

User Guide of Seismic Simulation, Survey, and Imaging (SSSI)

Entao Liu ^{*}and Lingchen Zhu [†]

January 13, 2015

1 Requirements and License

SSSI is a Matlab based package. Besides Matlab (2012a or later version recommend), you also need a C++ compiler to generate the mex function. SSSI is free for use in academic research. is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY. If you find any glitches within the package or you want to share your extensions to SSSI, please contact the authors. We will keep the packages updated and thank you for your contribution. By far, some source files in SSSI are already inherited from other open source package, please read the comments in codes for authorship and contact information.

2 Introduction

The SSSI is designed to provide a package for numerical exploration geophysics. The purpose of package is more on the educational side than the performance of the computation. Thus we build the package based on Matlab for the readability of the codes, easy of data visualization, etc. The major functions has been of the package are the following:

- Acoustic wave simulation for 2D/3D
- Elastic wave simulation for 2D

^{*}Center for Energy & Geo Processing (CeGP), School of ECE, Georgia Tech, e-mail: liuentao@gmail.com

[†]Center for Signal and Information Processing (CSIP), School of ECE, Georgia Tech e-mail: lczechu@gatech.edu

- Kirchhoff's migration
- Reverse Time Migration (RTM)
- Full Waveform Inversion (FWI)

The purpose of this manual is not to provide a comprehensive review of numerical simulation of wave equation and seismic imaging, but rather a guide for you to quickly learn how to use SSSI and what you can do with it. We will only show and explain necessary equations to keep it concise. Interested users are suggested to read the references therein for details.

3 Numerical Simulation of Acoustic Wave

3.1 Acoustics wave equation

The seismic method is one of the basic tools in exploration geophysics. People fire off man-made vibration sources (e.g. dynamite, vibroseis trucks, and air gun) then the wave propagates through the subsurface media. The reflection and refraction will be recorded by geophones (on shore) and hydrophones (off shore). In order to simulate this process with computers, it is crucial to simulate the wave propagation in the media accurately and efficiently. The real earth is an elastic media, such that the seismic waves contain both P-waves and S-waves. For simplicity, we some times only consider the P-wave field, which is mathematically described by an acoustic wave equation. It is verified by borehole data that the media density variations are not the main source of reflected waves. Therefore, it is reasonable to assume constant density of the media. Then the acoustic wave equation is

$$\frac{1}{c^2(\mathbf{x})} \frac{\partial^2 p(\mathbf{x}, t)}{\partial t^2} - \Delta p(\mathbf{x}, t) = f(\mathbf{x}, t), \quad (3.1)$$

where $p(\mathbf{x})$ is the field of pressure variation and $\mathbf{x} = (z, x)$ or $\mathbf{x} = (z, x, y)$ is the coordinates in the Cartesian coordinate system for the 2D and 3D case, respectively. Following the convention in geophysics, the z direction is pointing downwards. The $c(\mathbf{x})$ is the velocity of P-wave at location \mathbf{x} . The $f_s(\mathbf{x}, t)$ is the source term. Moreover, Δ in (3.1) is the Laplace operator defined as

$$\Delta := \begin{cases} \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} & \text{for 2D} \\ \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} & \text{for 3D} \end{cases} \quad (3.2)$$

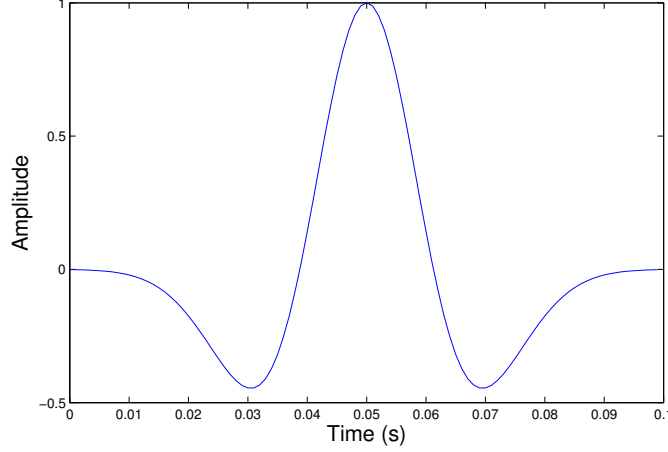


Figure 1: Ricker wavelet with peak frequency $f = 20\text{Hz}$

Ricker wavelet is a widely used seismic wavelet whose amplitude A with the peak frequency f at time t is computed as:

$$A = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 t^2}. \quad (3.3)$$

The `src/ricker.m` provides a Ricker wavelet generator which requires the peak frequency f , number of time samples n , sampling time dt , and peak location t_0 as inputs. The Figure 1 illustrates the waveform of a Ricker wavelet with peak frequency equals 20 Hz. In SSSI you can choose from several different wavelets for the source term (Ricker wavelet, Fuchs-Mueller wavelet, sine cube wavelet, etc), which can be generated by `src/waveletGenerator.m` by D. Koehn.

3.2 Finite Difference Method on Standard Grid

There are two popular methods of solving the wave equation numerically. One is the Finite Difference Method (FDM), the other is the Finite Elements Method (FEM) method. Each method has its pros and cons. Concretely, the FDM which is adopted by the SSSI is simple to implement. For most of the simulations, it has sufficient accuracy. The FEM usually provides more accurate results, which is based on adaptive meshing (multiscale) of the simulated region. However, its implementation is less straightforward, especially when adding boundary conditions.

In order to solve the wave equation with FDM, functions are represented by their values at certain discrete grid points and derivatives are approximated through difference in these values. For instance, in a 2D region we use the equi-distributed grid points $(z_i)_{0 \leq i \leq I-1}$, $(x_j)_{0 \leq j \leq J-1}$, and $(t_n)_{0 \leq n \leq N-1}$ given by $z_i = i\Delta z$, $x_j = j\Delta x$ and $t_n = n\Delta t$. Instead of solving the wave equation in a continuous domain (both in space and time) analytically, the FDM looks for a numerical solution on these grid points.

A crucial step of FDM for wave equations is to find good approximations of the derivatives in (3.1) with function values on the grid points. Let us begin with the definition of the derivative,

$$\begin{aligned} \frac{\partial p(z, x, t)}{\partial z} &= \lim_{\Delta z \rightarrow 0} \frac{p(z + \Delta z, x, t) - p(z, x, t)}{\Delta z} \\ &\approx \frac{p(z + \Delta z, x, t) - p(z, x, t)}{\Delta z}. \end{aligned} \quad (3.4)$$

assuming the finite grid size Δz is small enough the difference gives accurate estimation of the derivative. Apply this approximation twice. We can derive an estimation of the 2nd derivative in (3.1)

$$\begin{aligned} \frac{\partial^2 p(z, x, t)}{\partial z^2} &= \frac{\partial}{\partial z} \frac{\partial p(z, x, t)}{\partial z} \approx \frac{\frac{\partial p(z, x, t)}{\partial z} - \frac{\partial p(z - \Delta z, x, t)}{\partial z}}{\Delta z} \\ &\approx \frac{p(z + \Delta z, x, t) - 2p(z, x, t) + p(z - \Delta z, x, t)}{\Delta z^2}. \end{aligned} \quad (3.5)$$

For simplicity of the notations, we denote $p_{i,j}^{(n)} := p(i\Delta z, j\Delta x, n\Delta t)$, $f_{i,j}^{(n)} := f(i\Delta z, j\Delta x, n\Delta t)$, $v_{i,j} := v(i\Delta z, j\Delta x)$. Therefore, a finite difference expression of 2-D acoustic wave equation can be written as

$$\begin{aligned} \frac{1}{v_{i,j}^2} \frac{p_{i,j}^{(n+1)} - 2p_{i,j}^{(n)} + p_{i,j}^{(n-1)}}{\Delta t^2} - f_{i,j}^{(n)} \\ = \frac{p_{i+1,j}^{(n)} - 2p_{i,j}^{(n)} + p_{i-1,j}^{(n)}}{\Delta z^2} + \frac{p_{i,j+1}^{(n)} - 2p_{i,j}^{(n)} + p_{i,j-1}^{(n)}}{\Delta x^2}. \end{aligned} \quad (3.6)$$

Simple algebraic manipulations lead to a recursive expression of the wave equation,

$$\begin{aligned} p_{i,j}^{(n+1)} &= \frac{v_{i,j}^2 \Delta t^2}{\Delta z^2} \left(p_{i+1,j}^{(n)} - 2p_{i,j}^{(n)} + p_{i-1,j}^{(n)} \right) \\ &+ \frac{v_{i,j}^2 \Delta t^2}{\Delta x^2} \left(p_{i,j+1}^{(n)} - 2p_{i,j}^{(n)} + p_{i,j-1}^{(n)} \right) \\ &+ 2p_{i,j}^{(n)} - p_{i,j}^{(n-1)} + v_{i,j}^2 \Delta t^2 f_{i,j}^{(n)}. \end{aligned} \quad (3.7)$$

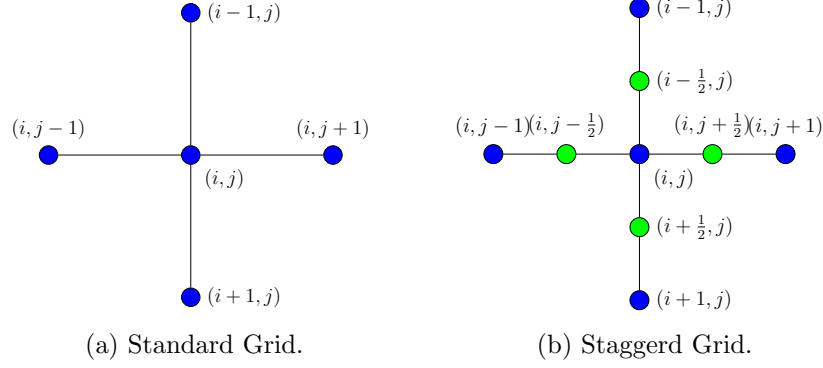


Figure 2: Discretization grids.

In (3.7), all values of p are computed on integer grid points as we illustrated in Figure 2a. The FDM on standard grid is an easy-to-understand numerical scheme, which serves as an excellent explanatory example. However, it is not adopted in SSSI because of its drawbacks in accuracy.

3.3 Finite Difference Method on Staggered Grid

With a sophisticated design, we can actually obtain higher order of approximation of the derivatives if we have access to the value of p on the half grid points, as denoted by the green dots in Figure 2b.

By Taylor expansion of a function $p(u)$ on the half grid points, where $u = z, x$; $k = 0, 1, 2, \dots$

$$p\left(u + \frac{2k+1}{2}\Delta u\right) = p(u) + \sum_{n=1}^{\infty} \frac{1}{n!} \frac{\partial^n p(u)}{\partial u^n} \left(\frac{2k+1}{2}\Delta u\right)^n \quad (3.8)$$

$$p\left(u - \frac{2k+1}{2}\Delta u\right) = p(u) + \sum_{n=1}^{\infty} \frac{(-1)^n}{n!} \frac{\partial^n p(u)}{\partial u^n} \left(\frac{2k+1}{2}\Delta u\right)^n \quad (3.9)$$

Take the difference between (3.8) and (3.9). It cancels all the terms when n is even. So it implies for $k = 0, 1, 2, \dots$

$$\begin{aligned} & \frac{p\left(u + \frac{2k+1}{2}\Delta u\right) - p\left(u - \frac{2k+1}{2}\Delta u\right)}{(2k+1)\Delta u} \\ &= \frac{\partial p(u)}{\partial u} + \sum_{n=1}^{\infty} \frac{1}{(2n+1)!} \frac{\partial^{(2n+1)} p(u)}{\partial u^{(2n+1)}} \left(\frac{2k+1}{2}\Delta u\right)^{2n}. \end{aligned} \quad (3.10)$$

We can approximate $\frac{\partial p(u)}{\partial u}$ as a weighted sum of finite difference in (3.10)

$$\begin{aligned}
\frac{\partial p(u)}{\partial u} &= \sum_{k=0}^{N-1} a_k \frac{p(u + \frac{2k+1}{2}\Delta u) - p(u - \frac{2k+1}{2}\Delta u)}{(2k+1)\Delta u} \\
&= \sum_{k=0}^{N-1} a_k \left[\frac{\partial p(u)}{\partial u} + \frac{\Delta u^2}{3! \cdot 2^2} (2k+1)^2 \frac{\partial^3 p(u)}{\partial u^3} \right. \\
&\quad + \frac{\Delta u^4}{5! \cdot 2^4} (2k+1)^4 \frac{\partial^5 p(u)}{\partial u^5} + \dots \\
&\quad \left. + \frac{\Delta u^{2N-2}}{(2N-1)! \cdot 2^{2N-2}} (2k+1)^{2N-2} \frac{\partial^{2N-1} p(u)}{\partial u^{2N-1}} + o(\Delta u^{2N}) \right].
\end{aligned} \tag{3.11}$$

If the weighted $\{a_k\}_{k=0}^{N-1}$ are chosen carefully, we can eliminate all the term on the right hand side of (3.11) and make the coefficient of $\frac{\partial p(u)}{\partial u} = 1$. These N weights generate a approximation of the derivative of order $2N$. Additionally, the derivative $\frac{\partial p(u)}{\partial u}$ and function itself $p(u)$ live on different colors of dots, i.e. $\frac{\partial p(u)}{\partial u}$ on half grid points, and $p(u)$ on integer grid points.

Solving $\{a_k\}_{k=0}^{N-1}$ by the following linear equations

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1^2 & 3^2 & \dots & (2N-1)^2 \\ 1^4 & 3^4 & \dots & (2N-1)^4 \\ \vdots & \vdots & \ddots & \vdots \\ 1^{2N-2} & 3^{2N-2} & \dots & (2N-1)^{2N-2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$N = 1 : a_0 = 1$$

$$N = 2 : a_0 = 9/8, a_1 = -1/24$$

$$N = 3 : a_0 = 75/64, a_1 = -25/384, a_2 = 3/640$$

$$\vdots$$

The source code which generates the appropriate weights can be found at `src/dCoef.m`. Moreover, the finite difference operator is fulfilled by `src/diffOperator.m`.

3.4 Absorbing Boundary Conditions (ABC)

For the sake of efficiency and storage, we only simulate wave propagation in a truncated region which contains the seismic source and array of sensors. In a real seismic survey, the seismic wave generated by the source should have

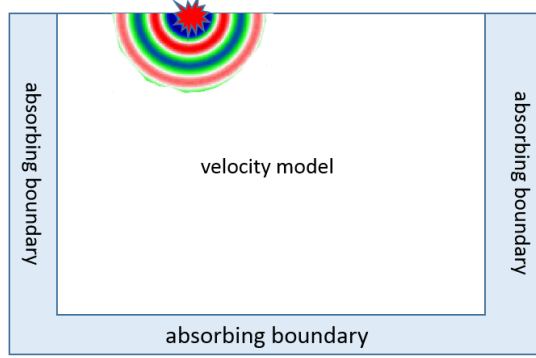


Figure 3: Absorbing boundaries of the 2D simulation region

no reflection on the simulation boundaries to mimic the unbounded media. If we do not take care with particular techniques and run the recursive formula, say (3.7), the numerical simulation will generate strong reflections on the boundaries which do not exist in the real earth. In order to attenuate the reflections, a common method is to impose the absorbing boundary conditions (ABC) [7, 5]. Figure 3 illustrates the 2D scenario, when we pad the original velocity model's left, right, bottom boundaries with absorbing boundaries of certain width (fulfilled by `src/extBoundary.m`). ABC will attenuated the wave amplitude in those absorbing boundaries and keep it unaltered out side of the absorbing boundaries. Throughout the SSSI, except specified explicitly, we consider the top of the simulated region to be the surface of the earth, which is a free surface. So no ABC applied to it.

Numerically, a simple but efficient method for the ABC is called sponge ABC [3]. Suggested by its name, the reflections are exponentially attenuated in the extended artificial boundary area by multiplying a factor $d(u) < 1$.

$$d(u) = e^{-\alpha^2 \text{dist}(u)^2}, \text{ where } u = x, z \quad (3.12)$$

where $\text{dist}(u)$ is the distance from u to the original boundary in the u direction.

The source code of which implements the acoustic wave propagation with sponge ABC is `src/fwdTimeSpongeFor2dAw.m`. Through out all names of the source files in SSSI, we follow this rule: **fwd** means forward (**rvs** means reverse); **Time** means the equation is solved in the time domain (**freq** means in frequency domain); **For2d** means 2 dimensional (**For3d** means 3 dimensional); and **Aw** is short for acoustic wave (**Ew** is for elastic wave). Although the waves are attenuated by ABC, however the reflection can not

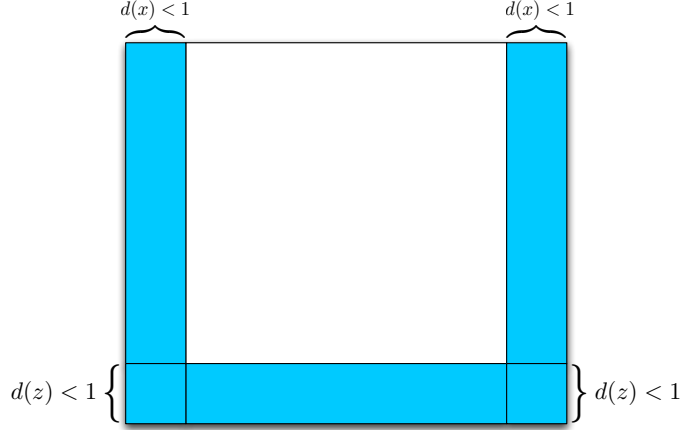


Figure 4: The damping profile $d(u)$

be completely eliminated. This is the reason in SSSI we utilize the following method called Perfectly Matched Layer as well.

3.4.1 Perfectly Matched Layer (PML)

Perfectly Matched Layer (PML) method which was originally formulated for use with Electromagnetic equations outperforms other existing methods. It is proven to be efficient for wave equations (both acoustic and elastic) as well [9]. The PML have a zero reflection coefficient for all angles of incidence and all frequencies before discretization. Moreover, a PML interface between a physical medium and the extended artificial boundary completely absorbs incident waves from the physical medium regardless of its incidence angle and frequency. By defining a damping profile $d(u)$ ($u = z, x, y$) function (see `src/dampPml.m`) such that $d(u) = 0$ inside the physical medium and $d(u) > 0$ in the PML region, a new complex coordinate \tilde{u} is introduced as

$$\tilde{u}(u) = u + \frac{1}{j\omega} \int_0^u d(s) ds. \quad (3.13)$$

Or equivalently,

$$\frac{\partial}{\partial \tilde{u}} = \frac{j\omega}{j\omega + d(u)} \frac{\partial}{\partial u} = s_u(j\omega) \frac{\partial}{\partial u}. \quad (3.14)$$

In the homogeneous media, the acoustic wave equation has the following solution $A \exp(-j(\mathbf{k} \cdot \mathbf{x} - \omega t))$, where A represents the amplitude and polarization of the plane wave. The $\mathbf{k} = k_x \hat{\mathbf{x}} + k_y \hat{\mathbf{y}} + k_z \hat{\mathbf{z}}$ denotes the wave

vector, which indicates the direction of wave propagation of the plane wave. $\mathbf{x} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$ is the position vector. After the replacement in (3.13), we can derive another solution of the acoustic equation as follows,

$$A \exp(-j(k_x \tilde{x} + k_y y + k_z z - \omega t)) = A \exp(-j(\mathbf{k} \cdot \mathbf{x} - \omega t)) \exp(-kx/\omega \int_0^x d(s) ds). \quad (3.15)$$

In the original simulated region the the new solution is equivalent to the old one, since $d_x = 0$. Additionally, in the $\hat{\mathbf{n}} = \hat{\mathbf{x}}$ direction, the wave amplitude decay with a coefficient $\exp(-kx/\omega \int_0^x d(s) ds)$ that is inversely proportional to the angular frequency ω of the plane wave.

In SSSI we use a non-split method, the Convolutional PML (CPML) [11, 14, 9], which is more reasonable to work with the acoustic (pressure) source. In time domain,

$$\frac{\partial}{\partial \tilde{u}} = s_u(t) * \frac{\partial}{\partial u} = \frac{\partial}{\partial u} - \left(d(u) H(t) e^{-d(u)t} \right) * \frac{\partial}{\partial u} \quad (3.16)$$

The convolution can be performed as follows,

$$\begin{cases} \frac{\partial^2 p}{\partial t^2} = v^2(P_z + P_x) \\ P_z = \frac{\partial A_z}{\partial z} + \Psi_z \\ P_x = \frac{\partial A_x}{\partial x} + \Psi_x \\ A_z = \frac{\partial p}{\partial z} + \Phi_z \\ A_x = \frac{\partial p}{\partial x} + \Phi_x \end{cases} \quad (3.17)$$

and

$$\begin{cases} \Psi_z^{(n)} = b_z \Psi_z^{(n-1)} + (b_z - 1) \partial_z^{(n-1)} A_z \\ \Psi_x^{(n)} = b_x \Psi_x^{(n-1)} + (b_x - 1) \partial_x^{(n-1)} A_x \\ \Phi_z^{(n)} = b_z \Psi_z^{(n-1)} + (b_z - 1) \partial_z^{(n-1)} p \\ \Phi_x^{(n)} = b_x \Psi_x^{(n-1)} + (b_x - 1) \partial_x^{(n-1)} p \\ b_z = e^{-d(z)\Delta t} \\ b_x = e^{-d(x)\Delta t} \end{cases} \quad (3.18)$$

where the

The implementation of acoustic wave propagation with CPML is `src/fwdTimeCpmlFor2dAw.m` and `src/fwdTimeCpmlFor3dAw.m`. For higher efficiency of the simulation, the `fwdTimeCpmlFor2dAw.m`, which is used frequently in the seismic imaging algorithms, is also implemented with C++ and mex file.

3.5 Numerical Artifacts and Instabilities

In seismic simulations, there are several parameters you should play with, such as the spatial grid, source function, sampling rate, etc. However, for the stability and accuracy of the numerical scheme, some premises should be honored when we adjust values for these parameters. Particularly, when Nyquist sampling criteria for the finite difference wave field simulation has not been satisfied on space or time domain, some numerical artifacts and instabilities will occur. In summary, when the spatial sampling rate is too low, the solution suffers numerical grid dispersion; when the temporal sampling rate is too low, the Courant instability happens.

To avoid the occurrence of grid dispersion the following criteria for the spatial grid spacing Δu has to be satisfied

$$\Delta u \leq \frac{\lambda_{\min}}{n} = \frac{V_{\min}}{nf_{\max}} \quad (3.19)$$

where n is the number of sampling points per wavelength, λ_{\min} and V_{\min} are the minimal wave length and velocity, and f_{\max} is the maximal frequency. Video clips of acoustic wave propagation with and without numerical grid dispersion can be found at: <https://www.youtube.com/watch?v=scBvd3FQ73U> and <https://www.youtube.com/watch?v=Q17tieZuhJQ>

In order to avoid the Courant instability, the time step Δt must be less than the time for the wave to travel between two adjacent sampling points with grid spacing Δu . In the 2D scenario,

$$\frac{\sqrt{2}V_{\max}\Delta t}{\min\{\Delta z, \Delta x\}} \leq \epsilon \leq 1. \quad (3.20)$$

Examples of acoustic wave propagation which are Courant stable and Courant instable can be found at <https://www.youtube.com/watch?v=G11pNm3jF8g> and <https://www.youtube.com/watch?v=wMAuhW7gzdM>. In the Courant instable case, noticing the color bar, the solution actually blows up over time.

One method to alleviate the numerical dispersion is flux-corrected transport (FCT) introduced by [8], which is implemented for both acoustic and elastic waves in SSSI (see `src/fctForAw.m` and `src/fctForEw.m`). In the

default simulation of acoustic and elastic waves in SSSI the application of FCT is muted, which may can slow down the simulation considerably. If possible, we strongly suggest to modify the sampling rate and Δu to avoid the grid dispersion than employing the FCT.

3.6 2-D Acoustic Wave Equation in Frequency Domain

In signal processing, a signal transformed into frequency domain may unveil hidden information in time domain and brings new processing techniques. The acoustic wave equation given in (3.1) which is time dependent can be transformed and solved in frequency domain as well. Taking Fourier transform of (3.1) with respect to t on both sides

$$\frac{\omega^2}{v^2(z, x)} P_\omega(z, x) + \nabla^2 P_\omega(z, x) = -F_\omega(z, x). \quad (3.21)$$

We spatially discretize the simulated region as we did for the FDM in time domain. Let $P_{i,j}^{(\omega)} := P_\omega(i\Delta z, j\Delta x)$, $F_{i,j}^{(\omega)} := F_\omega(i\Delta z, j\Delta x)$, $v_{i,j} := v(i\Delta z, j\Delta x)$ with 1st-order finite difference approximation of the Laplace operator. We obtain the following discrete equation for a specific frequency ω

$$\begin{aligned} \frac{\omega^2}{v_{i,j}^2} P_{i,j}^{(\omega)} + \left[\frac{P_{i-1,j}^{(\omega)} - 2P_{i,j}^{(\omega)} + P_{i+1,j}^{(\omega)}}{\Delta z^2} \right] \\ + \left[\frac{P_{i,j-1}^{(\omega)} - 2P_{i,j}^{(\omega)} + P_{i,j+1}^{(\omega)}}{\Delta x^2} \right] = -F_{i,j}^{(\omega)}. \end{aligned} \quad (3.22)$$

There are some advantages of Finite Difference in Frequency Domain (FDFD) over Finite Difference in Time Domain (FDTD), we will discuss momentarily. FDFD transforms wave equation for wave fields at a constant frequency into a linear system

$$\mathbf{A}^{(\omega)} \mathbf{p}^{(\omega)} = \mathbf{f}^{(\omega)}, \quad (3.23)$$

where \mathbf{f} and \mathbf{p} are the vectorized source term $F_{i,j}$ and pressure field $P_{i,j}$.

In order to solve acoustic wave equation in frequency ω , we only have to solve the matrix equation (3.23). The matrix \mathbf{A} is highly sparse (see Figure 5), because $P_{i,j}$ is only dependent on its adjacent grid points and such that each row of \mathbf{A} has at most five nonzero entries. Matlab has well developed functions to invert the sparse matrix. Other than the tool come along with Matlab, when the size of \mathbf{A} is huge, we recommend a well developed package with a Matlab interface called MUMPS (MULTifrontal Massively Parallel

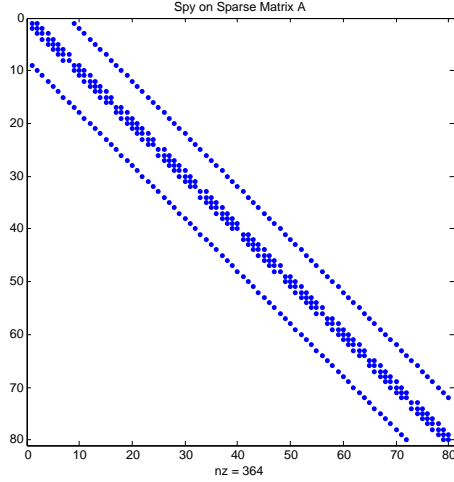


Figure 5: sparse FDFD matrix \mathbf{A} for a specific frequency ω . Blue dots denotes the nonzero elements.

sparse direct Solver) to solve the sparse matrix inversion problem. The concerning of data volume to store and process is another point which made FDFD excels FDTD for our application. Instead of processing the whole data set in time domain, we can solve the wave equation only in the important part of spectrum using FDFD, which considerably reduces the size of the data and makes large-scale problems (e.g., full waveform inversion) feasible. For speed purposes, since the wave equations are independent over frequencies, FDFD can be easily parallelized across frequencies.

An interesting examples is the Green's function, which is the solution of wave equation with an impulse source term. Numerically, the Green's function cannot be solved accurately with FDTD, because of the impulse function has components in arbitrarily high frequency. This will bring serve grid dispersion to the numerical solution. However this is not a problem for FDFD. The Green's Function G_ω in frequency domain, which solves

$$\frac{\omega^2}{v^2(z, x)} G_\omega(z, x) + \nabla^2 G_\omega(z, x) = \delta_\omega(z - z_0, x - x_0) \quad (3.24)$$

is a the solution of (3.23) with the source term \mathbf{f} contains all one's.

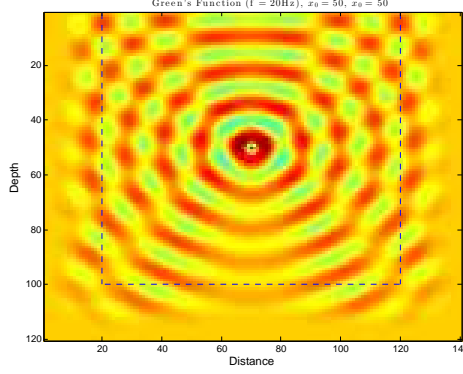


Figure 6: Real part of the Green's function at frequency 20 Hz

4 Numerical solution for elastic wave equation

In the real media, the seismic waves are in the form of elastic waves which are composed of both P-wave and S-wave

$$\left\{ \begin{array}{ll} v_z = v_{zp} + v_{zs}, & v_x = v_{xp} + v_{xs} \\ \frac{\partial v_{zp}}{\partial t} = \alpha^2 \frac{\partial A}{\partial z}, & \frac{\partial v_{xp}}{\partial t} = \alpha^2 \frac{\partial A}{\partial x} \\ \frac{\partial v_{zs}}{\partial t} = -\beta^2 \frac{\partial B}{\partial x}, & \frac{\partial v_{xs}}{\partial t} = \beta^2 \frac{\partial B}{\partial z} \\ A = \frac{\partial s_z}{\partial z} + \frac{\partial s_x}{\partial x}, & B = \frac{\partial s_x}{\partial z} - \frac{\partial s_z}{\partial x} \end{array} \right. \quad (4.1)$$

where (v_z, v_x) is the particle velocity field, v_{up} is P-wave field in u -direction ($u = x, z$), v_{us} is the S-wave fields in u -direction ($u = x, z$), and (s_z, s_x) : particle displacement vector. We can see that in this form, we separate the P-wave and S-wave fields in the solution [4]. Since we already split the velocity field into z and x directions, we use a split PML (SPML) for the FDFD. See `src/fwdTimeSpmlFor2dEw.m`

5 Migration

The goal of seismic imaging is to recover the subsurface structures with the recorded seismic data. This imaging process is typically computationally intense. There are many imaging algorithms. In SSSI we implemented a few representative ones, such as Kirchhoff's migration, Reverse Time Migration

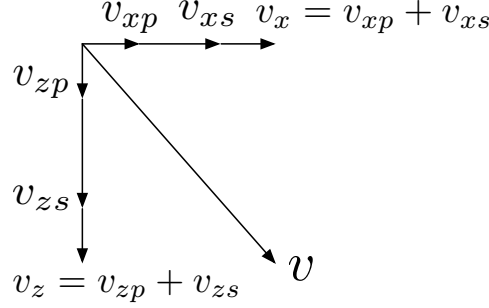


Figure 7: Velocity separation

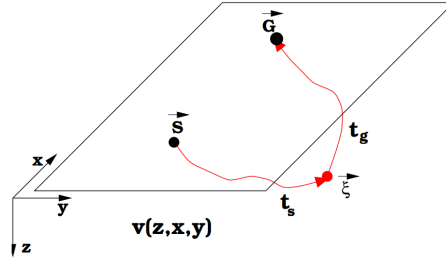


Figure 8: Reflector, and travel times (adapted from [2])

(RTM), Least square RTM (LSRTM), and Full Waveform Inversion (FWI) in frequency domain. All methods we implemented in SSSI are prestack.

5.1 Kirchhoff's migration

Let \mathbf{x}_r be the location of the receiver and \mathbf{x}_s be the location of the source. Then $\mathbf{m} = (\mathbf{x}_r + \mathbf{x}_s)/2$ is the midpoint, and $\mathbf{h} = (\mathbf{x}_r - \mathbf{x}_s)/2$ is the half offset. The involved derivation of Kirchhoff's migration (see [15] for details) is omitted here. Only the simple imaging formula is given as below

$$I(\xi) = \int_{\Omega_\xi} W(\xi, \mathbf{m}, \mathbf{h}) dt (t = t_d(\xi, \mathbf{m}, \mathbf{h}), \mathbf{m}, \mathbf{h}) d\mathbf{m} d\mathbf{h}, \quad (5.1)$$

where limited region Ω_ξ is centered around the location ξ in the m plane, called the migration aperture. $t_d = t_s[\mathbf{x}, \mathbf{s}, v(z, x, y)] + t_g[\mathbf{x}, \mathbf{g}, v(z, x, y)]$. For reflector $\xi = (z_\xi, x_\xi, y_\xi)$ in 3D and $\xi = (z_\xi, x_\xi)$ in 2D case.

An example of Kirchhoff's migration for the fault model (available in the `modelData` folder) can be found in `src/mainKirkCpmlFor2dAw_fault.m`,

which is inherited from an existing package [10]. The most important part of Kirchhoff's migration is the computation of travel times t_s and t_g . These can be done either with `src/ray2d.m` from [10] or `src/eikonal2d.m` which solves the 2D Eikonal equation using a fast sweeping method [17]. The Eikonal equation is of the form

$$|\nabla T(\mathbf{x})|c(\mathbf{x}) = 1, \quad \mathbf{x} \in \mathbb{R}^2 \quad (5.2)$$

subject to $T(\mathbf{x}) = 0$ for a boundary Ω . In the SSSI we consider the source point to be a single grid point \mathbf{x}_s . When we solve the Eikonal equation in a truncated region, $T(\mathbf{x})$ gives the time of first arrival in this region.

5.2 Reverse Time Migration (RTM)

The Reverse Time Migration (RTM) utilizes received surface data to locate the reflectors. RTM has a relatively old origin, however it has not been used routinely until recently because of its computation expense. In RTM a forward wave propagation and a reverse time wave propagation need to be simulated, which requires a lot of computation power. However, RTM has a number of advantages over conventional depth migration methods such as handling evanescent energy and no dip limitation [12, 1]. In summary, RTM is implemented in three steps. Firstly, record forward wave field $p_f(\mathbf{x}, t; \mathbf{x}_s)$ through the true velocity model. Secondly, solve the reverse wave field $p_r(\mathbf{x}, t)$ by the received surface data through the incident model, which is typically a smoothed version of the true model. The second step is equivalent to reverse the recorded traces in time. Then input those traces as source function and solve the wave field. Thirdly, superposition of above using an imaging condition (chosen from a number existing imaging conditions).

For instance, the imaging condition with cross-correlation is

$$I(\mathbf{x}) = \sum_{\mathbf{x}_s} \frac{\int_0^T p_f(\mathbf{x}, t; \mathbf{x}_s) p_r(\mathbf{x}, t) dt}{\int_0^T |p_f(\mathbf{x}, t; \mathbf{x}_s)|^2 dt + \epsilon^2}. \quad (5.3)$$

The RTM result can be viewed in this video clip <https://www.youtube.com/watch?v=1L-UpW-5xms&feature=youtu.be>.

RTM serves the imaging complex structures purposes as a reliable method. However, there are still some distortions caused by RTM crosstalk artifacts. Mathematically, the migration operator is just a adjoint of the forward modeling operator, it does not minimize the mismatch of the observed and simulated traces. Least Square RTM (LSRTM) introduced by [13] derives an

imaging condition by formulating migration as an inverse problem based on a least-squares misfit function.

$$\begin{aligned}
J(\Delta m) &= \frac{1}{2} \sum_{\omega} \sum_{\mathbf{x}_s} \sum_{\mathbf{x}_r} \left| P_{\omega}^{(\text{sim})}(\mathbf{x}_r; \mathbf{x}_s) - P_{\omega}^{(\text{obs})}(\mathbf{x}_r; \mathbf{x}_s) \right|^2 \\
&= \frac{1}{2} \sum_{\omega} \sum_{\mathbf{x}_s} \sum_{\mathbf{x}_r} \left| \omega^2 F_{\omega}(\mathbf{x}_s) \sum_{\mathbf{x}} \Delta m(\mathbf{x}) G_{\omega}(\mathbf{x}; \mathbf{x}_s) G_{\omega}(\mathbf{x}_r; \mathbf{x}) \right. \\
&\quad \left. - \delta P_{\omega}(\mathbf{x}_r; \mathbf{x}_s) \right|^2 \\
&= \frac{1}{2} \| \mathbf{L} \Delta m - \delta \mathbf{P}_{\omega} \|_2^2,
\end{aligned} \tag{5.4}$$

where Δm is the module perturbation (roughly saying, the reflectors), \mathbf{L} is the forward modeling operator. The superscript (sim) means simulated results and (obs) means observed. The optimized model perturbation by Gauss-Newton method

$$\Delta m^{(\text{opt})} = \left(\mathbf{L}^{\dagger} \mathbf{L} \right)^{-1} \mathbf{L}^{\dagger} \delta \mathbf{P}_{\omega}. \tag{5.5}$$

Apparently, LSRTM is the simply a preconditioned RTM migration result $\mathbf{L}^{\dagger} \delta \mathbf{P}_{\omega}$. Instead of solving the huge inverse matrix $(\mathbf{L}^{\dagger} \mathbf{L})^{-1}$, one can use a iterative method to implement LSRTM [6].

6 Full Waveform Inversion (FWI) in frequency domain

The migration operator should have a incident model (usually a smoothed version of the true model) as an input. However, an accurate initial velocity model is not always available. The Full Waveform Inversion (FWI) (see [16] as an overview) provides flexibility of updating the initial model iteratively. $J(\Delta m)$ can be optimized by more approaches: conjugate-gradient, quasi-Newton, etc.

Algorithm 1 Full Waveform Inversion

- 1: **while** $m(\mathbf{x})$ is not converged **do**
 - 2: Generate Green's functions at each shot and receiver for \mathbf{L}
 - 3: Generate scattering wave field $\delta \mathbf{P}_{\omega} = \mathbf{P}_{\omega}^{(\text{obs})} - \mathbf{G}_{\omega}(m) \mathbf{F}_{\omega}$
 - 4: $\Delta m^{(\text{opt})} = \arg \min J(\Delta m) = \arg \min \frac{1}{2} \| \mathbf{L} \Delta m - \delta \mathbf{P}_{\omega} \|_2^2$
 - 5: $m \leftarrow m + \Delta m$
 - 6: **end while**
-

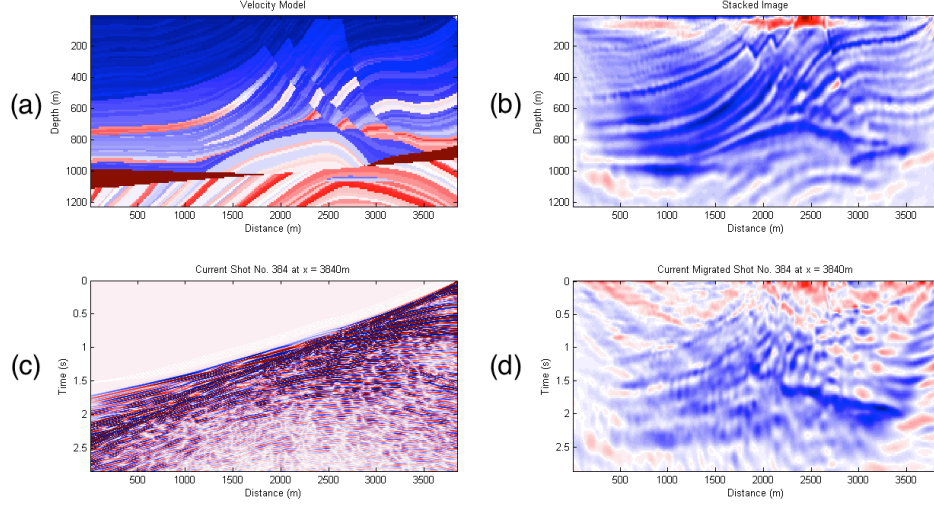


Figure 9: RTM results. (a) Marmousi model, (b) accumulated RTM result; (c) a sample shot record; (d) migrated result of the sample shot. The parameters settings are 15 Hz (center frequency) Ricker wavelet, 384 sensors, 384 shots, grid size 122×384 , $\Delta z = \Delta x = 24m$.

References

- [1] E. Baysal, D. D. Kosloff, and J. W. C. Sherwood, *Reverse time migration*, *Geophysics* **48** (1983), 1514–1524.
- [2] B. L. Biondi, *3d seismic imaging*, Investigations in Geophysics Series No. 14, SEG, Tulsa, Oklahoma, USA, 2006.
- [3] C. D. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef, *A nonreflecting boundary condition for discrete acoustic and elastic wave equations*, *Geophysics* **50** (1985), 705–708.
- [4] K. Chen, *Finite-difference simulation of elastic wave with separation in pure p- and s-modes*, *Journal of Computational Methods in Physics* **2014** (2014), no. ID108713, 1–15.
- [5] R. Clayton and B. Engquist, *Absorbing boundary conditions for acoustic and elastic wave equations*, *Bulletin of the Seismological Society of America* **67** (1977), 1529–1540.

- [6] S. Dong, J. Cai, M. Guo, S. Suh, Z. Zhang, B. Wang, and Z. Li, *Least-squares reverse time migration: towards true amplitude imaging and improving the resolution*, SEG Technical Program Expanded Abstract 2012 (Las Vegas), SEG, 2012, pp. 1–5.
- [7] B. Engquist and A. Majda, *Absorbing boundary conditions for numerical simulation of waves*, Proceedings of the National Academy of Sciences **74** (1977), 1765–1766.
- [8] T. Fei and K. Larner, *Elimination of numerical dispersion in finite-difference modeling and migration by flux-corrected transport*, Geophysics **60** (1995), no. 6, 1830–1842.
- [9] D. Komatitsch and R. Martin, *An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation*, Geophysics **72** (2007), SM155–SM167.
- [10] S. Kozola, *Large data in matlab: A seismic data processing case study*, 2011.
- [11] R. J. Luebbers and F. Hunsberger, *Fdtd for nth-order dispersive media*, IEEE Transactions on Antennas and Propagation **40** (1992), 1297–1301.
- [12] G. A. McMechan, *Migration by extrapolation of time-dependent boundary values*, Geophysical Prospecting **31** (1983), 413–420.
- [13] T. C. Nemeth, C. Wu, and G. T. Schuster, *Least-square migration of incomplete reflection data*, Geophysics **64** (1999), 208–221.
- [14] J. A. Roden and S. D. Gedney, *Convolution pml (cpml): An efficient fdtd implementation of the cfs-pml for arbitrary media*, Microwave and Optical Technology Letters **27** (2000), 334–339.
- [15] W. Schneider, *Integral formulation for migration in two and three dimensions*, Geophysics **43** (1978), 49–76.
- [16] J. Virieux and S. Operto, *An overview of full-waveform inversion in exploration geophysics*, Geophysics **74** (2009), no. 6, WCC1–WCC26.
- [17] H. Zhao, *A fast sweeping method for eikonal equations*, Mathematics of computation **74** (2004), no. 250, 603–627.