A FOURTH-ORDER ACCURATE EMBEDDED BOUNDARY METHOD FOR THE WAVE EQUATION

DANIEL APPELÖ[†] AND N. ANDERS PETERSSON[§][¶]

Abstract. A fourth-order accurate embedded boundary method for the scalar wave equation with Dirichlet or Neumann boundary conditions is described. The method is based on a compact Pade-type discretization of spatial derivatives together with a Taylor series method (modified equation) in time. A novel approach for enforcing boundary conditions is introduced which uses interior boundary points instead of exterior ghost points. This technique removes the small-cell stiffness problem for both Dirichlet and Neumann boundary conditions, is more accurate and robust than previous methods based on exterior ghost points, and guarantees that the solution is single-valued when slender bodies are treated. Numerical experiments are presented to illustrate the stability and accuracy of the method as well as its application to problems with complex geometries.

Key words. wave equation, embedded boundary, finite differences

AMS subject classifications. 35L45, 35B35

1. Introduction.

Finite difference discretizations with the boundary embedded in a Cartesian grid (EB-methods) provide an alternative to body-fitted overset, multi-block, or unstructured grid methods for solving partial differential equations in complex geometries. The geometry in an EB-method is represented by curves in 2D and surfaces in 3D, rather than by surface or volumetric grids, which significantly reduces storage requirements. For most EB-methods, the geometry only needs to be accessed while configuring the boundary condition stencils, which can be performed during a preprocessing stage. The data-structure for the boundary condition stencils is simple and all information can be stored in regular arrays. The configuration requires only local operations, making the method well-suited for parallel implementation. Consequently, no costly parallel grid generation is needed, as the Cartesian grid is trivially generated.

For wave propagation problems, EB-methods generally have small dispersion errors due to the perfect regularity of the Cartesian grid. They are also highly efficient in terms of floating point operations and memory use per degree of freedom. Despite these attractive features relatively few high-order accurate EB-methods for wave equations have been documented in the literature.

This paper presents a fourth-order accurate EB-method for the wave equation based on compact (Pade-type) finite-difference approximations of spatial derivatives combined with a modified equation approach for the time-discretization. The method imposes boundary conditions by assigning solution values to *boundary points* inside the boundary via interpolation, rather than using extrapolation to assign solution values to *ghost points outside* the boundary as was done in [8, 9, 10]. This subtle yet crucial difference improves previous second-order accurate approaches by removing the small-cell stiffness problem for both Neumann and Dirichlet boundary conditions. Moreover, placing *boundary points* inside the computational domain allows the solution to be "single-valued" for slender geometries, leading to significant algorithmic

[†]Computational and Applied Mathematics, California Institute of Technology, CA 91125

[§]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551

[¶]This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This is contribution LLNL-JRNL-417163 simplifications for complex geometries. A third advantage of placing *boundary points* inside the computational domain is that the boundary conditions are accounted for via interpolation rather than extrapolation, yielding smaller errors and better stability properties. To ensure long-time stability, a small amount of artificial dissipation is added.

EB-methods have been used to successfully solve a variety of problems from elasticity [29] to incompressible [23] and compressible flows [22]. While this paper will not review specific applications it will mention noteworthy papers on high-order-accurate EB-methods for wave equations. Using the integral evolution formula derived by Alpert, Greengard and Hagstrom [1], Li and Greengard [15] developed high-order (two to six) discretizations for the constant coefficient wave equation. Li et al. first computes a provisional solution ignoring the boundary conditions, and then solves an integral equation in order to correct the provisional solution. Another family of high-order (two to eight) accurate discretizations based on the same integral evolution formula was suggested by Wandzura et al. in [27, 28]. In this case a solution is first obtained by neglecting the boundary conditions and then corrected by a boundary projection. The discretization of the integral evolution is performed by a least squares fit constrained by order of accuracy requirements. Wandzura et al. approach the issue of stability in a novel way; if a particular discretization is not stable, they include more neighboring points until it is.

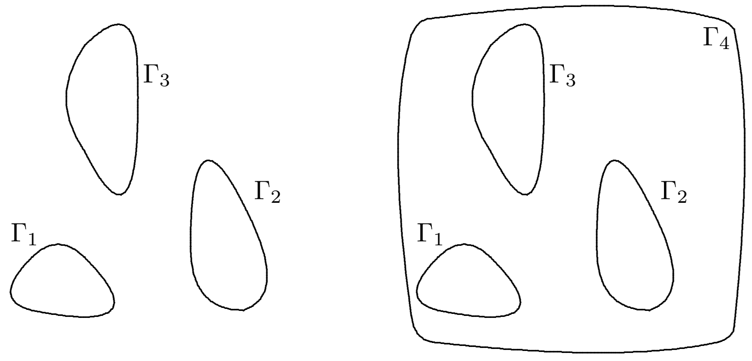
Lombard and Piraux [16] considers coupled acoustic and elastic waves and derives interface- and compatibility-conditions on the solution and its derivatives. Special discretizations near the boundary are generated using a least square procedure. The method presented in [16] is second-order accurate, but has been extended to fourth-order in [17].

Lyon and Bruno [18, 19, 4] proposed a framework based on Fourier-continuation and alternating direction methods, and devised high-order accurate discretizations for the scalar wave equation and the heat equation. Their method uses Fourier series representations of non-periodic functions to solve boundary value problems arising in ADI formulations. High-order-accuracy in time is achieved by Richardson extrapolation.

The proposed method consists of three distinct building blocks: boundary condition enforcement, spatial discretization, and temporal discretization. Each of these blocks can be modified independently of the others to suit a particular applications. For example, it would be straight forward to handle variable coefficients or mixed Neumann and Dirichlet boundary conditions. Our method is therefore more flexible than the methods suggested in Li et al. and Wandzura et al., which both are designed solely for the constant coefficient wave equation, or the unconditionally stable FC-AD solvers in [18], which assume Dirichlet boundary conditions.

The reminder of the paper is outlined as follows. §2 describes the aforementioned method in detail, including the pre-computations required to enforce the boundary conditions (§2.1), the temporal (§2.2) and spatial discretizations (§2.3), as well as various artificial dissipation operators included to ensure long-time stability (§2.4). In §3, several numerical experiments illustrating the stability and accuracy of the method are presented. Results are compared to previously published data and applications of the method to complex geometries are given. A summary and outline of future work is provided in §4.

2. Description of the method. We consider the wave equation in a two-dimensional domain $(x, y) \in \Omega$ in an isotropic medium with a constant speed of



(a) Exterior problem. (b) Interior problem.

FIG. 2.1. *Possible setups. Note that the setup (a) must be augmented by boundary conditions at infinity.*

sound (for simplicity set to one)

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f, \quad (x, y) \in \Omega,\ t \geq 0, \tag{2.1}$$

with initial data

$$u(x, y, 0) = g_0(x, y), \quad \frac{\partial u}{\partial t}(x, y, 0) = g_1(x, y), \quad (x, y) \in \Omega, \tag{2.2}$$

and boundary conditions of Dirichlet

$$u(x, y, t) = h_{\mathcal{D}}^{(i)}(x, y, t), \quad (x, y) \in \Gamma_l,\ t \geq 0,\ l = 1, \dots, n_{\mathcal{D}}, \tag{2.3}$$

or Neumann type

$$\frac{\partial u}{\partial n}(x, y, t) = h_{\mathcal{N}}^{(i)}(x, y, t), \quad (x, y) \in \Gamma_l,\ t \geq 0,$$
$$l = n_{\mathcal{D}} + 1, \dots, n_{\mathcal{D}} + n_{\mathcal{N}} = n_{\text{tot}}. \tag{2.4}$$

The boundary of the simply connected domain Ω is a collection of n_{tot} smooth curves Γ_l. For an exterior problem the curves Γ_l must be augmented by a boundary condition at infinity or by a non-reflecting boundary condition. For an interior problem one curve encloses the $n_{tot} - 1$ other curves, as shown in Figure 2.1.

2.1. Pre-computations. To find an approximate solution to equations (2.1)-(2.4) we assume, without restriction, that all curves describing the geometry are contained inside a rectangular domain $(x, y) \in \Omega = [0, L_x] \times [0, L_y]$ and cover Ω with the uniform grid (see Figure 2.2)

$$(x_i, y_j) = (ih, jh), \quad i = 0, \dots, n_x,\ j = 0, \dots, n_y.$$

The grid size $h > 0$ and the number of grid points are chosen such that $x_{n_x} = L_x$ and $y_{n_y} = L_y$. We denote a time dependent grid function by $u_{ij}(t) = u(x_i, y_j, t)$. Time is discretized on a equidistant grid $t_n = nk$ with time-step $k > 0$, and a time discrete grid function is denoted $u_{ij}^n = u(x_i, y_j, t_n)$.

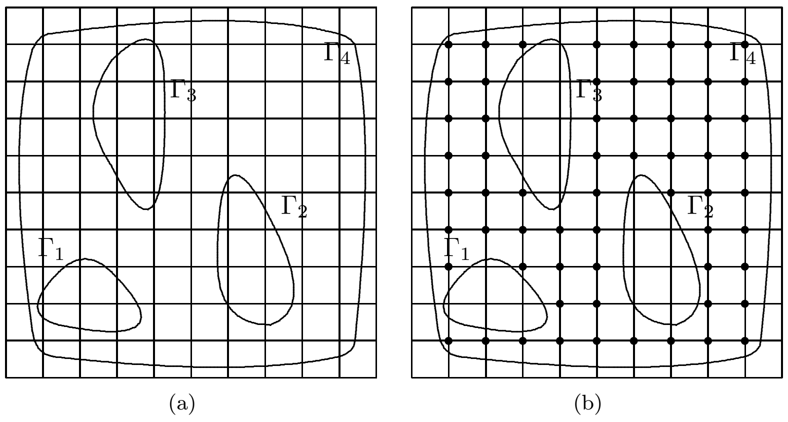


FIG. 2.2. *Discretization of the geometry without the mask shown (a) and with the mask shown (b), dots indicate $m_{ij} = 1$.*

The solution to (2.1) is sought only at grid points inside Ω. In a pre-computation step a mask grid function $m_{i,j}$ is set up such that

$$m_{i,j} = \begin{cases} 1, & (x_i, y_j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

The mask is particularly easy to set up when the boundary of Ω is defined by the implicit representation $\phi(x, y)$=0. In this case, $m_{i,j}$ simply follows by the sign of $\phi(x_i, y_j)$. When the boundary curves have a parametric representation, we start by setting $m_{i,j} = -1$ at all grid points to indicate that they are undefined. For an external problem, the outer edges of the grid are defined as inside by setting $m_{i,j} = 1$ along the lines $i = 0$, $i = n_x$, $j = 0$, and $j = n_y$. Correspondingly, for an internal problem, $m_{i,j} = 0$ on the outer edges. The mask is then defined in two stages. For each boundary curve Γ_l, we first identify all grid cells that are intersected by the boundary. The grid points at the corners of these grid cells are then marked as either inside or outside of Ω by setting m_{ij} to 1 or 0. After this step, each boundary curve Γ_l has a closed polygon of grid points with $m_{i,j} = 1$ just inside of Ω and a corresponding closed polygon of grid points with $m_{i,j} = 0$, just outside of Ω. In the interior of Ω, the mask can now be defined by "sweeping" line-by-line, i.e., for each horizontal grid line $j = 0, 1, \ldots, n_y$, we start at $i = 1$ and overwrite any undefined mask values according to

$$\text{if } m_{i,j} = -1, \text{ assign } m_{i,j} := m_{i-1,j}, \quad i = 1, 2, \ldots, n_x.$$

The same procedure is then repeated for each vertical grid line. An example of a mask grid function is shown in Figure 2.2.

Boundary conditions are enforced by assigning values to the grid points on the fringe of the computational domain Ω. These points are denoted *boundary points.* A *boundary point* (x_p, y_q) is distinguished by the following criterion: $m_{p,q} = 1$ and $m_{p+1,q} + m_{p-1,q} + m_{p,q+1} + m_{p,q-1} < 4$, i.e., the boundary point is inside Ω, but at least one of its nearest neighbors is outside. The *boundary points* define x- and *y-line segments* upon which approximations to the spatial derivatives in (2.1) are computed. For example, an *x-line segment* is defined as the collection of grid points

(x_p, y_q), $p = p_1, \dots, p_2$ with $m_{p,q} = 1$, starting and ending at *boundary points*, (x_{p_1}, y_q) and (x_{p_2}, y_q), respectively. The *boundary points* and the *line segments* are found by inspecting the mask.

To find approximations to spatial derivatives at all points inside the computational domain it is sufficient to know all *line segments*. However, to also account for boundary conditions some additional pre-computations must be performed to determine the stencil and coefficients in the formula for assigning solution values at each *boundary point*. The procedure for setting up such *boundary stencils* depends on the type of boundary conditions and the approach taken to enforce them. For Dirichlet boundary conditions a one-dimensional line-by-line approach can be used which is described in §2.1.1. The advantage of this approach is ease of implementation and a slightly smaller error constant than the more general approach described in §2.1.2. The latter technique, based on interpolation in the direction normal to the boundary, handles both Dirichlet and Neumann boundary conditions.

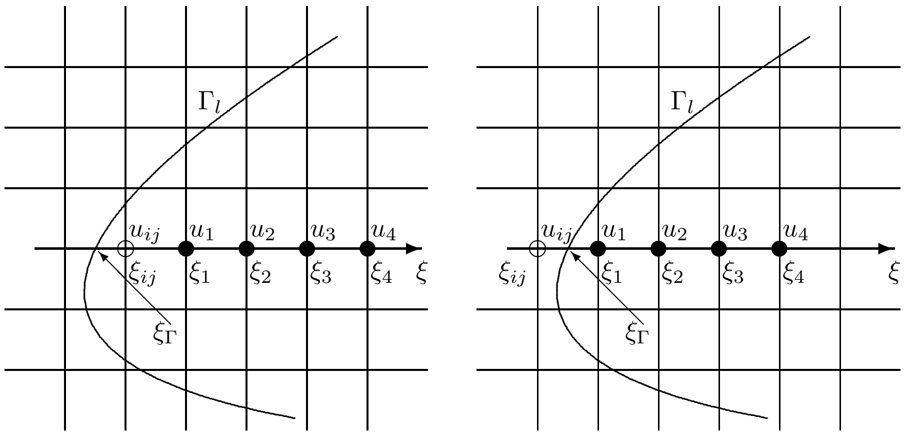


FIG. 2.3. *Enforcing Dirichlet boundary conditions by a line by line approach using interior* boundary points *(left) or exterior* ghost points *(right).*

2.1.1. Enforcing Dirichlet boundary conditions along grid lines. Let (x_i, y_j) be a *boundary point* associated with a boundary Γ_l where the solution is prescribed. To assign a value to u_{ij} we introduce a local one-dimensional coordinate system ξ along the grid line in x passing through (x_i, y_j) (left image of Figure 2.3), and construct an interpolating polynomial

$$\mathcal{I}u(\xi) = u_{ij} g_{ij}(\xi) + \sum_{\nu=1}^{4} u_\nu g_\nu(\xi). \tag{2.5}$$

Here $g_{ij}(\xi)$, $g_\nu(\xi)$ are the coefficients in the usual Lagrange polynomial basis. Now, by equating the interpolant evaluated on the boundary with the right hand side of the boundary condition

$$\mathcal{I}u(\xi_\Gamma) = h_\mathcal{D}(x_\Gamma, y_\Gamma, t),$$

we obtain an explicit expression for u_{ij}

$$u_{ij} = \frac{1}{g_{ij}(\xi_\Gamma)} \left(h_\mathcal{D}(x_\Gamma, y_\Gamma, t) - \sum_{\nu=1}^{4} u_\nu g_\nu(\xi_\Gamma) \right). \tag{2.6}$$

Note that once the intersection of the boundary, ξ_Γ, is found (e.g. by using a root-finding algorithm such as the secant method) the numbers $g_{ij}(\xi_\Gamma)$, $g_\nu(\xi_\Gamma)$, $\nu = 1,\ldots,4$, do not depend on time and can be pre-computed and stored. Also note that the truncation error of (2.6) is of order $\mathcal{O}(h^5)$. For convenience of implementation a fifth-order interpolant is used throughout this paper. For Dirichlet boundary conditions this yields a fifth-order accurate *boundary stencil* but for Neumann boundary conditions, which are based on the derivative of the interpolant (2.5), it yields a fourth-order accurate *boundary stencil*.

The placement of the *boundary point* is the subtle yet important distinction from previous papers [8, 9, 10]. In previous work the *ghost point* is placed outside the computational domain as pictured in the right image of Figure 2.3. This placement has the disadvantage that the denominator of (2.6):

$$g_{ij}(\xi_\Gamma) = \frac{\xi_\Gamma - \xi_1}{\xi_{ij} - \xi_1} \prod_{\nu=2}^{4} \frac{\xi_\Gamma - \xi_\nu}{\xi_{ij} - \xi_\nu},$$

may become arbitrary small when ξ_Γ is close to ξ_1 causing small-cell stiffness in an explicit time stepping procedure.

In contrast, for the approach suggested above, $\xi_\Gamma \le \xi_{ij} < \xi_1 = \xi_{ij} + h$, and thus

$$|\xi_\Gamma - \xi_1| \ge |\xi_{ij} - \xi_1| = h,$$

so $g_{ij}(\xi_\Gamma)$ is always bounded away from zero. This separation is an immediate consequence of using a Lagrange polynomial basis, i.e., all zeros of $g_{ij}(\xi)$ have $\xi \ge \xi_1 = \xi_{ij} + h$. Hence, placing the *boundary point* inside the computational domain removes the small-cell stiffness problem.

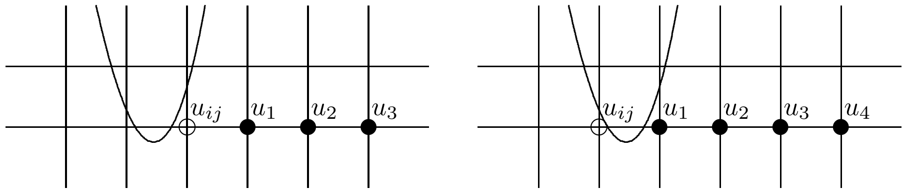


FIG. 2.4. *An advantage with placing the boundary points inside the computational domain is that they will be "single-valued" even when the geometry is slender. When the boundary points are placed outside the computational domain the situation to the right can occur. To the right the solution is "multi-valued" and the values u_{ij} and u_1 are both interior and boundary points.*

Another advantage with placing the *boundary points* inside the computational domain is that they will be "single-valued" even when the geometry is slender. The situation illustrated in Figure 2.4 leads to algorithmic difficulties in that two copies of the solution must be stored in *ghost points* that also are inside Ω. Such special treatment leads to more complicated implementations, especially in higher dimensions, so the algorithmic simplification obtained by using "single-valued" *boundary points* is important in practical applications.

Finally, when the *boundary points* are placed inside the computational domain the formula (2.6) assigns the value to u_{ij} by interpolation rather than extrapolation. In general, interpolation is preferred over extrapolation both for reasons of accuracy and numerical stability.

REMARK 1. *Note that for the line-by-line approach each* boundary point *can be assigned either from data along a horizontal or vertical line. If the unit boundary* *normal has components* $\mathbf{n} = (n_1, n_2)^T$*, we use horizontal grid lines if* $|n_1| > |n_2|$*, and vertical grid lines in the opposite situation.*

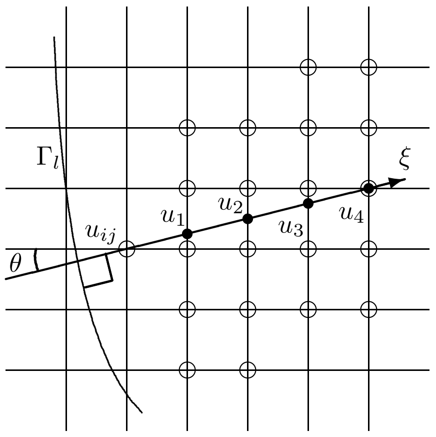


FIG. 2.5. *Enforcing the boundary conditions by constructing an interpolating polynomial in the normal direction. First the values of the solution at the empty circles are used to interpolate (vertically) values,* u_ν*,* $\nu = 2, \dots, 4$*, at the solid circles, then an interpolating polynomial is constructed along* ξ*. This polynomial is used to enforce the boundary condition as in the line by line approach.*

2.1.2. Enforcing the boundary conditions via interpolation in the normal direction. To enforce Neumann or Dirichlet boundary conditions in a *boundary point* (x_i, y_j), we find the straight line that passes through the *boundary point* and is normal to the boundary curve Γ_l, see Figure 2.5. Along that line a local coordinate ξ is introduced. When the angle, θ, between the line and the horizontal grid line passing through the *boundary point* is in the interval $0 \le \theta \le \pi/4$, temporary grid values, $u_1, \dots, u_4$, at the four next intersections with vertical grid lines are introduced. If $\pi/4 \le \theta \le \pi/2$, temporary values are introduced at the corresponding horizontal intersections (the three other quadrants are simply permutations of the first). The solution at the temporary grid points is found by interpolating vertical values. For example, in the situation depicted in Figure 2.5 the values would be given by

$$u_\nu = \sum_{q=q_l}^{q_h} \left(\prod_{\substack{p \neq q \\ p=q_l}}^{q_h} \frac{(\nu h \sin\theta - y_{j+p})}{(y_{j+q} - y_{j+p})} \right) u_{i+\nu\, j+q}.$$

Here q_l and q_h are chosen to center the interpolation stencil as well as possible around the intersection point $y = y_j + \nu h \sin\theta$, while only including interior grid points in the interpolation stencil.

Once the values $u_1, \dots, u_4$ are known, the formula (2.5) or its derivative with respect to ξ is used to find u_{ij} for Dirichlet

$$u_{ij} = \frac{1}{g_{ij}(\xi_\Gamma)} \left(h_{\mathcal{D}}(x_\Gamma, y_\Gamma, t) - \sum_{\nu=1}^{4} u_\nu g_\nu(\xi_\Gamma) \right), \tag{2.7}$$

or for Neumann boundary conditions

$$u_{ij} = \frac{1}{g'_{ij}(\xi_\Gamma)} \left(h_{\mathcal{N}}(x_\Gamma, y_\Gamma, t) - \sum_{\nu=1}^{4} u_\nu g'_\nu(\xi_\Gamma) \right). \tag{2.8}$$

As in the line by line approach, the construction of $g_{ij}(\xi)$ implies that its roots are well separated from ξ_Γ and there will be no small-cell stiffness. For Neumann boundary conditions, Rolle's theorem guarantees that the denominator of (2.8) is also always bounded away from zero.

Remark 2. *Temporary values are introduced at intersections between the grid and the normal in the above description but in the computer implementation of the boundary stencil only a list containing the location of the circled grid points and the weights at those points is stored.*

2.2. Approximation in time. Having described the pre-computations we now describe the inner-loop, starting with the temporal discretization. To get a fourth-order accurate approximation in time we use the *modified equation* [2, 6, 25] approach based on the Taylor expansion of $u(x, y, t)$ around the present time t_n, where for brevity we suppress the dependence on x and y,

$$u(t_{n+1}) \approx u(t_n) + \frac{k}{1!}u_t(t_n) + \frac{k^2}{2!}u_{tt}(t_n) + \frac{k^3}{3!}u_{ttt}(t_n) + \frac{k^4}{4!}u_{tttt}(t_n),$$
$$u(t_{n-1}) \approx u(t_n) - \frac{k}{1!}u_t(t_n) + \frac{k^2}{2!}u_{tt}(t_n) - \frac{k^3}{3!}u_{ttt}(t_n) + \frac{k^4}{4!}u_{tttt}(t_n).$$

A fourth-order approximation for $u(x, y, t_{n+1})$ with the local $\mathcal{O}(k^5)$ truncation error is obtained by adding the above equations

$$u(x, y, t_{n+1}) \approx 2u(x, y, t_n) - u(x, y, t_{n-1}) + k^2 u_{tt}(x, y, t_n) + \frac{k^4}{12}u_{tttt}(x, y, t_n).$$

The terms $u_{tt}(x, y, t_n)$ and $u_{tttt}(x, y, t_n)$ are replaced by spatial derivatives using the PDE (2.1). In particular, for smooth u satisfying (2.1) the following equality holds

$$\partial_t^2 u_{tt}(x, y, t_n) = \partial_t^2 \left(\Delta u(x, y, t_n) + f(x, y, t_n)\right) = \\ \Delta(\Delta u(x, y, t_n)) + \Delta f(x, y, t_n) + f_{tt}(x, y, t_n). \quad (2.9)$$

The evaluation of the right-hand-side of (2.9) requires finding approximations of various derivatives of u. Derivatives can be approximated by using a compact Padé scheme, as described in detail below, however other one-sided alternatives such as summation-by-parts approximations [11] are also possible.

Remark 3. *Note that the value of $u(x, y, -k)$ is needed to start the computation and can be obtained by further Taylor expansion and expressed in terms of the initial data and forcing as*

$$u(x, y, -k) \approx u(x, y, 0) - ku_t(x, y, 0) \\ + \frac{k^2}{2!}\left(\Delta u(x, y, 0) + f(x, y, 0)\right) - \frac{k^3}{3!}\left(\Delta u_t(x, y, 0) + f_t(x, y, 0)\right). \quad (2.10)$$

This approximation is sufficiently accurate to achieve a fourth-order accurate approximation of the solution. If highly accurate approximations of the gradient of the solution are needed, the slow-start procedure described in [8] should be used.

2.3. Approximation of spatial derivatives. Equation (2.9) contains both second and fourth derivatives of the solution. The temporal discretization is fourth-order accurate and to obtain an overall fourth-order accurate method, second derivatives must be approximated using a fourth-order accurate method. For simplicity of implementation, we also approximate the fourth derivatives by the same fourth-order accurate technique, even though it is sufficient to approximate these terms to second-order accuracy [2, 6, 25].

As mentioned above, the value of the solution is assigned at the *boundary points* of each line segment and no solution values are extrapolated to outside *ghost points*. Thus, an approximation of the derivatives that use only values on each line segment must be used. For this purpose we use the compact (or Padé) method, see Collatz [5] or Lele [13], which we outline now.

For a one-dimensional grid function u_i defined on a grid $x_i = ih$, $i = 0, \ldots, N$, the compact method approximates the second derivative of $u(x)$ by solving a banded system

$$(u_{xx})_i + \sum_{j=1}^{p} \alpha_j((u_{xx})_{i+j} + (u_{xx})_{i-j}) = \frac{1}{h^2}\left(\beta_0 u_0 + \sum_{l=1}^{q} \beta_l(u_{i+l} + u_{i-l})\right), \quad i = 1, 2, \ldots, N-2, N-1. \tag{2.11}$$

The coefficients α_j and β_l are found by equating the coefficients in front of increasing powers of h from the Taylor series expansions of $u(x)$ and $u_{xx}(x)$ around x_i. Here the classic fourth-order accurate Padé scheme [5] is used with $p = 1, q = 1$ and

$$\alpha_1 = 1/10, \quad \beta_0 = -12/5, \quad \beta_1 = 6/5.$$

The one-sided stencils

$$\begin{aligned}(u_{xx})_0 + 11(u_{xx})_1 &= \frac{1}{h^2}\left(13u_0 - 27u_1 + 15u_2 - u_3\right), \\ (u_{xx})_N + 11(u_{xx})_{N-1} &= \frac{1}{h^2}\left(13u_N - 27u_{N-1} + 15u_{N-2} - u_{N-3}\right),\end{aligned}$$

are used at the boundary points, x_0 and x_N.

Note that the above boundary closures only provide a third-order accurate approximation of u_{xx} at the boundary points. However, these values are not used by the time-stepping algorithm, since the boundary points are assigned values to satisfy the boundary conditions at the end of each time-step.

REMARK 4. *One drawback to compact schemes is that a system of equations has to be solved along each line segment. The main objection to solving those systems is not that it is time consuming (in fact, the most time consuming part is assembling the right hand side when v_{i+1}, v_i, v_{i-1} are not contiguous in memory), but rather that it is a global operation which complicates efficient parallelization. A common parallelization strategy for methods that use compact schemes is to transpose the global solution each time step. We argue that a better strategy would be to split the system of equations (2.11) on several CPUs realizing that the solution of the system can be computed approximately in any point by an explicit method to the same-order of accuracy.*

REMARK 5 (Time step restrictions). *On Cartesian grids with no embedded boundary the fourth-order modified equation time stepping has the advantage that the time steps can be larger than when a second-order method in time is used, see [2, 6, 25]. Unfortunately this advantage does not to carry over to the case when an embedded boundary is used. From numerical experiments with various geometries and grid sizes we have determined that the stability limit for the proposed method is* CFL ≈ 0.4. *For comparison,* CFL ≈ 0.9 *can be used without the embedded boundary.*

2.4. Artificial dissipation for long time simulations. Let $D^{\mathcal{P}}_{xx} u_{ij}$ denote the compact Padé approximation to $u_{xx}(x_i, y_j)$. Then the above described approximation of (2.1) can be written

$$\frac{u^{n+1}_{ij} - 2u^n_{ij} + u^{n-1}_{ij}}{k^2} = \left(D^{\mathcal{P}}_{xx} + D^{\mathcal{P}}_{yy}\right) u^n_{ij} + f^n_{ij} + \frac{k^2}{12}\left(\frac{f^{n+1}_{ij} - 2f^n_{ij} + f^{n-1}_{ij}}{k^2}\right)$$
$$+ \frac{k^2}{12}\left(\left(D^{\mathcal{P}}_{xx} D^{\mathcal{P}}_{xx} + 2D^{\mathcal{P}}_{yy} D^{\mathcal{P}}_{xx} + D^{\mathcal{P}}_{yy} D^{\mathcal{P}}_{yy}\right) u^n_{ij} + \left(D^{\mathcal{P}}_{xx} + D^{\mathcal{P}}_{yy}\right) f^n_{ij}\right),$$
$$\forall \{i, j \, : m_{ij} = 1\}. \quad (2.12)$$

When used together with an embedded boundary, the scheme (2.12) suffers from a weak instability and a small amount of artificial damping must be added to stabilize it. Here, we explore the use of three different damping terms:

$$\left[-d_4 h^3 \left(D_{4x} + D_{4y}\right) + d_6 h^5 \left(D_{6x} + D_{6y}\right) - d_8 h^7 \left(D_{8x} + D_{8y}\right)\right] \left(\frac{u^n_{ij} - u^{n-1}_{ij}}{k}\right), \quad (2.13)$$

that can be added to the right hand side of (2.12). The damping terms are approximations of $\frac{\partial^{2p}}{\partial x^{2p}} \frac{\partial u}{\partial t}$, $p = 2, 3, 4$ built from consistent approximations of $\frac{d^p}{dx^p}$, denoted by D_{px}. The explicit formulas are found in Appendix B, but generally

$$\frac{\partial^{2p}}{\partial x^{2p}} \frac{\partial u}{\partial t} \approx D^T_{px} B_p D^T_{px} \left(\frac{u^n_{ij} - u^{n-1}_{ij}}{k}\right).$$

The effect of artificial damping on the attainable order of accuracy for summation-by-parts discretizations on Cartesian grids was studied for first order systems in [21, 20]. It was found that the truncation error caused by the damping is of order $p + 1$ when the matrix B_p is chosen as the identity matrix. According to the theory in [20] the d_4-damping, which uses $B_2 = \text{diag}(0, 1, 1, \ldots, 1, 1, 0)$, should give second-order accuracy while the d_6- and d_8-damping, for which $B_3 = B_4 = I$, should give fourth- and fifth-order accuracy.

In §3 we experimentally study the convergence properties of the proposed method together with the different damping terms (denoted d_4-, d_6- or d_8-damping). Our findings using an EB-method are not entirely consistent with those of [20]. For example, we observe that the truncation-order of the damping is different for Dirichlet and Neumann boundary conditions. Also, for the d_4-damping we observe only second-order convergence when it is used together with Neumann boundary conditions, but third-order accurate when used together with Dirichlet boundary conditions.

3. Numerical experiments. In this section we present several experiments demonstrating the properties of the proposed method, as summarized in Algorithm 1.

Some of the results presented in this section will be compared to results presented in [9, 10] and it should be noted that the notation here differs slightly from that in [9, 10]. In particular, the integer N as used below, differs by one although the grid size h is the same as in [9, 10]. Note also that in §§3.1-3.3 we exclusively report errors in max-norm, but we report rate of convergence as

$$\text{rate} = \log_2 \frac{\|u_{\text{approx.}}(2h) - u_{\text{exact}}\|_\infty}{\|u_{\text{approx.}}(h) - u_{\text{exact}}\|_\infty},$$

while rate reported in [9, 10] is the fraction of consecutive errors without taking the logarithm (i.e. 4 here correspond to 16 in [9, 10]).

```
Data: Geometry, grid size $h$, time-step $k$, final time $t_{\text{end}}$
Result: The solution $u_{ij}^n$ at times $t = 0, k, \ldots, t_{\text{end}}$
begin
    Pre-computations:
    Setup mask, $m_{ij}$;
    Setup boundary points;
    Setup line segments;
    foreach boundary point do
        Setup boundary stencil;
    end
    Initial data:
    Assign initial data $u_{ij}^0$;
    Assign $u_{ij}^{-1}$ using (2.10);
    Time-stepping loop:
    for $t_n = nk$, $n = 0, \ldots, n_t$ do
        Assign values to $u_{ij}^n$ in all boundary points using the boundary stencils;
        Compute the right hand side of (2.12) and store in $F_{ij}^n$;
        Compute the damping terms (2.13) and store in $D_{ij}^n$;
        Compute $u_{ij}^{n+1} = 2u_{ij}^n - u_{ij}^{n-1} + k^2 F_{ij}^n + D_{ij}^n$;
        Cycle $u^{n-1} \leftarrow u^n$, $u^n \leftarrow u^{n+1}$;
    end
end
```

Algorithm 1: Fourth-order-accurate embedded boundary method for the wave equation.

3.1. Stability of the method. The purpose of this first experiment is to illustrate how we determine the values for d_4, d_6, d_8 that give a stable method. To do this we choose the forcing, initial and boundary conditions such that the solution is

$$u(x, y, t) = \sin(\omega(x - t)) \sin(\omega y), \quad \omega = 4\pi, \tag{3.1}$$

and solve (2.1) inside an ellipse with semi-axes $x_s = 1$, $y_s = 0.75$. The geometry is covered by the grid $(x_i, y_j) = (-1.1 + ih, -1.1 + jh)$, $i, j = 0, \ldots, N$, $h = 2.2/N$. The approximate solution is advanced to time $t = 100.0$ using $k/h \equiv \text{CFL} = 0.4$, and the max-norm error is monitored and used as an indicator of stability.

In Figure 3.1 the max-error as a function of time for a run without damping and for runs with $d_6 = 0.002$ and $d_6 = 0.0125$ is plotted. As is shown, the $d_6 = 0.0125$ simulation is stable. This computation was performed with $N = 200$ but in order to span as many scenarios for how the boundary can cut through the grid as possible, we repeat the procedure for $N = 400$ and $N = 800$. If all three grid sizes are stable for a certain value of the damping, it is accepted as an adequate value. Table 3.1 lists

Table 3.1
Values for the damping parameters that give a stable method.

B.c. method	d_4	d_6	d_8
Normal	0.0175	0.0125	0.002
Line-by-line	0.0175	0.0200	0.004

the smallest amounts of d_4, d_6 and d_8 damping that result in a stable method. These values have been determined by repeating the above experiment, enforcing boundary conditions either with the line-by-line approach or with the normal direction approach on grids with $N = 200, 400, 800$. The values given for the normal direction approach are suitable for both Neumann and Dirichlet boundary conditions.

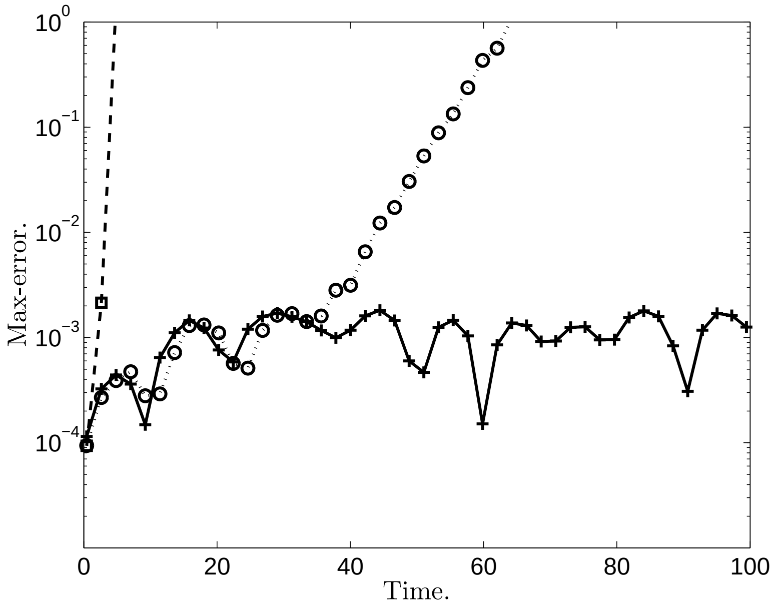


FIG. 3.1. *Stabilizing the method using the d_6-damping. The lines are: no damping (dashed with squares), $d_6 = 0.002$ (dotted with circles) and $d_6 = 0.0125$ (solid with plus signs).*

REMARK 6. *The values presented in Table 3.1 have successfully been used for numerous geometries and configurations. In our experience, suitable values for the damping can be determined in a single geometry as long as it is done using sufficiently fine grids. However, all calculations were performed with unit wave propagation speed and the damping coefficients must be scaled appropriately for other wave speeds.*

3.2. Convergence of a trigonometric exact solution. To study the convergence properties of the method for different boundary conditions and damping options, we continue to solve the problem described in §3.1 where the solution is given by (3.1). Using the values in Table 3.1 for the different damping terms, the solution is advanced and the max-error monitored to the end time $t_{\text{end}} = 2$ on grids with $N = 200$ and 400.

The results for Dirichlet conditions are displayed in Tables 3.2 and 3.3 and the results for Neumann conditions are displayed in Table 3.4. The output times for the error are the same as in [9, 10], enabling the comparison with the second-order and the interior fourth-order-accurate methods therein.

Comparing the results with $N = 200$ for the Dirichlet case with those in Table 1 and 2 in [9], we see that our compact fourth-order-accurate scheme achieves max-errors that are about an order of magnitude smaller than those of the interior fourth-order scheme. Not surprisingly, the improvements compared to the second-order scheme are even larger.

We find that for both approaches to enforce Dirichlet boundary conditions, the highest-order damping, d_8, gives the most accurate results, followed by the d_6-damping . The largest errors are obtained when d_4-damping is used. The line by line approach is consistently more accurate than the normal direction approach; this is expected since there is only a single interpolation in the line-by line approach. For both approaches the convergence rates are higher than expected for the d_4- and d_8-damping. When

TABLE 3.2
Convergence of the trigonometric exact solution (3.1) enforcing Dirichlet boundary conditions line-by-line. The results can be compared to Table 1 and 2 in [9].

	d_4			d_6			d_8		
t	200	400	rate	200	400	rate	200	400	rate
0.33	1.6(-4)	1.0(-5)	3.96	1.6(-4)	1.0(-5)	3.96	4.6(-5)	1.4(-6)	5.04
1.98	2.6(-4)	1.5(-5)	4.11	2.6(-4)	1.5(-5)	4.11	4.8(-5)	1.5(-6)	4.97
2.00	2.7(-4)	1.4(-5)	4.28	2.7(-4)	1.4(-5)	4.28	5.2(-5)	1.6(-6)	5.06

TABLE 3.3
Convergence of the trigonometric exact solution (3.1) enforcing Dirichlet boundary conditions by interpolation in the normal direction. The results can be compared to Table 1 and 2 in [9].

	d_4			d_6			d_8		
t	200	400	rate	200	400	rate	200	400	rate
0.33	2.6(-4)	2.6(-5)	3.31	1.3(-4)	8.6(-6)	3.91	8.7(-5)	3.3(-6)	4.71
1.98	2.8(-4)	2.3(-5)	3.59	2.2(-4)	1.0(-5)	4.43	1.5(-4)	4.3(-6)	5.16
2.00	2.9(-4)	2.2(-5)	3.71	1.8(-4)	8.9(-6)	4.31	1.5(-4)	4.3(-6)	5.16

TABLE 3.4
Convergence of the trigonometric exact solution (3.1) with Neumann boundary conditions. The results can be compared to Table 2 in [10].

	d_4			d_6			d_8		
t	200	400	rate	200	400	rate	200	400	rate
0.33	4.4(-3)	4.0(-4)	3.13	3.5(-3)	2.7(-4)	3.68	3.4(-3)	2.2(-4)	3.94
1.98	5.5(-3)	7.8(-4)	2.81	4.7(-3)	4.4(-4)	3.41	4.7(-3)	3.0(-4)	3.94
2.00	6.3(-3)	5.8(-4)	3.43	5.6(-3)	4.1(-4)	3.76	5.4(-3)	3.4(-4)	3.98

the d_8-damping is used the dominant error is likely the fifth-order interpolation.

For Neumann boundary conditions the results can be compared to the results in Table 2 in [10]. Here the compact fourth-order accurate scheme yields max-errors that are approximately five times smaller than those of the interior fourth-order scheme (denoted "predictor-corrector" in [10]).

The observed rates of convergence are lower compared to those observed for the Dirichlet case and the d_8-damping must be used to get a fourth-order accurate method. However it is interesting to note that the size of the errors are roughly comparable for all damping terms.

3.3. Convergence of inwards and outwards traveling waves. In this section we repeat the experiments with inwards and outwards traveling waves described in §§ 5.2 and 5.3 in [9] and in § 8 in [10]. In all examples the wave equation is solved inside a circle with internal forcing, initial data and boundary forcing chosen so that the solution becomes a radially propagating wave

$$u(x, y, t) = u(r, t) = \phi(r \pm t), \tag{3.2}$$

where

$$\phi(\xi) = \frac{1}{4}\left(1 + \tanh\frac{\xi - \xi_0}{\epsilon}\right)\left(1 - \tanh\frac{\xi - \xi_1}{\epsilon}\right).$$

In this section all boundary conditions are enforced via interpolation in the normal direction. We note that for these short-time simulations no artificial dissipation is required, but we still use d_6-damping for Dirichlet conditions and d_8-damping for Neumann conditions, as we believe they will be needed in many realistic long-time applications of the method.

3.3.1. Convergence of an inwards and outwards traveling wave with Dirichlet conditions. First, consider convergence of an inwards traveling wave with

$$u(x, y, t) = \phi(r + t), \quad \xi_0 = 2.2, \quad \xi_1 = 2.4, \quad \epsilon = 0.035,$$

in a circle $|r| \leq 2$ with Dirichlet boundary conditions. The circle is covered by a rectangle of size 4.2×4.2 discretized by a uniform grid with $h = 4.2/N$.

This problem is more challenging than the trigonometric exact solution as the wave has a much steeper gradient. The wave starts outside the domain and reaches its maximum on the boundary around $t \approx 0.3$ and then continues to propagate towards the center of the circle.

Table 3.5
Convergence results for an inwards traveling wave with Dirichlet boundary conditions.

	CFL = 0.4	CFL = 0.4		CFL = 0.1	CFL =0.1	
t	$N = 400$	$N = 800$	rate	$N = 400$	$N = 800$	rate
0.315	8.31(-3)	9.80(-4)	3.08	8.94(-3)	1.16(-3)	2.95
0.525	8.25(-3)	9.77(-4)	3.08	9.13(-3)	1.15(-3)	2.99
1.155	8.96(-3)	6.87(-4)	3.71	1.02(-2)	8.26(-4)	3.62
1.365	9.89(-3)	7.07(-4)	3.81	1.13(-2)	8.72(-4)	3.69

In Table 3.5 results from simulations using two CFL numbers, CFL = 0.4 and CFL = 0.1, and two discretizations, $N = 400$ and $N = 800$, are presented. For this experiment the errors are comparable with those presented in [9] (Table 3 and 4). This supports the argument made in [9] that for certain problems, consisting of waves residing mainly in the interior, the interior fourth-order correction can be quite effective.

Note that the rates of convergence are lower than four at the earlier times when the wave is entering the domain. We expect that the rates of convergence would approach four if the grid would be refined further, and that the errors for the compact method would be much smaller than those of the interior fourth-order method.

Table 3.6
Convergence results for an outwards traveling wave with Dirichlet boundary conditions.

	CFL = 0.4	CFL = 0.4		CFL = 0.1	CFL =0.1	
t	$N = 200$	$N = 400$	rate	$N = 200$	$N = 400$	rate
0.22	1.39(-3)	8.60(-5)	4.02	1.68(-3)	1.02(-4)	4.05
0.33	1.94(-3)	1.24(-4)	3.96	2.32(-3)	1.46(-4)	3.98
0.66	1.27(-2)	8.15(-4)	3.97	1.39(-2)	9.76(-4)	3.83
0.77	1.16(-2)	7.17(-4)	4.02	1.27(-2)	8.60(-4)	3.88

Next we consider the convergence of an outwards traveling wave with

$$u(x, y, t) = \phi(r - t), \quad \xi_0 = 0.2, \quad \xi_1 = 0.4, \quad \epsilon = 0.035,$$

in a circle $|r| \leq 1$ with Dirichlet boundary conditions. Now the circle is covered by a rectangle of size 2.2×2.2 discretized with $h = 2.2/N$. This wave reaches the boundary around $t \approx 0.5 - 0.6$ and exits the domain around $t \approx 0.8 - 0.9$.

The results, presented in Table 3.6, are obtained using the same two CFL numbers and number of grid points as in the previous experiment. For this experiment the compact fourth-order-accurate method consistently gives smaller errors (compared with Tables 5 and 6 in [9]), particularly at later times when the wave has reached the boundary.

The convergence rates for this experiment appear to be very close to 4 during the whole process with only a slight decrease at a late time for the smaller CFL number.

3.3.2. Convergence of an outwards traveling wave with Neumann conditions. Finally we perform a convergence study of an outwards traveling wave with

$$u(x, y, t) = \phi(r - t), \quad \xi_0 = 0.3, \quad \xi_1 = 0.5, \quad \epsilon = 0.07,$$

and Neumann boundary conditions. The circle is now $|r| \leq 1.5$ and is covered by a rectangle of size 3.2×3.2 discretized with $h = 3.2/N$.

TABLE 3.7
Convergence results for an outwards traveling wave with Neumann boundary conditions.

t	$N = 200$	$N = 400$	rate
0.50	2.74(-4)	1.78(-5)	3.94
0.75	3.97(-4)	2.56(-5)	3.96
1.00	3.09(-4)	1.02(-4)	4.93
1.25	3.02(-3)	8.59(-5)	5.14

Here we restrict the simulations to a single CFL = 0.4 and two discretizations, $N = 200$, $N = 400$. The results are presented in Table 3.7. The error levels for this example are much lower than those presented in Table 3 in [10]. In particular, at the finer discretization and late time errors are approximately 30 times smaller.

3.4. Analytical solution in an annular region. Consider the solution of (2.1) with $f(x, y, t) = 0$ in an annular region, $(x, y) \in r_i \leq r \equiv \sqrt{x^2 + y^2} \leq r_o$. In such a separable geometry the analytical solution (in polar coordinates) is composed of a superposition of modes

$$u_{mn}(r, \theta, t) = J_m(r\kappa_{mn}) \cos m\theta \, \cos \kappa_{mn} t. \tag{3.3}$$

Here $J_m(z)$ is the Bessel function of the first kind of order m, $(m = 0, 1, \ldots)$ and κ_{mn} is the nth zero of J_m. The function (3.3) is more oscillatory for larger values of m and n and thus more challenging to solve numerically. Throughout this section we set $m = 7$.

In the first experiment we use homogeneous Dirichlet boundary conditions on the inner and the outer boundary. We set the outer boundary to $r_o = 1$ and choose the radial normalization such that $u(1, \theta, t) = 0$ at the 7th root, i.e. we set $n = 7$, for which we have $\kappa_{77} = 31.4227941922$. The inner boundary is chosen to coincide with the first zero of $J_7(r)$, i.e $r_i = \kappa_{71}/\kappa_{77} = 11.086370019/\kappa_{77}$ so that the analytical solution of the problem is

$$u(x, y, t) = J_7(r\kappa_{77}) \cos 7\theta \, \cos \kappa_{77} t.$$

A plot of the of the initial data can be found in the left image of Figure 3.2. The solution is advanced for one period in time until $t = \frac{2\pi}{\kappa_{77}} = 0.1999562886971$.

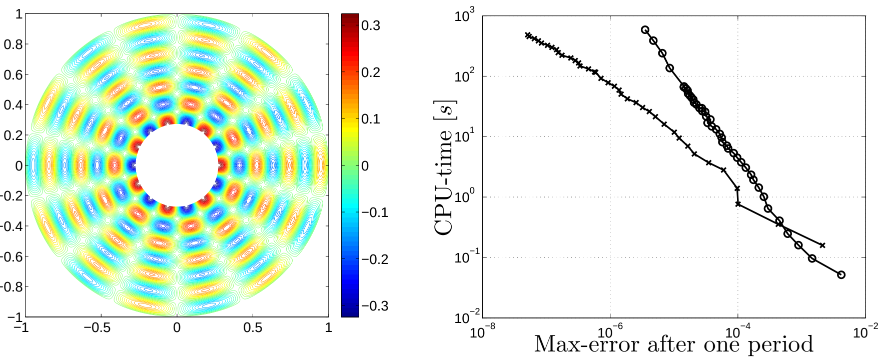


FIG. 3.2. *(Left) The function (3.3) with $m = n = 7$ and the inner and outer boundary chosen such that $u'(r_i, \theta, t) = u(r_o, \theta, t) = 0 = 0$. (Right) Comparison of used CPU-time as a function of max-error for the second-order-accurate method, [8], (circles) and the compact fourth-order method (crosses).*

TABLE 3.8
Errors measured in l_2- and l_∞-norm for the Bessel solution in an annulus with Dirichlet conditions.

N	l_2-err. d_4	rate	l_2-err. d_6	rate	l_2-err. d_8	rate
200	2.99(-4)		2.36(-4)		1.50(-4)	
400	3.36(-5)	3.15	1.49(-5)	3.99	5.84(-6)	4.69
800	4.13(-6)	3.03	8.55(-7)	4.12	1.89(-7)	4.95
N	l_∞-err. d_4	rate	l_∞-err. d_6	rate	l_∞-err. d_8	rate
200	6.32(-4)		9.70(-4)		4.30(-4)	
400	6.37(-5)	3.31	5.64(-5)	4.11	1.55(-5)	4.79
800	7.95(-6)	3.00	2.69(-6)	4.39	4.31(-7)	5.17

In Table 3.8 the l_2- and max-error at the final time are reported for a sequence of grids and for the three different damping terms. With this solution we see that the rate of convergence in both norms are approximately three and four for the d_4- and d_6-damping, while the rate of convergence is closer to five when d_8-damping is used. This is consistent with the results from the trigonometric solution and suggests that the errors from the fourth-order-accurate building-blocks of the method (differentiation and time-stepping) are smaller than the fifth-order-accurate building-blocks (interpolation and the d_8-damping).

In the second example we impose homogeneous Neumann boundary conditions on both the inner and the outer boundary. Again m and n are set to 7, but the location of the outer and inner boundary are adjusted so that they coincide with the zeros of $J_7'(r)$ that are closest to r_i and r_o from the previous experiment,

$$r_i = \frac{8.57783648971}{\kappa_{77}}, \quad r_o = \frac{29.7907485831}{\kappa_{77}}.$$

The results, reported in Table 3.9, are similar to those observed for the trigonometric solution with the exception that the observed rate of convergence more clearly tends to two for the d_4-damping. As the grid is refined, the rate of convergence approaches some value smaller than four for the d_6-damping and some value smaller than five for the d_8-damping.

Table 3.9

Errors measured in l_2- and l_∞-norm for the Bessel solution in an annulus with Neumann boundary conditions.

N	l_2-err. d_4	rate	l_2-err. d_6	rate	l_2-err. d_8	rate
200	1.33(-3)		1.37(-3)		3.43(-3)	
400	1.87(-4)	2.84	7.65(-5)	4.00	9.73(-5)	5.14
800	4.62(-5)	2.01	6.87(-6)	3.64	3.22(-6)	4.92
N	l_∞-err. d_4	rate	l_∞-err. d_6	rate	l_∞-err. d_8	rate
200	3.21(-3)		3.65(-3)		1.12(-3)	
400	5.19(-4)	2.63	2.28(-4)	4.16	3.50(-5)	5.00
800	1.40(-4)	1.89	1.83(-5)	3.48	1.24(-6)	4.82

In a third example we impose homogeneous Neumann boundary condition on the inner boundary and homogeneous Dirichlet boundary condition on the outer boundary, i.e. we set $r_i = 8.57783648971/\kappa_{77}$ and $r_o = 1$.

Table 3.10

Errors measured in l_2- and l_∞-norm for the Bessel solution in an annulus with Neumann and Dirichlet boundary conditions.

N	l_2-err. d_4	rate	l_2-err. d_6	rate	l_2-err. d_8	rate
200	4.90(-4)		2.90(-4)		1.40(-4)	
400	8.97(-5)	2.45	2.99(-5)	3.28	9.77(-6)	3.84
800	2.15(-5)	2.06	4.44(-6)	2.75	7.96(-7)	3.62
1600	5.05(-6)	2.09	5.83(-7)	2.93	4.05(-8)	4.30
N	l_∞-err. d_4	rate	l_∞-err. d_6	rate	l_∞-err. d_8	rate
200	2.00(-3)		9.40(-4)		4.29(-4)	
400	4.26(-4)	2.23	1.09(-4)	3.11	3.49(-5)	3.62
800	1.07(-4)	1.99	1.83(-5)	2.58	3.22(-6)	3.44
1600	2.43(-5)	2.15	2.49(-6)	2.88	1.53(-7)	4.39

In Table 3.10 the results from the mixed boundary condition problem are reported. As expected, the convergence rates for the mixed problem are dominated by the Neumann boundary condition and thus approach two, three, and four. Note that although the rate of convergence measured in both norms is higher for the Neumann case with d_8 than for the mixed case, the l_2-errors are almost an order of magnitude smaller for the mixed case.

As a final experiment, the solution to the mixed problem is computed on a sequence of grids and compared to the second-order accurate method described in [8]. For both methods we measure the max-error and the CPU time used to advance the solution by a single period in time. The timing is performed over the inner loop, the time for pre-computation is negligible and is therefore not included. Both methods are implemented in FORTRAN 90 and compiled with the Ifort compiler with `-O3` optimization and run on a 2GHz Intel Core 2 Duo MacBook.

The results of our comparison, displayed in the right image of Figure 3.2, show that the second-order method is only more efficient for accuracies worse than 0.1%. Recalling the classic result of Kreiss and Oliger [7], we anticipate that the benefits of the fourth order method will only increase for longer time simulations.

3.5. Eigenfrequencies in a membrane. In this experiment we explore an application of the proposed method, namely the extraction of resonant eigenfrequencies of a membrane via time-domain simulations, see e.g [24, 30]. The experiment also serves as an additional confirmation of the long time stability of the method.

TABLE 3.11
Values of the parameters for the membrane described by (3.4).

l	1	2	3	4	5	6
x_l	0.15	0.5	0.75	0.35	0.2	0.8
y_l	0.25	0.6	0.2	0.3	0.7	0.8
w_l	0.05	0.39	0.08	0.1	0.02	0.05

The boundary of the membrane is implicitly described by the equation

$$\varphi(x, y) = 0,$$

where

$$\varphi(x, y) = 0.5 - \sum_{l=1}^{6} e^{-\frac{(x-x_l)^2+(y-y_l)^2}{w_l^2}}, \tag{3.4}$$

with x_l, y_l and w_l given in Table 3.11. A picture of the membrane can be found in Figure 3.4.

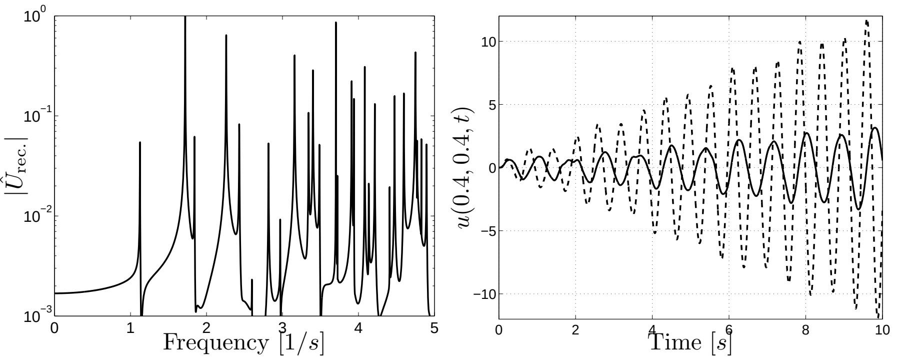


FIG. 3.3. *(Left) The (normalized) amplitude of the Fourier transform of the recorded signal. The resonant frequencies show up as spikes. (Right) Responses when the membrane is forced with a harmonic source with frequencies coinciding with the two lowest resonant frequencies (the solid line is for $f_r = 1.125$ and the dashed is for $f_r = 1.72$).*

To find the eigenfrequencies of the membrane, we force (2.1) by the pulse

$$f(x, y, t) = 1000(x - 0.5)e^{-\left(\frac{t-0.1}{0.07}\right)^2-\left(\frac{x-0.5}{0.01}\right)^2-\left(\frac{y-0.5}{0.01}\right)^2}.$$

The membrane is discretized on a grid covering the unit square with 2001 gridpoints in each direction (i.e. $h = 1/2000$) and the homogeneous Dirichlet boundary conditions are enforced using the line-by-line approach. The d_6-damping is used to ensure long-time stability of the solution, which is advanced to $t = 400$ (2,000,000 time steps).

The solution is recorded at $x = 0.4, y = 0.4$ and stored in a vector $U_{\text{rec}}(t)$ so that the resonant frequencies can be found by computing the Fourier transform, $\hat{U}_{\text{rec}}(f)$,

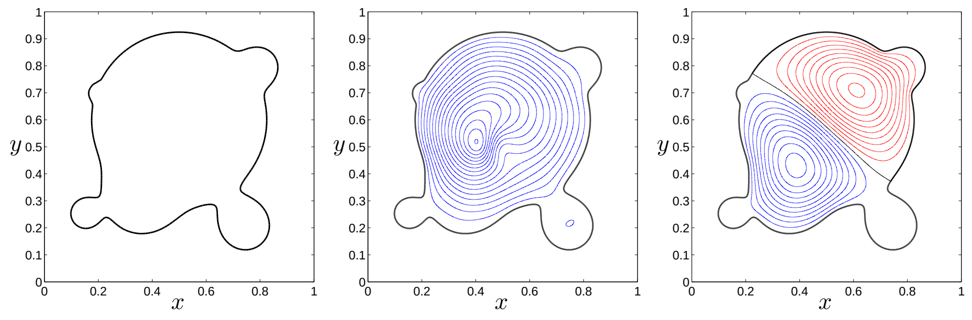


FIG. 3.4. *(Left) The shape of the membrane. (Middle) The first eigenmode at $t = 9.5$. The contours range from 0 to 4.5 with increments of 0.25. (Right) The second eigenmode at $t = 9.5$. The contours range from -10 to -1 with increments of 1 (red in the online version) and from 1 to 10 with increments of 1 (blue in the online version).*

of $U_{\text{rec}}(t)$. A plot displaying the spectrum for the first few frequencies can be found in Figure 3.3.

Once the frequencies are found, the eigenmodes can be approximated by forcing the problem with a harmonic source at the resonant frequency. Here we have forced it with

$$f(x,y,t) = 200\left(-2\frac{(x-0.45)}{0.05^2}\right)\sin(2\pi f_r t)\, e^{-\frac{(x-0.45)^2+(y-0.4)^2}{0.05^2}},$$

for $f_r = 1.125$ and $f_r = 1.72$. In Figure 3.3 the response at $x = 0.4, y = 0.4$ is plotted as a function of time. As expected, the amplitude grows linearly for a resonant frequency. Figure 3.4 plots the two first eigenmodes.

We note that this experiment is only meant to serve as an illustration of how resonant frequencies can be extracted using time-dependent simulations and that there are modern methods [24, 30] that significantly accelerates the above procedure.

3.6. A radiating unidentified object in free space. In the final experiment we study the radiation from an unidentified object in free-space. The purpose of this experiment is to demonstrate the use of non-reflecting boundary conditions together with the proposed method on a problem where the solution is propagated over many wavelengths.

The geometry of the unidentified object is defined as the zero contour of

$$\varphi(x,y) = 0.5 - 1.0e^{-\left(\frac{x^2}{0.3^2}+\frac{y^2}{0.04^2}\right)} - 1.2e^{-\left(\frac{x^2}{0.05^2}+\frac{(y-0.06)^2}{0.03^2}\right)},$$

see Figure 3.5. The object is placed in a rectangular domain $(x,y) \in [-1,1]^2$ discretized on a grid with spacing $h = 2/N$ and $N = 2500$. On the boundary of the object, we prescribe Dirichlet data in form of a smoothly started plane wave

$$u(x,y,t) = (1-e^{-5t^5})\cos(\omega(x-t)), \quad (x,y) \in \Gamma,\ t \geq 0,\ \omega = 600.$$

The outer, free-space, boundary condition is modeled by truncating the domain using a perfectly matched layer (derived in Appendix A)

$$u_{tt} = \frac{\partial}{\partial x}\left(u_x + \sigma^{(x)}\phi^{(1)}\right) + \frac{\partial}{\partial y}\left(u_y + \sigma^{(y)}\phi^{(2)}\right) + \sigma^{(x)}\phi^{(3)} + \sigma^{(y)}\phi^{(4)}, \tag{3.5}$$

where the auxiliary variables satisfy the equations

$$\begin{aligned}
\phi_t^{(1)} + (\alpha + \sigma^{(x)})\phi^{(1)} &= -u_x, \\
\phi_t^{(2)} + (\alpha + \sigma^{(y)})\phi^{(2)} &= -u_y, \\
\phi_t^{(3)} + (\alpha + \sigma^{(x)})\phi^{(3)} &= -u_{xx} - \frac{\partial}{\partial x}\left(\sigma^{(x)}\phi^{(1)}\right), \\
\phi_t^{(4)} + (\alpha + \sigma^{(y)})\phi^{(8)} &= -u_{yy} - \frac{\partial}{\partial y}\left(\sigma^{(y)}\phi^{(2)}\right).
\end{aligned} \tag{3.6}$$

The damping profiles $\sigma^{(z)}(z)$, $z = x, y$ are taken as

$$\sigma^{(z)}(z) = \frac{\sigma_{\max}}{2}\left(2 + \tanh\left(\frac{z - z_{\text{pml}}}{\delta_{\text{pml}}}\right) - \tanh\left(\frac{z + z_{\text{pml}}}{\delta_{\text{pml}}}\right)\right),$$

with $\sigma_{\max} = 15$, $z_{\text{pml}} = 0.75$, $\delta_{\text{pml}} = 0.01$. The complex frequency shift is chosen as $\alpha = 0.05$.

Now, as the modified equation approach is used to obtain a high-order-accurate time discretization we must compute

$$\begin{aligned}
u_{tttt} = &\frac{\partial^2}{\partial x^2}(u_{tt}) + \frac{\partial^2}{\partial y^2}(u_{tt}) \\
&+ \frac{\partial^2}{\partial t^2}\underbrace{\left(\frac{\partial}{\partial x}(\sigma^{(x)}\phi^{(1)}) + \frac{\partial}{\partial y}(\sigma^{(y)}\phi^{(2)}) + \sigma^{(x)}\phi^{(3)} + \sigma^{(y)}\phi^{(4)}\right)}_{\mathcal{F}}.
\end{aligned} \tag{3.7}$$

Once (3.5) is computed, the first two terms in (3.7) can be approximated using the compact scheme. The third term must be approximated to second-order accuracy. In this case it is done by first advancing the auxiliary variables using the classic fourth-order-accurate Adams method [14] to get $\phi(x, y, t_{n+1})$ so $\mathcal{F}^{n+1}$ can be computed. In our implementation a compact fourth-order-accurate method is used to approximate the first derivatives in $\mathcal{F}$. Once $\mathcal{F}^{n+1}$ is found, a second-order approximation is given by

$$\frac{\partial^2}{\partial t^2}\mathcal{F}^n \approx \frac{\mathcal{F}^{n+1} - 2\mathcal{F}^n + \mathcal{F}^{n-1}}{k^2}.$$

With the PML in place, the solution is advanced to $t = 3$, at which time most start-up transients have exited the computational domain and the solution has reached a time-harmonic state. The line-by-line approach is used to enforce the boundary conditions and the d_6-damping is used to stabilize the scheme.

The results of the simulation can be found in in Figure 3.5. The left image shows a snapshot of the solution at $t = 3$. The wavelength, λ, is about 0.01, so the total domain is $\sim 200\lambda$ wide and the unidentified object is $\sim 48\lambda$ wide. Note that only $(x, y) \in [-0.75, 0.75]^2$ is displayed in the figure. There are about 13 points per wavelength and it is noticeable that even with relatively few points per wavelength the dispersion error does not appear to be significant. The plot demonstrates the methods capability to simulate wave propagation problems with many wavelengths as well as complex geometry.

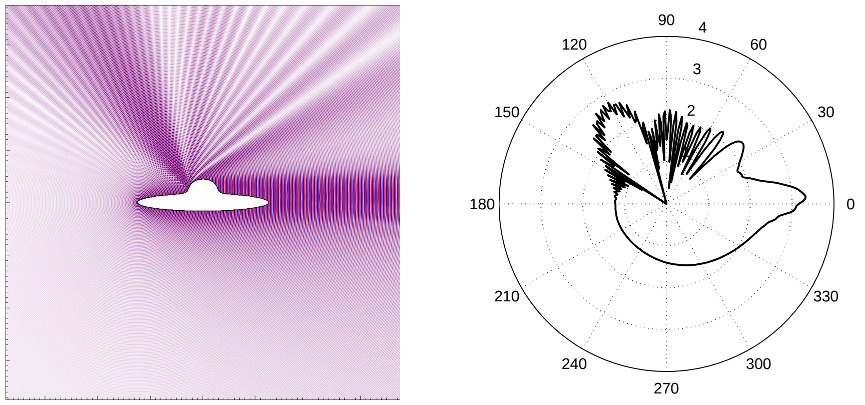


FIG. 3.5. *Left: Snapshot of $u(x, y, 3)$. The pictured domain is $(x, y) \in [-0.75, 0.75]^2$. To the right the quantity $I(\theta)$ is plotted.*

In order to compute the radiation pattern, the solution is recorded at $(x, y) = (0.5 \cos \theta, 0, 5 \sin \theta)$, $\theta = 0, d\theta, \ldots, 2\pi$, $d\theta = 2\pi/360$ at each time step. In the right image of Figure 3.5, the quantity

$$\max |\log_{10}(I(\theta))| - \log_{10}(I(\theta)),$$

is plotted. $I(\theta)$ is an intensity defined as the time integral of the square of the solution during one period, $T = 2\pi/\omega$,

$$I(\theta) = \int_{2}^{2+\frac{2\pi}{\omega}} |u(0.5 \cos \theta, 0.5 \sin \theta, \tau)|^2 \, d\tau.$$

4. Conclusions.

In summary, we have derived and demonstrated the applicability of a fourth-order accurate embedded boundary method for the wave equation. In our view, the strength of the method is its high-order of accuracy along with its relative simplicity. The fact that the method consist of separate building blocks that can easily be adjusted if other PDE or applications are considered is another strength.

There are many possible extensions of the method that are worth considering. These include:

1. Generalize the method to Maxwell's equations formulated as a system of second order equations.
2. Adopting the approach to compressible flow problems. This has been successfully done in [26, 12] for the earlier second-order accurate methods.
3. Raising the order of accuracy further. Raising the order of the compact method to six is possible by increasing the bandwidth of the linear system from three to five. Extending the temporal discretization and raising the order of the *boundary stencils* should also be straightforward but might require that new damping terms are considered.
4. Experimentation with other spatial and temporal discretizations, in particular summation-by-parts finite difference discretizations [11].

Finally, we note that this paper has mainly focused on the description of the method along with several numerical experiments and largely avoided detailed analysis of the discretization. The rationale for this emphasis is twofold. First, detailed analysis of the EB-methods of the type considered here has already been presented for the related second-order accurate methods in [8, 9, 10]. Analysis of the present method would largely follow along the lines of [8, 9, 10]. It would be algebraically more involved but probably lead to similar conclusions: there are weak instabilities that can be suppressed by suitably chosen artificial dissipation. Second, an aim of this paper was to promote the use of embedded boundary methods and we thus focused on describing the algorithms in sufficient detail to make them replicable by the interested reader.

Appendix A. PML with modified equation time-stepping. To be able to treat unbounded domains we enclose the computational domain within a perfectly matched layer. Following [3] we perform the usual s (s is the Laplace transform variable dual of t) dependent coordinate stretching and get:

$$\begin{aligned}\rho s^2 \hat{u} = {} & \left(1-\frac{\sigma^{(x)}}{s+\alpha+\sigma^{(x)}}\right)\frac{\partial}{\partial x}\left(\left(1-\frac{\sigma^{(x)}}{s+\alpha+\sigma^{(x)}}\right)\frac{\partial}{\partial x}\hat{u}\right)\\ & + \left(1-\frac{\sigma^{(y)}}{s+\alpha+\sigma^{(y)}}\right)\frac{\partial}{\partial y}\left(\left(1-\frac{\sigma^{(y)}}{s+\alpha+\sigma^{(y)}}\right)\frac{\partial}{\partial y}\hat{u}\right). \end{aligned}\tag{A.1}$$

To localize we introduce the auxiliary variables

$$\begin{aligned}\phi^{(1)} &= -\frac{1}{s+\alpha+\sigma^{(x)}}\frac{\partial}{\partial x}\hat{u}, \quad \phi^{(2)} = -\frac{1}{s+\alpha+\sigma^{(y)}}\frac{\partial}{\partial y}\hat{u},\\ \phi^{(3)} &= -\frac{1}{s+\alpha+\sigma^{(x)}}\frac{\partial}{\partial x}\left(\left(1-\frac{\sigma^{(x)}}{s+\alpha+\sigma^{(x)}}\right)\frac{\partial}{\partial x}\hat{u}\right),\\ \phi^{(4)} &= -\frac{1}{s+\alpha+\sigma^{(y)}}\frac{\partial}{\partial y}\left(\left(1-\frac{\sigma^{(y)}(y)}{s+\alpha+\sigma^{(y)}}\right)\frac{\partial}{\partial y}\hat{u}\right).\end{aligned}$$

This leads to the perfectly matched layer model (3.5) with auxiliary equations (3.6).

Appendix B. Artificial dissipation. Acting on a one-dimensional grid function $q_i, i = 1, \dots, s$, the fourth-order operator $w_i = D_4 q_i$ is

$$\begin{aligned} w_1 &= q_1 - 2q_2 + q_3,\\ w_2 &= -2q_1 + 5q_2 - 4q_3 + q_4,\\ w_i &= 6q_i - 4(q_{i+1} + q_{i-1} + q_{i+2} + q_{i-2}), \quad i = 3, \dots, s-2.\end{aligned}$$

The sixth-order operator $w_i = D_6 q_i$ is

$$\begin{aligned} w_1 &= -3q_1 + 9q_2 - 9q_3 + 3q_4,\\ w_2 &= 9q_1 - 28q_2 + 30q_3 - 12q_4 + q_5,\\ w_3 &= -9q_1 + 30q_2 - 37q_3 + 21q_4 - 6q_5 + q_6,\\ w_4 &= 3q_1 - 12q_2 + 21q_3 - 22q_4 + 15q_5 - 6q_6 + q_7,\\ w_i &= -20q_i + 15(q_{i-1} + q_{i+1}) - 6(q_{i-2} + q_{i+2}) + (q_{i-3} + q_{i+3}), \quad i = 5, \dots, s-4.\end{aligned}$$

The eight-order operator $w_i = D_8 q_i$ is

$$
\begin{aligned}
w_1 &= 3q_1 - 12q_2 + 18q_3 - 12q_4 + 3q_5,\\
w_2 &= -12q_1 + 49q_2 - 76q_3 + 54q_4 - 16q_5 + q_6,\\
w_3 &= 18q_1 - 76q_2 + 125q_3 - 100q_4 + 40q_5 - 8q_6 + q_7,\\
w_4 &= -12q_1 + 54q_2 - 100q_3 + 101q_4 - 64q_5 + 28q_6 - 8q_7 + q_8,\\
w_5 &= 3q_1 - 16q_2 + 40q_3 - 64q_4 + 72q_5 - 56q_6 + 28q_7 - 8q_8 + q_9,\\
w_i &= 70q_i - 56(u_{i-1} + u_{i+1}) + 28(u_{i-2} + u_{i+2}) - 8(u_{i-3} + u_{i+3}) + 1(u_{i-4} + u_{i+4}),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 6, \dots, s-5.
\end{aligned}
$$

REFERENCES

[1] B. Alpert, L. Greengard, and T. Hagstrom, *An integral evolution formula for the wave equation*, Journal of Computational Physics, 162 (2000), pp. 536–543.

[2] L. Anne, P. Joly, and H. Q. Tran, *Construction and analysis of higher order finite difference schemes for the 1d wave equation*, Computational Geosciences, 4 (2000), pp. 207–249.

[3] D. Appelö and G. Kreiss, *Application of a perfectly matched layer to the nonlinear wave equation*, Wave Motion, 44 (2007), pp. 531–548.

[4] O. Bruno and M. Lyon, *High-order unconditionally-stable FC-AD solvers for general smooth domains I. basic elements.*, Submitted, (2009).

[5] L. Collatz, *The Numerical Treatment of Differential Equations*, Springer-Verlag, New York, 1960.

[6] M. A. Dablain, *The application of high-order differencing to the scalar wave equation*, Geophysics, 51 (1986), pp. 54–66.

[7] H.-O. Keiss and J. Oliger, *Comparison of accurate methods for the integration of hyperbolic equations*, Tellus, 24 (1972), pp. 199–215.

[8] H.-O. Kreiss and N. A. Petersson, *A second order accurate embedded boundary method for the wave equation with Dirichlet data*, SIAM J. Sci. Comput., 27 (2006), pp. 1141–1167.

[9] H.-O. Kreiss, N. A. Petersson, and J. Yström, *Difference approximations for the second order wave equation*, SIAM Journal on Numerical Analysis, 40 (2002), pp. 1940–1967.

[10] ———, *Difference approximations of the neumann problem for the second order wave equation*, SIAM Journal on Numerical Analysis, 42 (2004), pp. 1292–1323.

[11] H.-O. Kreiss and G. Scherer, *Finite element and finite difference methods for hyperbolic partial differential equations*, in Mathematical Aspects of Finite Element in Partial Differential Equations, Academic Press, Inc., 1974.

[12] M. Kupiainen and B. Sjögreen, *A cartesian embedded boundary method for the compressible navier-stokes equations*, Journal of Scientific Computing, 41 (2009), pp. 94–117.

[13] S. K. Lele, *Compact finite difference schemes with spectral-like resolution*, Journal of Computational Physics, 103 (1992), pp. 16–42.

[14] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2007.

[15] 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.

[16] B. Lombard and J. Piraux, *Numerical treatment of two-dimensional interfaces for acoustic and elastic waves*, Journal of Computational Physics, 195 (2004), pp. 90–116.

[17] B. Lombard, J. Piraux, C. Gelis, and J. Virieux, *Free and smooth boundaries in 2-d finite-difference schemes for transient elastic waves*, Geophysical Journal International, 172 (2008), pp. 252–261.

[18] M. Lyon, *High-order unconditionally-stable FC-AD PDE solvers for general domains*, PhD thesis, Caltech, 2009.

[19] M. Lyon and O. Bruno, *High-order unconditionally-stable FC-AD solvers for general smooth domains II. elliptic, parabolic and hyperbolic pdes: Theoretical considerations*, Submitted, (2009).

[20] K. Mattsson, M. Svärd, and J. Nordström, *Stable and accurate artificial dissipation*, Journal of Scientific Computing, 21 (2004), pp. 57–79.

[21] P. Olsson, *The numerical behavior of high-order finite difference methods*, Journal of Scientific Computing, 9 (1994), pp. 445–466.

[22] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, and M. L. Welcome, *An adaptive cartesian grid method for unsteady compressible flow in irregular regions*, Journal of Computational Physics, 120 (1995), pp. 278–304.

[23] C. S. Peskin, *Flow patterns around heart valves: A numerical method*, Journal of Computational Physics, 10 (1972), pp. 252–271.

[24] T. Rylander, T. McKelvey, and M. Viberg, *Estimation of resonant frequencies and quality factors from time domain computations*, Journal of Computational Physics, 192 (2003), pp. 523–545.

[25] G. R. Shubin and J. B. Bell, *A modified equation approach to constructing fourth order methods for acoustic wave propagation*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 135–151.

[26] B. Sjögreen and N. A. Petersson, *A cartesian embedded boundary method for hyperbolic conservation laws*, Communications In Computational Physics, 2 (2007), pp. 1199–1219.

[27] J. Visher, S. Wandzura, and A. White, *Stable, high-order discretization for evolution of the wave equation in 1+1 dimensions*, Journal of Computational Physics, 194 (2004), pp. 395–408.

[28] S. Wandzura, *Stable, high-order discretization for evolution of the wave equation in 2 + 1 dimensions*, Journal of Computational Physics, 199 (2004), pp. 763 – 775.

[29] R. Weller and H. Shortely, *Calculation of stresses within the boundary of photoelastic models*, J. Appl. Mech, 6 (1939), pp. A71–A78.

[30] G. R. Werner and J. R. Cary, *Extracting degenerate modes and frequencies from time-domain simulations with filter-diagonalization*, Journal of Computational Physics, 227 (2008), pp. 5200–5214.