Implicit finite difference in time-space domain with the helix transform

Ohad Barak

ABSTRACT

Spectral factorization is a method of creating causal filters which have causal inverses. I use spectral factorization of an implicit finite-difference stencil of the two-way wave equation approximation in order to model wave propagation by a sequence of deconvolutions. I deconvolve this filter's coefficients with the wavefield propagating in a constant velocity medium using the helix approach. In comparison with explicit approximations, implicit approximations have unconditional stability, enabling the use of larger time steps during the modeling process. The advantages are both in reduced computation time, and in the extension and scalability to multiple dimensions enabled by the helix operator.

INTRODUCTION

Implicit finite difference is a widely used method in geophysical data processing, commonly utilized for the approximation of the differential wave equation when extrapolating wavefields. In comparison with explicit methods, implicit finite difference has unconditional numerical stability, thus enabling larger finite differencing steps during the computation. This is an attractive prospect, as the implication is a shorter processing time for wave extrapolation. However, implementing implicit finite difference in a multidimensional problem is not trivial. The method requires the solution of a sparse set of linear equations per each propagation step. The cost of solving these linear equations, and the computational complexity required, becomes unreasonable at anything greater than 2 dimensions. It is possible to split the solver so that only one dimension of the problem is computed at each propagation step, thus reducing the complexity and possibly the amount of resources required for the computation. However, this method may introduce azimuthal anisotropy to the solution if the actual differential equation being solved is non-separable.

Two concepts combine to greatly aid us in this matter. The first is the helix approach, envisaged in Claerbout (1997). The helix effectively enables us to treat multidimensional problems as one dimensional problems. Specifically, it enables execution of multidimensional convolutions as 1-D convolutions, and likewise for deconvolutions. Convolution equates to polynomial multiplication, while deconvolution equates to polynomial division. The application of convolution or deconvolution to

a data set is likened by Claerbout to winding a coil (the filter coefficients) around the data, where the data is treated as a long set of traces combined end-to-end along their fast axis, as shown in Figure 1.

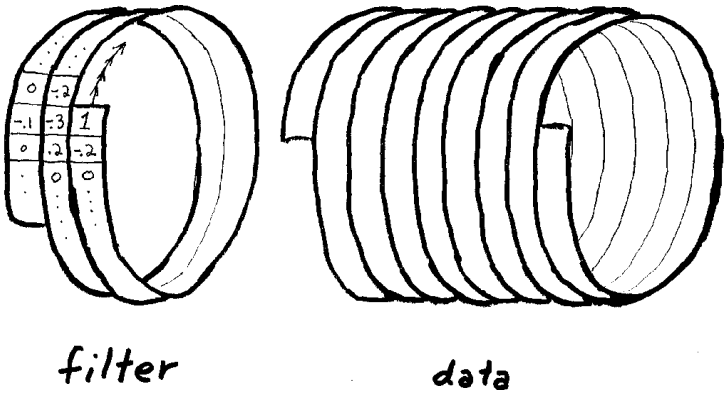

Figure 1: Sketch of the helix concept - convolution takes place by winding a "coil" of filter coefficients over a "coil" of data values (Claerbout (1997)) [**NR**]

The second is the concept of spectral factorization. The purpose of spectral factorization is to input a series of coefficients, and create an alternate set of causal filter coefficients which have a causal inverse. The result will usually be a minimum-phase filter. The autocorrelation of this new set of filter coefficients recreates the original values of the input series. The upshot of this is that application of the original series' coefficients to a dataset is akin to convolving the data with the spectrally factorized filter coefficients in one direction, and then convolving again in the other direction ("coiling" and then "uncoiling" the filter coefficients over the data). This effectively applies the filter and it's time reverse (adjoint) to the data, which amounts to multiplying the data by the original input series' coefficients. In the case of finite differencing, the "input" series might be the Laplacian, which when made to traverse over the data has the effect of a 2nd derivative approximation.

The spectral factorization concept enables us to represent a finite-difference operator as a forward and reverse convolution of filter coefficients. The helix concept disconnects us from the dimensionality of the problem, and enables simple application of 1D convolution and deconvolution to multidimensional problems. Together they enable an alternate method of propagating wavefields - by treating the finite-difference solution as a set of convolutions and deconvolutions.

My aim is to use the helix transform to propagate wavefields in the time-space domain, using an implicit finite-difference approximation of the 2-way acoustic wave equation. For this purpose, I formulate the proper implicit finite-difference weights with regard to the order of the difference approximation and the dimensionality of the problem, and use spectral factorization to create a causal filter with a causal

inverse, whose convolution will equal those coefficients. These filter coefficients are then applied to the wavefield by deconvolution, using the helical coordinate system.

Previous implementations of the helix

Fomel and Claerbout (1997) show how spectral factorization of an implicit finite-difference scheme and application of the resulting filter coefficients with the helix operator can recursively solve the heat conduction equation. They also present an implicit helix-based 3D velocity continuation method for post-stack data.

In Rickett et al. (1998) the one-way wave equation is iteratively solved in the frequency-space domain to downward-continue a wavefield. The extrapolation is robust and efficient. The shortcoming is that since factorization is done for coefficients in the frequency-space domain, the velocity is factorized into the resulting filter coefficients. This forces a workaround for dealing with lateral velocity variations. The suggested method is to create several filters, each with a reference velocity, to be applied to different parts of the wavefield.

EXPLICIT FINITE DIFFERENCE IN 2D USING SPECTRAL FACTORIZATION AND HELICAL COORDINATES

In order to test the general methodology and the programing modules, I first tested whether an explicit 2D finite-difference approximation in time-space domain done by spectral factorization and the helix transform properly emulates "standard" finite difference (i.e. using a differencing "star" which traverses over the data). The difference approximation was 2nd order in time and 2nd order in space:

$$\frac{P^{t+1}_{x,z} - 2P^{t}_{x,z} + P^{t-1}_{x,z}}{\Delta t^2} = C^2 \left(\frac{P^{t}_{x+1,z} - 2P^{t}_{x,z} + P^{t}_{x-1,z}}{\Delta x^2} + \frac{P^{t}_{x,z+1} - 2P^{t}_{x,z} + P^{t}_{x,z-1}}{\Delta z^2} \right) \quad (1)$$

where P is the pressure wavefield, t, x and z are the time and space coordinate indices, and Δt, Δx, and Δz are the temporal and spatial step sizes.

The 2D laplacian representing the spatial derivative's coefficients is:

$$\nabla^2 \approx \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix} \quad (2)$$

In helical coordinates, the laplacian is represented as a sparse 1-D array:

$$\nabla^2 \approx \begin{pmatrix} 1 & \cdots & 1 & -4 & 1 & \cdots & 1 \end{pmatrix} \quad (3)$$

The dots represent the number of elements required for the wrap-around of the helix along the "fast" axis of the 2D grid (see Figure 1). The greater the number of coefficients used by the spectral factorization, the closer the convolution of those coefficients will be to the desired operator, in this case the laplacian in Equation 3. In this experiment, I used the SEPlib `wilson` module to factorize the laplacian into 28 coefficients. The following is a partial selection of those coefficients:

$$h = \left(\begin{array}{ccccccccc} & & & & & 1.0 & -.363 & -.0024 & -.0013 \quad \cdots \\ \cdots & -.0024 & -.0048 & -.113 & -.3114 & & & & \end{array} \right).$$

Convolving this series with it's adjoint results in the following coefficients:

$$h' * h = \left(\begin{array}{ccccccccc} 0.9999771 & \cdots & 0.9999771 & -4.0000000 & 0.9999771 & \cdots & 0.9999771 \end{array} \right) \approx \nabla^2$$

The remaining coefficients at close proximity to the laplacian coefficients were of 4 or more magnitudes smaller than the ones displayed. The rest were all zeros. This led me to the conclusion that 28 coefficients are sufficient.

I used the SEPlib `helicon` module to convolve the coefficients in (4) with the data, coiling first in one direction, and then uncoiling in the other direction. The `helicon` module implements convolution by the linear operator:

$$y_k = x_k + \sum_{i=1}^{N_a} a_i x_{k-i}, \tag{4}$$

where x is the input data vector, a is a causal filter of length N_a, and y is the output vector. The module also implements the adjoint:

$$x_k = y_k + \sum_{i=1}^{N_a} a'_i y_{k+i}, \tag{5}$$

where a' is the time reverse of filter a.

The application of the forward and reverse convolution replicates the act of using the standard finite differencing "star" to approximate the spatial derivative. The spatial derivative is then used to estimate the temporal derivative, and forward propagate in time the values of the acoustic wavefield.

With reference to Equation 1, the propagation kernel is:

$$P^{t+1}_{x,z} = 2P^t_{x,z} - P^{t-1}_{x,z} + (h' * h)P^t_{x,z}. \tag{6}$$

A comparison of propagation of a source in the center of a 2D acoustic wavefield with regular finite differencing and with the helix derivative is shown in Figure 2.

Although qualitatively the two results are similar, the wavefield created by the helix spatial derivative displays some form of dispersion both leading and trailing the main wavelet. Furthermore, the wavefront itself and the dispersion appear to be propagating anisotropically, with a slightly faster component along the positive diagonal. A possible explanation of this outcome is that the filter operator which convolves the data does not have a symmetric angular coverage. Refering to Figure 3, the better coverage of updip angles is apparent in comparison with downdip angles, as the filter is coiled around the data (in the direction of the arrow). If this is indeed the cause of dispersion, one way it can be mitigated is by using a cube finite differencing stencil (Haohuan et al. (2009)).

Another problem which is not apparent in Figure 2 is the wrap-around effect of the helix operator on the data being propagated. Since the helix wraps the filter convolution around one of the dimensions, it will invariably mix the information from one side of the wavefield with that from the other side. The longer and more accurate the filter will be, the more pronounced this effect will be as well. There is no immediate remedy for this inherent property of the helix, except for the standard model-padding solution. However, since the general intention is to use the helix for 2-way wavefield propagation in reverse time migration, one way the wrap around effect can be disregarded is by utilizing random boundaries in the wavefield, as shown in Clapp (2009).

EXPLICIT VS. IMPLICIT FINITE DIFFERENCE APPROXIMATION OF THE 2-WAY ACOUSTIC WAVE EQUATION

Formulation for 1 dimension

The two-way acoustic wave equation in one dimension reads:

$$\frac{\partial^2 P}{\partial t^2} = C^2 \frac{\partial^2 P}{\partial x^2}. \tag{7}$$

In order to formulate it as an implicit finite-difference approximation, I first looked to the formulation of the implicit finite-difference operator of the 1-way wave equation in frequency-space domain in Claerbout (2009) Chapter 9. The Crank-Nicolson differencing method is used to create an implicit finite-difference approximation for downward-continuation of a wavefield:

$$\frac{P^x_{z+1} - P^x_z}{\Delta z} = \frac{v}{-i\omega 2}\left(\frac{P^{x+1}_z - 2P^x_z + P^{x-1}_z}{2\Delta x^2} + \frac{P^{x+1}_{z+1} - 2P^x_{z+1} + P^{x-1}_{z+1}}{2\Delta z^2}\right). \tag{8}$$

The Crank-Nicolson method achieves better accuracy by balancing the 2nd derivative between the current wavefield values (at depth z) and at the next as-of-yet unknown

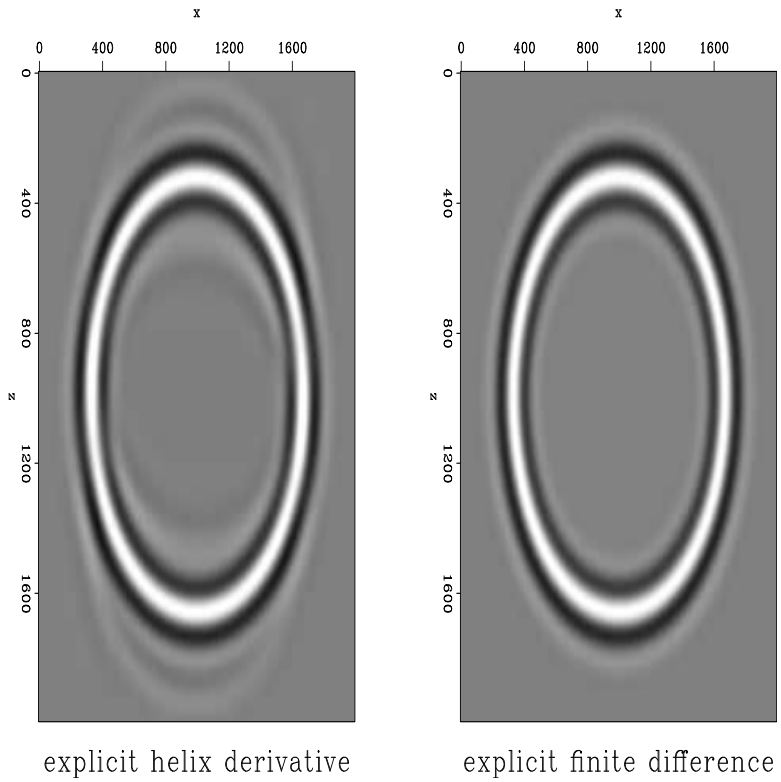

Figure 2: comparison of 2nd order in time and space finite difference Vs. helix derivative after spectrally factorizing the coefficients of the 2nd order spatial derivative and convolving them with the data using helical boundaries. The propagation here is with constant velocity $v = 3000m/s$. [**ER**]

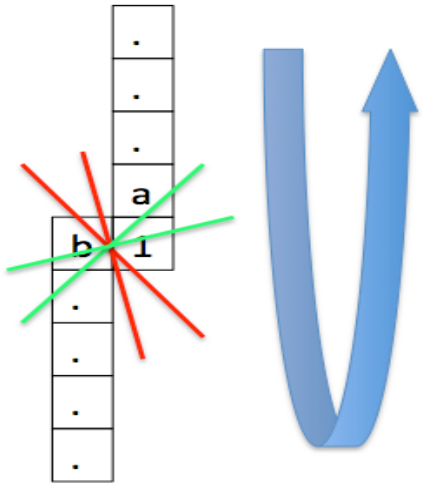

Figure 3: Angular anisotropy of helical filter operator [**NR**]

wavefield values (at depth $z+1$). Hence the division by 2 in the denominators of Equation 8. Borrowing from this methodology, I attempted to formulate a scheme which balances the values of the wavefield at known and unknown locations. Since Equation 7 has a second derivative on the left hand side (as opposed to first derivative in Equation 8), it implies that this balancing must be done over three time "locations". In one dimension, these considerations led me to the following approximation:

$$\frac{P_x^{t+1}-2P_x^t+P_x^{t-1}}{\Delta t^2}=\frac{C^2}{3\Delta x^2}\left[\left(P_{x+1}^{t+1}-2P_x^{t+1}+P_{x-1}^{t+1}\right)+\left(P_{x+1}^{t}-2P_x^{t}+P_{x-1}^{t}\right)+\left(P_{x+1}^{t-1}-2P_x^{t-1}+P_{x-1}^{t-1}\right)\right] \tag{9}$$

Note that the division by 3 effectively averages the spatial derivative between the three time steps: $t-1$, t and $t+1$. In order to propagate the wavefield, the values of the the wavefield at time $t+1$ must be equated to the values at times t and $t-1$.

Application of spectral factorization and the helix operator

The linear system which must be solved according to Equation 9 has the form:

$$\begin{pmatrix} U_0 & U_1 & 0 & 0 \\ U_1 & U_0 & U_1 & 0 \\ 0 & U_1 & U_0 & U_1 \\ 0 & 0 & U_1 & U_0 \end{pmatrix}\begin{pmatrix} P_1^{t+1} \\ P_2^{t+1} \\ P_3^{t+1} \\ P_4^{t+1} \end{pmatrix} = \begin{pmatrix} V_0 & V_1 & 0 & 0 \\ V_1 & V_0 & V_1 & 0 \\ 0 & V_1 & V_0 & V_1 \\ 0 & 0 & V_1 & V_0 \end{pmatrix}\begin{pmatrix} P_1^{t} \\ P_2^{t} \\ P_3^{t} \\ P_4^{t} \end{pmatrix} + \begin{pmatrix} W_0 & W_1 & 0 & 0 \\ W_1 & W_0 & W_1 & 0 \\ 0 & W_1 & W_0 & W_1 \\ 0 & 0 & W_1 & W_0 \end{pmatrix}\begin{pmatrix} P_1^{t-1} \\ P_2^{t-1} \\ P_3^{t-1} \\ P_4^{t-1} \end{pmatrix}. \tag{10}$$

For simplicity, we can combine all the constants into one: $\alpha=\frac{C^2\Delta t^2}{3\Delta x^2}$. The matrix coefficients in Equation 10 (the finite-difference weights) are then:
$U_0=1+2\alpha, \qquad U_1=-\alpha;$
$V_0=2-2\alpha, \qquad V_1=\alpha;$
$W_0=-1-2\alpha, \quad W_1=\alpha.$

In shorter notation, Equation 10 reads:

$$UP^{t+1}=VP^t+WP^{t-1}. \tag{11}$$

The solution of this linear system is:

$$P^{t+1}=U^{-1}\left(VP^t+WP^{t-1}\right). \tag{12}$$

To solve this system, we must perform polynomial division. The system is tridiagonal (and easily solvable) only for 1 dimension. For multiple dimensions, matrix

U is block diagonal. Additional non-zero elements appear at a certain offset from the diagonal, making the solution process more complicated. However, using spectral factorization, the finite-difference weights of matrix U can be factorized into a set of causal filter coefficients u and it's time reverse u^T. Using the helical approach to deconvolution, the system can be recast as:

$$P^{t+1} = (u^T u)^{-1} \left(V P^t + W P^{t-1} \right); \tag{13}$$

$$P^{t+1} = u^{-1} (u^T)^{-1} \left(V P^t + W P^{t-1} \right). \tag{14}$$

As stated above, polynomial division is equal to deconvolution. This means that the polynomial division in Equation 12 can be achieved by a set of two deconvolutions of the spectrally factorized coefficients u of matrix U. One deconvolution is done along the data in the reverse direction (application of the adjoint of the filter):

$$y_k = x_k - \sum_{i=1}^{N_a} a'_i y_{k-i}, \tag{15}$$

where a' is the time reversed filter coefficients of u. The other deconvolution is done in the forward direction:

$$x_k = y_k - \sum_{i=1}^{N_a} a_i x_{k-i}. \tag{16}$$

I used the SEPlib module `polydiv`, which uses the helical coordinates to do the deconvolutions (the polynomial division) in equations 15 and 16.
The wavefield propogation is done by the following sequence:

1. Spectrally factorize the coefficients of matrix U.
2. Multiply the saved wavefield at time $t-1$ by the coefficients of matrix W.
3. Multiply the saved wavefield at time t by the coefficients of matrix V.
4. Sum the results of the previous 2 steps into a result vector.
5. Uncoil the factorized coefficients u over the result vector (eq. 15).
6. Coil the factorized coefficients u over the result vector (eq. 16).

This sequence is repeated for each time step.

Formulation for 2 dimensions

The finite-difference weights for the 2-D case of matrix U (where $\Delta x = \Delta z$) are derived by using the same finite-difference approximation (2nd order in time and space) as in Equation 9. The derived linear equation system is similar to the one shown in (10), except that two off-diagonal bands appear at a certain offset from the main diagonal. The offset is equal to the number of elements of the "fast" axis of the 2D wavefield.

$$\begin{pmatrix} U_0 & U_1 & \dots & U_1 \\ U_1 & U_0 & U_1 & \dots \\ \dots & U_1 & U_0 & U_1 \\ U_1 & \dots & U_1 & U_0 \end{pmatrix} \begin{pmatrix} P_1^{t+1} \\ P_2^{t+1} \\ P_3^{t+1} \\ P_4^{t+1} \end{pmatrix} = \begin{pmatrix} V_0 & V_1 & \dots & V_1 \\ V_1 & V_0 & V_1 & \dots \\ \dots & V_1 & V_0 & V_1 \\ V_1 & \dots & V_1 & V_0 \end{pmatrix} \begin{pmatrix} P_1^{t} \\ P_2^{t} \\ P_3^{t} \\ P_4^{t} \end{pmatrix} + \begin{pmatrix} W_0 & W_1 & \dots & W_1 \\ W_1 & W_0 & W_1 & \dots \\ \dots & W_1 & W_0 & W_1 \\ W_1 & \dots & W_1 & W_0 \end{pmatrix} \begin{pmatrix} P_1^{t-1} \\ P_2^{t-1} \\ P_3^{t-1} \\ P_4^{t-1} \end{pmatrix}, \quad (17)$$

where the finite-difference weights are:
$U_0 = 1 + 4\alpha, \qquad U_1 = -\alpha;$
$V_0 = 2 - 4\alpha, \qquad V_1 = \alpha;$
$W_0 = -1 - 4\alpha, \quad W_1 = \alpha,$
and where $\alpha = \frac{C^2 \Delta t^2}{3 \Delta x^2}$.

The propagation methodology is the same as shown in Equation 14 - creation of a solution vector, and then two deconvolutions using the SEPlib `polydiv` module.

The main advantage in comparison to a standard linear equation system solver is in the reduced number of operations required to propagate the wavefield by one time step. The computation time scales linearly with the size of the data and the number of factorized filter coefficients, instead of the square of the size of the data. Coupled with the larger time steps which are made possible by an implicit finite-difference scheme, this methodology has the potential to reduce total processing times for wavefield propagation algorithms. The further advantage is the separation of the solution from the dimensionality of the problem.

Validity tests of derived coefficients

Before implementing the methodology discussed above, I tested whether the finite-difference weights derived in Equation 10 for the implicit approximation are indeed reversible by a set of convolution and deconvolution operators. The input impulse

used for the tests is shown in Figure 4. The sequence of operations applied to this impulse were:

1. forward deconvolution with the spectrally factorized coefficients of the above finite-difference weights.
2. adjoint deconvolution.
3. adjoint convolution.
4. forward convolution.

The sequence and it's results are shown in Figure 5. The final result is not a perfect impulse, however the remaining values in the wavefield are 6 or more orders of magnitude smaller than the central impulse.

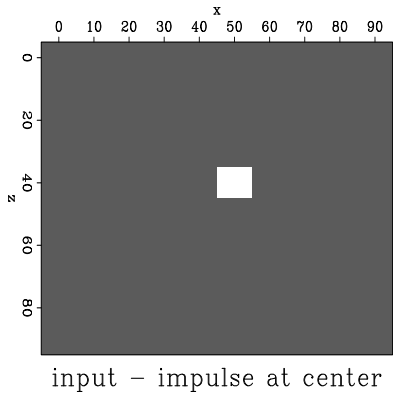

Figure 4: Input impulse used for helical convolution test [**ER**]

In order to test whether the derived implicit finite-difference coefficients are actually valid for wavefield propagation, the next step was to compare standard explicit finite difference to implicit propagation with these coefficients in one dimension. The implicit solution was done using the SEPlib module `rtris` which recursively solves a tridiagonal equation system. The results are in Figure 6. Severe dispersion is visible in the implicitly propagated wavefield. So far I have been unable to remove this dispersion without damaging the kinematics of the wavefield.
On the other hand, the wavefield derived by implicit finite difference does not diverge even when increasing the temporal time step beyond the stability limit of the explicit solver.

STANDARD IMPLICIT PROPAGATION VS. HELICAL IMPLICIT PROPAGATION

Implicit finite-difference propagation of a wavefield is the solution of the system shown in Equation 17. I tested wavefield propagation with constant velocity using a standard

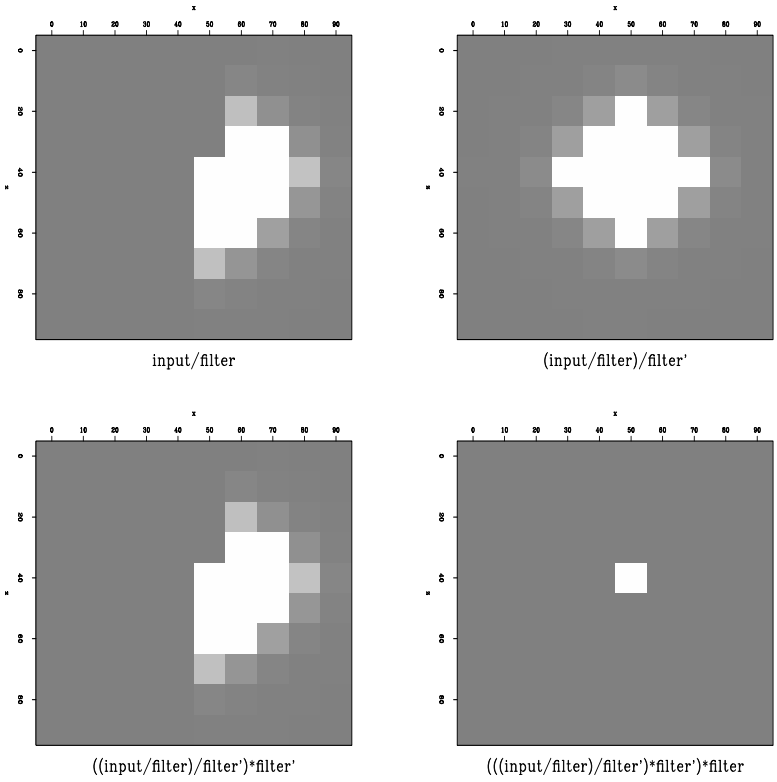

Figure 5: Results of convolving and deconvolving with filter coefficients whose convolution yields the finite-difference weights required by the implicit finite-difference approximation in Equation 17 [**ER**]

linear equation solver, and compared it to the wavefield propagated by repeated deconvolutions with spectrally factorized coefficients of the same finite-difference weights (Equation 14). Figure 7 shows this comparison - standard linear system solver is on the left, deconvolutions with spectrally factorized coefficients are on the right. The bottom figures were created with a larger time steps than the top ones. For the same time step size, the methods produce similar images. Both of the bottom images show greater dispersion than the top ones. This suggests that the dispersion (visible also in Figure 6) is an intrinsic property of the finite-difference weights derived by the formulation in Equation 9, and is not the result of the spectral factorization method.

Propagation with a linear equation solver is time consuming, which is why the wavefields in Figure 7 are rather small (only 100x100 elements). To test whether this method is stable over longer periods, I compared explicit wavefield propagation to the helical implicit propagation. These are shown in Figure 8. The figure is after 4 seconds of propagation, with a time step almost equal to the stability limit of the 2nd order explicit finite-difference scheme. The explicit propagation exhibits less dispersion. However, it diverges for greater time step sizes, whereas the implicit solution does not. The dispersion problem however does get worse for the implicit solver as the time step increases.

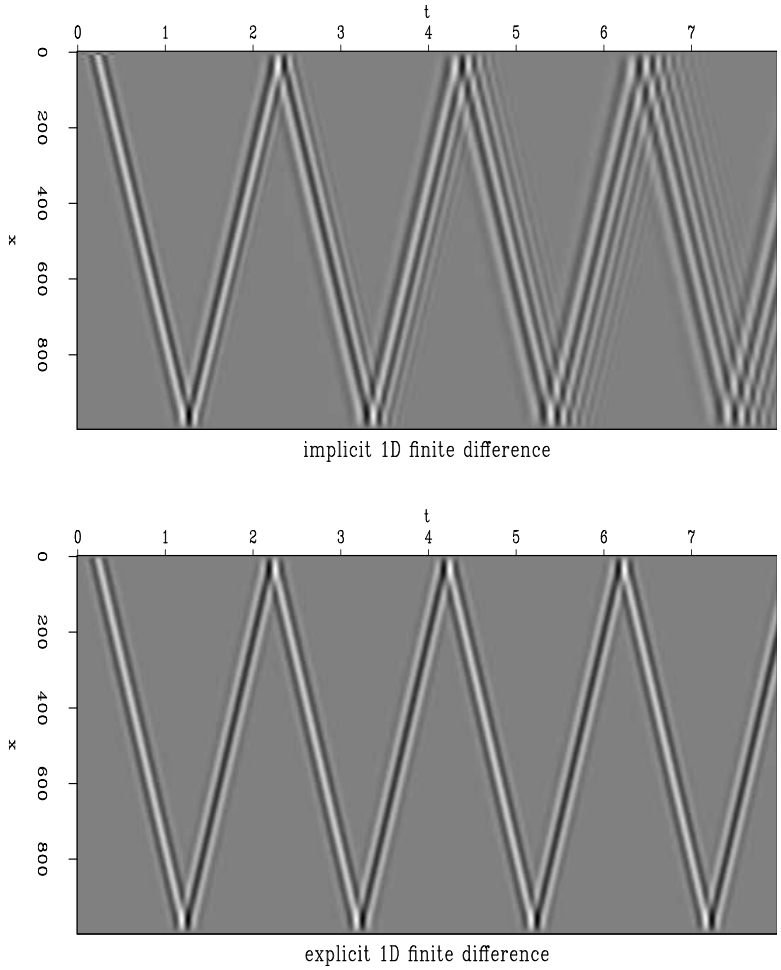

Figure 6: 1D Explicit Vs. Implicit finite difference with constant velocity $= 1000m/s$. Source is a Ricker wavelet with central frequency $= 12.5Hz$. The explicit time step was $\Delta t = 4msec$. The implicit time step was $\Delta t = 8msec$. $\Delta x = 10m$. [**ER**]

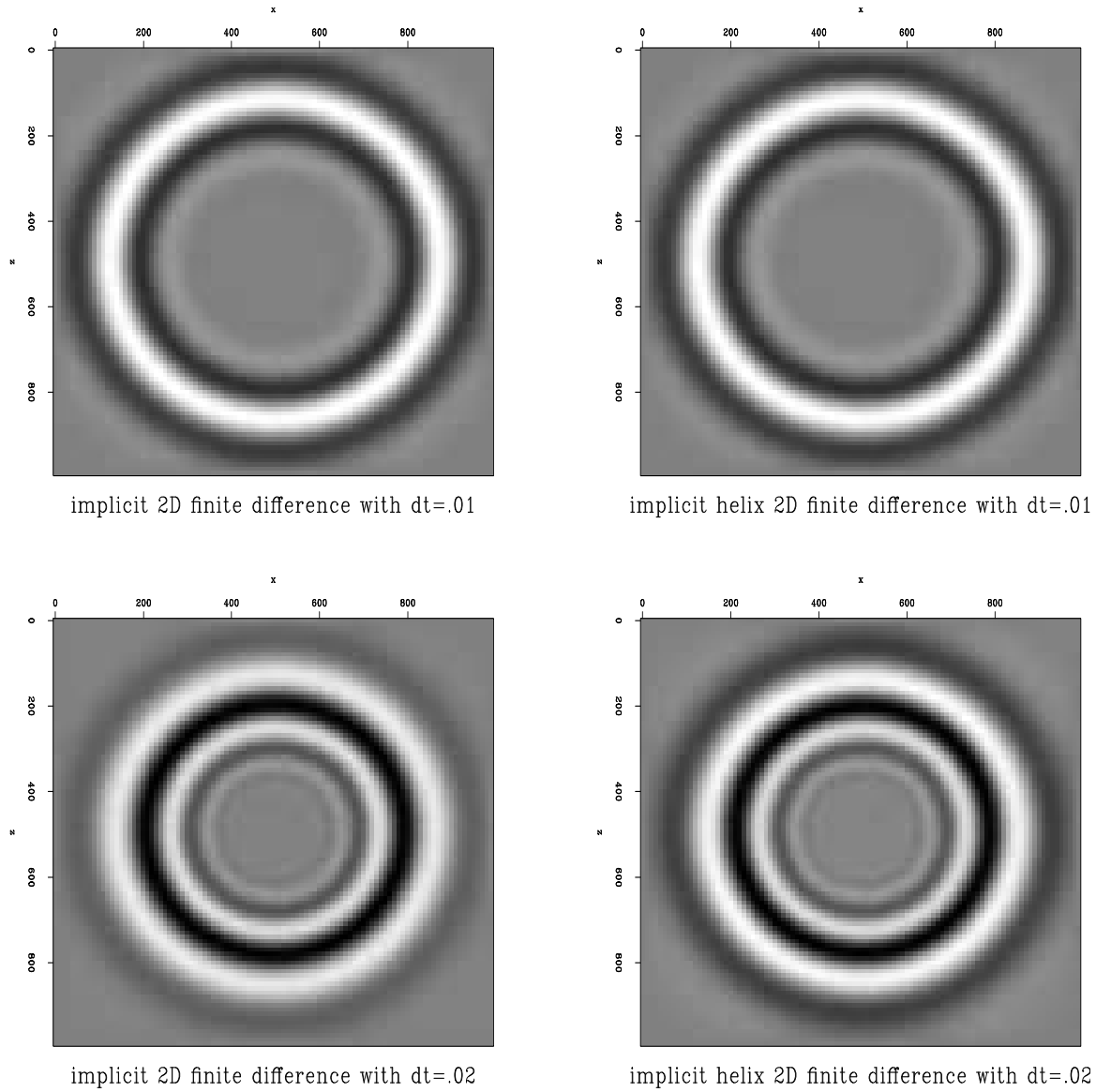

Figure 7: 2D Implicit Vs. Helical Implicit finite difference with constant velocity = $1000m/s$. Source is a Ricker wavelet with central frequency = $12.5Hz$. The time step for the top figures was $\Delta t = 10msec$, and for the bottom figures $\Delta t = 20msec$. $\Delta x = \Delta z = 10m$. [**CR**]

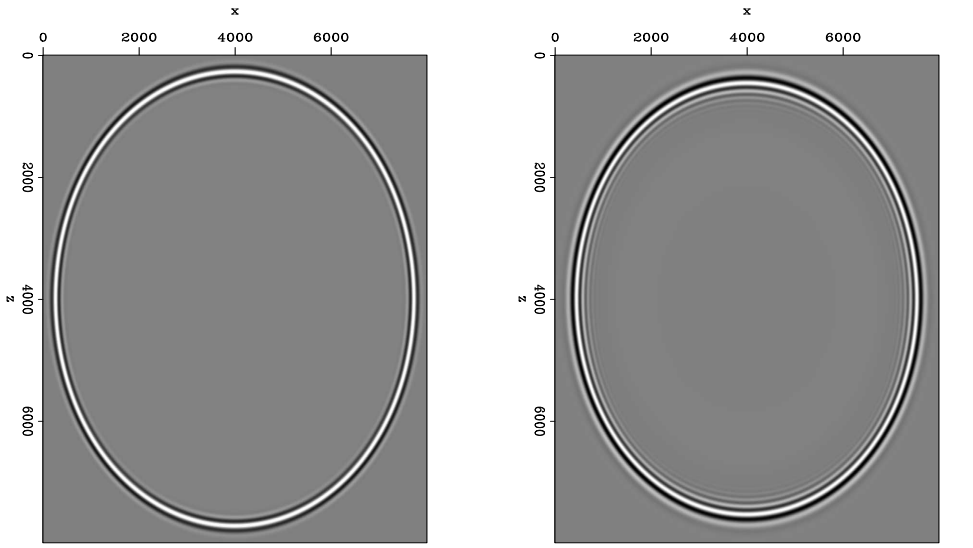

Figure 8: 2D Explicit Vs. Helical Implicit finite difference with constant velocity = $1000m/s$. Source is a Ricker wavelet with central frequency = $12.5Hz$. The time step was $\Delta t = 7msec$ - at the stability limit of the explicit scheme for the 2nd order in time and space approximation. $\Delta x = \Delta z = 10m$. [**ER**]

Non-separability of velocity from factorized implicit coefficients

The initial impetus for using deconvolutions by spectrally factorized coefficients of an implicit finite-difference scheme to approximate the 2-way wave equation was that the velocity could be separated from the scheme's coefficients. The hope was that if this was possible, a filter with constant coefficients could be created to handle propagation through a variable velocity medium. This has turned out to not be the case, at least for the formulation of the implicit scheme in Equation 9. If we look at the left hand side of Equation 17:

$$\begin{pmatrix} 1+4\alpha & -\alpha & \dots & -\alpha & \dots \\ -\alpha & 1+4\alpha & -\alpha & \dots & -\alpha \\ \dots & -\alpha & 1+4\alpha & -\alpha & \dots \\ -\alpha & \dots & -\alpha & 1+4\alpha & -\alpha \\ \dots & -\alpha & \dots & -\alpha & 1+4\alpha \end{pmatrix} \begin{pmatrix} P_1^{t+1} \\ P_2^{t+1} \\ P_3^{t+1} \\ P_4^{t+1} \\ P_5^{t+1} \end{pmatrix}. \tag{18}$$

It is clearly impossible to divide the system by α or any of it's constituent parts (in particular - the velocity) without changing the value of the diagonal weight. If the velocity is variable, this will result in a different set of finite-difference weights at various lines of the linear system in (17), requiring a different set of spectrally factorized coefficients wherever the velocity changes.

The option of using a "filter bank" for different parts of the wavefield according to the local velocity has already been discussed in Rickett et al. (1998), for wave propagation in the frequency-wavenumber domain. This may also be applicable to propagation in the time-space domain, but I have not yet tested it. It was my hope that this would be unecessary, and that a single set of filter coefficients could be utilized for the entire wavefield irrespective of velocity. This would make the propagation algorithm simpler, and more amenable to future parallelization schemes.

Stability analysis

The Von-Neumann stability analysis is useful in predicting the largest time steps possible for a particular order of a finite-difference scheme, for which the wavefield will not diverge. The application of this analysis to the 2nd order in time and space explicit finite-difference approximation of the 2-way wave equation follows. Assuming $h = \Delta x = \Delta z$, and using t as a time index, the approximation is:

$$U^{t+1}_{hj,hl} = 2U^{t}_{hj,hl} - U^{t-1}_{hj,hl} + \frac{C^2\Delta t^2}{h^2}(U^{t}_{h(j+1),hl} + U^{t}_{h(j-1),hl} + U^{t}_{hj,h(l+1)} + U^{t}_{hj,h(l-1)} - 4U^{t}_{hj,hl}). \tag{19}$$

The field being propagated is some function of time combined with a harmonic function of space:

$$U^t = F^t e^{i(k_x hj + k_z hl)}. \tag{20}$$

inserting 20 into 19 and then dividing by $e^{i(k_x hj + k_z hl)}$ yields:

$$F^{t+1} = 2F^t - F^{t-1} + \alpha\left(F^t e^{ik_x h} + F^t e^{-ik_x h} + F^t e^{ik_z h} + F^t e^{-ik_z h} - 4\right), \tag{21}$$

where $\alpha = \frac{c^2\Delta t^2}{h^2}$.

In order to have stable propagation, the amplification factor - the amplitude ratio between the future wavefield and the current wavefield, must be smaller or equal to 1. This is also a requirement for the ratio between the past wavefield and the current wavefield, as the time reversed wavefield must also remain stable. From this consideration we have:

$$\frac{F^{t+1}}{F^t} = \frac{F^{t-1}}{F^t} = g(k_x, k_z). \tag{22}$$

Dividing (21) by F^t, and using the trigonometric identity for cosine we get:

$$g(k_x, k_z) = 1 + \alpha\left(cos(k_x h) + cos(k_z h) - 2\right) = 1 + \alpha R. \tag{23}$$

R is bounded by $-4 \le R \le 0$, and the requirement is that the amplification factor

$|g| \leq 1$. It follows that

$$\begin{aligned} 1 - 4\alpha \leq g \leq 1, \\ |1 - 4\alpha| \leq 1, \\ \alpha \leq \frac{1}{2}. \end{aligned} \tag{24}$$

Since $\alpha = \frac{C^2 \Delta t^2}{h^2}$, this analysis provides us with a way to determine the maximum time step for a given minimum velocity and spatial differencing step.
The same derivation for the 2D implicit finite differencing weights as derived in Equation 17 yields the amplification factor:

$$g(k_x, k_z) = \frac{1 + \alpha R}{1 - 2\alpha R}. \tag{25}$$

The boundaries for R remain $-4 \leq R \leq 0$. Because α is necessarily positive, it follows that $|g| \leq 1$ for any α in this 2D implicit finite-difference scheme. The time step can be arbitrarily large without causing the wavefield to diverge. This, of course, does not mean that we will get a $useful$ wavefield with any arbitrary Δt.

CONCLUSION AND FUTURE WORK

It seems, at least in principle, that the general methodology described above for 2-way wave extrapolation with implicit finite difference by deconvolution using spectral factorization and the helix should be possible. The decrease in runtime in comparison to standard implicit finite difference application must be properly measured to assess the desirability of the method, although in my preliminary tests it is very evident that the method presented here is much faster than when using a standard solver.

The problem of dispersion of the wavefield when using the weights in Equation 10 must be overcome. It is not yet obvious to me what part of the observed dispersion can be attributed to the filter coefficients and what part to the finite-difference scheme. If the factorization is the main contributor, then one possible solution is to slightly increase the value of the diagonal element of matrix U in Equation 10, in order to improve the accuracy of the spectrally factorized filter coefficients. The connection between that and the number of factorized coefficients used in the filter is also as of yet unclear.

Another method of reducing the dispersion is to reformulate the implicit scheme with different weights. I have made one preliminary test in one dimension which suggests that doubling the value of the weights of matrix V in Equation 10 removes some of the dispersion. In any case, A more rigorous analysis of the phase velocity as a function of frequency of this or any other implicit scheme must follow.

Polynomial division is a necessarily sequential operation, as each output is dependent on the previous outputs (Equation 16). This seems to preclude any possibilty of parallelization of the process within each time step. I do not see a way around this at the moment, but where there's a will there's a way.

ACKNOWLEDGMENTS

I wish to thank Ali Almomin for closing my linear algebra holes and explaining what polynomial division actually is, and also Biondo Biondi and Robert Clapp for suggesting this propagation method (and patiently explaining it...).

REFERENCES

Claerbout, J., 2009, Basic earth imaging: Stanford University.

Claerbout, J. F., 1997, Multidimensional recursive filters via a helix: Stanford Exploration Project, **Report 95**, 1–13.

Clapp, R., 2009, Reverse time migration with random boundaries: SEG Expanded Abstracts, **28**, 2809.

Fomel, S. and J. F. Claerbout, 1997, Exploring three-dimensional implicit wavefield extrapolation with the helix transform: Stanford Exploration Project, **Report 95**, 43–61.

Haohuan, F., R. Clapp, O. Mencer, and O. Pell, 2009, Accelerating 3d convolutions using streaming architectures on fpgas: SEG Expanded Abstracts, **28**, 3035.

Rickett, J., J. F. Claerbout, and S. Fomel, 1998, Implicit 3-d depth migration by wavefield extrapolation with the helical boundary conditions: SEG Expanded Abstracts, 1124–1127.