12-15-2015

Method of lines transpose: an efficient A-stable solver for wave propagation

Matthew Causley

Andrew Christlieb

Eric Wolf

METHOD OF LINES TRANSPOSE: AN EFFICIENT A-STABLE SOLVER FOR WAVE PROPAGATION

MATTHEW CAUSLEY, ANDREW CHRISTLIEB, AND ERIC WOLF

Abstract. Building upon recent results obtained in [7, 8, 9], we describe an efficient second order, A-stable scheme for solving the wave equation, based on the method of lines transpose (MOL^T), and the resulting semi-discrete (i.e. continuous in space) boundary value problem. In [7], A-stable schemes of high order were derived, and in [9] a high order, fast $\mathcal{O}(N)$ spatial solver was derived, which is matrix-free and is based on dimensional-splitting.

In this work, are interested in building a wave solver, and our main concern is the development of boundary conditions. We demonstrate all desired boundary conditions for a wave solver, including outflow boundary conditions, in 1D and 2D. The scheme works in a logically Cartesian fashion, and the boundary points are embedded into the regular mesh, without incurring stability restrictions, so that boundary conditions are imposed without any reduction in the order of accuracy. We demonstrate how the embedded boundary approach works in the cases of Dirichlet and Neumann boundary conditions. Further, we develop outflow and periodic boundary conditions for the MOL^T formulation. Our solver is designed to couple with particle codes, and so special attention is also paid to the implementation of point sources, and soft sources which can be used to launch waves into waveguides.

Keywords: Method of Lines Transpose, Tranverse Method of Lines, Implicit Methods, Boundary Integral Methods, Alternating Direction Implicit Methods, ADI schemes

1. Introduction

Numerical solutions of the wave equation have been an area of investigation for many decades. The wave equation is ubiquitous in the physical world, arising in acoustics, electromagnetics, and fluid dynamics. Our main interest is in electromagnetic wave propagation, where traditional finite difference methods such as the finite-difference time-domain (FDTD) method are often used to solve Maxwell's equations. When the classical Yee scheme is used, the Courant-Friedrichs-Lewy (CFL) stability criterion restricts the time step to scale with the smallest cells in the domain. This becomes computationally prohibitive in a variety of interesting problems, such as electromagnetic scattering or waveguide design, where complex geometries need to be embedded in a Cartesian mesh, leading to small spatial cells near the boundaries. Alternatively, problems with multiple and disparate temporal or spatial scales, such as those presented by plasma simulations, require time steps and mesh spacings which are on the order of the shortest temporal and spatial scales. Due to the large number of charged particles in a typical simulation [4], and the high dimensionality of their distribution space, time steps are computationally prohibitive, and the amount of them must be minimized.

1991 *Mathematics Subject Classification.* Primary 65N12, 65N40, 35L05.

This work has been supported in part by AFOSR grants FA9550-11-1-0281, FA9550-12-1-0343 and FA9550-12-1-0455, NSF grant DMS-1115709, and MSU Foundation grant SPG-RG100059.

As a result, A-stable Maxwell solvers have been developed, which remove the CFL restriction and restore the ability of the user to define a time step which is based on the physical problem, rather than on its spatial discretization. Notable advances include the introduction of the FDTD-ADI [14, 15, 23, 24] algorithm, as well as several semi-implicit time split schemes. However, it remains difficult to preserve both accuracy, and A-stability when non-rectangular domains are required.

Alternatively, Maxwell's equations can be reformulated (e.g., using the scalar and vector potential formulation), so that each field component independently satisfies the second order wave equation. Method of lines (MOL) formulations of the wave equation are well-studied, and often lead to conditionally stable schemes. But when the discretization is first performed in time, following the method of lines transpose (MOL^T), the wave equation is found at discrete time levels by solving a semi-discrete boundary value problem. In [1, 2], the MOL^T is used to produce an exact integral formulation of the wave equation, where the solution is determined by a convolution over the domain of dependence against a space-time Green's function $G_d(x,t)$, in dimensions $d = 1, 2, 3$. However, more commonly the semi-discrete solution is obtained, in which the modified Helmholtz equation must be solved, and the corresponding semi-discrete Green's function $G(x, \Delta t)$ is the Yukawa, or modified Helmholtz kernel. The resulting boundary integrals methods can then be solved at each time step using fast summation algorithms, such as tree-codes [3, 12, 20, 21], or the fast multipole method (FMM) [17, 10, 13, 19, 16, 20, 11]. These methods often scale as $O(N)$ or $O(N \log N)$, but require substantial storage or precomputing stages.

It is worth noting however that in one spatial dimension, the modified Helmholtz kernel is a simple decaying exponential. This fact has been combined with alternate dimension implicit (ADI) splitting to achieve an A-stable wave solver with computational complexity of $O(N \log N)$ [6, 22], and $O(N)$ [8, 7], respectively. The scaling presented in [6, 22] is due to the Fourier continuation method, which makes use of the fast Fourier transform to compute a periodic extension of the the convolution integral, where the period is taken sufficiently longer than the domain, so that boundary conditions can be satisfied.

In our approach [7, 8, 9], the one-dimensional convolution is instead computed strictly over the computational domain, using polynomial interpolation. In [8], the solution is proven to be A-stable, second order accurate, and that the matrix formed by the discrete convolution can be applied in $\mathcal{O}(N)$ operations. However, the discrete convolution matrix was formed using piecewise linear integration, and it was found that a Lax-type correction was required to ensure convergence of the scheme in the semi-discrete limit $\Delta t \to 0$, where Δx is fixed. This issue was addressed in [9], where spatial discretization was extended to orders $M \geq 2$, on non-uniform grids while retaining $\mathcal{O}(N)$ complexity. In [7], the scheme was extended to higher orders in time using a novel approach, successive convolution. These higher order schemes were proven to be A-stable.

The purpose of this present work is to develop a robust embedded boundary approach for complex geometry for the MOL^T formulation. Because this paper is focused on developing a range of boundary conditions for the wave equation, we limit our attention to the 2nd order accurate solver. In addition to addressing standard closed (i.e., Dirichlet, Neumann and periodic) boundary conditions, we also develop an open, or outflow boundary condition which is suitable for our implicit solver. The method is extended to higher spatial dimensions using the factorization developed in [7], which is similar to but different from the traditional

ADI splitting. This means that solutions are computed line-by-line along dimensional sweeps leveraging the $O(N)$ 1D solver to construct a high dimensional $O(N)$ implicit method.

The rest of this paper is laid out as follows. In section 2, we use the method of lines transpose (MOL^T) to reformulate the 2D wave equation as an implicit, semi-discrete modified Helmholtz equation. The Helmholtz operator is inverted analytically using dimensional splitting, and we recast the solution as a series of one-dimensional boundary integral equations. We specifically show how to obtain a fully discretized, second order accurate solution by performing spatial quadrature, using a fast $\mathcal{O}(N)$ convolution algorithm.

In section 3, we modify the time-centered scheme of [7] to introduce artificial dissipation. This will be used to stabilize the embedded boundary method in our 2D algorithm, presented in section 4. We demonstrate our solution to be fast, second order accurate, and A-stable, even on non-rectangular domains with numerical results presented in section 5. We conclude the body of the paper with several remarks in section 6.

We also include in Appendix A, a table summarizing the main algorithmic aspects of our wave solver. In Appendix B, we show how point sources, which may represent charged plasma particles, or soft point sources launched into a waveguide, can be implemented.

2. Integral solution using MOL^T

We now develop a dimensionally split algorithm for solving the initial value problem

$$\begin{aligned} \nabla^2 u - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} &= -S(\mathbf{x},t), \quad \mathbf{x}\in\Omega, \quad t>0 \\ u(\mathbf{x},0) &= f(\mathbf{x}), \quad \mathbf{x}\in\Omega \\ u_t(\mathbf{x},0) &= g(\mathbf{x}), \quad \mathbf{x}\in\Omega, \end{aligned} \tag{1}$$

with consistent boundary conditions. We utilize the method of lines transpose (MOL^T) to perform a second order accurate temporal discretization, as was shown in [7]. In particular, the discretization is time-centered, and implicit in the spatial derivatives

$$\left(1-\frac{\nabla^2}{\alpha^2}\right)\left[u^{n+1}-2u^n+u^{n-1}+\beta^2 u^n\right](x,y) = \beta^2\left(u^n+\frac{1}{\alpha^2}S^n\right),$$

where

$$\alpha = \frac{\beta}{c\Delta t}, \qquad \beta>0,$$

and $\beta\le 2$ is chosen to enforce A-stability [7]. Inversion of this operator leads to a boundary integral formulation, where the Green's function is the Yukawa potential, defined for 2D in terms of the modified Bessel function $K_0(r)$. However we will employ our previously developed [7, 8] splitting algorithm, which has $\mathcal{O}(N)$ complexity

$$\mathcal{L}_x\mathcal{L}_y\left[u^{n+1}-2u^n+u^{n-1}+\beta^2 u^n\right](x,y) = \beta^2\left(u^n+\frac{1}{\alpha^2}S^n\right). \tag{2}$$

Here the subscripts denote the spatial component of univariate modified Helmholtz operators,

$$\mathcal{L}_x[u] := \left(1-\frac{\partial_{xx}}{\alpha^2}\right)u(x,y), \qquad \mathcal{L}_y[u] := \left(1-\frac{\partial_{yy}}{\alpha^2}\right)u(x,y). \tag{3}$$

The modified Helmholtz operators are formally inverted using the Green's function. We define convolution with this Green's function by the integral operator

$$I_x[u](x,y) := \frac{\alpha}{2}\int_a^b u(x',y)e^{-\alpha|x-x'|}dx', \quad a \le x \le b, \tag{4}$$

so that

$$\mathcal{L}_x^{-1}[u](x,y) := \underbrace{I_x[u](x)}_{\text{Particular Solution}} + \underbrace{Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}}_{\text{Homogeneous Solution}}, \tag{5}$$

where the coefficients A and B of the homogeneous solution are determined by applying boundary conditions. A similar definition holds for $\mathcal{L}_y^{-1}$. Formally inverting both operators to the right hand side we find the explicit equation

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2\mathcal{D}_{xy}[u^n] + \beta^2\mathcal{L}_x^{-1}\mathcal{L}_y^{-1}\left[\frac{1}{\alpha^2}S^n\right](x,y), \tag{6}$$

where the multidimensional operator is now

$$\mathcal{D}_{xy}[u] := u - \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}[u].$$

Since the application of $\mathcal{L}_x^{-1}$ is done for fixed y, and vice versa for $\mathcal{L}_y^{-1}$, the operator $\mathcal{D}_{xy}$ can be constructed in a line-by-line fashion, similar to ADI algorithms. It was also proven in [7] that the scheme (6) is A-stable, for $0 < \beta \le 2$.

Remark 1. *In the continuous case, $\mathcal{L}_x$ and $\mathcal{L}_y$ (and their inverses) commute, so $\mathcal{D}_{xy} = \mathcal{D}_{yx}$. However in practice the discretize operators will not commute, and some small spatial anisotropy is introduced. This can be controlled by applying both operators $\mathcal{D}_{xy}$ and $\mathcal{D}_{yx}$, and then averaging the result.*

Remark 2. *Similar to a traditional ADI formulation of the wave equation, this factorization produces a fourth order splitting error term,*

$$\mathcal{L}_x\mathcal{L}_y = \left(1 - \frac{\partial_{xx}}{\alpha^2}\right)\left(1 - \frac{\partial_{yy}}{\alpha^2}\right) = \left(1 - \frac{\partial_{xx} + \partial_{yy}}{\alpha^2} + \frac{\partial_{xx}\partial_{yy}}{\alpha^4}\right),$$

which can be compensated for by adding a term to the right hand side

$$\mathcal{L}_x\mathcal{L}_y\left[u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n\right] = \beta^2\left(u^n + \frac{1}{\alpha^2}S^n\right) + \beta^2\left(\mathcal{L}_x - 1\right)\left(\mathcal{L}_y - 1\right)[u^n]. \tag{7}$$

Note our use of the identity $(\mathcal{L}_x - 1)(\mathcal{L}_y - 1) = \partial_{xx}\partial_{yy}/\alpha^4$. Formally inverting both operators to the right hand side, the solution can be rearranged and found as

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2\mathcal{C}[u^n](x,y) + \beta^2\mathcal{L}_x^{-1}\mathcal{L}_y^{-1}\left[\frac{1}{\alpha^2}S^n\right](x,y), \tag{8}$$

where the convolution operator $\mathcal{C}$ is the operator defined in [7] as

$$\mathcal{C} := \mathcal{L}_x^{-1}\mathcal{D}_y + \mathcal{L}_y^{-1}\mathcal{D}_x = \mathcal{D}_{xy} - \mathcal{D}_x\mathcal{D}_y. \tag{9}$$

This form was used in [7] to achieve schemes of higher order through successive convolution, where removing splitting errors is of paramount concern.

2.1. **Fast convolution algorithm for the integral solution.** In [8], the particular solution (4) was discretized in space using the weighted midpoint and trapezoidal rules, which amounted to replacing u with a piecewise constant and linear approximation, respectively. More recently [9], we have detailed the spatial discretization of I_x to arbitrary order, while providing a means for its rapid evaluation. In this work we focus on the second order accurate implementation of this algorithm, and reiterate the relevant details.

This particular solution is first decomposed into a left and right oriented integral, split at $y = x$ so that

$$I[u](x) = I^L[u](x) + I^R[u](x), \tag{10}$$

where

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

These quantities can be updated locally, using exponential recursion

$$I^L[u](x) = I^L[u](x-\delta_L)e^{-\alpha\delta_L} + J^L[u](x), \quad J^L[u](x) := \frac{\alpha}{2}\int_0^{\delta_L} u(x-y)e^{-\alpha y}dy, \tag{11}$$

$$I^R[u](x) = I^R[u](x+\delta_R)e^{-\alpha\delta_R} + J^R[u](x), \quad J^R[u](x) := \frac{\alpha}{2}\int_0^{\delta_R} u(x+y)e^{-\alpha y}dy. \tag{12}$$

The recursive updates (11) and (12) are exact (in space), and making δ_L and δ_R small (typically, $\delta_L = \delta_R = \Delta x$) effectively localizes the contribution of the integrals.

Based on these observations, we now outline the fast convolution algorithm. In [9], we derived a spatial quadrature of general order $M \geq 2$ for irregular grids. Here, we will utilize a second-order accurate method on a uniform grid, which we explicitly develop.

Consider the domain (a, b) discretized by uniform grid points $x_1 = a < x_2 < \cdots < x_{N+1} = b$ of width $\Delta x = \frac{b-a}{N} = x_{j+1} - x_j$, $j = 1, ..., N$. Suppose, given a function f compactly supported in (a, b), we are to evaluate the convolution operator at each grid points, that is compute $I_j = I[f](x_j) = I^L[f](x_j)+I^R[f](x_j) = I_j^L+I_j^R$, in $O(N)$ operations. We proceed by evaluating the local integrals $J_j^L = J^L[f](x_j)$ and $J_j^L = J^L[f](x_j)$, as in (11) and (12). The local integrals may be evaluated with quadrature, or, if possible, analytically. A second-order accurate quadrature is given by

$$J_j^L \approx Pf(x_j) + Qf(x_{j-1}) + R(f(x_{j+1}) - 2f(x_j) + f(x_{j-1})) \tag{13}$$

$$J_j^R \approx Pf(x_j) + Qf(x_{j+1}) + R(f(x_{j+1}) - 2f(x_j) + f(x_{j-1})) \tag{14}$$

where defining $\nu = \alpha\Delta x$ and $d = e^{-\nu}$, the quadrature weights are given by

$$P = 1 - \frac{1-d}{\nu} \tag{15}$$

$$Q = -d + \frac{1-d}{\nu} \tag{16}$$

$$R = \frac{1-d}{\nu^2} - \frac{1+d}{2\nu}. \tag{17}$$

We summarize our fast method in Algorithm 1.

Algorithm 1 Fast convolution algorithm

(1) Compute J^L_{j+1} and J^R_j for $j = 1, ..., N$ via quadrature or analytical integration.

(2) Initialize $I^L_1 = 0$ and $I^R_{N+1} = 0$, and perform the exponential recursion, $I^L_{j+1} = J^L_{j+1} + e^{-\alpha\Delta x} I^L_j$ for $j = 1, ..., N$ and $I^R_{N-j+1} = J^R_{N-j+1} + e^{-\alpha\Delta x} I^R_{N-j+2}$ for $j = 1, ..., N$.

3. Artificial dissipation

As will be discussed in section 4.2.2, it is necessary to include some artificial dissipation in the numerical scheme to maintain stability with embedded boundary methods for Neumann boundary conditions. We first present a version of the wave solver based on a backwards difference formula (BDF) time discretization, leading to what we call a diffusive scheme, which is dissipative. This method has a larger truncation error than a centered scheme, does not possess a means to tune the level of dissipation, and also has an implicit source term (at time level $n+1$), which is problematic for application in the context of particle-in-cell (PIC) methods for the simulation of plasmas. The second-order centered (dispersive) scheme given above is therefore preferable, but are non-dissipative in their original forms. We give a method for adding tunable artificial dissipation terms into the centered scheme, while maintaining A-stability.

3.1. Diffusive wave solver.

We substitute the following backward difference formula (BDF) discretization:

$$u^{n+1}_{tt} = \frac{2u^{n+1} - 5u^n + 4u^{n-1} - u^{n-2}}{\Delta t^2} - \frac{11\Delta t^2}{12} u_{tttt}(x, \eta) \tag{18}$$

into the wave equation $\frac{1}{c^2} u_{tt} - \nabla^2 u = S(x, t)$.

Rearranging, defining $\alpha = \sqrt{2}/(c\Delta t)$ and dividing by α^2 gives the semi-discrete scheme

$$\left(-\frac{1}{\alpha^2}\Delta + 1\right) u^{n+1} = \frac{1}{2}\left(5u^n - 4u^{n-1} + u^{n-2}\right) + \frac{1}{\alpha^2} S(x, t^{n+1}) + O(\Delta t^4). \tag{19}$$

This method is A-stable and dissipative, but does not possess a mechanism for tuning the dissipation, has an inconvenient implicit source term (at time level $n+1$), and typically has a larger truncation error compared to the centered scheme.

3.2. Artificial dissipation in centered schemes.

3.2.1. *Artificial Dissipation in 1D.* We give a modified form of the centered version of the implicit wave solver with tunable artificial dissipation that retains the property of unconditional stability. We let ϵ denote a small artificial dissipation parameter, and $\mathcal{D}_x[u] = u - \mathcal{L}^{-1}[u] = u(x) - \frac{\alpha}{2}\int_{-\infty}^{\infty} e^{-\alpha|x-x'|} u(x')\, dx'$ be defined as usual.

Ignoring sources, we have the second order scheme with dissipation,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}_x[u^n] + \epsilon \mathcal{D}^2_x[u^{n-1}]. \tag{20}$$

We now prove the unconditional stability of this scheme for prescribed values of β. As in [7], we pass to the high-frequency limit.

We obtain the Von Neumann polynomial $\rho^2 - (2-\beta^2)\rho + (1-\epsilon)$. We can check that the roots of this polynomial will be complex if $0 < \beta \leq \sqrt{2+2\sqrt{1-\epsilon}}$, and that in this case the roots satisfy

$$|\rho|^2 = \frac{1}{4}\left((2-\beta^2)^2 + 4(1-\epsilon) - (2-\beta^2)\right) \tag{21}$$

$$= 1 - \epsilon < 1 \tag{22}$$

which shows both the stability and dissipative nature of the scheme. □

We note that the maximum allowed value of β is slightly smaller than what is allowed by the corresponding scheme without dissipation. A more detailed analysis shows that the effective damping rate is $\left(\frac{k^2}{k^2+\alpha^2}\right)^2 \epsilon$, meaning that high frequencies are more rapidly damped than low frequencies.

3.2.2. *Artificial Dissipation in 2D.* Using the notation defined in [7], and again ignoring sources, we have the second order scheme with dissipation,

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{C}[u^n] + \epsilon \mathcal{C}^2[u^{n-1}], \tag{23}$$

where now $\mathcal{D} = 1 - \mathcal{L}_x^{-1}\mathcal{L}_y^{-1}$ and $\mathcal{C} = \mathcal{L}_y^{-1}\mathcal{D}_x + \mathcal{L}_x^{-1}\mathcal{D}_y$. For further details on these operators, see [7]. Numerical experiments indicate that the 2D scheme with artificial dissipation is indeed unconditionally stable with the same maximum value of β as with the 1D schemes.

4. Boundary conditions

We will now discuss the implementation of boundary conditions. Since our algorithm is dimensionally split, we first develop the boundary conditions in one spatial dimension where the solution (6) reduces to

$$u^{n+1} - 2u^n + u^{n-1} = -\beta^2 \mathcal{D}_x[u^n] + \beta^2 \mathcal{L}_x^{-1}\left[\frac{1}{\alpha^2} S^n\right](x), \qquad a \leq x \leq b, \tag{24}$$

and where we consider the following boundary conditions

$$\text{Dirichlet:} \qquad u(a,t) = U_L(t), \quad u(b,t) = U_R(t), \tag{25}$$

$$\text{Neumann:} \qquad u_x(a,t) = V_L(t), \quad u_x(b,t) = V_R(t), \tag{26}$$

$$\text{Periodic} \qquad u(a,t) = u(b,t), \quad u_x(a,t) = u_x(b,t), \tag{27}$$

$$\text{Outflow:} \qquad u_t(a,t) = c u_x(a,t), \quad u_t(b,t) = -c u_x(b,t). \tag{28}$$

Once these boundary conditions have been derived, we use them to build a boundary solver in 2D.

4.1. **Boundary conditions in one dimension.** In 1D, this homogeneous solution requires the determination of two coefficients from the imposed boundary conditions and the endpoint values of the particular (integral) solution by solving a 2×2 linear system. We now show how to impose several common boundary conditions in 1D. These methods are extended to the 2D case in Section 4.2.

4.1.1. *1D Dirichlet boundary conditions.* Let us begin with Dirichlet boundary conditions (25). Evaluating the semi-discrete solution (24) at $x = a$ and b, we find

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

which, after solving for the unknown coefficients can be written as

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

with

$$w_a^D = I\left[u^n + \frac{1}{\alpha^2} S^n\right](a) - U_L(t^n) - \frac{U_L(t^{n+1}) - 2U_L(t^n) + U_L(t^{n-1})}{\beta^2},$$
$$w_b^D = I\left[u^n + \frac{1}{\alpha^2} S^n\right](b) - U_R(t^n) - \frac{U_R(t^{n+1}) - 2U_R(t^n) + U_R(t^{n-1})}{\beta^2},$$

and $\mu = e^{-\alpha(b-a)}$. Homogeneous boundary conditions are recovered upon setting $U_L(t) = U_R(t) = 0$. Solving the resulting linear system for the unknowns A^n and B^n gives

$$A = -\left(\frac{w_a^D - \mu w_b^D}{1 - \mu^2}\right), \quad B = -\left(\frac{w_b^D - \mu w_a^D}{1 - \mu^2}\right). \tag{29}$$

4.1.2. *1D Neumann boundary conditions.* For Neumann conditions, first observe that all dependence on x in the integral solution (5) is on the Green's function, which is a simple exponential function. Using this, we obtain the following identities

$$I'(a) = \alpha I(a), \quad I'(b) = -\alpha I(b). \tag{30}$$

Now, differentiating the semi-discrete solution (24), and applying the Neumann boundary conditions (26) at $x = a$ and b yields

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

which, after solving for the unknown coefficients can be written as

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

with

$$w_a^N = I\left[u^n + \frac{1}{\alpha^2} S^n\right](a) - \frac{1}{\alpha} V_L(t^n) - \frac{V_L(t^{n+1}) - 2V_L(t^n) + V_L(t^{n-1})}{\alpha\beta^2},$$
$$w_b^N = I\left[u^n + \frac{1}{\alpha^2} S^n\right](b) + \frac{1}{\alpha} V_R(t^n) + \frac{V_R(t^{n+1}) - 2V_R(t^n) + V_R(t^{n-1})}{\alpha\beta^2}.$$

Upon solving the linear system we obtain

$$A = \left(\frac{w_a^N + \mu w_b^N}{1-\mu^2} \right), \quad B = \left(\frac{w_b^N + \mu w_a^N}{1-\mu^2} \right). \tag{31}$$

Remark 3. *The cases of applying mixed boundary conditions at $x = a$ and b are not considered here, but the details follow from an analogous procedure to that demonstrated above.*

4.1.3. *1D Periodic boundary conditions.* We impose periodic boundary conditions, by assuming that

$$u^n(b) = u^n(a), \quad u_x^n(a) = u_x^n(b), \quad n \geq 0.$$

Enforcing this in the semi-discrete solution (24) then yields

$$\begin{aligned} I[u^n](a) + A + B\mu &= I[u^n](b) + A\mu + B, \\ \alpha \left(I[u^n](a) - A + B\mu \right) &= \alpha \left(-I[u^n](b) - A\mu + B \right), \end{aligned}$$

where we have used the identity (30) applied to derivatives of I. Solving this linear system is accomplished quickly by dividing the second equation by α, and either adding or subtracting it from the first equation, to produce

$$A = \frac{I[u^n](b)}{1-\mu}, \quad B = \frac{I[u^n](a)}{1-\mu}. \tag{32}$$

4.1.4. *1D Outflow boundary conditions.* When computing wave phenomena, whether we are interested in finite or infinite domains, it is often the case that we must restrict our attention to some smaller subdomain Ω of the problem, which does not include the physical boundaries. We say that Ω is the *computational domain*, and that the boundary $\partial\Omega$ is the non-physical, or *artificial boundary*. Under these circumstances, it is necessary to enforce an outflow, or non-reflecting boundary condition, which allows the wave to leave the computational domain, but not incur (non-physical) reflections at the artificial boundary.

For this reason, let us consider the free space solution

$$\mathcal{L}^{-1}[u](x) = \frac{\alpha}{2} \int_{-\infty}^{\infty} u(y) e^{-\alpha|x-y|} dy,$$

but where we are only interested in evaluating this expression for $x \in \Omega = [a, b]$. Then the contributions can be decomposed as

$$\begin{aligned} \mathcal{L}^{-1}[u](x) =& I[u](x) + \frac{\alpha}{2} \int_{-\infty}^{a} u(y) e^{-\alpha(x-y)} dy + \frac{\alpha}{2} \int_{b}^{\infty} u(y) e^{-\alpha(y-x)} dy \\ =& I[u](x) + Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}, \end{aligned} \tag{33}$$

where the homogeneous coefficients are

$$A = \frac{\alpha}{2} \int_{-\infty}^{a} u(y) e^{-\alpha(a-y)} dy, \tag{34}$$

$$B = \frac{\alpha}{2} \int_{b}^{\infty} u(y) e^{-\alpha(y-b)} dy, \tag{35}$$

which we observe do not depend on x. Since the coefficients A an B are the contributions of the integral to the left and right of $[a, b]$ respectively, they can be thought of as transmission conditions (rather than boundary conditions). We make use of this fact to develop outflow

boundary conditions, and it will serve as a key idea in our planed follow-up work on a domain decomposition algorithm and multi-core computing with our implicit wave solver.

For the one-dimensional wave equation the exact outflow boundary conditions (28) turn out to be local in space and time. We emphasize that this is only the case in one spatial dimension, but we shall utilize this fact to obtain an outflow boundary integral solution from the integral equation (33). We extend the support of our function to $(-\infty, \infty)$, and extend the definition of the outflow boundary conditions to the domains exterior to $[a, b]$

$$u_t + cu_x = 0, \quad x \geq b, \tag{36}$$

$$u_t - cu_x = 0, \quad x \leq a. \tag{37}$$

Next, assume the initial conditions have some compact support; for simplicity we will take this support to be $\Omega_0 = [a, b]$. Then after a time $t = t_n$, the domain of dependence of $u^n(x)$ is $\Omega_t = [a - ct_n, b + ct_n]$, since the propagation speed is c. Now the free space solution (33) becomes

$$\begin{aligned}\mathcal{L}^{-1}[u^n](x) =& \frac{\alpha}{2}\int_{a-ct_n}^{b+ct_n} e^{-\alpha|x-y|}u^n(y)dy \\ =& I[u^n](x) + A^n e^{-\alpha(x-a)} + B^n e^{-\alpha(b-x)}\end{aligned} \tag{38}$$

with coefficients

$$A^n = \frac{\alpha}{2}\int_{a-ct_n}^{a} e^{-\alpha(a-y)}u^n(y)dy, \tag{39}$$

$$B^n = \frac{\alpha}{2}\int_{b}^{b+ct_n} e^{-\alpha(y-b)}u^n(y)dy. \tag{40}$$

At first glance, these coefficients are not at all helpful, as they require computing integrals along spatial domains which not only are outside of the computational domain, but also grow linearly in time. However, we will now make use of the extended boundary conditions to turn these spatial integrals into time integrals, which exist at precisely the endpoints $x = a$ and b respectively. Consider first $x > b$. By assumption, this region contains only right traveling waves, $u(x, t) = u(x - ct)$, and by tracing backward along a characteristic ray we find

$$u(b + y, t) = u\left(b, t - \frac{y}{c}\right), \quad y > 0.$$

Thus,

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

and so B^n is equivalently represented by a convolution in time, rather than space. Now, knowing the history of u at $x = b$ is sufficient to impose outflow boundary conditions. Furthermore, we find in analog to equation (12), a temporal recurrence relation due to the

exponential

$$B^n = \frac{\alpha c}{2}\int_0^{\Delta t} e^{-\alpha c s} u\left(b, t_n - s\right) ds + e^{-\alpha c \Delta t}\left(\frac{\alpha c}{2}\int_0^{t_{n-1}} e^{-\alpha c s} u\left(b, t_{n-1} - s\right) ds\right)$$
$$= \frac{\beta}{2}\int_0^1 e^{-\beta z} u\left(b, t_n - z\Delta t\right) dz + e^{-\beta} B^{n-1},$$

where $\beta = \alpha c \Delta t$, by definition (3). Thus, the coefficient B^n, which imposes an outflow boundary condition at $x = b$, can be computed locally in both time and space. To maintain second order accuracy, we fit u with a quadratic interpolant

$$u(b, t_n - z\Delta t) \approx p(z) = u^n(b) - \frac{z}{2}\left(u^{n+1}(b) - u^{n-1}(b)\right) + \frac{z^2}{2}\left(u^{n+1}(b) - 2u^n(b) + u^{n-1}(b)\right)$$

and integrate the expression analytically to arrive at

$$B^n = e^{-\beta} B^{n-1} + \gamma_0 u^{n+1}(b) + \gamma_1 u^n(b) + \gamma_2 u^{n-1}(b) \tag{41}$$

where

$$\gamma_0 = \frac{E_2(\beta) - E_1(\beta)}{4} = \frac{(1 - e^{-\beta})}{2\beta^2} - \frac{(1 + e^{-\beta})}{4\beta}$$
$$\gamma_1 = \frac{E_0(\beta) - E_2(\beta)}{2} = -\frac{(1 - e^{-\beta})}{\beta^2} + \frac{1}{\beta}e^{-\beta} + \frac{1}{2}$$
$$\gamma_2 = \frac{E_2(\beta) + E_1(\beta)}{4} = \frac{(1 - e^{-\beta})}{2\beta^2} + \frac{(1 - 3e^{-\beta})}{4\beta} - \frac{e^{-\beta}}{2}.$$

In this outflow update equation (41), the quantities $u^{n+1}(b)$ and B^n are both unknown. In order to determine these values, we must also evaluate the update equation for u^{n+1} (24) at $x = b$

$$u^{n+1}(b) = 2u^n(b) - u^{n-1}(b) + \beta^2\left(-u^n(b) + I[u^n](b) + A^n\mu + B^n\right), \quad \mu = e^{-\alpha(b-a)}.$$

We now use these two equations to solve for $u^{n+1}(b)$, and eliminate it from the outflow update equation (41), so that

$$-\Gamma_0\mu A^n + (1 - \Gamma_0)B^n = e^{-\beta} B^{n-1} + \Gamma_0 I[u^n](b) + \Gamma_1 u^n(b) + \Gamma_2 u^{n-1}(b) \tag{42}$$

where

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

Remark 4. *While this procedure could be avoided by omitting $u^{n+1}(b)$ in the interpolation stencil, it turns out to be necessary to obtain convergent outflow boundary conditions.*

Likewise, upon considering $x < a$, we find

$$(1 - \Gamma_0)A^n - \Gamma_0\mu B^n = e^{-\beta} A^{n-1} + \Gamma_0 I(a) + \Gamma_1 u^n(a) + \Gamma_2 u^{n-1}(a). \tag{43}$$

Solving the resulting linear system produces

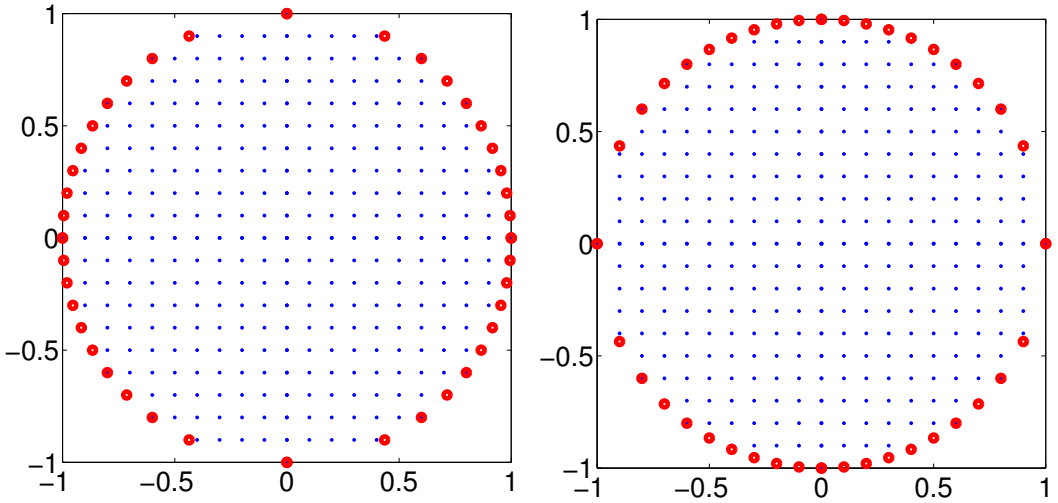
$$A^n = \frac{(1 - \Gamma_0)w_a^{\text{Out}} + \mu\Gamma_0 w_b^{\text{Out}}}{(1 - \Gamma_0)^2 - (\mu\Gamma_0)^2}, \quad B^n = \frac{(1 - \Gamma_0)w_b^{\text{Out}} + \mu\Gamma_0 w_a^{\text{Out}}}{(1 - \Gamma_0)^2 - (\mu\Gamma_0)^2}, \tag{44}$$

where

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

4.2. **Boundary conditions in two dimensions.** We now describe our approach for imposing boundary conditions up to second-order accuracy in two dimensions for our MOL^T formulation of the wave equation. Boundary conditions must be supplied for the intermediate sweep variable w. Since $w = u + O(c^2\Delta t^2)$, for second order accuracy it suffices for w to inherit the boundary condition imposed on the main solution variable u. In the case of rectangular, grid-aligned boundaries, the 1D boundary correction terms can be imposed in a line-by-line fashion. Typical applications of periodic and outflow boundary conditions can be imposed in this manner. In the case of complex boundary geometries that are not grid-aligned, Dirichlet boundary conditions can be imposed in a similar line-by-line fashion (by including irregular boundary points as the end points of the sweep lines), but Neumann boundary conditions require more careful treatment due to the resulting coupling of grid lines. Practically speaking, we only anticipate needing to impose Dirichlet and Neumann boundary conditions on complex boundary geometries, with outflow and periodic boundary conditions being imposed in a line-by-line approach using the 1D results from section 4.1 on rectangular domains. Hence, we will limit our discussion here to the implementation of Dirichlet and Neumann boundary conditions in two dimensions.

4.2.1. *Dirichlet boundary conditions in two dimensions.* Discretization of a general smooth domain Ω is accomplished by embedding it in a regular Cartesian mesh of say N_y horizontal (x) lines and N_x vertical (y) lines, and additionally incorporating the termination points of each line, which will lie on the boundary. For example, the lines and boundary points for a circle are shown in Figure 1.



(a) x lines (b) y lines

FIGURE 1. Mesh lines used by (a) x integration (45), and (b) y integration (46) for a circle. The red dots are placed on the boundary, and close each line.

Thus, a line $y = y_k$, which is discretized in x, and has endpoints a_k and b_k, determined as the intersection of the line with the boundary curve. Analogously we define endpoints of the line $x = x_j$ as c_j and d_j, and now have

$$\mathcal{L}_x^{-1}[u]_k(x) = \frac{\alpha}{2}\int_{a_k}^{b_k} e^{-\alpha|x-x'|}u(x',y)dx' + A_k e^{-\alpha(x-a_k)} + B_k e^{-\alpha(b_k-x)}, \tag{45}$$

$$\mathcal{L}_y^{-1}[u]_j(y) = \frac{\alpha}{2}\int_{c_j}^{d_j} e^{-\alpha|y-y'|}u(x,y')dy' + C_j e^{-\alpha(y-c_j)} + D_j e^{-\alpha(d_j-y)}, \tag{46}$$

for $1 \leq k \leq N_y$ and $1 \leq j \leq N_x$ respectively. If the boundary is defined using a level set function $C(x, y) = 0$, then the points $x = a_k, b_k$ can be found by solving $C(x, y_k) = 0$, and following from the analogous approach, the endpoints $y = c_j, d_j$ corresponding to $x = x_j$ are found.

Once the domain has been discretized and the lines have been defined, it remains to compute the discrete form of the scheme (6). Since the one-dimensional convolution algorithm presented in Section 2.1 is formulated for non-uniform grid points, the embedded boundary points do not affect the implementation of the horizontal (45) and vertical (46) sweeps. Thus, the only point of consideration that remains is that of boundary conditions.

First we consider an intermediate variable $w^{(1)}(x, y)$, defined by

$$w^{(1)} := \mathcal{L}_y\left[u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n\right],$$

and which, according to the implicit scheme (2), can be seen to satisfy

$$\mathcal{L}_x[w^{(1)}] = \beta^2\left(u^n + \frac{1}{\alpha^2}S^n\right).$$

Boundary conditions must be applied to $w^{(1)}$ at the boundary points which terminate each line $y = y_k$, defined as (a_k, y_k) and (b_k, y_k). Since u^{n+1}, u^n, and u^{n-1}, will be prescribed at these boundary points, we see that the boundary condition will be of the form

$$w^{(1)}(a_k, y_k) = \lim_{(x,y)\to(a_k,y_k)}\left(1 - \frac{\partial_{yy}}{\alpha^2}\right)\left(u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n\right)(x, y),$$

and the order of the limit and the partial derivatives commute only when partial derivatives of the boundary data can be constructed. But this is the case only when the tangent line at the boundary is in the y-direction, which is precisely why we must restrict our attention to the aforementioned cases for boundary conditions.

Upon introducing a second intermediate variable $w^{(2)}(x, y)$, defined by

$$w^{(2)} := u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n,$$

we make the observation that

$$\mathcal{L}_y[w^{(2)}] = w^{(1)}.$$

Boundary conditions are now applied to $w^{(2)}$ along the lines $x = x_j$, at the boundary points (x_j, c_j) and (x_j, d_j), where now no difficulties remain in considering

$$w^{(2)}(x_j, c_j) = \lim_{(x,y)\to(x_j,c_j)}\left(u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n\right)(x, y),$$

since the right hand side will be fully prescribed. We summarize this procedure in Algorithm 2.

Algorithm 2 Application of Boundary Conditions in 2 dimensions

(1) Initialize temporary variables $w^{(1)}$ and $w^{(2)}$, which are the same size u^n.

(2) For each horizontal line $y = y_k$, for $1 \leq k \leq N_y$, create the temporary variable $w_k^{(1)}(x)$, defined by

$$\mathcal{L}_x[w_k^{(1)}](x) = \beta^2 \left(u^n + \frac{1}{\alpha^2} S^n \right)(x, y_k).$$

The homogeneous coefficients are determined by the fact that

$$w^{(1)} = \mathcal{L}_y[u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n],$$

which is applied at $x = a_k, b_k$.

(3) For each vertical line $x = x_j$, for $1 \leq j \leq N_x$, create the temporary variable $w_j^{(2)}(y)$, defined by

$$\mathcal{L}_x[w_j^{(2)}](y) = w_k^{(1)}(x).$$

The homogeneous coefficients are determined by the fact that

$$w^{(2)} = [u^{n+1} - 2u^n + u^{n-1} + \beta^2 u^n],$$

which is applied at $y = c_j, d_j$.

(4) Solve for the update

$$u^{n+1} = 2u^n - u^{n-1} - \beta^2 u^n + w^{(2)}.$$

(5) The dimensional splitting error is corrected by adding

$$\beta^2 \mathcal{D}_x \mathcal{D}_y[u^n] = \beta^2 \mathcal{D}_y[u^n] + w^{(1)} - w^{(2)}.$$

4.2.2. *Neumann boundary conditions in two dimensions.* In implementing Neumann boundary conditions for boundary geometries conforming to grid lines, such as a rectangular domain, we can directly impose a two-point boundary correction. One way to extend this method to a general polygonal domain would be to use multiple overset grids, each aligned with a boundary segment, which communicate with the interior grid through interpolation on a ghost cell region, though we do not pursue that approach in this work.

For curved boundaries, an alternative approach is that of an embedded boundary method, which involves determining the Dirichlet values at the endpoints of each x- and y-sweep lines that result in the approximate satisfaction of the Neumann boundary condition (in effect, constructing an approximate Neumann-to-Dirichlet map). We present the implementation of an embedded boundary method for Neumann boundary conditions for the implicit wave solver on a curved boundary geometry. The approach taken here follows the work in [18], which proposes an embedded boundary method for Neumann boundary conditions with a finite difference method for the wave equation. The analysis in that work suggests that, on the continuous level, the modified equations and boundary conditions resulting from typical truncation error terms possess unstable boundary layer solutions, so that the addition of a dissipative term is necessary to achieve a stable method. This is consistent with our experience in the implementation described here, with the embedded boundary method

becoming unstable when applied to the non-dissipative dispersive solver, but remaining stable for the diffusive solver, which is dissipative.

In the following, we briefly describe the two-point boundary correction method for a grid-aligned rectangular boundary. We then describe the embedded boundary method for a 1D problem. This method requires an iterative procedure, which we show, in the setting of the 1D problem, to be a convergent contraction mapping with a rate of convergence that depends on the CFL number. We describe the implementation of the embedded boundary method in 2D, and finally give numerical results.

4.2.3. *Description of the Two-Point Boundary Correction Method.* In a rectangular domain where the boundaries conform to grid lines, it is straightforward to impose the two-point boundary correction terms in a line-by-line fashion, since in this case, the grid lines are not coupled through the normal derivative. As this is a simple extension of the 1D boundary correction algorithm, we do not elaborate further.

4.2.4. *Description of the Embedded Boundary Method and Proof of Convergence of the Iterative Solution in 1D.* We consider the situation of a one-dimensional domain $\{x_B < x\}$ with a single boundary point not aligned with the grid points, as displayed in Figure 2. We have grid points $x_0, x_1, ...$ with uniform grid spacing $x_{i+1} - x_i = \Delta x$, boundary location x_B, and ghost point location $x_G = x_0$. We define interior points to be any grid points lying within the domain (including the boundary), and exterior points to be any grid points lying outside of the domain. We define a ghost point to be any exterior point for which at least one of the neighboring points $x_{i\pm1}$ is an interior point. We neglect the right boundary in the present analysis for simplicity, though it can be extended to the case with both boundaries. We consider applying the diffusive version of the wave solver, having calculated the convolution integral $I(x)$, and now needing to find the value of the coefficient A such that the solution $u(x)$ at the next time step is given by

$$u(x) = I(x) + Ae^{-\alpha(x-x_G)}, \quad x \geq x_0$$

Given the value of the convolution integral and the solution at the ghost point, $I_G = I(x_G)$ and $u_G = u(x_G)$, respectively, the coefficient may be computed as $A = u_G - I_G$. We now describe the procedure for determining the value of the solution at the ghost point, u_G, that leads to a solution consistent with homogeneous Neumann boundary conditions to second-order accuracy. We construct a quadratic interpolant using the boundary condition and interior interpolation points $x_I = x_B + \Delta s_I$ and $x_{II} = x_B + 2\Delta s_I$, lying in between grid points x_m and x_{m+1}, and x_n and x_{n+1}, respectively. The interpolation distance will be chosen such that $\Delta x < \Delta s_I < (3/2)\Delta x$. We define the distances $\xi_G = x_B - x_G$, $\xi_I = x_I - x_B = \Delta s_I$, and $\xi_{II} = x_{II} - x_B = 2\Delta s_I$, and construct a quadratic Hermite-Birkhoff [5] interpolant $P(\xi)$ by imposing the conditions $P'(0) = 0$, $P(\xi_I) = u_I$ and $P(\xi_{II}) = u_{II}$. We then obtain the following second-order approximation to the ghost point value, given by

$$\begin{aligned} u_G &= P(\xi_G) + O(\Delta x^2) = u_I \frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2} + u_{II} \frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2} + O(\Delta x^2) \\ &= \gamma_I u_I + \gamma_{II} u_{II}. \end{aligned}$$

As the coefficients $\gamma_I = \frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2} > 0$ and $\gamma_{II} = \frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2} < 0$ are $O(1)$, we only need supply second-order accurate approximations to u_I and u_{II} to maintain overall second-order accuracy. (The coefficients would be $O(1/\Delta x)$ in the case of nonhomogeneous Neumann boundary conditions, which would require third-order accurate approximations to u_I and u_{II}. For simplicity, we consider only the case of homogeneous Neumann boundary conditions in the present work.) Such approximations may be obtained through linear interpolation, giving

$$\begin{aligned} u_I &= \sigma_I u_m + (1 - \sigma_I) u_{m+1} \\ u_{II} &= \sigma_{II} u_n + (1 - \sigma_{II}) u_{n+1} \end{aligned}$$

where $\sigma_I = \frac{x_{m+1} - x_I}{\Delta x}$, $\sigma_{II} = \frac{x_{n+1} - x_{II}}{\Delta x}$, and $u_j = u(x_j)$ are the values of the function at the uniform gridpoints for $j = m, m+1, n, n+1$.

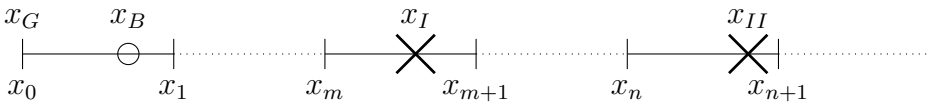


FIGURE 2. Boundary geometry in 1D.

Hence, to determine the ghost point value u_G that leads to a solution consistent with homogeneous Neumann boundary conditions, we should solve the following system of equations.

$$\begin{aligned} u_G &= \gamma_I(\sigma_I u_m + (1 - \sigma_I) u_{m+1}) + \gamma_{II}(\sigma_{II} u_n + (1 - \sigma_{II}) u_{n+1}) \\ u_m &= I_m + (u_G - I_G) e^{-\alpha(x_m - x_0)} \\ u_{m+1} &= I_{m+1} + (u_G - I_G) e^{-\alpha(x_{m+1} - x_0)} \\ u_n &= I_n + (u_G - I_G) e^{-\alpha(x_n - x_0)} \\ u_{n+1} &= I_{n+1} + (u_G - I_G) e^{-\alpha(x_{n+1} - x_0)} \end{aligned}$$

with γ_I, γ_{II}, σ_I and σ_{II} defined as above, and where $I_j = I(x_j)$ is the convolution integral evaluated at uniform grid points for $j = m, m+1, n, n+1$. This system can be solved formally for u_G, giving

$$\begin{aligned} u_G = \big[&\gamma_I \left(\sigma_I (I_m - I_G e^{-\alpha(x_m - x_G)}) + (1 - \sigma_I)(I_{m+1} - I_G e^{-\alpha(x_{m+1} - x_G)}) \right) + \\ &\gamma_{II} \left(\sigma_{II} (I_n - I_G e^{-\alpha(x_n - x_G)}) + (1 - \sigma_{II})(I_{n+1} - I_G e^{-\alpha(x_{n+1} - x_G)}) \right) \big] \div \\ \big[&1 - \gamma_I \left(\sigma_I e^{-\alpha(x_m - x_G)} + (1 - \sigma_I) e^{-\alpha(x_{m+1} - x_G)} \right) - \\ &\gamma_{II} \left(\sigma_{II} e^{-\alpha(x_n - x_G)} + (1 - \sigma_{II}) e^{-\alpha(x_{n+1} - x_G)} \right) \big] \end{aligned}$$

To show that this solution formula is well-defined, we argue that

$$\begin{aligned} 0 < K := &\gamma_I \left(\sigma_I e^{-\alpha(x_m - x_G)} + (1 - \sigma_I) e^{-\alpha(x_{m+1} - x_G)} \right) + \\ &+ \gamma_{II} \left(\sigma_{II} e^{-\alpha(x_n - x_G)} + (1 - \sigma_{II}) e^{-\alpha(x_{n+1} - x_G)} \right) < 1 \end{aligned}$$

for the relevant values of m, n and ξ_G, ξ_I, and ξ_{II}. We define $d = e^{-\alpha\Delta x}$, and noting that $0 < d < 1$, $m < n$, $\xi_G < \xi_I = \Delta s_I < \xi_{II} = 2\Delta s_I$, $\gamma_I > 0$, $\gamma_{II} < 0$, $0 \leq \sigma_I \leq 1$, and $0 \leq \sigma_{II} \leq 1$, we obtain

$$\begin{aligned} K &= \gamma_I d^m \left[\sigma_I + (1-\sigma_I)d\right] + \gamma_{II} d^n \left[\sigma_{II} + (1-\sigma_{II})d\right] \\ &\leq \gamma_I d^m + \gamma_{II} d^{n+1} \\ &= \frac{d^m \xi_{II}^2 - d^{n+1}\xi_I^2 + \xi_G^2(d^{n+1} - d^m)}{\xi_{II}^2 - \xi_I^2} \\ &\leq \frac{d^m \xi_{II}^2 - d^{n+1}\xi_I^2}{\xi_{II}^2 - \xi_I^2} \\ &= \frac{4\Delta s_I^2 d^m - d^{n+1}\Delta s_I^2}{4\Delta s_I^2 - \Delta s_I^2} = \frac{4d^m - d^{n+1}}{3} \end{aligned}$$

Now, since $\Delta x < \Delta s_I < (3/2)\Delta x$, we can see that it is the case that either $m = 1$ and $n = 2$ or 3, or that $m = 2$ and $n = 3$. It is then a matter of some simple calculus to check that that the functions $f_{m,n}(x) = (4x^m - x^{n+1})/3$ satisfy $f_{m,n}(x) < 1$ for $0 < x < 1$ and the given combinations of m and n. This proves that $K < 1$ for the relevant values of the parameters, so that the solution for u_G is well-defined. We note, however, that K approaches 1 as d approaches 1, that is, as the CFL number becomes large. Thus, we may expect an ill-conditioned system when the CFL number is very large.

To obtain the lower bound on K, we observe

$$\begin{aligned} K &= \gamma_I d^m \left[\sigma_I + (1-\sigma_I)d\right] + \gamma_{II} d^n \left[\sigma_{II} + (1-\sigma_{II})d\right] \\ &\geq \gamma_I d^{m+1} + \gamma_{II} d^n \\ &= \frac{d^{m+1}(\xi_{II}^2 - \xi_G^2) + d^n(\xi_G^2 - \xi_I^2)}{\xi_{II}^2 - \xi_I^2} \\ &= \frac{d^{m+1}(4\xi_I^2 - \xi_G^2) + d^n(\xi_G^2 - \xi_I^2)}{\xi_{II}^2 - \xi_I^2} \\ &= \frac{(d^{m+1} - d^n)(\xi_I^2 - \xi_G^2) + 3d^{m+1}\xi_I^2}{\xi_{II}^2 - \xi_I^2} > 0 \end{aligned}$$

In the two-dimensional case, the line-by-line solution method couples the ghost point values, and a general explicit solution formula is impossible to write down. In principle, one may write out and directly solve a linear system to obtain the ghost point values. Instead, we propose an iterative solution method that avoids the formation of a matrix. We now describe this iterative solution method and prove its convergence in the context of the one-dimensional problem described above.

Suppose we have the convolution integral evaluated at the gridpoints, I_j, and a k-th iterate for the ghost point value, u_G^k. Then we may obtain the next iterate by the formulas

$$\begin{aligned}
u_m^{k+1} &= I_m + (u_G^k - I_G)e^{-\alpha(x_m - x_0)} \\
u_{m+1}^{k+1} &= I_{m+1} + (u_G^k - I_G)e^{-\alpha(x_{m+1} - x_0)} \\
u_n^{k+1} &= I_n + (u_G^k - I_G)e^{-\alpha(x_n - x_0)} \\
u_{n+1}^{k+1} &= I_{n+1} + (u_G^k - I_G)e^{-\alpha(x_{n+1} - x_0)} \\
u_G^{k+1} &= \gamma_I(\sigma_I u_m^{k+1} + (1-\sigma_I)u_{m+1}^{k+1}) + \gamma_{II}(\sigma_{II} u_n^{k+1} + (1-\sigma_{II})u_{n+1}^{k+1})
\end{aligned}$$

where quantities are defined as above. Now, to prove the convergence of the interation, we show it is contractive. Taking the difference of two iterates, we have

$$\begin{aligned}
|u_G^{k+1} - u_G^k| &= |\gamma_I(\sigma_I(u_m^{k+1} - u_m^k) + (1-\sigma_I)(u_{m+1}^{k+1} - u_{m+1}^k)) + \\
&\quad \gamma_{II}(\sigma_{II}(u_n^{k+1} - u_n^k) + (1-\sigma_{II})(u_{n+1}^{k+1} - u_{n+1}^k))| \\
&= |\gamma_I \left(\sigma_I(u_G^k - u_G^{k-1})e^{-\alpha(x_m - x_G)} + (1-\sigma_I)(u_G^k - u_G^{k-1})e^{-\alpha(x_{m+1} - x_G)}\right) + \\
&\quad + \gamma_{II} \left(\sigma_{II}(u_G^k - u_G^{k-1})e^{-\alpha(x_n - x_G)} + (1-\sigma_{II})(u_G^k - u_G^{k-1})e^{-\alpha(x_{n+1} - x_G)}\right)| \\
&\leq K|u_G^k - u_G^{k-1}|
\end{aligned}$$

where $0 < K < 1$ as defined above. Hence, the Contraction Mapping Theorem implies that the iteration converges to a unique fixed point (which is the solution given in the formula above). We note again that K approaches 1 as the CFL number becomes large, so that the rate of convergence will become slower for larger CFL numbers.

4.2.5. *Description of the Method in 2D.* We now describe the implementation of the embedded Neumann boundary condition in the 2D case. We consider the situation displayed in Figure 3, in which we need to determine the value of our unknown u_G at the ghost point location (x_G, y_G). In the 2D case, we define a ghost point to be any exterior point (x_i, y_j) for which at least one of the neighboring points $(x_{i\pm1}, y_j)$ or $(x_i, y_{j\pm1})$ is an interior point. Similarly to the 1D case, we will construct a quadratic Hermite-Birkhoff boundary interpolant $P(\xi)$ along the direction normal to the boundary, which intersects the boundary curve Γ at location (x_B, y_B), and supply the interior interpolation point values u_I and u_{II}, at points (x_I, y_I) and (x_{II}, y_{II}), respectively, by further interpolation from interior grid points. These points are selected along the normal, in analogy to the 1D case, such that $\xi_I = |(x_I, y_I) - (x_B, y_B)| = \Delta s_I$ and $\xi_{II} = |(x_{II}, y_{II}) - (x_B, y_B)| = 2\Delta s_I$, where we will typically take $\Delta s_I = \sqrt{2}\Delta x$. We construct a quadratic Hermite-Birkhoff interpolant $P(\xi)$ by imposing the conditions $P'(0) = 0$, $P(\xi_I) = u_I$ and $P(\xi_{II}) = u_{II}$. Defining further the distance from the boundary to the ghost point $\xi_G = |(x_G, y_G) - (x_B, y_B)|$, we obtain, as in the 1D case, the following second-order approximation to the ghost point value, given by

$$\begin{aligned}
u_G = P(\xi_G) + O(\Delta x^2) &= u_I \frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2} + u_{II} \frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2} + O(\Delta x^2) \\
&= \gamma_I u_I + \gamma_{II} u_{II}.
\end{aligned}$$

where the coefficients are defined as $\gamma_I = \frac{\xi_{II}^2 - \xi_G^2}{\xi_{II}^2 - \xi_I^2} > 0$ and $\gamma_{II} = \frac{\xi_G^2 - \xi_I^2}{\xi_{II}^2 - \xi_I^2} < 0$. In the 2D case, we find approximations to u_I and u_{II} through bilinear interpolation. This is in contrast

to [18], who find the intersection of the normal with grid lines, then interpolate along the grid lines. We have also implemented a second-order accurate version of this approach and compared to the bilinear interpolation scheme proposed here. We have found that the two schemes behave similarly, however the bilinear interpolation scheme is slightly more accurate and simpler to code, not requiring to handle separate cases of intersection with horizontal and vertical grid lines. The bilinear interpolation scheme is standard, but we give it here for completeness. If the interpolation point u_I lies in a cell with corners (x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}) and (x_i, y_{j+1}), then we have the following approximation for u_I:

$$u_I = w_1 u_{i,j} + w_2 u_{i+1,j} + w_3 u_{i+1,j+1} + w_4 u_{i,j+1}$$

where $w_1 = \frac{(x_{i+1}-x_I)(y_{j+1}-y_I)}{\Delta x \Delta y}$, $w_2 = \frac{(x_I-x_i)(y_{j+1}-y_I)}{\Delta x \Delta y}$, $w_3 = \frac{(x_I-x_i)(y_I-y_j)}{\Delta x \Delta y}$ and $w_4 = \frac{(x_{i+1}-x_I)(y_I-y_j)}{\Delta x \Delta y}$. With this interpolation scheme established, we now outline the algorithm for the 2D dimensionally-split wave solver.

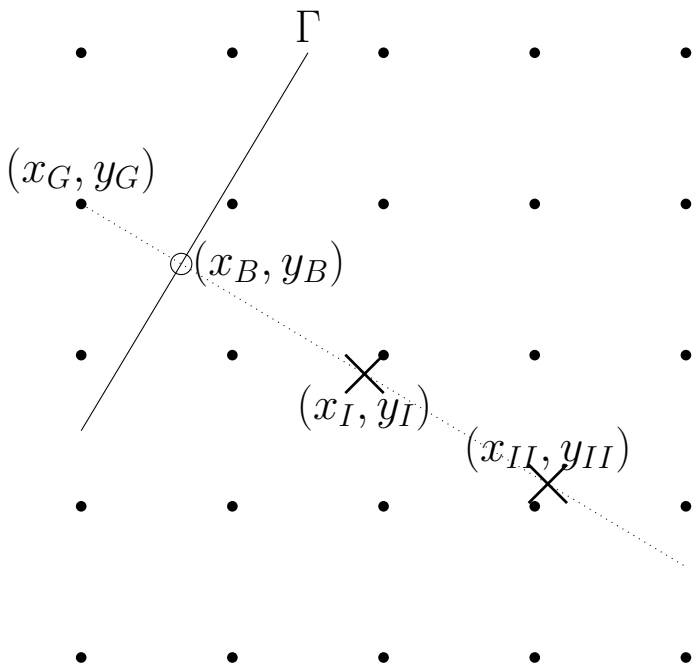


FIGURE 3. Boundary geometry in 2D.

The above interpolation procedure applies regardless of the variety of the wave solver that it is used with, provided that the wave solver has sufficient dissipation to maintain stability. We now describe the rest of the embedded boundary algorithm in the context of the diffusive wave solver, though it it may be similarly applied to the the dispersive scheme with artificial dissipation described above. In analogy to the iteration presented in the 1D case, the 2D iterative algorithm proceeds by using the interpolation scheme to provide values at the ghost points, which in turn provide new values for the boundary correction coefficients, which are then used to update the values at the interior grid points, comprising one full iteration. It should be noted that not all interior grid points need be updated in the iteration, only those near the boundary that lie within the boundary interpolation stencils.

Using values from previous time steps, the initial guess for the interior grid points in the boundary interpolation stencils may be given by extrapolation in time, as $u^{n+1,0} = 2u^n - u^{n-1}$ (linear extrapolation) or $u^{n+1,0} = 3u^n - 3u^{n-1} + u^{n-2}$ (quadratic extrapolation). Either extrapolated initial guess provides a modest reduction in the number of iterations required versus a zero initial guess, with only a slight further reduction in the number of iterations going from linear to quadratic extrapolation. An effective stopping criterion for iteration is $|u^{n+1,l+1} - u^{n+1,l}|_\infty < \epsilon$, where ϵ is some chosen tolerance, which may be chosen to be quite small, as the iteration is a fixed point interation. In the numerical example, we choose a tolerance of 10^{-15}, and we achieve convergence in less than 40 iterations at a CFL number of 2.

In applying the diffusive version of the wave solver, we assume we have previous time steps u^n, u^{n-1} and u^{n-2}. We have to solve the modified Helmholtz equation with homogeneous Neumann boundary conditions,

$$\left(1 - \frac{1}{\alpha^2}\nabla^2\right) u^{n+1} = \frac{1}{2}\left(5u^n - 4u^{n-1} + u^{n-2}\right) \text{ in } \Omega$$
$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma = \partial\Omega$$

where $\alpha = \frac{\sqrt{2}}{c\Delta t}$. We apply dimensional splitting to find

$$\left(1 - \frac{1}{\alpha^2}\nabla^2\right) u^{n+1} = \left(1 - \frac{1}{\alpha^2}\partial_{xx}\right)\left(1 - \frac{1}{\alpha^2}\partial_{yy}\right) u^{n+1} + O\left(\frac{1}{\alpha^4}\right)$$

so we define $w = \left(1 - \frac{1}{\alpha^2}\partial_{yy}\right) u$, and noting that $w = u + O\left((c\Delta t)^2\right)$ so that $\frac{\partial w}{\partial n} = \frac{\partial u}{\partial n} + O\left((c\Delta t)^2\right)$, we obtain the following approximate system

$$\left(1 - \frac{1}{\alpha^2}\partial_{xx}\right) w^{n+1} = \frac{1}{2}\left(5u^n - 4u^{n-1} + u^{n-2}\right) \text{ in } \Omega$$
$$\frac{\partial w}{\partial n}^{n+1} = 0 \text{ on } \Gamma = \partial\Omega$$
$$\left(1 - \frac{1}{\alpha^2}\partial_{yy}\right) u^{n+1} = w^{n+1} \text{ in } \Omega$$
$$\frac{\partial u}{\partial n}^{n+1} = 0 \text{ on } \Gamma = \partial\Omega$$

We now suppose our domain is embedded in a uniform Cartesian grid, with horizontal grid lines corresponding to $y = y_k$, $1 \le k \le N_y$ and vertical grid lines corresponding to $x = x_j$, $1 \le j \le N_x$. The embedded boundary algorithm will be applied when calculating the intermediate variable w^{n+1} in horizontal line sweeps as well as the solution variable u^{n+1} in vertical line sweeps. The iterations for these two variables are separate; first, the iterative procedure is applied to w to convergence, and then this value of w is used to compute u, and the iterative procedure is applied to u to convergence. However, in each iteration, the grid lines are coupled through the interpolation scheme, so that all grid lines must be iterated

together. The overall iterative algorithm is described in 3, with details specified for the iteration on w. The iteration on u is very similar, and so we omit the details.

Algorithm 3 Application of Neumann Boundary Conditions in 2 dimensions

(1) **(Initialization of ghost points)** Perform the interpolation scheme described above to obtain the values of u^n, u^{n-1}, and u^{n-2} at the ghost points, which are the endpoints of the horizontal and vertical grid lines.

(2) **(Evaluation of particular solution)** For each horizontal line $y = y_k$, for $1 \leq k \leq N_y$, with ghost (end) points $x = a_k$ and b_k find the particular solution $w_{p,k}^{n+1}$ for the intermediate variable $w_k^{n+1}(x)$ by evaluating the discrete convolution operator

$$w_{p,k}^{n+1}(x_j) = \frac{\alpha}{4} \int_{a_k}^{b_k} [5u^n - 4u^{n-1} + u^{n-2}](x', y_k) e^{-\alpha|x_j - x'|}\, dx'$$

for each grid point x_j in the horizontal line, including the ghost points.

(3) **(Boundary correction initialization)** For each horizontal line $y = y_k$, set the initial guess for the intermediate variable via extrapolation, $w_k^{n+1,0} = 3w_k^n - 3w_k^{n-1} + w_k^{n-2}$, on the interior points within the boundary interpolation stencil.

(4) **(Boundary correction iteration)** For each horizontal line $y = y_k$, perform the interpolation scheme using the interior values of $w_{p,k}^{n+1,l}$ to find the ghost point values. Using these ghost point values, apply the boundary correction on each line to obtain the updated intermediate variable,

$$w_k^{n+1,l+1}(x_j) = w_{p,k}^{n+1}(x_j) + A_k e^{-\alpha(x_j - a_k)} + B_k e^{-\alpha(b_k - x_j)}$$

for the values of x_j lying within the boundary interpolation stencil, where $A_k = \frac{w_k^{n+1,l}(a_k) - w_{p,k}^{n+1}(a_k) - \mu_k\left(w_k^{n+1,l}(b_k) - w_{p,k}^{n+1}(b_k)\right)}{1-\mu_k^2}$, $B_k = \frac{w_k^{n+1,l}(b_k) - w_{p,k}^{n+1}(b_k) - \mu_k\left(w_k^{n+1,l}(a_k) - w_{p,k}^{n+1}(a_k)\right)}{1-\mu_k^2}$, $\mu_k = e^{-\alpha(b_k - a_k)}$. Check for convergence, and if converged, store the intermediate variable w^{n+1}.

(5) Repeat this process for the vertical line sweeps, using the intermediate variable w^{n+1} to calculate the particular solution for u^{n+1}, then apply the bounday correction interation.

5. Numerical Results

5.1. **Double Circle Cavity.** In this example, we solve the wave equation with homogeneous Dirichlet boundary conditions on a 2D domain Ω which is, as in Figure 4, the union of two overlapping disks, with centers $P_1 = (-\gamma, 0)$ and $P_2 = (\gamma, 0)$, respectively, and each with radius R:

$$\Omega = \{(x,y) : |\,(x,y) - P_1| < R\} \cup \{(x,y) : |\,(x,y) - P_2| < R\}$$

where $|\,(x,y)\,| = \sqrt{x^2 + y^2}$ is the usual Euclidean vector norm, and $\gamma < R$.

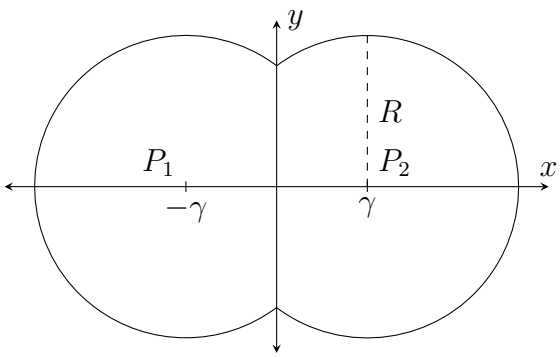
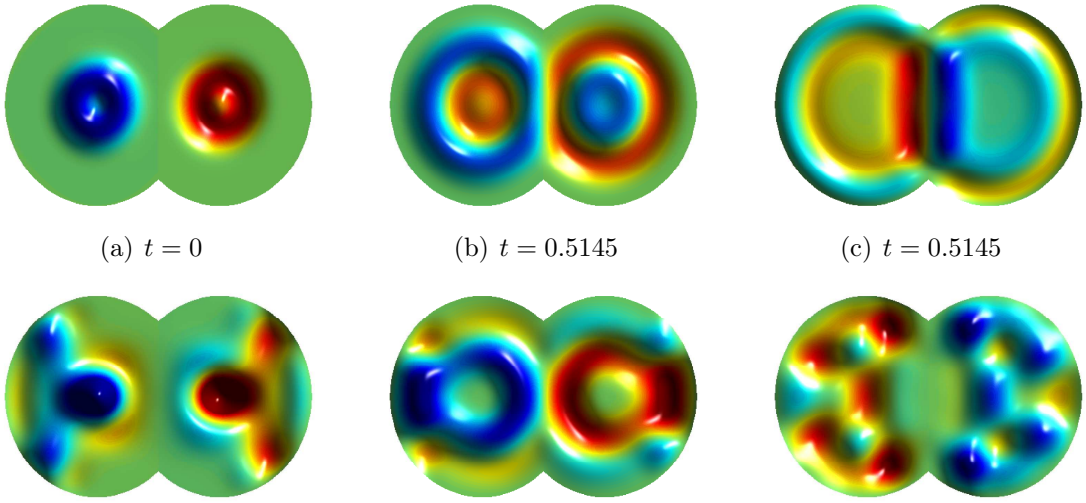


FIGURE 4. Double circle geometry.

This geometry is of interest due to, for example, its similarity to that of the radio frequency (RF) cavities used in the design of linear particle accelerators, and presents numerical difficulties due to the curvature of, and presence of corners in, the boundary. Our method avoids the staircase approximation used in typical finite difference methods to handle curved boundaries, which reduces accuracy to first order and may introduce spurious numerical diffraction.



(a) $t = 0$ (b) $t = 0.5145$ (c) $t = 0.5145$

(d) $t = 0.5145$ (e) $t = 0.9135$ (f) $t = 1$

FIGURE 5. Evolution of the double circle cavity problem.

As initial conditions, we choose

$$u(x,y,0) = \begin{cases} -\cos^6\left(\frac{\pi}{2}\left(\frac{|(x,y)-P_1|}{0.8\gamma}\right)^2\right) & |(x,y)-P_1| < 0.8\gamma \\ \cos^6\left(\frac{\pi}{2}\left(\frac{|(x,y)-P_2|}{0.8\gamma}\right)^2\right) & |(x,y)-P_2| < 0.8\gamma \\ 0 & \text{otherwise} \end{cases}$$

and

$$u_t(x, y, 0) = 0$$

for $(x, y) \in \Omega$. Selected snapshots of the evolution are given in Figure 5, and the results of a refinement study are given in Table 1. The discrete L^2 error was computed against a well-refined numerical reference solution ($\Delta x = 2.1875 \times 10^{-4}$); the error displayed in the table is the maximum over time steps with $t \in [0.28, 0.29]$. For this example, $R = 0.3$, $\gamma = 0.2$, $c = 1$, and the CFL is 2.

Δx	Δy	Δt	L^2 error	L^2 order
7.0000×10^{-3}	4.3333×10^{-3}	8.6667×10^{-3}	6.1437×10^{-3}	–
3.5000×10^{-3}	2.1667×10^{-3}	4.3333×10^{-3}	1.6829×10^{-3}	1.8681
1.7500×10^{-3}	1.0833×10^{-3}	2.1667×10^{-3}	4.3595×10^{-4}	1.9488
8.7500×10^{-4}	5.4167×10^{-4}	1.0833×10^{-3}	1.0515×10^{-4}	2.0517

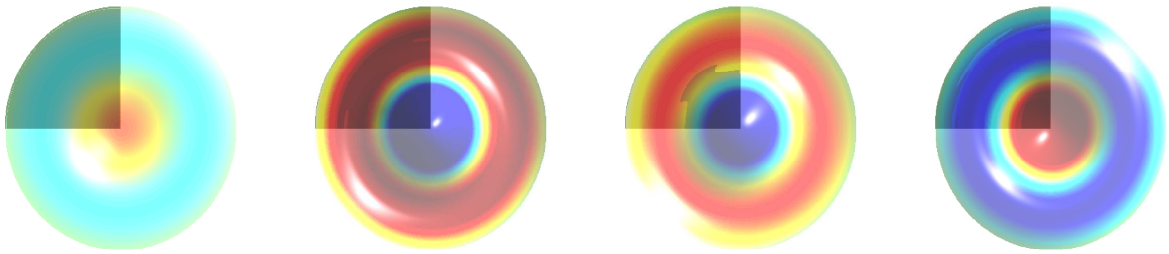
TABLE 1. Refinement study for the double circle cavity with Dirichlet BC. For the numerical reference solution, $\Delta x = 2.1875 \times 10^{-4}$, $\Delta y = 1.3542 \times 10^{-4}$, and $\Delta t = 2.7083 \times 10^{-4}$.

5.2. **Symmetry on a Quarter Circle.** With the goal of testing the capabilities of our boundary conditions as well as circular geometry, we construct standing modes on a circular wave guide of radius R, in two different ways. First, we solve the Dirichlet problem, with initial conditions

$$u(x, y, 0) = J_0\left(z_{20}\frac{r}{R}\right), \quad u_t(x, y, 0) = 0,$$

and exact solution $u = J_0\left(z_{20}\frac{r}{R}\right)\cos\left(z_{20}\frac{ct}{R}\right)$, where J_0 is the Bessel function of order 0, and $z_{20} = 5.5218$ is the 2-nd zero. Secondly, we use the symmetry of the mode to construct the solution restricted to the second quadrant, with homogeneous Neumann boundary conditions taken along the x and y axes.

In both cases, the solution converges to second order. An overlay of the two are shown in Figure 6, demonstrating the close agreement.



(a) $t = 0.25$ (b) $t = 0.50$ (c) $t = 0.75$ (d) $t = 1.00$

FIGURE 6. Two separate numerical constructions of a Bessel mode are superimposed, demonstrating that the solution on the quarter circle using Neumann boundary conditions is equivalent to that of the full circle.

5.3. **Bessel Mode with Neumann Boundary Conditions.** Here we present a numerical example of the embedded boundary method for homogeneous Neumann boundary conditions given in Section 4.2.2. We apply the method to a circular domain, for which analytical solutions exist. We consider a radially-symmetric Bessel mode with homogeneous Neumann boundary conditions, with an analytic solution given by

$$u(r,t) = J_0\left(Z_0\frac{r}{R}\right)\cos\left(Z_0\frac{ct}{R}\right), \tag{47}$$

where J_0 is the Bessel function of the first kind of order 0, $r = \sqrt{x^2+y^2}$, R is the radius of the domain, and $Z_0 \approx 3.8317$ is the smallest nonzero root of J_0' (so that $\frac{\partial u}{\partial \mathbf{n}}(R,t) = \frac{\partial u}{\partial r}(R,t) = \frac{Z_0}{R}J_0'(Z_0)\cos\left(Z_0\frac{ct}{R}\right) = 0$). In this example, we take radius $R = \pi/2$ and wave speed $c = 1$. An example of the embedded boundary grid used is given in Figure 7. We perform a refinement study with a fixed CFL number of 2, with the results in Figure 8 indicating the expected second-order convergence. We set the iteration tolerance to 10^{-15}, and we see convergence of the boundary correction iteration in fewer than 40 iterations. We note some oscillation of the L^∞ error about the line giving second-order accuracy, which we believe to be due to the grid points moving with respect to the boundary through the refinement, causing some variation in the maximum error.

5.4. **Periodic Slit Diffraction Grating.** In this example, we apply our method to model an infinite, periodic diffraction grating under an incident plane wave. Diffraction gratings are periodic structures used in optics to separate different wavelengths of light, much like a prism. The high resolution that can be achieved with diffraction gratings makes them useful in spectroscopy, for example, in the determination of atomic and molecular spectra. Our numerical experiment, depicted in Figure 9, demonstrates the use of our method with multiple boundary conditions and nontrivial geometry in a single simulation to capture complex wave phenomena.

An idealized slit diffraction grating consists of a reflecting screen of vanishing thickness, with open slits of aperture width a, spaced distance d apart, measured from the end of one slit to the beginning of the next (that is, the periodicity of the grating is d). We impose an incident plane wave of the form $u_{inc}(x,y,t) = \cos(\omega t + ky)$, where $k = 2\pi/a$ and $\omega = k/c$, where c is the wave speed. Periodic BCs at $x = \pm d/2$ (determining the periodicity of the grating), and homogeneous Dirichlet BCs are imposed at the screen. We also test outflow boundary conditions in multiple dimensions, which are imposed at $y = \pm L_y/2$.

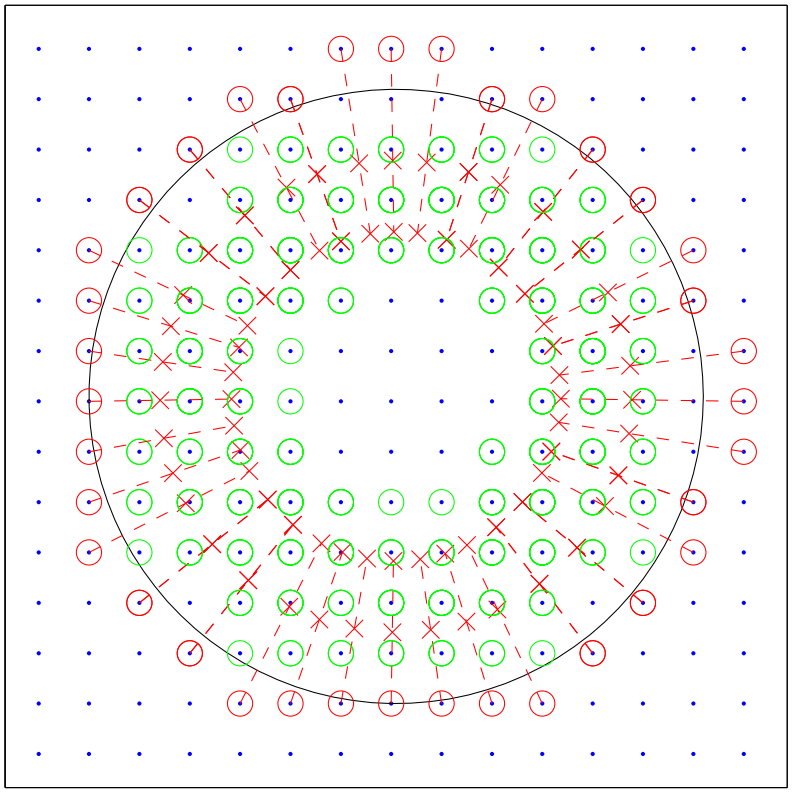


FIGURE 7. An example of the embedded boundary grid used. The red circled exterior grid points are the endpoints where a value is to be calculated via the interpolation procedure. The red crosses are the points where values are imposed on quadratic boundary interpolant along the normal direction (red dashed line). Values for the bilinear interpolants are supplied from the green circled interior grid points.

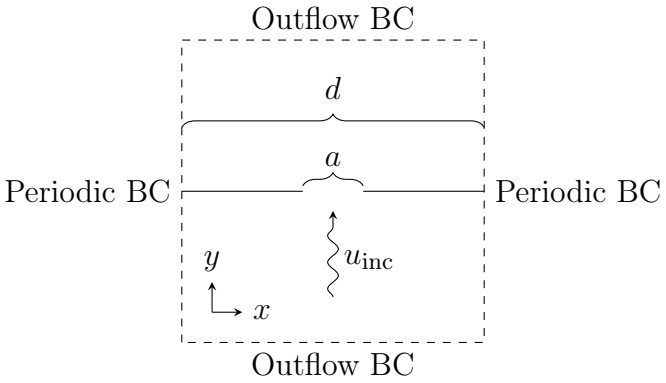


FIGURE 9. Periodic slit diffraction grating geometry

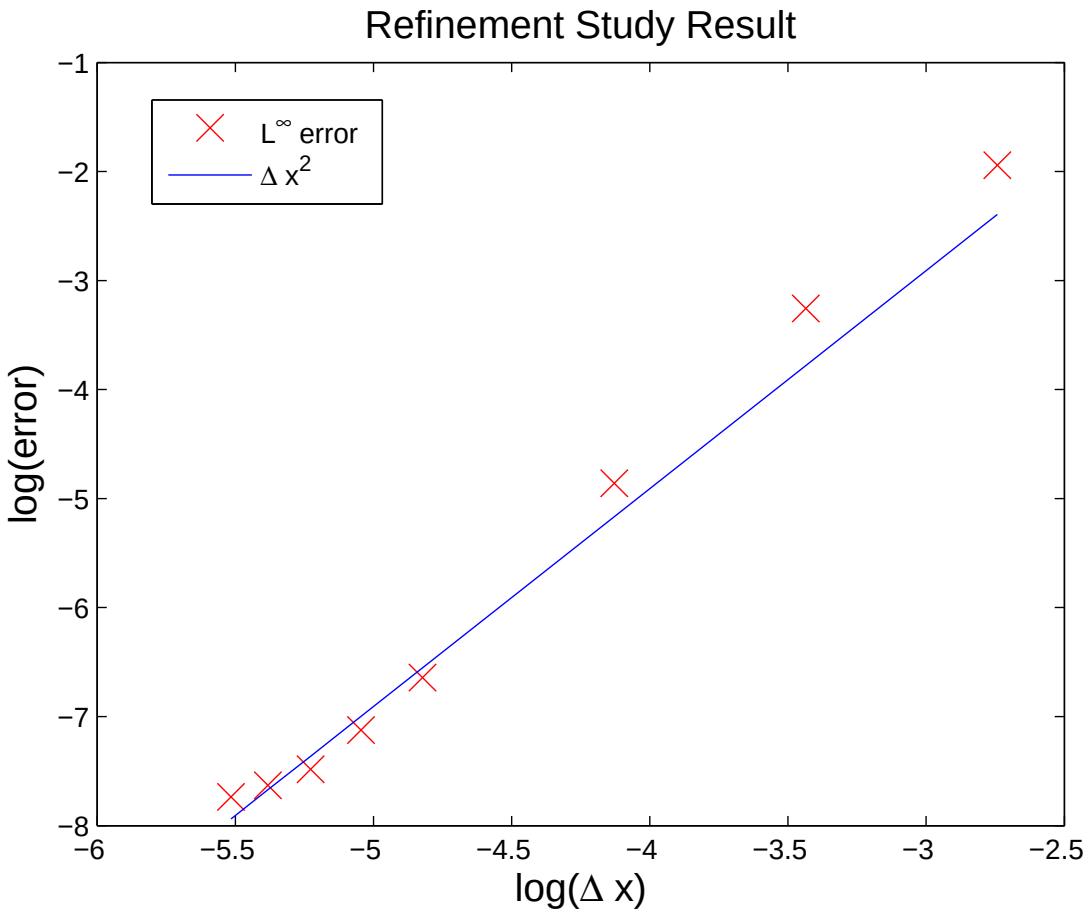
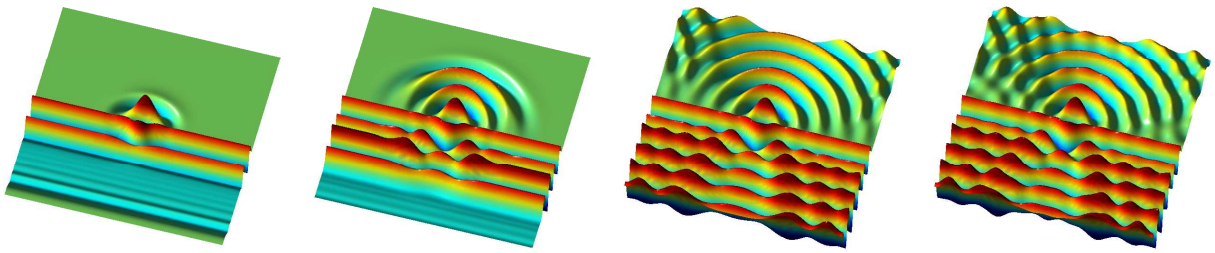


FIGURE 8. Refinement study for the Bessel mode in a circular domain with fixed CFL number of 2. Using quadratic boundary interpolant with bilinear interior interpolant.

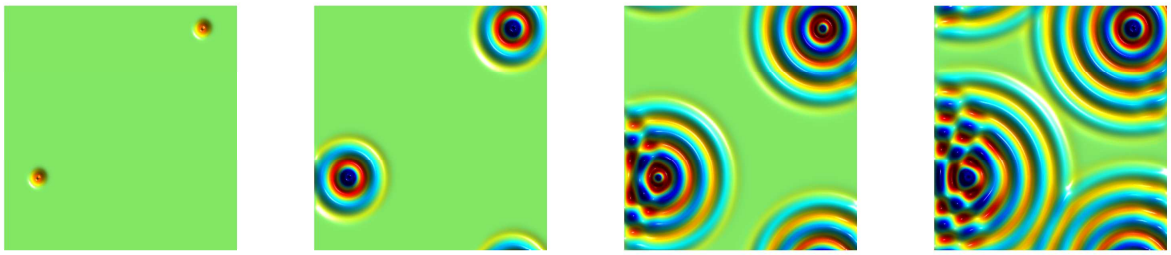


(a) $t = 0.31$ (b) $t = 0.51$ (c) $t = 1.01$ (d) $t = 2.01$

FIGURE 10. Evolution of the slit diffraction grating problem, with aperture width $a = 0.1$, grating periodicity $L_y = d = 1$, and wave speed $c = 1$. The CFL is fixed at 2.

In Figure 10, we observe the time evolution of the incident plane wave passing through the aperture, and the resulting interference patterns as the diffracted wave propagates across the periodic boundaries. The outflow boundary conditions allow the waves to propagate outside the domain. While a rigorous analysis of the efficacy of our outflow BCs is the subject of future work, the results look quite reasonable, as no spurious reflections are seen at the artificial boundaries.

5.5. **Point Sources and Outflow Boundary Conditions.** As a final example, we illustrate some of the more interesting features of our solver. We launch two point sources (from points not on the mesh), and employ Dirichlet, periodic and outflow boundary conditions along the edges of the domain. As can be seen in Figure 11, the point sources propagate perfectly, despite not being placed at grid points.



(a) $t = 0.05$ (b) $t = 0.40$ (c) $t = 0.75$ (d) $t = 1.00$

FIGURE 11. Point Sources emanating through the domain. Outflow boundary conditions are prescribed at the right, Dirichlet boundary conditions on the left, and the top and bottom edges are periodic.

6. Conclusion

In this paper we have presented a fast, A-stable, second order scheme for solving the wave equation. Using the method of lines transpose (MOL^T), the solution can be interpreted in the semi-discrete sense as a boundary integral solution, posed as a convolution against an exponential kernel. We have exploited this fact to develop a matrix-free, $O(N)$ fast spatial convolution algorithm, capable of embedding boundary points in a regular Cartesian mesh, without affecting the accuracy or stability.

In addition to demonstrating second order accuracy for wave propagation in a variety of non-Cartesian geometries, with time steps in excess of the usual CFL restriction, we have also developed a novel method for implementing outflow boundary conditions, as well as methods for launching waves, using soft sources, from points located at arbitrary locations (e.g., not located at a mesh point) inside the domain, which is of interest in particle-wave simulations such as those required in studying plasma. Several topics warrant further investigation. We are developing a domain decomposition approach to multi-core computing with our implicit wave solver based on the properties of the exponential kernel, where the subdomains require only pointwise data communication from adjoining edges. Future work will investigate the implementation of higher-order accurate boundary conditions for complex

boundary geometries and outflow boundary conditions, and further, apply these methods to Maxwell's equations both in electromagnetic scattering and plasma physics problems.

Appendix A. Summary Table For Second-Order Wave Solver

Wave Equation $\frac{1}{c^2}\frac{\partial u}{\partial t} - \nabla^2 u = S(x,t)$ To Modified Helmholtz Equation	Dispersive Scheme, $\alpha = \frac{2}{c\Delta t}$: $\left(-\frac{1}{\alpha^2}\nabla^2 + 1\right)\left(u^{n+1} + 2u^n + u^{n-1}\right) = 4u^n + \frac{4}{\alpha^2}S(x,t^n)$ Diffusive Scheme, $\alpha = \frac{\sqrt{2}}{c\Delta t}$: $\left(-\frac{1}{\alpha^2}\nabla^2 + 1\right)u^{n+1} = \frac{1}{2}\left(5u^n - 4u^{n-1} + u^{n-2}\right) + \frac{1}{\alpha^2}S(x,t^{n+1})$
Dimensionally Split Modified Helmholtz Equation (2D)	$\left(-\frac{1}{\alpha^2}\nabla^2 + 1\right)u = f \Rightarrow$ $\left(-\frac{1}{\alpha^2}\frac{\partial^2}{\partial x^2} + 1\right)\left(-\frac{1}{\alpha^2}\frac{\partial^2}{\partial y^2} + 1\right)u = f \Rightarrow$ $\left(-\frac{1}{\alpha^2}\frac{\partial^2}{\partial x^2} + 1\right)w = f, \quad \left(-\frac{1}{\alpha^2}\frac{\partial^2}{\partial y^2} + 1\right)u = w$
1D Integral Solution	$\left(-\frac{1}{\alpha^2}\frac{d^2}{dx^2} + 1\right)u = f$ on $(a,b) \Rightarrow$ $u(x) = \frac{\alpha}{2}\int_a^b f(x')e^{-\alpha\lvert x-x'\rvert}\,dx' + Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}$ $= I[f](x) + Ae^{-\alpha(x-a)} + Be^{-\alpha(b-x)}$
1D BC Correction Coefficients Dirichlet: $u(a) = u_a,\ u(b) = u_b$ Neumann: $u'(a) = v_a,\ u'(b) = v_b$ Periodic: $u(a) = u(b),\ u'(a) = u'(b)$	$A = \frac{(u_a - I_a) - \mu(u_b - I_b)}{1-\mu^2},\ B = \frac{(u_b - I_b) - \mu(u_a - I_a)}{1-\mu^2}$ $A = \frac{\mu(v_b + \alpha I_b) - (v_a - \alpha I_a)}{\alpha(1-\mu^2)},\ B = \frac{(v_b + \alpha I_b) - \mu(v_a - \alpha I_a)}{\alpha(1-\mu^2)}$ $A = \frac{I_b}{1-\mu},\ B = \frac{I_a}{1-\mu}$ $I_a = I[f](a),\ I_b = I[f](b),\ \mu = e^{-\alpha(b-a)}$
Fast Convolution Algorithm	$a = x_0 < x_1 < \cdots < x_N = b,$ $x_{j+1} - x_j + \Delta x,\ j = 0, ..., N-1$ $I_j = I[f](x_j) = \frac{\alpha}{2}\int_a^b f(x')e^{-\alpha\lvert x_j - x'\rvert}\,dx' = I_j^L + I_j^R$ $I_j^L = \frac{\alpha}{2}\int_a^{x_j} f(x')e^{-\alpha\lvert x_j - x'\rvert}\,dx',\ I_j^R = \frac{\alpha}{2}\int_{x_j}^b f(x')e^{-\alpha\lvert x_j - x'\rvert}\,dx'$ $I_0^L = 0,\ I_j^L = e^{-\alpha\Delta x}I_{j-1}^L + J_j^L,$ $J_j^L = \frac{\alpha}{2}\int_{x_{j-1}}^{x_j} f(x')e^{-\alpha\lvert x_j - x'\rvert}\,dx',\ j = 1, ..., N$ $I_N^R = 0,\ I_j^R = e^{-\alpha\Delta x}I_{j+1}^R + J_j^R,$ $J_j^R = \frac{\alpha}{2}\int_{x_j}^{x_{j+1}} f(x')e^{-\alpha\lvert x_j - x'\rvert}\,dx',\ j = N-1, ..., 0$
Second Order Quadrature	$J_j^R = Pf(x_j) + Qf(x_{j+1}) + R(f(x_{j+1}) - 2f(x_j) + f(x_{j-1}))$ $J_j^L = Pf(x_j) + Qf(x_{j-1}) + R(f(x_{j+1}) - 2f(x_j) + f(x_{j-1}))$ $P = 1 - \frac{1-d}{\nu},\ Q = -d + \frac{1-d}{\nu},\ R = \frac{1-d}{\nu^2} - \frac{1+d}{2\nu}$ $\nu = \alpha\Delta x,\ d = e^{-\nu}$

Appendix B. Treatment of point sources, and soft sources

We now consider the inclusion of source terms. We present the algorithm in 1D for simplicity, and observe that the extensions to multiple dimensions are analogous to those shown for dimensional splitting presented in section 2.

We are predominantly interested in the case where $S(x,t)$ consists of a large number of time dependent point sources. However, it is often the case that in electromagnetics problems, a soft source is prescribed to excite waves of a prescribed frequency, or range of frequencies,

within the domain. A soft source is so named because, although incident fields are generated at a prescribed fixed spatial location, no scattered fields are generated.

The implementation of a soft source $\sigma(t)$ at $x = x_s$ is accomplished by prescribing the source condition

$$u(x_s, t) = \sigma(t). \tag{48}$$

However, it can be shown that if we set

$$S(x,t) = \frac{2}{c}\sigma'(t)\delta(x - x_s) \tag{49}$$

and insert it into the wave equation, then the soft source condition (48) is satisfied, and the solutions are equivalent. Thus, a soft source is nothing more than a point source, whose time-varying field is integrated by the wave equation.

Upon convolving this source term with the Green's function according to (4), we find

$$\begin{aligned} I\left[\frac{1}{\alpha^2}S\right](x) =& \frac{1}{2\alpha}\int_a^b \left(\frac{2}{c}\sigma'(t_n)\delta(x - x_s)\right) e^{-\alpha|x-y|}dy \\ =& \frac{\Delta t}{\beta}\sigma'(t_n)e^{-\alpha|x-x_s|}, \end{aligned}$$

where the definition of $\alpha = \beta/(c\Delta t)$ has been utilized.

Remark 5. *It is often the case that taking the analytical derivative $\sigma'(t_n)$ is to be avoided, for various reasons. In this case, any finite difference approximation which is of the desired order of accuracy can be substituted.*

Likewise for general point sources,

$$S(x,t) = \sum_i \tilde{\sigma}_i(t)\delta(x - x_i)$$

the corresponding form of the source term is

$$I\left[\frac{1}{\alpha^2}S\right](x) = \frac{c\Delta t}{2\beta}\sum_i \tilde{\sigma}_i(t_n)e^{-\alpha|x-x_i|} \tag{50}$$

Therefore, it suffices to consider delta functions both for the implementation of soft sources, as well as including time dependent point sources.

References

1. B. Alpert, L. Greengard, and T. Hagstrom, *An Integral Evolution Formula for the Wave Equation*, Journal of Computational Physics **162** (2000), no. 2, 536–543.
2. ———, *Nonreflecting Boundary Conditions for the Time-Dependent Wave Equation*, Journal of Computational Physics **180** (2002), no. 1, 270–296.
3. J. Barnes and P. Hut, *A hierarchical O(N log N) force-calculation algorithm*, Nature **324** (1986), 446–449.
4. C.K. Birdsall and A.B. Langdon, *Plasma physics via computer simulation*, Course notes for electrical engineering and computer sciences, McGraw-Hill, 1976.
5. G. D. Birkhoff, *General mean value and remainder theorems with applications to mechanical differentiation and quadrature*, Transactions of the American Mathematical Society **7** (1906), no. 1, pp. 107–136 (English).
6. O. P. Bruno and M. Lyon, *High-order unconditionally stable FC-AD solvers for general smooth domains I . Basic elements*, Journal of Computational Physics **229** (2010), no. 6, 2009–2033.

7. M. Causley and A. Christlieb, *High order A-stable schemes for the wave equation using successive convolution*, SIAM Journal of Numerical Analysis **52** (2014), no. 1, 220–235.
8. M. Causley, A. Christlieb, B. Ong, and L. Van Groningen, *Method of Lines Transpose: An Implicit Solution to the Wave Equation*, Mathematics of Computation **83** (2014), no. 290, 2763–2786.
9. M. F. Causley, H. Cho, A. J. Christlieb, and D. C. Seal, *Method of lines transpose: High order L-stable O(N) schemes for parabolic equations using successive convolution*, ArXiv e-prints (2015).
10. H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys. **155** (1999), no. 2, 468–498.
11. H. Cheng, J. Huang, and T. J. Leiterman, *An adaptive fast solver for the modified Helmholtz equation in two dimensions*, Journal of Computational Physics **211** (2006), no. 2, 616–637.
12. A. J. Christlieb, R. Krasny, and J. P. Verboncoeur, *A treecode algorithm for simulating electron dynamics in a Penning-Malmberg trap*, Computer Physics Communications **164** (2004), no. 1-3, 306–310.
13. R. Coifman, V. Rokhlin, and S. Wandzura, *The fast multipole method for the wave equation: A pedestrian prescription*, IEEE Trans. Antennas and Propagation **35** (1993), no. 3, 7–12.
14. B. Fornberg, *A Short Proof of the Unconditional Stability of the ADI-FDTD Scheme*, **9810751**, 5–8.
15. J. Fornberg, B.and Zuev and J. Lee, *Stability and accuracy of time-extrapolated ADI-FDTD methods for solving wave equations*, **9810751**, no. November 2005.
16. Z. Gimbutas and V. Rokhlin, *A generalized fast multipole method for nonoscillatory kernels*, SIAM J. Sci. Comput. **24** (2002), no. 3, 796–817.
17. L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys. **73** (1987), no. 2, 325–348.
18. H. O. Kreiss, N. A. Petersson, and J. Yström, *Difference approximations of the neumann problem for the second order wave equation*, SIAM Journal on Numerical Analysis **42** (2004), no. 3, 1292–1323.
19. J. R. Li, *Low order approximation of the spherical nonreflecting boundary kernel for the wave equation*, Linear Algebra and its Applications **415** (2006), no. 2-3, 455–468.
20. P. Li, H. Johnston, and R. Krasny, *A Cartesian treecode for screened coulomb interactions*, Journal of Computational Physics **228** (2009), no. 10, 3858–3868.
21. K. Lindsay and R. Krasny, *A Particle Method and Adaptive Treecode for Vortex Sheet Motion in Three-Dimensional Flow*, Journal of Computational Physics **172** (2001), no. 2, 879–907.
22. M. Lyon and O. P. Bruno, *High-order unconditionally stable FC-AD solvers for general smooth domains II . Elliptic , parabolic and hyperbolic PDEs ; theoretical considerations*, Journal of Computational Physics **229** (2010), no. 9, 3358–3381.
23. T. Namiki, *3-d adi-fdtd method-unconditionally stable time-domain algorithm for solving full vector maxwell's equations*, Microwave Theory and Techniques, IEEE Transactions on **48** (2000), no. 10, 1743–1748.
24. D. N. Smithe, J. R. Cary, and J. A. Carlsson, *Divergence preservation in the adi algorithms for electromagnetics*, J. Comput. Phys. **228** (2009), no. 19, 7289–7299.

Mathematics Department, Kettering University, Flint, MI 48504
E-mail address: mcausley@kettering.edu

Department of Mathematics, Michigan State University, East Lansing, MI 48824
E-mail address: andrewch@math.msu.edu

Department of Mathematics, Michigan State University, East Lansing, MI 48824
E-mail address: wolferi@math.msu.edu