Seismic Simulation, Survey, and Imaging (S3I) Ver. 1.0

User Guide

Lingchen Zhu[a]
Entao Liu[b]
Lijun Zhu[c]
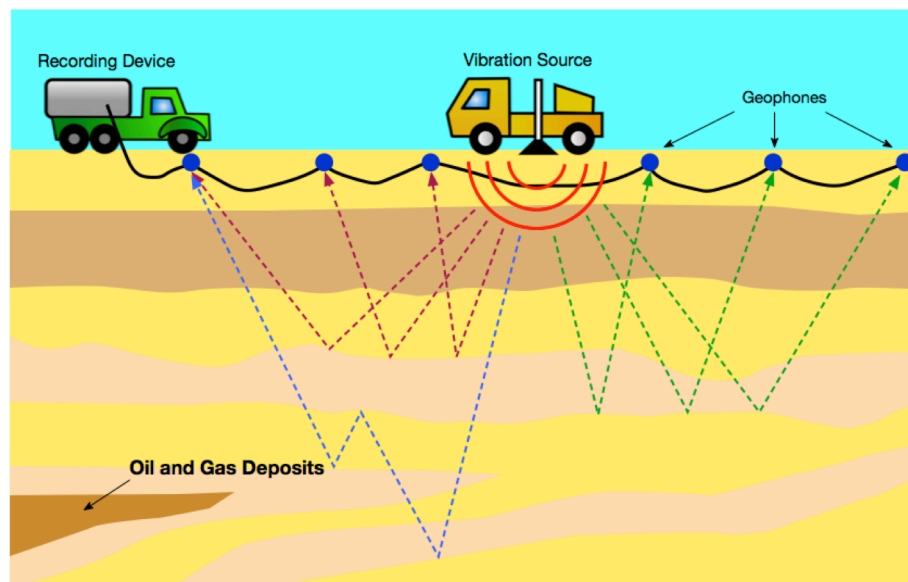
**Center for Energy and Geo Processing
at Georgia Tech and KFUPM**

Atlanta, GA, Feb. 2015

Contents

0   Requirements and License

The S3I is a MATLAB® based package. Besides MATLAB (2012a or later version recommend), you may also need a C++ compiler (e.g. gcc or Microsoft C/C++ compiler) to generate the MEX-files if none of the provided MEX-files is compatible with your system architecture. In order to run parallel computing, MPI is also required and here we recommend OpenMPI.

S3I is a free software package: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2.0 of the License only. This S3I package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY. If you find any glitches or bugs within it, please contact the authors. We appreciate you contributions.

1   Introduction

The S3I is designed to provide a package for numerical simulations in exploration geophysics. It targets students as well as professionals in exploration geophysics. The most important purpose is to provide an easy, well organized library to the interested users to learn some popular algorithms and numerical schemes in exploration geophysics rather than the high performance of the computation. Thus, MATLAB is adopted as the coding platform for the readability of the code, ease of data visualization, etc. It is well known that nested for-loops in MATLAB is much slower than the complied languages. In order to have make the S3I efficient, we used C to generate MEX-files for some frequently invoked functions. Currently, the major functions have been implemented in S3I are as follows:

- Acoustic wave simulation for 2D/3D

- Elastic wave simulation for 2D

- Kirchhoff's migration

- Reverse Time Migration (RTM)

- Least Square RTM

- Full Waveform Inversion (FWI)

With this user guide, the user may easily and quickly start using the S3I for appropriate applications. An exhaustive review of numerical simulation

of wave equation and seismic imaging is out of the scope of this user guide. Therefore, only necessary equations are showed and explained to keep the contents concise. The Interested user is referred to according references for more details.

2 Numerical Simulation of Acoustic Wave

2.1 Acoustics Wave Equation

The seismic method is one of the primary tools in exploration geophysics. In a seismic survey, man-made vibration sources (e.g. dynamite, viborseis trucks, and air guns) are fired off. Then the wave propagates through the subsurface geological media. The wave filed which contains direct wave, reflection, and refraction, will be recorded by the array of receivers (e.g. geophones and hydrophones). In order to simulate this physical process with computers, it is crucial to solve wave equations accurately and efficiently with numerical methods. The real earth is an elastic media, such that the seismic waves contain both P-wave and S-wave components. For simplicity, we some times only consider the P-wave field, which is 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 [13]. Therefore, we usually assume a constant density of the media. Then the acoustic wave equation can be written as

$$\frac{1}{v^2(\boldsymbol{x})}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} + f(\boldsymbol{x},t) = \nabla^2 p(\boldsymbol{x},t), \tag{2.1}$$

where $p(\boldsymbol{x})$ is the filed of pressure variation and $\boldsymbol{x} = (z,x)$ or $\boldsymbol{x} = (z,x,y)$ 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 $v(\boldsymbol{x})$ is the velocity of P-wave at location $\boldsymbol{x}$ and $f(\boldsymbol{x},t)$ is the source time function. Moreover, Δ in (2.1) is the Laplace operator defined as

$$\Delta := \begin{cases} \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial z^2} & \text{for 2D} \\ \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{\partial^2}{\partial z^2} & \text{for 3D} \end{cases} \tag{2.2}$$

The Ricker wavelet is a widely used for seismic source time function whose amplitude $A(t)$ with the peak frequency f at time t is computed as:

$$A(t) = (1 - 2\pi^2 \mathsf{f}^2 t^2)e^{-\pi^2 \mathsf{f}^2 t^2}. \tag{2.3}$$

4

The `src/ricker.m` provides a Ricker wavelet generator which requires the peak frequency f, number of time samples n, sampling interval 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 the S3I the user can choose from several different wavelets for the source time function (Ricker wavelet, Fuchs-Mueller wavelet, sine cube wavelet, etc) which can be generated using `src/waveletGenerator.m`.
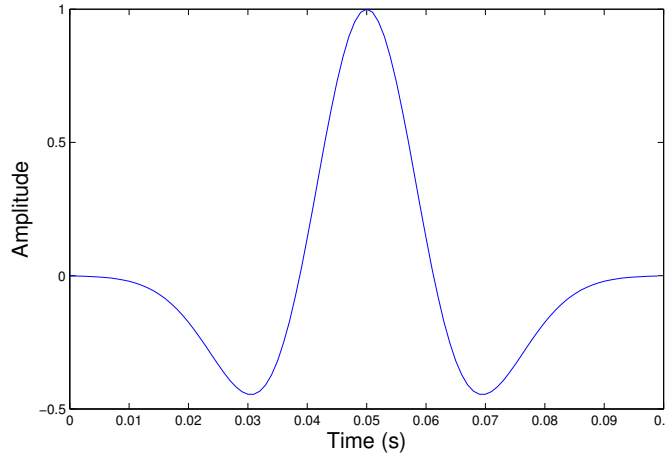

Figure 1: Ricker wavelet with peak frequency $f = 20$Hz

2.2   Finite Difference Method on Standard Grid

Currently, there are several popular method to solve the wave equation numerically, such as pseudo-spectral method, Finite Difference Method (FDM), Finite Elements Method (FEM) method, etc. Each method has its pros and cons. Concretely, the pseudo-spectral method, which is based on Fourier transform in spatial domain, provides accurate and efficient solutions for the smoothly varied media but not for sharply varied media [16, 10, 11]. The FDM [1, 6] adopted by S3I is widely used for its ease to implement and high accuracy and efficiency. However, the FDM is incapable to provide accurate solution for media with complex geometry structure and rugged topography. The FEM usually provides more accurate results than FDM with a higher computational and storage expense [17].

In order to solve the wave equation with FDM, the continuous functions and models are represented by their values at grid points and and deriva-

5

tives are approximated by linear combination of these values. For instance, in a 2D region we use the uniformly 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 provides for an approximated solution on these grid points.

The estimation of derivatives in (2.1) is crucial in FDM. Let us revisit with the definition of the partial derivative,

$$
\begin{aligned}
\frac{\partial p(z,x,t)}{\partial z} &= \lim_{\Delta z \to 0} \frac{p(z+\Delta z, x, t) - p(z,x,t)}{\Delta z} \\
&\approx \frac{p(z+\Delta z, x, t) - p(z,x,t)}{\Delta z}.
\end{aligned}
\tag{2.4}
$$

Assuming the grid spacing Δz is small enough, $(p(z+\Delta z, x, t) - p(z,x,t))/\Delta z$ is an accurate estimation of the derivative. After applying this approximation twice, we derive an central difference scheme for the second-order derivative in (2.1)

$$
\begin{aligned}
\frac{\partial^2 p(z,x,t)}{\partial z^2} &= \frac{\partial}{\partial z} \frac{\partial p(z,x,t)}{\partial z} \approx \frac{\frac{\partial p(z,x,t)}{\partial z} - \frac{\partial p(z-\Delta z, x, t)}{\partial z}}{\Delta z} \\
&\approx \frac{p(z+\Delta z, x, t) - 2p(z,x,t) + p(z-\Delta z, x, t)}{\Delta z^2}.
\end{aligned}
\tag{2.5}
$$

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

$$
\begin{aligned}
&\frac{1}{v_{i,j}^2} \frac{p_{i,j}^{(n+1)} - 2p_{i,j}^{(n)} + p_{i,j}^{(n-1)}}{\Delta t^2} - f_{i,j}^{(n)} \\
&= \frac{p_{i+1,j}^{(n)} - 2p_{i,j}^{(n)} + p_{i-1,j}^{(n)}}{\Delta z^2} + \frac{p_{i,j+1}^{(n)} - 2p_{i,j}^{(n)} + p_{i,j-1}^{(n)}}{\Delta x^2}.
\end{aligned}
\tag{2.6}
$$

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

$$
\begin{aligned}
p_{i,j}^{(n+1)} &= \frac{v_{i,j}^2 \Delta t^2}{\Delta z^2} \left(p_{i+1,j}^{(n)} - 2p_{i,j}^{(n)} + p_{i-1,j}^{(n)} \right) \\
&\quad + \frac{v_{i,j}^2 \Delta t^2}{\Delta x^2} \left(p_{i,j+1}^{(n)} - 2p_{i,j}^{(n)} + p_{i,j-1}^{(n)} \right) \\
&\quad + 2p_{i,j}^{(n)} - p_{i,j}^{(n-1)} + v_{i,j}^2 \Delta t^2 f_{i,j}^{(n)}.
\end{aligned}
\tag{2.7}
$$

In (2.7), all values of p are computed on integer grid points as illustrated in Figure 2(a). The FDM on standard grid is an easy-to-understand numerical scheme, which serves as an excellent explanatory example. However, S3I used another scheme, which will discuss momentarily, for its better accuracy.
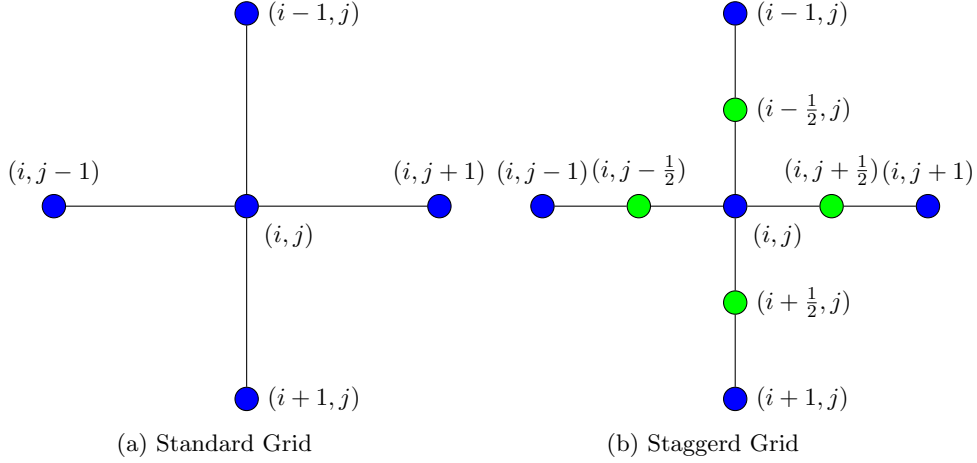

(a) Standard Grid   (b) Staggerd Grid

Figure 2: Discretization Grids

2.3   Finite Difference Method on Staggered Grid

With a sophisticated design, it turns out we can 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 2(b).

By Taylor expansion of a function $p(u)$ on the half grid points

$$p\left(u + \frac{2k+1}{2}\Delta u\right) = p(u) + \sum_{n=1}^{\infty} \frac{1}{n!}\frac{\partial^n p(u)}{\partial u^n}\left(\frac{2k+1}{2}\Delta u\right)^n, \qquad (2.8)$$

$$p\left(u - \frac{2k+1}{2}\Delta u\right) = p(u) + \sum_{n=1}^{\infty} \frac{(-1)^n}{n!}\frac{\partial^n p(u)}{\partial u^n}\left(\frac{2k+1}{2}\Delta u\right)^n, \qquad (2.9)$$

where $u = z, x$; $k = 0, 1, 2, \ldots$. The difference between (2.8) and (2.9)

7

cancels all the terms with even n. So it implies that for $k = 0, 1, 2, \ldots,$

$$
\begin{aligned}
&\frac{p\left(u + \frac{2k+1}{2}\Delta u\right) - p\left(u - \frac{2k+1}{2}\Delta u\right)}{(2k+1)\Delta u} \\
&= \frac{\partial p(u)}{\partial u} + \sum_{n=1}^{\infty} \frac{1}{(2n+1)!} \frac{\partial^{(2n+1)} p(u)}{\partial u^{(2n+1)}} \left(\frac{2k+1}{2}\Delta u\right)^{2n}.
\end{aligned}
\tag{2.10}
$$

We can approximate $\frac{\partial p(u)}{\partial u}$ using a linear combination of the finite differences based on (2.10) as follows

$$
\begin{aligned}
\frac{\partial p(u)}{\partial u} &= \sum_{k=0}^{N-1} a_k \frac{p\left(u + \frac{2k+1}{2}\Delta u\right) - p\left(u - \frac{2k+1}{2}\Delta u\right)}{(2k+1)\Delta u} \\
&= \sum_{k=0}^{N-1} a_k \left[\frac{\partial p(u)}{\partial u} + \frac{\Delta u^2}{3! \cdot 2^2}(2k+1)^2 \frac{\partial^3 p(u)}{\partial u^3} \right. \\
&+ \frac{\Delta u^4}{5! \cdot 2^4}(2k+1)^4 \frac{\partial^5 p(u)}{\partial u^5} + \cdots \\
&\left. + \frac{\Delta u^{2N-2}}{(2N-1)! \cdot 2^{2N-2}}(2k+1)^{2N-2} \frac{\partial^{2N-1} p(u)}{\partial u^{2N-1}} + o(\Delta u^{2N}) \right].
\end{aligned}
\tag{2.11}
$$

If the weights $\{a_k\}_{k=0}^{N-1}$ are assigned properly, all the term on the right hand side of (2.11) can be eliminated except $\frac{\partial p(u)}{\partial u}$ and the residual term $o(\Delta u^{2N})$. This N terms linear combination servers as an approximation of the derivative of order $2N$. Additionally, the derivative $\frac{\partial p(u)}{\partial u}$ and function $p(u)$ itself live on different colors of dots, i.e. $\frac{\partial p(u)}{\partial u}$ on half gird points, and $p(u)$ on integer grid points. This staggered grid does not implies we actually need access of the values on half grid points when we solve the wave equation using FDM because using this approximation twice, the second-order derivative $\frac{\partial^2 p(u)}{\partial u}$ and the $p(u)$ both live on integer grid points.

Based on the goal of choosing the weights $\{a_k\}_{k=0}^{N-1}$, they should satisfy the following system of linear equations

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

$$N = 1 : a_0 = 1$$
$$N = 2 : a_0 = 9/8, a_1 = -1/24$$
$$N = 3 : a_0 = 75/64, a_1 = -25/384, a_2 = 3/640$$
$$\vdots$$

`src/dCoef.m` gives a solution for the above system of arbitrary N. Moreover, the finite difference operator on staggered grid is fulfilled by `src/diffOperator.m` which can perform arbitrarily high order of approximation. The default value of `nDiffOrder = 3`, which generates 6-th order of approximation of the spatial derivatives, is usually good enough.

2.4 Absorbing Boundary Conditions (ABC)

In real seismic surveys, seismic waves propagate in unbounded half-space media. For the sake of computational efficiency and storage, a seismic survey is only simulated in a truncated region. If no special techniques are applied on the boundary of the simulated region, the recursive formula (e.g. (2.7)) would generate strong reflections which do not physically exist in the real seismic survey. In order to attenuate the reflections, a common method is to impose the absorbing boundary conditions (ABC) [8, 5]. Figure 3 illustrates the 2D scenario, when we pad the original truncated velocity model's left, right, bottom boundaries with absorbing boundaries of a certain width (fulfilled by `src/extBoundary.m`). ABC will attenuated the wave amplitudes in those absorbing boundaries and keep the waveform unaltered outside of the absorbing boundaries. Throughout the S3I, except specified explicitly, we consider the top of the simulated region to be the surface of the earth, which is a free surface without ABC applied.
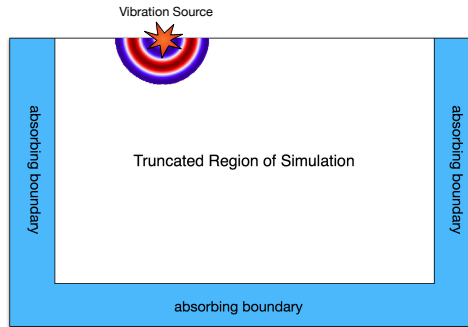

Figure 3: Absorbing boundaries of a 2-D simulation region

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

$$d(u) = e^{-\alpha^2 \text{dist}(u)^2}, \text{ where } u = x, z \qquad (2.12)$$

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

The source file which implements the acoustic wave propagation with sponge ABC is `src/fwdTimeSpongeFor2dAw.m`. Throughout of the source files in the S3I, we follow the following naming rules: `fwd` means forward (`rvs` means reverse); `Time` means the equation is solved in the time domain (`freq` means in frequency domain); `For2d` means 2D (`For3d` means 3D); and `Aw` is short for acoustic wave (`Ew` is for elastic wave). Although the waves are considerably attenuated by ABC, the reflection still can not be completely eliminated. This is the motivation in the S3I we utilize the method called Perfectly Matched Layer as well.

2.4.1 Perfectly Matched Layer (PML)

The Perfectly Matched Layer (PML) method was originally formulated for use of electromagnetic equations. In seismic simulations the PML is proven to be effective for wave equations (both acoustic and elastic) as well [14]. It has 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

$$\tilde{u}(u) = u + \frac{1}{j\omega} \int_0^u d(s)\mathrm{d}s. \qquad (2.13)$$

Equivalently,

$$\frac{\partial}{\partial \tilde{u}} = \frac{j\omega}{j\omega + d(u)} \frac{\partial}{\partial u} = s_u(j\omega) \frac{\partial}{\partial u}. \qquad (2.14)$$

In homogeneous media, the acoustic wave equation has a solution $A \exp(-j(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t))$, where A represents the amplitude and polarization of the plane wave. The $\boldsymbol{k} = k_x \hat{\mathbf{x}} + k_y \hat{\mathbf{y}} + k_z \hat{\mathbf{z}}$ denotes the wave vector, which indicates the direction of wave propagation of the plane wave. $\boldsymbol{x} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$ is the
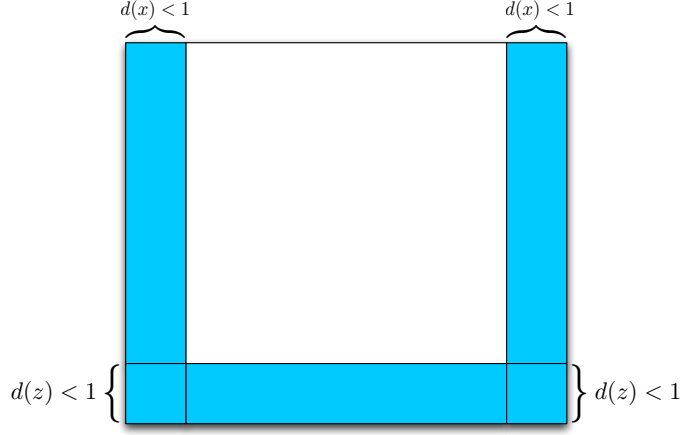
Figure 4: The damping profile $d(u)$

position vector. By substitution in (2.13), we can derive another solution of the acoustic equation

$$A \exp(-j(k_x \tilde{x} + k_y y + k_z z - \omega t)) =$$
$$A \exp(-j(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t)) \exp(-kx/\omega \int_0^x d(s)ds). \tag{2.15}$$

In the original simulated region the the new solution is equivalent to the old one, since $d_x = 0$. Additionally, in the $\hat{\boldsymbol{n}} = \hat{\boldsymbol{x}}$ 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 the S3I we implement a non-split method, the Convolutional PML (CPML) [18, 22, 14], which is more natural for an acoustic (pressure) source function. In time domain,

$$\frac{\partial}{\partial \tilde{u}} = s_u(t) * \frac{\partial}{\partial u} = \frac{\partial}{\partial u} - \left(d(u)H(t)e^{-d(u)t} \right) * \frac{\partial}{\partial u}. \tag{2.16}$$

11

The convolution can be performed as follows,

$$
\begin{cases}
\dfrac{\partial^2 p}{\partial t^2} = v^2(P_z + P_x) \\[2mm]
P_z = \dfrac{\partial A_z}{\partial z} + \Psi_z \\[2mm]
P_x = \dfrac{\partial A_x}{\partial x} + \Psi_x \\[2mm]
A_z = \dfrac{\partial p}{\partial z} + \Phi_z \\[2mm]
A_x = \dfrac{\partial p}{\partial x} + \Phi_x
\end{cases}
\tag{2.17}
$$

and

$$
\begin{cases}
\Psi_z^{(n)} = b_z \Psi_z^{(n-1)} + (b_z - 1)\partial_z^{(n-1)} A_z \\[2mm]
\Psi_x^{(n)} = b_x \Psi_x^{(n-1)} + (b_x - 1)\partial_x^{(n-1)} A_x \\[2mm]
\Phi_z^{(n)} = b_z \Psi_z^{(n-1)} + (b_z - 1)\partial_z^{(n-1)} p \\[2mm]
\Phi_x^{(n)} = b_x \Phi_x^{(n-1)} + (b_x - 1)\partial_x^{(n-1)} p \\[2mm]
b_z = e^{-d(z)\Delta t} \\[2mm]
b_x = e^{-d(x)\Delta t}
\end{cases}
\tag{2.18}
$$

The implementations of acoustic wave propagation with CPML are `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 MEX-files.

2.5   Parallel Computing Using OpenMPI

Parallel computing based on the distributed memory model is supported in the S3I. One can run our code on a computer cluster across multiple computing nodes. As is shown in Figure 5, these computing nodes can be regarded as separate computers connected by a fast network. Each node has its own independent processor(s) and memory, takes charge of its own calculations and communicates with its direct neighboring nodes.

Parallel computing for distributed memory model requires passing messages among different nodes. The standard protocol to achieve this requirement is Message Passing Interface (MPI) of which there are various implementations for different programming languages. The S3I uses OpenMPI[12]
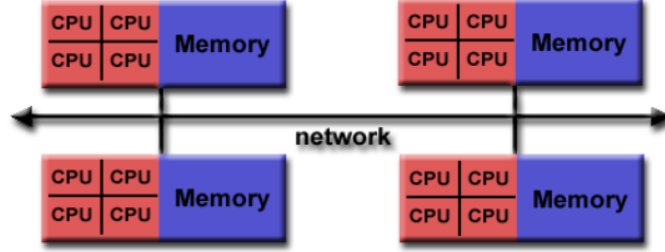
Figure 5: Distributed memory model of a computer cluster

that has been widely accepted by many TOP500 supercomputers to implement distributed-memory parallelism and achieves high efficiency.

In MPI parallel computing, the same program runs on all computing nodes independently after it is correctly compiled by `mpicc`. Each node is assigned a unique identifying number (called "rank"), and the program source code can include logic so that different code paths are followed on different nodes. Since MPI operates in a distributed fashion, any message transfer between nodes must be done explicitly via routine calls provided by OpenMPI, such as `MPI_Send`, `MPI_Recv`, `MPI_Scatter`, `MPI_Gather`, etc.

(a) 2-D data division

(b) 3-D data division

Figure 6: Workload Division

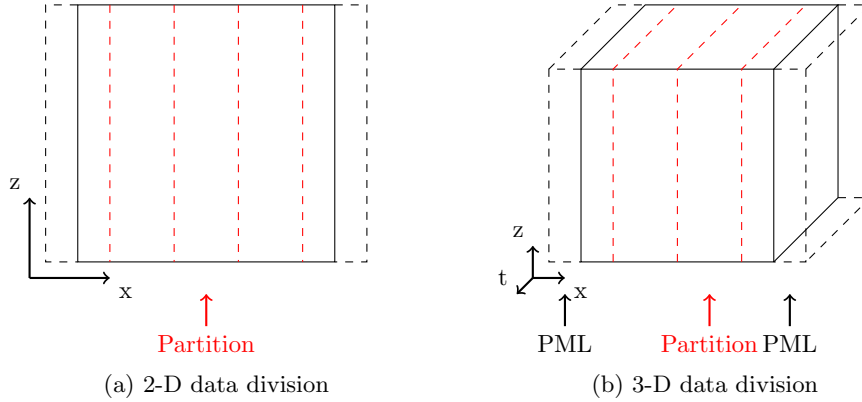In the S3I, the acoustic wave propagation process can be simulated in such a parallel manner. Suppose there are P nodes, each node p ($p = 0, 1, \ldots, P - 1$) takes over a local partition from two-dimensional velocity model $\{v_{i,j}\}$ and three-dimensional input source term $\left\{ f_{i,j}^{(n)} \right\}$ to perform the simulation locally. Assuming that all nodes have similar computational

capacity, these local partitions can be almost evenly divided along the x-axis as follows,

$$n_{x_p} = \begin{cases} \left\lfloor \dfrac{n_x}{P} \right\rfloor + 1, & \text{if } p < \text{mod}(n, P) \\ \left\lfloor \dfrac{n_x}{P} \right\rfloor, & \text{else} \end{cases} \tag{2.19}$$

where n_{x_p} is x-axis length of the local partition assigned to node p and n_x is the total x-axis length of velocity model and input source term including the thickness of PML on both left and right sides. Such a workload division process can be demonstrated in Figure 6 where local partitions are separated by red dash lines. From the 1st-order finite difference approximation of wave equation in (2.7), we can observe that the calculation of any grid value on the next time step $p_{i,j}^{(n+1)}$ only depends on the values of its four direct neighboring grids on the current time step $p_{i-1,j}^{(n)}, p_{i+1,j}^{(n)}, p_{i,j-1}^{(n)}, p_{i,j+1}^{(n)}$, and its value on the current and previous time step $p_{i,j}^{(n)}, p_{i,j}^{(n-1)}$, respectively. In our practical simulations, the forward P-wave propagation process is simulated by (2.17) and (2.18) where ∂_z and ∂_x is approximated by a weighted sum of finite differences described in (2.11). Hereby values of more neighboring grids on the current time step are involved. While calculating the grid values along the boundary of each partition, their neighboring grid values involved in the calculation only exist in the previous or next partition which is stored in another node. Therefore, in order to avoid calculation error, these grid values must be transferred via MPI before they are needed. The final global result will be gathered after all nodes finish calculations of their own partitions.

Parallel forward and backward acoustic wave propagation is implemented in `src/fd_cmex/fwdTimeCpmlFor2dAw_openmpi_mex.c` and `src/fd_cmex/rvsTimeCpmlFor2dAw_openmpi_mex.c`, respectively. In order to evaluate the performance, a MATLAB wrapper interface `mainRtmTimeCpmlFor2dAwOpenMPI.m` sets up simulation parameters. One can call

```
mpirun -n <NP> matlab -nodisplay -r
"mainRtmTimeCpmlFor2dAwOpenMPI"
```

in the console to run the parallel forward acoustic wave propagation, where `<NP>` refers to the number of processors to use.

2.6　Numerical Artifacts and Instabilities

In seismic simulations, there are a number of parameters whose values need to be specified, such as the grid spacing, source function frequency, sampling rate, etc. For the stability and accuracy of the numerical scheme, some premises should be honored when we adjust these parameters. Particularly, when Nyquist sampling criteria for the finite difference wave field simulation has not been satisfied in 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 harmful grid dispersion the following criteria for the spatial grid spacing Δu has to be satisfied

$$\Delta u \leq \frac{\lambda_{\min}}{n} = \frac{V_{\min}}{n f_{\max}} \qquad (2.20)$$

where n is the number of sampling points per wavelength, $\lambda_{\min}$ 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`

To be Courant stable, 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 2D scenarios,

$$\frac{\sqrt{2} V_{\mathbf{max}} \Delta t}{\min\{\Delta z, \Delta x\}} \leq \epsilon \leq 1. \qquad (2.21)$$

Examples of acoustic wave propagation which are Courant stable and Courant instable can be found at `https://www.youtube.com/watch?v=Gl1pNm3jF8g` and `https://www.youtube.com/watch?v=wMAuhW7gzdM`. In the Courant instable case, noting the color bar, the solution actually blows up over time.

An approach to alleviate the numerical dispersion is flux-corrected transport (FCT) introduced by [9], which is implemented for both acoustic and elastic waves in the S3I (see `src/fctForAw.m` and `src/fctForEw.m`). In the default simulations of acoustic and elastic waves the FCT correction is turned off, since it can slow down the simulation significantly. If possible, we strongly suggest to modify the sampling rate and Δu to avoid the grid dispersion rather than employing FCT.

2.7 2D 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 (2.1) which is time dependent can be transformed and solved in frequency domain as well. Taking Fourier transform of (2.1) with respect to t on both sides, we obtain

$$\frac{\omega^2}{c^2(z,x)}P_\omega(z,x) + \nabla^2 P_\omega(z,x) = -F_\omega(z,x). \qquad (2.22)$$

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

$$
\begin{aligned}
\frac{\omega^2}{c_{i,j}^2}P_{i,j}^{(\omega)} &+ \left[\frac{P_{i-1,j}^{(\omega)} - 2P_{i,j}^{(\omega)} + P_{i+1,j}^{(\omega)}}{\Delta z^2}\right] \\
&+ \left[\frac{P_{i,j-1}^{(\omega)} - 2P_{i,j}^{(\omega)} + P_{i,j+1}^{(\omega)}}{\Delta x^2}\right] = -F_{i,j}^{(\omega)}.
\end{aligned}
\qquad (2.23)
$$

There are some advantages of Finite Difference in Frequency Domain (FDFD) over Finite Difference in Time Domain (FDTD) that we will discuss shortly. The Fourier transform convert the second-order derivative with respect to t in the wave equation into a product. So the wave field at a certain frequency ω is determined by a system of linear equations

$$\boldsymbol{A}^{(\omega)}\boldsymbol{p}^{(\omega)} = \boldsymbol{f}^{(\omega)}, \qquad (2.24)$$

where $\boldsymbol{f}^{(\omega)}$ and $\boldsymbol{p}^{(\omega)}$ are the vectorized source term $F_{i,j}$ and pressure field $P_{i,j}$.

The matrix $\boldsymbol{A}$ is highly sparse (see Figure 7), because $P_{i,j}$ is only dependent on its adjacent grid points and such that each row of $\boldsymbol{A}$ contains at most five nonzero entries. MATLAB has been optimized to invert the sparse matrix. Other than the tool come along with MATLAB, when the size of $\boldsymbol{A}$ is huge, we recommend a well developed package with a MATLAB interface called MUMPS (MUltifrontal Massively Parallel sparse direct Solver) to solve the sparse matrix inversion problem. The concerns on data storage and process is another factor which made FDFD excels FDTD for our application. Instead of processing the whole data set in time domain, we
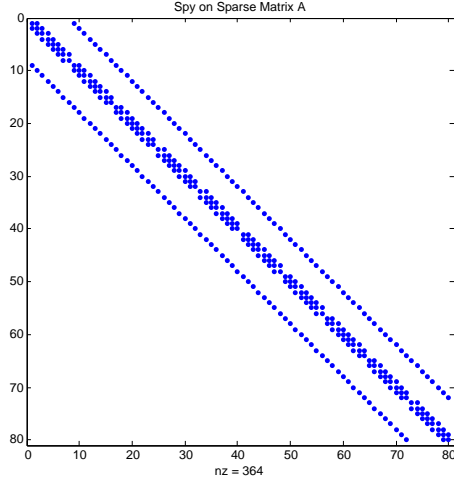
Figure 7: sparse FDFD matrix $\boldsymbol{A}$ for a specific frequency ω. Blue dots denote the nonzero elements.

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. Since the wave equations are independent over frequencies, FDFD can be easily parallelized across frequencies to achieve a better processing speed.

An interesting example 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 using FDTD, because the impulse function has components in arbitrarily high frequency. The impulse source function brings serve grid dispersion to the numerical solution. However this is not an issue at all for FDFD. The Green's Function G_ω in frequency domain, which solves

$$\frac{\omega^2}{c^2(z,x)}G(z,x;\omega) + \nabla^2 G(z,x;\omega) = \delta_\omega(z - z_0, x - x_0) \qquad (2.25)$$

is a the solution of (2.24) with the source term $\boldsymbol{f}$ contains all one's.
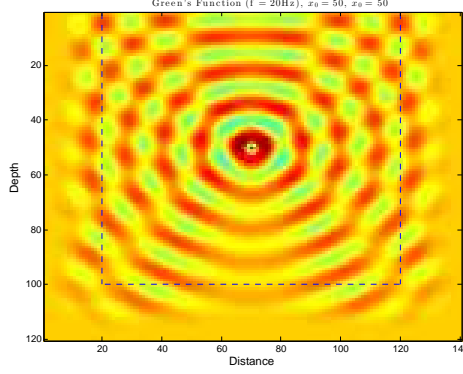
17

Figure 8: Real part of the Green's function at frequency 20 Hz for a region with PML

3 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. The particle velocity field satisfies

$$
\begin{cases}
v_z = v_{zp} + v_{zs}, & v_x = v_{xp} + v_{xs} \\
\dfrac{\partial v_{zp}}{\partial t} = \alpha^2 \dfrac{\partial A}{\partial z}, & \dfrac{\partial v_{xp}}{\partial t} = \alpha^2 \dfrac{\partial A}{\partial x} \\
\dfrac{\partial v_{zs}}{\partial t} = -\beta^2 \dfrac{\partial B}{\partial x}, & \dfrac{\partial v_{xs}}{\partial t} = \beta^2 \dfrac{\partial B}{\partial z} \\
A = \dfrac{\partial s_z}{\partial z} + \dfrac{\partial s_x}{\partial x}, & B = \dfrac{\partial s_x}{\partial z} - \dfrac{\partial s_z}{\partial x}
\end{cases}
\tag{3.1}
$$

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) is particle displacement vector. We can see that in this form the P-wave and S-wave fields are separated in the solution [4]. Thus a split PML (SPML) for the FDFD is adopted in S3I (see `src/fwdTimeSpmlFor2dEw.m`).

4 Migration

The goal of seismic imaging is to recover the subsurface structures with the recorded seismic data. This imaging process is typically computationally intense. Migration is one of the most important approaches. In the S3I we implemented a few common migration algorithms, such as Kirchhoff's
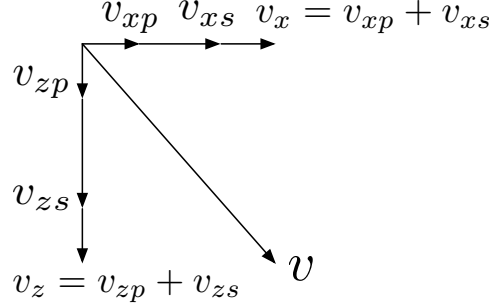
18

Figure 9: Velocity separation

migration, Reverse Time Migration (RTM), Least square RTM (LSRTM). All methods we implemented in the S3I are prestack.

4.1 Kirchhoff's Migration

Let $\boldsymbol{x}_r$ be the location of the receiver and $\boldsymbol{x}_s$ be the location of the source. Then $\boldsymbol{m} = (\boldsymbol{x}_r + \boldsymbol{x}_s)/2$ is the midpoint, and $\boldsymbol{h} = (\boldsymbol{x}_r - \boldsymbol{x}_s)/2$ is the half offset. The involved derivation of Kirchhoff's migration (see [24] for details) is omitted here. Only the simple imaging formula is given as below

$$I(\xi) = \int_{\Omega_\xi} W(\xi, \boldsymbol{m}, \boldsymbol{h}) dt(t = t_d(\xi, \boldsymbol{m}, \boldsymbol{h}), \boldsymbol{m}, \boldsymbol{h}) d\boldsymbol{m} d\boldsymbol{h}, \qquad (4.1)$$

where limited region Ω_ξ is centered around the location ξ in the m plane, called the migration aperture. $t_d = t_s[\boldsymbol{x}, \boldsymbol{s}, v(z, x, y)] + t_r[\boldsymbol{x}, \boldsymbol{g}, v(z, x, y)]$, where t_s is the travel time from the reflector ξ to $\boldsymbol{x}_s$ and t_r is the travel time from ξ to $\boldsymbol{x}_r$, see Figure 10. For reflector $\xi = (z_\xi, x_\xi, y_\xi)$ in 3D and $\xi = (z_\xi, x_\xi)$ in 2D case.
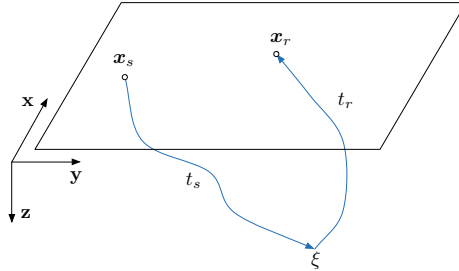

Figure 10: Reflector, and travel time

19
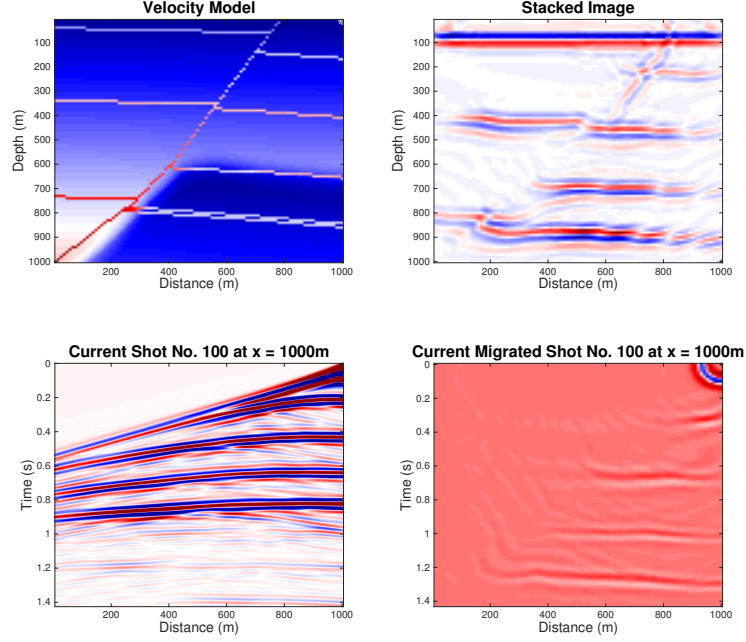
Figure 11: Kirkchhoff's migration result of the faults model

An example of Kirchhoff's migration (see Figure 11) for the fault model (available in the `modelData` folder) can be found in `mainKircCpmlFor2dAw_fault.m`, which is inherited from an existing package [15]. The most important part of Kirchhoff's migration is the computation of travel times t_s and t_r. These can be done either with `src/ray2d.m` from [15] or `src/eikonal2d.m` which solves the 2D Eikonal equation using a fast sweeping method [27].

The Eikonal equation is of the form

$$|\nabla T(\boldsymbol{x})|c(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in \mathbb{R}^2 \tag{4.2}$$

subject to $T(\boldsymbol{x}) = 0$ for a boundary Γ. In the S3I we consider the source point to be a single grid point $\boldsymbol{x}_s$. When we solve the Eikonal equation in a truncated region, $T(\boldsymbol{x})$ gives the time of first arrival in this region.

4.2 Reverse Time Migration (RTM)

The Reverse Time Migration (RTM) is a two-way wave-equation migration for accurate imaging in and below areas with complex subsurface structure. In the literature RTM has a relatively old origin, however it has not been used routinely until recently because of its computational intensity. In RTM a forward wave propagation and a reverse 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, e.g. handing evanescent energy and no dip limitation [19, 2]. In summary, RTM is implemented in three steps.

1. record forward wave field $p_f(\boldsymbol{x}, t; \boldsymbol{x}_s)$ through the true velocity model

2. solve the reverse wave field $p_r(\boldsymbol{x}, t)$ by the received surface data through the incident model, which is typically a smoothed version of the true model

3. combine the above using an imaging condition (chosen from a number existing imaging conditions)

Physically, there is no wave field propagates backward in time. It is only for the purpose of imaging. The second step above is equivalent to reverse the recorded traces in time then input those traces as source functions and solve a forward wave field.

For instance, an imaging condition using cross-correlation is

$$I(\boldsymbol{x}) = \sum_{\boldsymbol{x}_s} \frac{\int_0^T p_f(\boldsymbol{x}, t; \boldsymbol{x}_s) p_r(\boldsymbol{x}, t) \mathrm{d}t}{\int_0^T |p_f(\boldsymbol{x}, t; \boldsymbol{x}_s)|^2 \mathrm{d}t + \epsilon^2}. \tag{4.3}$$

Worth noting that using the imaging conditions (4.1), (4.3), and many other imaging conditions, the values $I(\boldsymbol{x})$ are not necessarily to be meaningful in the model domain. The image $I(\boldsymbol{x})$ only depicts the locations of the reflectors. Based on a migrated image, we may only qualitatively claim that the stronger model perturbation gives greater values in $I(\boldsymbol{x})$.

The RTM result of Marmousi model is illustrated with Figure 12. Moreover, the aggregating process across source shots for a fault model can be viewed in this video clip `https://www.youtube.com/watch?v=lL-UpW-5xms&feature=youtu.be`. The implementation of RTM is fulfilled using several methods in `mainRtmTimeCpmlFor2dAw.m`, `mainRtmTimeCpmlFor2dAwOpenMPI.m`, and `mainRtmTimeSpongeFor2dAw.m`.
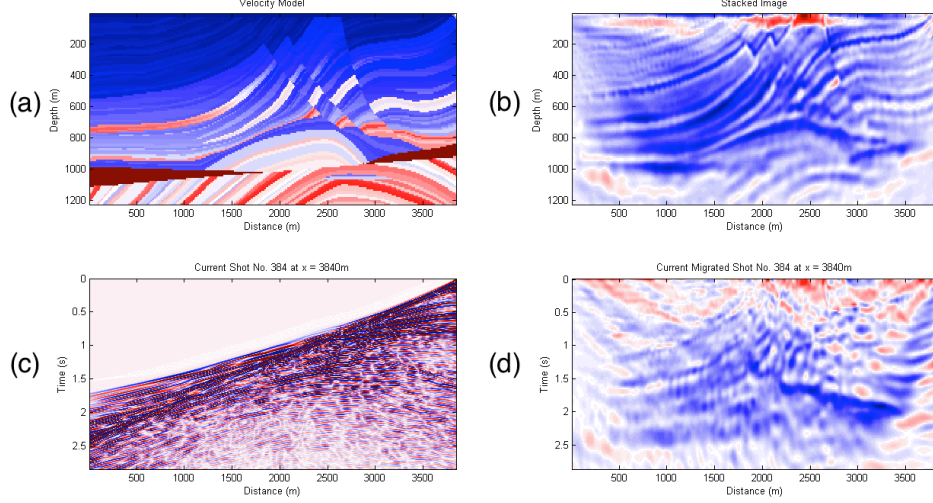
Figure 12: RTM results. (a) Marmousi model, (b) accumulated RTM result; (c) a sample shot record; (d) migrated result of the sample shot. The parameters settings are 15 Hz (center frequency) Ricker wavelet, 384 sensors, 384 shots, grid size 122×384, $\Delta z = \Delta x = 24$ m.

5 Least Square Reverse Time Migration (LS-RTM)

Generally speaking, RTM is a reliable method of imaging complex structures. However, there are still some distortions caused by RTM crosstalk artifacts. Mathematically, the migration operator is just the adjoint of the forward modeling operator, it does not minimize the mismatch of the observed and simulated traces. Least Square RTM (LSRTM) introduced by [20] derives an imaging condition by formulating migration as an inverse problem based on a least-squares misfit function. The implementation of LSRTM can either be in time domain or frequency domain[21].

Denote by $m(\boldsymbol{x}) = 1/c^2(\boldsymbol{x})$ the model which is the square slowness at point $\boldsymbol{x}$. We consider the decomposition of model m into a smooth model (initial guess) and a perturbation:

$$m(\boldsymbol{x}) = m_0(\boldsymbol{x}) + \delta m(\boldsymbol{x}), \tag{5.1}$$

where m_0 is the smooth model and perturbation $\delta m(\boldsymbol{x}) \ll m_0(\boldsymbol{x})$. Accordingly, we split the wave field $p(\boldsymbol{x}, t)$ into

$$p = p_0 + \delta p, \tag{5.2}$$

22

where p_0 solves the wave equation in the incident model $m_0(\boldsymbol{x})$. In light of the advantages we have mentioned above, we formulate the problem in frequency domain. For the LSRTM in time domain, one can use the iterative method proposed in [7]. Similarly, $P(\boldsymbol{x}; \boldsymbol{x}_s, \omega)$, the waveform generated by source at position $\boldsymbol{x}_s$, can also be split into two parts: the incident wavefield P_0 and δP. The $P_0(\boldsymbol{x}; \boldsymbol{x}_s, \omega)$ is determined by the smooth model and source function via the following acoustic wave equation:

$$\mathcal{L}(m_0)P_0(\boldsymbol{x}; \boldsymbol{x}_s, \omega) = F_s(\omega), \tag{5.3}$$

where operator $\mathcal{L}(m_0) = (-\omega^2 m_0(\boldsymbol{x}) - \Delta)$ and the $F_s(\omega)$ is the source function at source location $\boldsymbol{x}_s$. Using Born approximation [25], the scattered filed can be approximated by

$$\mathcal{L}(m_0)\delta P(\boldsymbol{x}; \boldsymbol{x}_s, \omega) = \omega^2 \delta m P_0(\boldsymbol{x}; \boldsymbol{x}_s, \omega). \tag{5.4}$$

Let $G(\boldsymbol{x}; \boldsymbol{x}_s, \omega)$ denote the Green's functions from the shot position $\boldsymbol{x}_s$ to a point in the model space $\boldsymbol{x}$, which is the solution of

$$\mathcal{L}(m_0)G(\boldsymbol{x}; \boldsymbol{x}_s, \omega) = \delta(\boldsymbol{x} - \boldsymbol{x}_s). \tag{5.5}$$

It is worth noting that the Green's function G only depends on the smooth model m_0. In least square migration, with an accurate smooth model, i.e. initial guess, we can calculate the Green's function and save it for the use of every iteration without updating.

Then the incident field u_0 at the surface point $\boldsymbol{x}$ becomes

$$P_0(\boldsymbol{x}; \boldsymbol{x}_s, \omega) = F_s(\omega)G(\boldsymbol{x}; \boldsymbol{x}_s, \omega). \tag{5.6}$$

And the secondary filed as the surface point $\boldsymbol{y}$ can be written as

$$\delta P(\boldsymbol{y}; \boldsymbol{x}_s, \omega) = \omega^2 \sum_{\boldsymbol{x}} \delta m(\boldsymbol{x})P_0(\boldsymbol{x}; \boldsymbol{x}_s, \omega)G(\boldsymbol{y}; \boldsymbol{x}, \omega). \tag{5.7}$$

For $\boldsymbol{y} = \boldsymbol{x}_r$, we obtain the Born approximation of the synthetic field at the receiver position $\boldsymbol{x}_r$. In general, the synthetic data for one frequency ω, a shot positioned at $\boldsymbol{x}_s$, and a receiver positioned at $\boldsymbol{x}_r$ can be given by a linear operator $L(\boldsymbol{x}_s, \boldsymbol{x}_r)$ acting on the model $m(\boldsymbol{x})$, as follows:

$$\delta P(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) = L(\boldsymbol{x}_s, \boldsymbol{x}_r)\delta \boldsymbol{m} = \omega^2 F_s(\omega) \sum_{\boldsymbol{x}} \delta m(\boldsymbol{x})G(\boldsymbol{x}; \boldsymbol{x}_s, \omega)G(\boldsymbol{x}_r; \boldsymbol{x}, \omega)$$

$$= \omega^2 [..., F_s(\omega)G(\boldsymbol{x}; \boldsymbol{x}_s, \omega)G(\boldsymbol{x}_r; \boldsymbol{x}, \omega), ...]\delta \boldsymbol{m}. \tag{5.8}$$

We can write the forward model operator as a $n_s n_r \times K$ (K is the number of grid points) matrix $\boldsymbol{L}$

$$\boldsymbol{L}(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) = \omega^2 [..., L(\boldsymbol{x}_s, \boldsymbol{x}_r)^*, ...]^*. \tag{5.9}$$

The cost function J is the ℓ_2 norm of the mismatch between the simulated and observed traces

$$
\begin{aligned}
J(\delta \boldsymbol{m}) &= \frac{1}{2} \sum_{\omega, \boldsymbol{x}_s, \boldsymbol{x}_r} \left| P^{(\text{sim})}(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) - P^{(\text{obs})}(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) \right|^2 \\
&= \frac{1}{2} \sum_{\omega, \boldsymbol{x}_s, \boldsymbol{x}_r} \left| \omega^2 F_s(\omega) \sum_{\boldsymbol{x}} \delta m(\boldsymbol{x}) G_\omega(\boldsymbol{x}; \boldsymbol{x}_s) G_\omega(\boldsymbol{x}_r; \boldsymbol{x}) \right. \\
&\quad \left. - \delta P^{(obs)}(\boldsymbol{x}_r, \boldsymbol{x}_s; \omega) \right|^2 \\
&= \frac{1}{2} \left\| \boldsymbol{L}\delta\boldsymbol{m} - \delta\boldsymbol{P}^{(obs)} \right\|_2^2,
\end{aligned}
\tag{5.10}
$$

where $\delta\boldsymbol{m}$ is the model perturbation (roughly saying, the reflectors), $\boldsymbol{L}$ is the forward modeling operator. The superscript (sim) means simulated results and (obs) means observed. The optimized model perturbation is

$$\delta\boldsymbol{m}^{(\text{opt})} = \left(\boldsymbol{L}^*\boldsymbol{L}\right)^{-1} \boldsymbol{L}^* \delta\boldsymbol{P}^{(obs)}. \tag{5.11}$$

Apparently, LSRTM is the simply a preconditioned RTM migration result $\boldsymbol{L}^* \delta\boldsymbol{P}_\omega^{(obs)}$. Instead of solving the huge inverse matrix $(\boldsymbol{L}^*\boldsymbol{L})^{-1}$, we use the Gauss-Newton method with a diagonal approximation of the Hessian matrix. In the case of large acquisition aperture, wide frequency band, and slow variation of the velocity model, the Hessian matrix is almost diagonal. In a simulation such as Marmousi model, the Hessian matrix is diagonal dominated.

Concretely, the gradient of the cost function is

$$\boldsymbol{g}_k(\delta\boldsymbol{m}) = \text{Re}\left(\sum_{\omega, \boldsymbol{x}_s, \boldsymbol{x}_r} \omega^2 \bar{F}_s(\omega) \bar{G}_\omega(\boldsymbol{x}; \boldsymbol{x}_s) \bar{G}_\omega(\boldsymbol{x}_r; \boldsymbol{x}) \Delta d(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) \right), \tag{5.12}$$

where $\Delta d(\boldsymbol{x}_r; \boldsymbol{x}_s, \omega) = \boldsymbol{L}\delta\boldsymbol{m} - \delta\boldsymbol{P}^{(obs)}$. As discussed in the preceding sections, the Hessian matrix can be approximated by its diagonal elements.

$$\boldsymbol{H}(\boldsymbol{m}) = \text{Re}\left(\sum_{\omega, \boldsymbol{x}_s, \boldsymbol{x}_r} \omega^4 |F_s(\omega)|^2 |G_\omega(\boldsymbol{x}; \boldsymbol{x}_s)|^2 |G_\omega(\boldsymbol{x}_r; \boldsymbol{x})|^2 \right). \tag{5.13}$$

Then

$$\delta\boldsymbol{m}^{k+1} = \delta\boldsymbol{m}^k - \boldsymbol{H}^{-1}\boldsymbol{g}_k(\delta\boldsymbol{m}^k), \tag{5.14}$$

24

where the initial value $\delta \boldsymbol{m}^0 = \boldsymbol{L}^* \delta \boldsymbol{P}$ can be obtained using RTM with an imaging condition that maps into the model domain. In the LSRTM, since we do not update the incident model $\boldsymbol{m}_0$, the Green's functions and Hessian only need to be computed once and stored for use in the later iteration. Its implementation can be easily modified with the source code for FWI in the next section.

6 Full Waveform Inversion (FWI) in Frequency Domain

The migration operator requires an incident model (usually a smoothed version of the true model) as an input. However, in complex media, building an accurate smooth model is challenging. In contrast, the starting model of FWI (see [26] as an overview) can be very rough, since in each iteration the updated model will be a new starting point for the following iteration. However, the Green's function should also be updated in order to converge fast. In the frequency domain, the formulation is similar to the LSRTM. The only thing different is we will update the incident model as well as the Green's functions over iterations. Applying the same approach, the cost function $J(\delta \boldsymbol{m})$ can be optimized by conjugate-gradient or quasi-Newton in the model domain or in a sparse domain using Projected Quasi-Newton (PQN) algorithm[23], see `mainAdjointFwiFreqCpmlFor2dAw.m` and `mainPQNFwiFreqCpmlFor2dAw.m`.

Algorithm 1 Full Waveform Inversion

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

For Marmousi model, the result of FWI is illustrated in Figure 13.
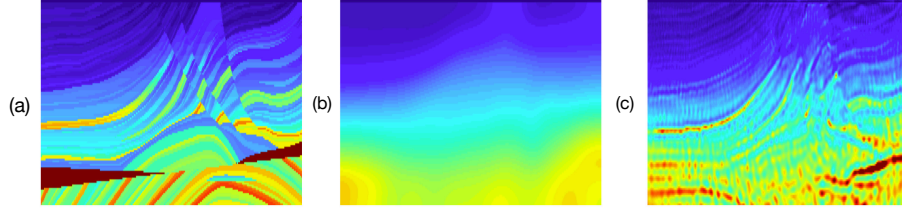
Acknowledgement

Figure 13: FWI result. (a) Marmousi model (b) starting model (c) FWI results after 15 iterations. The parameters settings are 15 Hz (center frequency) Ricker wavelet, 384 sensors, 384 shots, grid size 122×384, $\Delta z = \Delta x = 24$ m.

files for authorship and contact information. Particularly, we want to thank Stuart Kozola, the author of the package of Large Data in MATLAB: A Seismic Data Processing Case Study[15]. His excellent work motivates us to develop the S3I.

A   Comparison with Other Open Source Packages

Geophysicists have been working on developing seismic data processing and imaging packages for decades. There are a number of open source packages available online, such as Seismic Unix, SEPLib/SEPLib3d, Madagascar, and CREWES. In part because of this long history, many source code of these packages are written in C and Fortran 77. These code are computationally efficient but with poor readability and extensibility. A comparison in several aspects are illustrated in Figure 14. The S3I have the following advantages over the Linux/Unix based packages:

1. Rich functions and toolboxes in MATLAB, e.g. data visualization, data I/O, signal processing, and parallel computing.

2. Ease of coding and debugging for further development.

3. Cross-platform: Windows/Linux/Unix.

4. Easy to use: GUI, simple MATLAB script manipulations, no shell scripting.

5. Available code reuse: data formatting, spare transforms, advanced optimization algorithms.

	Languages	OS	Interface	Data Dimension
S3I	MATLAB, C	Windows/Unix/ Linux	Matlab script and GUI	2D and 3D (partial function)
Seismic Unix	C, Fortran	Unix/Linux	Command Line	2D
SEPLib/SEPLib3d	C, Fortran	Unix/Linux	Command Line	2D/3D
Madagascar	C, Python	Unix/Linux	Command Line	2D
CREWES	MATLAB	Windows/Unix/ Linux	Matlab script and GUI	2D and 3D (partial function)

Figure 14: Comparison of S3I with other open source packages

Compared with CREWES, which is also based on MATLAB and contains many functions, S3I targets on more specific goals. As its name would suggest, the S3I contains more organized and optimized source code to implement large scale parallel computing and provides a tutorial of the numerical methods in the user guide. It is well known the computing efficiency of MATLAB is not as good as C and Fortran, especially when the code contains a large number of for-loops. Thus, some slow but frequently called functions in S3I are implemented with C MEX-files which significantly speed up the simulation.

B   List of Source Code

The appendix provides a list of source code in `./src` and briefly explains their uses.
D
`dampPml.m`   Generates the model for damping parameter for PML
`dCoef.m`   Computes the coefficients of the finite difference using staggered-grid

E
`eikonal2d.m`   Computes the travel time by solving the Eikonal equation in 2D rectangular domain for a single source
`extBoundary.m`   Performs the zero padding around the given 2D or 3D model

F
`./fd_cmex`   source files in C language for finite difference. Run `make.m` to

compile MEX-files for current system architecture

`./fd_matlab` source files in Matlab for finite difference, which is not used but only a reference in default

`fctForAw.m` Performs the FCT method to alleviate numerical grid dispersion for acoustic wave

`fctForEw.m` Performs the FCT method to alleviate numerical grid dispersion for elastic wave

`freqSolver.m` Solves the 2D acoustic wave equation in frequency domain with Enguist-Majda absorbing boundary conditions

`freqSolverCpml.m` Solves the 2D acoustic wave equation in frequency domain with Nonsplit Convolutional-PML

`fwdTimeCpmlFor3dAw.m` Simulates 3D acoustic wave forward propagation in time domain with Convolutional-PML

`fwdTimeSpmlFor2dEw.m` Simulates 2D elastic wave forward propagation in time domain with Split-PML

`fwdTimeSpongeFor2dAw.m` Simulates 2D acoustic wave forward propagation in time domain with sponge absorbing boundary conditions

L

`layerModelCenerator.m` Generates 2D or 3D layered model

M

`migrate.m` Migrates a shot record for a given travel time between source and receiver using a Kirchoff migration algorithm

`misfitFuncModel.m` Calculate the least-square misfit function with respect to the perturbation model $\delta\boldsymbol{m}$. Used in FWI

P

`./PQN` the Projected Quasi-Newton (PQN) package for optimization (thanks to Mark Schmidt, Ewout van den Berg, Michael Friedlander, Kevin Murphy from University of British Columbia, Canada)

`plotTrace.m` Plots seismic traces with filled wiggles

R

`ray2d.m` Computes the travel time for 2D model and single point source

`ricker.m` Generates Ricker wavelet for the source time function

`rvsTimeSpongeFor2dAw.m` Simulates wave propagation in reverse time with sponge absorbing boundary conditions

S

`seismic.m`   Defines the color space for plotting seismic data

`shot2RecTime.m`   Computes the travel time from a source to a receiver for given travel time map

W

`waveletGenerator.m`   Generates different wavelet as candidates of the source time function

`wiggle.m`   displays seismic data as wiggle plus filled lobes

References

[1] R. M. Alford, K. R. Kelly, and D. M. Boore, *Accuracy of finite-difference modeling of the acoustic wave equation*, Geophysics **39** (1974), 834–842.

[2] E. Baysal, D. D. Kosloff, and J. W. C. Sherwood, *Reverse time migration*, Geophysics **48** (1983), 1514–1524.

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

[7] 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 Techinical Program Expanded Abstract 2012 (Las Vegas), SEG, 2012, pp. 1–5.

[8] B. Engquist and A. Majda, *Absorbing boundary conditions for numerical simulation of waves*, Proceedings of the Natinoal Academy of Sciences **74** (1977), 1765–1766.

[9] 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.

[10] B. Fornberg, *On a fourier method for the integration of hyperbolic equations*, Soc. Industr. Appl. Math. J. Numer. Anal. **12** (1975), 509–528.

[11] _____, *The pseudospectral method: Comparisions with finite differences for the elastic wave equation*, Geophysics **52** (1987), no. 4, 483–501.

[12] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall, *Open MPI: Goals, concept, and design of a next generation MPI implementation*, Proceedings, 11th European PVM/MPI Users' Group Meeting (Budapest, Hungary), September 2004, pp. 97–104.

[13] P. Hood, *Developments in geophysical methods*, ch. Migration, Applied Science, London, 1981.

[14] 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.

[15] S. Kozola, *Large data in matlab: A seismic data processing case study*, March 2011.

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

[17] J. M. Kurt, *Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations*, Geophysics **49** (1984), 533–549.

[18] R. J. Luebbers and F. Hunsberger, *Fdtd for nth-order dispersive media*, IEEE Transactions on Antennas and Propagation **40** (1992), 1297–1301.

[19] G. A. McMechan, *Migration by extrapolation of time-dependent boundary values*, Geophysical Prospecting **31** (1983), 413–420.

[20] T. C. Nemeth, C. Wu, and G. T. Schuster, *Least-square migration of incomplete reflection data*, Geophysics **64** (1999), 208–221.

[21] H. Ren, H. Wang, and S. Chen, *Least-squares reverse time migratio in frequency domain using the adjoint-state method*, J. Geophys. Eng. **10** (2013), 1–7.

[22] 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.

[23] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy, *Optimizing costly functions with simple constraints: A limited-memory porjected quasi-newton algorithm*, AISTATS'09, 2009, pp. 456–463.

[24] W. Schneider, *Integral formulation for migration in two and three dimensionas*, Geophysics **43** (1978), 49–76.

[25] A. Tarantola, *Theoretical background for the inversion of seismic waveforms, including elasticity and attenuation*, Pure and Applied Geophysics **128** (1988), 365–399.

[26] J. Virieux and S. Operto, *An overivew of full-waveform inversion in exploration geophysics*, Geophysics **74** (2009), no. 6, WCC1–WCC26.

[27] H. Zhao, *A fast sweeping method for eikonal equations*, Mathematics of computation **74** (2004), no. 250, 603–627.