UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

An Adaptive Local Discrete Convolution Method for the Numerical Solution of Maxwell's Equations

Permalink

<https://escholarship.org/uc/item/35c9f1cn>

Author

Lo, Boris

Publication Date

2019

Peer reviewed|Thesis/dissertation

An Adaptive Local Discrete Convolution Method for the Numerical Solution of Maxwell's
Equations

by

Boris Tsz Kiu Lo

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Graduate Group in Applied Science and Technology

and the Designated Emphasis

in

Computational and Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Phillip Colella, Chair
Professor Panayiotis Papadopoulos
Professor Per-Olof Persson

Spring 2019

An Adaptive Local Discrete Convolution Method for the Numerical Solution of Maxwell's
Equations

Copyright 2019
by
Boris Tsz Kiu Lo

Abstract

An Adaptive Local Discrete Convolution Method for the Numerical Solution of Maxwell's Equations

by

Boris Tsz Kiu Lo

Doctor of Philosophy in Graduate Group in Applied Science and Technology
and the Designated Emphasis in Computational and Data Science and Engineering

University of California, Berkeley

Professor Phillip Colella, Chair

A high order multilevel numerical solver for Maxwell's equations using local discrete convolutions is presented. Maxwell's equations are embedded in a system of wave equations. The solution to this system can be derived with the method of spherical means. When the source terms are prescribed functions then there is an explicit update formula using the propagator form of the solution and Duhamel's formula. And when the source terms are dependent on the fields, the propagator can be embedded in the system of differential equations by a change of variables by Lawson's method. The spherical means operations are discretized using the framework from Tornberg *et al.* [84] and on uniform spacing rectangular grids the procedure becomes constant coefficient stencil operations. While the source term integration is handled with a quadrature scheme or a time integrator. This method can be parallelized with standard domain decomposition. Since this method places the electric and magnetic fields on the same grids, as opposed to staggered grids used in standard grid based solvers for Maxwell's equations, and that it has no time stepping restriction, because it is a propagator based method, it can be extended to use local mesh refinement with only simple interpolating and sampling operators.

For my parents.

Contents

Contents	ii
List of Figures	v
List of Tables	vi
List of Algorithms	vii
1 Introduction	1
1.1 Motivation	1
1.2 Numerical Methods for Maxwell's equations	1
1.2.1 Spatial Discretizations	2
1.2.2 Propagator Methods	2
1.3 Thesis Overview	4
2 Analytic Solution of Maxwell's Equations	5
2.1 The Auxiliary System: Maxwell's Equations as a System of Wave Equations	6
2.2 Solution of the 3D Wave Equation	8
2.2.1 Duhamel's Principle for the Inhomogenous 3D Wave Equation	11
2.2.2 Field Dependent Source Terms	12
2.3 Solution of the Auxiliary System	14
2.3.1 Connection with Retarded Field Solutions	14
2.3.2 Connection with the Maxwell Propagator	17
3 Local Discrete Convolution Method (LDCM)	20
3.1 Centered-Difference Approximations	20
3.2 Discretizing Spherical Means	22
3.3 High Order <i>B</i> -Splines	25
3.3.1 Smooth High Order <i>B</i> -Splines	27
3.4 Timestepping: Lawson's method and Closed Newton-Cotes Quadrature	29
3.5 Spatial Error Analysis	31
3.6 Efficient Application of the Propagator	32
3.6.1 Hockney's Algorithm	33
3.6.2 Quad Precision Arithmetic for High Order Kernels	35

3.6.3	Domain Decomposition	35
3.6.4	Cost Comparisons	35
3.7	Constraint Enforcement	36
3.8	Single Level Algorithm	39
3.8.1	Extension to 2D	40
3.9	Relationship with Other Propagator methods	40
3.9.1	Spherical Means in Shin (2018)	40
3.9.2	Spherical Means for Finite Element Meshes	42
3.9.3	Source Integration in PSATD	42
4	AMR for LDCM	43
4.1	Basic AMR Operators	43
4.1.1	Sampling	44
4.1.2	Interpolation	44
4.2	Regridding	44
4.3	Open Boundary Conditions	45
4.4	AMR LDCM Algorithm	46
5	Computational Framework	49
5.1	The Chombo Package	49
5.2	C++ Classes for AMR LDCM	50
5.3	B-Spline Generator	51
6	Numerical Tests	53
6.1	Plane Wave Propagation	53
6.1.1	Plane Wave Propagating in $\hat{\mathbf{x}}$	54
6.1.2	Non-Axis Aligned Plane Wave	54
6.2	Divergence-Free Current Source	55
6.3	Smooth Charge Distribution	56
6.3.1	Translating Charge Source	57
6.3.2	Electrostatics Test	58
6.3.3	Regridding Test	59
6.4	Performance Results	59
7	Conclusions	65
7.1	Future Work	66
A	Method of Spherical Means	67
A.1	d'Alembert's Formula for the Homogenous 1D Wave Equation	67
A.2	Euler-Poisson-Darboux Equation for the Spherical Mean Variable	68
A.3	Method of Spherical Means: Solution of the Homogenous 3D Wave Equation	70
B	Spherical Means Kernels Generator Code	71

C High Order B -Splines	80
Bibliography	83

List of Figures

3.1	Phase error for discrete propagator	32
3.2	Amplitude error for discrete propagator	33
3.3	Decaying amplitude for a plane wave	34
3.4	Ratios of Centered-Difference Approximations for Divergence Filter	39
4.1	Illustration of nested rectangular grid in 1D.	44
4.2	Illustration of sampling and interpolating regions for two levels.	45
4.3	Illustration of spherical means at domain boundary.	47
6.1	Convergence results for the 5 th order solver for the axis-aligned plane wave problem	54
6.2	Convergence results for the 5 th order solver for the non-axis aligned plane wave problem	55
6.3	Convergence results for divergence-free current problem.	56
6.4	Convergence results for translating charge source problem.	58
6.5	Convergence results for translating charge source returning to rest problem. . . .	59
6.6	$\nabla \cdot \mathbf{E}$ plot for translating charge source returning to rest problem showing no reflected waves at refinement boundaries.	60
6.7	E_x for the translating charge problem with regridding.	61
6.8	Convergence results for the translating charge problem with regridding.	62

List of Tables

3.1	Values for filter parameter, η	40
6.1	Problem parameters and convergence rates from Richardson error estimate in ℓ_2 for E_y, E_z, B_y, B_z using the 15 th order solver for the axis aligned plane wave problem.	55
6.2	Performance result for the 5 th order solver using pure MPI with eight patches per core.	63
6.3	Performance result for the 5 th order solver using pure MPI with one patch per core.	63
6.4	Performance result for the 5 th order solver using hybrid MPI-OpenMP with one MPI task per socket and eight patches per core.	63
6.5	Performance result for the 5 th order solver using hybrid MPI-OpenMP with two MPI tasks per socket and eight patches per core.	63
6.6	Performance as a function of patch size for the 5 th order solver with flat MPI with $N = 129$	64

List of Algorithms

3.1	Single Level LDCM for Maxwell's Equations	41
4.1	Regridding Algorithm	46
4.2	AMR LDCM for Maxwell's Equations	48

Acknowledgments

First and foremost, I would like to thank my Ph.D. advisor, Phillip Colella, for his unending patience and mentorship which made this research possible and made these six years a positive experience. Thank you for your insights and guidance, most especially during times when it seemed like there was no solution to the immediate obstacle. You have taught me disciplined thinking and methodical analysis when confronted with the unknown. I would also like to thank the members of ANAG. Special thanks to Dan Graves, Terry Ligocki, and Brian Van Straalen for their detailed explanations and answers to my many questions regarding Chombo. Thanks to Hans Johansen for his advice and comments about structuring this dissertation.

Finally, this research was supported by the Office of Advanced Scientific Computing Research of the US Department of Energy and used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility; both under Contract No. DE-AC02-05CH11231.

Chapter 1

Introduction

1.1 Motivation

The current computer architecture trend has led to efforts for new numerical methods for solving partial differential equations; methods where the processor-to-processor communication is minimized in exchange for more, and even some redundant, computation [96]. This is because data transfer is orders of magnitude more costly than floating point operations. To this end, this dissertation seeks a high order numerical method based on discrete convolutions with compact support kernels. Higher order numerical methods have fewer degrees of freedom and therefore for the same level of error they require less data transfer than their lower order counterparts. Discrete convolutions using well-characterized kernels and fast Fourier transform are computationally intensive. Compact support convolution kernels limits the required amount of data transfer to be with only neighbors in the problem domain. Coupled with a sensible load balancing algorithm, this leads to communication only between processors that are physically near each other instead of the far more expensive global all-to-all communication. Local discrete convolutions have been studied for use with manycore processors in the case of Poisson's equation with the method of local corrections (MLC) [6, 69, 87].

1.2 Numerical Methods for Maxwell's equations

This section gives a brief overview of numerical methods for Maxwell's equations as well as exponential integrator methods in general and their use in solving the wave equation in particular. It then discusses recent development in propagator methods for Maxwell's equations and the relationship between the work detailed in this dissertation and these other efforts. And finally it presents an overview of artificial boundary conditions for unbounded domain problems.

1.2.1 Spatial Discretizations

By far the most popular numerical method for Maxwell's equations is the finite-difference time-domain (FDTD) scheme, such as Yee's Method [99] based on staggered Cartesian grids and its many higher order versions [36, 45, 59, 66, 75, 86, 102, 103]. The main motivation for these higher order FDTD schemes is to reduce numerical dispersion from the second order Yee scheme [48]. There is also a finite difference method in which the fields reside on the same grid [80]. There had also been early efforts in developing finite volume methods for Maxwell's equations [81, 101]. Since then, there is ongoing research to tackle problems like discontinuous media [52, 76]. Recently, second order [10, 34, 91] and fourth order [21, 90] finite volume methods with adaptive mesh refinement (AMR) capabilities have been explored.

For unstructured grids, in addition to finite volume methods, the finite element method has been used extensively for electromagnetics [54, 71]. A common finite element discretization used for Maxwell's equations is using Nédélec elements as the basis function [72, 73] or higher order variants [41]. There are also finite elements based on Whitney forms [17, 97] or discontinuous Galerkin [23]. However, unstructured grids are computationally expensive. This led to the development of hybrid methods which takes advantage of the geometric flexibility of unstructured grids near boundaries and computational efficiency of structured grids in homogeneous regions [7, 32, 77, 78, 98, 100].

In addition, for rectangular grids, there is the psuedospectral time-domain (PSTD) method which computes the derivatives in Fourier space [63]. The multiresolution time-domain (MRTD) method based on orthonormal wavelet expansions in space [58]. The local spectral time-domain (LSTD) method which interpolates the fields with regularized delta kernels in space and then applying the differential operators to these regularized kernels [9].

There are also Green's function methods for Maxwell's equations in the presence of interfaces and objects [19], but these methods avoid the spherical means integral directly because of its singular nature and instead solve a matrix equation by applying the method of moments on the integral equation.

Finally, there are mesh-free methods, based on their success in mechanics and hydrodynamics problems: smoothed particle electromagnetics (SPEM) [4], element-free Galerkin (EFG) [22], and meshless local Petrov-Galerkin (MLPG) [29].

1.2.2 Propagator Methods

For the wave equation and Maxwell's equations, there is an analytic form for the kernel. In literature, *propagator* is sometimes used to refer to the kernel (solution to the homogeneous problem) and sometimes used to refer to the Green's function (solution to the inhomogeneous problem with delta function as forcing term). In this dissertation, propagator refers to the kernel. Propagator methods discretizes the propagator directly, either in real or Fourier space, and results in an explicit update for the fields rather than solving a linear system. This type of approach has been explored in 2D for the wave equation in [5] and further examined in [46, 62].

There are several numerical methods in literature using the propagator of Maxwell's equations as a starting point. Remaining in real space and discretizing the solution to an embedded wave equation system results in the LDCM. There are also methods that use the solution from the direct Maxwell propagator. This propagator is nonlocal and was used as the predecessor to LDCM, the same spherical means convolution kernels are present but this propagator also includes nonlocal Helmholtz decomposition operators [65]. Another approach while remaining in real space is to revert back to the spherical means integrals and discretize them directly with spherical quadratures [82]. An alternative to the spherical means integral is to apply the propagator and derivatives in Fourier space. This results in the pseudospectral analytic time domain (PSATD) method [44, 89] and the path integral time domain method (PITD) [74].

Propagator methods are part of a larger class of numerical methods known as exponential integrators or exponential time difference (ETD) [50]. Exponential integrators are based on discretizing the solution operator, or kernel, for the homogeneous evolution equation or some linear part of it. One of the main advantages of exponential integrators is that they are free from time step restrictions. So if they are embedded in a larger problem then any time step constraint will only come from other parts of the problem, which could be more complicated or physically interesting. For many problems, there is no analytic closed form expression for the solution operator, but it can be written abstractly as the exponential of the linear differential operator. The linear operator is discretized and the action of the operator matrix exponential on a vector is evaluated using Krylov subspace methods [27, 55]. While implicit methods would also eliminate the time stepping constraint, it has been shown in [49] that Krylov subspace approximations for the action of a matrix exponential converges faster than Krylov subspace approximations for solving a linear system from the same linear operator.

The local discrete convolution method (LDCM) is a numerical method for the 3D Maxwell's equations in free space on structured rectangular grids. Maxwell's equations are embedded in a system of classical wave equations. The solution to the homogeneous problem is derived using the method of spherical means resulting in the propagator. The propagator is used with Duhamel's formula, if the source terms are prescribed, or Lawson's method [60], if the source terms are dependent on the fields. The spherical means integrals in the propagator can be written as convolutions with spherical delta distributions. These spherical delta distributions are regularized using functions with compact support so that they can be represented on a rectangular grid using the framework from [84]. Since the regularized delta distributions have compact support, the resultant discrete convolutions are local.

The starting point for LDCM was from examining PSATD and to develop a method that eliminates nonlocality while retaining high order accuracy and no CFL constraint. In real space, the propagator to the embedded system is local. A major part of this work is to ensure that the spatial discretization to the propagator remains local.

1.3 Thesis Overview

The rest of this dissertation is organized as follows. Chapter 2 presents the analytic solution to Maxwell's equations. It starts with embedding Maxwell's equations into classical wave equations with initial value constraints. One key feature for this set of equations is that the error from violating these constraints simply propagate away with the same wave speed as the fields. Then an overview for the solution of the wave equation is presented and its results are applied to the auxiliary embedded system. It finally shows the equivalence of the solution derived in this manner with the solutions derived in classical literature.

Chapter 3 presents the discretization procedure from the analytic propagator to its representation on a rectangular grid. The 3D spherical delta distributions that appear in the propagator are regularized following the framework in Tornberg *et al.* [84] and special emphasis is placed on describing the construction of the necessary 1D regularized kernel using high order cardinal B -splines. In particular, the discretized propagator also has compact support and so it can be applied with efficient computational techniques in practice: domain decomposition and Hockney's Algorithm [51]. It also describes a local filtering method to help diffuse the error induced by not satisfying the divergence constraints. The complete algorithm is outlined at the end.

Chapter 4 presents LDCM in the AMR setting. It gives a description of the fine-to-coarse sampling and coarse-to-fine interpolation operators. These operators are also used for regridding. It also discusses using AMR to simulate open boundary conditions. Chapter 5 presents an implementation of LDCM. Chapter 6 presents the results for some numerical tests. The problems chosen are to demonstrate the method's flexibility under different types of source terms and behavior with AMR. The first problem is a plane wave propagating with periodic boundary conditions because this is the eigenfunction for Maxwell's equations. Then, a divergence free current source and a moving charge source problem; both of which has analytic solution for the static cases. Finally, it presents some performance results showing parallel weak scaling for the method.

Chapter 7 summarizes the developments within this dissertation and ends with a discussion on future works.

Chapter 2

Analytic Solution of Maxwell's Equations

Maxwell's equations describe the evolution of electric and magnetic fields in the presence of charge and current densities. They can be expressed in cgs units with initial conditions

$$\nabla \cdot \mathbf{E} = 4\pi\rho, \quad (2.1)$$

$$\nabla \cdot \mathbf{B} = 0, \quad (2.2)$$

$$\frac{\partial \mathbf{E}}{\partial t} = c\nabla \times \mathbf{B} - 4\pi\mathbf{J}, \quad (2.3)$$

$$\frac{\partial \mathbf{B}}{\partial t} = -c\nabla \times \mathbf{E}, \quad (2.4)$$

$$\mathbf{E}(\mathbf{x}, 0) = \mathbf{E}_0, \quad (2.5)$$

$$\mathbf{B}(\mathbf{x}, 0) = \mathbf{B}_0. \quad (2.6)$$

where $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic field, c is the speed of light, $\mathbf{J}$ is a current density, and ρ is a charge density. In uniform medium, c is a constant. In general, $\rho, \mathbf{J}$ can be dependent on $\mathbf{E}, \mathbf{B}$. It is assumed that the initial conditions satisfy the two divergence constraints.

This chapter describes the solution to Maxwell's equations with initial values and sources. The equations are embedded in a system of classical wave equations. In 3D, the solution to the homogeneous wave equation is given by the method of spherical means. With the solution operator for the homogeneous problem, the solution to the inhomogeneous case can be obtained by Duhamel's principle. The details for the wave equation given in this dissertation follow directly from literature on the topic [35, 94]. These solutions are rewritten in convolution notation and finally their link with some classically given solutions to Maxwell's equations are shown.

2.1 The Auxiliary System: Maxwell's Equations as a System of Wave Equations

Instead of solving Maxwell's equations directly, an auxiliary system where its solutions are also solutions to Maxwell's equations is solved instead. Introducing the auxiliary variables, $\Psi = \nabla \times \mathbf{E}$ and $\Phi = \nabla \times \mathbf{B}$, the evolution equations associated are given by taking the curl of (2.3) and (2.4) respectively

$$\begin{aligned}\nabla \times \frac{\partial \mathbf{E}}{\partial t} &= c \nabla \times \nabla \times \mathbf{B} - 4\pi \nabla \times \mathbf{J}, \\ \frac{\partial \Psi}{\partial t} &= c \nabla \times \nabla \times \mathbf{B} - 4\pi \nabla \times \mathbf{J},\end{aligned}$$

using the vector identity $\nabla \times \nabla \times \mathbf{B} = \nabla(\nabla \cdot \mathbf{B}) - \Delta \mathbf{B}$,

$$\frac{\partial \Psi}{\partial t} = c \nabla(\nabla \cdot \mathbf{B}) - c \Delta \mathbf{B} - 4\pi \nabla \times \mathbf{J},$$

substituting in (2.2)

$$\frac{\partial \Psi}{\partial t} = -c \Delta \mathbf{B} - 4\pi \nabla \times \mathbf{J}.$$

The same steps are used for Φ . The evolution equations for the four field quantities are

$$\frac{\partial \mathbf{E}}{\partial t} = c \Phi - 4\pi \mathbf{J}, \tag{2.7}$$

$$\frac{\partial \Phi}{\partial t} = c \Delta \mathbf{E} - 4\pi c \nabla \rho, \tag{2.8}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -c \Psi, \tag{2.9}$$

$$\frac{\partial \Psi}{\partial t} = -c \Delta \mathbf{B} - 4\pi \nabla \times \mathbf{J}. \tag{2.10}$$

with four constraints

$$\Phi = \nabla \times \mathbf{B}, \tag{2.11}$$

$$\Psi = \nabla \times \mathbf{E}, \tag{2.12}$$

$$\nabla \cdot \mathbf{E} = 4\pi \rho, \tag{2.13}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{2.14}$$

Suppose the constraints are not satisfied, consider the four error quantities associated with not satisfying the constraints

$$\mathbf{K}_B = \Phi - \nabla \times \mathbf{B}, \tag{2.15}$$

$$\mathbf{K}_E = \Psi - \nabla \times \mathbf{E}, \tag{2.16}$$

$$D_B = \nabla \cdot \mathbf{B}, \tag{2.17}$$

$$D_E = \nabla \cdot \mathbf{E} - 4\pi \rho. \tag{2.18}$$

The time evolution for these quantities can be readily derived using (2.7)-(2.10)

$$\begin{aligned}\frac{\partial \mathbf{K}_B}{\partial t} &= \frac{\partial \Phi}{\partial t} - \nabla \times \frac{\partial \mathbf{B}}{\partial t} \\ &= c(-\nabla \times \nabla \times \mathbf{E} + \nabla(\nabla \cdot \mathbf{E})) - 4\pi \nabla \rho + c \nabla \times \Psi \\ &= c \nabla \times \mathbf{K}_E + c \nabla D_E,\end{aligned}\tag{2.19}$$

Likewise

$$\frac{\partial \mathbf{K}_E}{\partial t} = -c \nabla \times \mathbf{K}_B - c \nabla D_B.\tag{2.20}$$

For the scalar constraints

$$\frac{\partial D_B}{\partial t} = \nabla \cdot \frac{\partial \mathbf{B}}{\partial t} = -c \nabla \cdot \Psi = -c \nabla \cdot \mathbf{K}_E,\tag{2.21}$$

$$\begin{aligned}\frac{\partial D_E}{\partial t} &= \nabla \cdot \frac{\partial \mathbf{E}}{\partial t} - 4\pi \rho \\ &= c \nabla \cdot \Phi - 4\pi \nabla \cdot \mathbf{J} - 4\pi \rho \\ &= c \nabla \cdot \mathbf{K}_B.\end{aligned}\tag{2.22}$$

In summary the four evolution equations for the error quantities are

$$\frac{\partial \mathbf{K}_B}{\partial t} = c \nabla \times \mathbf{K}_E + c \nabla D_E,\tag{2.23}$$

$$\frac{\partial \mathbf{K}_E}{\partial t} = -c \nabla \times \mathbf{K}_B - c \nabla D_B,\tag{2.24}$$

$$\frac{\partial D_B}{\partial t} = -c \nabla \cdot \mathbf{K}_E,\tag{2.25}$$

$$\frac{\partial D_E}{\partial t} = c \nabla \cdot \mathbf{K}_B.\tag{2.26}$$

If the constraints are satisfied initially, then the right hand side of the equations are 0 and thus the error will continue to be 0 for all time. In addition, the symbol of the linear operator associated with this system has the eigenvalues $\pm ic|\mathbf{k}|$ each with a multiplicity of 4. Therefore, any small perturbation will propagate away with speed c . The auxiliary system readily cleans out constraint violation by propagating the error away with the same speed as the electromagnetic fields. This is important because in the context of the modified equation, the truncation error can be viewed as additional source term in the original equations which may cause the constraints to be violated. For this system (2.7)-(2.10), this error propagates away and does not affect the well-posedness of the system. This is in contrast to magnetohydrodynamics, where the problem becomes ill-posed [28, 40].

Satisfying the constraints ensure that the solutions to this auxiliary system are also solutions to the original Maxwell's equations. This is because the constraints are used to go from Maxwell's equations to the auxiliary equations and so in order to recover Maxwell's equations these constraints must be valid. (2.7)-(2.8) form a system of wave equation while (2.9)-(2.10) form another. The two sets are coupled through the constraints and source terms.

2.2 Solution of the 3D Wave Equation

Since the auxiliary system is a set of first order wave equations, its solutions can be derived using standard methods for solving the wave equation. This section describes the solution to the scalar wave equation. Consider the inhomogeneous 3D scalar wave equation with initial conditions

$$\begin{aligned}\frac{\partial^2 u}{\partial t^2} &= c^2 \Delta u + f + \frac{\partial g}{\partial t}, \\ u(t=0) &= u_0, \\ \frac{\partial u}{\partial t}(t=0) &= \phi_0,\end{aligned}\tag{2.27}$$

where f, g are known functions of $\mathbf{x}, t$ independent of u . Let $\phi = \frac{\partial u}{\partial t}$, then this can be rewritten as the following first order system

$$\begin{aligned}\frac{\partial u}{\partial t} &= c\phi + g, \\ \frac{\partial \phi}{\partial t} &= c\Delta u + \frac{1}{c}f, \\ u(t=0) &= u_0, \\ \phi(t=0) &= \phi_0.\end{aligned}\tag{2.28}$$

If $f = g = 0$, the solution to the homogeneous 3D wave equation is given by the method of spherical means [35, 94], which is described in detail in Appendix A

$$u(\mathbf{x}, t) = \frac{1}{c} \frac{\partial \bar{u}_0}{\partial t} + \bar{\phi}_0,\tag{2.29}$$

$$\phi(\mathbf{x}, t) = \Delta \bar{u}_0 + \frac{1}{c} \frac{\partial \bar{\phi}_0}{\partial t},\tag{2.30}$$

where $\bar{u}_0, \bar{\phi}_0$ denote the modified spherical mean (A.15). (2.29)-(2.30) can be rewritten with notation that shows radius of the sphere

$$u(\mathbf{x}, t) = \mathcal{H}^{ct}[u_0] + \mathcal{G}^{ct}[\phi_0],\tag{2.31}$$

$$\phi(\mathbf{x}, t) = \mathcal{G}^{ct}[\Delta u_0] + \mathcal{H}^{ct}[\phi_0],\tag{2.32}$$

where

$$(\mathcal{G}^{ct}[q])(\mathbf{x}) \equiv \frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) dS(\mathbf{z}),\tag{2.33}$$

$$(\mathcal{H}^{ct}[q])(\mathbf{x}) \equiv \frac{1}{c} \frac{\partial}{\partial t} \left(\frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) dS(\mathbf{z}) \right).\tag{2.34}$$

Since $\mathcal{G}^{ct}[q]$ is non-zero only the sphere with radius ct , it is an operator with compact support. $\mathcal{H}^{ct}[q]$ is also an operator with compact support. To show this [3]

$$(\mathcal{H}^{ct}[q])(\mathbf{x}) = \frac{1}{c} \frac{\partial}{\partial t} \left(\frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) dS(\mathbf{z}) \right)$$

Let $\mathbf{z} = \mathbf{x} + ct\mathbf{y}$, then $dS(\mathbf{z}) = c^2 t^2 dS(\mathbf{y})$ and in $\mathbf{y}$ -space the sphere becomes a unit sphere centered at the origin

$$\begin{aligned} &= \frac{1}{c} \frac{\partial}{\partial t} \left(\frac{1}{4\pi ct} \int_{\partial B_{1(0)}} q(\mathbf{x} + ct\mathbf{y}) c^2 t^2 dS(\mathbf{y}) \right) \\ &= \frac{\partial}{\partial t} \left(\frac{t}{4\pi} \int_{\partial B_{1(0)}} q(\mathbf{x} + ct\mathbf{y}) dS(\mathbf{y}) \right) \end{aligned}$$

Applying the chain rule

$$= \frac{1}{4\pi} \int_{\partial B_{1(0)}} q(\mathbf{x} + ct\mathbf{y}) dS(\mathbf{y}) + \frac{t}{4\pi} \int_{\partial B_{1(0)}} \nabla_{\mathbf{y}} q(\mathbf{x} + ct\mathbf{y}) \cdot c\mathbf{y} dS(\mathbf{y})$$

Converting back to $\mathbf{z}$

$$\begin{aligned} &= \frac{1}{4\pi} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) \frac{dS(\mathbf{z})}{c^2 t^2} + \frac{ct}{4\pi} \int_{\partial B_{ct}(\mathbf{x})} ct \nabla_{\mathbf{z}} q(\mathbf{z}) \cdot \left(\frac{\mathbf{z} - \mathbf{x}}{ct} \right) \frac{dS(\mathbf{z})}{c^2 t^2} \\ &= \frac{1}{4\pi c^2 t^2} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) dS(\mathbf{z}) - \frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} \nabla_{\mathbf{z}} q(\mathbf{z}) \cdot (\mathbf{x} - \mathbf{z}) dS(\mathbf{z}) \end{aligned}$$

Therefore

$$\mathcal{H}^{ct}[q] = \frac{1}{ct} \mathcal{G}^{ct}[q] - \sum_{i=1}^3 \mathcal{G}_i^{ct} \left[\frac{\partial q}{\partial z_i} \right], \quad (2.35)$$

where $\mathcal{G}_i^{ct}[q]$ is a weighted spherical mean

$$(\mathcal{G}_i^{ct}[q])(\mathbf{x}) = \frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} (z_i - x_i) q(\mathbf{z}) dS(\mathbf{z}). \quad (2.36)$$

By (2.35), $\mathcal{H}^{ct}[q]$ does not only have compact support but it has the same support as $\mathcal{G}^{ct}[q]$.

2.2.0.1 Properties of $\mathcal{G}^{ct}$ and $\mathcal{H}^{ct}$

While the spatial form of the convolution kernels are convenient for their physical interpretation, their Fourier transforms reveal some useful properties with regards to their dependence on t [11]. Consider the Fourier transform in space of the spherical means

$$\mathcal{F}[\mathcal{G}^{ct}[q]] = \int_{\mathbb{R}^3} e^{-i\mathbf{k} \cdot \mathbf{x}} \left(\frac{1}{4\pi ct} \int_{\partial B_{ct}(\mathbf{x})} q(\mathbf{z}) dS(\mathbf{z}) \right) d\mathbf{x}$$

Using the same coordinate transform, $\mathbf{z} = \mathbf{x} + ct\mathbf{y}$, as above

$$\begin{aligned} &= \int_{\mathbb{R}^3} e^{-i\mathbf{k}\cdot\mathbf{x}} \left(\frac{1}{4\pi ct} \int_{\partial B_1(0)} q(\mathbf{x} + ct\mathbf{y}) c^2 t^2 dS(\mathbf{y}) \right) d\mathbf{x} \\ &= \frac{ct}{4\pi} \int_{\partial B_1(0)} \left(\int_{\mathbb{R}^3} e^{-i\mathbf{k}\cdot\mathbf{x}} q(\mathbf{x} + ct\mathbf{y}) d\mathbf{x} \right) dS(\mathbf{y}) \end{aligned}$$

Multiplying the integrand by $1 = \exp(ict\mathbf{k} \cdot \mathbf{y}) \exp(-ict\mathbf{k} \cdot \mathbf{y})$

$$= \frac{ct}{4\pi} \int_{\partial B_1(0)} e^{-ict\mathbf{k}\cdot\mathbf{y}} \left(\int_{\mathbb{R}^3} e^{-i\mathbf{k}\cdot(\mathbf{x}+ct\mathbf{y})} q(\mathbf{x} + ct\mathbf{y}) d\mathbf{x} \right) dS(\mathbf{y})$$

Converting back to $\mathbf{z}$ and noting that $\frac{d\mathbf{z}}{d\mathbf{x}} = 1$

$$= \frac{ct}{4\pi} \int_{\partial B_1(0)} e^{-ict\mathbf{k}\cdot\mathbf{y}} \left(\int_{\mathbb{R}^3} e^{-i\mathbf{k}\cdot\mathbf{z}} q(\mathbf{z}) d\mathbf{z} \right) dS(\mathbf{y})$$

The inner paranthesis is $\tilde{q} \equiv \mathcal{F}[q]$

$$= \tilde{q} \frac{ct}{4\pi} \int_{\partial B_1(0)} e^{-ict\mathbf{k}\cdot\mathbf{y}} dS(\mathbf{y})$$

Aligning the coordinates so that $\mathbf{k}$ points in y_2 and noting $|\mathbf{y}| = 1$ for unit sphere

$$\begin{aligned} &= \tilde{q} \frac{ct}{4\pi} \int_0^{2\pi} d\phi \int_0^\pi e^{-ict|\mathbf{k}|\cos\theta} \sin\theta d\theta \\ &= \tilde{q} \frac{ct}{2} \int_0^\pi e^{-i|\mathbf{k}|ct\cos\theta} \sin\theta d\theta \\ &= \tilde{q} \frac{ct}{2} \int_0^\pi [-\cos(|\mathbf{k}|ct\cos\theta) + i\sin(|\mathbf{k}|ct\cos\theta)] \sin\theta d\theta \\ &= \tilde{q} \frac{ct}{2} \left(\frac{\sin(|\mathbf{k}|ct\cos\theta)}{c|\mathbf{k}|t} \Big|_0^\pi - i \frac{\cos(|\mathbf{k}|ct\cos\theta)}{c|\mathbf{k}|t} \Big|_0^\pi \right) \\ &= \frac{\sin(c|\mathbf{k}|t)}{|\mathbf{k}|} \tilde{q}. \end{aligned} \tag{2.37}$$

Taking the time derivative and dividing by c gives

$$\mathcal{F}[\mathcal{H}^{ct}[q]] = \cos(c|\mathbf{k}|t) \tilde{q}. \tag{2.38}$$

Since these operators are multiplications in Fourier space, formally they can be written as convolutions in physical space. This is the starting point for the predecessor of LDCM [65].

From basic trigonometry

$$\mathcal{G}^{-ct}[f] = -\mathcal{G}^{ct}[f], \quad (2.39)$$

$$\mathcal{H}^{-ct}[f] = \mathcal{H}^{ct}[f], \quad (2.40)$$

$$\mathcal{G}^{ct}[f] \Big|_{t=0} = 0, \quad (2.41)$$

$$\mathcal{H}^{ct}[f] \Big|_{t=0} = f. \quad (2.42)$$

And also, since the Fourier transform of the Laplacian is $-|\mathbf{k}|^2$

$$\frac{\partial \mathcal{H}^{ct}[f]}{\partial t} = -c\mathcal{G}^{ct}[\Delta f]. \quad (2.43)$$

Furthermore, the spherical means operators commute with spatial differential operators.

2.2.1 Duhamel's Principle for the Inhomogenous 3D Wave Equation

For the inhomogenous case, the solution is given by Duhamel's principle, which states that the solution to a linear inhomogenous evolution equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \mathcal{L}[u] &= f, \\ u(\mathbf{x}, 0) &= u_0, \end{aligned} \quad (2.44)$$

where $\mathcal{L}$ is a linear differential operator that has no time derivatives, is given by

$$u(\mathbf{x}, t) = \mathcal{P}^t[u_0] + \int_0^t \mathcal{P}^{t-s}[f]ds, \quad (2.45)$$

where $\mathcal{P}^t[\cdot] = e^{t\mathcal{L}}[\cdot]$ is the solution operator, or *propagator*, for the homogenous problem. The propagator for the homogenous wave equation is

$$\mathcal{P}^{ct} \left[\begin{pmatrix} u \\ \phi \end{pmatrix} \right] \equiv \begin{pmatrix} \mathcal{H}^{ct}[u] + \mathcal{G}^{ct}[\phi] \\ \mathcal{G}^{ct}[\Delta u] + \mathcal{H}^{ct}[\phi] \end{pmatrix}. \quad (2.46)$$

The propagator satisfies the following properties

$$\mathcal{P}^t[\mathcal{P}^s[\mathbf{u}]] = \mathcal{P}^{t+s}[\mathbf{u}], \quad (\text{semigroup}) \quad (2.47)$$

$$\alpha\mathcal{P}^t[\mathbf{u}] + \beta\mathcal{P}^t[\mathbf{v}] = \mathcal{P}^t[\alpha\mathbf{u} + \beta\mathbf{v}], \quad (\text{linearity}) \quad (2.48)$$

$$\mathcal{P}^0[\mathbf{u}] = \mathbf{u}. \quad (2.49)$$

Therefore, the solution to the 3D inhomogenous wave equation is

$$\begin{pmatrix} u(t) \\ \phi(t) \end{pmatrix} = \mathcal{P}^{ct} \left[\begin{pmatrix} u_0 \\ \phi_0 \end{pmatrix} \right] + \int_0^t \mathcal{P}^{c(t-s)} \left[\begin{pmatrix} g \\ \frac{1}{c}f \end{pmatrix} \right] ds. \quad (2.50)$$

Note that for $g = 0$, this corresponds to the classical form of the wave equation (where there is only one inhomogenous term f), substituting (2.35) into (2.50) gives

$$\begin{aligned} u(\mathbf{x}, t) = & \frac{1}{4\pi c^2 t^2} \int_{\partial B_{ct}(\mathbf{x})} [u_0(\mathbf{y}) + \nabla u_0(\mathbf{y}) \cdot (\mathbf{y} - \mathbf{x}) + t\phi_0(\mathbf{y})] dS(\mathbf{y}) + \\ & \frac{1}{c} \int_0^t \frac{1}{4\pi c(t-s)} \int_{\partial B_{c(t-s)}(\mathbf{x})} f(\mathbf{y}) dS(\mathbf{y}) ds, \end{aligned} \quad (2.51)$$

which is Kirchhoff's formula for the inhomogenous wave equation [35].

2.2.2 Field Dependent Source Terms

First, consider the two variable system of the form

$$\frac{dv}{dt} = \lambda v, \quad (2.52)$$

$$\frac{dw}{dt} = f(v, w), \quad (2.53)$$

where λ^{-1} is the fast time scale in the problem. Introducing a change of variables using the propagator, $\hat{v} = e^{-\lambda t}v$, the system becomes

$$\frac{d\hat{v}}{dt} = 0, \quad (2.54)$$

$$\frac{dw}{dt} = f(e^{\lambda t}\hat{v}, w). \quad (2.55)$$

The propagator integrates away the fast time scale in the problem and now this problem can be solved with any time integrator with time step being constrained only by the slower time scale, and thus the time steps can be larger. This idea can be applied to the wave equation (2.28).

When f, g are independent of u, ϕ , (2.50) is an explicit update formula to evolve u, ϕ to from initial conditions to t . However, there are many applications of Maxwell's equations where $\rho, \mathbf{J}$ are dependent on $\mathbf{E}, \mathbf{B}$. This is analogous to f, g being dependent functions of u, ϕ in the wave equation. In this case, (2.50) is then only an implicit expression for $u(t), \phi(t)$. Suppose this dependence is captured by a set of evolution equations with associated initial

conditions, then the system is extended into

$$\frac{\partial u}{\partial t} = c\phi + g, \quad (2.56)$$

$$\frac{\partial \phi}{\partial t} = c\Delta u + f, \quad (2.57)$$

$$\frac{\partial f}{\partial t} = h_f(u, \phi, f, g), \quad (2.58)$$

$$\frac{\partial g}{\partial t} = h_g(u, \phi, f, g), \quad (2.59)$$

where h_f, h_g are known functions. Assuming that the propagator for the wave equation can be computed accurately for any ct , it is possible to leverage that by a change of variables [60]. Let $\mathbf{u} = (u, \phi, f, g)^T$ and $\mathbf{f} = (g, f, h_f, h_g)^T$, consider $\hat{\mathbf{u}} = \mathcal{P}_{ext}^{-ct}[\mathbf{u}]$, and defining the extended propagator as

$$\mathcal{P}_{ext}^{ct} \left[\begin{pmatrix} u \\ \phi \\ f \\ g \end{pmatrix} \right] = \begin{pmatrix} \mathcal{P}^{ct} \left[\begin{pmatrix} u \\ \phi \end{pmatrix} \right] \\ f \\ g \end{pmatrix}. \quad (2.60)$$

Then

$$\frac{\partial \hat{\mathbf{u}}}{\partial t} = \mathcal{P}_{ext}^{-ct}[\mathbf{f}(\mathbf{x}, t, \mathcal{P}_{ext}^{ct}[\hat{\mathbf{u}}])], \quad (2.61)$$

$$\hat{\mathbf{u}}(\mathbf{x}, 0) = \mathbf{u}(\mathbf{x}, 0). \quad (2.62)$$

This removes the stiff time scale stemming from the wave equation part of the problem. To show that this IVP can recover the solution to the wave equation, let $\mathbf{u} = (u, \phi)^T$ and $\mathbf{f} = (g, f)^T$ where $\mathbf{f}$ is once again prescribed and independent of u, ϕ . Integrate in time

$$\hat{\mathbf{u}}(\mathbf{x}, t) - \hat{\mathbf{u}}(\mathbf{x}, 0) = \int_0^t \mathcal{P}^{-cs}[\mathbf{f}(\mathbf{x}, s)]ds,$$

and propagate forward to recover $\mathbf{u}$

$$\mathbf{u}(\mathbf{x}, t) = \mathcal{P}^{ct}[\mathbf{u}(\mathbf{x}, 0)] + \int_0^t \mathcal{P}^{c(t-s)}[\mathbf{f}(\mathbf{x}, s)]ds,$$

which is the same as (2.50).

2.3 Solution of the Auxiliary System

Consider the case where $\rho, \mathbf{J}$ are known functions independent of $\mathbf{E}, \mathbf{B}$. By (2.50), the solutions to (2.7)-(2.10) are

$$\begin{aligned} \begin{pmatrix} \mathbf{E}(\mathbf{x}, \tau + t) \\ \Phi(\mathbf{x}, \tau + t) \end{pmatrix} &= \mathcal{P}^{ct} \left[\begin{pmatrix} \mathbf{E}(\mathbf{x}, t) \\ \Phi(\mathbf{x}, t) \end{pmatrix} \right] - 4\pi \int_t^{\tau+t} \mathcal{P}^{c(\tau+t-s)} \left[\begin{pmatrix} \mathbf{J}(\mathbf{x}, s) \\ c\nabla\rho(\mathbf{x}, s) \end{pmatrix} \right] ds, \quad (2.63) \\ \begin{pmatrix} \mathbf{B}(\mathbf{x}, \tau + t) \\ \Psi(\mathbf{x}, \tau + t) \end{pmatrix} &= \mathcal{P}^{-ct} \left[\begin{pmatrix} \mathbf{B}(\mathbf{x}, t) \\ \Psi(\mathbf{x}, t) \end{pmatrix} \right] - 4\pi \int_t^{\tau+t} \mathcal{P}^{-c(\tau+t-s)} \left[\begin{pmatrix} 0 \\ \nabla \times \mathbf{J}(\mathbf{x}, s) \end{pmatrix} \right] ds, \quad (2.64) \end{aligned}$$

where applying the spherical means operators to a vector quantity is defined as the spherical means of each component independently. Expanding the propagator out and dropping the $\mathbf{x}$ dependence

$$\mathbf{E}(\tau + t) = \mathcal{H}^{ct}[\mathbf{E}(\tau)] + \mathcal{G}^{ct}[\Phi(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{H}^{c(\tau+t-s)}[\mathbf{J}] + c\mathcal{G}^{c(\tau+t-s)}[\nabla\rho])ds, \quad (2.65)$$

$$\Phi(\tau + t) = \mathcal{G}^{ct}[\Delta\mathbf{E}(\tau)] + \mathcal{H}^{ct}[\Phi(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{G}^{c(\tau+t-s)}[\Delta\mathbf{J}] + c\mathcal{H}^{c(\tau+t-s)}[\nabla\rho])ds, \quad (2.66)$$

$$\begin{aligned} \mathbf{B}(\tau + t) &= \mathcal{H}^{-ct}[\mathbf{B}(\tau)] + \mathcal{G}^{-ct}[\Psi(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{G}^{-c(\tau+t-s)}[\nabla \times \mathbf{J}])ds \\ &= \mathcal{H}^{ct}[\mathbf{B}(\tau)] - \mathcal{G}^{ct}[\Psi(\tau)] + 4\pi \int_{\tau}^{\tau+t} (\mathcal{G}^{c(\tau+t-s)}[\nabla \times \mathbf{J}])ds, \quad (2.67) \end{aligned}$$

$$\begin{aligned} \Psi(\tau + t) &= \mathcal{G}^{-ct}[\Delta\mathbf{B}(\tau)] + \mathcal{H}^{-ct}[\Psi(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{H}^{-c(\tau+t-s)}[\nabla \times \mathbf{J}])ds \\ &= -\mathcal{G}^{ct}[\Delta\mathbf{B}(\tau)] + \mathcal{H}^{ct}[\Psi(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{H}^{c(\tau+t-s)}[\nabla \times \mathbf{J}])ds. \quad (2.68) \end{aligned}$$

These are the integral form of the solution to the auxiliary system and in particular the $\mathbf{E}, \mathbf{B}$ solutions are also solutions to Maxwell's equations.

2.3.1 Connection with Retarded Field Solutions

Classically, the electric and magnetic fields given by time varying charge and current sources, that are independent of the fields, are given by the retarded field solutions [53, 43]. And so it can be shown that the retarded field solutions with appropriate initial conditions recovers (2.65)-(2.68). For the scalar wave equation in 3D, exchanging the order of integration on the source term turns the spherical means representation of the solution to the retarded potential representation of the solution [35]. Starting with (2.3)

$$\frac{\partial \mathbf{E}}{\partial t} = c\nabla \times \mathbf{B} - 4\pi\mathbf{J},$$

Taking curl of both sides

$$\frac{\partial \nabla \times \mathbf{E}}{\partial t} = c \nabla \times \nabla \times \mathbf{B} - 4\pi \nabla \times \mathbf{J},$$

Using $\nabla \times \nabla \times \mathbf{B} = \nabla(\nabla \cdot \mathbf{B}) - \Delta \mathbf{B}$

$$\frac{\partial \nabla \times \mathbf{E}}{\partial t} = c(\nabla(\nabla \cdot \mathbf{B}) - \Delta \mathbf{B}) - 4\pi \nabla \times \mathbf{J},$$

Substitute in (2.4) and (2.2)

$$\begin{aligned} -\frac{1}{c} \frac{\partial^2 B}{\partial t^2} &= -c \Delta \mathbf{B} - 4\pi \nabla \times \mathbf{J}, \\ \Delta \mathbf{B} - \frac{1}{c^2} \frac{\partial^2 B}{\partial t^2} &= -\frac{4\pi}{c^2} \nabla \times \mathbf{J}. \end{aligned}$$

This is the second order wave equation for $\mathbf{B}$. Taking the curl of (2.4) and following the same steps yields the wave equation for $\mathbf{E}$

$$\Delta \mathbf{E} - \frac{1}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} = 4\pi \nabla \rho + \frac{4\pi}{c^2} \frac{\partial \mathbf{J}}{\partial t}, \quad (2.69)$$

$$\Delta \mathbf{B} - \frac{1}{c^2} \frac{\partial^2 \mathbf{B}}{\partial t^2} = -\frac{4\pi}{c} \nabla \times \mathbf{J}. \quad (2.70)$$

The Green's function for the free space scalar wave equation is given by [11]

$$\Delta G - \frac{1}{c^2} \frac{\partial^2 G}{\partial t^2} = -4\pi \delta(\mathbf{x} - \mathbf{x}') \delta(t - t'), \quad (2.71)$$

$$G(\mathbf{x}, t; \mathbf{x}', t') = \frac{\delta\left((t - t') \mp \frac{|\mathbf{x} - \mathbf{x}'|}{c}\right)}{|\mathbf{x} - \mathbf{x}'|}. \quad (2.72)$$

Taking the plus sign from (2.72), using it component-wise on (2.69)-(2.70), and eliminating the t' integration with the delta distribution gives the retarded field solutions

$$\mathbf{E}(\mathbf{x}, t) = \int_{\mathbb{R}^3} \frac{1}{|\mathbf{x} - \mathbf{x}'|} \left[-\nabla' \rho - \frac{1}{c^2} \frac{\partial \mathbf{J}}{\partial t'} \right]_{\text{ret}} d\mathbf{x}', \quad (2.73)$$

$$\mathbf{B}(\mathbf{x}, t) = \int_{\mathbb{R}^3} \frac{1}{c|\mathbf{x} - \mathbf{x}'|} [\nabla' \times \mathbf{J}]_{\text{ret}} d\mathbf{x}', \quad (2.74)$$

where $[f(\mathbf{x}', t')]_{\text{ret}} = f(\mathbf{x}', t - |\mathbf{x} - \mathbf{x}'|/c)$. This is the form of the retarded field solutions to Maxwell's equations given in standard electromagnetic literature.

In general, (2.73)-(2.74) is valid $\forall (\mathbf{x}, t) \in \mathbb{R}^3 \times \mathbb{R}$. In particular they are valid also for the initial value problem (IVP). Since (2.69)-(2.70) are second order in time two initial conditions

are needed for each equation

$$\mathbf{E}(\mathbf{x}, 0) = \mathbf{E}_0, \quad (2.75)$$

$$\mathbf{B}(\mathbf{x}, 0) = \mathbf{B}_0, \quad (2.76)$$

$$\left. \frac{\partial \mathbf{E}}{\partial t} \right|_{t=0} = c \nabla \times \mathbf{B}_0 - 4\pi \mathbf{J}_0, \quad (2.77)$$

$$\left. \frac{\partial \mathbf{B}}{\partial t} \right|_{t=0} = -c \nabla \times \mathbf{E}_0, \quad (2.78)$$

where $\mathbf{J}_0 = \mathbf{J}(\mathbf{x}, 0)$, again assuming $\mathbf{E}_0, \mathbf{B}_0$ satisfy (2.1)-(2.2), and (2.3)-(2.4) are used to obtain the first derivative initial conditions. Using Green's Theorem [42], the retarded field solutions with initial conditions are given by

$$\begin{aligned} \mathbf{E}(\mathbf{x}, t) = & \int_{\mathbb{R}^3} \frac{1}{4\pi c^2} \frac{\partial}{\partial t} (G(\mathbf{x}, t; \mathbf{x}', 0)) \mathbf{E}_0 d\mathbf{x}' + \int_{\mathbb{R}^3} \frac{1}{4\pi c^2} G(\mathbf{x}, t; \mathbf{x}', 0) (c \nabla \times \mathbf{B}_0 - 4\pi \mathbf{J}_0) d\mathbf{x}' + \\ & \int_{\mathbb{R}^3} \frac{1}{|\mathbf{x} - \mathbf{x}'|} \left[-\nabla' \rho - \frac{1}{c^2} \frac{\partial \mathbf{J}}{\partial t'} \right]_{\text{ret}} d\mathbf{x}', \end{aligned} \quad (2.79)$$

$$\begin{aligned} \mathbf{B}(\mathbf{x}, t) = & \int_{\mathbb{R}^3} \frac{1}{4\pi c^2} \frac{\partial}{\partial t} (G(\mathbf{x}, t; \mathbf{x}', 0)) \mathbf{B}_0 d\mathbf{x}' - \int_{\mathbb{R}^3} \frac{1}{4\pi c^2} G(\mathbf{x}, t; \mathbf{x}', 0) c \nabla \times \mathbf{E}_0 d\mathbf{x}' + \\ & \int_{\mathbb{R}^3} \frac{1}{c|\mathbf{x} - \mathbf{x}'|} [\nabla' \times \mathbf{J}]_{\text{ret}} d\mathbf{x}'. \end{aligned} \quad (2.80)$$

These equations can be shown to be identical to the $\mathbf{E}$ (2.65), $\mathbf{B}$ (2.67) solutions for the auxiliary system. Note that

$$\begin{aligned} \int_{\mathbb{R}^3} G(\mathbf{x}, t; \mathbf{x}', t') f(\mathbf{x}', t') d\mathbf{x}' &= \int_{\mathbb{R}^3} f(\mathbf{x}', t') \delta \left(t' - t + \frac{|\mathbf{x} - \mathbf{x}'|}{c} \right) \frac{1}{|\mathbf{x} - \mathbf{x}'|} d\mathbf{x}' \\ &= 4\pi \int_{\mathbb{R}^3} f(\mathbf{x}', t') c \frac{\delta(|\mathbf{x} - \mathbf{x}'| - c(t - t'))}{4\pi |\mathbf{x} - \mathbf{x}'|} d\mathbf{x}' \\ &= 4\pi \int_{\mathbb{R}^3} f(\mathbf{x}', t') c \frac{\delta(|\mathbf{x} - \mathbf{x}'| - c(t - t'))}{4\pi c(t - t')} d\mathbf{x}' \\ &= 4\pi c \mathcal{G}^{c(t-t')} [f(t')]. \end{aligned} \quad (2.81)$$

Taking the time derivative

$$\begin{aligned} \int_{\mathbb{R}^3} \frac{1}{c} \frac{\partial}{\partial t} (G(\mathbf{x}, t; \mathbf{x}', t') f(\mathbf{x}', t')) d\mathbf{x}' &= 4\pi c \frac{1}{c} \frac{\partial \mathcal{G}^{c(t-t')} [f(t')]}{\partial t} \\ &= 4\pi c \mathcal{H}^{t-t'} [f(t')]. \end{aligned} \quad (2.82)$$

Finally, the retarded time operator, $[\cdot]_{\text{ret}}$, can be converted into spherical means as well. For an IVP, $t' \geq 0$. In addition, $|\mathbf{x} - \mathbf{x}'|/c \geq 0$ and so $t' \leq t$. Putting both together gives that

$t' \in [0, t]$ and using (2.81)

$$\begin{aligned} \int_{\mathbb{R}^3} \frac{1}{|\mathbf{x} - \mathbf{x}'|} [f(\mathbf{x}', t')]_{\text{ret}} d\mathbf{x}' &= \int_{\mathbb{R}^3} \left(\int_0^t f(\mathbf{x}', t') \delta \left(t' - t + \frac{|\mathbf{x} - \mathbf{x}'|}{c} \right) dt' \right) \frac{1}{|\mathbf{x} - \mathbf{x}'|} d\mathbf{x}' \\ &= 4\pi \int_0^t c \mathcal{G}^{t-t'} [f(t')] dt'. \end{aligned} \quad (2.83)$$

Using (2.81)-(2.83) with (2.80)

$$\mathbf{B}(\mathbf{x}, t) = \mathcal{H}^{ct}[\mathbf{B}_0] - \mathcal{G}^{ct}[\nabla \times \mathbf{E}_0] + 4\pi \int_0^t \mathcal{G}^{t-t'} [\nabla \times \mathbf{J}] dt', \quad (2.84)$$

which is equal to (2.67) with $s = t'$ and $\tau = 0$. For $\mathbf{E}$, first consider the solution without initial values

$$\begin{aligned} \mathbf{E}(\mathbf{x}, t) &= \int_{\mathbb{R}^3} \frac{1}{|\mathbf{x} - \mathbf{x}'|} \left[-\nabla' \rho - \frac{1}{c^2} \frac{\partial \mathbf{J}}{\partial t'} \right]_{\text{ret}} d\mathbf{x}' \\ &= 4\pi \int_0^t -c \mathcal{G}^{t-t'} [\nabla' \rho] - \frac{1}{c} \mathcal{G}^{t-t'} \left[\frac{\partial \mathbf{J}}{\partial t'} \right] dt'. \end{aligned} \quad (2.85)$$

Using integration by parts on the $\mathbf{J}$ term

$$\begin{aligned} \int_0^t \mathcal{G}^{t-t'} \left[\frac{\partial \mathbf{J}}{\partial t'} \right] dt' &= \left(\mathcal{G}^{t-t'} [\mathbf{J}] \right) \Big|_0^t - \int_0^t \frac{\partial \mathcal{G}^{t-t'} [\mathbf{J}]}{\partial t'} dt' \\ &= -\mathcal{G}^{ct}[\mathbf{J}_0] + \int_0^t c \mathcal{H}^{t-t'} [\mathbf{J}] dt'. \end{aligned}$$

Using (2.81)-(2.83) and (2.85)-(2.86) with (2.79),

$$\begin{aligned} \mathbf{E}(\mathbf{x}, t) &= \mathcal{H}^{ct}[\mathbf{E}_0] - \mathcal{G}^{ct}[\nabla \times \mathbf{B}_0] - \frac{4\pi}{c} \mathcal{G}^{ct}[\mathbf{J}_0] + \frac{4\pi}{c} \mathcal{G}^{ct}[\mathbf{J}_0] + 4\pi \int_0^t \mathcal{H}^{t-t'} [\mathbf{J}] - c \mathcal{G}^{t-t'} [\nabla \rho] dt' \\ &= \mathcal{H}^{ct}[\mathbf{E}_0] - \mathcal{G}^{ct}[\nabla \times \mathbf{B}_0] - 4\pi \int_0^t \mathcal{H}^{t-t'} [\mathbf{J}] + c \mathcal{G}^{t-t'} [\nabla \rho] dt', \end{aligned} \quad (2.86)$$

which is equal to (2.65) again with $s = t'$ and $\tau = 0$.

If this formulation is chosen as the starting point, taking the curls and applying the constraints gives (2.66) and (2.68). At first glance it would seem following this formulation is computationally cheaper because there are only two field quantities to evolve instead of four. However, once a quadrature scheme is selected, the two formulations result in identical discretized equations.

2.3.2 Connection with the Maxwell Propagator

(2.65)-(2.68) can also be derived starting from the propagator from Maxwell's equations directly [83, 68]. This derivation shows that the $\nabla \rho$ term in the integrals are there to

offset the error in propagating the longitudinal part of the electric field. Taking the Fourier transform in $\mathbf{x}$, the two evolution equations in Maxwell's equations for the homogenous case are

$$\frac{\partial}{\partial t} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \\ \tilde{B}_x \\ \tilde{B}_y \\ \tilde{B}_z \end{pmatrix} = ic \begin{pmatrix} 0 & 0 & 0 & 0 & -k_y & k_z \\ 0 & 0 & 0 & -k_z & 0 & k_x \\ 0 & 0 & 0 & k_y & -k_x & 0 \\ 0 & k_y & -k_z & 0 & 0 & 0 \\ k_z & 0 & -k_x & 0 & 0 & 0 \\ -k_y & k_x & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \\ \tilde{B}_x \\ \tilde{B}_y \\ \tilde{B}_z \end{pmatrix} \quad (2.87)$$

The matrix exponential for the linear differential operator is given by

$$\tilde{\mathcal{P}}_{maxwell}^{ct} = \begin{pmatrix} \cos(c|\mathbf{k}|t)(\mathbb{I} - \hat{\mathbf{k}}\hat{\mathbf{k}}^T) & \frac{\sin(c|\mathbf{k}|t)}{|\mathbf{k}|}i\mathbf{k} \times \\ -\frac{\sin(c|\mathbf{k}|t)}{|\mathbf{k}|}i\mathbf{k} \times & \cos(c|\mathbf{k}|t)(\mathbb{I} - \hat{\mathbf{k}}\hat{\mathbf{k}}^T) \end{pmatrix}, \quad (2.88)$$

where $\mathbb{I}$ is the 3x3 identity matrix and

$$\mathbf{k} \times = \begin{pmatrix} 0 & k_y & -k_z \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{pmatrix}.$$

Taking the inverse Fourier transform gives

$$\mathcal{P}_{maxwell}^{ct} \left[\begin{pmatrix} \mathbf{E} \\ \mathbf{B} \end{pmatrix} \right] = \begin{pmatrix} \mathcal{H}^{ct}[\mathbb{P}[\mathbf{E}]] + \mathcal{G}^{ct}[\nabla \times \mathbf{B}] \\ -\mathcal{G}^{ct}[\nabla \times \mathbf{E}] + \mathcal{H}^{ct}[\mathbb{P}[\mathbf{B}]] \end{pmatrix}, \quad (2.89)$$

where for $\mathbf{A} \in C^2(\mathbb{R}^3)$

$$\mathbb{P}[\mathbf{A}] = (\mathbb{I} - \text{grad } \Delta^{-1} \text{div})\mathbf{A}, \quad (2.90)$$

is the Helmholtz decomposition operator that gives the divergence free part of the field. We will also define the complement projection operator

$$\mathbb{Q}[\mathbf{A}] = \text{grad } \Delta^{-1} \text{div } \mathbf{A}, \quad (2.91)$$

which gives the curl free part of the field. (2.89) can be simplified because for Maxwell's equations, $\nabla \cdot \mathbf{B} = 0$ and there are no source terms associated with the evolution of $\mathbf{B}$

$$\mathcal{P}_{maxwell}^{ct} \left[\begin{pmatrix} \mathbf{E} \\ \mathbf{B} \end{pmatrix} \right] = \begin{pmatrix} \mathcal{H}^{ct}[\mathbb{P}[\mathbf{E}]] + \mathcal{G}^{ct}[\nabla \times \mathbf{B}] \\ -\mathcal{G}^{ct}[\nabla \times \mathbf{E}] + \mathcal{H}^{ct}[\mathbf{B}] \end{pmatrix}. \quad (2.92)$$

Given fields at time τ , the solution at $\tau + t$ with this propagator is then given by

$$\mathbf{E}(\tau + t) = \mathbb{P}[\mathbf{E}(\tau + t)] + \mathbb{Q}[\mathbf{E}(\tau + t)], \quad (2.93)$$

$$\mathbb{P}[\mathbf{E}(\tau + t)] = \mathcal{H}^{ct}[\mathbb{P}[\mathbf{E}(\tau)]] + \mathcal{G}^{ct}[(\nabla \times \mathbf{B}(\tau))] - 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)}[\mathbb{P}[\mathbf{J}]] ds, \quad (2.94)$$

$$\mathbf{B}(\tau + t) = \mathcal{H}^{ct}[\mathbf{B}(\tau)] - \mathcal{G}^{ct}[(\nabla \times \mathbf{E}(\tau))] + 4\pi \int_{\tau}^{\tau+t} \mathcal{G}^{c(\tau+t-s)}[(\nabla \times \mathbf{J})] ds. \quad (2.95)$$

The other part of the electric field is obtained by using Gauss's Law (2.1)

$$\mathbb{Q}[\mathbf{E}(\tau + t)] = -4\pi \nabla(\Delta^{-1} \rho(\tau + t)). \quad (2.96)$$

By inspection, (2.67) and (2.95) are the same. For $\mathbf{E}$, first

$$\begin{aligned} -4\pi \int_{\tau}^{\tau+t} c \mathcal{G}^{c(\tau+t-s)}[\nabla \rho] ds &= -4\pi \int_{\tau}^{\tau+t} c \mathcal{G}^{c(\tau+t-s)}[\Delta \nabla(\Delta^{-1} \rho)] ds \\ &= \int_{\tau}^{\tau+t} \frac{\partial \mathcal{H}^{c(\tau+t-s)}}{\partial s} [\mathbb{Q}[\mathbf{E}]] ds \\ &= \mathcal{H}^{c(\tau+t-s)}[\mathbb{Q}[\mathbf{E}]] \Big|_{\tau}^{t+\tau} - \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)} \left[\frac{\partial \mathbb{Q}[\mathbf{E}]}{\partial s} \right] ds \\ &= \mathcal{H}^0[\mathbb{Q}[\mathbf{E}(\tau + t)]] - \mathcal{H}^{ct}[\mathbb{Q}[\mathbf{E}(\tau)]] + 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)} \left[\nabla \left(\Delta^{-1} \frac{\partial \rho}{\partial s} \right) \right] ds \\ &= \mathbb{Q}[\mathbf{E}(\tau + t)] - \mathcal{H}^{ct}[\mathbb{Q}[\mathbf{E}(\tau)]] + 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)} [\nabla(\Delta^{-1} \nabla \cdot \mathbf{J})] ds \\ &= \mathbb{Q}[\mathbf{E}(\tau + t)] - \mathcal{H}^{ct}[\mathbb{Q}[\mathbf{E}(\tau)]] + 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)} [\mathbb{Q}[\mathbf{J}]] ds. \end{aligned} \quad (2.97)$$

Using (2.97) and writing out (2.93)

$$\begin{aligned} \mathbf{E}(\tau + t) &= \mathbb{P}[\mathbf{E}(\tau + t)] + \mathbb{Q}[\mathbf{E}(\tau + t)] \\ &= \mathcal{H}^{ct}[\mathbb{P}[\mathbf{E}(\tau)]] + \mathcal{G}^{ct}[\nabla \times \mathbf{B}(\tau)] - 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)}[\mathbb{P}[\mathbf{J}]] ds + \mathbb{Q}[\mathbf{E}(\tau + t)] \\ &= \mathcal{H}^{ct}[\mathbf{E}(\tau)] + \mathcal{G}^{ct}[\nabla \times \mathbf{B}(\tau)] - 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)}[\mathbf{J}] ds + \\ &\quad \left(\mathbb{Q}[\mathbf{E}(\tau + t)] - \mathcal{H}^{ct}[\mathbb{Q}[\mathbf{E}(\tau)]] + 4\pi \int_{\tau}^{\tau+t} \mathcal{H}^{c(\tau+t-s)}[\mathbb{Q}[\mathbf{J}]] ds \right) \\ &= \mathcal{H}^{ct}[\mathbf{E}(\tau)] + \mathcal{G}^{ct}[\nabla \times \mathbf{B}(\tau)] - 4\pi \int_{\tau}^{\tau+t} (\mathcal{H}^{c(\tau+t-s)}[\mathbf{J}] + c \mathcal{G}^{c(\tau+t-s)}[\nabla \rho]) ds, \end{aligned} \quad (2.98)$$

which is the same as (2.65). It is clear that $\nabla \rho$ integration term cancels the error created by propagating $\mathbb{Q}[\mathbf{E}]$ and $\mathbb{Q}[\mathbf{J}]$, and adds in the correct solution for $\mathbb{Q}[\mathbf{E}]$.

Chapter 3

Local Discrete Convolution Method (LDCM)

For the computational domain, first consider a single-level rectangular grid with uniform spacing h in all spatial directions. All the field quantities are approximated on the same grid points. Consider first the case where there are no sources. The cornerstone of LDCM is to discretize the propagator and apply it on the data on the rectangular grid. Formally, the spherical means integrals can be represented as convolutions with continuous spherical delta distributions. This suggests a numerical scheme for the spherical means by discretizing these delta distributions and computing the discrete convolutions. This is done using the framework outlined in [84] which shows how to discretize integration on arbitrary surfaces using regularized delta distributions for rectangular grids. The three parameters in consideration are the order of accuracy, the support, and the smoothness of the resultant discrete convolution kernel. These parameters are linked by the underlying choice of the 1D regularizer, the cardinal B -spline. For LDCM, the main idea is to retain the locality of the propagator even in its discrete form, this is achieved by using standard centered-difference approximations for the derivatives and creating convolution kernels with compact support. This is because convolutions with kernels with compact support can be efficiently computed with Hockney's method [51] and a discretization with compact support can be parallelized using domain decomposition. If the source terms are prescribed functions then one can use a quadrature scheme, such as closed Newton-Cotes quadrature for (2.65)-(2.68). Otherwise, the temporal discretization reduces to choosing a time integrator for (2.61). LDCM is compared with other propagator based numerical methods and an analysis for its phase and amplitude errors are given. The chapter then discusses some implementation issues and concludes with a summary of the overall LDCM algorithm.

3.1 Centered-Difference Approximations

In order to discretize the propagator, it is necessary to discretize the derivative operators. This can be achieved with centered-difference approximations. Centered-differences approx-

imations for derivatives are derived from taking the difference of the Taylor expansion of the function evaluated at different points and then taking the weighted difference of the resultant series where the weights are tuned to eliminate as many terms as possible in the series. For example, the second-order accurate centered-difference formula for the first derivative can be derived starting with the Taylor series of the function f around x evaluated at two points

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \dots, \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \dots, \end{aligned}$$

taking the difference of the two series

$$\begin{aligned} f(x+h) - f(x-h) &= 2hf'(x) + \frac{h^3}{3}f'''(x) + \mathcal{O}(h^4), \\ f'(x) &= \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2). \end{aligned}$$

In order to generate higher order approximations, it is necessary to include more points to eliminate the higher order error terms in the approximation. Given an order of accuracy, q , the weights for a centered-difference approximation can be derived with the following procedure. Define

$$v = \left[-\frac{q}{2}, -\frac{q}{2} + 1, \dots, -\frac{q}{2} + q \right]. \quad (3.1)$$

Let exponentiating a vector be defined as element wise power and define $A \in \mathbb{R}^{(q+1) \times (q+1)}$

$$A = \begin{bmatrix} v^0 \\ v^1 \\ v^2 \\ \vdots \\ v^q \end{bmatrix}. \quad (3.2)$$

Then the weights for the first and second derivatives are given by

$$Ad_1 = \mathbf{e}_1, Ad_2 = 2\mathbf{e}_2, \quad (3.3)$$

where $\mathbf{e}_1 = [0, 1, 0, 0, \dots, 0]$ and $\mathbf{e}_2 = [0, 0, 1, 0, 0, \dots, 0]$. The centered-difference approximations are

$$f'(x) = \frac{1}{h} \sum_{m=-q/2}^{q/2} d_{1m} f(x+mh) + \mathcal{O}(h^q) \quad (3.4)$$

$$f''(x) = \frac{1}{h^2} \sum_{m=-q/2}^{q/2} d_{2m} f(x+mh) + \mathcal{O}(h^q) \quad (3.5)$$

Note that the support of the q -th order centered-difference approximation for the first and second derivative is $[-q/2, q/2]$. Furthermore, the coefficients for the first derivative is antisymmetric about zero and the coefficients for the second derivative is symmetric about zero.

3.2 Discretizing Spherical Means

The next piece in the spatial discretization scheme is converting the spherical means integrals into a constant coefficient stencil operation on the rectangular grid just like the centered-difference approximations for the derivatives [65]. That is, it is necessary to construct a function $\delta_h(\partial B_{ct}(\mathbf{y}), g, \mathbf{x}_j)$ such that

$$\sum_{j \in \mathbb{Z}^d} \delta_h(\partial B_{ct}(\mathbf{y}), g(\mathbf{z}, \mathbf{y}), \mathbf{x}_j) f(\mathbf{x}_j) \approx \int_{\partial B_{ct}(\mathbf{y})} g(\mathbf{z}, \mathbf{y}) f(\mathbf{z}) d\mathbf{z}, \quad (3.6)$$

where $g = 1$ or $g = z_i - y_i$, depending on the operator being discretized on the grid. $\delta_h(\partial B_{ct}(\mathbf{y}), g, \mathbf{x}_j)$ is also known as a regularized delta distribution and creating high dimensional regularized delta distributions with arbitrary weights and geometry is discussed in [84]. This section outlines the main theorems and definitions for the framework from this paper. The three considerations in regularizing a delta distribution are: its accuracy when used as an interpolant, the size of its support, and its smoothness. The framework described in this section will link these three parameters together. For the purposes of the numerical scheme, the accuracy of the regularized delta distribution is a tuning parameter that depends on the desired order of accuracy for the approximate solution. That is, if a $\mathcal{O}(h^q)$ one-step error is required, then the finite difference scheme for the derivative operators and the regularized delta distribution must also be at least q -order accurate. For fixed CFL studies, the overall error will be $\mathcal{O}(h^{q-1})$, hereafter the order of accuracy of the propagator refers to this overall error and not the one step error.

In order to update the fields with the propagator, it is necessary to compute the spherical means centered at every grid point. Since the sphere and weighting function are both only a function of distance from the center, this results in stencils that are the same for every point. Therefore it is sufficient to create only one regularizer with a sphere centered at $\mathbf{y} = 0$, evaluate it to generate the stencil, and compute the propagator updates with discrete convolutions.

The basis of generating regularized high dimensional delta distributions is a regularized 1D delta distribution [14]. The accuracy of a regularized 1D delta distribution is dependent on the number of discrete moment conditions it satisfies.

Definition 1 (Discrete moment conditions). *Given a grid spacing $h > 0$, a function $\delta_h : \mathbb{R} \rightarrow \mathbb{R}$ is in the function class Q^q if the support of δ_h contains $[-mh, mh]$ for some $m > 0$*

and

$$h \sum_{j \in \mathbb{Z}} (jh - \bar{x})^r \delta_h(jh - \bar{x}) = \begin{cases} 1, & r = 0, \\ 0, & 1 \leq r < q \end{cases}, \quad (3.7)$$

for any $\bar{x} \in \mathbb{R}$.

The conditions in Definition 1 are simply discrete analogues of the continuous moment conditions

$$\int_{-\infty}^{\infty} (x - \bar{x})^r \delta(x - \bar{x}) dx = \begin{cases} 1, & r = 0, \\ 0, & 1 \leq r < q, \end{cases}, \quad (3.8)$$

which are satisfied by the delta distribution for arbitrarily large q . For sufficiently regular functions f , the discrete moment conditions are sufficient for consistency of δ_h .

Proposition 1 (Consistency of 1D discrete delta distribution [14]). *For $\delta_h \in Q^q$ and $f \in C^q$*

$$f(\bar{x}) - h \sum_{j \in \mathbb{Z}} f(jh) \delta_h(jh - \bar{x}) = \mathcal{O}(h^q), \quad (3.9)$$

for any $\bar{x} \in \mathbb{R}$.

This can be seen by Taylor expanding $f(jh)$ around $f(\bar{x})$ and applying the discrete moment conditions up to order q ,

$$h \sum_{j \in \mathbb{Z}} f(jh) \delta_h(jh - \bar{x}) = h \sum_{j \in \mathbb{Z}} \delta_h(jh - \bar{x}) \left[\sum_{p=0}^{q-1} \frac{h^p}{p!} (jh - \bar{x})^p f^{(p)}(\bar{x}) + \mathcal{O}(h^q) \right] \quad (3.10)$$

With a $\delta_h \in Q^q$, it is straightforward to construct higher-dimensional regularized delta distributions as a product of 1D regularized delta distributions satisfying (3.7).

Theorem 1 (Consistency of multi-dimensional discrete delta distribution [84]). *Let Γ be a continuous and bounded surface, $g \in C(\mathbb{R}^d)$, and $\delta_{h_k} \in Q^q$ for $k = 1, \dots, d$. Define the multi-dimensional function*

$$\delta_h(\Gamma, g, \mathbf{x}) \equiv \int_{\Gamma} \prod_{k=1}^d \delta_{h_k}(x_k - z_k) w(\mathbf{z}) d\mathbf{z}, \quad (3.11)$$

where $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$ and $\mathbf{z} = (z_1, \dots, z_d) \in \Gamma$. Suppose $f \in C^r(\mathbb{R}^d)$. Then

$$\left(\prod_{k=1}^d h_k \right) \sum_{j \in \mathbb{Z}^d} \delta_h(\Gamma, g, \mathbf{x}_j) f(\mathbf{x}_j) - \int_{\Gamma} w(\mathbf{z}) f(\mathbf{z}) d\mathbf{z} = \mathcal{O}(h^q), \quad (3.12)$$

where $\mathbf{x}_j = (j_1 h, \dots, j_d h)$ is a Cartesian gridpoint.

For the spherical means integrals stencil, $\Gamma = \partial B_{ct}(0)$ and $w(\mathbf{z}) = g(\mathbf{z}, 0)$. Since there is an analytic expression for the integrand of (3.11), the integral can be evaluated with Atkinson quadrature [8]¹

$$\begin{aligned} \delta_h(\partial B_{ct}(0), g, \mathbf{x}) &= \int_{\partial B_{ct}(0)} \left(\prod_{k=1}^d \delta_{h_k}(x_k - z_k) g(\mathbf{z}, 0) \right) d\mathbf{z} \\ &\approx \frac{\pi(ct)^2}{m} \sum_{j=1}^{2m} \sum_{i=1}^m w_i \left(\prod_{k=1}^d \delta_{h_k}(x_k - z_{ij,k}) g(\mathbf{z}_{ij}, 0) \right), \end{aligned} \quad (3.13)$$

where $\mathbf{z}_{ij}$ has polar coordinates (ct, θ_i, ϕ_j) with $\cos \theta_i$ and w_i the Gauss-Legendre nodes and weights on $[-1, 1]$ and ϕ_j evenly spaced on $[0, 2\pi]$. This quadrature scheme integrates exactly any polynomial f that has degree less than $2m$. If f is k times differentiable, then the error associated with this quadrature scheme is

$$\left| \int_{\partial B_r} f(\mathbf{z}) d\mathbf{z} - \frac{\pi r^2}{m} \sum_{j=1}^{2m} \sum_{i=1}^m w_i f(\mathbf{z}_{ij}) \right| \leq \frac{C}{(2m-1)^k}. \quad (3.14)$$

This fully describes the discretization of a spherical means integral with the sphere centered at the origin. In order to discretize the propagator on a rectangular grid, for every grid point there are spherical means integrals centered on that point. However, since the weighting function is only a function of distance away from the center of the sphere, the discretization over the entire grid can be represented as a stencil operation where the stencil is the spherical means integral centered at the origin. With the spherical means stencil created, the continuous spherical means operator on the grid can now be calculated with discrete convolutions. Therefore

$$\mathcal{G}^{ct}[f] \approx G^{ct} * f \equiv \frac{1}{4\pi ct} \delta_h(\partial B_{ct}(0), 1, \mathbf{x}) * f, \quad (3.15)$$

$$\mathcal{G}_l^{ct}[f] \approx G_l^{ct} * f \equiv \frac{1}{4\pi ct} \delta_h(\partial B_{ct}(0), z_l, \mathbf{x}) * f, \quad (3.16)$$

where

$$G^{ct}[k] = \frac{ct}{4m} \sum_{j=1}^{2m} \sum_{i=1}^m \left[w_i \left(\prod_{d=1}^3 \delta_{h_d}(x_{k,d} - z_{ij,d}) \right) \right], \quad (3.17)$$

$$G_l^{ct}[k] = \frac{ct}{4m} \sum_{j=1}^{2m} \sum_{i=1}^m \left[w_i \left(\prod_{d=1}^3 \delta_{h_d}(x_{k,d} - z_{ij,d}) \right) z_{ij,l} \right], \quad (3.18)$$

where $\mathbf{x}_k$ is a Cartesian gridpoint.

¹Since the geometry is a sphere there is also the more efficient and specialized Lebedev quadrature [61].

3.3 High Order B -Splines

In order to construct (3.13), it is necessary to construct a regularized 1D delta distribution, $\delta_{h_k}(z)$. To derive a function $\delta_h \in Q^q$, first define the unscaled approximant W such that

$$\delta_h(x) = \frac{1}{h} W(x/h). \quad (3.19)$$

Theorem 2 from [79] gives sufficient conditions for the discrete moment conditions in terms of the behavior of the Fourier transform of W .

Theorem 2 (Continuous moment conditions [79]). *Consider the approximation formula,*

$$f_{app}(\bar{x}) = \sum_{j \in \mathbb{Z}} W(j - \bar{x}) f(j) \quad (3.20)$$

Suppose that W decays sufficiently quickly, i.e. , $|W(x)| \leq A \exp(-B|x|)$ for some constants A and B . Then, the interpolation formula is of degree q if the following two conditions hold:

1. *The function $\tilde{W}(k) - 1$ has a zero of order q at $k = 0$.*
2. *The function $\tilde{W}(k)$ has a zero of order q at $k = 2\pi j$ for integer $j \neq 0$.*

The first condition in Theorem 2 is equivalent to the continuous moment conditions in (3.8) and the second condition arises from the periodic summation of the spectrum of W due to sampling. Due to a theorem in [84] there is the following minimum bound on the support of functions in Q^q .

Theorem 3 (Minimum support for an approximate delta function [84]). *There exists a function $W \in Q^q$ if and only if the support of W contains the interval $[-q/2, q/2]$. Furthermore, for each choice of q there is a unique W that achieves this minimum support.*

Theorem 3 links the order of accuracy of the interpolant, and therefore the desired order of accuracy for the numerical solution, and the support of the interpolant. However, this theorem does not impose any guarantees or restrictions on the smoothness of the interpolant. In general, W is not smooth (or even continuous). In order for the error (3.14) from the quadrature (3.13) to converge, the integrand must be sufficiently smooth. However, the approximate delta distribution is used as a convolution kernel and the discrete convolution and spherical quadrature commute. Therefore, the smoothness of the fields and source term can be used for the quadrature to converge. This allows the method to use approximants with minimum support. Using an approximant with compact support allows for the discrete convolution kernels to retain the locality property of the continuous spherical delta distributions. One natural candidate for functions in Q^q with compact support are high order cardinal B -splines [26, 16, 18, 79, 70].

A cardinal B -spline of order q is a piecewise polynomial function of degree $q - 1$. The points where the polynomials meet are known as knots and for a cardinal B -spline the knots

are equidistant. Furthermore, the first $q - 1$ derivatives are continuous across the knots. Cardinal B -splines of degree q can be defined via the recursion

$$M_q = M_{q-1} * M_1, \quad M_1 = \chi_{[-1/2, 1/2]}, \quad (3.21)$$

where $\chi_{[a, b]}$ is the indicator function of the interval $[a, b]$. $M_q \in C^{q-1}$ is supported on the interval $[-q/2, q/2]$ and its Fourier transform is given by

$$\tilde{M}_q(k) = \left(\frac{\sin(k/2)}{k/2} \right)^q. \quad (3.22)$$

which has zeros of order q at non-zero integer multiples of 2π . Unfortunately, $\tilde{M}_q(k) - 1$ has zeros of only order 2 at $k = 0$, restricting B -splines to only second-order approximations of the discrete delta distribution [70].

Based on these facts, suppose for simplicity that q is even (for odd q a minimum support interpolant is not continuous), introduce the maximal order minimal support (MOMS) interpolant [16]

$$\tilde{W}_q(k) = \left(\sum_{p=0}^{q/2-1} a_{2p} k^{2p} \right) \tilde{M}_q(k). \quad (3.23)$$

Since the inverse Fourier transform of $(ik)^p = \frac{d^p}{dx^p}$

$$W_q(x) = \sum_{p=0}^{q/2-1} a_{2p} (-1)^p M_q^{(2p)}(x), \quad (3.24)$$

i.e. $W_q(x)$ is a linear combination of the B -spline and its derivatives, leading once again to a piecewise polynomial spline supported on $[-q/2, q/2]$. Since $M_q(x) \in C^{q-2}$ and this formula takes up to the $q-2$ derivatives of $M_q(x)$, $W_q(x) \in C^0$. Furthermore, $\tilde{W}_q(k)$ still has zeros of order q at $j2\pi$ for integer $j \neq 0$, regardless of the choice of coefficients a_p . Finally, $\tilde{W}_q(k)$ is real and even, therefore leading to a $W_q(x)$ that is real and symmetric. It remains to choose these coefficients such that $\tilde{W}_q(k) - 1$ has zeros of order q at $k = 0$.

The Taylor expansion of $\tilde{M}_q(k)$ about $k = 0$ is

$$\tilde{M}_q(k) = \sum_{p=0}^{q/2-1} b_{2p} k^{2p} + \mathcal{O}(k^q). \quad (3.25)$$

Since $\tilde{M}_q(k)$ is an even function, all odd coefficients are necessarily zero. Then, the Taylor expansion of $\tilde{W}_q(k)$ about $k = 0$ is

$$\tilde{W}_q(k) = \sum_{m=0}^{q/2-1} \left(\sum_{p=0}^m b_{2p} a_{2m-2p} \right) k^{2m} + \mathcal{O}(k^q). \quad (3.26)$$

To ensure $\tilde{W}_q(k) - 1$ has zeros of the appropriate order at $k = 0$, choose a_p such $a_0 b_0 = 1$ and the rest of the coefficients to be 0. This leads to $q/2$ equations in $q/2$ unknowns in a triangular system of the form

$$\begin{bmatrix} b_0 & 0 & 0 & \dots & 0 \\ b_2 & b_0 & 0 & \dots & 0 \\ b_4 & b_2 & b_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{q-2} & b_{q-4} & b_{q-6} & \dots & b_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_2 \\ a_4 \\ \vdots \\ a_{q-2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (3.27)$$

Since $\tilde{M}_q(0) = b_0 = 1$, this system is nonsingular, thus yielding a unique set of coefficients that lead to a $\tilde{W}_q(k)$ satisfying the conditions of Theorem 2. Each column of the matrix is the first q coefficient to the Taylor series expansion of the basis function $k^i \tilde{M}_n$ for the even powers.

Discrete delta distributions matching moment conditions to order q correspond directly with interpolation kernels on uniform grids that integrate exactly polynomials of less than order q [84], so the fact that both methods achieve the minimum support size of $[-q/2, q/2]$ means they are one and the same by Theorem 3. To summarize, the steps to regularize a spherical distribution with radius ct with a minimal support regularizer, given a desired order of accuracy q and uniform grid spacing h is:

1. Create a high order B -spline as a 1D regularized delta, δ_h
 - a) Compute the first q Taylor expansion coefficients of $\left(\frac{\sin(k/2)}{k/2}\right)^q$
 - b) Solve for the coefficients a_{2p} using (3.27)
 - c) Construct $M_q(x)$ using the recursion relation (3.21)
 - d) Construct $W_q(x)$ using (3.24)
2. Use the product $\prod_{j=1}^d \delta_h(x_j)$ to define (3.13)
3. Integrate the product over the sphere with radius ct and with appropriate weight (3.11)
4. Use spherical quadrature to compute the integral (3.13)

3.3.1 Smooth High Order B -Splines

While using minimal support, and thus C^0 , high order B -splines are sufficient for the convolution kernels, this is not sufficient for the coarse-fine interpolants used in AMR. The accuracy of the spherical quadrature (3.14) depends on the smoothness of the integrand and this smoothness requirement has been imposed on the fields instead because discrete convolution and the numerical quadrature commute. Therefore, the interpolant must be

sufficiently smooth to guarantee that the interpolated fields have the required smoothness. This same high order B -spline framework can be used to construct high order interpolants with higher smoothness. By using high order B -splines, it can be seen that smoothness (m) is connected with support size (s) and accuracy (q) by $s = q + m$. Therefore, kernels with slightly larger support but higher degrees of smoothness can be constructed with this method by modifying (3.23). If the desired order of accuracy is q and the desired smoothness is m , then it follows that

$$\tilde{W}_q(k) = \left(\sum_{p=0}^{q/2-1} a_{2p} k^{2p} \right) \tilde{M}_{q+m}(k). \quad (3.28)$$

However, to ensure that the high order B -spline is real and symmetric, only even power derivatives has been used. This means that for certain level of continuity and accuracy would require a base B -spline with order (and thus support) that's higher than the integer sum of continuity and accuracy. There is a slight variant to for W_q that utilizes all derivatives, *i.e.* both even and odd, and so it is possible to achieve smaller support for $W_q \in C^m$, $m > 0$ [18]

$$W_q(x) = \sum_{p=0}^m a_p x^p \frac{\partial^p}{\partial x^p} M_{q+m}. \quad (3.29)$$

This is based on the observation that for $r \leq n$

$$\begin{aligned} \frac{1}{(-1)^r r!} \int_{\mathbb{R}} (x')^r \frac{\partial^r}{\partial x^r} [M_n(x - x')] dx' &= \frac{1}{(-1)^r r!} \int_{\mathbb{R}} \frac{\partial^r}{\partial x^r} [(x')^r] (-1)^r M_n(x - x') dx' \\ &= \frac{1}{(-1)^r r!} \int_{\mathbb{R}} r! (-1)^r M_n(x - x') dx' = 1. \end{aligned} \quad (3.30)$$

So functions of the form $\{x \frac{\partial M_n}{\partial x}, x^2 \frac{\partial^2 M_n}{\partial x^2}, \dots, x^n \frac{\partial^n M_n}{\partial x^n}\}$ can also be used (the factor $\frac{1}{(-1)^p p!}$ will be absorbed into a_p). The analysis from (3.30), shows that products with polynomials can be used so long as the power of the polynomial is equal to the order of the derivative. This is because the integration by parts moves the derivatives to the polynomial so they can all be eliminated as long as there are enough derivatives to start with. However, as discussed above, the derivatives of M_n also have the appropriate order of zeros to be basis functions. Therefore, it is sufficient that the power of the polynomial is less than or equal to the order of the derivative. So the general form of interpolants of this form is

$$W_q(x) = \sum_{r=0}^m \sum_{d=0}^{n-r} a_{r,d} x^r \frac{\partial^d}{\partial x^d} M_n, \quad (3.31)$$

where $n = q + m$, q is the desired order of accuracy, m is the desired level of smoothness, and $a_{r,d}$ are weights to be determined with lexicographic order on the index pair (r, d) . This interpolant only takes up to powers of m because in Fourier space, products with polynomials

become derivatives on the base B -spline, and the zeros at integer multiples of 2π are lowered by one order for each derivative. Once again, the strategy is to represent the ordered basis functions in Fourier space, assemble the Taylor series coefficients column-wise in a matrix, and then solving for the weights by imposing the zero order conditions up to degree q . The Taylor series for the basis functions in Fourier space can be easily obtained by applying the derivatives and polynomial multiplications on the Taylor series of $\tilde{M}_{q+m}$. Let the linear system be written as $\mathbf{B}\mathbf{a} = \mathbf{e}_0$, since this system is underspecified, select the solution with minimal norm which is given by $\mathbf{a} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{e}_0$. Unlike MOMS, which is a special case of (3.31), there is no guarantee that this system has a solution. However, this procedure can be used to generate the smoothest interpolant given a desired order and support size by iteratively creating and solving this system with successively lower d_{max} until the system no longer has any solution. In general, the higher smoothness interpolants derived in this fashion are nonzero at the knots and less than one at $x = 0$. An example Mathematica script implementing this algorithm is given in Appendix B.

There is an equivalent linear system to solve for the weights based on the continuous moment conditions in real space instead of Taylor series and zero order conditions in Fourier space. Let $\{B_i\}$ be the set of basis functions. If there are M basis functions and the desired order of accuracy is q then the weights $\{a_i\}$ are given by the solution of

$$\begin{bmatrix} \int_{\mathbb{R}} B_0 dx & \int_{\mathbb{R}} B_1 dx & \int_{\mathbb{R}} B_2 dx & \int_{\mathbb{R}} B_3 dx & \cdots & \int_{\mathbb{R}} B_M dx \\ \int_{\mathbb{R}} x B_0 dx & \int_{\mathbb{R}} x B_1 dx & \int_{\mathbb{R}} x B_2 dx & \int_{\mathbb{R}} x B_3 dx & \cdots & \int_{\mathbb{R}} x B_M dx \\ \int_{\mathbb{R}} x^2 B_0 dx & \int_{\mathbb{R}} x^2 B_1 dx & \int_{\mathbb{R}} x^2 B_2 dx & \int_{\mathbb{R}} x^2 B_3 dx & \cdots & \int_{\mathbb{R}} x^2 B_M dx \\ & & \vdots & & & \\ \int_{\mathbb{R}} x^q B_0 dx & \int_{\mathbb{R}} x^q B_1 dx & \int_{\mathbb{R}} x^q B_2 dx & \int_{\mathbb{R}} x^q B_3 dx & \cdots & \int_{\mathbb{R}} x^q B_M dx \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (3.32)$$

Similar to the Fourier space system, this can be solved for the solution with minimal norm and there is no guarantee that a solution exists.

3.4 Timestepping: Lawson's method and Closed Newton-Cotes Quadrature

When $\rho, \mathbf{J}$ also evolve based on the fields $\mathbf{E}, \mathbf{B}$, in general, there is no explicit update formula to step from t to $t + \Delta t$. But as shown in Sec 2.2.2, the propagator can be used to remove the time scale associated with the electromagnetic waves and the system can be written as

$$\begin{aligned} \frac{\partial \hat{\mathbf{u}}}{\partial t} &= \mathcal{P}_{ext}^{-ct}[\mathbf{f}(\mathbf{x}, t, \mathcal{P}_{ext}^{ct}[\hat{\mathbf{u}}])], \\ \hat{\mathbf{u}}(\mathbf{x}, 0) &= \mathbf{u}(\mathbf{x}, 0), \end{aligned}$$

where $\hat{\mathbf{u}} = \mathcal{P}_{ext}^{-ct}[\mathbf{u}]$. Applying a Runge-Kutta (RK) scheme for this problem leads to Lawson's method, or generalized Runge-Kutta process [60]. Let the subscript h denote the discretized version of the quantities on the rectangular grid with spacing h and $\mathbf{u}_h^n \approx \mathbf{u}_h(n\Delta t)$. Then, for example, using standard RK4 to evolve from $\mathbf{u}_h^n$ to $\mathbf{u}_h^{n+1}$

$$\begin{aligned} \mathbf{u}^{(0)} &= \mathbf{u}_h^n \\ \mathbf{u}^{(1)} &= \mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{u}^{(0)}] + \frac{\Delta t}{2} \mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{f}_h^n(\mathbf{u}^{(0)})] \\ \mathbf{u}^{(2)} &= \mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{u}^{(0)}] + \frac{\Delta t}{2} [\mathbf{f}_h^{n+\frac{1}{2}}(\mathbf{u}^{(1)})] \\ \mathbf{u}^{(3)} &= \mathcal{P}_{ext}^{\Delta t}[\mathbf{u}^{(0)}] + \Delta t \mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}(\mathbf{u}^{(2)})] \\ \mathbf{u}_h^{n+1} &= \mathcal{P}_{ext}^{\Delta t}[\mathbf{u}^{(0)}] + \frac{\Delta t}{6} \left(\mathcal{P}_{ext}^{\Delta t}[\mathbf{f}_h^n(\mathbf{u}^{(0)})] + 2\mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}(\mathbf{u}^{(1)})] + 2\mathcal{P}_{ext}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}(\mathbf{u}^{(2)})] + \mathbf{f}_h^{n+1}(\mathbf{u}^{(3)}) \right) \end{aligned}$$

When $\rho, \mathbf{J}$ are prescribed functions independent of $\mathbf{E}, \mathbf{B}$, there is the update formula given by Duhamel's principle. For the time integral terms in (2.65)-(2.68), there is a great deal of freedom in choosing the quadrature scheme to approximate these integrals. LDCM uses closed Newton-Cotes quadrature schemes, which is quadrature with equidistant quadrature points including the end points. The purpose of choosing this type of scheme instead of Gaussian quadrature which would be more accurate for the same number of quadrature points is to take advantage of the linearity and semigroup property of the propagator and construct only one propagator. Dropping the c superscript in the propagator

$$\mathbf{u}_h^{n+1} = \mathcal{P}^{\Delta t}[\mathbf{u}_h^n] + \int_t^{t+\Delta t} \mathcal{P}^{t+\Delta t-s}[\mathbf{f}_h] ds \quad (3.33)$$

$$\approx \mathcal{P}^{\Delta t}[\mathbf{u}_h^n] + \sum_{m=0}^M w_m \mathcal{P}^{\Delta t-s_i}[\mathbf{f}_h(s_m)], \quad (3.34)$$

where w_i are quadrature weights and s_i are the $M+1$ quadrature points. For closed Newton-Cotes quadrature, the weights are found by integrating a Lagrange interpolating polynomial passing through the quadrature points [92, 95]

$$w_m = \frac{(-1)^{M-m}}{m!(M-m)!} \int_0^M t(t-1)\cdots(t-m+1)(t-m-1)\cdots(t-M)dt. \quad (3.35)$$

For quadrature points with arbitrary spacing, one would need to construct a $M+2$ propagators. However, for equidistant quadrature points with spacing $\Delta s = \Delta t/M$ only one propagator with Δs is needed

$$\int_t^{t+\Delta t} \mathcal{P}^{t+\Delta t-s}(\mathbf{f}_h) ds \approx \sum_{m=0}^M w_m \mathcal{P}^{\Delta t-s_m}(\mathbf{f}_h(s_m)) = \sum_{m=0}^M w_m \mathcal{P}^{(M-m)\Delta s} \mathbf{f}_h(s_m). \quad (3.36)$$

Therefore with $s_m = t + m\Delta s$

$$\mathbf{u}_h(t + \Delta t) \approx \mathcal{P}^{\Delta s}[\dots \mathcal{P}^{\Delta s}[\mathcal{P}^{\Delta s}[\mathbf{u}_h^n + w_0 \mathbf{f}_h(s_0)] + w_1 \mathbf{f}_h(s_1)] \dots w_{M-1} \mathbf{f}_h(s_{M-1})] + w_M \mathbf{f}_h(s_M), \quad (3.37)$$

because for any $\mathbf{g}$ at $s_M = t + \Delta t$, $\mathcal{P}^{t+\Delta t-s_M}[\mathbf{g}] = \mathcal{P}^0[\mathbf{g}] = \mathbf{g}$. This memory saving from using equidistant quadrature points also translates directly into choosing time integrators with equidistant stages when using Lawson's method. Closed Newton-Cotes quadrature applied to Duhamel's Formula can be recovered from Lawson's method. For example, if $\mathbf{f}$ is independent of $\mathbf{u}$, using RK4

$$\begin{aligned}\mathbf{u}_h^{n+1} &= \mathcal{P}^{\Delta t}[\mathbf{u}_h^n] + \frac{\Delta t}{6} \left(\mathcal{P}^{\Delta t}[\mathbf{f}_h^n] + 2\mathcal{P}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}] + 2\mathcal{P}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}] + \mathbf{f}_h^{n+1} \right) \\ &= \mathcal{P}^{\Delta t}[\mathbf{u}_h^n] + \frac{\Delta t}{6} \left(\mathcal{P}^{\Delta t}[\mathbf{f}_h^n] + 4\mathcal{P}^{\frac{\Delta t}{2}}[\mathbf{f}_h^{n+\frac{1}{2}}] + \mathbf{f}_h^{n+1} \right),\end{aligned}$$

which is Simpson's Rule, a 4th order closed Newton-Cotes quadrature scheme.

3.5 Spatial Error Analysis

The phase and amplitude error of a numerical method describes the error induced by the discretization on the spectrum of the eigenmodes of the solution operator. Let $L = \nabla^2$, the Fourier transform of the exact propagator (2.46) as used in (2.63) is

$$\tilde{\mathcal{P}}^{ct} = \begin{pmatrix} \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} & 0 & 0 \\ 0 & \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} & 0 \\ 0 & 0 & \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} \\ \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} & 0 & 0 \\ 0 & \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} & 0 \\ 0 & 0 & \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} \end{pmatrix}. \quad (3.38)$$

The eigenvalues for this operator are $\lambda_{\pm} = \tilde{H}^{ct} \pm \tilde{G}^{ct}\sqrt{\tilde{L}}$ each with multiplicity 3. In the exact continuous case (2.37)-(2.38)

$$\begin{aligned}\lambda_{\pm} &= \tilde{H}^{ct} \pm \tilde{G}^{ct}\sqrt{\tilde{L}} \\ &= \cos(c|\mathbf{k}|t) \pm \frac{\sin(c|\mathbf{k}|t)}{|\mathbf{k}|} \sqrt{-|\mathbf{k}|^2} \\ &= \cos(c|\mathbf{k}|t) \pm i \sin(c|\mathbf{k}|t) \\ &= e^{\pm ic|\mathbf{k}|t}\end{aligned}$$

which represents waves propagating forwards or backwards. In the discrete case, these operators (3.17)-(3.18) are stencil operators

$$C = \sum_{j=-m/2}^{m/2} w_j S_j, \quad (3.39)$$

Figure 3.1: Phase error for the propagator with three different orders of accuracy discretizations. This is a 2D slice where $k_y = 0$ with $\theta = \tan^{-1} \left(\frac{k_z}{k_x} \right) \in [0, \frac{\pi}{2}]$ and $h|\mathbf{k}| \in [0, \pi]$.

where m is the support of the stencil operator and S_j is the shift operator $S_j(x_k) = x_{j+k}$. Its Fourier transform is given by

$$\tilde{C} = \sum_{j=-m/2}^{m/2} w_j e^{ijh}. \quad (3.40)$$

So $\lambda_{\pm}$ is a product of weighted sums of complex exponentials. Let $c = h = t = 1$, the phase error of the method is then $\text{Arg}(\lambda_{\pm})/|\mathbf{k}| - 1$ and the amplitude error is $|\lambda_{\pm}| - 1$. Fig. 3.1 shows the phase error for three different discretizations of the propagator. As shown in the figure, as the propagator is higher and higher order, the region where the error is large becomes smaller and smaller and shifts more towards the high wave number modes. Fig. 3.2 shows the amplitude error for the same three discretizations of the propagator. The error has the same qualitative behavior as the phase error, but the magnitude of the error is less than one therefore the method is damping the high wave number modes. The region of damping also coincides with the region of high phase error, so the method is inducing phase error at high wave number modes but it is also damping these errors. Fig. 3.3 shows the amplitude decay of a plane wave. The higher order propagator shows slower decay as expected.

3.6 Efficient Application of the Propagator

The choice of remaining in physical space and using discrete convolution kernels with compact support gives two ways to efficiently apply the propagator in practice. Discrete convolutions with compact support kernels can be evaluated using Hockney's algorithm. Since the propagator has support that can be calculated, it is possible to parallelize in space by domain decomposition.

Figure 3.2: Amplitude error for the propagator with three different orders of accuracy discretizations. This is a 2D slice where $k_y = 0$ with $\theta = \tan^{-1} \left(\frac{k_z}{k_x} \right) \in [0, \frac{\pi}{2}]$ and $h|\mathbf{k}| \in [0, \pi]$.

3.6.1 Hockney's Algorithm

The naive approach for computing discrete convolutions is to loop over the input data and compute the weighted sum centered at the current point. If the size of the stencil is s and the number of input points is N then the computational cost for this method scales as $\mathcal{O}(sN)$. However, in multiple dimensions, the data access pattern for this method will lead to a large number of cache misses. A more cache friendly approach is to loop over the weights, compute the entire input data multiplied by the current weight, and add the result to the output array with appropriate shifting. The amount of computation is identical to the naive approach but since this method is operating on contiguous blocks of memory, it can be significantly faster. However, an even better approach is using Hockney's algorithm [51], based on Fast Fourier Transforms (FFT). Hockney's algorithm computes the convolution by embedding the kernel in an array the same size as the input, extending the size of both arrays by the support of the kernel, compute their FFTs, element-wise multiply the two Fourier transforms, and finally inverse transform the result. This method thus scales as $\mathcal{O}((s + N) \log(s + N) + N)$ which is smaller than $\mathcal{O}(sN)$ for large s, N . Furthermore, the element-wise multiplication is cache friendly. Hockney's algorithm is as follows, given the following discrete convolution

$$f(j) = \sum_{\ell=-\infty}^{\infty} G(j - \ell)\rho(\ell), \quad (3.41)$$

where $j = 0, \dots, N - 1$ is the domain of interest and $G(r)$ is nonzero only in the domain $0 \leq r \leq L - 1$. It is not necessary for L and N to be related but for the propagator $L < N$. Consider the range of ℓ for $G(j - \ell)$ is nonzero, since everything outside that range would

Figure 3.3: Amplitude for a plane wave in xy , wavelength $\lambda = 4h$ in 2D with $c = 1$, $\Delta t = h = \frac{1}{64}$ and initial amplitude $|E_y|_\infty = 1$. The amplitude decays as expected from the damping in the numerical scheme.

not contribute to the infinite sum.

$$0 \leq j - \ell \leq L - 1, \quad (3.42)$$

$$0 \geq \ell - j \geq 1 - L, \quad (3.43)$$

$$j \geq \ell \geq 1 - L + j. \quad (3.44)$$

To get bounds for ℓ , substitute for the upper bound with $\max(j) = N - 1$ and the lower bound with $\min(j) = 0$ which would give $\ell \in [-(L - 1), N - 1]$ and therefore

$$f(j) = \sum_{\ell=-\infty}^{\infty} G(j - \ell)\rho(\ell) = \sum_{\ell=-(L-1)}^{N-1} G(j - \ell)\rho(\ell). \quad (3.45)$$

Now, using Fourier transforms to compute this finite sum. In physical space it is equivalent to computing the following

$$\sum_{\ell=-(L-1)}^{N-1} \tilde{G}(j - \ell)\tilde{\rho}(\ell), \quad (3.46)$$

where for $r \in [-(L-1), N-1]$,

$$\tilde{G}(r) = G(r), \quad (3.47)$$

$$\tilde{\rho}(r) = \rho(r), \quad (3.48)$$

and $\tilde{G}(r), \tilde{\rho}(r)$ are periodic outside this domain. Consider any fix $j \in [0, N-1]$, the domain of ℓ that is relevant for this j is as shown before $\ell \in [1-L+j, j]$. But $[1-L+j, j] \subseteq [1-L, N-1]$, $\forall j \in [0, N-1]$. In the domain $[1-L, N-1]$, $\tilde{G} = G, \tilde{\rho} = \rho$ by construction and therefore $\forall j \in [0, N-1]$, (3.46) is equal to (3.45).

3.6.2 Quad Precision Arithmetic for High Order Kernels

As long as the choice of time integrator or quadrature scheme remains fixed, with Hockney's Algorithm, it is sufficient to construct the convolution kernels as described in Sec. 3.2, and then taking and storing their Fourier transforms once at the beginning of the code. For high order convolution kernels, it is necessary to use higher order terms in the Taylor expansion of the B -spline which has small coefficients. This leads to the overall regularizer to be a high order piecewise polynomials where the high order terms are multiplied by small coefficients (see Appendix C). When finally evaluating these polynomials in the spherical quadrature scheme, double precision arithmetic has been found insufficient to compute these products. This leads to an overall degradation in the accuracy of the method, as extreme as lowering the accuracy from an expected $\mathcal{O}(h^{15})$ to $\mathcal{O}(h)$. Empirically, it has been sufficient, up to 15th order, to compute the B -splines, the spherical quadratures (3.17)-(3.18), and Fourier transform the kernels in quad precision but then storing the result in double precision and running the rest of the code in double precision.

3.6.3 Domain Decomposition

If the computational domain is large, one method to parallelize the computation is to divide the domain up into patches and assign each patch to a computational unit. Suppose the desired order of accuracy for the numerical solution is q . Centered-difference stencils have support $[-q/2, q/2]$ while the convolution kernels using a minimal support regularizer also has support $[-q/2, q/2]$. Therefore, each grid point is affected only by data at most q grid points away. Therefore, for a large computational domain, it is possible to decompose the domain into overlapping patches. The patches are disjoint except for a layer of q ghost nodes which can be discarded after applying the propagator. Once the rectangular grid is broken up into these overlapping patches and data from neighboring patches have been copied into the ghost nodes, each patch can be updated independently.

3.6.4 Cost Comparisons

In order to quantify the computational benefits of LDCM, suppose the fields are evolved in a unit cube in $\mathbb{R}^3$ using domain decomposition such that each patch has N^3 nodes with

spacing h to some time T where $cT/h > 1$. Also let the size of the finite difference stencil be p . For LDCM, the number of ghost nodes required in one direction will be $\sigma = \lceil cT/h \rceil$. For an explicit finite difference (FD) method, if $\Delta t = h/c$, then the number of time steps required will be $N_t = \sigma$. In LDCM, the dominant floating point operations (FLOPs) cost is in computing the Fourier transforms for Hockney's algorithm. Using fast fourier transforms, the cost of transforming a vector in $\mathbb{R}^m$ is $\mathcal{O}(m \log m)$ and thus the FLOPs cost for the method scales as $\mathcal{O}((N + p + \sigma)^3 \log(N + p + \sigma))$. For a FD method, the dominant FLOPs cost is doing pointwise operations for each node at each time step and thus the cost scales as $\mathcal{O}(\sigma(N + p)^3)$. Taking the ratio of the LDCM to FD FLOPs costs, the leading order term shows a reduction in FLOPs cost per patch: $\mathcal{O}\left(\log(N + p + \sigma) \left(\frac{1}{N+p} + \frac{1}{\sigma}\right)\right)$.

The other cost to consider is the data movement cost (*i.e.* communication cost). The amount of data one patch needs to send to its neighbors, *i.e.* the number of ghost nodes per patch, is $\mathcal{O}((N + \sigma + p)^3 - N^3)$. For a FD method, one also needs to send ghost node data to its neighbors but communication needs to occur at every time step and so the total amount of data sent per patch is $\mathcal{O}(\sigma[(N + p)^3 - N^3])$. Taking the ratio of the LDCM to FD communication costs, the leading order term shows a reduction in communication costs per patch: $\mathcal{O}\left(\frac{1}{p} + \frac{1}{\sigma}\right)$.

3.7 Constraint Enforcement

Even though the divergence constraints are preserved by the continuous time evolution, deviations from (2.11)-(2.14) may be generated by discretization error. One method to help remedy this is to apply local filters [67] of the form

$$\mathbf{E} := \mathbf{E} + \eta(\mathcal{R}\mathbf{E} - 4\pi\nabla\rho), \quad (3.49)$$

$$\mathbf{B} := \mathbf{B} + \eta\mathcal{R}\mathbf{B}, \quad (3.50)$$

$$\mathcal{R}_{ij} = \partial_{x_i}\partial_{x_j}, \quad (3.51)$$

where $\eta \sim \mathcal{O}(h^2)$ is a constant and $\mathcal{R}$ is a matrix valued operator with the diagonal terms discretized with centered-difference approximations to the second derivative while the off-diagonal terms are products of centered-difference approximations to the first derivatives. This filtering step corresponds to applying an explicit diffusion step to the error in the longitudinal fields. For the curl constraints, re-initialized Φ, Ψ at the beginning of each time step. For a sufficiently large ghost node region, this allows for parallelization across patches and therefore it is not necessary to allocate memory for the entirety of the auxiliary variables; it is only necessary to create them at a per-patch basis.

With the addition of this filter, in the absence of sources, the $\mathbf{E}, \Phi$ updates in Fourier

space are

$$\begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \\ \tilde{\Phi}_x \\ \tilde{\Phi}_y \\ \tilde{\Phi}_z \end{pmatrix} = \underbrace{\begin{pmatrix} 1 + \eta\tilde{\partial}_x^2 & \eta\tilde{\partial}_x\tilde{\partial}_y & \eta\tilde{\partial}_x\tilde{\partial}_z & 0 & 0 & 0 \\ \eta\tilde{\partial}_x\tilde{\partial}_y & 1 + \eta\tilde{\partial}_y^2 & \eta\tilde{\partial}_y\tilde{\partial}_z & 0 & 0 & 0 \\ \eta\tilde{\partial}_x\tilde{\partial}_z & \eta\tilde{\partial}_y\tilde{\partial}_z & 1 + \eta\tilde{\partial}_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\tilde{\mathcal{A}} := \mathbb{I} + \eta\tilde{\mathcal{R}}} \underbrace{\begin{pmatrix} \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} & 0 & 0 \\ 0 & \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} & 0 \\ 0 & 0 & \tilde{H}^{ct} & 0 & 0 & \tilde{G}^{ct} \\ \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} & 0 & 0 \\ 0 & \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} & 0 \\ 0 & 0 & \tilde{G}^{ct}\tilde{L} & 0 & 0 & \tilde{H}^{ct} \end{pmatrix}}_{\tilde{\mathcal{P}}^{ct}} \begin{pmatrix} \tilde{E}_{x,0} \\ \tilde{E}_{y,0} \\ \tilde{E}_{z,0} \\ \tilde{\Phi}_{x,0} \\ \tilde{\Phi}_{y,0} \\ \tilde{\Phi}_{z,0} \end{pmatrix}. \quad (3.52)$$

The condition for stability in ℓ_2 is

$$\|\tilde{\mathcal{A}}\tilde{\mathcal{P}}^{ct}\mathbf{u}\|_2 \leq \|\mathbf{u}\|_2, \quad \forall \mathbf{u} \in \mathbb{R}^6. \quad (3.53)$$

The quantity on the left can be bounded by the individual norms

$$\|\tilde{\mathcal{A}}\tilde{\mathcal{P}}^{ct}\mathbf{u}\|_2 \leq \|\tilde{\mathcal{A}}\|_2 \|\tilde{\mathcal{P}}^{ct}\|_2 \|\mathbf{u}\|_2$$

by definition of ℓ_2 matrix norm

$$= \sigma_{\max}(\tilde{\mathcal{A}}) \sigma_{\max}(\tilde{\mathcal{P}}^{ct}) \|\mathbf{u}\|_2$$

the singular values are absolute values of eigenvalues

$$= \lambda_0(\tilde{\mathcal{A}}) \lambda_0(\tilde{\mathcal{P}}^{ct}) \|\mathbf{u}\|_2,$$

where the $\lambda_0(X) = \max_j |\lambda_j(X)|$. So it suffices that the eigenvalues for the two operators are individually bounded to be below one for all $\mathbf{k}$. Sec. 3.5 discusses the eigenvalues of $\tilde{\mathcal{P}}^{ct}$ and showed that they are bounded. The eigenvalues of $\tilde{\mathcal{A}}$ can all be represented by $1 + \eta\lambda_j(\tilde{\mathcal{R}})$ for $j = 1, \dots, 6$. Three of the eigenvalues of $\tilde{\mathcal{R}}$ are zero owing to the bottom right 3×3 identity block in $\tilde{\mathcal{A}}$. For the other three eigenvalues, first consider the top left 2×2 block in $\tilde{\mathcal{R}}$ in and let D_i, S_i be the symbol (and hence they are functions of $\mathbf{k}$) of the discretization to the first derivative and second derivative, respectively, in i

$$\tilde{\mathcal{R}}_{2D} = \begin{pmatrix} S_x & D_x D_y \\ D_x D_y & S_y \end{pmatrix}. \quad (3.54)$$

The eigenvalues of $\tilde{\mathcal{R}}_{2D}$ are

$$\lambda_{\pm} = \frac{S_x + S_y \pm \sqrt{(S_x - S_y)^2 + 4D_x^2 D_y^2}}{2}. \quad (3.55)$$

The condition for these eigenvalues to be bounded between $S_x + S_y$ and zero is then

$$\begin{aligned} \sqrt{(S_x - S_y)^2 + 4D_x^2 D_y^2} &\leq S_x + S_y, \\ S_x^2 - 2S_x S_y + S_y^2 + 4D_x^2 D_y^2 &\leq S_x^2 + 2S_x S_y + S_y^2, \\ D_x^2 D_y^2 &\leq S_x S_y. \end{aligned} \quad (3.56)$$

Fig. 3.4 shows the ratio of $\frac{D^2}{S}$ using the 6-, 8-, 16-th order centered-difference approximations for the first and second derivatives. As the figure shows, the ratio is less than or equal to one for all wave modes representable on a grid with spacing h thus the condition for the eigenvalues of $\tilde{\mathcal{R}}$ to be bounded are satisfied with these choices.

For the 3D case

$$\tilde{\mathcal{R}}_{3D} = \begin{pmatrix} S_x & D_x D_y & D_x D_z \\ D_x D_y & S_y & D_y D_z \\ D_x D_z & D_y D_z & S_z \end{pmatrix}. \quad (3.57)$$

Let D_i, S_i be restricted to centered-difference approximations, $|D_i^2| \leq |S_i|$ and D_i^2, S_i are real nonpositive functions, so $S_i = D_i^2 + \epsilon_i$ for some real nonpositive function ϵ_i . Then

$$\tilde{\mathcal{R}}_{3D} = \begin{pmatrix} D_x^2 & D_x D_y & D_x D_z \\ D_x D_y & D_y^2 & D_y D_z \\ D_x D_z & D_y D_z & D_z^2 \end{pmatrix} + \begin{pmatrix} \epsilon_x & 0 & 0 \\ 0 & \epsilon_y & 0 \\ 0 & 0 & \epsilon_z \end{pmatrix}. \quad (3.58)$$

The matrix on the left has two eigenvalues: zero with multiplicity two and $D_x^2 + D_y^2 + D_z^2 \leq 0$. The eigenvalues for the matrix on the right are $\epsilon_x, \epsilon_y, \epsilon_z$. Furthermore, the two matrices are Hermitian. Therefore, the eigenvalues of $\tilde{\mathcal{R}}_{3D}$ can now be bounded using the (dual) Weyl inequalities on the maximum and minimum eigenvalues [93, 1] or the monotonicity principles [56] which state that for two Hermitian matrices A, B

$$\lambda_{\max}(A + B) \leq \lambda_{\max}(A) + \lambda_{\max}(B), \quad (3.59)$$

$$\lambda_{\min}(A + B) \geq \lambda_{\min}(A) + \lambda_{\min}(B). \quad (3.60)$$

Applying these inequalities to $\tilde{\mathcal{R}}_{3D}$

$$\max_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) \leq 0 + \max_{\mathbf{k}} \{\epsilon_x, \epsilon_y, \epsilon_z\},$$

but $\epsilon_i \leq 0$ because they are nonpositive, so

$$\max_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) \leq 0.$$

Figure 3.4: The ratios $\frac{D^2}{S}$ for three different order of approximations.

For the minimum eigenvalue

$$\begin{aligned} \min_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) &\geq D_x^2 + D_y^2 + D_z^2 + \min_{\mathbf{k}} \{\epsilon_x, \epsilon_y, \epsilon_z\}, \\ \min_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) &\geq D_x^2 + D_y^2 + D_z^2 + \epsilon_x + \epsilon_y + \epsilon_z, \\ 0 &\geq \max_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) \geq \min_{\mathbf{k}} \lambda(\tilde{\mathcal{R}}_{3D}) \geq S_x + S_y + S_z. \end{aligned}$$

This leads to the bound

$$\lambda_0(\tilde{\mathcal{A}}) \leq |1 + \eta(S_x + S_y + S_z)|. \quad (3.61)$$

Since S_i is a negative function of k_i , stability requires that

$$\begin{aligned} 0 \leq \eta &\leq \frac{1}{\max_{k_x} |S_x| + \max_{k_y} |S_y| + \max_{k_z} |S_z|}, \\ 0 \leq \eta &\leq \frac{1}{3 \max_{k_x} |S_x|}. \end{aligned} \quad (3.62)$$

Table 3.1 shows the maximum value of η for several discretizations to the derivatives.

3.8 Single Level Algorithm

Algorithm 3.1 outlines the steps for the single level LDCM.

With domain decomposition, consider the computational domain, Ω , partitioned into subdomains patches, Algorithm 3.1 is modified as follows, for all patches:

Table 3.1: Values for the filter parameter, η , for 2-, 6-, 8-, 16-th order accurate centered-difference approximations for the first and second derivatives.

Order	η
2	$\frac{1}{12}h^2$
6	$\frac{15}{272}h^2$
8	$\frac{105}{2048}h^2$
16	$\frac{1576575}{35127296}h^2$

1. at the beginning of each quadrature step, copy field values in ghost region from neighboring processors,
2. apply propagator to update local field values, invalidating values in ghost region.

The minimum width of the ghost region is $c\Delta s + p$, where p is the half width of the regularized delta plus half of the order of the method (from the centered-difference stencil).

3.8.1 Extension to 2D

For 2D problems, where none of the fields and sources depend on z but the z component of the vector quantities still exist, 2D discrete convolution kernels can be generated by constructing G_h^t, H_h^t and taking a sum over z . These kernels are still local with the same support, whose support is the entire disk, rather than a thickened sphere.

3.9 Relationship with Other Propagator methods

The spherical means discretization and quadrature approach for the source terms described in this chapter are only one approach to discretizing (2.65)-(2.67). Shin (2018) uses a sampling and quadrature discretization for the spherical means integral [82]. And in PSATD, the source term integration is done by assuming a form for the source terms and then computing the integrals exactly in Fourier space [89].

3.9.1 Spherical Means in Shin (2018)

While LDCM computes the propagator using discrete convolutions with regularized delta distributions, an alternative approach is to revert back to the spherical means integrals and approximate them with a quadrature scheme [82]. Suppose $ct = mh$ for $m \in \mathbb{Z}^+$, then the sphere of radius ct centered at a grid index (i, j, k) contains the six points in the set

Algorithm 3.1 Single Level LDCM for Maxwell's Equations

```

Initialize  $\mathbf{E}_h^{(0)}$  and  $\mathbf{B}_h^{(0)}$ 
Initialize Newton-Cotes quadrature weights  $\{w_m\}_{m=0}^M$ 
Compute  $G_h^{\Delta s}$ , and  $H_h^{\Delta s}$  based on step-size in time and space
/* Begin time-stepping loop */
for  $n = 1, 2, \dots$  do
    /* Initialize the fields for this time step */
     $\mathbf{E}_h^{(n)} \leftarrow \mathbf{E}_h^{(n-1)}$ ,  $\mathbf{B}_h^{(n)} \leftarrow \mathbf{B}_h^{(n-1)}$ 
    /* Compute Auxiliary Variables */
     $\Psi_h^{(n)} \leftarrow \nabla \times \mathbf{E}_h^{(n)}$ ,  $\Phi_h^{(n)} \leftarrow \nabla \times \mathbf{B}_h^{(n)}$ 
    /* Newton-Cotes quadrature */
    for  $m = 0, 1, \dots, M-1$  do
        /* Add source term for node  $t_{n,m} = (n-1)\Delta t + m\Delta s$  */
         $\mathbf{E}_h^{(n)} \leftarrow \mathbf{E}_h^{(n)} - w_m 4\pi \mathbf{J}_h(t_{n,m})$ 
         $\Phi_h^{(n)} \leftarrow \Phi_h^{(n)} - w_m 4\pi c \nabla \rho_h(t_{n,m})$ 
         $\Psi_h^{(n)} \leftarrow \Psi_h^{(n)} - w_m 4\pi \nabla \times \mathbf{J}_h(t_{n,m})$ 
        /* Update the fields */
        
$$\begin{bmatrix} \mathbf{E}_h^{(n)} \\ \Phi_h^{(n)} \end{bmatrix} \leftarrow \begin{bmatrix} H_h^{\Delta s} * \mathbf{E}_h^{(n)} + G_h^{\Delta s} * \Phi_h^{(n)} \\ (G_h^{\Delta s} * \nabla^2) * \mathbf{E}_h^{(n)} + H_h^{\Delta s} * \Phi_h^{(n)} \end{bmatrix}$$

        
$$\begin{bmatrix} \mathbf{B}_h^{(n)} \\ \Psi_h^{(n)} \end{bmatrix} \leftarrow \begin{bmatrix} H_h^{\Delta s} * \mathbf{B}_h^{(n)} - G_h^{\Delta s} * \Psi_h^{(n)} \\ -(G_h^{\Delta s} * \nabla^2) * \mathbf{B}_h^{(n)} + H_h^{\Delta s} * \Psi_h^{(n)} \end{bmatrix}$$

    end for
    /* Final Newton-Cotes point */
     $\mathbf{E}_h^{(n)} \leftarrow \mathbf{E}_h^{(n)} - w_M 4\pi \mathbf{J}_h(t_{n,M})$ 
    /* Apply Filter */
     $\mathbf{E}_h^{(n)} \leftarrow \mathbf{E}_h^{(n)} + \eta(\mathcal{L}\mathbf{E}_h^{(n)} - 4\pi \nabla_h \rho_h^{(n+1)})$ 
     $\mathbf{B}_h^{(n)} \leftarrow \mathbf{B}_h^{(n)} + \eta(\mathcal{L}\mathbf{B}_h^{(n)})$ 
end for

```

$Z = \{(i+m, j, k), (i-m, j, k), (i, j+m, k), (i, j-m, k), (i, j, k+m), (i, j, k-m)\}$. Then

$$\frac{1}{4\pi c^2 t^2} \int_{\partial B_{ct}(\mathbf{x})} f(\mathbf{z}) dS(\mathbf{z}) \approx \sum_{z \in Z} \frac{1}{6} f(zh) + \mathcal{O}(h^4). \quad (3.63)$$

While this scheme also has compact support and it is easy to implement, it is restrictive in that the sphere must be a certain size and that it is not obvious how to extend this quadrature to higher orders of accuracy.

3.9.2 Spherical Means for Finite Element Meshes

An abstract view on the discretization of the spherical integral is that

$$\int_{\partial B_r(\mathbf{x})} f(\mathbf{z}) dS(\mathbf{z}) \approx \mathcal{Q}\mathcal{A}f, \quad (3.64)$$

where $\mathcal{Q}$ is a quadrature operator and $\mathcal{A}$ is an operator that approximates f on the quadrature points. In LDCM, $\mathcal{Q}$ is Atkinson spherical quadrature and $\mathcal{A}$ interpolates grid values using high order B -splines, and the composition, $\mathcal{Q}\mathcal{A}$, is computed. For Shin (2018), $\mathcal{Q}$ is the six point quadrature and $\mathcal{A}$ is the grid sampling operator. However, suppose f is already approximated as a piecewise polynomial over the domain, then the quadrature can be applied directly on this approximation. This allows spherical means to be used for wave equation problems beyond rectangular grids such as when the domain is discretized with a finite element mesh.

3.9.3 Source Integration in PSATD

PSATD [44, 89] is another propagator method for Maxwell's equations. The method uses an exact representation of the Maxwell propagator in Fourier space. The method transforms the fields and sources into Fourier space, multiply by the propagator, and then transform back. The derivatives are replaced by high order approximations cooresponding to Yee centering of the field variables.

PSATD also presents another approach to handling the source term integration by assuming some simple form for $\rho, \mathbf{J}$ and computing the integral exactly. Computationally such assumptions can be beneficial when evaluating the source terms is expensive, such as when they are themselves governed by a set of evolution equations. Suppose $\mathbf{J}$ is constant over a time step and $\rho^{n+s} = \rho^n + (\rho^{n+1} - \rho^n)s/\Delta t$. Looking at only the source term integral for $\mathbf{E}$ update (2.65) in Fourier space

$$\begin{aligned} & \int_0^{\Delta t} \left(\cos(c|\mathbf{k}|(\Delta t - s)) \tilde{\mathbf{J}} + ic \frac{\sin(c|\mathbf{k}|(\Delta t - s))}{|\mathbf{k}|} \mathbf{k} \tilde{\rho} \right) ds \\ &= -\frac{\sin(c|\mathbf{k}|\Delta t)}{c|\mathbf{k}|} \tilde{\mathbf{J}}^n + i \frac{\mathbf{k}}{|\mathbf{k}|^2} \left[\left(\frac{\sin(c|\mathbf{k}|\Delta t)}{|\mathbf{k}|c\Delta t} - 1 \right) \tilde{\rho}^{n+1} + \left(\cos(c|\mathbf{k}|\Delta t) - \frac{\sin(c|\mathbf{k}|\Delta t)}{|\mathbf{k}|c\Delta t} \right) \tilde{\rho}^n \right]. \end{aligned} \quad (3.65)$$

This leads to Eq. (29) in [89]. The differences between PSATD and LDCM is that PSATD it is only local in the spectral limit, it requires a large number of ghost nodes to achieve this limit, while LDCM is a local algorithm and the number of ghost nodes required can be computed exactly, and also PSATD is not high order in time due to its source term coupling.

Chapter 4

AMR for LDCM

Adaptive Mesh Refinement (AMR) is a technique that allows more computational resources to be spent in regions of interest, such as near sources, shocks, or wavefronts, and focus less on areas in the computational domain that are less significant [13]. This is achieved by identifying these computational areas of interest and introduce additional grids that only cover these areas with a smaller grid spacing. This procedure can be applied iterative, *i.e.* putting finer grids on top of these fine grids. The problem is then solved on all the grids and then a syncing or correction procedure is applied to produce an overall valid solution over the entire computational domain. As the simulation progresses, these regions of interest will change over time as well and it will be necessary to regrid.

For LDCM, a nested hierarchy of rectangular grids with constant refinement ratio is used. The new grid is generated by refining the existing finest rectangular grid in the identified region. Fig. 4.1 illustrates this in 1D, in particular these grids are guaranteed to have overlapping grid points. If the refinement factor is $r \in \mathbb{Z}^+$ and the grid spacing at the coarsest level is h , then let Ω_l denote the union of rectangular grids with grid spacing $h_l := h/r^l$. In the current implementation of AMR LDCM, the refinement ratio is constant. So if there is a region where one might want to be included in Ω_l then there must be grids in $\Omega_{0,\dots,l-1}$ covering that region as well. While a constant refinement ratio is not strictly necessary, it makes the implementation much simpler. Since LDCM does not have any CFL restriction, the levels can be propagated in parallel with the same time step. This chapter describes the sampling and interpolation operators used to communicate the values between levels and regridding.

4.1 Basic AMR Operators

With the addition of the nested hierarchy of rectangular grids, the method now needs operators to communicate the field values between levels. For AMR LDCM, level l will only communicate directly with levels $l - 1$ and $l + 1$. This section gives a description of the sampling operator which is used to communicate values from fine level to coarse level and the interpolation operator which is used to communicate values from the coarse level to fine

Figure 4.1: Illustration of two levels in 1D with factor of four refinement. The black bolded points are the grid points that exist on both grids.

level.

4.1.1 Sampling

The sampling operator, $\mathcal{S}$, replaces the value at the coarse level using the value at the fine level on every node that the two levels intersect. When there are more than two levels, the sampling begins from the finest level.

4.1.2 Interpolation

Similar to the ghost regions for each patch in the single level algorithm for domain decomposition, a ghost region, $\Omega_{l,g}$, is defined for each level l . The width of this region is also defined in the same way. When there are more than two levels, the interpolating begins from the coarsest level. Fig. 4.2 illustrates the sampling and interpolating regions in 2D for two levels.

The choice of the interpolant requires special attention. As discussed in Sec. 3.2, $\delta_h \in C^0$ because it is chosen to have minimal support. The error in the spherical quadrature (3.14) depends on the smoothness of the integrand, but since discrete convolution and numerical quadrature commutes, this smoothness requirement can be applied on the fields. In contrast, the coarse-fine interpolation must be sufficiently smooth for the desired order of accuracy so that the solver will achieve the desired accuracy. This means that the interpolant used here must also be high order and have sufficient number of derivatives. Once again, high order B -splines are used, only with larger support.

4.2 Regridding

As the simulation progresses, the regions of interest might also change depending on the problem. This requires generating new grids and discarding some old ones. Suppose level l , $l > 0$, requires regridding, let $\Omega_{l,discard} \cup \Omega_{l,keep}$ be the level before regridding and $\Omega_{l,keep} \cup \Omega_{l,new}$ be the level after regridding. First sample values down from $\Omega_{l,discard}$, then interpolate onto $\Omega_{l,new}$ using the same sampling and interpolating operators as before. The regridding algorithm is outlined in detail in 4.1.

Figure 4.2: Illustration of two levels with factor of two refinement. The blue area is the computational region of interest for the fine grid and the red region is the ghost nodes for the fine level. The sampling operator is applied to the blue region while the interpolation operator is applied to the red region.

4.3 Open Boundary Conditions

The radiation of wave energy out to the far field is important for many wave problems and electromagnetism is no exception, but practically the computational domain needs to be truncated. There have been many techniques developed for this open/radiation/transparent boundary condition to simulate an infinite domain [47, 85]. There is the non-reflecting boundary conditions (NRBC), which imposes boundary conditions on the artificial domain boundary so there is no (or little) spurious reflections [38, 39]. Another popular method is the perfectly matched layer (PML), which surrounds the computational domain with an artificial absorption layer [12]. In this work, the approach taken is to use AMR to place the artificial boundary far away from the sources. This is done because the main concern is testing the viability and properties of the LDCM. Since the electromagnetic waves decay like $1/r$ away from the sources, the waves can be decayed to machine precision and cause no reflections. Although, in principle, PML can be readily implemented with LDCM.

The algorithm as described in Algorithm 3.1 does not take into any consideration of the boundary conditions of the problem. Analytically, the propagator assumes open or periodic boundary conditions. However, numerically the domain does not extend to infinity, and

Algorithm 4.1 Regridding Algorithm

```

for levels  $\Omega_l$ ,  $l = 1, \dots, L - 1$  do
  if regrid then
    /* Sample down starting from topmost level */
    for  $k = L - 1, \dots, j$  do
       $\mathbf{E}_{k-1}^{(n)} \leftarrow \mathcal{S}[\mathbf{E}_k^{(n)}]$  on  $\Omega_k \cap \Omega_{k,discard}$ 
       $\mathbf{B}_{k-1}^{(n)} \leftarrow \mathcal{S}[\mathbf{B}_k^{(n)}]$  on  $\Omega_k \cap \Omega_{k,discard}$ 
      /* Discard part of domain that has been sampled from */
       $\Omega_k \leftarrow \Omega_k \setminus (\Omega_k \cap \Omega_{l,discard})$ 
    end for
     $\Omega_j \leftarrow \Omega_j \cup \Omega_{l,new}$ 
    /* Interpolate from level  $l - 1$  */
     $\mathbf{E}_l^{(n)} \leftarrow \mathcal{I}[\mathbf{E}_{l-1}^{(n)}]$  on  $\Omega_{l,new}$ 
     $\mathbf{B}_l^{(n)} \leftarrow \mathcal{I}[\mathbf{B}_{l-1}^{(n)}]$  on  $\Omega_{l,new}$ 
  end if
end for

```

so for points near the computational domain boundary it is not possible to compute the spherical means. Fig. 4.3 illustrates this problem where part of the sphere lies outside the computational domain for grid points near the boundary. Using Hockney's Algorithm to compute the discrete integral amounts to inserting zeros for the part of the sphere that lies outside the computational domain. This creates a discontinuity in the fields across the domain boundary if the fields are nonzero near the boundary. This discontinuity in $\mathbf{E}$, in particular, generates an artificial current sheet at the domain boundary which in turn is a source for unphysical electromagnetic waves reflecting back into the computational domain. Another approach that has been tested is to use the current value at the grid point $\mathbf{x}$ instead of zero for the part of the sphere outside the computational domain. However, this approach also generates small magnitude electromagnetic waves reflecting back into the domain. For the test cases shown in Chapter 6, zero is used for the grid points outside the domain.

4.4 AMR LDCM Algorithm

Since (2.7)-(2.10) is a system of linear differential equations, the overall solution can be generated by linear superposition in the AMR setup; the solution is given by a composite where it takes the finest level values for any subdomain. For example, in the two level case, let $\mathbf{U} = (\mathbf{E}, \mathbf{B})^T$, then the solution is given by,

$$\mathbf{U} = \begin{cases} \mathbf{U}_1, & \text{on } \Omega_1 \\ \mathbf{U}_0, & \text{on } \Omega_0 \setminus \Omega_1 \end{cases} \quad (4.1)$$

Figure 4.3: A 2D illustration of the spherical means geometry near the computational domain boundary $\partial\Omega_0$. For the point $\mathbf{x}$, the solid part of the sphere represents points where there are values while the dotted part of the sphere lies outside the domain and thus there are no values.

Let $f_j^{(n)}$ denote discretized f on level j and at time $t_n = n\Delta t$, AMR LDCM is outlined in Algorithm 4.2. If the width of $\Omega_{l,g}$ is defined with the overall time step, Δt , then it is only necessary to interpolate once at the beginning of each time step. If it is defined with the quadrature step, Δs , then it is necessary to interpolate at the beginning of each quadrature step.

Algorithm 4.2 AMR LDCM for Maxwell's Equations

```

Initialize Newton-Cotes quadrature weights  $\{w_m\}_{m=0}^M$ 
for all levels  $\Omega_l$ ,  $l = 0, \dots, L - 1$  do
    Initialize  $\mathbf{E}_l^{(0)}$  and  $\mathbf{B}_l^{(0)}$ 
    Compute  $G_{h/r^l}^{\Delta s}$  and  $H_{h/r^l}^{\Delta s}$ 
end for
/* Begin time-stepping loop */
for  $n = 1, 2, \dots$  do
    Identify regridding regions and regrid
    for all levels  $\Omega_l$ ,  $l = 0, \dots, L - 1$  do
        /* Initialize the fields for this time step */
         $\mathbf{E}_l^{(n)} \leftarrow \mathbf{E}_l^{(n-1)}$ ,  $\mathbf{B}_l^{(n)} \leftarrow \mathbf{B}_l^{(n-1)}$ 
        /* Apply interpolation operator except for level  $L - 1$  */
         $\mathbf{E}_{l+1}^{(n)} \leftarrow \mathcal{I}[\mathbf{E}_l^{(n)}]$  on  $\Omega_{l+1,g}$ 
         $\mathbf{B}_{l+1}^{(n)} \leftarrow \mathcal{I}[\mathbf{B}_l^{(n)}]$  on  $\Omega_{l+1,g}$ 
        /* Apply sampling operator except for level 0 */
         $\mathbf{E}_{l-1}^{(n)} \leftarrow \mathcal{S}[\mathbf{E}_l^{(n)}]$  on  $\Omega_l$ 
         $\mathbf{B}_{l-1}^{(n)} \leftarrow \mathcal{S}[\mathbf{B}_l^{(n)}]$  on  $\Omega_l$ 
    end for
    /* Apply single level operators over  $\Omega_l \cup \Omega_{l,g}$  */
    for all levels  $\Omega_l$ ,  $l = 0, \dots, L - 1$  do
        Compute  $\Phi_l^{(n)}$  and  $\Psi_l^{(n)}$ 
        Apply Newton-Cotes quadrature
        Apply filter
    end for
end for

```

Chapter 5

Computational Framework

Many of the choices made in deriving AMR LDCM stems from the goal that it can be implemented as an efficient parallel algorithm. Parallelization is vital to execute large scale three dimensional simulations. But parallelization brings its own development challenges and there are many software packages which abstract away these complexities. It is up to the developer to select the appropriate tools to help build the application. The two main software packages used for the implementation of the algorithms described in this dissertation are the Chombo package [24] and FFTW [37], which is used to compute the Fourier transforms.

5.1 The Chombo Package

The Chombo package is a set of tools to help solve partial differential equations on nested rectangular grids in the AMR setting. It provides **C++** data structures to abstract away the underlying interprocessor communication using message passing interface (MPI) commands and abstractions for distributing data among processors. The user then defines the discretizations and solvers for these data structures. Although a large set of the Chombo library is dedicated for cell-centered grids, many of its lower level data structures support node-centered grids and are still directly applicable to LDCM. Another feature of the library is ChomboFortran, a set of **Fortran** and **C++** macros that interfaces the low level Chombo data structures with Fortran. This allows the user to write the core computational parts of the code in Fortran, leveraging the highly optimized **Fortran** compilers, with these macros and ChomboFortran will automatically generate the equivalent **Fortran 77** code as well as the requisite **C++** header files. The user can then call these subroutines with the **C++** side macros. The following are the three libraries that is used:

- **BaseTools**: A set of low level tools used the following two libraries.
- **BoxTools**: A set of data structures to manage data on structured rectangular grids.
- **ParticleTools**: A set of data structures to manage particle data.

In particular, the fundamental classes used in **BoxTools** are the following:

- **Vector<T>**: Chombo custom vector class.
- **Box**: A d -dimensional cell-centered rectangular box in integer space, defined by the two points $(i_{1,min}, i_{2,min}, \dots, i_{d,min}) \in \mathbb{Z}^d$ and $(i_{1,max}, i_{2,max}, \dots, i_{d,max}) \in \mathbb{Z}^d$.
- **DisjointBoxLayout**: A set of cell-centered disjoint **Box** objects. It contains the mapping of boxes to processors.
- **BoxIterator**: Index **Box** objects in a **DisjointBoxLayout**.
- **NodeFArrayBox**: Data on the surrounding grid nodes defined by a cell-centered **Box**.
- **LevelData<T>**: A templated class that represents data defined on a **DisjointBoxLayout** and creates ghost nodes around each **Box**. One of the main features of this class is that it provides built-in functions for ghost nodes communication. A level of field data is thus represented with **LevelData<NodeFArrayBox>** and a collection of levels with **Vector<LevelData<NodeFArrayBox>*>**.

In addition, the classes used in **ParticleTools** are:

- **Particle**: An object that has charge, position, and velocity.
- **ParticleData<T>**: A set of **Particle**-like objects with communication functions.
- **ParticleValidRegion**: Defines the parts of the computational domain that each processor is responsible for using a **DisjointBoxLayout**, **ParticleValidRegion** for the next coarsest level, and refinement ratio. It is used by **ParticleData** to send particles between processors. Using the same **DisjointBoxLayout** for this class and for **LevelData** ensures that each processor owns the particles in the part of the domain where it is evolving the fields.

5.2 C++ Classes for AMR LDCM

Below are the custom classes used in AMR LDCM. Following the Chombo library paradigm, C++ is used for its object-oriented programming tools and to facilitate communication. The actual computations are done by passing the stencil weights and data into **Fortran** via **ChomboFortran**.

- **EMProblem**: the main driver class; creates instances of the primary variable and solver, time steps through the problem, and calls I/O functions
- Solver classes
 - **multiLevelSolver**: advances fields by one time step, supports closed Newton-Cotes quadrature and Lawson-RK4 for particle problems
 - **maxwellProp**: creates and applies the propagator

- Primary Variables
 - `AMREMF`Field: Encapsulates the fields and collection of particles, uses the AMR Helper Classes to interpolate and sample between levels.
 - `PICEM`Particle: Child of `Particle` class, stores intermediate information for RK4.
- AMR Helper Classes
 - `NodeSample`: Samples on `LevelData<NodeFArray>` data
 - `NodeInterp`: Interpolates on `LevelData<NodeFArray>` data
- Initial Conditions and Sources Classes
 - `EMSourceAndIC`: abstract class that all the other classes below are derived from, provides an interface for `EMProblem` and `multiLevelSolver`
 - `planeWave`: plane wave initial condition, no sources
 - `movingChargeSrc`: moving charge distribution source with known analytic electrostatic solution, initial condition is the electrostatic solution
 - `laser`: linear polarized Gaussian laser initial condition, no sources
 - `svenCurrentSrc`: divergence free current source from [21], initial condition is zero
- Auxiliary Classes
 - `NodeSumOp`: helper class to mimic a sum-reduce operation on ghost nodes instead of copy-replace, used for particle deposition
 - `richardson`: helper class for convergence rate calculation

ChomboFortran is used for the stencil computations at the lowest level of these classes. The following files contain these subroutines.

- `EMParticleMesh.ChF`: deposits particles onto grid and interpolates fields to particle locations
- `interpKernel.ChF`: interpolates data from a coarse level to the next finer level
- `derivKernel.ChF`: compute derivatives

5.3 *B*-Spline Generator

The coefficients and weights for the high order *B*-splines are generated using Mathematica with `bsplineGenerator.nb` which is given in full in Appendix B. The notebook follows the framework outlined in Sec. 3.3.1 and outputs the coefficients for the piecewise polynomials which are then copy and pasted directly into the C++ code for either constructing

the kernels or for coarse-fine interpolation. Appendix B also includes a code snippet to create the convolution kernels using Chombo data structures and coefficients derived from `bsplineGenerator.nb`.

Chapter 6

Numerical Tests

This chapter presents the convergence results for several test problems. These problems are representative of different types of source terms. The first problem is plane wave with no sources propagating with periodic boundary conditions. The plane wave is an eigenfunction for Maxwell's equations. The second problem is a divergence free current source term. The third problem is a moving charge density. For the divergence free current and moving charge problems, there is an analytic expression for the static case and that is used as the initial condition. Finally, the weak scaling of the solver is shown through several different parallelizations on NERSC Cori Phase I.

The results are computed using 5th and 15th order solvers. Let $W_{q,p}$ denote the q -th order C^p B-spline (given in Appendix C). The 5th order solver uses $W_{6,0}$ as the basis for the regularized delta and $W_{6,6}$ for interpolation between refinement levels. The solver uses the 4-point 3/8 Simpson's rule for closed Newton-Cotes quadrature. The 15th order solver uses $W_{16,0}$ as the basis for the regularized delta. For both solvers, the speed of light is one, the number of quadrature points θ direction for spherical quadrature is twenty-four, and the refinement ratio is four. Since $c = 1$, the CFL for level l is $\sigma_l = \frac{\Delta t}{h_l}$ with the notation that $\sigma = \sigma_0$.

The figures of the fields in this chapter are produced using Visit [20].

6.1 Plane Wave Propagation

Consider plane waves of the form

$$\mathbf{E}(\mathbf{x}, t) = \sin(\mathbf{k} \cdot \mathbf{x} - \omega t) \boldsymbol{\psi} \quad (6.1)$$

$$\mathbf{B}(\mathbf{x}, t) = \sin(\mathbf{k} \cdot \mathbf{x} - \omega t) (\hat{\mathbf{k}} \times \boldsymbol{\psi}) \quad (6.2)$$

where $\mathbf{k}, \boldsymbol{\psi} \in \mathbb{R}^3$, $\mathbf{k} \cdot \boldsymbol{\psi} = 0$, and $\omega = |\mathbf{k}|$. $\boldsymbol{\psi}, \hat{\mathbf{k}} \times \boldsymbol{\psi}, \mathbf{k}$ form an orthogonal basis for $\mathbb{R}^3$. For numerical tests it will be desirable to specify the propagation direction, $\mathbf{k} = \frac{2\pi}{\lambda} \hat{\mathbf{k}}$. Define

$$\boldsymbol{\psi} = (\mathbb{I} - \hat{\mathbf{k}} \hat{\mathbf{k}}^T) \mathbf{v}, \quad (6.3)$$

$\mathbf{v} = (1, 1, 1)$ is chosen arbitrarily.

6.1.1 Plane Wave Propagating in $\hat{\mathbf{x}}$

The first numerical test is with a plane wave propagating in the x direction with $\lambda = 1$ and $\sigma = 1$. In this case, $\mathbf{k} = 2\pi(1, 0, 0)$, $\boldsymbol{\psi} = (0, 1, 1)$, and $\hat{\mathbf{k}} \times \boldsymbol{\psi} = (0, -1, 1)$. Specifically, (6.1)-(6.2) becomes

$$\mathbf{E}(\mathbf{x}, t) = \begin{pmatrix} 0 \\ \sin(2\pi x - 2\pi t) \\ \sin(2\pi x - 2\pi t) \end{pmatrix}, \quad \mathbf{B}(\mathbf{x}, t) = \begin{pmatrix} 0 \\ -\sin(2\pi x - 2\pi t) \\ \sin(2\pi x - 2\pi t) \end{pmatrix}. \quad (6.4)$$

The computational domain for this problem is a unit box, with N grid points at each direction, and one level of refinement at $[\frac{1}{2}, \frac{5}{8}]^3$. Figure 6.1 shows the expected 5th order convergence on both levels for E_y .

This problem is repeated using a 2D (with the z direction summed out) 15th order solver for both a single level case and the case with the same refinement region in xy as above. The problem parameters and convergence rates are summarized in Table 6.1. For the two level case, $W_{6,6}$ was used for coarse-fine interpolation and therefore the convergence rate is only 5.8 as expected from fields that are only C^6 .

Figure 6.1: E_y ℓ_∞ error at $t = \frac{10}{32}$ for the plane wave problem. The test cases are $N = (33, 65, 129, 257)$ with $N_{timesteps} = (10, 20, 40, 80)$ respectively. Least squares fit on the coarse error shows convergence rate $h^{4.795}$ and for the fine error $h^{4.806}$.

6.1.2 Non-Axis Aligned Plane Wave

In order to use periodic boundary conditions, it is necessary that the components of $\mathbf{k}$ are integer multiples of 2π . The next test is ran with the 5th order solver with the same two

Table 6.1: Problem parameters and convergence rates from Richardson error estimate in ℓ_2 for E_y, E_z, B_y, B_z using the 15th order solver for the axis aligned plane wave problem.

	Single level	Two levels
Grid points per direction (N)	(33, 65, 129)	(65, 129, 257)
Wavelength (λ)	1	$\frac{8}{64}$
Final Time (t)	$\frac{10}{32}$	$\frac{20}{64}$
Convergence Rate	14.87	5.8

levels and $\mathbf{k} = 2\pi(1, 3, 5)$. This gives $\boldsymbol{\psi} = (\frac{26}{35}, \frac{8}{35}, -\frac{2}{7})$ and $\mathbf{k} \times \boldsymbol{\psi} = (-\frac{2}{\sqrt{35}}, \frac{4}{\sqrt{35}}, -\frac{2}{\sqrt{35}})$. Table 6.2 shows the result and for both levels it shows the expected 5th order convergence.

Figure 6.2: $E_x \ell_\infty$ error at $t = \frac{10}{32}$ for the non-axis aligned plane wave problem. The test cases are $N = (65, 129, 257)$ with $N_{timesteps} = (20, 40, 80)$ respectively. Least squares fit on the coarse error shows convergence rate $h^{5.056}$ and for the fine error $h^{5.104}$.

6.2 Divergence-Free Current Source

The next test is a divergence-free current source. The results for this test and the tests using the smooth charge source problem described in the next section are also presented in [64]. Such a source term does not have a charge distribution associated with it and so it is the

Figure 6.3: ℓ_∞ error values and convergence results for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$ for the divergence-free current problem as a function of time in Ω_2 . On the left are the normalized ℓ_∞ errors for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$. The errors for E_x are obtained from the difference of sampled field values from the $N = 257$ with $N = 129$ and also from sampled $N = 129$ with $N = 65$ test case. The E_x error is normalized by $|\frac{4\pi}{\nu} \max_{r,z} J_x| \approx |10.2341 \sin(2\pi\nu t)|$ and $\nabla \cdot \mathbf{E}$ is normalized by $|\frac{4\pi}{\nu a} \max_{r,z} J_x| \approx |545.8187 \sin(2\pi\nu t)|$. On the right are the associated convergence rates.

next step in difficulty from a no source problem. The current is of the form [21]

$$J_x(x, y, z, t) = -100 \frac{y - y_0}{r} \sin\left(\frac{\pi r}{2a}\right) \cos^{10}\left(\frac{\pi r}{2a}\right) \cos^{11}\left(\frac{\pi(z - z_0)}{d}\right) \sin(2\pi\nu t), \quad (6.5)$$

$$J_y(x, y, z, t) = 100 \frac{x - x_0}{r} \sin\left(\frac{\pi r}{2a}\right) \cos^{10}\left(\frac{\pi r}{2a}\right) \cos^{11}\left(\frac{\pi(z - z_0)}{d}\right) \sin(2\pi\nu t), \quad (6.6)$$

$$J_z(x, y, z, t) = 0, \quad (6.7)$$

where $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. The initial conditions for the fields are zero. This test is computed with open boundary conditions with parameters: $a = \frac{3}{160}$, $d = \frac{13}{320}$, $x_0 = y_0 = z_0 = 0.5$, $\nu = 20$. This test is performed on fixed grids with two refinement levels: $\Omega_1 = [\frac{3}{8}, \frac{5}{8}]^3$ and $\Omega_2 = [\frac{15}{32}, \frac{17}{32}]^3$, $N = (65, 129, 257)$ with $\Delta t = (\frac{1}{1024}, \frac{1}{2048}, \frac{1}{4096})$ respectively, this corresponds to CFL = 1 at the finest level, out to $t_{final} = \frac{200}{1024}$. Figure 6.3 shows the E_x Richardson convergence rate estimate and the associated ℓ_∞ error as well as the absolute convergence rate and associated ℓ_∞ errors for $\nabla \cdot \mathbf{E} - 4\pi\rho$ on the three grids in Ω_2 as a function of time step and as expected the solution shows 4th order convergence because of the source integration scheme.

6.3 Smooth Charge Distribution

This next set of problems introduces a non zero right hand side to Gauss's Law and represents the next step from a divergence-free current. This problem will be used to test a moving charge distribution, recovery of the electrostatic solution, and the regridding algorithm. The

charge distribution is spherically symmetric and decays smoothly to zero from its center to the edge of a sphere with radius R_0 . This charge distribution and its electric potential are given by

$$\nabla^2 \phi = -4\pi\rho, \quad (6.8)$$

$$\rho(\mathbf{x}) = a \begin{cases} (r - r^2)^6, & r < 1 \\ 0, & r \geq 1 \end{cases}, \quad (6.9)$$

$$\phi(\mathbf{x}) = -4\pi R_0^2 a \begin{cases} \frac{r^8}{72} - \frac{r^9}{15} + \frac{3r^{10}}{22} - \frac{5r^{11}}{33} + \frac{5r^{12}}{12} - \frac{3r^{13}}{91} + \frac{r^{14}}{210} - \frac{1}{24024}, & r < 1 \\ -\frac{1}{45045r}, & r \geq 1 \end{cases}, \quad (6.10)$$

where $r = \frac{1}{R_0} \|\mathbf{x} - \mathbf{x}_0\|$ and $\mathbf{x}_0$ is the center of the charge distribution. This potential generates the electric field

$$\mathbf{E}(\mathbf{x}) = -\nabla\phi = 4\pi R_0 a \hat{r} \begin{cases} \frac{r^7}{9} - \frac{3r^8}{5} + \frac{15r^9}{11} - \frac{5r^{10}}{3} + \frac{15r^{11}}{13} - \frac{3r^{12}}{7} + \frac{r^{13}}{15}, & r < 1 \\ \frac{1}{45045r^2}, & r \geq 1 \end{cases}, \quad (6.11)$$

$$\hat{r} = \sin\theta \cos\varphi \hat{x} + \sin\theta \sin\varphi \hat{y} + \cos\theta \hat{z}, \quad (6.12)$$

$$\theta = \arccos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right), \quad (6.13)$$

$$\varphi = \arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right), \quad (6.14)$$

where the coordinates (x, y, z) in the last three formulas are relative to $\mathbf{x}_0$. $\phi(\mathbf{x})$ is the C^7 version of the charge distribution used in [57]. For the tests in this section, this is used as the initial condition for $\mathbf{E}$ with $\mathbf{B} = 0$. The tests in this section are computed with the 4th order solver.

6.3.1 Translating Charge Source

Suppose instead of a fixed center $\mathbf{x}_0$, the center of the charge distribution (and thus the distribution as a whole) is allowed to move. This will induce a current which needs to be accounted for. Let $\mathbf{v}_0 \equiv \dot{\mathbf{x}}_0$

$$\rho(\mathbf{x}, t) = \begin{cases} a(r(t) - r(t)^2)^6, & r < 1 \\ 0, & r \geq 1 \end{cases}, \quad (6.15)$$

$$\mathbf{J}(\mathbf{x}, t) = \begin{cases} \mathbf{v}_0 a(r(t) - r(t)^2)^6, & r < 1 \\ 0, & r \geq 1 \end{cases}, \quad (6.16)$$

$$\mathbf{v}_0 = \nu d\pi \frac{35}{16} \sin^7(2\pi\nu t). \quad (6.17)$$

Figure 6.4: ℓ_∞ error values and convergence results for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$ for the translating spherical charge distribution problem as a function of time in Ω_2 . On the left are the normalized ℓ_∞ errors for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$. The errors for E_x are obtained from the difference of sampled field values from the $N = 257$ with $N = 129$ and also from sampled $N = 129$ with $N = 65$ test case. The E_x error is normalized by the max norm of the electrostatic solution (≈ 0.0694795) and $\nabla \cdot \mathbf{E} - 4\pi\rho$ error is normalized by $\max_{\mathbf{x}} 4\pi\rho \approx 30.6796$. On the right are the associated convergence rates.

This test is performed on fixed grids with two refinement levels: $\Omega_1 = [\frac{3}{8}, \frac{5}{8}]^3$ and $\Omega_2 = [\frac{15}{32}, \frac{17}{32}]^3$, with parameters: $a = 10^4, d = \frac{1}{256}, \nu = \frac{1024}{80}, R_0 = \frac{1}{72}, \mathbf{x}_0 = (\frac{127}{256}, \frac{127}{256}, \frac{127}{256})$, $= (\cos \frac{\sqrt{3}}{3} \cos \frac{\sqrt{2}}{3}, \sin \frac{\sqrt{3}}{3} \cos \frac{\sqrt{2}}{3}, \sin \frac{\sqrt{2}}{3})$, $N = (65, 129, 257)$ with $\Delta t = (\frac{1}{1024}, \frac{1}{2048}, \frac{1}{4096})$ respectively, this corresponds to $\text{CFL} = 1$ at the finest level, out to $t_{\text{final}} = \frac{200}{1024}$. The spatial grid and time steps are the same as the divergence-free current test. Figure 6.4 shows the E_x Richardson convergence rate estimate and the associated ℓ_∞ error as well as the absolute convergence rate and associated ℓ_∞ errors for $\nabla \cdot \mathbf{E} - 4\pi\rho$ on the three grids in Ω_2 as a function of time step and as expected the solution shows 4th order convergence.

6.3.2 Electrostatics Test

If the charge distribution moves to a final location and stops, then the fields will return to their electrostatic values with the new center. To test this, the same discretization and parameters are used but the charge distribution is stopped after $t = \frac{40}{1024}$ and then run out to $t_{\text{final}} = \frac{100}{1024}$ to show that the solver recovers the electrostatic solution. Figure 6.5 shows the E_x Richardson convergence rate estimate and the associated ℓ_∞ error as well as the absolute convergence rate and associated ℓ_∞ errors for $\nabla \cdot \mathbf{E} - 4\pi\rho$ on the three grids in Ω_2 as a function of time step and as expected the solution shows 4th order convergence. In addition, Figure 6.6 shows that there are no reflected waves at the refinement boundaries at the final time.

Figure 6.5: ℓ_∞ error values and convergence results for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$ for the stopped spherical charge distribution problem as a function of time in Ω_2 . On the left are the normalized ℓ_∞ errors for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$. The errors for E_x are obtained from the difference of sampled field values from the $N = 257$ with $N = 129$ and also from sampled $N = 129$ with $N = 65$ test case. The E_x error is normalized by the max norm of the electrostatic solution (≈ 0.0694795) and $\nabla \cdot \mathbf{E} - 4\pi\rho$ error is normalized by $\max_{\mathbf{x}} 4\pi\rho \approx 30.6796$. On the right are the associated convergence rates. The vertical line indicates the time at which the charge distribution stops moving.

6.3.3 Regridding Test

The translating charge distribution was also used to test the regridding algorithm with $\hat{\mathbf{x}} = \nu d\pi \sin(2\nu t)\hat{\mathbf{x}}$, $a = \frac{1}{160}$, $d = \frac{1}{64}$, $\mathbf{x}_0 = (\frac{31}{64}, \frac{1}{2}, \frac{1}{2})$, $\nu = \frac{1024}{80}$, $t_{final} = \frac{800}{1024}$, and other parameters being the same. Ω_1 is kept the same and fixed but Ω_2 is subject to be regridded. It starts with $\Omega_{2,a}$ and changes between $\Omega_{2,a}$ and $\Omega_{2,b}$ whenever the x coordinate of the center of the charge distribution crosses $\frac{63}{128}$, where $\Omega_{2,a}$ is the rectangular prism defined by the corner points $(\frac{29}{64}, \frac{17}{32}, \frac{17}{32})$ and $(\frac{33}{64}, \frac{17}{32}, \frac{17}{32})$, and $\Omega_{2,b} = [\frac{15}{32}, \frac{17}{32}]^3$; effectively Ω_2 oscillates in the x direction with amplitude $\frac{1}{64}$ in the direction of the charge motion. Fig 6.7 shows E_x and the regridding domains for $N = 129$. Figure 6.8 shows the E_x Richardson convergence rate estimate and the associated ℓ_∞ error as well as the absolute convergence rate and associated ℓ_∞ errors for $\nabla \cdot \mathbf{E} - 4\pi\rho$ on the three grids in Ω_2 as a function of time step and our solution shows 5th order convergence.

6.4 Performance Results

The performance of the solver is timed with the 5th order solver on the single level axis aligned plane wave problem without periodic boundary conditions. These tests are performed on NERSC Cori Haswell partition; each compute node has two processors, one socket per processor, and each processor has sixteen physical cores and each core has two hyperthreads. For these tests, the hyperthreading capability has not been used. The problem sizes are characterized by N , the number of grid points in each spatial direction, and the total

Figure 6.6: $\log_{10}(|\nabla \cdot \mathbf{E} - 4\pi\rho|/\max_{\mathbf{x}} 4\pi\rho)$ at $z = 0.5$ for the stopped translating spherical charge distribution problem at $t = \frac{100}{1024}$ for $N = 129$ showing that there are no reflected waves at the refinement boundaries.

number of ghost nodes g , thus each patch has $(N + g)^3$ nodes. The four parallelizations tested are:

1. Pure MPI: Eight $(33 + 12)^3$ nodes patches per MPI task and one MPI task per core. Table 6.2
2. Pure MPI: One $(65 + 12)^3$ nodes patch per MPI task and one MPI task per core. Table 6.3
3. MPI-OpenMP hybrid: Eight $(33 + 12)^3$ nodes patches per MPI task and one MPI task per socket. Table 6.4
4. MPI-OpenMP hybrid: Eight $(33 + 12)^3$ nodes patches per MPI task and two MPI tasks per socket. Table 6.5

The solver is ran with two problem sizes, $N = 257$ with $N_{timesteps} = 20$ and $N = 513$ with $N_{timesteps} = 40$, where the bigger problem is given eight times as much computational resource as the smaller problem. For perfect weak scaling, it is expected the time spent by the solver for the bigger problem is twice that of the time spent for the smaller problem.

user: borislo
Wed Apr 25 18:39:43 2018

(a) $t = \frac{256}{2048}$; charge distribution moving to the right, it has almost reached its rightmost position. $\Omega_2 = \Omega_{2,b}$

user: borislo
Wed Apr 25 18:39:45 2018

(b) $t = \frac{480}{2048}$; charge distribution is at its leftmost position. $\Omega_2 = \Omega_{2,a}$

user: borislo
Wed Apr 25 18:39:46 2018

(c) $t = \frac{864}{2048}$; charge distribution moving to the left. $\Omega_2 = \Omega_{2,b}$

user: borislo
Wed Apr 25 18:39:48 2018

(d) $t = \frac{1600}{2048}$; final time step. $\Omega_2 = \Omega_{2,a}$

Figure 6.7: E_x minus the instantaneous electrostatic solution, at $z = \frac{1}{2}$, for the spherical charge distribution problem with regridding for $N = 129$.

Figure 6.8: ℓ_∞ error values and convergence results for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$ for the regriding spherical charge distribution problem as a function of time in Ω_2 . On the left are the normalized ℓ_∞ errors for E_x and $\nabla \cdot \mathbf{E} - 4\pi\rho$. The errors for E_x are obtained from the difference of sampled field values from the $N = 257$ with $N = 129$ and also from sampled $N = 129$ with $N = 65$ test case. The E_x error is normalized by the max norm of the electrostatic solution (≈ 0.0312658) and $\nabla \cdot \mathbf{E} - 4\pi\rho$ error is normalized by $\max_{\mathbf{x}} 4\pi\rho \approx 30.6796$. On the right are the associated convergence rates. The vertical lines are the times at which regriding occurs.

The tables indeed demonstrate weak scaling for the solver for all four parallelizations. The time spent in the propagator, $t_{\text{propagator}}$, and the time that data is being copied into the ghost nodes, t_{copy} , are both only twice as long for the bigger problem. The bandwidth for data motion is also estimated for these tests. For the first two parallelizations, the estimated message size per MPI communication call is the product of the number of ghost nodes $((65 + 12)^3 - 65^3)$, bytes per double (8), number of components (12). This gives a message size of 17 MB. The total data volume moved is then the message size times the number of MPI tasks. There are 6 MPI exchanges per time step. The estimated bandwidth is then given by total data volume divided per time per communication. This is an exact calculation for test 2, however for test 1 this assumes that the eight patches within a core form a cube. For test 3, the estimated number of ghost nodes is given by $(257 + 12)(129 + 12)^2$, this gives a message size of 103 MB. Finally for test 4, the estimated number of ghost nodes is given by $(65 + 12)(129 + 12)^2$, this gives a message size of 43 MB. According to the benchmarking results presented in Figure 1 in [31], the Cori ping-pong benchmark internode bandwidth caps out at 8 GB/s. While the in-node fetching data from DRAM roundtrip has a cap at ~ 100 GB/s. The estimates in Table 6.2-6.5 show a far larger estimated bandwidth than the ping-pong benchmark. This suggests most of the data motion is happening within node and that the Chombo patch-to-processor assignment is sensible.

The last performance test studies the scaling with respect to the patch size. This test is done using pure MPI, with one patch per MPI task and one MPI task per core, for $N = 129$ problem with $N_{\text{timesteps}} = 10$. The number of ghost nodes selected for this test are selected so that $N + g = 2^a 3^b 5^c 7^d 11^e 13^f$, where $e + f$ is either 0 or 1, and the other exponents are

Table 6.2: There are eight patches per core and one MPI task per core. Each patch has $(33 + 12)^3$ nodes. The time are all reported in seconds. $t_{copy} = t_{communication} - t_{MPI_WaitAll}$

N	N_{cores}	$N_{timesteps}$	$t_{propagator}$	$t_{communication}$	$t_{MPI_WaitAll}$	t_{copy}	Estimated Bandwidth (MB/s)
257	64	20	44.01	12.09	6.66	5.43	20137.71
513	512	40	89.78	23.71	13.66	10.05	157091.81

Table 6.3: There is one patch per core and one MPI task per core. Each patch has $(65 + 12)^3$ nodes. The time are all reported in seconds. $t_{copy} = t_{communication} - t_{MPI_WaitAll}$

N	N_{cores}	$N_{timesteps}$	$t_{propagator}$	$t_{communication}$	$t_{MPI_WaitAll}$	t_{copy}	Estimated Bandwidth (MB/s)
257	64	20	44.82	13.03	11.33	1.7	11837.35
513	512	40	90.1	31.52	28.03	3.49	76556.34

Table 6.4: There is one MPI task per socket and the computation has eight patches per core. Each patch has $(33 + 12)^3$ nodes. OpenMP is used to parallelize the calculation within a socket. The time are all reported in seconds. $t_{copy} = t_{communication} - t_{MPI_WaitAll}$

N	N_{cores}	$N_{timesteps}$	$t_{propagator}$	$t_{communication}$	$t_{MPI_WaitAll}$	t_{copy}	Estimated Bandwidth (MB/s)
257	64	20	60.5	43.02	8.64	34.38	5713.34
513	512	40	125.59	100.68	25.37	75.31	31131.76

Table 6.5: There are two MPI tasks per socket and the computation has eight patches per core. Each patch has $(33 + 12)^3$ nodes. OpenMP is used to parallelize the calculation within a socket. The time are all reported in seconds. $t_{copy} = t_{communication} - t_{MPI_WaitAll}$

N	N_{cores}	$N_{timesteps}$	$t_{propagator}$	$t_{communication}$	$t_{MPI_WaitAll}$	t_{copy}	Estimated Bandwidth (MB/s)
257	64	20	61.39	24.18	3.51	20.67	11793.64
513	512	40	120.66	70.26	29.96	40.3	22107.18

arbitrary. These sizes are optimized for FFTW [37]. The result is presented in Table 6.6 and it can be seen that the time spend in the propagator reduces by a factor of eight only going from no parallelization to decomposing the domain into eight patches. As the patch size reduces, the number of ghost nodes become comparable to N_{patch} and thus the patches are not strictly reduced by a factor of eight for each successive test. Therefore, the time spend in the propagator does not exhibit a factor of eight reduction.

Table 6.6: Testing the scaling as a function of number of grid nodes per patch. This test uses one patch per core for 10 timesteps. The total number of grid points per patch is $(N_{patch} + N_{ghost})^3$.

N_{patch}	N_{ghost}	N_{cores}	$t_{propagator}$	$t_{communication}$
129	36	1	161.99	4.29
65	26	8	21.40	4.36
33	22	64	6.53	1.97
17	22	512	1.60	1.70

Chapter 7

Conclusions

A high order propagator method using local discrete convolutions for Maxwell's equations is presented. The main motivation is computational efficiency. Propagator methods have more flexible time step constraint for a given level of accuracy and high-order methods require fewer degrees of freedom. Discretization using local discrete convolutions and co-located grids allows the method to be parallelized with domain decomposition and adaptive mesh refinement. The fundamental pieces in the propagator are the spherical means integrals. These integrals are discretized on a uniform rectangular grid using regularized delta distributions with compact support. This converts the propagator updates into stencil operations on the grid which can then be computed with discrete convolutions. Since the discrete convolution kernels have compact support, the update over a large computational domain can be parallelized with standard domain decomposition and the convolutions can be computed exactly using Hockney's method. In the presence of charge and current source terms, if these sources are known functions, independent of the fields, then they can be integrated with a quadrature scheme. If these source terms are dependent of the fields and are themselves governed by a set of evolution equations, then the propagator can be embedded into the overall system with a change of variables and then a time integrator can be applied to this overall system. Because the method uses the same grids for both the electric and magnetic fields, the method can be extended to use local mesh refinement much easier than traditional staggered grid based methods. And since the method is unconditionally stable, one can advance the fields on different grids with the same time step.

A selection of numerical tests are performed using 5th, and 15th order accurate solvers. These tests represent a sample of source terms in electromagnetic problems. The plane wave with no source terms is an eigenmode of Maxwell's equations. An axis aligned and a non-axis aligned plane waves are used as initialize conditions and evolved with periodic boundary conditions. The two solvers demonstrated the expected accuracy in single level and two level tests. Next, a divergence free current source with no charge density is tested. The 5th order solver again showed the expected 5th order accuracy in a two level test. Then, a smoothly spatially decaying radial charge density with rigid motion is used to test the regridding capability and shows that there is only minimal errors reflecting back into the fine level from the coarse-fine boundary. For the two test problem sizes, the solver demonstrate weak

scaling for both in the propagator step and the communication step minus the time waiting for the data to be transfered over the network.

7.1 Future Work

For the wave equation, the Green's function is related to the propagator via a simple relationship [11]

$$G(\mathbf{x}, t|\mathbf{x}', t') = \theta(t - t')K(\mathbf{x}, t|\mathbf{x}', t), \quad (7.1)$$

where G is the Green's function, θ is the Heaviside step function, and K is the propagator. Therefore much of this work can be readily extended to the existing Green's function methods for Maxwell's equations and wave equation problems in general.

Another avenue for future work is to test the field solver's capabilities in the PIC setting. Although the tests we have performed for a single test particle problem have been inconclusive, there are some lessons we have learned. The issue is that for a singular distribution, *i.e.* a point charge, the charge and current depositions must be handled with care otherwise spurious self forcing will lead to unphysical solutions. Dealing with spurious self forcing is not trivial even in the electrostatic case [25].

In electromagnetics, the approach to this problem is to ensure that the deposited current and charge satisfy the discrete continuity equation exactly. The method for this has been the Boris correction which involves a global elliptic solve [15]. To combat this computational inefficiency, various local approximations have been explored [89]. The existing deposition schemes rely on the approximation that the particle moves in a straight line in a time step [33, 89]. The general idea is to integrate the continuity equation exactly to compute the current deposition or a correction to it, and it requires further research to develop a scheme suitable in the context of Lawson's method with LDCM (co-located spatio-temporal grid and non constant current within a time step). This can be done by filtering the current so that it satisfies the discrete continuity equation, this is analogous to the approach outlined in [89]. Another approach is to take advantage of the structure of the auxiliary equations. As shown in Sec. 2.3.2, the $\nabla\rho$ source term that appears in the evolution equation for Φ (2.8) serves to correct the error in propagating the longitudinal component of the electric field and current source. Therefore, if the charge and current deposition schemes don't satisfy the continuity equation, one idea is to insert a $\delta\rho$ term to compensate for propagating the erroneous current. Yet another test is to incorporate a LDCM field solver in existing open source PIC projects such as Warp-X [88] or SMILEI [30].

Appendix A

Method of Spherical Means

Consider the homogenous 3D scalar first order wave equation with initial conditions

$$\begin{aligned}\frac{\partial u}{\partial t} &= c\phi, \\ \frac{\partial \phi}{\partial t} &= c\Delta u, \\ u(t=0) &= u_0, \\ \phi(t=0) &= \phi_0.\end{aligned}\tag{A.1}$$

The solution to this equation is given by the Method of Spherical Means [35, 94]. It is constructed from the solution to the 1D scalar wave equation applied onto the modified spherical mean quantity. The derivation given in this appendix follows from [3, 2, 35].

A.1 d'Alembert's Formula for the Homogenous 1D Wave Equation

Starting with the 1D wave equation

$$\begin{aligned}\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} &= 0, \\ u(t=0) &= u_0(x), \\ \frac{\partial u}{\partial t}(t=0) &= \phi_0(x).\end{aligned}\tag{A.2}$$

Introducing a change of variables $\mu = x + ct, \eta = x - ct$, the equation can be rewritten

$$\left(\frac{\partial}{\partial x} + \frac{\partial}{c\partial t}\right) \left(\frac{\partial}{\partial x} - \frac{\partial}{c\partial t}\right) u = \frac{\partial^2 u}{\partial \mu \partial \eta} = 0.\tag{A.3}$$

The solution to this equation is

$$\begin{aligned}u(\mu, \eta) &= F(\mu) + G(\eta) \\ u(x, t) &= F(x + ct) + G(x - ct).\end{aligned}\tag{A.4}$$

Taking the time derivative

$$\frac{\partial u}{\partial t} = cF'(x + ct) - cG'(x - ct). \quad (\text{A.5})$$

Taking $t = 0$

$$u(x, 0) = F(x) + G(x) = u_0 \quad (\text{A.6})$$

$$\left. \frac{\partial u}{\partial t} \right|_{t=0} = cF'(x) - cG'(x) = \phi_0. \quad (\text{A.7})$$

Integrating the second equation

$$cF(x) - cG(x) = \int_{-\infty}^x \phi_0(s) ds + C. \quad (\text{A.8})$$

Solving for F, G

$$F(x) = -\frac{1}{2c} \left(-cu_0 - \left(\int_{-\infty}^x \phi_0(s) ds + C \right) \right) \quad (\text{A.9})$$

$$G(x) = -\frac{1}{2c} \left(-cu_0 + \left(\int_{-\infty}^x \phi_0(s) ds + C \right) \right). \quad (\text{A.10})$$

Substituting back into (A.4)

$$u(x, t) = \frac{1}{2} [u_0(x - ct) + u_0(x + ct)] + \frac{1}{2c} \int_{x-ct}^{x+ct} \phi_0(s) ds. \quad (\text{A.11})$$

This is d'Alembert's formula for the 1D wave equation.

A.2 Euler-Poisson-Darboux Equation for the Spherical Mean Variable

For the 3D homogenous wave equation, first consider the spherical mean variable

$$\hat{u}(\mathbf{x}, t, r) = \frac{1}{4\pi r^2} \int_{\partial B_r(\mathbf{x})} u(\mathbf{z}, t) dS(\mathbf{z}), \quad (\text{A.12})$$

where $\partial B_r(\mathbf{x})$ is the sphere of radius r centered at $\mathbf{x}$. As $r \rightarrow 0$, the spherical mean approaches to a delta distribution centered at $\mathbf{x}$

$$\lim_{r \rightarrow 0^+} \hat{u}(\mathbf{x}, t, r) = u(\mathbf{x}, t). \quad (\text{A.13})$$

The spherical mean variable obeys the Euler-Poisson-Darboux equation. With a change of variables $\mathbf{z} = \mathbf{x} + r\mathbf{y}$

$$\begin{aligned}\hat{u}(\mathbf{x}, t, r) &= \frac{1}{4\pi r^2} \int_{\partial B_{r(\mathbf{x})}} u(\mathbf{z}, t) dS(\mathbf{z}) \\ &= \frac{1}{4\pi} \int_{\partial B_{1(0)}} u(\mathbf{x} + r\mathbf{y}, t) dS(\mathbf{y}).\end{aligned}$$

Taking the r derivative

$$\frac{\partial \hat{u}(\mathbf{x}, t, r)}{\partial r} = \frac{1}{4\pi} \int_{\partial B_{1(0)}} \nabla u(\mathbf{x} + r\mathbf{y}, t) \cdot \mathbf{y} dS(\mathbf{y})$$

Converting back to $\mathbf{z}$

$$\begin{aligned}&= \frac{1}{4\pi r^2} \int_{\partial B_{r(\mathbf{x})}} \nabla u(\mathbf{z}, t) \cdot \frac{\mathbf{z} - \mathbf{x}}{r} dS(\mathbf{z}) \\ &= \frac{1}{4\pi r^2} \int_{\partial B_{r(\mathbf{x})}} \nabla u(\mathbf{z}, t) \cdot \hat{\mathbf{n}} dS(\mathbf{z})\end{aligned}$$

By the Divergence Theorem

$$= \frac{1}{4\pi r^2} \int_{B_{r(\mathbf{x})}} \Delta u(\mathbf{z}, t) d\mathbf{z}$$

Since u satisfies the wave equation

$$= \frac{1}{4\pi r^2} \int_{B_{r(\mathbf{x})}} \frac{\partial^2 u}{\partial t^2} d\mathbf{z}.$$

Taking another r derivative

$$\begin{aligned}\frac{\partial r^2 \hat{u}(\mathbf{x}, t, r)}{\partial r} &= \frac{1}{4\pi} \int_{\partial B_{r(\mathbf{x})}} \frac{\partial^2 u}{\partial t^2} dS(\mathbf{z}) \\ &= \frac{r^2}{4\pi r^2} \int_{\partial B_{r(\mathbf{x})}} \frac{\partial^2 u}{\partial t^2} dS(\mathbf{z}) \\ &= r^2 \hat{u}_{tt}.\end{aligned}$$

Finally using the chain rule on the LHS

$$\begin{aligned}2r\hat{u}_r + r^2\hat{u}_{rr} &= r^2\hat{u}_{tt} \\ \hat{u}_{tt} - \hat{u}_{rr} - \frac{2}{r}\hat{u}_r &= 0.\end{aligned}\tag{A.14}$$

This is the Euler-Poisson-Darboux Equation in 3D.

A.3 Method of Spherical Means: Solution of the Homogenous 3D Wave Equation

Defining the modified spherical mean as

$$\bar{u} = r\hat{u} \quad (\text{A.15})$$

Then taking two time derivatives

$$\bar{u}_{tt} = r\hat{u}_{tt}$$

Using the Euler-Poisson-Darbous Equation (A.14)

$$\begin{aligned} &= r\hat{u}_{rr} + 2\hat{u}_r \\ &= (r\hat{u}_r + \hat{u})_r \\ &= (r\hat{u})_{rr} \\ &= \bar{u}_{rr}, \end{aligned}$$

That is, the modified spherical mean quantity satisfies the 1D wave equation

$$\bar{u}_{tt} - \bar{u}_{rr} = 0, \quad (\text{A.16})$$

the initial conditions are

$$\bar{u}(0) = r\hat{u}(0) = r\hat{u}_0 = \bar{u}_0, \quad (\text{A.17})$$

$$\bar{u}_t(0) = r\hat{u}_t(0) = r\hat{\phi}_0 = \bar{\phi}_0. \quad (\text{A.18})$$

Using d'Alembert's Formula (A.11) for $\bar{u}$

$$\bar{u}(x, t) = \frac{1}{2}[\bar{u}_0(r - ct) + \bar{u}_0(r + ct)] + \frac{1}{2c} \int_{r-ct}^{r+ct} \bar{\phi}_0(s) ds. \quad (\text{A.19})$$

Using the limit of the spherical mean variable (A.13)

$$\begin{aligned} \lim_{r \rightarrow 0^+} \frac{\bar{u}}{r} &= \lim_{r \rightarrow 0^+} \left[\frac{1}{2r} [\bar{u}_0(r - ct) + \bar{u}_0(r + ct)] + \frac{1}{2cr} \int_{r-ct}^{r+ct} \bar{\phi}_0(s) ds \right], \\ u(x, t) &= \frac{1}{c} \frac{\partial \bar{u}_0}{\partial t} + \bar{\phi}_0. \end{aligned} \quad (\text{A.20})$$

For $\phi(x, t)$, take the time derivative and divide by c

$$\frac{1}{c} \frac{\partial u}{\partial t} = \frac{1}{c^2} \frac{\partial^2 \bar{u}_0}{\partial t^2} + \frac{1}{c} \frac{\partial \bar{\phi}_0}{\partial t},$$

since $\bar{u}$ satisfies the wave equation

$$\phi(x, t) = \Delta \bar{u}_0 + \frac{1}{c} \frac{\partial \bar{\phi}_0}{\partial t}. \quad (\text{A.21})$$

(A.20)-(A.21) are the solution to the 3D homogenous wave equation from the Method of Spherical Means.

Appendix B

Spherical Means Kernels Generator Code

These are the codes used to generate the convolution kernels on a rectangular grid for LDCM. There are two pieces. The first is the Mathematica notebook, `bsplineMathematica.nb`, used to generate high order B -splines as described in Sec. 3.3. The script outputs the continuity, weights, the piecewise polynomial in functional form, the coefficients of the polynomial in array form and list form, and plots the interpolant. The weights are referred to as **alphas** instead of a . Here it is making an interpolant with support width eight and accuracy six. The second piece is a C++ code that takes the coefficients from the notebook and makes the Fourier transformed version of the kernels.

```
(* Generate high order B-spline *)

In[1]:= (* Input Parameters *)
support = 8;
order = 6;

In[3]:= (* Compute the Taylor series expansion of base B-Spline *)
n = support;
taylorOrder = order + 4;
f[k_] := Series[(Sin[k/2] / (k/2))^n, {k, 0, taylorOrder}]

In[6]:= (* Max order of derivatives and polynomials to multiply by IN FOURIER SPACE *)
maxDeriv = n - order;
maxPoly = n - 1;
```

```

In[8]:= (* Iteratively create system, solve,
remove polynomial multiplications and repeat until unsolvable *)
(* B alphas = rhs *)
Clear[k];
pCeil = maxPoly;
While[True,
  B = {};
  For[d = 0, d <= maxDeriv, d++,
    actualMaxPoly = Min[maxPoly - d, pCeil];
    For[r = 0, r <= actualMaxPoly, r++,
      (* Taylor series expansion of basis function *)
      AppendTo[B, CoefficientList[k^r D[f[k], {k, d}], k][[1 ;; order]]];
  ];
  B = Transpose[B];
  rhs = ConstantArray[0, order];
  rhs[[1]] = 1;
  (* Least norm solution *)
  Quiet@Check[tmp = Transpose[B].LinearSolve[B.Transpose[B], rhs], Break[]];
  alphas = tmp;
  actualMaxPolySave = actualMaxPoly;
  pCeil = pCeil - 1;
  If[pCeil < 0, Break[]];
]
Text[Row[{"Found interpolant in " TraditionalForm[C^(n - 1 - (pCeil + 1))]]]]

```

Out[11]= Found interpolant in C^6

```

In[12]:= (* Pack the nonzero weights into tuples of
(weight, derivative taken, polynomial multiplied) *)
currentAlphas = {};
For[d = 0, d <= maxDeriv, d++,
  For[r = 0, r <= actualMaxPolySave, r++,
    If[Abs[alphas[[d (1 + actualMaxPolySave) + r + 1]]] > 0,
      AppendTo[currentAlphas, {alphas[[d (1 + actualMaxPolySave) + r + 1]], d, r}];
    ]
  ]
]
currentAlphas

```

Out[14]= $\left\{ \left\{ \frac{1581}{251}, 0, 0 \right\}, \left\{ \frac{4419}{1004}, 1, 1 \right\}, \left\{ \frac{1995}{251}, 2, 0 \right\} \right\}$

```
In[15]:= (* Create base B-spline in real space using recursive formula *)
M[x_, y_] = Piecewise[{{0, y ≤ 0}, {1, 0 < y ≤ 1}, {0, y > 1}}];
Mnew[x_, y_] = M[x, y];
For[d = 1, d < n, d++, Mold[x_, y_] = Mnew[x, y];
  Mnew[x_, y_] = Simplify[x Mold[x, y]/d + ((d + 1 - x)/d) Mold[x - 1, y - 1];]
```

```
In[18]:= regions = {};
numRegions = n;
For[ii = 0, ii < numRegions, ii++,
  AppendTo[regions, Expand[Mnew[x + (n/2), ii + (1/2)]]]]
```

```
In[21]:= (* Loop over currentAlphas to build basis
  element then sum to make the high order interpolant V *)
V = ConstantArray[0, {numRegions}];
numAlphas = Length[currentAlphas];
For[k = 1, k ≤ numAlphas, k++, alpha = currentAlphas[[k, 1]];
  j = currentAlphas[[k, 3]];
  i = currentAlphas[[k, 2]];
  For[r = 1, r ≤ numRegions, r++,
    basisElementRegion = D[x^i regions[[r]]/I^(i + j), {x, j}];
    V[[r]] += alpha * basisElementRegion;]]
coeffs = {};
For[kk = 1, kk ≤ numRegions, kk++,
  AppendTo[coeffs, Reverse[CoefficientList[V[[kk]], x]]]]
```

```
In[26]:= (* The piecewise polynomial *)
pieces = {};
For[kk = 1, kk ≤ numRegions, kk++,
  AppendTo[pieces, {Expand[V[[kk]]], -(n/2) + kk - 1 ≤ x ≤ -(n/2) + kk}]]
piecefunc[x_] = Piecewise[pieces];
piecefunc[x] // TraditionalForm
```

Out[29]//TraditionalForm=

$$\left\{ \begin{array}{ll} -\frac{19x^9}{12048} - \frac{133x^8}{3012} - \frac{225859x^7}{421680} - \frac{221003x^6}{60240} - \frac{23299x^5}{1506} - \frac{30793x^4}{753} - \frac{49184x^3}{753} - \frac{208208x^2}{3765} - \frac{53632x}{3765} + \frac{32512}{5271} & -4 \leq x \leq -3 \\ \frac{133x^9}{12048} + \frac{665x^8}{3012} + \frac{114139x^7}{60240} + \frac{109283x^6}{12048} + \frac{79303x^5}{3012} + \frac{283423x^4}{6024} + \frac{75215x^3}{1506} + \frac{170023x^2}{6024} + \frac{90923x}{15060} - \frac{17653}{42168} & -3 \leq x \leq -2 \\ -\frac{133x^9}{4016} - \frac{399x^8}{1004} - \frac{39659x^7}{20080} - \frac{104409x^6}{20080} - \frac{23443x^5}{3012} - \frac{14175x^4}{2008} - \frac{7553x^3}{1506} - \frac{32207x^2}{10040} - \frac{2933x}{15060} + \frac{13081}{14056} & -2 \leq x \leq -1 \\ \frac{665x^9}{12048} + \frac{665x^8}{3012} + \frac{2419x^7}{12048} - \frac{2437x^6}{12048} + \frac{2723x^4}{3012} - \frac{4543x^2}{3012} + \frac{19177}{21084} & -1 \leq x \leq 0 \\ -\frac{665x^9}{12048} + \frac{665x^8}{3012} - \frac{2419x^7}{12048} - \frac{2437x^6}{12048} + \frac{2723x^4}{3012} - \frac{4543x^2}{3012} + \frac{19177}{21084} & 0 \leq x \leq 1 \\ \frac{133x^9}{4016} - \frac{399x^8}{1004} + \frac{39659x^7}{20080} - \frac{104409x^6}{20080} + \frac{23443x^5}{3012} - \frac{14175x^4}{2008} + \frac{7553x^3}{1506} - \frac{32207x^2}{10040} + \frac{2933x}{15060} + \frac{13081}{14056} & 1 \leq x \leq 2 \\ -\frac{133x^9}{12048} + \frac{665x^8}{3012} - \frac{114139x^7}{60240} + \frac{109283x^6}{12048} - \frac{79303x^5}{3012} + \frac{283423x^4}{6024} - \frac{75215x^3}{1506} + \frac{170023x^2}{6024} - \frac{90923x}{15060} - \frac{17653}{42168} & 2 \leq x \leq 3 \\ \frac{19x^9}{12048} - \frac{133x^8}{3012} + \frac{225859x^7}{421680} - \frac{221003x^6}{60240} + \frac{23299x^5}{1506} - \frac{30793x^4}{753} + \frac{49184x^3}{753} - \frac{208208x^2}{3765} + \frac{53632x}{3765} + \frac{32512}{5271} & 3 \leq x \leq 4 \end{array} \right.$$

```
In[30]:= (* Coefficients for piecewise polynomial *)
```

```
coeffs // MatrixForm
```

```
Out[30]//MatrixForm=
```

$$\begin{pmatrix} -\frac{19}{12048} & -\frac{133}{3012} & -\frac{225859}{421680} & -\frac{221003}{60240} & -\frac{23299}{1506} & -\frac{30793}{753} & -\frac{49184}{753} & -\frac{208208}{3765} & -\frac{53632}{3765} & \frac{32512}{5271} \\ \frac{133}{12048} & \frac{665}{3012} & \frac{114139}{60240} & \frac{109283}{12048} & \frac{79303}{3012} & \frac{283423}{6024} & \frac{75215}{1506} & \frac{170023}{6024} & \frac{90923}{15060} & -\frac{17653}{42168} \\ -\frac{133}{4016} & -\frac{399}{1004} & -\frac{39659}{20080} & -\frac{104409}{20080} & -\frac{23443}{3012} & -\frac{14175}{2008} & -\frac{7553}{1506} & -\frac{32207}{10040} & -\frac{2933}{15060} & \frac{13081}{14056} \\ \frac{665}{12048} & \frac{665}{3012} & \frac{2419}{12048} & -\frac{2437}{12048} & 0 & \frac{2723}{3012} & 0 & -\frac{4543}{3012} & 0 & \frac{19177}{21084} \\ -\frac{665}{12048} & -\frac{665}{3012} & -\frac{2419}{12048} & -\frac{2437}{12048} & 0 & \frac{2723}{3012} & 0 & -\frac{4543}{3012} & 0 & \frac{19177}{21084} \\ \frac{133}{12048} & -\frac{399}{3012} & \frac{39659}{12048} & -\frac{104409}{12048} & \frac{23443}{3012} & -\frac{14175}{3012} & \frac{7553}{3012} & -\frac{32207}{3012} & \frac{2933}{3012} & \frac{13081}{21084} \\ \frac{4016}{12048} & \frac{1004}{3012} & \frac{20080}{12048} & \frac{20080}{12048} & \frac{3012}{3012} & \frac{2008}{3012} & \frac{1506}{3012} & \frac{10040}{3012} & \frac{15060}{3012} & \frac{14056}{21084} \\ -\frac{133}{4016} & \frac{665}{1004} & -\frac{114139}{20080} & \frac{109283}{20080} & -\frac{79303}{3012} & \frac{283423}{2008} & -\frac{75215}{1506} & \frac{170023}{10040} & -\frac{90923}{15060} & -\frac{17653}{14056} \\ \frac{133}{12048} & -\frac{399}{3012} & \frac{39659}{12048} & -\frac{104409}{12048} & \frac{23443}{3012} & -\frac{14175}{3012} & \frac{7553}{3012} & -\frac{32207}{3012} & \frac{2933}{3012} & \frac{13081}{21084} \\ \frac{4016}{12048} & \frac{1004}{3012} & \frac{20080}{12048} & \frac{20080}{12048} & \frac{3012}{3012} & \frac{2008}{3012} & \frac{1506}{3012} & \frac{10040}{3012} & \frac{15060}{3012} & \frac{14056}{21084} \\ -\frac{19}{12048} & -\frac{133}{3012} & -\frac{225859}{421680} & -\frac{221003}{60240} & -\frac{23299}{1506} & -\frac{30793}{753} & -\frac{49184}{753} & -\frac{208208}{3765} & -\frac{53632}{3765} & \frac{32512}{5271} \end{pmatrix}$$

```
In[31]:= (* Flattening makes it easy to insert into C++ *)
```

```
(* Only outputs the positive side of the interpolant since it is symmetric *)
```

```
Flatten[coeffs[[n/2 + 1 ;; n]]]
```

$$\begin{aligned} \text{Out[31]} = & \left\{ -\frac{665}{12048}, \frac{665}{3012}, -\frac{2419}{12048}, -\frac{2437}{12048}, 0, \frac{2723}{3012}, 0, -\frac{4543}{3012}, 0, \frac{19177}{21084}, \frac{133}{4016}, -\frac{399}{1004}, \frac{39659}{20080}, \right. \\ & -\frac{104409}{20080}, \frac{23443}{3012}, -\frac{14175}{2008}, \frac{7553}{1506}, -\frac{32207}{10040}, \frac{2933}{15060}, \frac{13081}{14056}, -\frac{133}{12048}, \frac{665}{3012}, \\ & -\frac{114139}{60240}, \frac{109283}{12048}, -\frac{79303}{3012}, \frac{283423}{6024}, -\frac{75215}{1506}, \frac{170023}{6024}, -\frac{90923}{15060}, -\frac{17653}{42168}, \frac{19}{12048}, \\ & -\frac{133}{3012}, \frac{225859}{421680}, -\frac{221003}{60240}, \frac{23299}{1506}, -\frac{30793}{753}, \frac{49184}{753}, -\frac{208208}{3765}, \frac{53632}{3765}, \left. \frac{32512}{5271} \right\} \end{aligned}$$

```
In[32]:= (* Plot the interpolant *)
```

```
halfSupp = support/2;
```

```
Plot[piecefunc[x], {x, -(halfSupp + 5), halfSupp + 5}, PlotRange -> All]
```

```
Out[33]=
```

```
(* End of notebook *)
```

```

1  /*
2   This is the C++ code to generate the Fourier transformed versions of the convolution kernels, G and H
3  */
4
5  #include "NodeFArrayBox.H"
6  #include "Interval.H"
7  #include "fftw3.h"
8  #include <cmath>
9
10 #define MAXWELL_USE_LONG 1 // comment out if don't want to use quad precision
11 #ifdef MAXWELL_USE_LONG
12     typedef long double kernelDataType;
13     typedef BaseFab<long double> kernelStorageType;
14     #define M_PI 3.141592653589793238462643383279502884L
15     #define gleps 1.e-18
16 #else
17     typedef Real kernelDataType;
18     typedef NodeFArrayBox kernelStorageType;
19     #define gleps 1.e-14
20 #endif
21
22 // used to convert quad to double precision
23 #if MAXWELL_USE_LONG
24     #include <fenv.h>
25     #pragma STDC FENV_ACCESS_ON
26 #endif
27
28 // some code to compute first derivatives
29 void firstDerivative(kernelStorageType& a_input,
30                     kernelStorageType& a_output,
31                     int dim)
32 {}
33
34 /*
35  Compute N Gauss-Legendre weights and points on the interval [-1,1],
36  same algorithm as lgwt.m by Greg von Winckel - 02/25/2004 in matlab
37 */
38 void glpts(Vector<kernelDataType>& a_weights,
39            Vector<kernelDataType>& a_points,
40            int a_Ntheta)
41 {
42     int N = a_Ntheta - 1;
43     int N1 = N + 1;
44     int N2 = N + 2;
45
46     kernelDataType* xu = new kernelDataType [N1];
47     kernelDataType dx = 2.0 / N;
48     xu[0] = -1;
49     xu[N1 - 1] = 1;
50     for (int j = 1; j < N1 - 1; j++)
51     {
52         xu[j] = xu[j - 1] + dx;
53     }
54
55     kernelDataType* y = new kernelDataType [N1];
56     for (int j = 0; j < N1; j++)
57     {
58         #ifdef MAXWELL_USE_LONG
59             y[j] = cos((2 * j + 1) * M_PI / (2 * N + 2)) + (0.27 / N1) * sin(M_PI * xu[j] * N / N2);
60         #else
61             y[j] = cos((2 * j + 1) * M_PI / (2 * N + 2)) + (0.27 / N1) * sin(M_PI * xu[j] * N / N2);
62         #endif
63     }
64
65     kernelDataType* L = new kernelDataType [N1 * N2];
66     kernelDataType* Lp = new kernelDataType [N1];
67     for (int j = 0; j < N1 * N2; j++)
68     {
69         L[j] = 0;
70     }
71
72     kernelDataType* y0 = new kernelDataType [N1];
73     kernelDataType eps = gleps;
74     kernelDataType error = 1;
75     while (error > eps)
76     {
77         for (int j = 0; j < N1; j++)
78         {
79             L[j * N2] = 1;
80             L[j * N2 + 1] = y[j];
81         }
82         for (int k = 2; k < N2; k++)
83         {
84             for (int j = 0; j < N1; j++)
85             {
86                 L[j * N2 + k] = ( (2 * k - 1) * y[j] * L[j * N2 + k - 1] - (k - 1) * L[j * N2 + k - 2] ) / k;
87             }
88         }
89         for (int j = 0; j < N1; j++)
90         {
91             Lp[j] = N2 * ( L[j * N2 + N1 - 1] - y[j] * L[j * N2 + N2 - 1] ) / (1 - y[j] * y[j]);

```

```

92     y0[j] = y[j];
93     y[j] = y0[j] - L[j * N2 + N2 - 1] / Lp[j];
94 }
95
96 error = 0;
97 for (int j = 0; j < N1; j++)
98 {
99     error = std::max(error, std::abs(y[j] - y0[j]));
100 }
101 }
102
103 for (int j = 0; j < N1; j++)
104 {
105     a_points[j] = (-1 * (1 - y[j]) + (1 + y[j])) / 2;
106     a_weights[j] = 2 / ((1 - y[j] * y[j]) * Lp[j] * Lp[j]) * ((kernelDataType)N2 / N1) * ((kernelDataType)N2 / N1);
107 }
108 delete [] L;
109 delete [] Lp;
110 delete [] y0;
111 delete [] y;
112 delete [] xu;
113 }
114
115 /*
116  Evaluate the 1D delta of the appropriate order at point a_x
117  Inputs: a_x: the point to evaluate at
118  a_h: grid spacing
119  a_delta: The coefficients of the delta at positive side,
120  given by the Flatten line in bsplineMathematica.nb
121  a_nint: support of interpolant divided by two
122  a_pieceOrder: order of polynomials + 1
123 */
124 kernelDataType evalDelta(kernelDataType a_x,
125                          kernelDataType a_h,
126                          Vector<kernelDataType>& a_delta,
127                          int a_nint,
128                          int a_pieceOrder)
129 {
130     a_x = abs(a_x) / a_h;
131     if (a_x >= a_nint)
132     {
133         return 0;
134     }
135     else
136     {
137         kernelDataType solution = 0.;
138         int idx = floor(a_x);
139         int s = a_delta.size() / a_nint - 1;
140         for (int j = 0; j < a_pieceOrder; j++)
141         {
142             solution += a_delta[j + idx * a_pieceOrder] * pow(a_x, s - j);
143         }
144         return solution / a_h;
145     }
146 }
147
148 /*
149  Builds the kernel and puts it in a_kernel
150  Inputs: a_kernel
151  a_stepSize: time step
152  a_h: grid spacing
153  a_c: speed of light
154  a_Ntheta: number of theta points in Atkinson quadrature
155  The radius of the sphere is R = m_stepSize * m_c
156 */
157 void buildKernel(kernelStorageType& a_kernel,
158                 Box& a_domain,
159                 kernelDataType a_stepSize,
160                 kernelDataType a_h,
161                 kernelDataType a_c,
162                 int a_Ntheta)
163 {
164     // variables for the 1d delta
165     // coefficients for a 6th order C^0 delta
166     kernelDataType coeffDel[] = { -(1. / 12), 1. / 4, 5. / 12, -(5. / 4), -(1. / 3), 1.0, 1. / 24, -(3. / 8), 25. / 24, -(5. /
167                                     8), -(13. / 12), 1.0, -(1. / 120), 1. / 8, -(17. / 24), 15. / 8, -(137. / 60), 1.0
168     };
169     Vector<kernelDataType> delta;
170     delta.assign(coeffDel, coeffDel + 18);
171     // how many intervals are there in the positive side (support/2)
172     int nint = 3;
173     int support = 2 * nint;
174     // number of terms in polynomials (order of polynomials + 1)
175     int pieceOrder = 6;
176
177     // variables for spherical quadrature
178     Vector<kernelDataType> weights(a_Ntheta), mu(a_Ntheta), thetas(a_Ntheta);
179     kernelDataType phi, quadX, quadY, quadZ, evalX, evalY, evalZ, theta, temp;
180
181     // prefactor from Atkinson quadrature
182     #ifndef MAXWELL_USE_LONG
183     long double factor = M_PI1 / m_Ntheta * R * R;

```



```

184 #else
185 Real factor = M_PI / m_Ntheta * R * R;
186 #endif
187
188 // Grab the GL points and weights and compute acos of all the mu (Atkinson quadrature)
189 glpts(weights, mu, a_Ntheta);
190 for (int j = 0; j < a_Ntheta; j++)
191 {
192     thetas[j] = acos(mu[j]);
193 }
194
195 // spherical integration loop
196 for (int j = 0; j < 2 * a_Ntheta; j++)
197 {
198     // use a_Ntheta points in theta direction
199     for (int k = 0; k < a_Ntheta; k++)
200     {
201 #ifdef MAXWELL_USE_LONG
202     phi = j * M_PI / m_Ntheta;
203 #else
204     phi = j * M_PI / m_Ntheta;
205 #endif
206     theta = thetas[k];
207     quadX = R * cos(phi) * sin(theta);
208     quadY = R * sin(phi) * sin(theta);
209     quadZ = R * cos(theta);
210     // Here we try to restrict to just the grid points that will be nonzero for speed
211     // (box must be defined with no offset to physical space)
212     // the +3 is just in case
213     int x_hi = (int) (quadX / a_h + (support + 3) / 2);
214     int y_hi = (int) (quadY / a_h + (support + 3) / 2);
215     int z_hi = (int) (quadZ / a_h + (support + 3) / 2);
216     int x_lo = (int) (quadX / a_h - (support + 3) / 2);
217     int y_lo = (int) (quadY / a_h - (support + 3) / 2);
218     int z_lo = (int) (quadZ / a_h - (support + 3) / 2);
219     IntVect lo(x_lo, y_lo, z_lo), hi(x_hi, y_hi, z_hi);
220     Box smallBox(lo, hi);
221     BoxIterator boxIt(smallBox);
222
223     // Calculate impact of this quadrature point on the small box
224     for (boxIt.reset(); boxIt.ok(); ++boxIt)
225     {
226         evalX = m_h[0] * boxIt()[0];
227         evalY = m_h[1] * boxIt()[1];
228         evalZ = m_h[2] * boxIt()[2];
229         temp = factor * weights[k] *
230             evalDelta(quadX - evalX, a_h, delta, nint, pieceOrder) *
231             evalDelta(quadY - evalY, a_h, delta, nint, pieceOrder) *
232             evalDelta(quadZ - evalZ, a_h, delta, nint, pieceOrder);
233         // G
234         a_kernel(boxIt(), 0) += temp;
235         // G_i
236         a_kernel(boxIt(), 2) += temp * quadX;
237         a_kernel(boxIt(), 3) += temp * quadY;
238         a_kernel(boxIt(), 4) += temp * quadZ;
239     }
240 }
241 }
242
243 // compute G_i d/dx_i
244 kernelStorageType derivHelperFab(a_kernel.box(), 1);
245 for (int j = 0; j < 3; j++)
246 {
247     kernelStorageType aliasG(Interval(j + 2, j + 2), a_kernel);
248     firstDerivative(aliasG, derivHelperFab, j);
249     aliasG.copy(derivHelperFab);
250 }
251
252 // Construct H = G + sum_i G_i d/dx_i
253 // put it in second component of a_kernel
254 kernelStorageType aliasG(Interval(0, 0), a_kernel);
255 kernelStorageType aliasH(Interval(1, 1), a_kernel);
256 kernelStorageType aliasX(Interval(2, 2), a_kernel);
257 kernelStorageType aliasY(Interval(3, 3), a_kernel);
258 kernelStorageType aliasZ(Interval(4, 4), a_kernel);
259 aliasH.copy(aliasG);
260
261 #ifdef MAXWELL_USE_LONG
262 BoxIterator domainBoxIt(a_domain);
263 for (domainBoxIt.reset(); domainBoxIt.ok(); ++domainBoxIt)
264 {
265     aliasH(domainBoxIt(), 0) -= (aliasX(domainBoxIt(), 0) + aliasY(domainBoxIt(), 0) + aliasZ(domainBoxIt(), 0));
266     aliasH(domainBoxIt(), 0) /= (4. * M_PI * R * R);
267     aliasG(domainBoxIt(), 0) /= (4. * M_PI * R);
268 }
269 #else
270 aliasH -= aliasX;
271 aliasH -= aliasY;
272 aliasH -= aliasZ;
273
274 aliasG /= (4. * M_PI * R);
275 aliasH /= (4. * M_PI * R * R);

```

```

276 #endif
277 }
278
279 /* fourier transform the kernels
280 Inputs: a_kernel: the kernels in real space, stored in first two components
281 a_ftKernels: the fourier transform of the kernels
282 a_support: the support of the kernel
283 */
284 void fourierTransformKernels(kernelStorageType& a_kernel,
285                             Vector<fftw_complex*>& a_ftKernel,
286                             int a_support)
287 {
288 #ifdef MAXWELL_USE_LONG
289 // rounds quads to nearest double when casted down
290 fesetround(FE_TONEAREST);
291 #endif
292
293 int ncomp = 3;
294 vector<fftw_complex*>* fftKer = &(m_G_freq);
295
296 Box dDomaink = a_kernel.box();
297 IntVect size = dDomaink.size();
298 int Nx = size[0];
299 int Ny = size[1];
300 int Nz = size[2];
301 int dNx = Nx + a_support;
302 int dNy = dNx;
303 int dNz = dNx;
304
305 // used to shift the fourier transformed kernels
306 int shift = -(dDomaink.smallEnd()[0] - 1);
307
308 // create a domain, extended by support of kernel
309 IntVect extended = dDomaink.smallEnd();
310 IntVect growth(a_support, a_support, a_support);
311 extended += growth;
312 dDomaink.setSmall(extended);
313
314 // FFTW for real-to-complex transforms, the output is truncated on the fastest traversed index
315 // The memory in Chombo is stored so that the first index (x) is the fastest traversed index
316 int Nf = dNy * (dNx / 2 + 1);
317 for (int kerIdx = 0; kerIdx < 2; kerIdx++)
318 {
319 // embed the kernel into an array with this extended domain
320 kernelStorageType doubKer(dDomaink, 1);
321 doubKer.setVal(0.0);
322 kernelStorageType alias(Interval(kerIdx, kerIdx), a_kernel);
323 doubKer.copy(alias);
324 kernelDataType* kerPtr = doubKer.dataPtr(0);
325
326 // set up FFTW plans and fourier transform the kernel
327 #ifdef MAXWELL_USE_LONG
328 kernelDataType* dummydouble = fftwl_alloc_real(dNx * dNy * dNz);
329 fftw_complex* fourierKernel = fftwl_alloc_complex(Nf);
330 fftw_plan tmpPlan = fftw_plan_dft_r2c_2d(dNz, dNy, dNx, fourierKernel, FFTW_ESTIMATE);
331 fftwl_execute_dft_r2c(tmpPlan, kerPtr, fourierKernel);
332 #else
333 kernelDataType* kerPtr = doubKer.dataPtr(0);
334 kernelDataType* dummydouble = fftw_alloc_real(dNx * dNy * dNz);
335 fftw_complex* fourierKernel = fftw_alloc_complex(Nf);
336 fftw_plan tmpPlan = fftw_plan_dft_r2c_2d(dNz, dNy, dNx, fourierKernel, FFTW_ESTIMATE);
337 fftw_free(dummydouble);
338 fftw_execute_dft_r2c(tmpPlan, kerPtr, fourierKernel);
339 #endif
340
341 // allocate memory for output
342 a_ftKernel[kerIdx] = fftw_alloc_complex(Nf);
343
344 // need shift the kernels in fourier space appropriately because the kernel in real space is centered at middle of array
345 int idx = 0;
346 kernelDataType arg, sarg, carg, imag, real;
347 for (int j = 0; j < dNy; j++)
348 {
349 for (int i = 0; i < dNx / 2 + 1; i++)
350 {
351 #ifdef MAXWELL_USE_LONG
352 arg = 2.*M_PI * (i + j) * m_shift / dNx;
353 #else
354 arg = 2.*M_PI * (i + j) * m_shift / dNx;
355 #endif
356 sarg = -sin(arg);
357 carg = cos(arg);
358 real = fourierKernel[idx][0];
359 imag = fourierKernel[idx][1];
360 (a_ftKernel[kerIdx])[idx][0] = (Real) (real * carg - imag * sarg);
361 (a_ftKernel[kerIdx])[idx][1] = (Real) (real * sarg + imag * carg);
362 idx++;
363 }
364 }
365
366 #ifdef MAXWELL_USE_LONG
367 fftwl_destroy_plan(tmpPlan);

```

```

368     fftw_free(fourierKernel);
369 #else
370     fftw_destroy_plan(tmpPlan);
371     fftw_free(fourierKernel);
372 #endif
373 }
374 }
375
376 // example making the kernels
377 int main()
378 {
379     int N                = 32;
380     int Ntheta           = 24;
381     kernelDataType stepSize = 1.;
382     kernelDataType h      = 1.;
383     kernelDataType c      = 1.;
384
385     // create the domain, box needs to be from -N/2 to N/2
386     IntVect lo = (-N / 2) * IntVect::Unit;
387     IntVect hi = (N / 2) * IntVect::Unit;
388     Box domain(lo, hi);
389
390     // make the storage for the kernels
391     kernelStorageType kernel;
392 #ifdef MAXWELL_USE_LONG
393     Box nodeDomain = domain;
394     nodeDomain.surroundingNodes();
395     kernel.resize(nodeDomain, 5);
396 #else
397     kernel.resize(domain, 5);
398 #endif
399     kernel.setVal(0.0);
400
401     // build the kernels
402     // G is in component 0 and H in component 1
403     build_kernel(kernel,
404                 domain,
405                 stepSize,
406                 h,
407                 c,
408                 Ntheta);
409
410     // fourier transform kernels
411     Vector<fftw_complex*> ftKernel(2);
412     int kernelSupport = 2 * (ceil((c * stepSize / h) + support));
413     fourierTransformKernels(kernel,
414                             ftKernel,
415                             kernelSupport);
416
417     return 0;
418 }

```

Appendix C

High Order B -Splines

For completeness here are the q -th order C^p cardinal B-splines, $W_{q,p}$, that were used.

$$W_{4,0}(x) = \begin{cases} \frac{|x|^3}{2} - |x|^2 - \frac{|x|}{2} + 1 & : |x| \in [0, 1] \\ -\frac{|x|^3}{6} + |x|^2 - \frac{11|x|}{6} + 1 & : |x| \in [1, 2] \\ 0, & |x| > 2 \end{cases} \quad (\text{C.1})$$

$$W_{6,0}(x) = \begin{cases} -\frac{|x|^5}{12} + \frac{|x|^4}{4} + \frac{5|x|^3}{12} - \frac{5|x|^2}{4} - \frac{|x|}{3} + 1 & : |x| \in [0, 1] \\ \frac{|x|^5}{24} - \frac{3|x|^4}{8} + \frac{25|x|^3}{24} - \frac{5|x|^2}{8} - \frac{13|x|}{12} + 1 & : |x| \in [1, 2] \\ -\frac{|x|^5}{120} + \frac{|x|^4}{8} - \frac{17|x|^3}{24} + \frac{15|x|^2}{8} - \frac{137|x|}{60} + 1 & : |x| \in [2, 3] \\ 0, & |x| > 3 \end{cases} \quad (\text{C.2})$$

$$W_{8,0}(x) = \begin{cases} \frac{|x|^7}{144} - \frac{|x|^6}{36} - \frac{7|x|^5}{72} + \frac{+7|x|^4}{18} + \frac{+49|x|^3}{144} - \frac{49|x|^2}{36} - \frac{|x|}{4} + 1 & : |x| \in [0, 1] \\ -\frac{|x|^7}{240} + \frac{+|x|^6}{20} - \frac{7|x|^5}{40} + \frac{+77|x|^3}{80} - \frac{21|x|^2}{20} - \frac{47|x|}{60} + 1 & : |x| \in [1, 2] \\ \frac{|x|^7}{720} - \frac{|x|^6}{36} + \frac{+77|x|^5}{360} - \frac{7|x|^4}{9} + \frac{+889|x|^3}{720} - \frac{7|x|^2}{36} - \frac{29|x|}{20} + 1 & : |x| \in [2, 3] \\ -\frac{|x|^7}{5040} + \frac{+|x|^6}{180} - \frac{23|x|^5}{360} + \frac{+7|x|^4}{18} - \frac{967|x|^3}{720} + \frac{+469|x|^2}{180} - \frac{363|x|}{140} + 1 & : |x| \in [3, 4] \\ 0 & : |x| > 4 \end{cases} \quad (\text{C.3})$$

$$\begin{aligned}
W_{16,0}(x) = & \left\{ \begin{aligned}
& \frac{|x|^{15}}{203212800} - \frac{|x|^{14}}{25401600} - \frac{|x|^{13}}{1451520} + \frac{|x|^{12}}{181440} + \frac{533|x|^{11}}{14515200} - \frac{533|x|^{10}}{1814400} \\
& \quad - \frac{9581|x|^9}{10160640} + \frac{9581|x|^8}{1270080} + \frac{353639|x|^7}{29030400} - \frac{353639|x|^6}{3628800} \\
& \quad - \frac{54613|x|^5}{725760} + \frac{54613|x|^4}{90720} + \frac{266681|x|^3}{1411200} - \frac{266681|x|^2}{176400} - \frac{|x|}{8} + 1 : |x| \in [0, 1] \\
& - \frac{|x|^{15}}{261273600} + \frac{|x|^{14}}{10886400} - \frac{|x|^{13}}{2612736} - \frac{|x|^{12}}{155520} + \frac{1027|x|^{11}}{18662400} + \frac{13|x|^{10}}{129600} \\
& \quad - \frac{5291|x|^9}{2612736} + \frac{1859|x|^8}{1088640} + \frac{8198047|x|^7}{261273600} - \frac{91091|x|^6}{1555200} \\
& - \frac{39871|x|^5}{186624} + \frac{6487|x|^4}{12960} + \frac{1021879|x|^3}{1814400} - \frac{36383|x|^2}{25200} - \frac{191|x|}{504} + 1 : |x| \in [1, 2] \\
& \quad \frac{|x|^{15}}{435456000} - \frac{|x|^{14}}{10886400} + \frac{29|x|^{13}}{21772800} - \frac{|x|^{12}}{155520} - \frac{1027|x|^{11}}{31104000} \\
& \quad + \frac{91|x|^{10}}{194400} - \frac{21593|x|^9}{21772800} - \frac{1573|x|^8}{217728} + \frac{15139553|x|^7}{435456000} + \frac{23023|x|^6}{1555200} \\
& - \frac{485329|x|^5}{1555200} + \frac{2821|x|^4}{9720} + \frac{8382163|x|^3}{9072000} - \frac{98149|x|^2}{75600} - \frac{1627|x|}{2520} + 1 : |x| \in [2, 3] \\
& - \frac{|x|^{15}}{958003200} + \frac{|x|^{14}}{17107200} - \frac{13|x|^{13}}{9580032} + \frac{7|x|^{12}}{427680} - \frac{6773|x|^{11}}{68428800} + \frac{91|x|^{10}}{777600} \\
& \quad + \frac{9607|x|^9}{4354560} - \frac{247|x|^8}{19440} + \frac{1010477|x|^7}{87091200} + \frac{169663|x|^6}{1555200} \\
& - \frac{1127711|x|^5}{3421440} - \frac{1183|x|^4}{26730} + \frac{24996637|x|^3}{19958400} - \frac{125009|x|^2}{118800} - \frac{25961|x|}{27720} + 1 : |x| \in [3, 4] \\
& \quad \frac{|x|^{15}}{2874009600} - \frac{|x|^{14}}{39916800} + \frac{113|x|^{13}}{143700480} - \frac{|x|^{12}}{71280} + \frac{31733|x|^{11}}{205286400} - \frac{91|x|^{10}}{86400} \\
& \quad + \frac{10309|x|^9}{2612736} - \frac{247|x|^8}{90720} - \frac{11443757|x|^7}{261273600} + \frac{96031|x|^6}{518400} \\
& - \frac{1998841|x|^5}{10264320} - \frac{12467|x|^4}{23760} + \frac{30023681|x|^3}{19958400} - \frac{182051|x|^2}{277200} - \frac{35201|x|}{27720} + 1 : |x| \in [4, 5] \\
& - \frac{|x|^{15}}{12454041600} + \frac{|x|^{14}}{141523200} - \frac{173|x|^{13}}{622702080} + \frac{|x|^{12}}{155520} - \frac{6641|x|^{11}}{68428800} + \frac{97|x|^{10}}{97200} \\
& \quad - \frac{30761|x|^9}{4354560} + \frac{36641|x|^8}{1088640} - \frac{8624911|x|^7}{87091200} + \frac{191213|x|^6}{1555200} \\
& \quad + \frac{158633|x|^5}{684288} - \frac{22523|x|^4}{19440} + \frac{403330957|x|^3}{259459200} + \frac{961|x|^2}{982800} - \frac{605453|x|}{360360} + 1 : |x| \in [5, 6] \\
& \quad \frac{|x|^{15}}{87178291200} - \frac{|x|^{14}}{838252800} + \frac{|x|^{13}}{17791488} - \frac{19|x|^{12}}{11975040} + \frac{14321|x|^{11}}{479001600} - \frac{1079|x|^{10}}{2721600} \\
& \quad + \frac{115769|x|^9}{30481920} - \frac{202189|x|^8}{7620480} + \frac{11716153|x|^7}{87091200} - \frac{5276479|x|^6}{10886400} \\
& + \frac{28312703|x|^5}{23950080} - \frac{5194631|x|^4}{2993760} + \frac{1690990363|x|^3}{1816214400} + \frac{7257689|x|^2}{5821200} - \frac{811373|x|}{360360} + 1 : |x| \in [6, 7] \\
& - \frac{|x|^{15}}{1307674368000} + \frac{|x|^{14}}{10897286400} - \frac{47|x|^{13}}{9340531200} + \frac{|x|^{12}}{5987520} - \frac{26921|x|^{11}}{7185024000} \\
& \quad + \frac{109|x|^{10}}{1814400} - \frac{324509|x|^9}{457228800} + \frac{4783|x|^8}{762048} - \frac{54576553|x|^7}{1306368000} + \frac{2271089|x|^6}{10886400} \\
& - \frac{277382447|x|^5}{359251200} + \frac{2065639|x|^4}{997920} - \frac{35118025721|x|^3}{9081072000} + \frac{13215487|x|^2}{2802800} - \frac{1195757|x|}{360360} + 1 : |x| \in [7, 8] \\
& 0 : |x| > 8
\end{aligned} \right.
\end{aligned}$$

(C.4)

$$W_{6,6}(x) = \left\{ \begin{array}{ll} -\frac{665|x|^9}{12048} + \frac{665|x|^8}{3012} - \frac{2419|x|^7}{12048} - \frac{2437|x|^6}{12048} + \frac{2723|x|^4}{3012} - \frac{4543|x|^2}{3012} + \frac{19177}{21084} & : |x| \in [0, 1] \\ \frac{133|x|^9}{4016} - \frac{399|x|^8}{1004} + \frac{39659|x|^7}{20080} - \frac{104409|x|^6}{20080} \\ + \frac{23443|x|^5}{3012} - \frac{14175|x|^4}{2008} + \frac{7553|x|^3}{1506} - \frac{32207|x|^2}{10040} + \frac{2933|x|}{15060} + \frac{13081}{14056} & : |x| \in [1, 2] \\ -\frac{133|x|^9}{12048} + \frac{665|x|^8}{3012} - \frac{114139|x|^7}{60240} + \frac{109283|x|^6}{12048} \\ - \frac{79303|x|^5}{3012} + \frac{283423|x|^4}{6024} - \frac{75215|x|^3}{1506} + \frac{170023|x|^2}{6024} - \frac{90923|x|}{15060} - \frac{17653}{42168} & : |x| \in [2, 3] \\ \frac{19|x|^9}{12048} - \frac{133|x|^8}{3012} + \frac{225859|x|^7}{421680} - \frac{221003|x|^6}{60240} \\ + \frac{23299|x|^5}{1506} - \frac{30793|x|^4}{753} + \frac{49184|x|^3}{753} - \frac{208208|x|^2}{3765} + \frac{53632|x|}{3765} + \frac{32512}{5271} & : |x| \in [3, 4] \\ 0 & : |x| > 4 \end{array} \right. \quad (C.5)$$

$$W_{4,7}(x) = \left\{ \begin{array}{ll} -\frac{2614045|x|^{11}}{1231471392} + \frac{2614045|x|^{10}}{307867848} + \frac{1246375|x|^9}{615735696} - \frac{2158155|x|^8}{51311308} \\ + \frac{10201175|x|^7}{1231471392} + \frac{10379455|x|^6}{102622616} - \frac{1425802|x|^4}{12827827} - \frac{138943625|x|^2}{538768734} + \frac{308075485}{538768734} & : |x| \in [0, 1] \\ \frac{522809|x|^{11}}{410490464} - \frac{1568427|x|^{10}}{102622616} + \frac{14389377|x|^9}{205245232} - \frac{3547155|x|^8}{25655654} \\ + \frac{56509415|x|^7}{1231471392} + \frac{24214059|x|^6}{102622616} - \frac{9439087|x|^5}{25655654} + \frac{10707557|x|^4}{25655654} \\ - \frac{24966025|x|^3}{51311308} + \frac{8505715|x|^2}{359179156} - \frac{14281645|x|}{153933924} + \frac{210144205}{359179156} & : |x| \in [1, 2] \\ -\frac{522809|x|^{11}}{1231471392} + \frac{2614045|x|^{10}}{307867848} - \frac{43666681|x|^9}{615735696} + \frac{4035040|x|^8}{12827827} \\ - \frac{938909005|x|^7}{1231471392} + \frac{85270845|x|^6}{102622616} + \frac{4295011|x|^5}{12827827} - \frac{59240097|x|^4}{25655654} \\ + \frac{202680975|x|^3}{51311308} - \frac{4996465975|x|^2}{1077537468} + \frac{442730995|x|}{153933924} - \frac{283592665}{1077537468} & : |x| \in [2, 3] \\ \frac{74687|x|^{11}}{1231471392} - \frac{522809|x|^{10}}{307867848} + \frac{87582637|x|^9}{4310149872} - \frac{6887695|x|^8}{51311308} \\ + \frac{1504938535|x|^7}{2873433248} - \frac{119864359|x|^6}{102622616} + \frac{29278133|x|^5}{25655654} + \frac{10007802|x|^4}{12827827} \\ - \frac{48474600|x|^3}{12827827} + \frac{1626295520|x|^2}{269384367} - \frac{261150080|x|}{38483481} + \frac{1044600320}{269384367} & : |x| \in [3, 4] \\ 0 & : |x| > 4 \end{array} \right. \quad (C.6)$$

Bibliography

- [1] 254A, notes 3a: *Eigenvalues and sums of Hermitian matrices*. <https://terrytao.wordpress.com/2010/01/12/254a-notes-3a-eigenvalues-and-sums-of-hermitian-matrices/>. Accessed: 2019-01-03.
- [2] *d'Alembert's formula*. https://en.wikipedia.org/wiki/D%27Alembert%27s_formula. Accessed: 2018-12-23.
- [3] *Wave equation in higher dimensions*. <https://web.stanford.edu/class/math220a/handouts/waveequation3.pdf>. Accessed: 2018-12-23.
- [4] G. Ala, E. Francomano, A. Tortorici, E. Toscano, and F. Viola, *Smoothed particle electromagnetics: A mesh-free solver for transients*, Journal of Computational and Applied Mathematics, 191 (2006), pp. 194–205.
- [5] B. Alpert, L. Greengard, and T. Hagstrom, *An integral evolution formula for the wave equation*, Journal of Computational Physics, 162 (2000), pp. 536 – 543.
- [6] C. R. Anderson, *A method of local corrections for computing the velocity field due to a distribution of vortex blobs*, Journal of Computational Physics, 62 (1986), pp. 111–123.
- [7] U. Andersson, *Time-domain methods for the Maxwell equations*, PhD thesis, Numerisk analys och datalogi, 2001.
- [8] K. Atkinson, *Numerical integration on the sphere*, The Journal of the Australian Mathematical Society. Series B. Applied Mathematics, 23 (1982), pp. 332–347.
- [9] G. Bao, G. Wei, and S. Zhao, *Local spectral time-domain method for electromagnetic wave propagation*, Optics letters, 28 (2003), pp. 513–515.
- [10] A. Barbas and P. Velarde, *Development of a Godunov method for Maxwell's equations with adaptive mesh refinement*, Journal of Computational Physics, 300 (2015), pp. 186–201.
- [11] G. Barton, *Elements of Green's functions and propagation: potentials, diffusion, and waves*, Oxford University Press, 1989.

- [12] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, Journal of computational physics, 114 (1994), pp. 185–200.
- [13] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, Journal of computational Physics, 82 (1989), pp. 64–84.
- [14] R. Beyer and R. LeVeque, *Analysis of a one-dimensional model for the immersed boundary method*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 332–364.
- [15] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*, CRC press, 2004.
- [16] T. Blu, P. Thevenaz, and M. Unser, *MOMS: Maximal-order interpolation of minimal support*, IEEE Transactions on Image Processing, 10 (2001), pp. 1069–1080.
- [17] A. Bossavit, *Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism*, IEE Proceedings A-Physical Science, Measurement and Instrumentation, Management and Education-Reviews, 135 (1988), pp. 493–500.
- [18] A. Chaniotis and D. Poulikakos, *High order interpolation and differentiation using B-splines*, Journal of Computational Physics, 197 (2004), pp. 253–274.
- [19] W. C. Chew, M. S. Tong, and B. Hu, *Integral equation methods for electromagnetic and elastic waves*, Synthesis Lectures on Computational Electromagnetics, 3 (2008), pp. 1–241.
- [20] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, *VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data*, in High Performance Visualization—Enabling Extreme-Scale Scientific Insight, Oct 2012, pp. 357–372.
- [21] S. Chilton, *A fourth-order adaptive mesh refinement solver for Maxwell’s Equations*, PhD thesis, UC Berkeley, 2013.
- [22] V. Cingoski, N. Miyamoto, and H. Yamashita, *Element-free Galerkin method for electromagnetic field computations*, IEEE Transactions on Magnetism, 34 (1998), pp. 3236–3239.
- [23] B. Cockburn, F. Li, and C.-W. Shu, *Locally divergence-free discontinuous Galerkin methods for the Maxwell equations*, Journal of Computational Physics, 194 (2004), pp. 588–610.
- [24] P. Colella, D. T. Graves, J. N. Johnson, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. V. Straalen, *Chombo Software Package for AMR Applications - Design Document*. Lawrence Berkeley National Laboratory Technical Report LBNL-6616E, 2011.

- [25] P. Colella and P. C. Norgaard, *Controlling self-force errors at refinement boundaries for AMR-PIC*, Journal of Computational Physics, 229 (2010), pp. 947–957.
- [26] G. Cottet and P. Koumoutsakos, *Vortex Methods: Theory and Practice*, Cambridge University Press, 2000.
- [27] S. M. Cox and P. C. Matthews, *Exponential time differencing for stiff systems*, Journal of Computational Physics, 176 (2002), pp. 430–455.
- [28] R. K. Crockett, P. Colella, R. T. Fisher, R. I. Klein, and C. F. McKee, *An unsplit, cell-centered godunov method for ideal MHD*, Journal of Computational Physics, 203 (2005), pp. 422–448.
- [29] M. Dehghan and R. Salehi, *A meshless local Petrov–Galerkin method for the time-dependent Maxwell equations*, Journal of Computational and Applied Mathematics, 268 (2014), pp. 93–110.
- [30] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, et al., *Smilei: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation*, Computer Physics Communications, 222 (2018), pp. 351–373.
- [31] D. Doerfler, B. Austin, B. Cook, J. Deslippe, K. Kandalla, and P. Mendygral, *Evaluating the networking characteristics of the Cray XC-40 intel knights landing-based cori supercomputer at NERSC*, Concurrency and Computation: Practice and Experience, 30 (2018), p. e4297.
- [32] F. Edelvik and G. Ledfelt, *A comparison of time-domain hybrid solvers for complex scattering problems*, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 15 (2002), pp. 475–487.
- [33] T. Z. Esirkepov, *Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor*, Computer Physics Communications, 135 (2001), pp. 144–153.
- [34] A. B. Espa and D. P. V. Mayol, *Development of AMREM: a Maxwell’s equations solver with adaptive mesh refinement*, PhD thesis, Universidad Politécnica de Madrid, 2015.
- [35] L. C. Evans, *Partial Differential Equations*, American Mathematical Society, second ed., 2010.
- [36] J. Fang, *Time domain finite difference computation for Maxwell’s equations*, PhD thesis, UC Berkeley, 1989.
- [37] M. Frigo and S. G. Johnson, *FFTW: An adaptive software architecture for the FFT*, IEEE, 1998, pp. 1381–1384.
- [38] D. Givoli, *Non-reflecting boundary conditions*, Journal of computational physics, 94 (1991), pp. 1–29.

- [39] ———, *High-order local non-reflecting boundary conditions: a review*, Wave motion, 39 (2004), pp. 319–326.
- [40] S. Godunov, *Symmetric form of the equations of magnetohydrodynamics*, Numerical Methods for Mechanics of Continuum Medium, 1 (1972), pp. 26–34.
- [41] R. D. Graglia, D. R. Wilton, and A. F. Peterson, *Higher order interpolatory vector bases for computational electromagnetics*, IEEE transactions on antennas and propagation, 45 (1997), pp. 329–342.
- [42] G. Green, *An essay on the application of mathematical analysis to the theories of electricity and magnetism*, Wezäta-Melins Aktiebolag, 1828.
- [43] D. J. Griffiths and R. College, *Introduction to electrodynamics*, vol. 3, prentice Hall Upper Saddle River, NJ, 1999.
- [44] I. Haber, R. Lee, H. H. Klein, and J. P. Boris, *Advances in electromagnetic plasma simulation techniques*, Proceedings of the Sixth Conference on Numerical Simulation of Plasmas, (1973), pp. 46–48.
- [45] M. F. Hadi and M. Piket-May, *A modified FDTD (2, 4) scheme for modeling electrically large structures with high-phase accuracy*, IEEE Transactions on Antennas and Propagation, 45 (1997), pp. 254–264.
- [46] T. Hagstrom, *High-resolution difference methods with exact evolution for multidimensional waves*, Applied Numerical Mathematics, 93 (2015), pp. 114 – 122. International Conference Difference Schemes and Applications in Honor of the 90-th Birthday of Professor V. S. Ryabenkii.
- [47] T. Hagstrom and S. Lau, *Radiation boundary conditions for Maxwell’s equations: a review of accurate time-domain formulations*, Journal of Computational Mathematics, (2007), pp. 305–336.
- [48] J. S. Hesthaven, *High-order accurate methods in time-domain computational electromagnetics: A review*, in Advances in imaging and electron physics, vol. 127, Elsevier, 2003, pp. 59–123.
- [49] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM Journal on Numerical Analysis, 34 (1997), pp. 1911–1925.
- [50] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), pp. 209–286.
- [51] R. Hockney, *Potential calculation and some applications*, vol. 9 of Methods in Computational Physics, Academic Press, 1970, pp. 136–212.

- [52] T. Z. Ismagilov, *Second order finite volume scheme for Maxwell's equations with discontinuous electromagnetic properties on unstructured meshes*, Journal of Computational Physics, 282 (2015), pp. 33–42.
- [53] J. D. Jackson, *Classical electrodynamics*, Wiley, 1999.
- [54] J.-M. Jin, *The finite element method in electromagnetics*, John Wiley & Sons, 2015.
- [55] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1214–1233.
- [56] T. Kato, *Perturbation theory for linear operators*, vol. 132, Springer Science & Business Media, 2013.
- [57] C. Kavouklis and P. Colella, *Computation of volume potentials on structured grids using the method of local corrections*, arXiv preprint arXiv:1702.08111, (2017).
- [58] M. Krumpholz and L. P. Katehi, *MRTD: New time-domain schemes based on multiresolution analysis*, IEEE Transactions on Microwave Theory and Techniques, 44 (1996), pp. 555–571.
- [59] K. Lan, Y. Liu, and W. Lin, *A higher order (2, 4) scheme for reducing dispersion in FDTD algorithm*, IEEE transactions on Electromagnetic Compatibility, 41 (1999), pp. 160–165.
- [60] J. D. Lawson, *Generalized Runge-Kutta processes for stable systems with large lipschitz constants*, SIAM Journal on Numerical Analysis, 4 (1967), pp. 372–380.
- [61] V. I. Lebedev, *Values of the nodes and weights of quadrature formulas of Gauss–Markov type for a sphere from the ninth to seventeenth order of accuracy that are invariant with respect to an octahedron group with inversion*, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 15 (1975), pp. 48–54.
- [62] J.-R. Li and L. Greengard, *High order marching schemes for the wave equation in complex geometry*, Journal of Computational Physics, 198 (2004), pp. 295 – 309.
- [63] Q. H. Liu, *The PSTD algorithm: A time-domain method requiring only two cells per wavelength*, Microwave and optical technology letters, 15 (1997), pp. 158–165.
- [64] B. Lo and P. Colella, *An adaptive local discrete convolution method for the numerical solution of Maxwell's equations*, Communications in Applied Mathematics and Computational Science. In press.
- [65] B. Lo, V. Minden, and P. Colella, *A real-space Green's function method for the numerical solution of Maxwell's equations*, Communications in Applied Mathematics and Computational Science, 11-2 (2016), pp. 143–170.

- [66] C. Manry, S. L. Broschat, and J. B. Schneider, *Higher-order FDTD methods for large problems*, Appl. Comput. Electromagn. Soc. J., 10 (1995), pp. 17–29.
- [67] B. Marder, *A method for incorporating Gauss' law into electromagnetic PIC codes*, Journal of Computational Physics, 68 (1987), pp. 48–55.
- [68] E. Marx and D. Maystre, *Dyadic Green functions for the time-dependent wave equation*, Journal of Mathematical Physics, 23 (1982), pp. 1047–1056.
- [69] P. McCorquodale, P. Colella, G. Balls, and S. Baden, *A local corrections algorithm for solving Poisson's equation in three dimensions*, Communications in Applied Mathematics and Computational Science, 2 (2007), pp. 57–81.
- [70] J. Monaghan, *Extrapolating B splines for interpolation*, Journal of Computational Physics, 60 (1985), pp. 253 – 262.
- [71] P. Monk et al., *Finite element methods for Maxwell's equations*, Oxford University Press, 2003.
- [72] J.-C. Nédélec, *Mixed finite elements in $\mathbb{R}^3$* , Numerische Mathematik, 35 (1980), pp. 315–341.
- [73] ———, *A new family of mixed finite elements $\mathbb{R}^3$* , Numerische Mathematik, 50 (1986), pp. 57–81.
- [74] R. D. Nevels, J. A. Miller, and R. E. Miller, *A path integral time-domain method for electromagnetic scattering*, IEEE Transactions on Antennas and Propagation, 48 (2000), pp. 565–573.
- [75] P. G. Petropoulos, *Phase error control for FD-TD methods of second and fourth order accuracy*, IEEE transactions on antennas and propagation, 42 (1994), pp. 859–862.
- [76] M. Remaki, *A new finite volume scheme for solving Maxwells system*, COMPEL-The international journal for computation and mathematics in electrical and electronic engineering, 19 (2000), pp. 913–932.
- [77] D. J. Riley and C. D. Turner, *Volmax: A solid-model-based, transient volumetric Maxwell solver using hybrid grids*, IEEE Antennas and Propagation Magazine, 39 (1997), pp. 20–33.
- [78] T. Rylander and A. Bondeson, *Stable FEM-FDTD hybrid method for Maxwell's equations*, Computer Physics Communications, 125 (2000), pp. 75–82.
- [79] I. Schoenberg, *Contributions to the problem of approximation of equidistant data by analytic functions*, in I. J. Schoenberg Selected Papers, C. de Boor, ed., Contemporary Mathematicians, Birkhäuser Boston, 1988, pp. 3–57.

- [80] J. Shang, *High-order compact-difference schemes for time-dependent Maxwell equations*, Journal of Computational Physics, 153 (1999), pp. 312–333.
- [81] V. Shankar, A. H. Mohammadian, and W. F. Hall, *A time-domain, finite-volume treatment for the Maxwell equations*, Electromagnetics, 10 (1990), pp. 127–145.
- [82] J. Shin and R. D. Nevels, *A time-domain propagator numerical method for computational electromagnetics*, IEEE Journal on Multiscale and Multiphysics Computational Techniques, 3 (2018), pp. 80–87.
- [83] C.-T. Tai, *Dyadic Green functions in electromagnetic theory*, Institute of Electrical & Electronics Engineers (IEEE), 1994.
- [84] A.-K. Tornberg and B. Engquist, *Numerical approximations of singular source terms in differential equations*, Journal of Computational Physics, 200 (2004), pp. 462 – 488.
- [85] S. V. Tsynkov, *Numerical solution of problems on unbounded domains. a review*, Applied Numerical Mathematics, 27 (1998), pp. 465–532.
- [86] E. Turkel and A. Yefet, *On the construction of a high order difference scheme for complex domains in a Cartesian grid*, Applied Numerical Mathematics, 33 (2000), pp. 113–124.
- [87] B. Van Straalen, *Method of Local Corrections Solver for Manycore Architectures*, PhD thesis, UC Berkeley, 2018.
- [88] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng, et al., *Warp-x: A new exascale computing platform for beam-plasma simulations*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 909 (2018), pp. 476–479.
- [89] J.-L. Vay, I. Haber, and B. B. Godfrey, *A domain decomposition method for pseudo-spectral electromagnetic simulations of plasmas.*, J. Comput. Physics, 243 (2013), pp. 260–268.
- [90] G. Vogman, *Fourth-order conservative Vlasov-Maxwell solver for Cartesian and cylindrical phase space coordinates*, PhD thesis, UC Berkeley, 2016.
- [91] Z. Wang, A. Przekwas, and Y. Liu, *A FV-TD electromagnetic solver using adaptive Cartesian grids*, Computer Physics Communications, 148 (2002), pp. 17–29.
- [92] E. W. Weisstein, *Newton-Cotes formulas. From Mathworld—a Wolfram web resource.* <http://mathworld.wolfram.com/Newton-CotesFormulas.html>. Accessed: 2019-02-05.
- [93] H. Weyl, *Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung)*, Mathematische Annalen, 71 (1912), pp. 441–479.

- [94] G. B. Whitham, *Linear and nonlinear waves*, John Wiley & Sons, 1974.
- [95] E. Whittaker and G. Robinson, *The Calculus of Observations: A Treatise on Numerical Mathematics*, New York: Dover, 1967.
- [96] S. Williams, A. Waterman, and D. Patterson, *Roofline: an insightful visual performance model for multicore architectures*, Communications of the ACM, 52 (2009), pp. 65–76.
- [97] M.-F. Wong, O. Picon, and V. F. Hanna, *A finite element method based on Whitney forms to solve Maxwell equations in the time domain*, IEEE Transactions on Magnetics, 31 (1995), pp. 1618–1621.
- [98] R.-B. Wu and T. Itoh, *Hybridizing FD-TD analysis with unconditionally stable FEM for objects of curved boundary*, in Microwave Symposium Digest, 1995., IEEE MTT-S International, IEEE, 1995, pp. 833–836.
- [99] K. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Transactions on antennas and propagation, 14 (1966), pp. 302–307.
- [100] K. S. Yee and J. S. Chen, *Conformal hybrid finite difference time domain and finite volume time domain*, IEEE transactions on antennas and propagation, 42 (1994), pp. 1450–1455.
- [101] —, *The finite-difference time-domain (FDTD) and the finite-volume time-domain (FVTD) methods in solving Maxwell's equations*, IEEE Transactions on Antennas and Propagation, 45 (1997), pp. 354–363.
- [102] J. L. Young, D. Gaitonde, and J. Shang, *Toward the construction of a fourth-order difference scheme for transient em wave simulation: Staggered grid approach*, IEEE Transactions on Antennas and Propagation, 45 (1997), pp. 1573–1580.
- [103] S. Zhao and G. Wei, *High-order FDTD methods via derivative matching for Maxwell's equations with material interfaces*, Journal of Computational Physics, 200 (2004), pp. 60–103.