Computing 3D Frequency-Domain Waveform Kernels for c(x,y,z) Media

Jonathan B. Ajo-Franklin
Earth Resources Laboratory
Massachusetts Institute of Technology

Kurt T. Nihei
Chevron Energy Technology Company

Xander Campman
Earth Resources Laboratory
Massachusetts Institute of Technology

M. Nafi Toksöz
Earth Resources Laboratory
Massachusetts Institute of Technology

May 28, 2006

Abstract
Seismic tomography, as typically practiced on both the exploration, crustal, and global scales, considers only the arrival times of selected sets of phases and relies primarily on WKBJ theory during inversion. Since the mid 1980’s, researchers have explored, largely on a theoretical level, the possibility of inverting the entire seismic record. Due to the ongoing advances in CPU performance, full waveform inversion is finally becoming feasible on select problems with promising results emerging from frequency-domain methods. However, frequency-domain techniques using sparse direct solvers are currently constrained by memory limitations in 3D where they exhibit a $O(n^4)$ worst-case bound on memory usage. We sidestep this limitation by using a hybrid approach, calculating frequency domain Green’s functions for the scalar wave equation by driving a high-order, time-domain, finite-difference (FDTD) code to steady state using a periodic source. The frequency-domain response is extracted using the phase sensitive detection (PSD) method recently developed by Nihei and Li (2006). The resulting algorithm has an $O(n^3)$ memory footprint and is amenable to parallelization in the space, shot, or frequency domains. We demonstrate this approach by generating waveform inversion kernels for fully c(x,y,z) models. Our test examples include a realistic VSP experiment using the geometry and velocity models obtained from a site in Western Wyoming, and a deep crustal reflection/refraction profile based on the LARSE II geometry and the SCEC community velocity model. We believe that our 3D solutions to the scalar Helmholtz equation, for models with upwards of 100 million degrees of freedom, are the largest examples documented in the open geophysical literature. Such results suggest that iterative 3D waveform inversion is an achievable goal in the near future.

1 Introduction
The tomographic reconstruction of subsurface properties from remote seismic measurements is a central problem in seismology on exploration, regional, and global scales. Traditional tomographic algorithms invert
the arrival times of different seismic phases to determine subsurface velocity structures (Aki and Lee, 1976; Nolet, 1987). Traveltime tomography, while relatively robust, has a host of limitations due to both to its reliance on the high-frequency asymptotic theory (rays) and its use of a very restricted attribute of the seismic waveform, a factor which limits the resolution of the reconstructed image. Traditional transmission traveltime tomography techniques are capable of recovering velocity models at resolutions near the width of the first Fresnel zone, or about $\sqrt{\lambda L}$ where $\lambda$ is wavelength and $L$ is the approximate S/R offset (Williamson, 1991). Wavefield tomography is an alternative approach that attempts to fit the entire seismic waveform, thereby simultaneously incorporating information encoded in amplitudes, scattered arrivals, and other observed modes. Full wavefield tomography should allow accurate imaging of features on the order of $\lambda$, probably near $\lambda/4$; physical model experiments have convincingly demonstrated that features on this scale can be mapped using such techniques (Pratt, 1999).

For most of the 1990’s, full wavefield tomography was applied primarily to synthetic models (Pratt and Worthington, 1990) and laboratory datasets (Pratt and Goulty, 1991; Pratt, 1999; Dessa and Pascal, 2003; Watanabe et al., 2004) with the exception of several high quality crosswell experiments (Song et al., 1995; Pratt and Shipp, 1999). Recent increases in computational power, advancement in pre-processing techniques, and improvement of inversion strategies have finally enabled non-linear wavefield tomography to be applied to surface 2D reflection/refraction datasets, as shown by Ravaut et al. (2004) within a crustal imaging context, and Shipp and Singh (2002) on long-offset marine streamer data. Although full wavefield inversion methods are still in their infancy, the stage seems set for significant advancements in the coming years. While the wavefield tomography problem can be posed in both the time (Tarantola, 1984; Mora, 1988) and frequency-domains (Pratt and Worthington, 1990), the latter approach has several advantages from both theoretical and computational perspectives. The primary limitation of the methods cited above are extremely large computational requirements, particularly in 3D where no results have yet been shown in the open literature.

In the 2005 ERL consortium report (Ajo-Franklin, 2005) we detailed some of the advantages of approaching the wavefield inverse problem in the frequency-space domain, and developed a set of 2D forward solvers based on directly solving the Helmholtz equation using finite differences. In this report we introduce a technique recently developed by Nihei and Li (2006) for generating solutions to the Helmholtz equation using a hybrid approach where the frequency domain response is recovered by driving a standard time-domain modeling code to steady state. A key component of this approach is the use of a phase-sensitive detection (PSD) algorithm to recover a complex monochromatic wavefield from a steady-state time-domain solution using only a small amount of additional storage. We implement the PSD algorithm on top of an efficient explicit finite-difference solver (FDTD) which uses high order spatial derivatives to minimize the resulting memory footprint. The resulting FDTD + PSD approach allows us to calculate frequency domain Green’s functions for large 3D models on single processor machines. By parallelizing our code in the shot domain, we are capable of computing full kernels for the wavefield inversion problem on modest 3D problems in reasonable time frames.

In the following sections we will (1) briefly develop the wavefield tomography problem to better understand the discrete Born kernel; (2) describe the FDTD + PSD technique as implemented using a high order scalar solver, and (3) demonstrate our algorithm by computing Green’s functions and wavepaths for several realistic 3D models. The demonstration benchmarks include models with upwards of 100 million ($\approx 475^3$) unknowns on a single CPU. These problems are an order of magnitude larger than the current state-of-the-art for iterative Helmholtz solvers. We will conclude our discussion with a concrete outline of the steps required to move frequency-space domain wavefield inversion into a fully 3D world.

## 2 Waveform Tomography in a Nutshell

Since our work on 3D frequency-domain solvers is largely motivated by the inverse problem, we will describe one approach to posing the non-linear waveform tomography problem. Additionally, this exercise helps us to build an intuition for the properties and behavior of the waveform kernel. We construct the inverse problem by first linearizing the wave equation using the Born approximation, and then embedding this linear problem
in an iterative non-linear inversion algorithm. We begin with the forward problem, in our case the scalar Helmholtz equation, which can be written as,

$$\nabla^2 u(\omega) + \omega^2 c(x)^{-2} u(\omega) = -s(x, \omega),$$  \hspace{1cm} (1)

where \( u \) is pressure, \( \omega \) is angular frequency \((2\pi f)\), \( c \) is a spatially varying propagation velocity, and \( s \) is an explicit source term. \( c \) and \( s \) are shown in functional form with respect to a spatial coordinate to emphasize their arbitrary spatial characteristics. From here on we will omit the dependence on angular frequency \((\omega)\). Implicit in equation 1 is the existence of boundary conditions on the domain. For all examples shown in this document, boundaries are designed to absorb outgoing energy to simulate an unbounded domain with the wave field satisfying radiation conditions at infinity.

In the seismic experiment, we assume that sources exist at \( J \) discrete points notated as \( x_j \) and that the pressure is recorded at \( K \) locations written as \( x_k \). The Born linearization maps a perturbation in the model \((\Delta m)\) around a background model \((c_o)\),

$$\Delta m(x') = c(x')^{-2} - c_o(x')^{-2}$$  \hspace{1cm} (2)

to a perturbation in data \((\Delta u)\),

$$\Delta u(x|s) = u(x|s) - u_o(x|s)$$  \hspace{1cm} (3)

where \( u \) is the total field and \( u_o \) is the field in the background model. The form \( u(x|s) \) denotes the field at all points \( x \) generated by a source at location \( s \). The Born approximation can then be written as,

$$u(x|s) = u_o(x|s) + \omega^2 \int_{D} G_o(x|x') \Delta m(x') u_o(x'|s) \, dx'$$  \hspace{1cm} (4)

where \( G_o \) is the Green’s function for the background model and the integral is over the domain \( D \) which encompasses the model perturbation. In this context, the Green’s function is the solution to Helmholtz equation with a spatial delta function as the source term. Isolating the data perturbation \( \Delta u \) and re-grouping the integrand in equation 4 yields,

$$\Delta u(x|s) = \int_D [G_o(x|x') u_o(x'|s)] \Delta m(x') \, dx'.$$  \hspace{1cm} (5)

Equation 5 is now exposed as a Fredholm integral equation where the bracketed terms in the integrand should be viewed as the kernel, \( A \), or

$$\Delta u(x|s) = \int_D A(x, x'|s) \Delta m(x') \, dx'.$$  \hspace{1cm} (6)

Up to this point we have not discretized the model, \( m \), but clearly this step must occur before we seek a numerical solution to the inverse problem. If we choose constant velocity, rectilinear, pixels as our basis function, the discrete form of the Born kernel for a single frequency can be written as,

$$A_{kn}(j) = \omega^2 \sum_{cell_n} G_o(x'|x_j) G_o(x'|x_k) \, dx'.$$  \hspace{1cm} (7)

In equation 7 the integral over cells is included to allow the use of different mesh spacings for the forward and inverse problems. When both meshes are identical, this integral drops out and is replaced by an area or volume coefficient. From this point on, the discretized model and data are represented by \( m \) and \( d \) respectively. \( A \) is of dimensions data by model \((K \times N)\). As can be seen from equation 7, each row of \( A \) is the product of two Green’s functions corresponding to the source and receiver for a given datum. Each row is a surface in model space, the wave theoretic equivalent of a ray in traveltime tomography. Woodward (Woodward, 1992) aptly described these surfaces as Born wavepaths. We express the monochromatic wavepath, \( \mathcal{L} \), for a given source/receiver pair as \( m \)

$$\mathcal{L}[x|x_j, x_k] = \omega^2 G_o(x|x_j) G_o(x|x_k)$$  \hspace{1cm} (8)
Figure 1: A sample wavepath, \( \mathcal{L} \): Panel (A) depicts a source/receiver pair located in a velocity model with a vertical gradient. Panels (B) and (C) show frequency domain Green’s functions for the source \( G(x|x_j) \) and receiver \( G(x|x_k) \) respectively. Panel (D) shows the corresponding wavepath, \( \mathcal{L} \).

Figure 1 schematically depicts the construction of \( \mathcal{L} \); panel (A) shows a velocity model with a vertical gradient while the open circles denote a single source/receiver pair. Panels (B) and (C) show the corresponding Green’s functions for the source and receiver locations, \( G(x|x_j) \) and \( G(x|x_k) \). Panel (D) shows \( \mathcal{L} \), the scaled product of the two Green’s functions. In the forward problem, model perturbations are integrated over \( \mathcal{L} \) while in the inverse problem, operations of \( A^T \) spread data residuals across this surface. An important observation is that \( \mathcal{L} \) is intrinsically non-local, unlike its ray-theoretic counterpart. Model parameters at large distances from the central Fresnel zone are still included in the sensitivity domain of \( \mathcal{L} \). Another important and somewhat obvious property of \( \mathcal{L} \) is that its geometry and structure are tightly coupled to the velocity model.

Figure 2 shows four versions of the classic Marmousi velocity model and the corresponding wavepaths for a single S/R pair. Well-behaved “distorted ellipse” wavepath geometries are visible for only homogeneous and quasi-1D models; beyond these limited models, \( \mathcal{L} \) has a rougher, less organized structure. Unlike some finite-frequency traveltime kernels, \( \mathcal{L} \) does not have zero values along the central (ray) axis.

Calculation of \( \mathbf{A} \) requires generation of at least \( K + J \) Green’s functions. \( \mathbf{A} \) has very different properties than the kernels usually seen in ray-theoretic traveltime tomography algorithms. The fact that \( \mathbf{A} \) is dense makes explicit storage of this matrix impractical for all but the smallest 2D test problems. We generate kernels for just such a toy problem to illustrate the differences between ray and wave based operators. In this case, the model is a homogeneous square of dimensions \( 100 \times 100 \) with the domain interior sampled by a \( 20 \times 20 \) crosswell survey, a highly underdetermined problem with 400 data and 10000 unknowns. Figure 3 shows the ray-theoretic \( \mathbf{A} \) on the left in comparison to a Born kernel on the right.

The column sum of \( \mathbf{A} \) is one measure of data sensitivity to perturbations at different points in the model, usually referred to as a hit-map in ray-based tomography where it records the length sum of rays that intersect a given pixel. Figure 4 again shows a comparison of these sensitivity maps for ray (left) and wave (right) based tomography. A key observation is that the wave-theoretic sensitivity maps displays a relatively complete coverage of the model domain, including zones at the top, bottom, and even behind the source/receiver network. In contrast, the ray based hit map exhibits maximum sensitivity in the central region within the array.

Armed with an understanding and an explicit expression for \( \mathbf{A} \), we can now write the linearized wave equation tomography problem in a regularized least-squares sense as,

\[
\begin{bmatrix}
\mathbf{A} \\
\lambda_x \mathbf{W}_x \\
\lambda_z \mathbf{W}_z 
\end{bmatrix}
\Delta \mathbf{m} =
\begin{bmatrix}
\Delta \mathbf{d} \\
0 \\
0 
\end{bmatrix},
\] (9)

where \( \mathbf{W}_x \) and \( \mathbf{W}_z \) are a pair of anisotropic regularization operators with corresponding scaling parameters.
Figure 2: Sample wavepaths ($\mathcal{L}$) in different versions of the Marmousi model: Column A depicts velocity models while column B shows the corresponding wavepaths for the marked S/R pair.

Figure 3: Images of $A$ based on rays or waves: Panel (A) shows a ray-based traveltime tomography kernel while panel (B) depicts the Born kernel for the same S/R geometry.
Figure 4: Sensitivity maps for both ray and wave-theoretic kernels: Panel (A) depicts a ray-theoretic “hit-map” while panel (B) shows the equivalent sensitivity map for the Born kernel.

\( \lambda_x \) and \( \lambda_z \). The solution to equation 9 can be embedded within an iterative process to solve the non-linear problem, the resulting algorithm being similar to the Gauss-Newton approach. In this context we can see that \( \mathbf{A} \) is in fact the Jacobian of the non-linear problem. A bare bones inversion algorithm might be written compactly as,

1. Initialize our working model \( \mathbf{m}_c \) to our starting model \( \mathbf{m}_o \).

2. While convergence criterion is not met,
   
   (a) Compute synthetic data, \( \mathbf{d}_{est} = \mathbf{A}(\mathbf{m}_c) \).
   
   (b) Compute data residual, \( \Delta \mathbf{d} = \mathbf{d}_{obs} - \mathbf{d}_{est} \).
   
   (c) Compute \( \Delta \mathbf{m} \) by solving equation 9.
   
   (d) Update the model, \( \mathbf{m}_c = \mathbf{m}_c + \alpha \Delta \mathbf{m} \).

3. Done.

As can be seen from the pseudocode, each non-linear iteration involves computing the current data misfit and solving a regularized linear problem to generate a model update vector. The approach we have outlined is just one of several strategies to solving the non-linear problem. Many popular techniques skip linearization and directly perform gradient steps using a sequence of back-propagation operations (?). We will postpone an in-depth examination of the non-linear problem for later papers and concern ourselves with the central computational task: Efficient calculation of Green's functions in the frequency-space domain. Generation of \( \mathbf{A} \) requires at least \( K + J \) Green's functions while even gradient calculations for adjoint methods require \( \min(K,J) + 1 \). In 3D, these calculations can easily exceed the limits of currently available computational resources, particularly in terms of memory requirements. In this case, the way in which algorithms scale to large models is particularly important, and hence we will consider the asymptotic time/space complexity (big “O”) proper for each algorithm examined. Since parallel computation is almost a prerequisite for effective modeling and inversion in three dimensions, the ability to port techniques to multiprocessor architectures, such as the ACES cluster used at ERL, is crucial.
3 Frequency-Domain Modeling and the FDTD + PSD Method

A variety of methods have been proposed to solve the scalar Helmholtz problem, equation 1, each with different computational characteristics. Up to the present, sparse LU decomposition algorithms have been the dominant approach for solving the discrete Helmholtz problem (Stekl and Pratt, 1998; Pratt, 1999; Dessa and Pascal, 2003; Ravaut et al., 2004; Watanabe et al., 2004) due to their guaranteed convergence and high efficiency when the LU factors can be stored in core memory. More recent work on Krylov techniques (Plessix and Mulder, 2003; Erlangga et al., 2004b,a,c; Tang, 2004), suggests that iterative techniques may be a competitive alternative in 3D assuming that suitable preconditioners can be developed. Our previous report (Ajo-Franklin, 2005) outlined the main limitations of both sparse LU and iterative approaches. Sparse LU’s “Achille’s heel” has an asymptotic memory footprint of $O(n^4)$ in 3D where $n$ is the number of samples along an axis of the computational domain. Iterative methods are currently hobbled by the absence of a reliable preconditioner which results in poor (or non) convergence for rough models or high frequencies. Further work on preconditioners, particularly the ‘shifted-Laplace’ approaches (Erlangga et al., 2004c), may yield improvements but reasonable efficiency on 3D problems of any size has yet to be demonstrated.

An alternative to directly solving the Helmholtz equation is to calculate the frequency domain response using a time domain solver. Such strategies can leverage highly efficient explicit FDTD methods to tackle problems outside the current capabilities of pure frequency domain techniques. One approach to extraction of frequency response functions from time domain simulations is to excite the model in question with a broadband pulse, record the pressure field at every point in space at every time step, and then perform a Fourier transformation on the resulting 4D volume along the temporal axis. This 4D volume is almost guaranteed to exceed available core memory in all but the smallest cases; for the SCEC example shown later in this document ($252 \times 514 \times 132$ spatial samples) at least 380 Gb of RAM would be required for the FFT approach. Out-of-core algorithms might succeed if carefully constructed but have non-trivial disk IO requirements since the entire 3D pressure field must be stored at every time step.

We adopt a slightly different approach and calculate frequency domain Green’s functions by driving a time domain simulation to steady-state by excitation with a monochromatic signal. The phase-sensitive detection (PSD) algorithm advocated by Nihei and Li (2006) provides an efficient strategy for computing the complex amplitude of the time harmonic wavefield for a modest amount of additional memory. In a nutshell, the PSD approach drives a time-domain solver to steady state using a sinusoidal source with a known reference phase and amplitude. The frequency-spatial domain (FX) Green’s function is extracted by performing a running cross-correlation between the pilot signal and the measured pressure field at all points in the computational volume. This approach requires only the storage of these additional property volumes for the in-and out-of-phase correlation components. The key to successful implementation of the PSD method is an efficient time-domain solver; in our case we use an explicit high order finite-difference code carefully tuned to minimize memory requirements. We have found that our composite FDTD + PSD algorithm is capable of generating Green’s functions for problems with upwards of $500^3$ unknowns on a single workstation.

3.1 The Phase Sensitive Detection Algorithm

We will now outline the PSD algorithm; greater detail is provided in Nihei and Li (2006), available on request from the second author. We begin with a time-harmonic pilot signal, $u_r(t)$, with reference amplitude $A_r$ and reference phase $\theta_r$. The pressure time-series $u_r(t)$ is used to drive the time-domain solution to steady-state. The in-phase and out-of-phase components of $u_r(t)$ are,

$$u_{r,0^\circ}(t) = A_r \sin(\omega t + \theta_r)$$

$$u_{r,90^\circ}(t) = A_r \cos(\omega t + \theta_r)$$

where $\omega$ is angular frequency. In subsequent expressions we drop the explicit time-dependence $(t)$ for notational convenience. In the PSD algorithm, the pressure time series at every point in space is cross-correlated
with both the original and phase-shifted pilot signals through integral expressions of the form,

\[ X_u = \frac{1}{n \Delta t} \int_0^n \Delta t [u u_{r0}] \, dt \approx \Delta t \sum_{i=0}^n u_i u_{r0} \]  \hspace{1cm} (12)

\[ Y_u = \frac{1}{n \Delta t} \int_0^n \Delta t [u u_{r90}] \, dt \approx \Delta t \sum_{i=0}^n u_i u_{r90} \]  \hspace{1cm} (13)

where the simulation is run for \( n \) time-steps of length \( \Delta t \). \( X_u \) and \( Y_u \) are the cross-correlations of \( u \) with \( u_{r0} \) and \( u_{r90} \) respectively. Since the simulation is discrete in time, both integrals can be written as summations over \( n \) using the midpoint approximation. Since the summations are run over a large number of time-steps, the most memory efficient approach is an “on the fly” recursion of the form,

\[ X_{u}^{i+1} = X_{u}^{i-1} + \Delta t \left[ u^i u_{r0}^i \right] \]  \hspace{1cm} (14)

\[ Y_{u}^{i+1} = Y_{u}^{i-1} + \Delta t \left[ u^i u_{r90}^i \right] \]  \hspace{1cm} (15)

where \( i \) values in the exponent denote time step number. Although time integration runs from 0 to \( n \Delta t \) in equations 12 and 13, the integration process can also be “turned on” at a point late in the simulation process when the wavefield is approaching steady-state, thereby eliminating the computational cost of the cross-correlation at intermediate steps. Once the simulation and consequently \( X_u \) and \( Y_u \) converge, the amplitude \( (A_s) \) and phase \( (\theta_s) \) of the frequency domain Green’s function can be extracted at all points in the modeling domain,

\[ A_s = \frac{2 \sqrt{(X_u^2 + Y_u^2)}}{A_r} \]  \hspace{1cm} (16)

\[ \theta_s = \tan^{-1} \left[ \frac{Y_u}{X_u} \right] + \theta_r \]  \hspace{1cm} (17)

As mentioned previously, the only additional costs of PSD beyond the underlying time-domain solver are the storage of \([X_u, Y_u]\) and the recursive updates required for the cross-correlations shown in equations 14 and 15, quantities which are relatively small in comparison to other algorithmic components.

### 3.2 A 10th Order Scalar Time-Domain Solver

The PSD algorithm can be implemented on top of almost any time-domain modeling code since only the pressure time series is required for the cross-correlation step. While many numerical schemes have been proposed for simulation of seismic wave propagation, explicit time-domain finite-difference (TDFD) methods, originally developed in the early 1970’s (Alterman and Karal, 1968; Alford et al., 1974; Kelly et al., 1976), continue to enjoy enormous popularity due to their efficiency, coding simplicity (Mufti et al., 1996), and easy parallelization using domain decomposition techniques (Villarreal and Scales, 1997).

We solve a scalar time-domain wave equation written in the 2nd order hyperbolic form,

\[ \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} = \nabla^2 u . \]  \hspace{1cm} (18)

where \( c \) is a spatially varying velocity. Density is assumed to be constant within the modeling domain. Numerical solution of equation 18 requires discretization of two derivatives, the time derivative and the spatial Laplacian of the pressure field. In our case we use the explicit finite-difference approximation for both terms. We use a 2nd order central difference approximation to the time derivative of the form,

\[ \frac{\partial^2 u}{\partial t^2} = \frac{u^{n+1} - 2u^n + u^{n-1}}{(\Delta t)^2}, \]  \hspace{1cm} (19)
where $\Delta t$ is a time-step determined by a required stability condition. While adoption of higher order temporal derivatives could increase the maximum stable time-step, they would also necessitate storing extra pressure fields in memory, a classic trade-off between speed and memory requirements. In 3D, we believe that the benefits of a smaller memory footprint outweigh the value of larger time-steps since our goal is to be able to run large problems on single CPUs in a shot-parallel fashion. Substitution of the temporal difference operator (19) into the wave equation (18) and regrouping terms around $u^{n+1}$ yields an explicit time-stepping formula

$$u^{n+1}_{i,j,k} = \Delta t^2 c^2_{i,j,k} \nabla^2 u^n_{i,j,k} + 2u^n_{i,j,k} - u^{n-1}_{i,j,k} + s_{i,j,k},$$

where subscripts indicate $(i, j, k)$ location within the computational mesh. We also add a source term, $s$, to provide for excitation of the system. The discretization of the laplacian operator in equation 20 is the next step in building our FDTD scheme. Following previous authors (Dablain, 1986; Wu et al., 1996), we opt for the use of high order spatial derivatives which allows for coarse sampling of the wavefield without significantly increasing numerical dispersion. We typically used a centered 10th order difference operator for the laplacian but our current implementation allows the user to choose any even order operator between 2 and 10 (2/4/6/8/10). We refer the reader to the external literature for the spatial FD coefficients (Dablain, 1986; Wu et al., 1996), particularly the useful table provided in Fornberg (1996). In most cases we find that sampling around 3 grid points per wavelength is sufficient to avoid numerical dispersion. The cost associated with higher order approximations to the laplacian is an increased number of floating point and memory access operations for every grid point update. As should be clear from equation 20 any FDTD scheme of this class is of complexity $O(n^3n_t)$ in time and $O(n^3)$ in space since every point in the volume is updated once at each time step.

In addition to an update scheme, we must also incorporate an ABC to eliminate reflections from the edges of the computational domain. We use the sponge boundary condition of Cerjan et al. (1985) which, while not the most efficient ABC, is simple to implement and robust in practice. We use the relationship developed by Lines et al. (1999) for calculation of stability conditions for the higher-order difference operators. Assuming a 2nd order time-stepping scheme, the maximum stable $\Delta t$ value can be expressed as,

$$\Delta t \leq \frac{2h_{min}}{c_{max} \sqrt{a_2}}$$

where $c_{max}$ is the maximum velocity, $h_{min}$ is the minimum grid spacing, and $a_2$ is the sum of the absolute values of the weights for $\nabla^2 u$. $a_2$ is always positive and increases as a function of dimension assuming the same order spatial operator.

Our adoption of very high-order (eg. 10th) FD formulation was motivated by the dramatic possible reductions in memory footprint, a key trait for effective 3D modeling. The 2nd order spatial operators, used in our earlier FDTD implementations, typically requires 12 samples per minimum wavelength in contrast to 3 needed for a 10th order operator. For a problem domain of 50 $\lambda$ cubed, a 10th order scheme has only 1/64th the memory footprint, a compelling argument when targeting 32 bit architectures. 10th order schemes exhibit dispersion properties only slightly worse than pseudospectral methods (Kosloff and Baysal, 1982; Fornberg, 1996) which calculate spatial derivatives in the fourier domain. At the same time, high-order FD schemes exhibit several advantages over the pseudospectral approach. In contrast with Fourier operations, FD spatial-difference operators are local, making parallelization using domain decomposition techniques straightforward (Villarreal and Scales, 1997). The same locality improves cache performance on a hardware level. From a software engineering perspective, FD methods are simpler to code, particularly in the parallel case where distributed FFT algorithms can add an extra layer of complexity.

### 3.3 Problem Complexity

As mentioned previously, the FDTD+PSD solution has a computational complexity of $O(n_t n^3)$ and a memory requirement of $O(n^3)$ where $n_t$ is the number of simulated time-steps and $n$ is the number of sampled points along a single spatial axis assuming a mesh where $n_x = n_y = n_z$. In most cases, $n_t$ is proportional to $n$.
making the complexity of the algorithm approximately \(O(n^4)\). Equivalently, the space/time complexity of the algorithm is linear in the number of field parameters.

The memory requirements for our 2nd/10th scheme are quite modest. The 2nd order time integration process requires only three pressure fields \((u_{n-1}, u_n, u_{n+1})\), each consisting of \(n^3\) floating point values. Two more fields are required to accumulate the in- and out-of-phase components of the cross-correlation process, and a final field is used to store the velocity model. In entirety, the solver has a memory requirement of \(6n^3\).

The outer loop of the FDTD + PSD algorithm has 4 major components, the wavefield update calculation, application of the absorbing boundary conditions, the exchange of pressure fields, and the cross-correlation step (PSD). We will analyze each of these steps in terms of the number of floating point and memory operations. Note that these analyses are based on the source code of a clean, well-factored implementation; the actual number of operations performed in hardware is dependent on several secondary factors including the choice of programming language and the compiler optimization level.

Most of the algorithm execution is spent in the wavefield update calculation. As is clear from equation 20, the update can be partitioned into the evaluation of the 10th order spatial laplacian and the 2nd order time integration. The 10th order FD stencil extends 5 points in each of the cardinal directions. If factored appropriately, the evaluation of each directional derivative requires 16 floating point operations (10 adds and 6 multiplies) and 11 access operations into the 3D array. An additional 10 integer adds are required for index calculation. The time integration step is less expensive and requires only 2 floating point multiplies, 4 adds, 3 access operations, and 1 assignment. In total, the update at each grid point requires at least 54 floating point, 30 integer, and 36 memory access operations.

Exchanging the pressure fields in the time update process requires 2 access operations and 2 assignments for each grid location. Application of the absorbing boundary condition consists of 6 triple for-loops over the pad zones on the domain boundary. At each point, the wavefield is exponentially damped, a process which requires 3 access operations, 2 assignments, and 2 multiplies. The exponential is pre-evaluated and stored in a table to avoid a function call in the inner loop. Since the ABC is only active in the pad region, only a small fraction of the total number of grid locations are modified. The running cross-correlation step is a recursive summation for both an in- and out-of-phase components. This step necessitates 4 access operations, 2 assignments, 4 multiplies, and 2 adds. As can be seen from this break-down, the largest computational component of the algorithm is the wavefield update step, and within this update, the evaluation of the spatial laplacian.

### 3.4 Parallelization

The goal of our solver design was to minimize memory requirements to allow the solution of reasonably large 3D problems on a single cluster node. Our current implementation is parallelized over shot location, an approach which minimizes communication costs on high-latency parallel architectures. We typically use a small subset of the nodes available on the ACESgrid cluster at MIT. A series of Python scripts are used to distribute shot specifications to the chosen computational nodes, invoke the serial modeling code, and collect the resulting frequency-domain Green’s functions. The entire system is run within the PBS queuing framework. Since the serial version of the code is used, this approach is limited by the memory present on individual compute nodes; for larger 3D problems domain-decomposition methods will be necessary but should be straightforward to implement within the FDTD+PSD framework (Villarreal and Scales, 1997). Despite this limitation, with a memory footprint of only \(6n^3\) floating point values, problems as large as \(450^3\) can be handled on 32 bit nodes with 4 Gb of RAM.

### 4 Green’s Functions and Kernel Calculations: 3D Examples

To demonstrate the efficacy of the FDTD + PSD technique for calculating Green’s functions and wavepaths for the Born kernel, we developed several realistic 3D test problems in addition to a series of more routine benchmarks designed to examine scaling characteristics.
4.1 The LARSE II Survey and the SCEC Model

Active source crustal imaging experiments, particularly long offset reflection/refraction surveys, might also benefit from wavefield tomography methods. While many such experiments have quasi-2D geometries, they are often acquired on crooked lines and traverse zones with large out-of-plane property contrasts. As a demonstration of our algorithm in this context, we generated waveform kernels for portions of the LARSE II (Low Angeles Regional Seismic Experiment) survey. The subsurface property values used for calculating the required Green’s functions were extracted from the fourth version of the Southern California Earthquake Center’s (SCEC) community velocity model (Magistrale et al., 2000; Kohler et al., 2003). The active source component of the LARSE II experiment (Fuis et al., 2001; Godfrey et al., 2002), jointly managed by the US Geological Survey and SCEC, was undertaken to improve understanding of the fault networks and sedimentary basins beneath and adjoining the heavily populated LA Basin. Line 2 of the profile, acquired in 1999, runs north/south traversing the Santa Monica Mountains and San Fernando Valley before crossing the Transverse Ranges and the San Andreas Fault Zone, terminating in the Mojave. We use the line 2 source/receiver geometry as the basis for our modeling exercise.

A narrow north/south swath of the SCEC model was chosen to cover all of the onshore receiver locations which were part of line 2 in the LARSE II experiment. The velocity model was discretized as a 200 \times 462 \times 80 mesh with a spatial sampling of 300 m; this corresponds to a 60 \times 140 km swath extending to 24 km depth. The only modification applied to the SCEC model was boosting all velocity values to at least 1500 m/s, thereby eliminating very near surface structure. Considering the depth of the imaging targets (\geq 1 km), this modification seems justified and simplifies the modeling process by allowing coarser spatial sampling.

3D Green’s functions and wavepaths were calculated at 1.65 Hz for a subset of source/receiver pairs using the scalar FDTD+PSD algorithm. A 26 point absorbing pad was used on all boundaries; the full problem size was 252 \times 514 \times 132 corresponding to 17 million degrees of freedom. At 1.65 Hz, the 300 m grid spacing corresponds to roughly 3 samples per minimum wavelength, which is sufficient to avoid numerical dispersion when using a 10th order solver. The maximum velocity present in the model was 7500 m/s, a value which forced use of very small time steps (\approx 0.016s) to guarantee stability in the forward calculation. As a result, each simulation required 5600 time steps (90 s) to reach steady state. Computation of a single Green’s function took 2.2 hours of wall-clock time on a Pentium 4 workstation with minimal external load. Using 16 nodes of the ACESgrid cluster, we were able to calculate the full set of Green’s functions for all 47 onshore receivers and 63 shot locations in \approx 18 hours. Once this table of Green’s functions is stored to disk, each row of the discrete kernel can be sequentially generated at relatively low computational cost.

Figure 5 depicts a 3D representation of both the SCEC v.4 velocity model (A) and \mathcal{L}, the waveform kernel (B) for one particular long offset pair. Within panel (A) the blue (low velocity) regions are basins which transition to higher velocity crystalline basement at depth. The wavepath depicted in panel (B) shows highly variable structure, particularly in the heterogeneous regions above 10 km depth.

Figure 6 shows constant depth (panel A) and constant Y (panel B) slices of the wavepath shown in 5, with black lines corresponding to iso-contours of the velocity model. As can be seen in the depth slice at 0.6 km, segments of the near-surface structure are poorly sampled with more energy refracting beneath the low velocity basins. The depth slice at 3.0 km shows distinct partitioning of sensitivity across sharp velocity boundaries.

4.2 A VSP Survey From Western Wyoming

In a second test, we explored the monochromatic scattering signature of an array of synthetic crack-like structures embedded in a layered medium. The model was designed to replicate the near-offset component of a 4D VSP survey acquired in Wyoming. The 1D velocity structure was obtained from an LSQ inversion of check-shot traveltime data. The model was discretized into a 220 \times 160 \times 730 sample model with a grid spacing of 5 m. This particular choice yields a 3D problem with 25.7 million degrees of freedom.

The first panel of figure 7 depicts the velocity model used (first panel) including the vertical scatterers at approximately 2500 m depth. The red circles on the surface correspond to the 8 shot locations modeled
Figure 5: $\mathcal{L}$ for the SCEC/LARSE model: Panel (A) depicts color maps covering five profiles extracted from the SCEC 4.0 velocity model. Panel (B) shows the corresponding slices into the 1.65 Hz waveform kernel for one source/receiver pair. Colors in (B) corresponds to the real component of $\mathcal{L}$ normalized over $[-1,1]$ and clipped at 0.8.
Figure 6: Slices of $\mathcal{L}$ for the SCEC/LARSE model: Panel (A) depicts three depth slices of $\mathcal{L}$ at 0.6, 3, and 9 km. Panel (B) shows three $[X,Z]$ slices parallel to the LARSE line at Y locations of 3, 24, and 48 km. Colors in all panels correspond to the real component of $\mathcal{L}$ normalized over $[-1,1]$ and clipped at 0.8. Black lines in all plots correspond to iso-contours in the underlying SCEC velocity model.
while the black profile shows the 130 receiver depths recorded in the experiment. Panel 2 shows the scattered
wavefield generated by the “fractures” at 100 Hz for the shot closest to the well head.

The last three panels of figure 7 show constant Y slices of the velocity model, source Green’s function,
and the scattered wavefield respectively. As expected, the fracture set preferentially forward scatters energy
with some diffractions visible from the fracture tips. By using a 3D model, we were also able to incorporate
the full 3D geometry of the experiment including out-of-plane well deviation.

4.3 The SEG 3D Salt Model

Our last test case was a resampled version of the SEG/EAGE salt model. The structure of the salt model was
defined in 1992 by a team of academic and industrial geoscientists with broad experience in salt tectonics,
seismic modeling, and imaging algorithms. The resulting velocity model includes features typical of the US
Gulf Coast’s salt provinces, a zone of continuing exploration interest.

We compute 3D wavefield kernels for a central swath of the 3D model with dimensions of 10 × 4 × 4
km sampled at 20 m spacing yielding a 500 × 200 × 200 mesh. Inclusion of the absorbing pad on all
boundaries increases the size of the problem to 580 × 280 × 280, approximately 45.5 million degrees of
freedom. Frequency domain Green’s functions were calculated at 10, 15, and 20 Hz.

Figure 8 shows a 15 Hz wavepath for a source/receiver pair near the surface. Brown lines indicate velocity
iso-contours. This wavepath is particularly difficult to interpret and exhibits minimal structure due to the
strong scattering off of the salt dome.

4.4 Performance Summary

In addition to these three realistic examples, a series of simple 3D models were run with the goal of evaluating
scaling and performance on three architectures. Figure 9 shows the wall clock time required for 100 fd steps
on a notebook (1.7 GHz Pentium M), a fast desktop workstation (2.4 GHz Pentium 4), and a slow 64 bit
server (1.2 Ghz Itanium 2) for a large range of problem sizes. Each 3D model consisted only of a 2-layer
slab. Also included are the sizes for our 3 more realistic examples and the largest iterative solution we have
been able to find in the open literature, a 120³ problem discussed in Erlangga et al. (2004a). When viewing
plot 9 one should note that both axes are log scaled; the 475³ benchmark problem we calculate is almost
2 orders of magnitude greater in size than the largest iterative solution. As can be seen, the FDTD+PSD
solver scales linearly with problem size on all 3 architectures with performance directly correlated with clock
speed. Maximum problem size was determined by available RAM, 512 Mb, 2 Gb, and 6 Gb for the notebook,
workstation, and server respectively.

5 Future Directions

In conclusion, we successfully implemented a 3D solver suitable for calculating frequency-domain Green’s
functions and wavepath kernels for problems with upwards of 100 million unknowns. The solver couples a
high order explicit FDTD code to the PSD algorithm to allow for recovery of monochromatic wavefields. We
are currently developing a wavefield tomography system based on this forward solver. A pertinent question is
whether or not such a system will be able to solve reasonable 3D problems with the computational resources
currently available to us.

Consider a small/mid-sized VSP problem, similar to a dataset we are currently processing. Assume a 3D
model of approximately 10 million unknowns or about 215³ samples representing a problem domain of 60
wavelengths. For a survey with 200 shot and 200 receiver locations, 40,000 s/r pairs must be considered. Each
Green’s function requires about 1 hour to calculate on a typical compute node. Since we need 400 Green’s
functions for calculation of \( A \), a 64 node subset of the cluster would require about 6.5 hrs to generate the
required wavefields. The products \( Am \) and \( A^T d \) are computed on a row-wise basis by loading in wavefields
from disk, a step which is I/O rather than CPU limited. We estimate that the linearized 3D problem will
Figure 7: Synthetic results from a 3D VSP experiment: Panels (1) and (2) depict 3D representations of the velocity model and the scattered wavefield due to the fracture-like features located at 2500 m depth. Panels (2), (3), and (4) show a constant Y slice of the velocity model, source Green’s function, and scattered wavefields respectively.
Figure 8: A 15 Hz wavepath in the SEG/EAGE salt model: Colors indicate wavepath amplitudes while brown lines depict iso-contours of the background velocity model.
Figure 9: FDTD + PSD solver performance on 3 architectures: Solid blue lines indicate the size of the 3 more realistic examples discussed previously.
require about 10 hours in this configuration. Assuming 20 non-linear steps are needed, the entire 3D non-
linear problem could be solved in 8 days for each frequency. This figure will decreases linearly with increasing
available compute nodes, but should be treated with skepticism considering our limited experience with the
full inverse problem. We believe that such figures demonstrate that we are on the verge of being able to
solve mid-sized 3D VSP problems on relatively modest parallel architectures.

A significant part of our future work will involve explicit consideration of the inverse problem using the
3D forward solver we have outlined. Our first investigation will explore the non-linear problem in more
detail with a particular focus on deciding between traditional adjoint methods and more complicated Gauss-
Newton schemes which rely on access to $A$. We are also interested in examining regularization and weighting
methods for the non-linear problem, possibly including data-space $L^p$ norms and IRLS compactness metrics
in model-space. A final topic of interest is the use of wavefield tomography on time-lapse data, i.e. inverting
a differential wavefield for changes in model properties.

6 Acknowledgments

We would like to thank the Founding Members Consortium of the Earth Resources Laboratory for its
generous support and guidance. Partial support came from the Shell Gamechanger program.

References

Ajo-Franklin, J. (2005). Frequency-domain modelling techniques for the scalar wave equation : an intro-
duction. Earth Resources Laboratory Annual Report.


helmholtz problems. Technical report, Delft University of Technology.


solver for the two-way wave equation based on a complex shifted-laplace operator. In SEG International


