Factoring the wave equation for fast, implicit numerical solutions

Michael P. Lamoureux[1], and RJ Vestrum[1]

ABSTRACT

Numerical solutions to the wave equations that arise in seismic imaging have been richly developed over the last 70+ years, and typically depend on **explicit** time-stepping or Fourier techniques for fast, accurate solutions. The counterpart to explicit methods are **implicit** methods which have enjoy features such as unconditional stability, but typically are computationally prohibitive in two and three spatial dimensions. We demonstrate here a fast, stable implicit time-stepping numerical method for solving the wave equation in two dimensions and higher, that makes use of novel operator factorization in a grid algebra. Several computational demonstrations of the method are presented as well.

This is a summary of the main results described in the MSc thesis of the second author.

INTRODUCTION

Finding efficient numerical methods for solving systems of differential equations that describe physical processes is a central challenge in scientific computation. In seismic imaging, a key problem is finding numerical solutions for the various wave equation models that describe seismic wave propagation through the earth's subsurface. Several numerical methods for calculating the wave propagation in space involve approximating the wavefield on a grid of nodes or cells in space and modelling its propagation along the grid. A key step is representing the wave operator as a linear matrix acting on the vector space of functions evaluated at the grid points. For even modest grids used in finite-differences, say 100×100 in 2D or $100\times100\times100$ in 3D, the vector spaces that appear have 10,000 or 1,000,000 dimensions and the associated linear operators are represented as matrices of size $10{,}000\times10{,}000$ or $1{,}000{,}000\times1{,}000{,}000$. Fortunately, these matrices are typically sparse, and there exist sparse matrix methods and iterative methods that can manage the high dimensions of the problem.

There is considerable interest in developing implicit solutions for finite-difference methods of modelling wave propagation, in order to make use of the inherent advantages of implicit methods, such as unconditional stability and accuracy. Some early work on implicit methods for multidimensional PDEs goes back to [Stone(1968)]; more recent developments focus on such things as the alternating direction implicit (ADI) schemes as seen in [Qin(2009)]. A central challenge in the implicit method is that, somewhere along the line, a system of linear equations has to be solved. Here, the high dimensions are a particular problem and even general sparse matrix methods only partially alleviate the challenge.

In one dimension Claerbout demonstrated that implicit method are cheaper since the increased accuracy allows the use of larger time steps. However for space dimensions higher

[1]University of Calgary

than one, the implicit method becomes prohibitively costly [Claerbout(1985)]. On the plus side, often implicit methods are unconditionally stable. [Smith and Smither(1985)].

We note, however, that there is an underlying structure in the linear systems that arises from the fact that the organized grid of points in 2D or 3D gives additional geometric relationships between coefficients in the linear operator. The general sparse matrix techniques do not readily take advantage of these geometric relationships, so there is an opportunity to develop algorithm efficiencies from this extra information. As an alternative to the sparse matrix methods, we aim to use the underlying geometric structure to perform “matrix operations” directly on the structured linear system to produce more efficient linear algebraic methods.

For a finite-difference operator expressed as a sparse matrix with the non-zero values falling on a few diagonals, standard $\mathbf{LU}$ decomposition ‘fills in’ a very large number of zeros between the diagonals [Smith and Smither(1985)]. The same issue arises with Cholesky factorization.

The focus of the research to find factorizations similar to $\mathbf{LU}$ decomposition of such a linear system, that preserve the local structure. There has long been hope for such an approach: “It seems fortunate that the table contains many zeroes, and we are led to hope for a rapid solution method for the simultaneous equations." [Claerbout(1985)]

In this work, we demonstrate a representation of these linear operators via a graph of arrows and nodes. The nodes represent grid points, arrows represent non-zero entries in the associated linear operator connecting one node to another, with weights on the arrows representing the value of the coefficient in the associated linear term. We then show how elementary row operations are performed on this grid representation, and show how back-substitution can quickly solve the linear system for operators with certain restrictions on the arrows that appear. We demonstrate a factorization of linear operators, similar to the LU factorization for regular matrices, but making use of the structure of the grid representation. In factored form, we can then solve the system of linear equations in $\mathcal{O}(N)$ steps.

We then apply this factorization to the Laplace operator as it arises in the implicit method for solving the wave equation in 2D and 3D .

Some initial steps in this research were started by the first author some five years ago. [Lamoureux and Hardeman-Vooys(2016)] The results reported here are contained in more detail in the MSc thesis of the second author. [Vestrum(2021)]

TIME STEPPING SOLUTION TO WAVE EQUATION

Explicit methods

The acoustic wave equation in three dimensions can be expressed in the form

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u(x, y, z, t). \tag{1}$$

where ∇^2 represents the Laplace operator and c the velocity of propagation [Selvadurai(2000)]. An explicit time stepping method of numerical solution is obtained by representing the wave function u as a collection of time samples $u^k = u(x, y, z, t_k)$ at time step t_k, sampled on a spatial grid of $N_x \times N_y \times N_z$ points. The differential operators in the wave equation are replaced with discrete difference operators,

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\Delta t^2} = \frac{c^2}{\Delta x^2}\mathcal{L}u^k, \tag{2}$$

where $\Delta t, \Delta x$ are the time and space step sizes, and $\mathcal{L}$ is a discrete difference operator that approximates the Laplacian.[2] Rearranging the equation gives the explicit time stepping method,

$$u^{k+1} = (2 + \epsilon\mathcal{L})u^k - u^{k-1}, \tag{3}$$

which tells us how to compute the wave function u at time step $k + 1$ in terms of the function values at previous time steps k and $k - 1$. Here. we have defined $\epsilon = \frac{c^2\Delta t^2}{\Delta x^2}$ as the CFL parameter. For an explicit method to be stable, this parameter ϵ must be small, which requires our time step to be sufficient small for a given velocity c and spatial sampling Δx. [Myint-U. and Debnath(2006)].

Implicit methods

To obtain an **implicit** method which is unconditionally stable, we replace the u^k on the RHS of Eqn 2 by u^{k+1}, to obtain

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\Delta t^2} = \frac{c^2}{\Delta x^2}\mathcal{L}u^{k+1}, \tag{4}$$

which we now rewrite as

$$(1 - \epsilon\mathcal{L})u^{k+1} = 2u^k - u^{k-1}, \tag{5}$$

where again we define $\epsilon = \frac{c^2\Delta t^2}{\Delta x^2}$. This last equation is a linear system of the form $\mathbf{A}\mathbf{u} = \mathbf{v}$, which in principle is very challenging to solve as it represents $N_xN_yN_z$ linear equations in $N_xN_yN_z$ unknowns. In other words, matrix $\mathbf{A} = 1 - \epsilon\mathcal{L}$ is a large $N \times N$ matrix, where $N = N_xN_yN_z$ is the number of points in the spatial grid.

In the next sections, we will see that grid algebras can be used to rapidly solve this system, in only $O(N)$ steps, making the new method comparable to the explicit methods while adding unconditional stability.

Augmented implicit methods

We note there is considerable freedom in selecting an implicit method, as there are several ways to introduce the sample u^{k+1} to the RHS of Eqn 2. For instance, similar

[2] A popular choice is the the standard 7-point discrete Laplacian $\mathcal{L}u(x, y, z) = u(x \pm \Delta x, y, z) + u(x, y \pm \Delta x, z) + u(x, y, z \pm \Delta x) - 6u(x, y, z)$.

to a Crank-Nicolson method [Myint-U. and Debnath(2006)], we may replace u^k with the average of u^k and u^{k+1} to obtain the formula

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\Delta t^2} = \frac{c^2}{\Delta x^2}\mathcal{L}\left(\frac{u^k + u^{k+1}}{2}\right). \tag{6}$$

Rearranging gives

$$(1 - \frac{\epsilon}{2}\mathcal{L})u^{k+1} = 3u^k - u^{k-1} - (1 - \frac{\epsilon}{2}\mathcal{L})u^k, \tag{7}$$

or equivalently

$$u^{k+1} = (1 - \frac{\epsilon}{2}\mathcal{L})^{-1}(3u^k - u^{k-1}) - u^k, \tag{8}$$

which again involves solving a large linear system of the form $\mathbf{Au} = \mathbf{v}$.

The fast computational methods in the next sections will also work for linear systems of this form, for the augmented implicit solution.

GRID REPRESENTATION OF LINEAR OPERATORS

Since the size of matrix $\mathbf{A}$ is $N \times N$, where $N = N_x N_y N_z$ is the number of points in the grid, the number of coefficients in $\mathbf{A}$ is N^2, and the computations required to solve the system $\mathbf{Au} = \mathbf{v}$ is going to grow on the order of N^2. To avoid this scaling cost, we are going to take advantage of the geometry of the problem and create a grid representation of our linear operators.

First, in the simple case where we have a one dimension grid with four points, the matrix equation $\mathbf{Au} = \mathbf{v}$ such as

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{9}$$

can be represented as a bipartite graph, with inputs u_j on the left side of the graph (see Figure 1) and outputs v_k on the right, along with arrows with weights connecting the inputs to outputs.

It is convenient to collapse the inputs and outputs to shared nodes, giving a representation as a direction multigraph, as shown in Figure 2 (left). The linear operator by itself is then succinctly described as a multigraph with weights, shown in Figure 2 (right), where the weight for each off-diagonal element in the matrix is shown with an arrow, while the diagonal weight is written inside the node.

It is elementary to extend this representation to two and three dimensions, expressing a linear operator on a 2D or 3D grid as a collection of nodes and arrows, with weights encoding the coefficients of the given linear operator. For example, on a 2D 4x4 grid, we express the usual discrete Laplacian (approximating ∇^2) as the weighted multigraph shown in Figure 3.

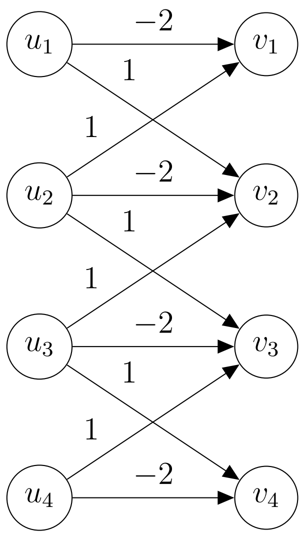

FIG. 1. Bipartite graph representation of a 4×4 linear operator.

Note that this is a relatively compact representation of a 16x16 linear operator with 64 non-zero coefficients. With inputs u_{ij} and outputs v_{ij}, this linear operator can be expressed in the usual indexed form

$$v_{ij} = -4u_{ij} + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}, \tag{10}$$

for all indices $1 \leq i, j \leq 4$.

The representation in Figure 3 makes it clear there is a nearest-neighbour structure to the Laplace operator, which is not immediately obvious in the usual 16x16 sparse matrix representation of the linear operator. Any linear operator $\mathbf{A}$ acting on a 2D grid of data points u_{ij} can be represented in this manner, with arrows and weights, provided we allow for arrows that can point between any two nodes in the grid. The point is, the nearest-neighbour and other topological relations in the 2D grid that are encoded in the linear operator can be more readily apparent in this grid representation.

In three dimensions, the approach is the same. We begin with a three dimensional grid with dimensions $N_x \times N_y \times N_z$, which contains $N = N_x N_y N_z$ nodes. A linear operator on this grid is a matrix of dimension $N \times N$ which can be represented as a collection of nodes and arrows, with weights attached to arrows and nodes. While it would be difficult to draw such a 3D graph for a large grid, it is illustrative to demonstrate show how a linear operator applies to a single node. In Figure 4, we see the 3D cube with the three standard directions X, Y, Z. It is useful to think of these “directions" as shift operators on the grid, corresponding to an arrow that connects an output to an input via a shift in one direction.

In our diagrams, a single shift in the directions x, y, z is represented by an arrow from the centre point along the axis of the shift, giving a linear operator denoted as X, Y, or Z respectively. A shift in the opposite direction is labeled X^*, Y^* or Z^* as these correspond to the adjoint linear operator (matrix transpose). We can compose operators, and represent as a product of arrows. For instance in Figure 5 the black arrow representing a shift of

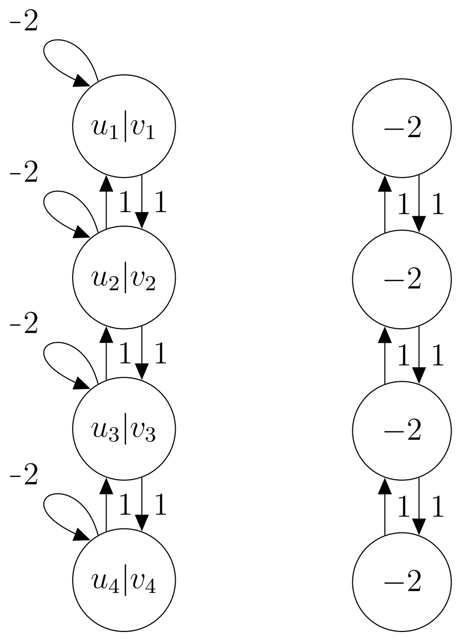

FIG. 2. Multigraph representation of a 4×4 linear operator, with and without inputs/outputs.

X^*Y^*Z is shown from the centre, with the coloured arrows added to show each of these shifts applied one at time.

With this method of drawing 3D linear operators, we can show a representation of the 3D discrete Laplacian that is built on the diagonals of the stencil's cube. Figure 6 shows a representation of the 3D discrete Laplacian built on the diagonals of the stencil's cube. This represents the linear operator given by the formulas

$$\begin{aligned} v_{ijk} = -8u_{ijk} &+ u_{i-1,j-1,k-1} + u_{i-1,j-1,k+1} + u_{i-1,j+1,k-1} + u_{i-1,j+1,k+1} \\ &+ u_{i+1,j-1,k-1} + u_{i+1,j-1,k+1} + u_{i+1,j+1,k-1} + u_{i+1,j+1,k+1}, \end{aligned} \tag{11}$$

for all indices$1 \leq i \leq N_x, 1 \leq j \leq N_y, 1 \leq k \leq N_z$.

As in the 2D case, the point is that certain linear operators, such as those with only nearest neighbour dependence, will have very simple grid structures in this representation. By comparison, the corresponding matrix representation will be sparse and structured (banded), but the structure would not easily reveal the nearest neighbour property. We can see structures in the grid representation which will not be obvious in the usual matrix representation of the linear operator.

By using the grid representations for the linear operators, the next sections shows how to use this structure for find computationally fast algorithms for inverting systems of linear operators defined on these grid.

SOLVING $\mathbf{Au} = \mathbf{v}$ BY BACK-SUBSTITUTION

As with triangular matrices, when a grid operator has a certain form, it is easy to solve the related system $\mathbf{Au} = \mathbf{v}$ using back substitution. For triangular matrices this requires the

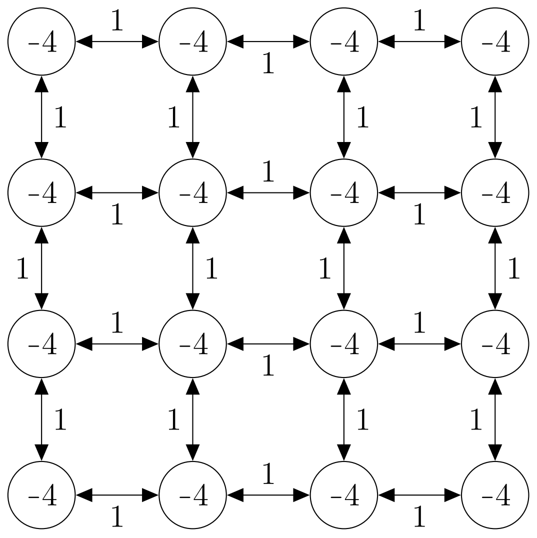

FIG. 3. Linear operator by itself, with weights and nodes.

matrix to be upper triangular, with every entry below the diagonal being zero. For instance a grid operator with all the arrows going down and/or to the right, can be expressed as an upper triangular matrix, and it is in a form that is quickly solved via back-substitution.

More generally, suppose a 2D grid representation has non-zero arrows contained to an open half plane. In this case, it is then possible to express this in a corresponding linear (grid) system $\mathbf{Au} = \mathbf{v}$ in which the matrix $\mathbf{A}$ is upper triangular. Thus the system can be solved that the system can be solved by back-substitution.

As a particular example, suppose we have a 2D grid operator that only has down and right pointing arrows, with only nearest neighbour arrows non-zero. For instance, suppose our linear system of equations is expressed as

$$v_{i,j} = au_{i,j} + bu_{i-1,j} + cu_{i,j-1} \tag{12}$$

for all indices $1 \leq i \leq N_x, 1 \leq j \leq N_y$, and some fixed coefficients a, b, c.[3] In this case, we can invert the linear system $\mathbf{Au} = \mathbf{v}$ by rearranging Eqn 12 in the form

$$u_{i,j} = (v_{i,j} - bu_{i-1,j} - cu_{i,j-1})/a \tag{13}$$

and solving recursively from this formula for $u_{i,j}$, where we let the indices i, j run in increasing order from 1 to N_x and to N_y, respectively. The point is, by the time the algorithm is trying to compute $u_{i,j}$ in Eqn 13, it has already solved for the unknowns $u_{i-1,j}$ and $u_{i,j-1}$ on the RHS of this formula, so indeed the RHS is known. This is the heart of back substitution.

Notice that in Eqn 13 there are only 5 floating point operations (two multiplications, two subtractions and one division) per coefficient $u_{i,j}$, so the inversion takes only $5N$ FLOPS

[3]The non-constant coefficient case can be handled easily as well.

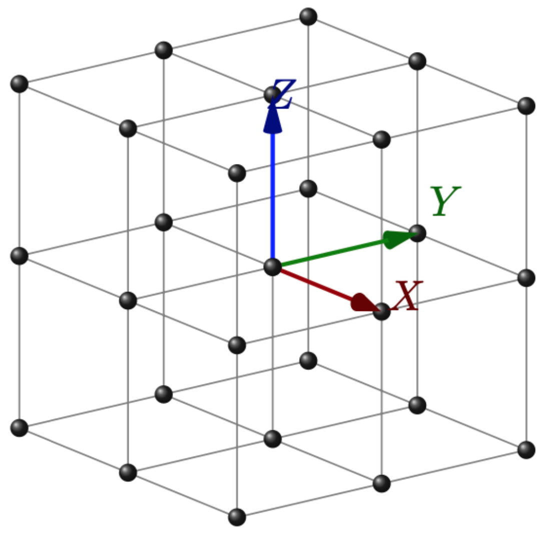

FIG. 4. The 3x3x3 grid and standard shift operators.

to compute. These is how we are able to achieve an inverse in $O(N)$ operations, instead of the typical $O(N^2)$ for a general linear system.

A similar result is true for 3D grid. If a grid operator can be expressed as shifts where all the shifts are in the same one direction along each axes, then this can be expressed as an upper triangular matrix and can be solved easily by back substitution. Therefore any shift with all the arrows in the same half space can be solved by back substitution.

FACTORING IN THE GRID ALGEBRA

We want to find a fast way to solve the linear system of equations that arise in the implicit methods for the wave equation. There is a specific linear operator, or matrix, that comes up in the implicit method, which is just the identity operator minus a multiple of the Laplacian, $I - \epsilon\mathcal{L}$. This is a sparse, positive definite symmetric matrix, so in principle it can be factored using standard methods in linear algebra, such as the Cholesky factorization. [Press(2009)] However, these standard methods typically ignore the sparse structure of the matrix, and lead to fill-in, which will result in slow methods to invert the linear system. Our goal is to find a factorization that somehow respect the simple, nearest neighbour structure on the computational grid.

As a motivating example, consider factoring the shifted Laplacian $I - \epsilon\mathcal{L}$ on a 2D, uniformly spaced grid. Define the shift operators X, Y by the formulas

$$(Xu)(x,y) = u(x-h,y), \qquad (Yu)(x,y) = u(x,y-h), \tag{14}$$

where here the grid spacing $\Delta x = \Delta y = h$ is the same in each spatial direction. The adjoints of these operator are shifts in the opposite direction, so

$$(X^*u)(x,y) = u(x+h,y), \qquad (Y^*u)(x,y) = u(x,y+h). \tag{15}$$

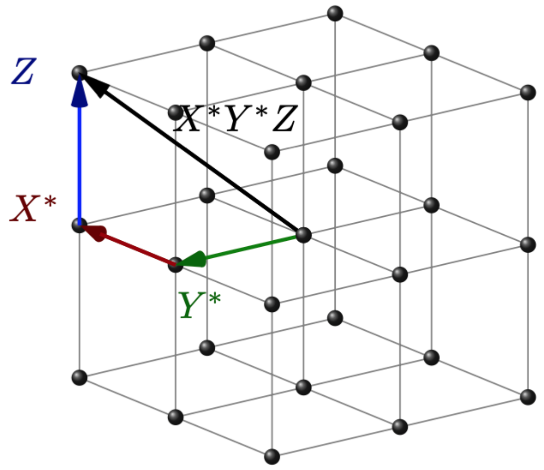

FIG. 5. A linear operator X^*Y^*Z represented as a composition of three shifts.

With this notation, the standard discrete Laplacian is defined as the linear operator given by a sum of shifts $\mathcal{L}_0 = X + X^* + Y + Y^* - 4I$, where I is the identity operator (identity matrix).

A more general Laplacian takes an affine combination of the standard Laplacian $\mathcal{L}_0$ and the diagonal Laplacian $\mathcal{L}_1 = (XY + X^*Y + XY^* + X^*Y^* - 4I)/2$, to yield the formula

$$\mathcal{L}_\gamma = (1-\gamma)\mathcal{L}_0 + \gamma\mathcal{L}_1. \tag{16}$$

To factor the shifted Laplacian, we look for a solution to the equation

$$I - \epsilon\mathcal{L}_\gamma = (aI + bX + cY + dXY)(aI + bX^* + cY^* + dX^*Y^*), \tag{17}$$

for a given $\epsilon > 0$ and unknowns a, b, c, d and γ. Expanding the product on the RHS and equating coefficients of the terms I, X, Y, XY, X^*Y gives a system of five equations in five unknowns. A solution is given as

$$\begin{aligned} a &= (1+\sqrt{1+4\epsilon})^2/4 \\ b &= -\epsilon \\ c &= -\epsilon \\ d &= (1-\sqrt{1+4\epsilon})^2/4 \\ \gamma &= -2\epsilon \end{aligned} \tag{18}$$

It is useful to look at this solution in the grid algebra formulation, which is presented in Figure 7. On the left of the figure is the arrow diagram for the shifted 9 point Laplacian, $I - \epsilon\mathcal{L}$. On the right are the arrow diagrams for the factor $F = aI + bX + cY + dXY$ and its adjoint $F^* = aI + bX^* + cY^* + dX^*Y^*$. Notice that one factor has all the arrows going right and up, so the related inverse problem can be solved quickly by back-substitution. The adjoint factor F^* has arrows in the opposite direction, so it is also quickly inverted.

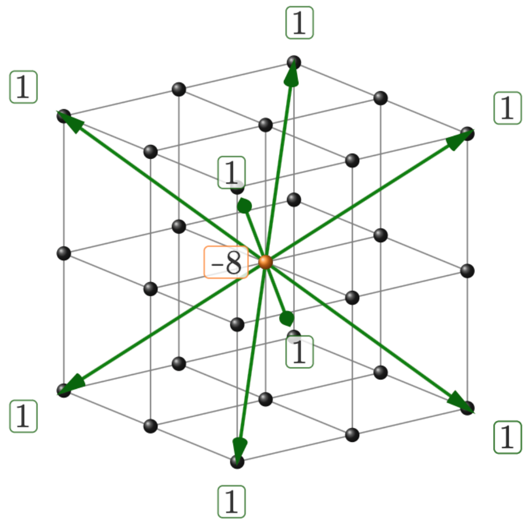

FIG. 6. The 9-point Laplacian stencil on a 3D grid, with weights indicated.

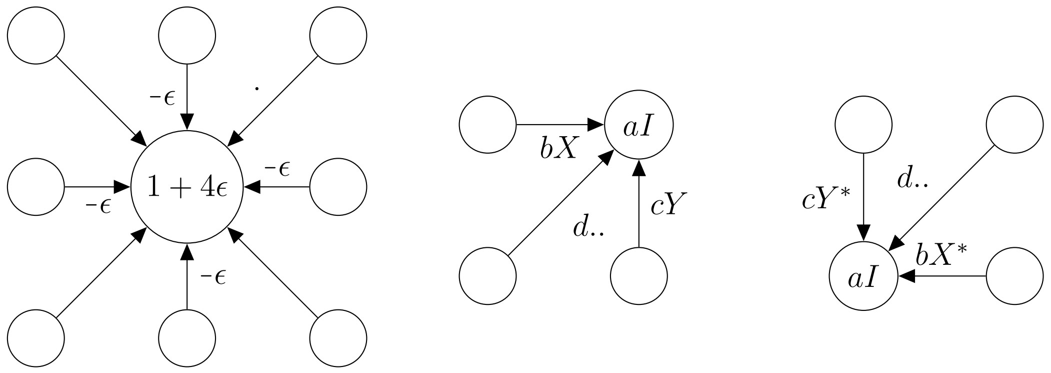

FIG. 7. Arrows for the shifted 2D Laplacian, and factors $F = aI + bX + cY + dXY$ and F^*.

There are a few features to note about this solution. First, the extra parameter γ is really needed in order to solve the system of five equations. Second, the solution obtains γ as a negative number, so the resulting discrete approximation to the Laplacian is not a usual convex combination of $\mathcal{L}_0$ and $\mathcal{L}_1$, but simply affine. Since ϵ is typically small, this approximation is close to $\mathcal{L}_0$, the standard discrete Laplacian. Third, the solution is not unique – for instance, the roles of a and d can be reversed. However, there are numerical advantages in keep the matrix diagonal element a large (close to one) and off-diagonal d small (close to ϵ^2).

Finally, this is a special case of a Cholesky factorization of the operator $I - \epsilon\mathcal{L}_\gamma$ since the factor $aI + bX + cY + dXY$ is represented by a lower triangular matrix, and the other factor is its transpose. By a judicious choice of parameter γ, we are able to make these factors limited to nearest neighbour arrows only, in a restricted set of directions (e.g. right, up, and right-up), allowing for fast inversion by back-substitution.

It is worth highlighting this fact: by including the free parameter γ, we are able to "fix" the fill-in problem with the usual Cholesky factorization, and end up with a nearest neighbour factor. This is an unexpected and highly useful mathematical result that results in the speed up of the implicit solution to the PDE problem.

Our initial attempts to factor the operator used numerical methods to find an approximate solution to the factoring problem, since the resulting system of equations involves five quadratics in five unknowns. Such a complex system is not typically solvable in exact formula, and there were no obvious simplifications. However, over the course of an MSc thesis [Vestrum(2021)], this research led to the exact, algebraic solution shown above, which also guided our search for a solution in higher dimensions, on non-uniform grids.

2D RECTANGULAR GRID SOLUTION

In this section we demonstrate a solution to the factorization of the shifted Laplacian in the general case of a rectangular grid in two dimensions.

A rectangular grid in 2D has different spatial steps sizes Δx, Δy in the x and y directions. In this case, we redefine the shift operators X, Y by the formulas

$$(Xu)(x,y) = u(x-\Delta x, y), \qquad (Yu)(x,y) = u(x, y-\Delta y), \tag{19}$$

and the adjoints of these operator are shifts in the opposite direction, so

$$(X^*u)(x,y) = u(x+\Delta x, y), \qquad (Y^*u)(x,y) = u(x, y+\Delta y). \tag{20}$$

For a non-square grid, the discrete Laplacians need to be normalized in order to get a proper approximation to the continuous Laplacian. It is convenient to define two scaling parameters

$$\lambda_x = (h/\Delta x)^2, \qquad\qquad \lambda_y = (h/\Delta y)^2,$$

that relate the different step sizes to a uniform step h.[4] With this notation, the normalized standard discrete Laplacian is given as

$$\mathcal{L}_0 = \lambda_x(X+X^*) + \lambda_y(Y+Y^*) - 2(\lambda_x+\lambda_y)I \tag{21}$$

and the diagonal Laplacian is

$$\mathcal{L}_1 = \frac{\lambda_x+\lambda_y}{4}(XY + X^*Y + XY^* + X^*Y^* - 4I) \tag{22}$$
$$+ \frac{\lambda_x-\lambda_y}{2}(X - Y + X^* - Y^*). \tag{23}$$

We have normalized the standard and diagonal Laplacians so they agree with the definition for a uniform square lattice, as in the previous section, where $\lambda_x = \lambda_y = 1$. A computation with Taylor series verifies that the interpolated Laplacian

$$\mathcal{L}_\gamma = (1-\gamma)\mathcal{L}_0 + \gamma\mathcal{L}_1 \tag{24}$$

[4]We could set $h = \Delta x$ but it is convenient to use this general form.

is a second order approximate to the continuous Laplacian, for all values of the free parameter γ. That is,

$$\mathcal{L}_\gamma u = h^2 \nabla^2 u + O(h^4). \tag{25}$$

We can factor as in the previous section, and obtain the following result:

Theorem 1 *Fix $\epsilon > 0$. A necessary and sufficient condition for a local grid factorization of the shifted 9-point Laplacian $1 - \epsilon\mathcal{L}_\gamma$ on a 2D rectangular lattice in the form*

$$1 - \epsilon\mathcal{L}_\gamma = (aI + bX + cY + dXY)(aI + bX^* + cY^* + dX^*Y^*)$$

is that

$$\gamma = -\frac{4\epsilon\lambda_x\lambda_y}{\lambda_x + \lambda_y}. \tag{26}$$

In this case, the coefficients for operator $F = aI + bX + cY + dXY$ are given as

$$\begin{aligned} a &= (1 + \sqrt{1 + 4\epsilon\lambda_x})(1 + \sqrt{1 + 4\epsilon\lambda_y})/4 \\ b &= (1 - \sqrt{1 + 4\epsilon\lambda_x})(1 + \sqrt{1 + 4\epsilon\lambda_y})/4 \\ c &= (1 + \sqrt{1 + 4\epsilon\lambda_x})(1 - \sqrt{1 + 4\epsilon\lambda_y})/4 \\ d &= (1 - \sqrt{1 + 4\epsilon\lambda_x})(1 - \sqrt{1 + 4\epsilon\lambda_y})/4. \end{aligned} \tag{27}$$

The proof of the theorem is a careful algebraic computation, involved five quadratic equations in the five unknowns a, b, c, d, γ. The details are contained in the MSc thesis of the second author [Vestrum(2021)].

Consequently, the operator $1 - \epsilon\mathcal{L}_\gamma = FF^*$ can be quickly inverted, by first inverting the first factor operator $F = aI + bX + cY + dXY$ using back-substitution, and then inverting the second factor F^* using forward-substition. The result is an implicit method of speed O(N), where N is the number of points in the 2D grid. This is as fast as the usual explicit methods.

Some numerical examples are given at later sections in this article.

3D AND HIGHER D SOLUTIONS

In this section we demonstrate a solution to the factorization of the shifted Laplacian in the general case of a rectangular grid in three dimensions. That is, where the steps in the x, y and z directions can be different sizes. We also present the solution in bigger than three dimensions, which is of mathematical interest.

With many numerical experiments during the course of this research, it became clear that a solution to the factor problem in 3D should be possible. However, one must be flexible in the choice of the discrete difference operator that is used to approximate the Laplacian. In two dimensions, there are two linearly independent for the discrete Laplacian – for instance the standard Laplacian with the 4 edges, and the diagonal Laplacian.

Interpolating between the two introduces one free parameter, which we called γ in the last section.

In three dimensions, it turns out there are eleven linearly independent choices for the discrete Laplacian, although in most applications one usually only sees three standard ones, the 7, 9, and 13 point stencils. By using all eleven choices, one obtains ten degrees of freedom in the choice of a (normalized) Laplacian, which is enough freedom to solve the factorization problem. The result is the following theorem.

Theorem 2 *Fix $\epsilon > 0$. A solution to the local grid factorization of the shifted affine combination of Laplacians $1 - \epsilon\mathcal{L}_\Diamond$ on a 3D rectangular lattice in the form*

$$1 - \epsilon\mathcal{L}_\Diamond = FF^* \tag{28}$$

with factor

$$F = a_{000}I + a_{100}X + a_{010}Y + a_{001}Z + a_{110}XY + a_{101}XZ + a_{011}YZ + a_{111}XYZ \tag{29}$$

is obtained by choosing the coefficients for the factored operator F as

$$\begin{aligned}
a_{000} &= (1+\Lambda_x)(1+\Lambda_y)(1+\Lambda_z)/8\\
a_{100} &= (1-\Lambda_x)(1+\Lambda_y)(1+\Lambda_z)/8\\
a_{010} &= (1+\Lambda_x)(1-\Lambda_y)(1+\Lambda_z)/8\\
a_{001} &= (1+\Lambda_x)(1+\Lambda_y)(1-\Lambda_z)/8\\
a_{110} &= (1-\Lambda_x)(1-\Lambda_y)(1+\Lambda_z)/8\\
a_{101} &= (1-\Lambda_x)(1+\Lambda_y)(1-\Lambda_z)/8\\
a_{011} &= (1+\Lambda_x)(1-\Lambda_y)(1-\Lambda_z)/8\\
a_{111} &= (1-\Lambda_x)(1-\Lambda_y)(1-\Lambda_z)/8.
\end{aligned} \tag{30}$$

Here, parameters $\Lambda_x, \Lambda_y, \Lambda_z$ specify constants defined by the grid spacing and ϵ,

$$\Lambda_x = \sqrt{1 + 4\epsilon\lambda_x} \qquad \Lambda_y = \sqrt{1 + 4\epsilon\lambda_y} \qquad \Lambda_z = \sqrt{1 + 4\epsilon\lambda_z}.$$

The difference operator $\mathcal{L}_\Diamond = (1 - FF^*)/\epsilon$ is a second order accurate approximation to the continuous Laplacian. That is,

$$\mathcal{L}_\Diamond u = h^2\nabla^2 u + O(h^4). \tag{31}$$

The proof of this result is contained in the thesis of the second author. [Vestrum(2021)]

For simplicity in stating the theorem, we are breezing over a few obvious extensions from 2D to 3D. The linear operators X, Y, Z are shifts on the 3D grid, by step sizes $\Delta x, \Delta y, \Delta z$ in the x, y and z directions, respectively. The parameters $\lambda_x, \lambda_y, \lambda_z$ specify the relative spacing of the 3D rectangular grid to a common spacing of h

$$\lambda_x = (h/\Delta x)^2, \qquad \lambda_y = (h/\Delta y)^2, \qquad \lambda_z = (h/\Delta z)^2, \tag{32}$$

and the linear operation $\mathcal{L}_\diamondsuit$ is an affine combination of 11 independent discrete Laplacians,

$$\mathcal{L}_{11} = (1 - \sum_{i=1}^{10} \gamma_i)\mathcal{L}_{\gamma_0} + \sum_{i=1}^{10} \gamma_i \mathcal{L}_{\gamma_i} \tag{33}$$

whose coefficients can be calculated from the formulas above. However, it is easier just to write $\mathcal{L}_\diamondsuit = (1 - FF^*)/\epsilon$.

It is worth noting that for $\epsilon > 0$ small, the coefficient that appear in factor F have specific sizes. The coefficient of the identity, a_{000}, is close to one, the coefficients for X, Y, Z are of order ϵ, the coefficients of XY, XZ, YZ are of order ϵ^2 and the coefficient of XYZ is of order ϵ^3.

Higher dimensions $> 3D$

From the 3D example, we can deduce the form of the result in $N > 3$ dimensions.[5] We write the factor F as a linear combination of products of shift operators $X_1, X_2, \ldots X_N$, with

$$F = \sum_{i_1 i_2 \ldots i_N} a_{i_1 i_2 \ldots i_n} X_1^{i_1} X_2^{i_2} \cdots X_N^{i_N} \tag{34}$$

where the indices i_k take values 0 and 1 only. The formula for the coefficients are then given as products

$$a_{i_1 i_2 \ldots i_N} = (1 + (-1)^{i_1}\Lambda_1)(1 + (-1)^{i_2}\Lambda_2) \cdots (1 + (-1)^{i_N}\Lambda_N) \tag{35}$$

where the constants $\Lambda_k = \sqrt{1 + 4\epsilon\lambda_k}$ are related to the relative grid spacings $\lambda_k = (h/\Delta x_k)^2$ and the parameter ϵ.

The proof that F results in the factorization of the shifted Laplacian, $I - \epsilon\mathcal{L}_\diamondsuit = FF^*$ follows the 3D proof in the thesis, as an extended exercise in algebra and multidimensional Taylor series formulas. A clever use of multi-index notation as found in multidimensional PDE texts is helpful here. [Folland(2020)]

COMPUTATIONAL EXAMPLES

To verify that the factorization is correct and that the implicit method is indeed fast, of $O(N)$ in computation, we implemented the method in Python on a number of test cases. The code is available on our Github repositories.

2D example

The first example is in two dimensions, representing an acoustic wave travelling at 3000 meter per second, in a region of size 2000 meters by 2000 meters. The initial waveform is a Gaussian impulse centred in the middle of the computational region. Three examples are

[5]Although this is not necessary in seismic imaging, it is a very elegant mathematical result worth highlighting.

given, with grids sizes of 151 by 151 grid points, 151 by 85 points, and 151 by 76 points. This corresponds to a spatial step size of 13.3 meters in the x direction, and 13.3, 23.8 and 26.7 meters in the y direction, respectively. With a time step size of 1 millisecond, we time step through a total of 300 milliseconds of elapsed time, or 300 times steps per simulation.

The results of the calculation are shown in Figure 8, for the square grid (left column), and two rectangular grids (middle and right column). Observe that even though the grid spacing has changed, the computation shows the correct spherical propagation of the waveform in each case. The velocity of the waveform is also correct, as we see the wavefront moving at about 300 meters per 100 millisecond, which is 3000 meters per second. Some artefacts of the numerical approach appear in the rectangular grid examples: for instance, in the bottom right image of Figure 8, corresponding to a step size of 26.7 meters in the y direction at 300 milliseconds into the simulation, we see a shadow near the trailing edge of the wavefront, at the top and bottom of the waveform. This could be the result of the wide spatial sampling in the y direction.

3D examples

The next two examples are in three dimensions, similar to the two dimension example in the previous section. The first represents an acoustic wave travelling at 3000 meter per second, in a region of size 2000 meters along each of the three dimensions. The initial waveform is a Gaussian impulse centred in the middle of the computational region. Three examples are given, with grids sizes of $151 \times 151 \times 151$ grid points, then $151 \times 85 \times 151$ points, and $151 \times 76 \times 151$ points. This corresponds to a spatial step size of 13.3 meters in the x and z directions, and 13.3, 23.8 and 26.7 meters in the y direction, respectively. With a time step size of 1 millisecond, we time step through a total of 300 milliseconds of elapsed time, or 300 times steps per simulation. We are using the augmented implicit method in this example, to verify its performance.

The results of the calculation are shown in Figure 9, for the uniform cubic grid (left column), and two rectangular grids (middle and right column). We present the 2D section through the $z = 0$ plane. Observe that even though the grid spacing has changed, the computation shows the correct spherical propagation of the waveform in each case. The velocity of the waveform is also correct, as we see the wavefront moving at about 300 meters per 100 millisecond, which is 3000 meters per second. Some artefacts of the numerical approach appear in the rectangular grid examples: for instance, in the bottom right image of Figure 9, corresponding to a step size of 26.7 meters in the y direction at 300 milliseconds into the simulation, we see a shadow near the trailing edge of the wavefront, at the top and bottom of the waveform. This could be the result of the wide spatial sampling in the y direction.

Notice that the grid size corresponds to a linear problem size of $N \approx 3.5$ million data points, for which a standard matrix inversion would be computationally prohibitive. The implicit methods described in this article are of speed $O(N)$, as verified in these numerical experiments.

For the final 3D example, shown in Figure 10, we use two sources slightly displaced from the centre of the computational region, of opposite polarity. Again, the computation region is 2000 meters along each dimension, velocity is 3000 meters per second, and time step is 1 millisecond, for a total time of 250 milliseconds. We use a coarser rectangular grid of $101 \times 101 \times 51$ grid points, for a problem size of $N \approx 500,000$ data points. In Figure 10 we see the waveforms advancing in spherical wavefronts as expected, with the proper physical interference correctly represented when the wavefronts pass through each other.

FURTHER WORK

This paper reports on the work for factorization of the shifted Laplacian on square, cubic and rectangular grids in 2D and 3D. The factorization also extends to higher dimensions. We have verified that a similar factorization approach works for triangular grids as well. It is also possible to create factors with "arrows to the right" only, that is factors of the form $F = aI + bX + cXY + dXY^*$. This form is particularly suited for GPU computations, as the back-substitution steps can be performed on one column in parallel. GPU implementations with the standard factorization are also possible, using parallel computations along diagonals.

The variable velocity case, for horizontally stacked layers, can be approached by adjusting the factors $\Lambda_x, \Lambda_y, \Lambda_z$ to implement a change of velocity across layers. The general variable velocity case appears to need a more general approach, where the coefficients a, b, c, d in the factorization (2D case) are replaced by functions of grid position, for instance $a = a(x, y)$.

We observe that there are some similarities between the grid algebra method and the alternating direction implicit (ADI) schemes developed by other authors (eg see [Qin(2009)]). These methods involve addition of perturbation terms to permit operator splitting and results in something similar to a factoring. This is worth investigating further. ADI method are also useful for dealing with boundary conditions, which would be helpful for the grid algebra work.

We are exploring how to best use the implicit method, now that we have a computationally efficient version. The unconditional stability of the implicit method allows one to take large step sizes in the numerical computations, which should be advantageous when stepping through a low frequency waveform in the initial phases of full waveform inversion

SUMMARY

We have shown a factorization of the wave equation on computation grids, which permits a fast and accurate implicit method for a finite difference solution to the acoustic (seismic) wave equation. Numerical demonstrations in 2D and 3D were presented, to demonstrate the utility of the method and to verify the fast, $O(N)$ computational speed.

ACKNOWLEDGEMENTS

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research

Council of Canada) through the grant CRDPJ 543578-19 and the Discovery Grant RGPIN-2020-04561 of the first author.

REFERENCES

[Claerbout(1985)] Claerbout, J. F., 1985, Imaging the earth's interior, vol. 1: Blackwell scientific publications Oxford.

[Folland(2020)] Folland, G. B., 2020, Introduction to Partial Differential Equations: Second Edition: Princeton University Press.
URL `https://doi.org/10.1515/9780691213033`

[Lamoureux and Hardeman-Vooys(2016)] Lamoureux, M. P., and Hardeman-Vooys, H. K., 2016, Grid algebra in finite difference schemes: CREWES Research Report, **28**.

[Myint-U. and Debnath(2006)] Myint-U., T., and Debnath, L., 2006, Linear Partial Differential Equations for Scientists and Engineers: Birkhäuser Boston.

[Press(2009)] Press, W., 2009, Numerical Recipes in C++: The Art of Scientific Computing: Cambridge University Press.

[Qin(2009)] Qin, J., 2009, The new alternating direction implicit difference methods for the wave equations: Journal of Computational and Applied Mathematics, **230**, No. 1, 213–223.

[Selvadurai(2000)] Selvadurai, A., 2000, Partial Differential Equations in Mechanics 1: Fundamentals, Laplace's Equation, Diffusion Equation, Wave Equation, Engineering online library: Springer.

[Smith and Smither(1985)] Smith, G., and Smither, M., 1985, Numerical Solution of Partial Differential Equations: Finite-difference Methods, Oxford applied mathematics and computing science series: Clarendon Press.

[Stone(1968)] Stone, H. L., 1968, Iterative solution of implicit approximations of multidimensional partial differential equations: SIAM Journal on Numerical Analysis, **5**, No. 3, 530–558.

[Vestrum(2021)] Vestrum, R., 2021, Local Factorization of Multidimensional Differential Operators to Optimize Implicit Solution Methods: M.Sc. thesis, University of Calgary, Calgary, AB, Canada.

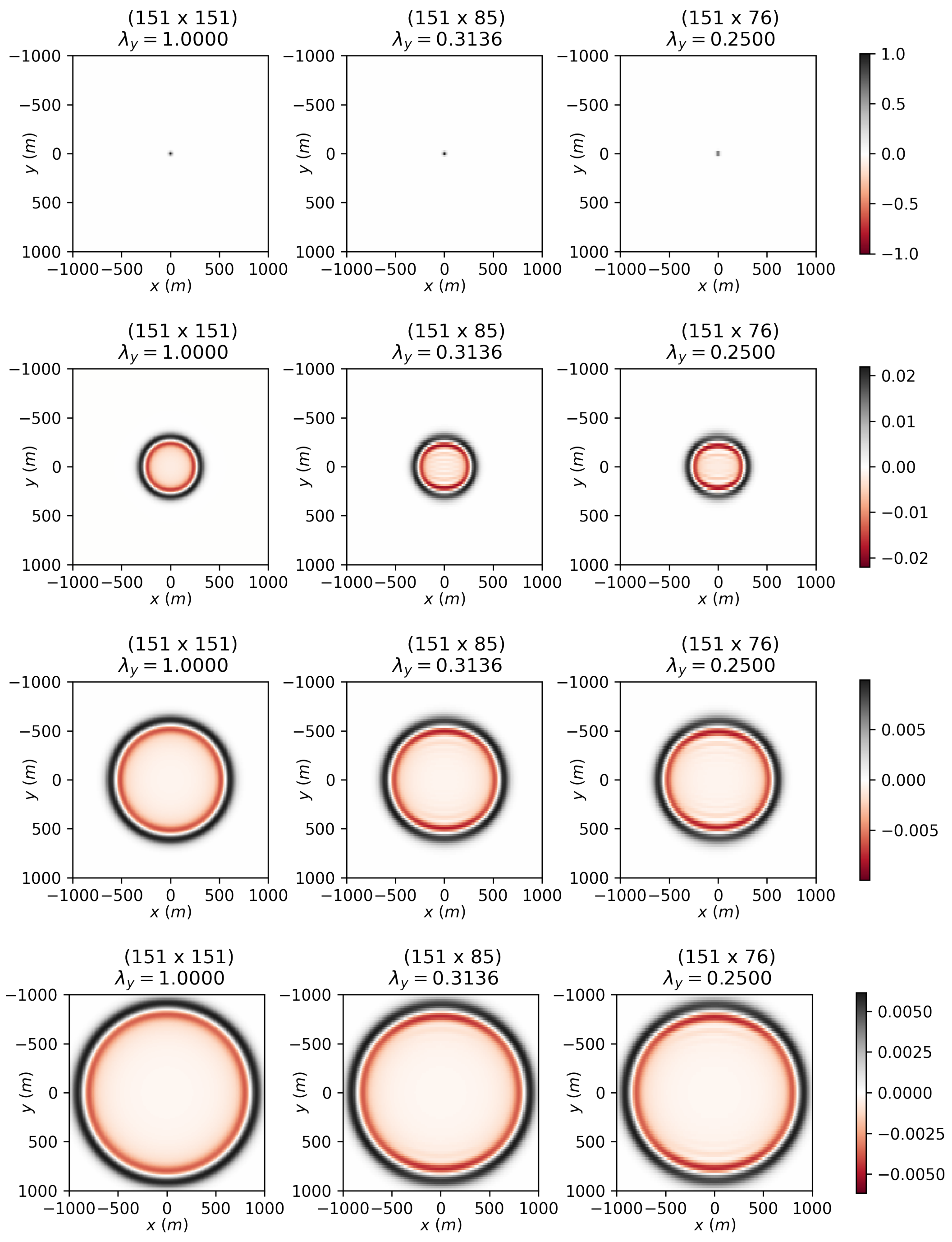

FIG. 8. 2D implicit method, three different grid spacings, $t = 0, 100, 200, 300$ msec

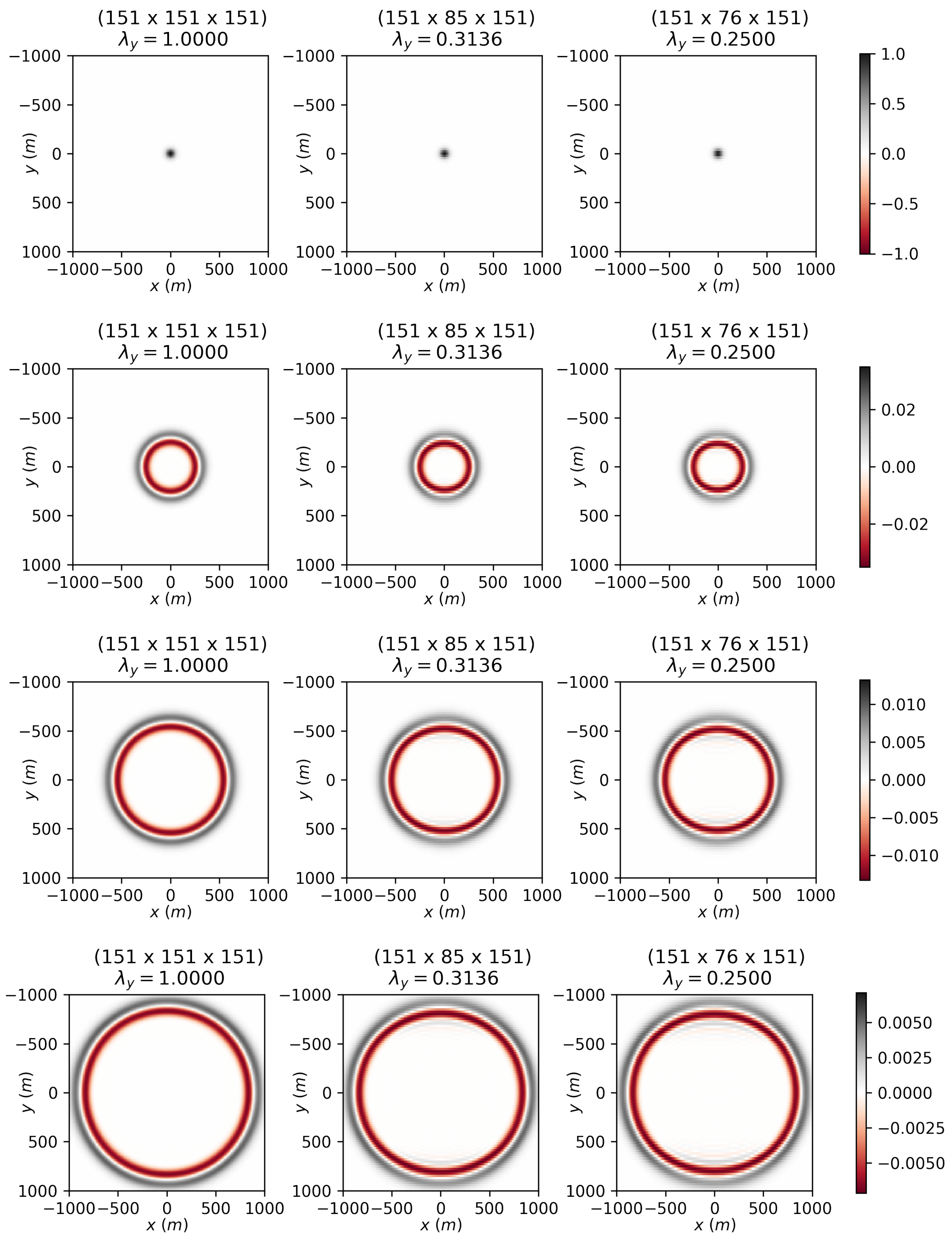

FIG. 9. 3D implicit, three grid spacings, $z = 0$ slice, $t = 0, 100, 200, 300$ msec.

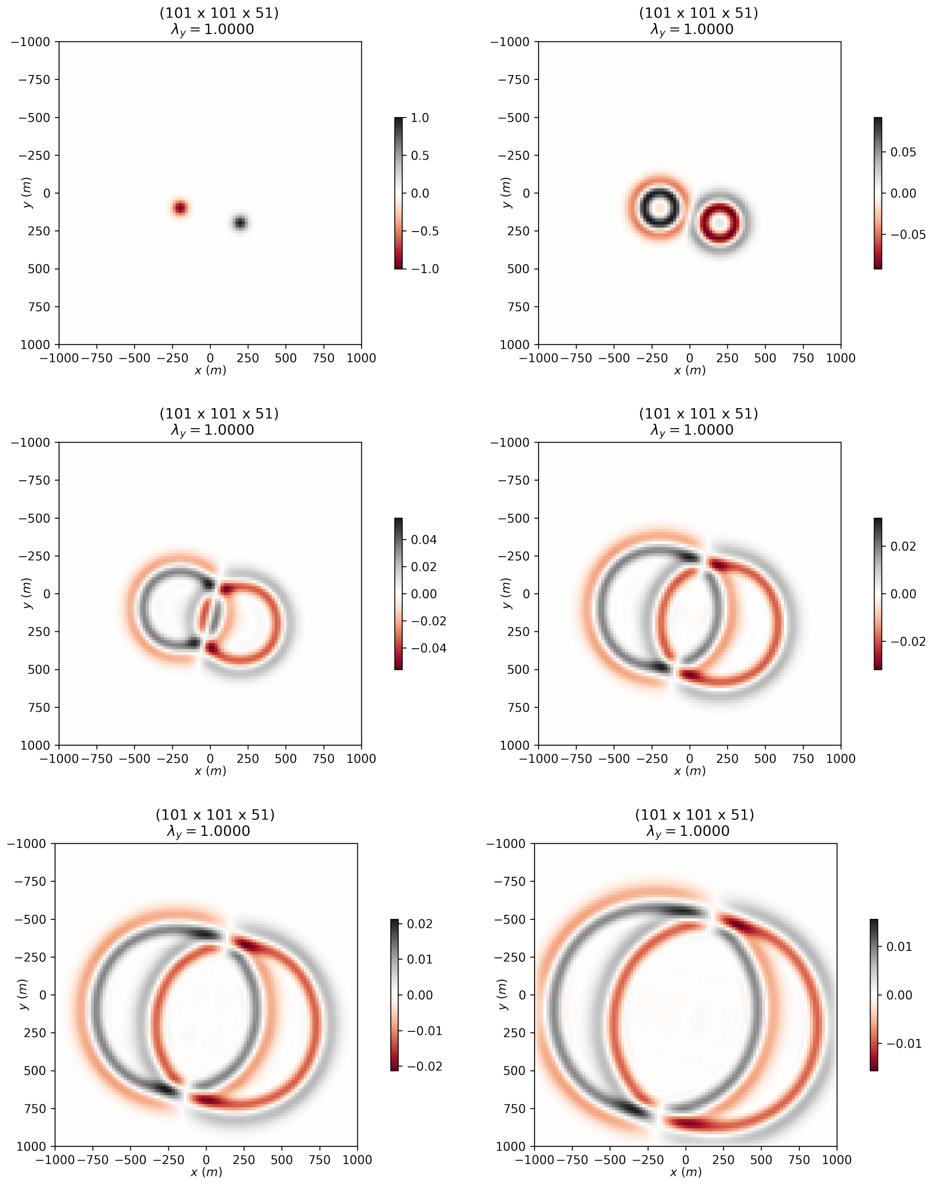

FIG. 10. 3D implicit with two sources, $t = 0, 50, 100, 150, 200, 250$ msec.