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

Lingchen Zhu*, Entao Liu† and Lijun Zhu‡

February 10, 2015

Contents

*Center for Signal and Information Processing (CSIP), School of Electrical and Computer Engineering, Georgia Institute of Technology, E-mail: lczhu@gatech.edu

†Center for Energy & Geo Processing (CeGP), School of Electrical and Computer Engineering, Georgia Institute of Technology, E-mail: entao.liu@ece.gatech.edu

‡Center for Energy & Geo Processing (CeGP), School of Electrical and Computer Engineering, Georgia Institute of Technology, E-mail: lijun.zhu@gatech.edu

0 Requirements and License

The 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 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 program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY. If you find any glitches within the package, please contact the authors. We appreciate you contributions.

1 Introduction

The SSSI is designed to provide a package for numerical exploration geophysics. The purpose of package is more about providing a package for the interested users to learn some popular algorithms and numerical schemes in exploration geophysics than the high performance of the computation as commercial software. Thus, we choose the Matlab as the coding platform for its readability of the codes, ease of data visualization, etc. It is well known that nested for-loops on Matlab is comparatively slow. In order to have a better efficiency, we use C to generate mex files for some frequently invoked functions. Currently, the major functions has been of the package are the following:

- Acoustic wave simulation for 2-D/3-D
- Elastic wave simulation for 2-D
- Kirchhoff's migration
- Reverse Time Migration (RTM)
- Full Waveform Inversion (FWI)

An exhaustive review of numerical simulation of wave equation and seismic imaging is out of the scope of this user guide. 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 the contents concise. Interested users are suggested to read the references therein for details.

2 Numerical Simulation of Acoustic Wave

2.1 Acoustics wave equation

The seismic method is one of the basic tools in exploration geophysics. In seismic survey, man-made vibration sources (e.g. dynamite, viborseis trucks, and air gun) are fired off. Then the wave propagates through the subsurface media. The wave filed which contains direct wave, reflection, and refraction, will be recorded by array of receivers (e.g. geophones and hydrophones). In order to simulate this process with computers, it is crucial to solve the wave propagation numerically in the complex media accurately and efficiently. 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 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 [10]. Therefore, it is reasonable to assume a constant density of the media. Then the acoustic wave equation is of the the form

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

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 2-D and 3-D 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 $\boldsymbol{x}$. The $f(\boldsymbol{x},t)$ is the source term. Moreover, Δ in (2.1) is the Laplace operator defined as

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

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

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

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`.

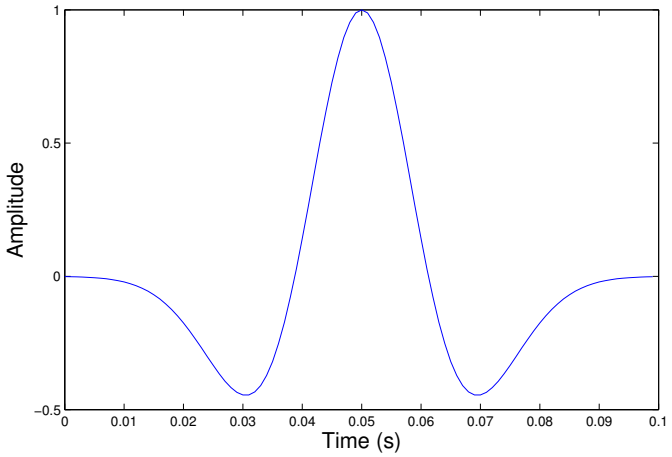


Figure 1: Ricker wavelet with peak frequency $\mathsf{f} = 20\text{Hz}$

2.2 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 and derivatives are approximated through difference in these values. For instance, in a 2-D region we use the equi-distributed grid points $(z_i)_{0\le i\le I-1}$, $(x_j)_{0\le j\le J-1}$, and $(t_n)_{0\le n\le N-1}$ 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 (2.1) with function values on the grid points. Let us

begin with the definition of the derivative,

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

Assuming the finite grid size Δz is small enough, $(p(z+\Delta z,x,t)-p(z,x,t))/\Delta z$ is an accurate estimation of the derivative. Apply this approximation twice. We can derive an estimation of the 2nd 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 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

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

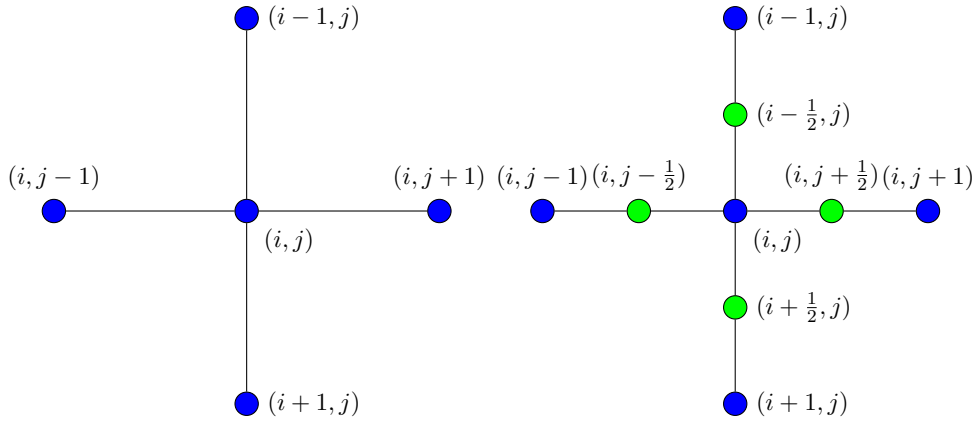
Simple algebraic manipulations lead to a recursive expression of the wave equation,

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

In (2.7), 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.

2.3 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.



(a) Standard Grid (b) Staggerd Grid

Figure 2: Discretization Grids

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

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

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

where $u=z,x;\ k=0,1,2,\ldots$. Take the difference between (2.8) and (2.9) which cancels all the terms when n is even. So it implies for $k=0,1,2,\ldots$

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

We can approximate $\frac{\partial p(u)}{\partial u}$ as a weighted sum of finite differences based on

(2.10)

$$\frac{\partial p(u)}{\partial u} = \sum_{k=0}^{N-1} a_k \frac{p\left(u + \frac{2k+1}{2}\Delta u\right) - p\left(u - \frac{2k+1}{2}\Delta u\right)}{(2k+1)\Delta u}$$

$$= \sum_{k=0}^{N-1} a_k \left[\frac{\partial p(u)}{\partial u} + \frac{\Delta u^2}{3! \cdot 2^2}(2k+1)^2 \frac{\partial^3 p(u)}{\partial u^3} \right. \tag{2.11}$$

$$+ \frac{\Delta u^4}{5! \cdot 2^4}(2k+1)^4 \frac{\partial^5 p(u)}{\partial u^5} + \cdots$$

$$\left. + \frac{\Delta u^{2N-2}}{(2N-1)! \cdot 2^{2N-2}}(2k+1)^{2N-2} \frac{\partial^{2N-1} p(u)}{\partial u^{2N-1}} + o(\Delta u^{2N}) \right].$$

If the weighted $\{a_k\}_{k=0}^{N-1}$ are chosen carefully, we can eliminate all the term on the right hand side of (2.11) 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 gird points, and $p(u)$ on integer grid points.

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

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

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

The source code which generates the appropriate weights can be found at `src/dCoef.m`. 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` generates 6-th order of approximation of the spatial derivatives.

2.4 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 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 (2.7), 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 2-D 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 outside 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. Therefore, no absorbing boundary added to it.

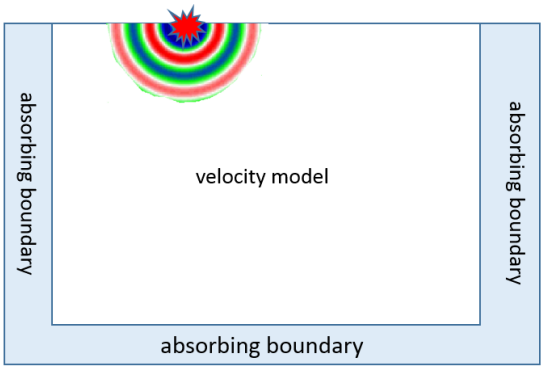


Figure 3: Absorbing boundaries of the 2-D simulation region

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 $d(u) < 1$.

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

where $\mathrm{dist}(u)$ 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-D (`For3d` means 3-D); and `Aw` is short for acoustic wave (`Ew` is for elastic wave). Although the waves are attenuated by ABC, however the reflection can not be completely eliminated. This is the reason in SSSI we utilize the following method called Perfectly Matched Layer as well.

2.4.1 Perfectly Matched Layer (PML)

Perfectly Matched Layer (PML) method which was originally formulated for use with electromagnetic equations which outperforms other existing methods. The PML is proven to be efficient for wave equations (both acoustic and elastic) as well [11]. 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

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

Or equivalently,

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

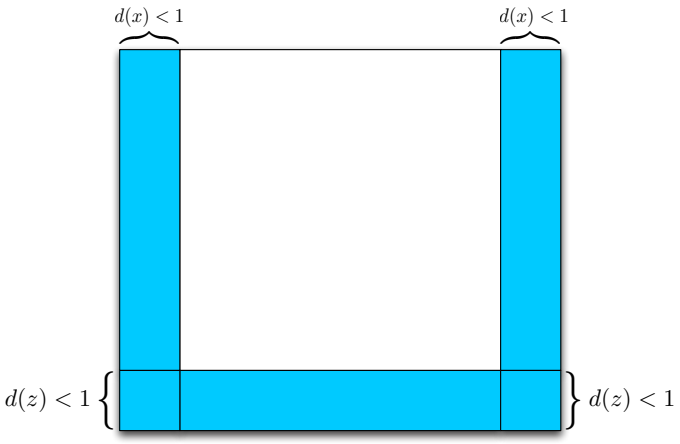


Figure 4: The damping profile $d(u)$

In the homogeneous media, the acoustic wave equation has the following 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 position vector. After the replacement in (2.13), we can derive another solution of the acoustic equation as follows,

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

In the original simulated region the the new solution is equivalent to the old one, since $d_x=0$. Additionally, in the $\hat{\mathbf{n}}=\hat{\mathbf{x}}$ 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) [13, 16, 11], which is more reasonable to work with the acoustic (pressure) source. In time domain,

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

The convolution can be performed as follows,

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

and

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

The implemention 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.

2.5 Parallel Computing using OpenMPI

Parallel computing based on the distributed memory model is supported in SSSI. 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.

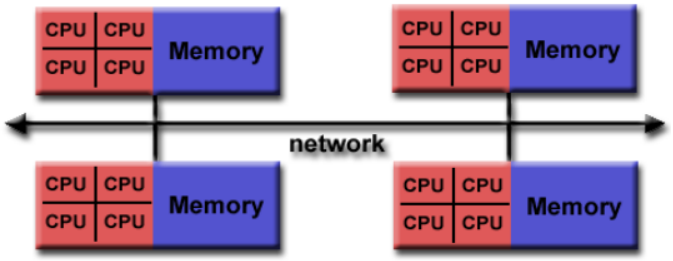
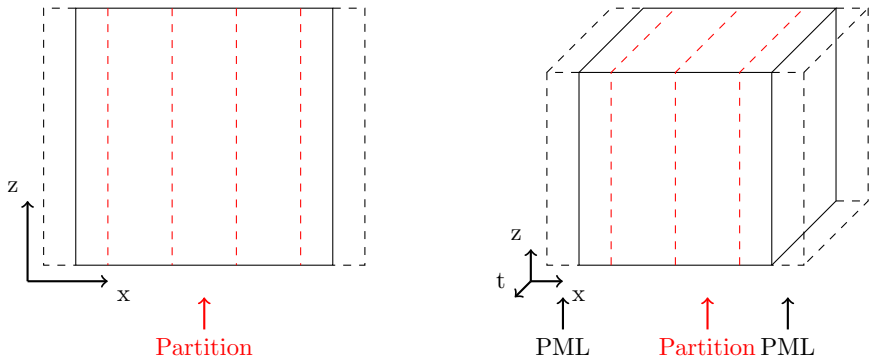


Figure 5: Distributed memory model of a computer cluster

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. SSSI uses OpenMPI[9] 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.

In SSSI, the acoustic wave propagation process can be simulated in such a parallel manner. Suppose there are P nodes, each node p ($p = 0, 1, \ldots, P-$



(a) Standard Grid (b) Staggerd Grid

Figure 6: Workload Division

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,

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

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 `test_mpi_FwdWaveTimeCpmlFor2dAw.m` sets up simulation parameters. One can call

```
mpirun -n <NP> matlab -nodisplay -r
"test_mpi_FwdWaveTimeCpmlFor2dAw"
```

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 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 harmful grid dispersion the following criteria for the spatial grid spacing Δu has to be satisfied

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

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`

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 grid spacing Δu. In the 2-D scenario,

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

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, 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 simulations 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.

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

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

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

$$\begin{aligned}\frac{\omega^2}{c^2_{i,j}}P^{(\omega)}_{i,j} + &\left[\frac{P^{(\omega)}_{i-1,j} - 2P^{(\omega)}_{i,j} + P^{(\omega)}_{i+1,j}}{\Delta z^2}\right] \\ &+ \left[\frac{P^{(\omega)}_{i,j-1} - 2P^{(\omega)}_{i,j} + P^{(\omega)}_{i,j+1}}{\Delta x^2}\right] = -F^{(\omega)}_{i,j}.\end{aligned} \tag{2.23}$$

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

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

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

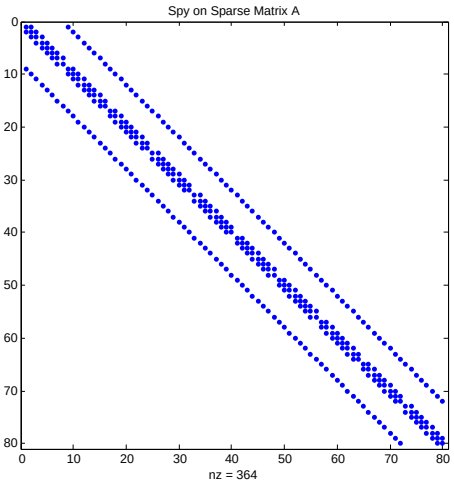


Figure 7: sparse FDFD matrix $\boldsymbol{A}$ for a specific frequency ω. Blue dots denotes the nonzero elements.

In order to solve acoustic wave equation in frequency ω, we only have to solve the matrix equation (2.24). 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 $\boldsymbol{A}$ has 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 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 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 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

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

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

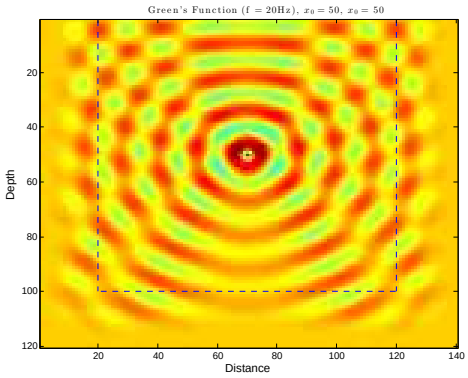


Figure 8: Real part of the Green's function at frequency 20 Hz

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

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

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): 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`.

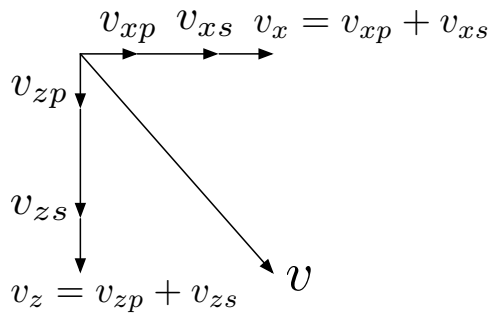


Figure 9: Velocity separation

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 SSSI we implemented a few representative migration algorithms, such as Kirchhoff's migration, Reverse Time Migration (RTM), Least square RTM (LSRTM). All methods we implemented in SSSI 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 [17] for details) is omitted here. Only the simple imaging formula is given as below

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

where limited region Ω_ξ is centered around the location ξ in the m plane, called the migration aperture. $t_d = t_s[\boldsymbol{x}, \boldsymbol{s}, v(z,x,y)] + t_g[\boldsymbol{x}, \boldsymbol{g}, v(z,x,y)]$. For reflector $\xi = (z_\xi, x_\xi, y_\xi)$ in 3-D and $\xi = (z_\xi, x_\xi)$ in 2-D 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 a existing package [12]. 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 [12] or `src/eikonal2d.m` which solves the 2-D Eikonal equation using a fast sweeping method [19]. The

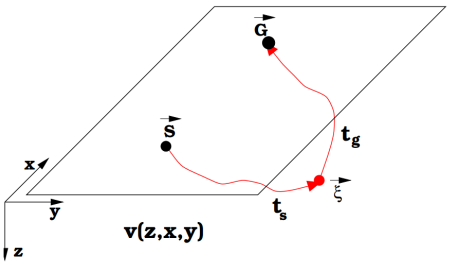


Figure 10: Reflector, and travel times (adapted from [2])

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 SSSI 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. 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 handing evanescent energy and no dip limitation [14, 1]. 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. superposition of above using an imaging condition (chosen from a number existing imaging conditions)

The second step is equivalent to reverse the recorded traces in time. Then input those traces as source function and solve the wave field.

For instance, an imaging condition with 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 noticing that, using the imaging conditions (4.1), (4.3), and many other imaging conditions, the values $I(\boldsymbol{x})$ are not necessarily to be in the model domain. It 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 the image.

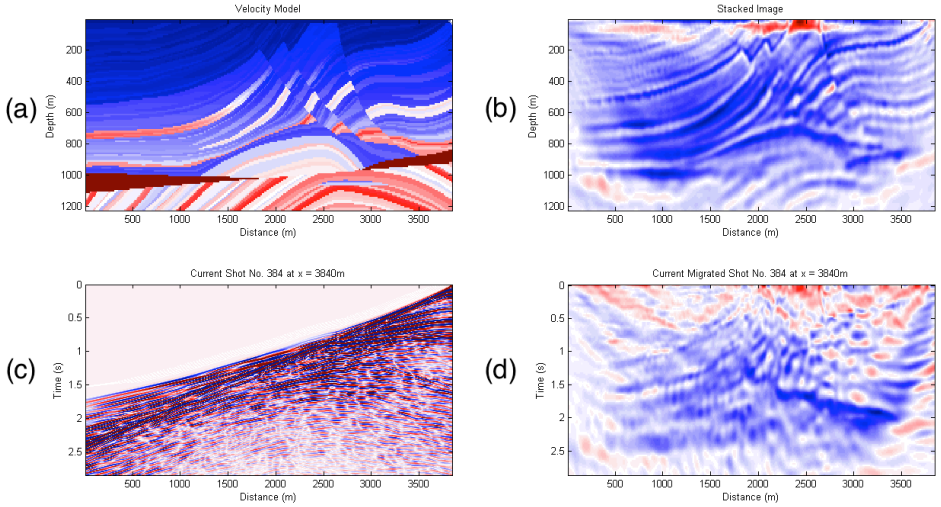


Figure 11: RTM results. (a) Marmousi model, (b) accumulated RTM result; (c) a sample shot record; (d) migrated result of the sample shot. The parameters settings are 15 Hz (center frequency) Ricker wavelet, 384 sensors, 384 shots, grid size 122×384, $\Delta z = \Delta x = 24m$.

The RTM result of Marmousi model is illustrated with Figure 11. In order to show the aggregating process across source shots, another result for faults model can be viewed in this video clip `https://www.youtube.com/watch?v=lL-UpW-5xms&feature=youtu.be`.

5 Least Square Reverse Time Migration (LS-RTM)

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 [15] 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 time domain or frequency domain.

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

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

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

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

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 in [6]). Accordingly, $P(\boldsymbol{x}_s, \boldsymbol{x}; \omega)$, 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 $P_0(\boldsymbol{x}_s, \boldsymbol{x}; \omega)$ is determined by the smooth model and source function via the following acoustic wave equation:

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

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

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

Denote $G(\boldsymbol{x}_s, \boldsymbol{x}; \omega)$ as 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

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

It is worth noting that the Green's function G is only dependent on the smooth model m_0. In least square migration, with an accurate smooth model, i.e. starting model, 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

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

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

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

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:

$$\begin{aligned} \delta P(\boldsymbol{x}_r, \boldsymbol{x}_s; \omega) &= L(\boldsymbol{x}_s, \boldsymbol{x}_r) \delta \boldsymbol{m} = \omega^2 F_s(\omega) \sum_{\boldsymbol{x}} \delta m(\boldsymbol{x}) G(\boldsymbol{x}_s, \boldsymbol{x}; \omega) G(\boldsymbol{x}, \boldsymbol{x}_r; \omega) \\ &= \omega^2 [..., F_s(\omega) G(\boldsymbol{x}_s, \boldsymbol{x}; \omega) G(\boldsymbol{x}, \boldsymbol{x}_r; \omega), ...] \delta \boldsymbol{m} \end{aligned} \tag{5.8}$$

We can write the forward model operator as a $n_s n_r \times K$ matrix $\boldsymbol{L}$

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

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

where δm is the module 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

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

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 diagonal. In a simulation such as Marmousi model, the Hessian matrix is diagonal dominated.

The gradient is

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

where $\Delta d(\boldsymbol{x}_s,\boldsymbol{x}_r;\omega) = \boldsymbol{L}\delta\boldsymbol{m} - \delta\boldsymbol{P}^{(obs)}$.

As discussed in the preceding sections, the Hessian matrix can be approximated by its diagonal elements.

$$\boldsymbol{H}(\boldsymbol{m}) = \mathrm{Re}\Big(\sum_{\omega,\boldsymbol{x}_s,\boldsymbol{x}_r} \omega^4 |F_s(\omega)|^2 |G(\boldsymbol{x}_s,\boldsymbol{x};\omega)|^2 |G(\boldsymbol{x},\boldsymbol{x}_r;\omega)|^2\Big) \quad (5.13)$$

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

where the initial value $\delta\boldsymbol{m}^0$ can be obtained using RTM $\boldsymbol{L}^*\delta\boldsymbol{P}$ 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.

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. In complex media, building an accurate smooth model is challenging. In contrast, the starting model of FWI (see [18] as an overview) can be very rough, since in each iteration the updated model will be 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 more approaches: conjugate-gradient, quasi-Newton, etc.

Algorithm 1 Full Waveform Inversion

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

Acknowledgement

Some of SSSI source files are inherited from other open source package. We are grateful for those contributions. Please read the comments in codes for authorship and contact information. Particularly, we want to thank Stuart Kozola[12], the author of the package of Large Data in MATLAB: A Seismic Data Processing Case Study. His excellent work motivates us to develop SSSI. We have tried very hard to find However, some source files are lack of authorship.

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*, Inverstigations 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 Techinical 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 Natinoal 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] 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.

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

[11] 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.

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

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

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

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

[16] 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.

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

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

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