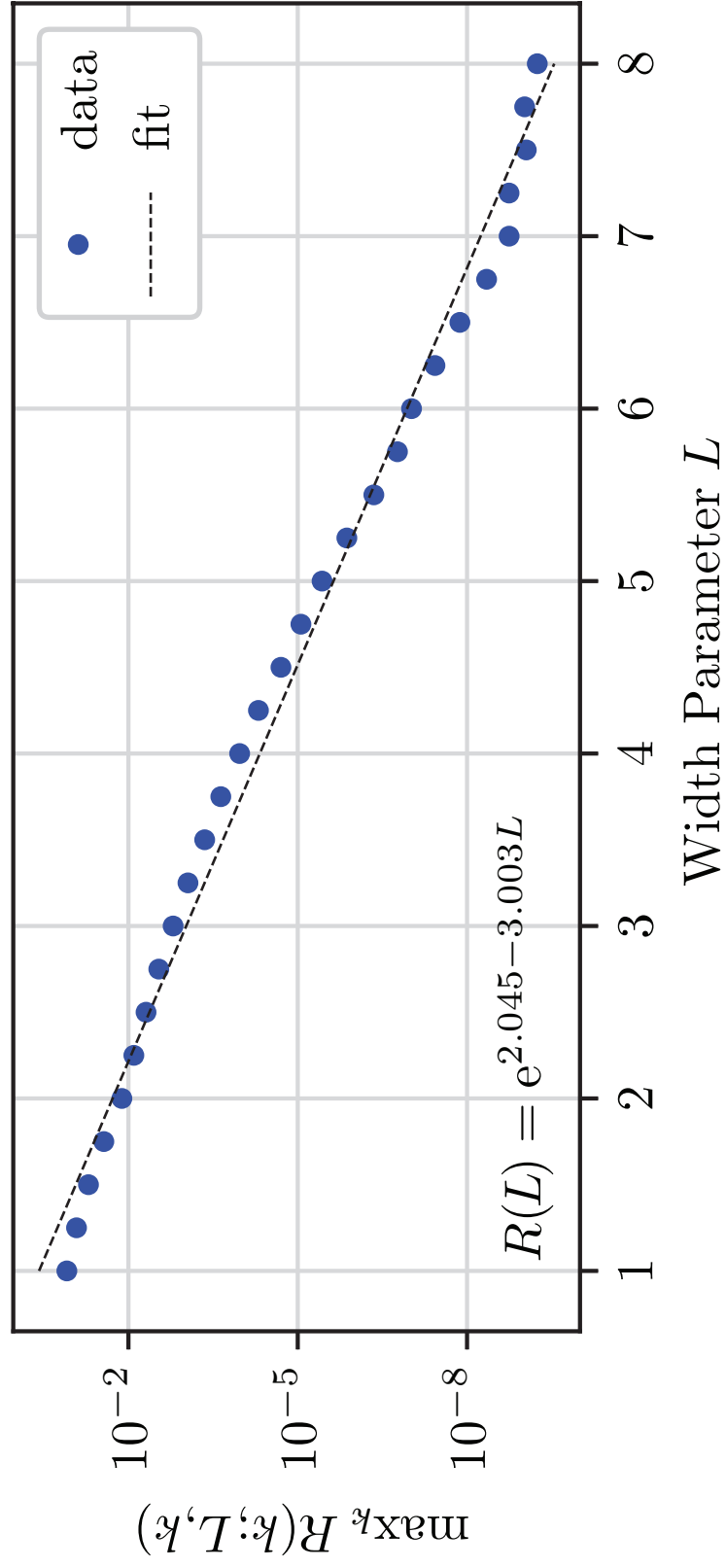


Figure 6.7: Contour plot of the reflection coefficient $R(k; L, k)$ where the maximum was taken over $k \in [0, 10]$.

of domain decomposition for the ABC, the FDTD method with ABC was more easily implemented using PyOpenCL [106].

6.3 Examples in 1D

Example 6.1 (NSE, 1D Soliton). Consider the NSE given by

$$i\partial_t\psi = \hat{p}_x^2\psi - 2|\psi|^2\psi, \quad (6.8)$$

and a 1D soliton propagating to the right with the initial conditions $\psi(x, 0) = \psi_0(x, 0)$ and $\psi(x, \Delta t/2) = \psi_0(x, \Delta t/2)$ where

$$\psi_0(x, t) = \text{sech}(x - 4t)e^{i(2x-3t)}, \quad (6.9)$$

and the physical domain is given by $-20 \leq x \leq 15.5$. These initial conditions describe a soliton moving to the right; therefore, to absorb this right-traveling wave, we will use the fractional FDTD scheme given by Eq. (5.20). Using what knowledge from the numerical experiments in the previous section, we determined the following parameters for the FML. By examining the wavenumber in Eq. (4.2), we let $\kappa = 2$. After deciding the tolerable order of reflection to be $R_{\text{tol}} = 10^{-4}$, we used the equation for L and rounded to choose $L = 4$. The mesh for this simulation was defined by $h = 0.05$ and $\Delta t = h^2/5$ and the outermost width was chosen to be $\varepsilon = 10h$. This gives the FML to be the interval $[15.5, 20]$ in our computation.

Fig. 6.8 shows $|\psi(x, t)|^2$ for a soliton where the fractional FDTD method with FML was employed as well as a plot of $\alpha(x)$. One can see from the figure that the soliton stays intact while it is in the physical domain ($t \leq 10$). At $t = 12$, even though the soliton began to enter the FML, there is no visible reflection at the interface. At $t = 14$, the soliton has

fully entered the FML. While there are clear distortions to the waveform, there are no visible reflections at the computational boundary. At $t = 16$, the wave continues to propagate out of the computational domain without reflection due to the transparent computation boundary. For $t > 18$, the wave is no longer inside the computational boundary, and, at $t = 20$, the reflection coefficient for this simulation was calculated to be $R = 5.6 \cdot 10^{-4}$ which is the same order as R_{tol} .

Example 6.2 (SE, 1D Particle). Consider the SE given by

$$i\partial_t\psi = \hat{p}_x^2\psi + V\psi \quad (6.10)$$

on the interval $[-20, 20]$ where the potential is defined by $V = 0$ if $x < 0$ and $V = 16$ if $x \geq 0$ and a 1D particle propagating to the right with initial conditions $\psi(x, 0) = \psi_0(x, 0)$ and $\psi(x, \Delta t/2) = \psi_0(x, \Delta t/2)$ where

$$\psi_0(x, t) = e^{-(x-2kt)^2} e^{i(kx-k^2t)}, \quad (6.11)$$

where wave number is given by $k = 5$. We consider the physical domain to be $[-20, 17.5]$. Since $V = 0$ on the left boundary, we let $\kappa = \kappa_- = k = 5$. Since $V = 16$ on the right boundary, we let $\kappa = \kappa_+ = \sqrt{k^2 - V} = 3$. The mesh for this simulation was defined by $h = 0.05$ and $\Delta t = h^2/10$. After deciding the order for the reflection coefficient to be $R_{\text{max}} = 10^{-3}$, we chose $L = 3$, and the width of the outermost layer was chosen to be $\varepsilon = 10h$. Hence, for this computation, the FML was implemented on the interval $(17.5, 19.5)$ and the TBC was used on the interval $[19.5, 20]$.

Shown in Fig. 6.9, we can see that the particle hits the potential barrier around $t = 2$. At $t = 6$, the particle begins to enter the FML and propagate outside the computational

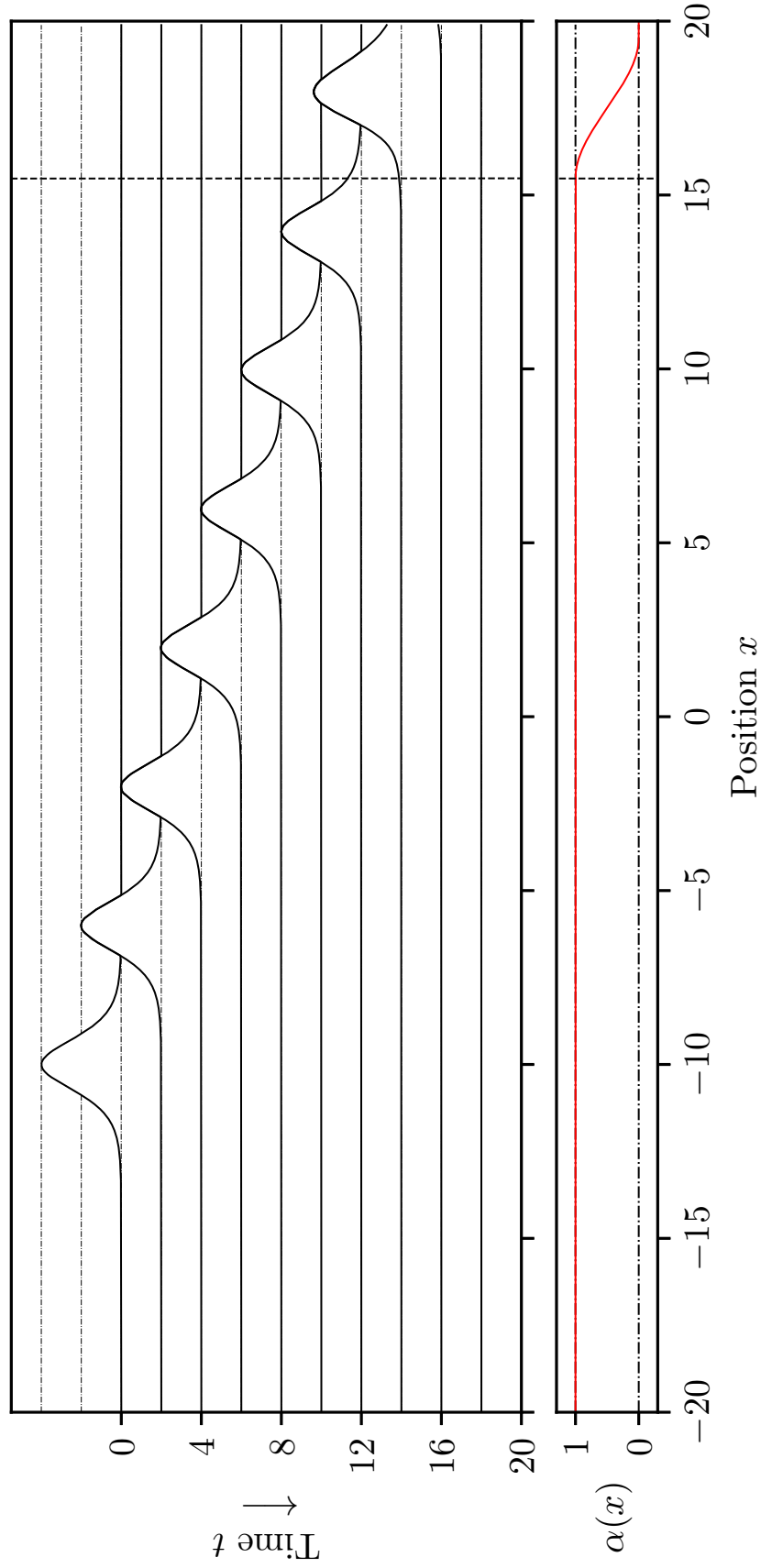


Figure 6.8: (Example 6.1). *Top:* Simulation of a 1D soliton traveling toward the right boundary using the fractional FDTD method with FML. *Bottom:* Plot of fractional order $\alpha(x)$.

domain. At $t = 20$, the reflection coefficient for this simulation was calculated to be $R = 8.4 \cdot 10^{-3}$ which is a similar order as R_{tol} .

6.4 Examples in 2D

In this section we present three distinct 2D examples.

Reflection in 2D. For 2D simulations, the results for $L_1(R_{\text{tol}})$ would only apply to 1D slices of the simulation. As such, the total *global* reflection in 2D is approximated by

$$R_{\text{tol}} = N_0 e^{2.045 - 3.003L}, \quad (6.12)$$

where $N_0 = \frac{b-a}{h}$. Letting R_{tol} be the expected reflection, we found that we should choose L such that $L = L_1(R_{\text{tol}}/N_0)$.

Example 6.3 (NSE, 2D Solitons). We considered the 2D Schrödinger equation

$$i\partial_t \psi = (\hat{p}_x^2 + \hat{p}_y^2)\psi + \lambda|\psi|^2\psi \quad (6.13)$$

where the physical domain is given by $[-20, 20] \times [-20, 20]$. We tried two different solutions both of which have initial conditions that describe solitons traveling towards the northeast corner. The first case is a soliton where the phase and group velocities are aligned. The second case is a soliton where the group and phase velocities are not aligned.

Case 1. For this case, we chose $\lambda = -4$, and the initial conditions were given by the equation $\psi(x, y, 0) = \psi_0(x, y, 0)$ and $\psi(x, y, \Delta t/2) = \psi_0(x, y, \Delta t/2)$ where

$$\psi(x, y, t) = \text{sech}(x + y - 5 - 8t) e^{2i(x+y-5-3t)}. \quad (6.14)$$

The mesh for this simulation was defined by $h = 0.1$ and $\Delta t = h^2/10$, and $\varepsilon = 10h = 1$.

We chose $\kappa_x = \kappa_y = 2$ to be the wave numbers along the x - and y -directions respectively.

Since $N_0 = 400$ for this simulation, by choosing $R_{\text{tol}} \approx 10^{-4}$ for our reflection tolerance

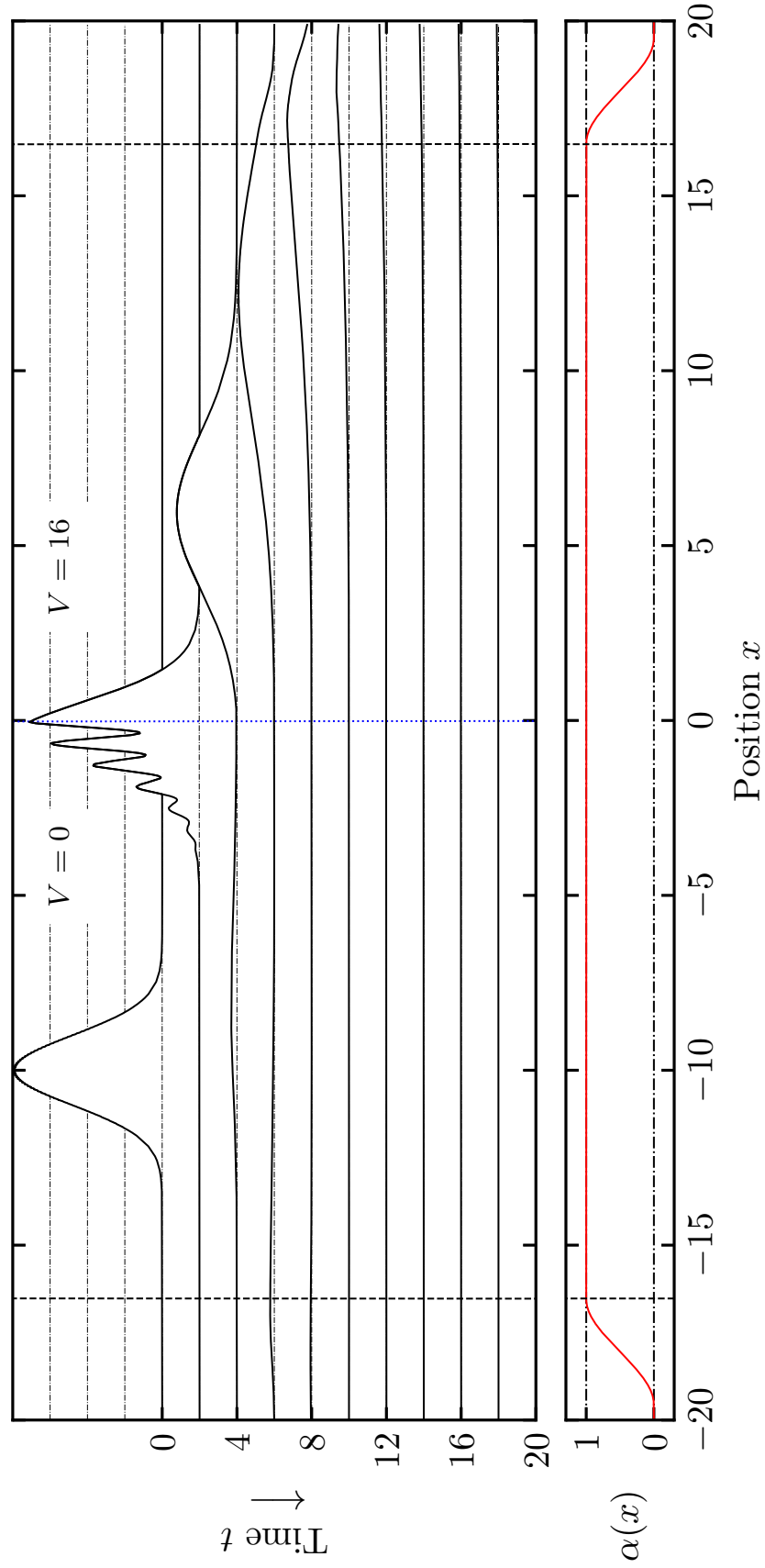


Figure 6.9: (Example 6.2). *Top:* Simulation of a 1D particle traveling to the right using fractional FDTD method with FML, where there is a step potential at $x = 0$. *Bottom:* Plot of fractional order $\alpha(x)$.

means that $L(R_{\text{tol}}/N_0) = 5.74$ which we will round up to choose $L = 6$. With $\varepsilon = 1$, this gave a computational domain of $[-27, 27] \times [-27, 27]$ where the FML is implemented in regions with $20 < |x| < 26$ or $20 < |y| < 26$.

The results for this example are shown in Fig. 6.10. Fig. 6.10a shows the exact solution limited to the physical domain where the wave propagated freely outside the computational domain. Fig. 6.10b shows the FDTD solution with ABC where $c_x = c_y = V_x = V_y = 4$, and Fig. 6.10c shows the FDTD solution with FML. At $t = 3$, the global reflection for the FDTD method with ABC was $R_{\text{ABC}} = 0.3277$, and, for the fractional FDTD method with FML, the reflection was $R_{\text{FML}} = 0.2930$.

Fig. 6.11 shows an enlarged snapshot at $t = 3$. From this figure, one sees that ABC does absorb the wave; however, there is a high-frequency reflection visible towards the back of the wave. This high-frequency reflection is not present when using the FDTD method with FML. While the wave is distorted when using the FML, this distortion is primarily confined to the FML and does not substantially reenter the physical domain interest. The distortion is because inside the FML the wave slows down in the normal direction causing refraction.

As shown in Fig. 6.12, we ran the FML simulation for a bit longer until $t = 14$ to show that the wave does, in fact, leave the computational domain. The final reflection coefficient for the fractional FDTD method with FML was calculated to be $R_{\text{FML}}|_{t=14} = 2 \cdot 10^{-3}$ which is slightly larger than R_{tol} . This is may be because purely 2D, nonlinear effects are not accounted for in the Eq. (6.12) because it was obtained based on an empirical formula.

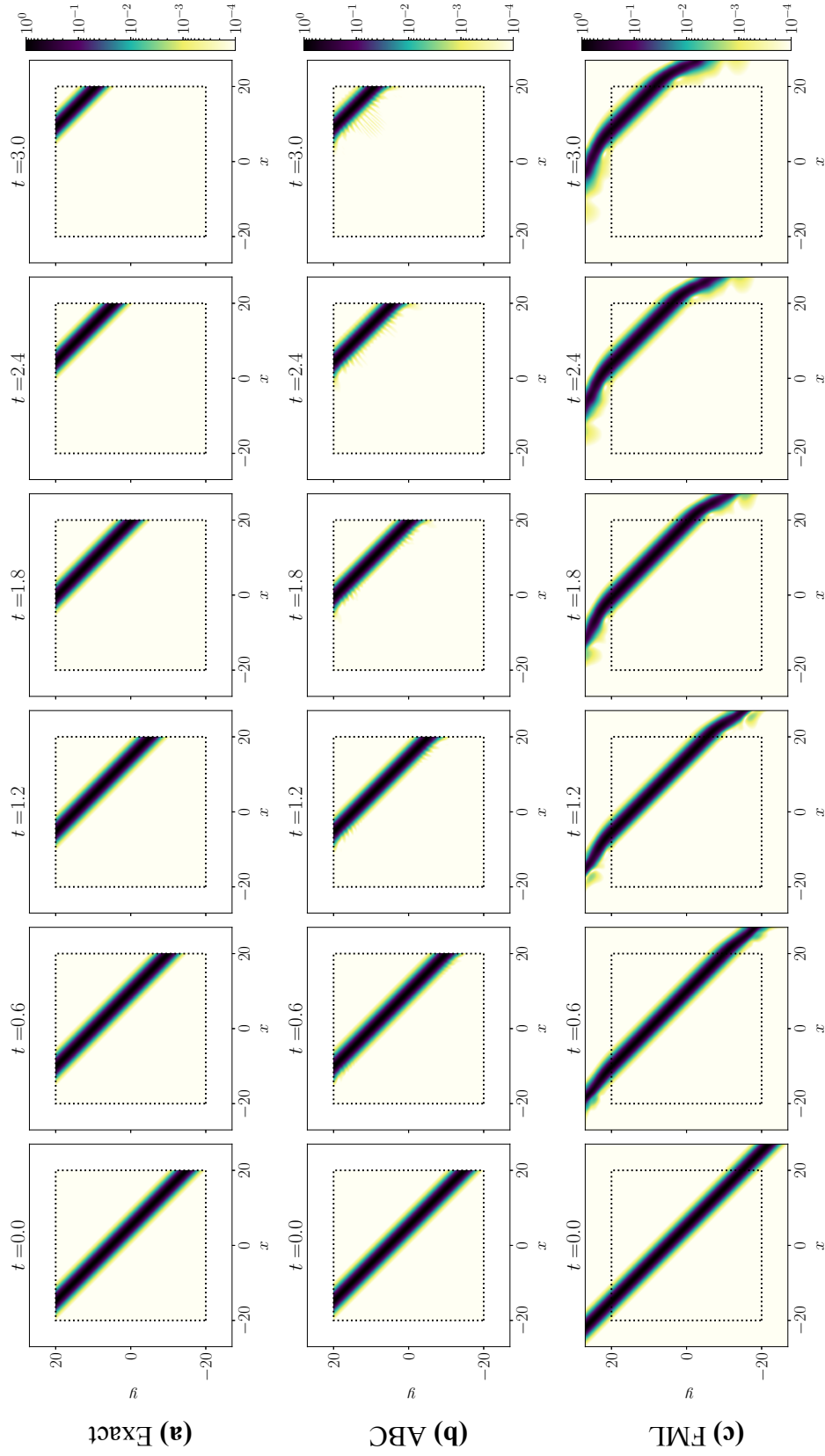


Figure 6.10: (Example 6.3, Case 1). Surface plot of $|\psi|^2$ of a 2D soliton using where ψ is determined by (a) the analytical solution, (b) the FDTD method with ABC, (c) the fractional FDTD method with FML. The interior of each dotted square represents the physical domain.

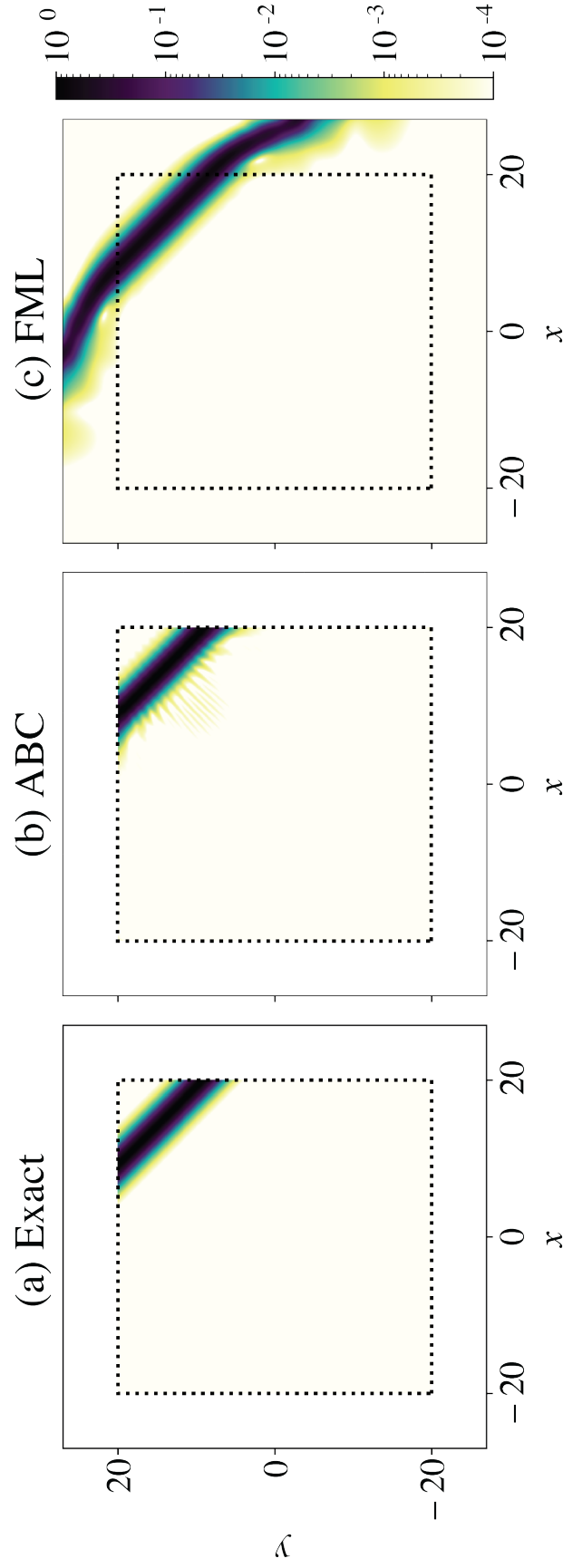


Figure 6.11: (Example 6.3, Case 1). Close up comparison of ABC method and fractional FDTD method with FML at $t = 3.0$. Notice the high-frequency distortions present when using the ABC method, but not present when using the FML method.

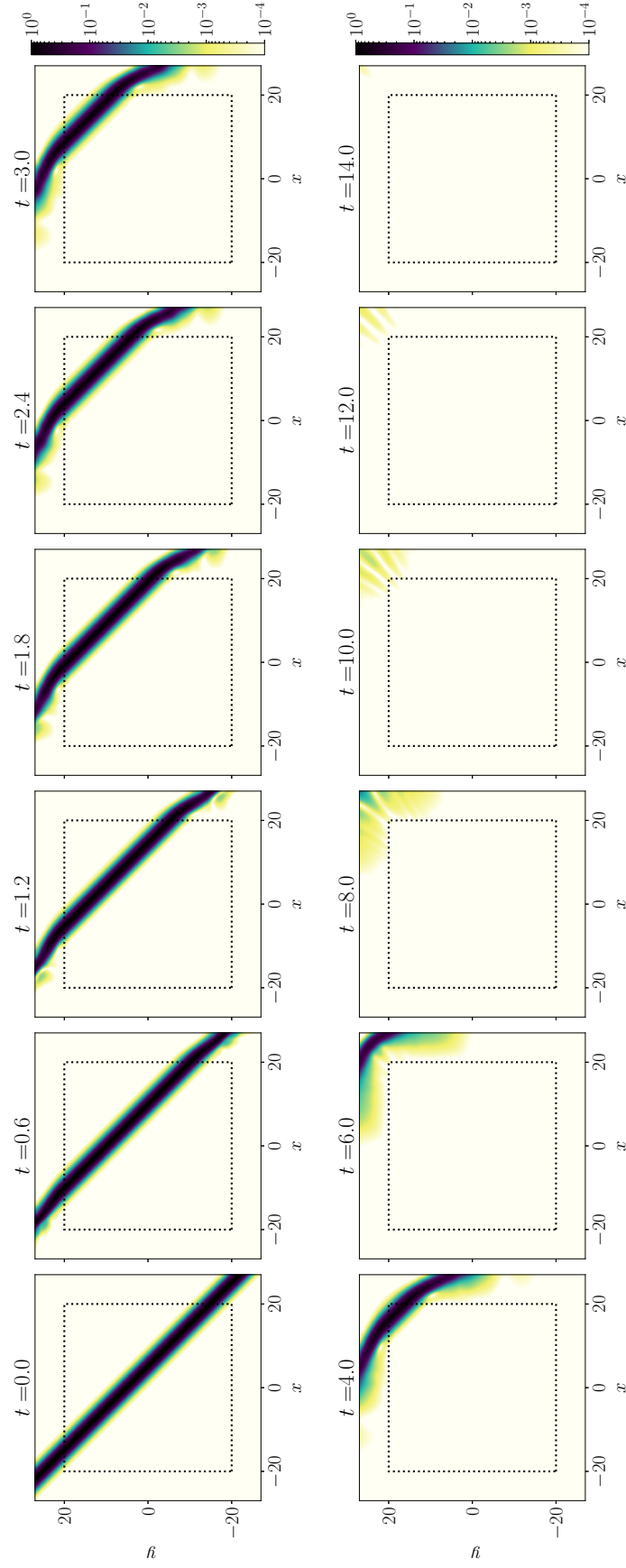


Figure 6.12: (Example 6.3, Case 1). The simulation where the FML was used was allowed to run longer to show that the wave was, in fact, absorbed.

Case 2. Now consider the NSE, but with $\lambda = -2$. For this case, the initial conditions were based on

$$\psi(x, y, t) = \operatorname{sech} \left(\frac{\sqrt{3}}{2}(x - 15) + \frac{1}{2}y - 2\sqrt{6}t \right) e^{i(\sqrt{2}(x-15) + \sqrt{6}y - 7t)}. \quad (6.15)$$

Unlike the previous initial conditions, these describe a soliton with group and phase velocities that are not aligned. For the simulation, we used the same mesh and padding regions as in Case 1. Since the soliton has different wave numbers for the x - and y -directions, we chose $\kappa_x = \sqrt{2}$ and $\kappa_y = \sqrt{6}$.

Fig. 6.13a shows the exact solution limited to the physical domain, where the wave propagated freely outside the computational domain. In Fig. 6.13b the FDTD method with ABC where $c_x = 2\sqrt{2}$, $c_y = 2\sqrt{6}$, $V_x = 2$, $V_y = 6$. Fig. 6.13c shows the fractional FDTD method with FML. At $t = 3$, one can see that the ABC solution produces reflection back into the physical domain which distorts of the soliton. However, the FML does not exhibit the same reflection. At $t = 5$, the solution using ABC reflects off the corner, while the solution using FML does not reflect back into the physical domain.

Example 6.4 (NSE, Collision of 2D Gaussian Packets). For the final example, we simulated the collision of two wave packets propagating under the NSE given as follows:

$$\partial_t \psi = (\hat{p}_x^2 + \hat{p}_y^2) \psi - 2|\psi|^2 \psi, \quad (6.16)$$

where the initial conditions were given by $\psi(x, y, 0) = \psi_0(x, y, 0)$ and $\psi(x, y, \Delta t/2) = \psi_0(x, y, \Delta t/2)$ with $\psi_0 = \psi_+ + \psi_-$ and

$$\psi_{\pm}(x, y, t) = \frac{1}{\sqrt[4]{9\pi}} e^{-((x \pm 10)^2 + y^2)/18} e^{i(\mp kx - k^2 t)}. \quad (6.17)$$

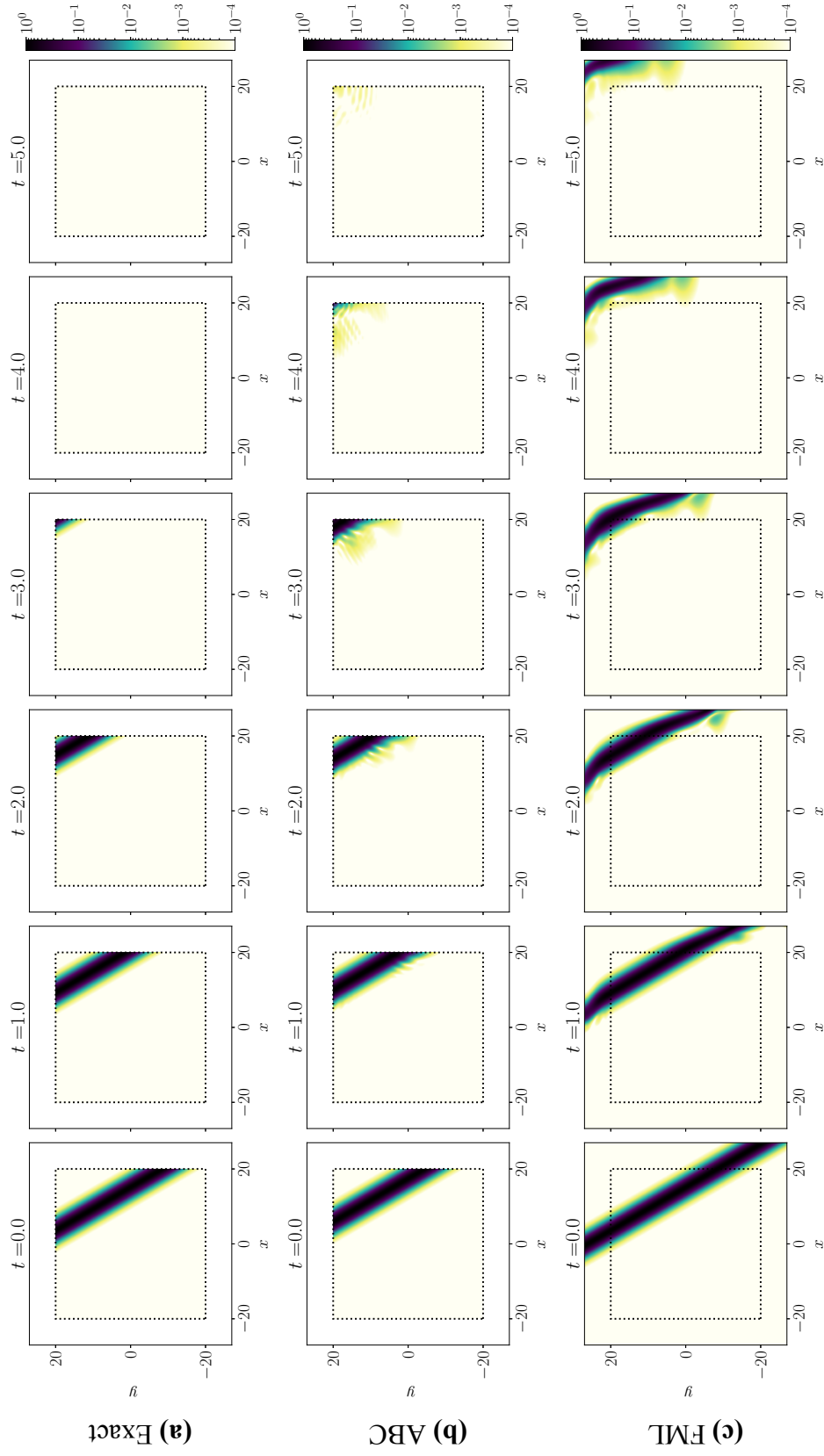


Figure 6.13: (Example 6.3, Case 2). Surface plot of $|\psi|^2$ of a 2D soliton using where ψ is determined by (a) the analytical solution, (b) FDTD method with ABC, (c) fractional FDTD method with FML. The interior of each black-dotted square represents the physical domain.

Here, the wavenumber was chosen to be $k = 4$, and the physical domain was given by $[-20, 20] \times [-20, 20]$. The mesh for this simulation was defined by $h = 0.1$ and $\Delta t = h^2/10$, and we let $\varepsilon = 10h = 1$. We chose $\kappa_x = \kappa_y = k$. The width parameter was chosen to be $\delta = 3$ from which Eq. (4.11) predicts a reflection coefficient $R_{\max} \approx 0.3$. Given ε and δ as described, the computational domain is $[-24, 24] \times [-24, 24]$.

The results for this example are shown in 6.14. Fig. 6.14a shows the plot of the solution where boundary values are set to be zero. Fig. 6.14b shows the solution obtained based on the FDTD method with ABC where $c_x = c_y = V_x = V_y = 4$. Fig. 6.14c shows the solution obtained based on the fractional FDTD method with FML. For all cases, the solution is similar up to $t = 2.4$ since the wave packets have not had time to reflect off the computational boundary. After $t = 3.6$, the solutions begin to differ. One can see that the solution with boundaries set to zero allows the entire wave to be reflected back into the computational domain. The FDTD method with ABC allows the majority of the wave to propagate freely; however, there is a very small amount of reflection back into the physical domain. The fractional FDTD method with FML shows no visible reflection at this scale. The reflection coefficients were calculated to be $R_{\text{ABC}}|_{t=6} = 2 \cdot 10^{-3}$ and $R_{\text{FML}}|_{t=6} = 2 \cdot 10^{-6}$. The value of R_{FML} is five orders of magnitude smaller than what is predicted by our 2D calculation for L . Again, this is likely due to the because the equation was calculated empirically.

6.5 Discussion

Not only does the fractional FDTD method with FML outperform the FDTD method with ABC, but also from Fig. 6.6 it appears that one may increase the size of L to obtain an

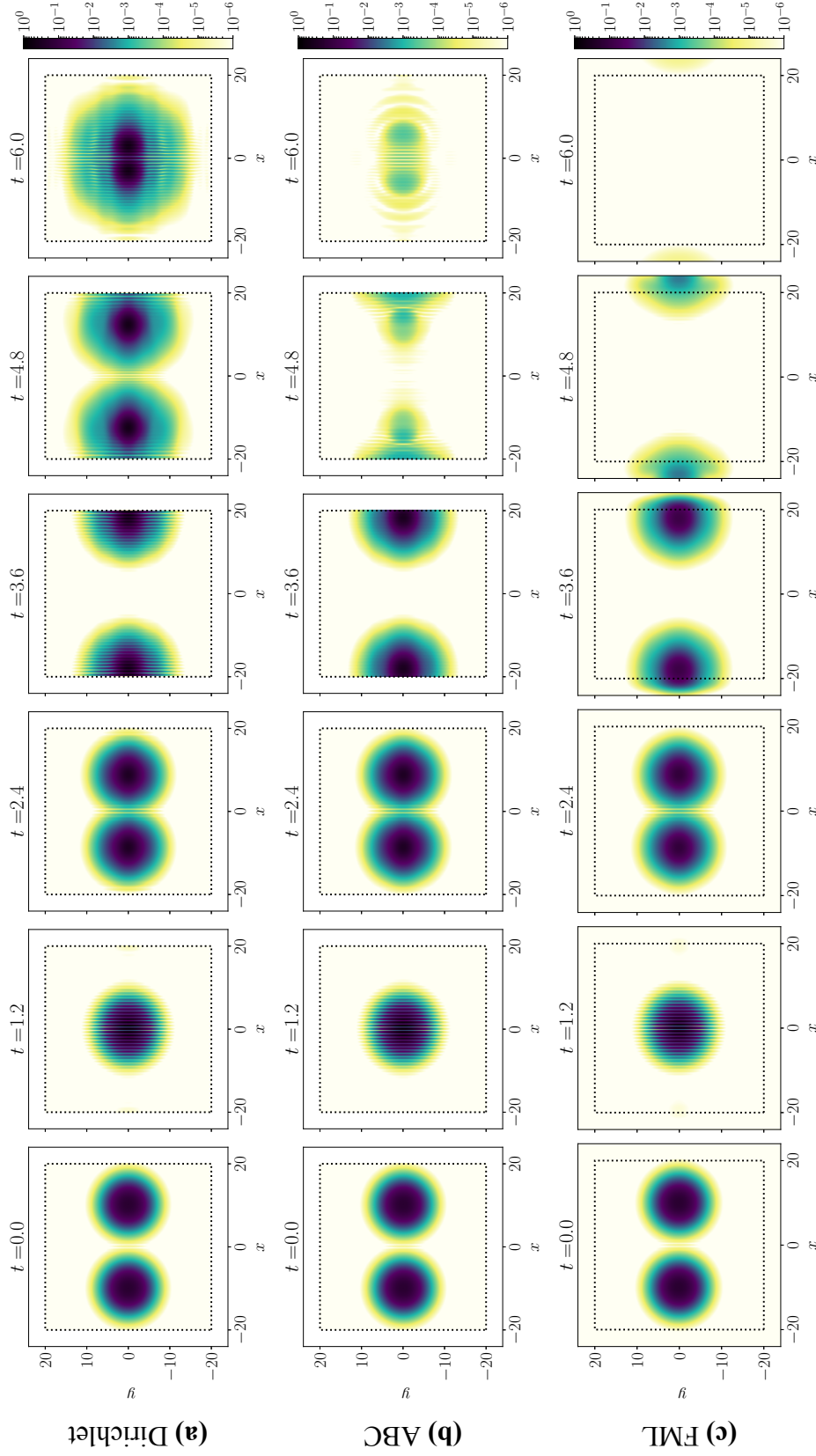


Figure 6.14: (Example 6.4). Plot of $|\psi|^2$ for two 2D Gaussian packets colliding where ψ is determined by (a) the FDTD method where function values outside the domain of physical interest are set to be zero, (b) FDTD method with ABC, (c) fractional FDTD method with FML. The interior of each black-dotted square represents the physical domain.

arbitrarily small reflection coefficient. Moreover, from Fig. 6.4, for solitons with larger momenta k , the reflection coefficient is not heavily dependent on the particular choice of κ as long as κ is sufficiently large. This addresses the final key disadvantage of the ABC method. For each method, the FDTD method with ABC provided larger reflection coefficients, than the fractional FDTD method with FML. Notably, for simulation with the 2D Gaussian packets, the reflection coefficient for the FDTD method with ABC was 1000 times larger than the reflection coefficient for the fractional FDTD method with FML.

6.6 Chapter Summary

In this chapter, discussed methods to determine the parameters for the fractional FDTD method with FML. We showed that the fractional FDTD method with FML provides a solution to the disadvantage FDTD method with ABC discussed in the beginning and end of Chapter 5. We also tested the method with several numerical examples including 1D particle propagation, 1D and 2D soliton propagation, as well as the collision of Gaussian packets in 2D.

CHAPTER 7

CONCLUSION

In this dissertation, we have addressed the problem of using the FDTD method and G-FDTD on unbounded domains. To do so, we developed two boundary methods for solving the linear and nonlinear Schrödinger equation. The first method is an ABC based on the Engquist-Majda one-way wave equations. While this method performs well and can be made to be adaptive, there are three key disadvantages to the using the ABC method: (1) there is inherent reflection due to the interface of two differential equations, (2) the ABC requires domain decomposition which is tedious to develop and implement especially in higher dimensions, and (3) one must guess the wave number of the incoming wave fairly accurately, or there can be substantial reflection at the boundary.

To address these challenges, the second method introduces a fractional-order momentum operator, from which a fractional FDTD scheme can be developed. By allowing the order of the fractional momentum to vary gradually over a fractional momentum layer so that the SE is transformed into its associated one-way wave equations, we smooth the transition between the interface which addresses the first issue. The second issue is addressed by the fractional momentum operator itself. Any domain decomposition is handled within the operator, and is mainly determined by the fractional order, therefore, when implementing the fractional FDTD method, there is no need to consider each boundary edge.

The third issue is addressed by considering the effects the fractional Schrödinger equation has on outgoing waves versus incoming waves. If constructed carefully, outgoing waves are allowed to pass through the FML freely and exit the simulation, while incoming waves are absorbed exponentially with time. Therefore, as long as the transition is gradual enough—that is, the width of the FML is chosen to be sufficiently large—any reflected portions of outgoing waves will decay before reentering the domain of physical interest. In some cases, we found that the fractional FDTD method with FML would reduce the reflection coefficient by a factor of 1000 as compared to using the FDTD method with ABC.

Future research into this topic may include exploring a generalized fractional FDTD method with FML, as well as other methods to improve the accuracy simulations such as fractional fourth-order central difference operator like the one introduced in Eq. (2.28). More broadly, this fractional FDTD method with FML provides a novel approach to creating boundary conditions to solve partial differential equations on unbounded domains. It may be fruitful to explore the application of this method to other wave equations such as Maxwell's equations for electromagnetism or the Klein-Gordon and Dirac equations.

REFERENCES

- [1] D. H. Peregrine. Two-dimensional superfluid flows in inhomogeneous Bose-Einstein condensates. *J. Austral. Math. Soc. Ser. B*, 25:16–43, 1983.
- [2] C. Sulem and P.-L. Sulem. *The Nonlinear Schrödinger Equation: Self-Focusing and Wave Collapse*, volume 139 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1999.
- [3] Z. Yan, V. V. Konotop, A. V. Yulin, and W. M. Liu. Two-dimensional superfluid flows in inhomogeneous Bose-Einstein condensates. *Phys. Rev. E*, 85:016601, 2012.
- [4] A. G. Kalocsai and J. W. Haus. Nonlinear Schrödinger equation for optical media with quadratic nonlinearity. *Phys. Rev. A*, 49:574–585, 1994.
- [5] Y. Shi, A. E. Borovik, and J. E. Hearst. Elastic rod model incorporating shear and extension, generalized nonlinear Schrödinger equation, and novel closed-form solutions for supercoiled DNA. *J. Phys. Chem.*, 103:3166–3183, 1995.
- [6] P. Ao, D. J. Thouless, and X.-M. Zhu. Nonlinear Schrödinger equation for superconductors. *Mod. Phys. Lett. B*, 9:755–761, 1995.
- [7] S. F. Mingaleev, P. L. Christiansen, Y. B. Gaididei, M. Johansson, and K.Ø . Rasmussen. Models for energy and charge transport and storage in biomolecules. *J. Biol. Phys.*, 25:41–63, 1999.
- [8] T. C. Bishop, R. Cortez, and O. O. Zhmudsky. Investigation of bend and shear waves in a geometrically exact elastic rod model. *J. Comput. Phys.*, 193:642–665, 2004.
- [9] M. Ablowitz, I. Bakirtas, and B. Ilan. On a class of nonlocal nonlinear Schrödinger equations and wave collapse. *Eur. Phys. J. Spec. Top.*, 147:343–362, 2007.
- [10] A. Chabchoub, N. Hoffmann, M. Onorato, and N. Akhmediev. Super rogue waves: Observation of a higher-order breather in water waves. *Phys. Rev. X*, 2:011015, 2012.
- [11] N. K. Vitanov, A. Chabchoub, and N. Hoffmann. Deep-water waves: on the nonlinear Schrödinger equation and its solutions. *J. Theoret. Appl. Mech.*, 43:43–54, 2013.
- [12] W.-M. Liu and E. Kengne. *Schrödinger Equations in Nonlinear Systems*. Springer Nature, Singapore, 1st edition, 2019.

- [13] W. Yu, W. Liu, H. Triki, Q. Zhou, A. Biswas, and M. R. Belić. Control of dark and anti-dark solitons in the (2+1)-dimensional coupled nonlinear Schrödinger equations with perturbed dispersion and nonlinearity in a nonlinear optical system. *Nonlinear Dyn.*, 97:471–483, 2019.
- [14] W. J. Sonnier and C. I. Christov. Strong coupling of Schrödinger equations: Conservative scheme approach. *Math. Comp. Simulat.*, 69:514–525, 2005.
- [15] W. Bao. Ground states and dynamics of multicomponent Bose–Einstein condensates. *SIAM Multiscale Model. Simul.*, 2:210–236, 2004.
- [16] R. M. Caplan and R. Carretero-González. A modulus-squared Dirichlet boundary condition for time-dependent complex partial differential equations and its application to the nonlinear Schrödinger equation. *SIAM J. Sci. Comput.*, 36:A1–A19, 2014.
- [17] D. M. Sullivan. *Electromagnetic Simulation Using the FDTD Method*. Wiley-IEEE Press, Hoboken, NJ, 2nd edition, 2013.
- [18] F. I. Moxley, III, D. Chuss, and W. Dai. A generalized finite-difference time-domain scheme for solving nonlinear Schrödinger equations. *Comput. Phys. Commun.*, 184:1834–1841, 2013.
- [19] I. Alonso-Mallo and N. Reguera. Discrete absorbing boundary conditions for Schrödinger-type equations: Construction and error analysis. *SIAM J. Numer. Anal.*, 41:1824–1850, 2003.
- [20] I. Alonso-Mallo and N. Reguera. Discrete absorbing boundary conditions for Schrödinger-type equations: Practical implementation. *Math. of Comput.*, 73:127–142, 2003.
- [21] J. Szeftel. Absorbing boundary conditions for nonlinear scalar partial differential equations. *Comput. Methods. Appl. Mech. and Engr.*, 195:3760–3775, 2006.
- [22] C. Zheng. Exact nonreflecting boundary conditions for one-dimensional cubic nonlinear Schrödinger equations. *J. Comput. Phys.*, 215:552–565, 2006.
- [23] S. Jiang and L. Greengard. Efficient representation of nonreflecting boundary conditions for the time-dependent Schrödinger equation in two dimensions. *Commun. Pure Appl. Math.*, 61:261–288, 2007.
- [24] A. Zisowsky and M. Ehrhardt. Discrete artificial boundary conditions for nonlinear Schrödinger equations. *Math. Comput. Model.*, 47:1264–1283, 2008.
- [25] Z. Chen, J. Zhang, and Z. Yu. Solution of the time-dependent Schrödinger equation with absorbing boundary conditions. *J. Semicond.*, 30:012001, 2009.

- [26] X. Antoine, C. Besse, and P. Klein. Absorbing boundary conditions for general Schrödinger equations. *SIAM J. Sci. Comput.*, 33:1008–1033, 2011.
- [27] F. I. Moxley, III, F. Zhu, and W. Dai. A generalized finite-difference method with absorbing boundary condition for solving a time-dependent linear Schrödinger equations. *AJCM*, 2:163–172, 2012.
- [28] B. Wang and D. Liang. The finite difference scheme for nonlinear Schrödinger equations on unbounded domain by artificial boundary conditions. *Appl. Numer. Math.*, 128:183–204, 2018.
- [29] J. Zhang, Z. Xu, and X. Wu. Unified approach to split absorbing boundary conditions for nonlinear Schrödinger equations. *Phys. Rev. E*, 78:026709, 2008.
- [30] J. Zhang, Z. Xu, and X. Wu. Unified approach to split absorbing boundary conditions for nonlinear Schrödinger equations: Two-dimensional case. *Phys. Rev. E*, 79:046711, 2009.
- [31] X. Antoine, C. Besse, and V. Mouysset. Artificial boundary conditions for one-dimensional cubic nonlinear Schrödinger equations. *Math. of Comput.*, 73:1779–1799, 2004.
- [32] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical evaluation of waves. *Math. Comp.*, 31:629–651, 1977.
- [33] G. Mur. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations. *IEEE Trans. Electromagn. Compat.*, 23:377–382, 1981.
- [34] L. Trefethen and L. Halpern. Well-posedness of one-way wave equations and absorbing boundary conditions. *Math. of Comput.*, 47:421–435, 1986.
- [35] X. Antoine and C. Besse. Unconditionally stable discretization schemes of non-reflecting boundary conditions for the one-dimensional Schrödinger equation. *J. Comput. Phys.*, 188:157–175, 2003.
- [36] X. Antoine, A. Arnold, C. Besse, M. Ehrhardt, and A. Schädle. A review of transparent and artificial boundary conditions techniques for linear and nonlinear Schrödinger equations. *Commun. Comput. Phys.*, 4:729–796, 2008.
- [37] Z. Chen, J. Zhang, and Z. Yu. A perfectly matched layer approach to the nonlinear Schrödinger wave equations. *J. Comput. Phys.*, 227:537–556, 2007.
- [38] Z. Xu, H. Han, and X. Wu. Adaptive absorbing boundary conditions for Schrödinger-type equations: application to nonlinear and multi-dimensional problems. *J. Comput. Phys.*, 225:1577–1589, 2007.

- [39] J. Cole and D. Zhu. Improved version of the second-order Mur absorbing boundary condition based on a nonstandard finite difference model. *ACES J.*, 24:375–381, 2009.
- [40] S. Zhou and X. Cheng. Numerical solution to coupled nonlinear Schrödinger equations on unbounded domains. *Math. Comp. Simulat.*, 80:2362–2373, 2010.
- [41] T. P. Stefanski, N. Chavannes, and N. Kuster. Multi-GPU accelerated finite-difference time-domain solver in open computing language. *PIERS*, 7:71–74, 2011.
- [42] X. Antoine, E. Lorin, and Q. Tang. A friendly review of absorbing boundary conditions and perfectly matched layers for classical and relativistic quantum waves equations. *Mol. Phys.*, 115:1861–1879, 2017.
- [43] S. Ji, Y. Yang, G. Pang, and X. Antoine. Accurate artificial boundary conditions for the semi-discretized linear Schrödinger and heat equations on rectangular domains. *Comput. Phys. Commun.*, 222:84–93, 2018.
- [44] J. P. Wilson. Generalized finite-difference time-domain method with absorbing boundary conditions for solving the nonlinear Schrödinger equation on a GPU. *Comput. Phys. Commun.*, 235:279–292, 2018.
- [45] V. Vaibhava. On the nonreflecting boundary operators for the general two dimensional Schrödinger equation. *J. Math. Phys.*, 60:011509, 2019.
- [46] J. J. Sakurai and J. Napolitano. *Modern Quantum Mechanics*. Pearson, Boston, 2nd edition, 2011.
- [47] P. G. Kevrekidis and D. J. Frantzeskakis. Solitons in coupled nonlinear Schrödinger models: A survey of recent developments. *Rev. in Phys.*, 1:140–153, 2016.
- [48] W. Bao, Q. Tang, and Z. Xu. Numerical methods and comparison for computing dark and bright solitons in the nonlinear Schrödinger equation. *J. Comput. Phys.*, 235:423–445, 2013.
- [49] N. H. Sweilam, S. M. Al-Mekhlafi, and A. O. Albalawi. A novel variable-order fractional nonlinear Klein-Gordon model: A numerical approach. *Numer. Meth. for Partial Differential Equations*, 35:1617–1629, 2019.
- [50] M. S. Ali, M. Shamsi, H. Khosravian-Arab, D. F. M. Torres, and F. Bozorgnia. A space–time pseudospectral discretization method for solving diffusion optimal control problems with two-sided fractional derivatives. *Journal of Vibration and Control*, 25:1080–1095, 2018.
- [51] N. H. Sweilam, T. A. Assiri, and M. M. Abou Hassan. Numerical solutions of nonlinear fractional Schrödinger equations using nonstandard discretizations. *Numer. Meth. for Partial Differential Equations*, 33:1399–1419, 2017.

- [52] N. Liu and W. Jiang. A numerical method for solving the time fractional Schrödinger equation. *Adv. Comput. Math.*, 44:1235–1248, 2018.
- [53] X. Antoine, Q. Tang, and J. Zhang. On the numerical solution and dynamical laws of nonlinear fractional Schrödinger/Gross-Pitaevskii equations. *Int. J. Comput. Math.*, 95:1423–1443, 2018.
- [54] M. Al-Raei and M. S. El-Daher. A numerical method for fractional Schrödinger equation of Lennard-Jones potential. *Phys. Lett. A*, 383:125831, 2019.
- [55] J. Fleck, Jr., J. Morris, and M. Feit. Time-dependent propagation of high energy laser beams through the atmosphere. *Appl. Phys.*, 10:129–160, 1976.
- [56] B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, Cambridge, 1998.
- [57] J. A. C. Weideman and B. M. Herbst. Split-step methods for the solution of the nonlinear Schrödinger equation. *SIAM J. Numer. Anal.*, 23:485–507, 1986.
- [58] M. Delfour, M. Fortin, and G. Payr. Finite-difference solutions of a non-linear Schrödinger equation. *J. Comput. Phys.*, 44:277–288, 1981.
- [59] A. G. Bratsos. A linearized finite-difference scheme for the numerical solution of the nonlinear cubic Schrödinger equation. *Korean J. Comput. Appl. Math.*, 8:459–467, 2001.
- [60] P. J. Bryant. Nonlinear wave groups in deep water. *Stud. Appl. Math.*, 61:1–30, 1979.
- [61] Q.-S. Chang and L.-B. Xu. A numerical method for a system of generalized nonlinear Schrödinger equations. *J. Comput. Math.*, 4:191–199, 1986.
- [62] W. Dai and R. Nassar. A finite difference scheme for the generalized nonlinear Schrödinger equation with variable coefficients. *J. Comput. Math.*, 18:123–132, 2000.
- [63] L. M. Degtyarev and V. Krylov. A method for the numerical solution of problems of the dynamics of wave fields with singularities. *USSR Comp. Math. Math. Phys.*, 17: 172–179, 1977.
- [64] D. F. Griffiths, A. R. Mitchell, and J. L. Morris. A numerical study of the nonlinear Schrödinger equation. *Comput. Methods. Appl. Mech. and Engr.*, 45:177–215, 1984.
- [65] F. Ivanauskas and M. Radziunas. On convergence and stability of the explicit difference method for solution of nonlinear Schrödinger equations. *SIAM J. Numer. Anal.*, 36:1466–1481, 1999.

- [66] P. L. Nash and L. Chen. Efficient finite difference solutions to the time-dependent Schrödinger equation. *J. Comput. Phys.*, 139:266–268, 1997.
- [67] T. R. Taha and M. J. Ablowitz. Analytical and numerical aspects of certain nonlinear evolution equations: numerical Korteweg-de Vries equation. *J. Comput. Phys.*, 55:231–253, 1984.
- [68] E. Twizell, A. Bratsos, and J. Newby. A finite-difference method for solving the cubic Schrödinger equation. *Math. Comp. Simulat.*, 43:67–75, 1997.
- [69] T. Utsumi, T. Aoki, J. Koga, and M. Yamagiwa. Solutions of the 1D coupled nonlinear Schrödinger equations by the CIP-BS method. *Commun. Comput. Phys.*, 1:261–275, 2006.
- [70] W. Bao and D. Jaksch. An explicit unconditionally stable numerical method for solving damped nonlinear Schrödinger equations with a focusing nonlinearity. *SIAM J. Numer. Anal.*, 41:1406–1426, 2003.
- [71] O. Karakashian and C. Makridakis. A space-time finite element method for the nonlinear Schrödinger equation: the continuous Galerkin method. *SIAM J. Numer. Anal.*, 36:1779–1807., 1999.
- [72] B. D. Shizgal and H. Chen. The quadrature discretization method (QDM) in the solution of the Schrödinger equation with nonclassical basis functions. *J. Chem. Phys.*, 104:4137–4150, 1996.
- [73] K. Leung, B. D. Shizgal, and H. Chen. The quadrature discretization method (QDM) in comparison with other numerical methods of solution of the Fokker–Planck equation for electron thermalization. *J. Math. Chem.*, 24:291–319, 1998.
- [74] H. Chen and B. D. Shizgal. The quadrature discretization method (QDM) in the solution of the Schrödinger equation. *J. Math. Chem.*, 24:321–343, 1998.
- [75] J. Lo and B. D. Shizgal. Spectral convergence of the quadrature discretization method in the solution of the Schrödinger and Fokker-Planck equations: comparison with Sinc methods. *J. Chem. Phys.*, 125:194108, 2006.
- [76] A. Taflove and S. C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Springer Series in Computational Physics. Artech House, Boston, 3rd edition, 2005.
- [77] F. I. Moxley, III, T. Byrnes, F. Fujiwara, and W. Dai. Generalized finite-difference time-domain quantum method for the N -body interacting Hamiltonian. *Comput. Phys. Commun.*, 183:2434–2440, 2012.
- [78] F. I. Moxley, III, D. T. Chuss, and W. Dai. A generalized finite-difference time-domain quantum method for the N -body interacting Hamiltonian. In A. B. Gumel,

editor, *Mathematics of Continuous and Discrete Dynamical Systems*, volume 618 of *Contemporary Mathematics*, chapter 9, pages 181–194. American Mathematical Society, Providence, Rhode Island, 2014.

- [79] F. I. Moxley, III, T. Byrnes, B. Ma, Y. Yan, and W. Dai. A G-FDTD scheme for solving multi-dimensional open dissipative Gross–Pitaevskii equations. *J. Comput. Phys.*, 282:303–316, 2015.
- [80] F. I. Moxley, III, J. P. Dowling, W. Dai, and T. Byrnes. Sagnac interferometer with coherent vortex superposition states in exciton-polariton condensates. *Phys. Rev. A*, 93:053603, 2016.
- [81] J. Shen, W. E. I. Sha, Z. Huang, M. Chen, and X. Wu. High-order symplectic fdtd scheme for solving a time-dependent Schrödinger equation. *SIAM J. Numer. Anal.*, 41:1406–1426, 2003.
- [82] K. W. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, Cambridge, 2nd edition, 2005.
- [83] S. Willard. *General Topology*. Addison-Wesley Publishing Co., Danvers, MA, 1st edition, 1970.
- [84] C. Kittel. *Introduction to Solid State Physics*. John Wiley and Sons, Inc., Danvers, MA, 8th edition, 2005.
- [85] Z. Z. Sun and G. H. Gao. *The Finite Difference Method for Fractional Order Differential Equations*. Science Press, Beijing, 1st edition, 2015. (Chinese).
- [86] Z. Z. Sun and W. Dai. A new accurate numerical method for solving heat conduction in a double-layered film with the Neumann boundary condition. *Numer. Meth. for Partial Differential Equations*, 30:1291–1314, 2014.
- [87] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, et al. Debunking the 100x GPU vs. GPU myth: an evaluation of throughput computing on CPU and GPU. *ACM SIGARCH Computer Architecture News*, 38:451–460, 2010.
- [88] E. Buber and B. Diri. Performance analysis and CPU vs GPU comparison for deep learning. In *2018 6th International Conference on Control Engineering and Information Technology*, pages 1–6. IEEE, 2018.
- [89] B. I. Schneider. The impact of heterogeneous computer architectures on computational physics. *Computing in Science Engineering*, 17:9–13, 2015.
- [90] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *2018 32nd Conference on Neural Information Processing Systems*, pages 1–13. IEEE, 2018.

- [91] T. P. Stefanski, S. Benkler, N. Chavannes, and N. Kuster. Parallel implementation of the finite-difference time-domain method in open computing language. In *2010 International Conference on Electromagnetics in Advanced Applications*, pages 1–4. IEEE, 2010.
- [92] J. Thompson and K. Schlachter. An Introduction to the OpenCL Programming Model. Technical report, NYU: Media Research Lab, New York, New York, 2012.
- [93] J.-H. Huang. Accelerated computing: The path forward. SC15 Conference Slides, 2015.
- [94] A. Klöckner. OpenCL: The open standard for parallel programming of heterogeneous systems, 2009. URL <https://documen.tician.de/pyopencl>.
- [95] A. Klöckner. Welcome to PyOpenCL’s Documentation!, 2009. URL <https://documen.tician.de/pyopencl>.
- [96] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Comput.*, 38:157–174, 2012.
- [97] T. P. Stefanski, N. Chavannes, and N. Kuster. Multi-GPU Accelerated Finite-difference Time-domain Solver in Open Computing Language. *PIERS Online*, 7: 71–74, 2010.
- [98] M. M. Meerschaert and C. Tadjeran. Finite difference approximations for two-sided space-fractional partial differential equations. *Appl. Numer. Math.*, 56:80–90, 2005.
- [99] R. Almeida, D. Tavares, and F. M. Torres. *The Variable-Order Fractional Calculus of Variations*. Springer, Gewerbestrasse, Switzerland, 2019.
- [100] R. Herrmann. *Fractional Calculus: An Introduction for Physicists*. World Scientific, GigaHedron, Germany, 2nd edition, 2014.
- [101] R. Hilfer. *Applications of Fractional Calculus in Physics*. World Scientific, Hoboken, NJ, 2000.
- [102] N. Laskin. Fractional quantum mechanics and lévy path integrals. *Phys. Lett. A*, 268: 298–305, 2000.
- [103] N. Laskin. Fractional Schrödinger equation. *Phys. Rev. E*, 66:056108, 2002.
- [104] N. Laskin. *Fractional Quantum Mechanics*. World Scientific, Singapore, 1st edition, 2018.

- [105] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [106] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, et al. PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation. *Parallel Comput.*, 38:157–174, 2012. ISSN 0167-8191.