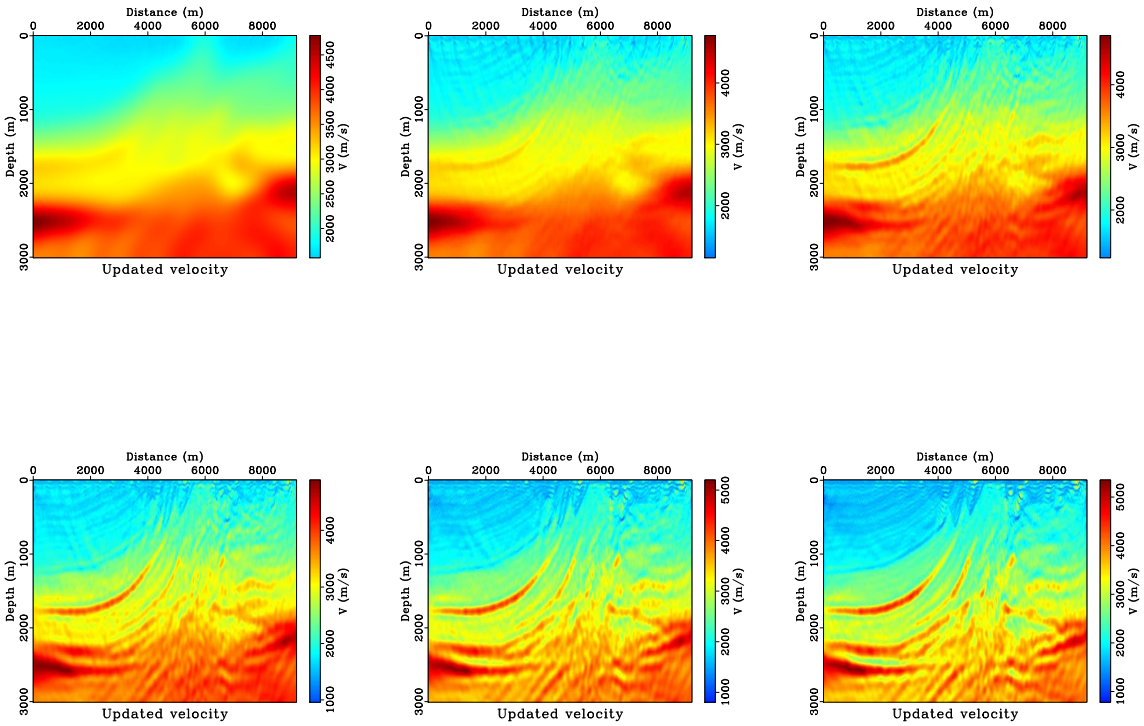
NATIONAL ENGINEERING LABORATORY FOR OFFSHORE OIL EXPLORATION

Sergey Fomel, Jinghuai Gao, Baoli Wang, and Pengliang Yang



XJTU — TABLE OF CONTENTS

RTM using effective boundary saving: A staggered grid GPU implementation

Pengliang Yang, Jinghuai Gao*, and Baoli Wang†*
**Xi'an Jiaotong University, National Engineering Laboratory for Offshore Oil Exploration, Xi'an, China, 710049*
†CCTEG Xi'an Research Institute, Xi'an, China, 710077

ABSTRACT

GPU has become a booming technology in reverse time migration (RTM) to perform the intensive computation. Compared with saving forward modeled wavefield on the disk, RTM via wavefield reconstruction using saved boundaries on device is a more efficient method because computation is much faster than CPU-GPU data transfer. In this paper, we introduce the effective boundary saving strategy in backward reconstruction for RTM. The minimum storage requirement for regular and staggered grid finite difference is determined for perfect reconstruction of the source wavefield. Particularly, we implement RTM using GPU programming, combining staggered finite difference scheme with convolutional perfectly matched layer (CPML) boundary condition. We demonstrate the validity of the proposed approach and CUDA codes with numerical example and imaging of benchmark models.

INTRODUCTION

One-way equation based imaging techniques are inadequate to obtain accurate images in complex media due to propagation direction changes in the background model (Biondi, 2006). These approaches are extremely limited when handling the problems of turning waves in the model containing sharp wave-speed contrasts and steeply dipping reflectors. As an advanced imaging technology without dip and extreme lateral velocity limitation, reverse time migration (RTM) was proposed early (Baysal et al., 1983; McMechan, 1983), but not practical in terms of stringent computation and memory requirement. However, it gained increasingly attention in recent years due to the tremendous advances in computer capability. Until recently, 3D prestack RTM is now feasible to obtain high fidelity images (Yoon et al., 2003; Guitton et al., 2006).

Nowadays, graphics processing unit (GPU) is a booming technology, widely used to mitigate the computational drawbacks in seismic imaging and inversion, from one-way depth migration (Liu et al., 2012b; Lin and Wang, 2012) to two-way RTM (Hussain et al., 2011; Micikevicius, 2009; Clapp et al., 2010), from 2D to 3D (Micikevicius,

2009; Abdelkhalek et al., 2009; Foltinek et al., 2009; Liu et al., 2013a; Michéa and Komatitsch, 2010), from acoustic media to elastic media (Weiss and Shragge, 2013), from isotropic media to anisotropy (Guo et al., 2013; Suh and Wang, 2011; Liu et al., 2009). The investigators have studied many approaches: the Fourier integral method (Liu et al., 2012c), spectral element method (Komatitsch et al., 2010b), finite element method (Komatitsch et al., 2010a) as well as the rapid expansion method (REM) with pseudo-spectral approach (Kim et al., 2013). A variety of applications were conducted, for instance, GPU-based RTM denoising (Ying et al., 2013), iterative velocity model building (Ji et al., 2012), multi-source RTM (Boonyasiriwat et al., 2010), as well as least-square RTM (Leader and Clapp, 2012).

The superior speedup performance of GPU-based imaging and inversion has been demonstrated by numerous studies. One key problem of GPU-based RTM is that the computation is much faster while the data exchange between host and device always takes longer time. Many researchers choose to reconstruct the source wavefield instead of storing the modeling time history on the disk, just saving the boundaries. Unlike most GPU-based imaging and inversion studies, this paper is devoted to the practical technical issues instead of speedup performance. Starting from the computational strategies by Dussaud et al. (2008), we determine the minimum storage requirement in backward wavefield reconstruction for regular and staggered grid finite difference. We implement RTM with staggered finite difference scheme combined with convolutional perfectly matched layer (CPML) boundary condition using GPU programming. We demonstrate the validity of the proposed approach and CUDA codes with numerical test and imaging of benchmark models.

OVERVIEW OF RTM AND ITS COMPUTATION

In the case of constant density, the acoustic wave equation is written as

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = \nabla^2 p(\mathbf{x},t;\mathbf{x}_s) + f(t)\delta(\mathbf{x}-\mathbf{x}_s), \tag{1}$$

where $p(\mathbf{x},t;\mathbf{x}_s)$ is the wavefield excited by the source at the position $\mathbf{x}=\mathbf{x}_s$, $v(\mathbf{x})$ stands for the velocity in the media, $\nabla^2 = \nabla\cdot\nabla = \partial_{xx}+\partial_{zz}$, $f(t)$ denotes the source signature. For the convenience, we eliminate the source term hereafter and use the notation $\partial_u = \frac{\partial}{\partial u}$ and $\partial_{uu} = \frac{\partial}{\partial u^2}$, $u = x, z$. The forward marching step can be specified after discretization as

$$p^{k+1} = 2p^k - p^{k-1} + v^2\Delta t^2\nabla^2 p^k. \tag{2}$$

Based on the wave equation, the principle of RTM imaging can be interpreted as the cross-correlation of two wavefields at the same time level, one computed by forward time recursion, the other computed by backward time stepping (Symes, 2007). Mathematically, the cross-correlation imaging condition can be expressed as

$$I(\mathbf{x}) = \sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x},t;\mathbf{x}_s)p_g(\mathbf{x},t;\mathbf{x}_g), \tag{3}$$

where $I(\mathbf{x})$ is the migrated image at point $\mathbf{x}$; and $p_s(\cdot)$ and $p_g(\cdot)$ are the source wavefield and receiver (or geophone) wavefield. The normalized cross-correlation imaging condition is designed by incorporating illumination compensation:

$$I(\mathbf{x}) = \sum_{s=1}^{ns} \frac{\int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x}, t; \mathbf{x}_s) p_g(\mathbf{x}, t; \mathbf{x}_g)}{\int_0^{t_{\max}} \mathrm{d}t p_s(\mathbf{x}, t; \mathbf{x}_s) p_s(\mathbf{x}, t; \mathbf{x}_s)}. \tag{4}$$

There are some possible ways to do RTM computation. The simplest one may be just storing the forward modeled wavefields on the disk, and reading them for imaging condition in the backward propagation steps. This approach requires frequent disk I/O and has been replaced by wavefield reconstruction method. The so-called wavefield reconstruction method is a way to recover the wavefield via backward reconstructing or forward remodeling, using the saved wavefield snaps and boundaries. It is of special value for GPU computing because saving the data in device variables eliminates data transfer between CPU and GPU. By saving the last two wavefield snaps and the boundaries, one can reconstruct the wavefield of every time step, in time-reversal order. The checkpointing technique becomes very useful to further reduce the storage (Symes, 2007; Dussaud et al., 2008). Of course, it is also possible to avert the issue of boundary saving by applying the random boundary condition, which may bring some noises in the migrated image (Clapp, 2009; Clapp et al., 2010; Liu et al., 2013b,a).

EFFECTIVE BOUNDARY SAVING

Here we mainly focus on finding the part of boundaries which is really necessary to be saved (referred to as the effective boundary in this paper), even though there are many other practical implementation issues in GPU-based RTM (Liu et al., 2012a). In what follows, we introduce the effective boundary saving for regular grid and staggered grid finite difference. All analysis will be based on 2D acoustic wave propagation in RTM. In other cases, the wave equation may change but the principle of effective boundary saving remains the same.

Which part of the wavefield should be saved?

To reconstruct the modeled source wavefield in backward steps rather than read the stored history from the disk, one can reuse the same template by exchanging the role of p^{k+1} and p^{k-1}, that is,

$$p^{k-1} = 2p^k - p^{k+1} + v^2 \Delta t^2 \nabla^2 p^k. \tag{5}$$

We conduct the modeling (and the backward propagation in the same way due to template reuse):

$$\begin{aligned} &for\ ix, iz... \quad p_0(:) = 2p_1(:) - p_0(:) + v^2(:)\Delta t^2 \nabla^2 p_1(:) \\ &ptr = p_0; p_0 = p_1; p_1 = ptr; //\text{exchange pointer} \end{aligned}$$

where $(:) = [ix, iz]$, p_0 and p_1 are p^{k+1}/p^{k-1} and p^k, respectively. When the modeling is finished, only the last two wave snaps (p^{nt} and p^{nt-1}) as well as the saved boundaries are required to do the backward time recursion.

As you see, RTM begs for an accurate reconstruction before applying the imaging condition using the backward propagated wavefield. The velocity model is typically extended with sponge absorbing boundary condition (ABC) (Cerjan et al., 1985) or PML and its variants (Komatitsch and Martin, 2007) to a larger size. In Figure 1, the original model size $A_1A_2A_3A_4$ is extended to $C_1C_2C_3C_4$. In between is the artificial boundary ($C_1C_2C_3C_4 \backslash A_1A_2A_3A_4$). Actually, the wavefield we intend to reconstruct is not the part in extended artificial boundary $C_1C_2C_3C_4 \backslash A_1A_2A_3A_4$ but the part in the original model zone $A_1A_2A_3A_4$. We can reduce the boundary load further (from whole $C_1C_2C_3C_4 \backslash A_1A_2A_3A_4$ to part of it $B_1B_2B_3B_4$) depending on the required grids in finite difference scheme, as long as we can maintain the correctness of wavefield in $A_1A_2A_3A_4$. We do not care about the correctness of the wavefield neither in $A_1A_2A_3A_4$ nor in the effective zone $B_1B_2B_3B_4$ (i.e. the wavefield in $C_1C_2C_3C_4 \backslash B_1B_2B_3B_4$). Furthermore, we only need to compute the imaging condition in the zone $A_1A_2A_3A_4$, no concern with the part in $C_1C_2C_3C_4 \backslash A_1A_2A_3A_4$.

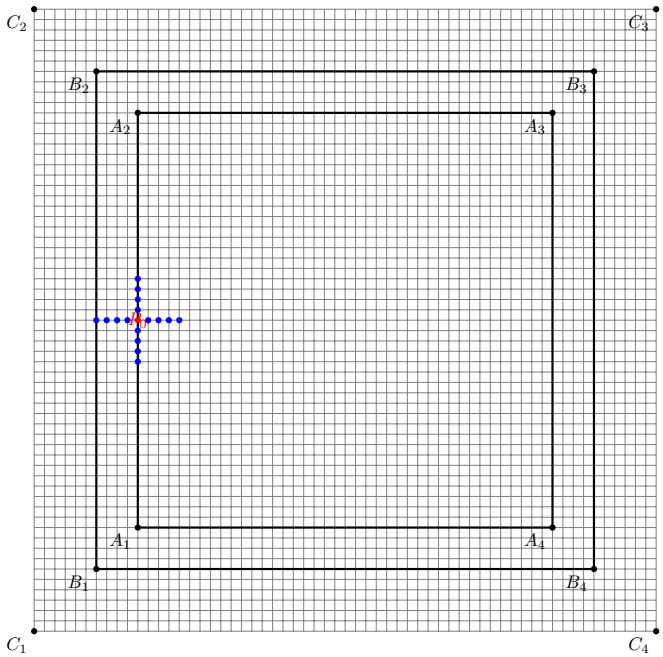


Figure 1: Extend the model size with artificial boundary. $A_1A_2A_3A_4$ indicates the original model size ($nz \times nx$). $C_1C_2C_3C_4$ is the extended model size $(nz + 2nb)(nx + 2nb)$. $B_1B_2B_3B_4 \backslash A_1A_2A_3A_4$ is the effective boundary area.

Table 1: Finite difference coefficients for regular grid (Order-2N)

i	-4	-3	-2	-1	0	1	2	3	4
$N=1$				1	-2	1			
$N=2$			-1/12	4/3	-5/2	4/3	-1/12		
$N=3$		1/90	-3/20	3/2	-49/18	3/2	-3/20	1/90	
$N=4$	-1/560	8/315	-1/5	8/5	-205/72	8/5	-1/5	8/315	-1/560

Effective boundary for regular grid finite difference

Assume $2N$-th order finite difference scheme is applied. The Laplacian operator is specified by

$$\begin{aligned} \nabla^2 p^k &= \partial_{xx} p^k + \partial_{zz} p^k \\ &= \frac{1}{\Delta z^2} \sum_{i=-N}^{N} c_i p^k[ix][iz+i] + \frac{1}{\Delta x^2} \sum_{i=-N}^{N} c_i p^k[ix+i][iz] \end{aligned} \tag{6}$$

where c_i is given by Table 1, see a detailed derivation in Fornberg (1988). The Laplacian operator has x and z with same finite difference structure. For x dimension only, the second derivative of order $2N$ requires at least N points in the boundary zone, as illustrated by Figure 2. In 2-D case, the required boundary zone has been plotted in Figure 3a. Note that four corners in $B_1B_2B_3B_4$ in Figure 1 are not needed. This is exactly the boundary saving scheme proposed by Dussaud et al. (2008).

Keep in mind that we only need to guarantee the correctness of the wavefield in the original model zone $A_1A_2A_3A_4$. However, the saved wavefield in $A_1A_2A_3A_4 \backslash B_1B_2B_3B_4$ is also correct. Is it possible to further shrink it to reduce number of points for saving? The answer is true. Our solution is: *saving the inner N layers on each side neighboring the boundary* $A_1A_2A_3A_4 \backslash D_1D_2D_3D_4$, as shown in Figure 3b. We call it the effective boundary for regular finite difference scheme.

After nt steps of forward modeling, we begin our backward propagation with the last 2 wavefield snap p^{nt} and p^{nt-1} and saved effective boundaries in $A_1A_2A_3A_4 \backslash D_1D_2D_3D_4$. At that moment, the wavefield is correct for every grid point. (Of course, the correctness of the wavefield in $A_1A_2A_3A_4$ is guaranteed.) At time k, we assume the wavefield in $A_1A_2A_3A_4$ is correct. One step of backward propagation means $A_1A_2A_3A_4$ is shrunk to $D_1D_2D_3D_4$. In other words, the wavefield in $D_1D_2D_3D_4$ is correctly reconstructed. Then we load the saved effective boundary of time k to overwrite the area $A_1A_2A_3A_4 \backslash D_1D_2D_3D_4$. Again, all points of the wavefield in $A_1A_2A_3A_4$ are correct. We repeat this overwriting and computing process from one time step to another ($k \rightarrow k-1$), in reverse time order. The wavefield in the boundary $C_1C_2C_3C_4 \backslash A_1A_2A_3A_4$ may be incorrect because the points here are neither saved nor correctly reconstructed from the previous step.

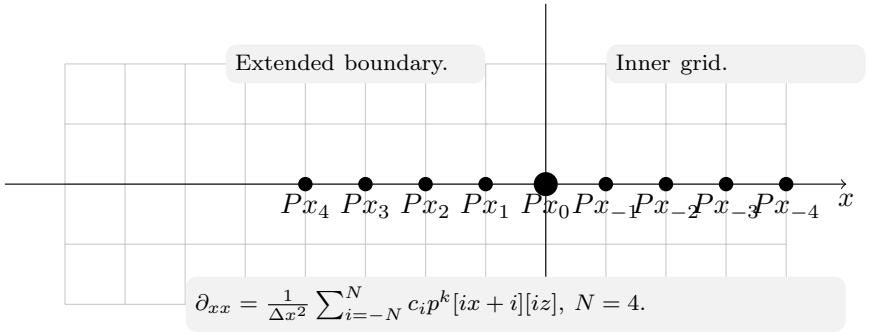


Figure 2: 1-D schematic plot of required points in regular grid for boundary saving. Computing the laplacian needs N points in the extended boundary zone, the rest $N+1$ points in the inner model grid. N points is required for boundary saving.

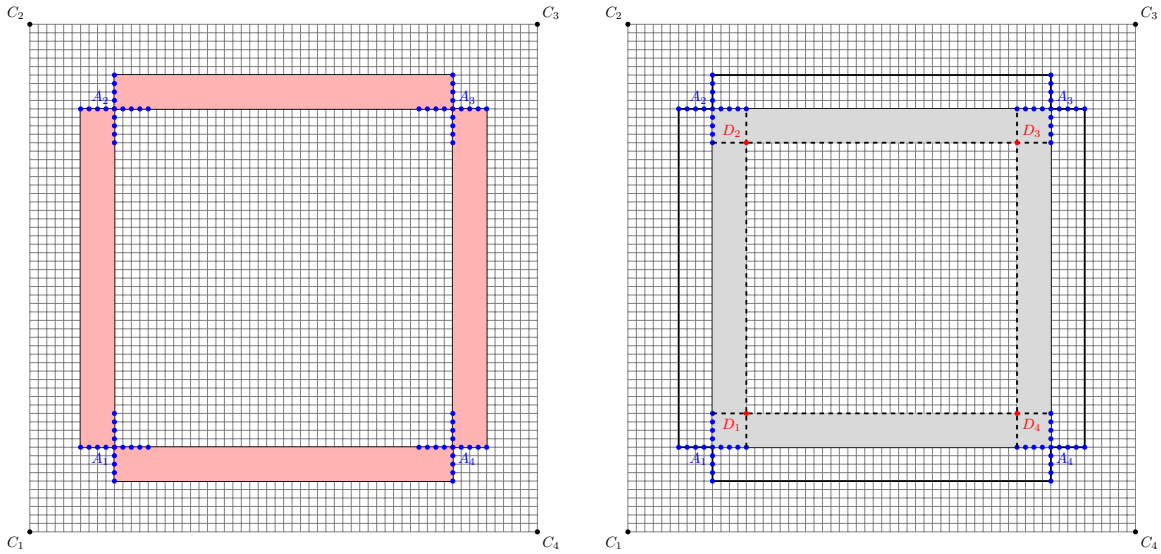


Figure 3: A 2-D sketch of required points for boundary saving for regular grid finite difference: (a) The scheme proposed by Dussaud et al. (2008) (red zone). (b) Proposed effective boundary saving scheme (gray zone).

Table 2: Finite difference coefficients for staggered grid (Order-$2N$)

i	1	2	3	4
$N=1$	1			
$N=2$	1.125	-0.0416667		
$N=3$	1.171875	-0.0651041667	0.0046875	
$N=4$	1.1962890625	-0.079752604167	0.0095703125	-0.000697544642857

Effective boundary for staggered grid finite difference

The limitation of boundary saving strategy proposed in Dussaud et al. (2008) is that only regular grid finite difference scheme is considered in RTM. In the case of staggered grid, half grid points are employed to obtain higher accuracy for finite difference. Recursion from time k to $k+1$ (or $k-1$) may not be realized with ease due to the Laplacian operator, which involves the second derivative. An effective approach is to split Eq. (5) into several first derivative equations or combinations of first derivative and second derivative equations. The first derivative is defined as

$$\partial_u f = \frac{1}{\Delta u}\left(\sum_{i=1}^{N} c_i(f[u+i\Delta u/2]-f[u-i\Delta u/2])\right), u=z,x \tag{7}$$

where the finite difference coefficients are listed in Table 2.

The use of half grid points in staggered grid makes the effective boundary a little different from that in regular grid. To begin with, we define some intermediate auxiliary variables: $Ax := \partial_x p$, $Az := \partial_z p$, $Px := \partial_x Ax$ and $Pz := \partial_z Az$. Thus the acoustic wave equation reads

$$\begin{cases} \dfrac{\partial^2 p}{\partial t^2} = v^2\,(Px+Pz) \\ Px = \partial_x Ax, Pz = \partial_z Az \\ Ax = \partial_x p, Az = \partial_z p \end{cases} \tag{8}$$

It implies that we have to conduct 2 finite difference steps (one for Ax and Az and the other for Px and Pz) to compute the Laplacian in one step of time marching. Take 8-th order ($2N=8$) finite difference in x dimension for example. As can be seen from Figure 4, computing ∂_{xx} at Px_0 needs the correct values at Ax_4,Ax_3,Ax_2,Ax_1 in the boundary; computing Ax_1,Ax_2,Ax_3,Ax_4 needs the correct values at Px_4,Px_5,Px_6,Px_7 in the boundary. An intuitive approach is saving N points of Ax ($Ax_1,\ldots,Ax_4$) and N points of Px ($Px_4,\ldots,Px_7$). The saving procedure guarantees the correctness of these points in the wavefield. Another possible approach is just saving the $2N-1$ points of Px ($Px_1,\ldots,Px_7$). In this way, the values of $Ax_1,\ldots,Ax_4$ can be correctly obtained from the calculation of the first derivative. The latter method is preferable because it is much easier for implementation while requiring less points. Speaking two dimensionally, some points in the four corners at in $B_1B_2B_3B_4$ of Figure 1 may be still

necessary to store, as shown in Figure 5a. The reason is that you are working with Laplacian, not second derivative in one dimension. Again, we switch our boundary saving part from out of $A_1A_2A_3A_4$ to $A_1A_2A_3A_4 \backslash D_1D_2D_3D_4$. Less grid points are required to guarantee correct reconstruction while points in the corner are no longer needed. Therefore, *the proposed effective boundary for staggered finite difference needs* $2N-1$ *points to be saved on each side*, see Figure 5b.

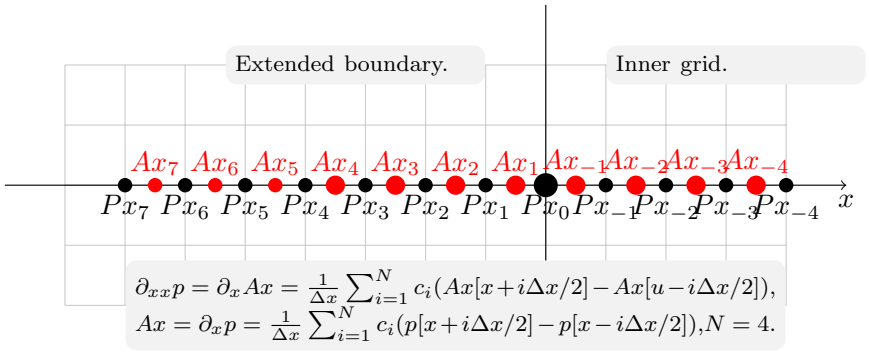


Figure 4: $2N$-th order staggered grid finite difference: correct backward propagation needs $2N-1$ points on one side. For $N=4$, computing ∂_{xx} at Px_0 needs the correct values at Ax_4, Ax_3, Ax_2, Ax_1 in the boundary; computing Ax_4,Ax_3, Ax_2, Ax_1 needs the correct values at Px_4, Px_5, Px_6, Px_7 in the boundary. Thus, $2N-1=7$ points in boundary zone is required to guarantee the correctness of the inner wavefield.

Storage analysis

For the convenience of complexity analysis, we define the size of the original model as $nz \times nx$. In each direction, we pad the model with the nb points on both sides as the boundary. Thus, the extended model size becomes $(nz+2nb)(nx+2nb)$. Conventionally one has to save the whole wavefield within the model size on the disk. The required number of points is

$$nz \cdot nx. \tag{9}$$

According to Dussaud et al. (2008), for $2N$-th order finite difference in regular grid, N points on each side are added to guarantee the correctness of inner wavefield. The saving amount of every time step is

$$2N \cdot nz + 2N \cdot nx = 2N(nz+nx). \tag{10}$$

In the proposed effective boundary saving strategy, the number becomes

$$2N \cdot nz + 2N \cdot nx - 4N^2 = 2N(nz+nx) - 4N^2. \tag{11}$$

In the case of staggered grid, there are $2N-1$ points on each side. Allowing for four corners, the number for the effective boundary saving is

$$2(2N-1)nz + 2(2N-1)nx - 4(2N-1)^2 = 2(2N-1)(nz+nx) - 4(2N-1)^2 \tag{12}$$

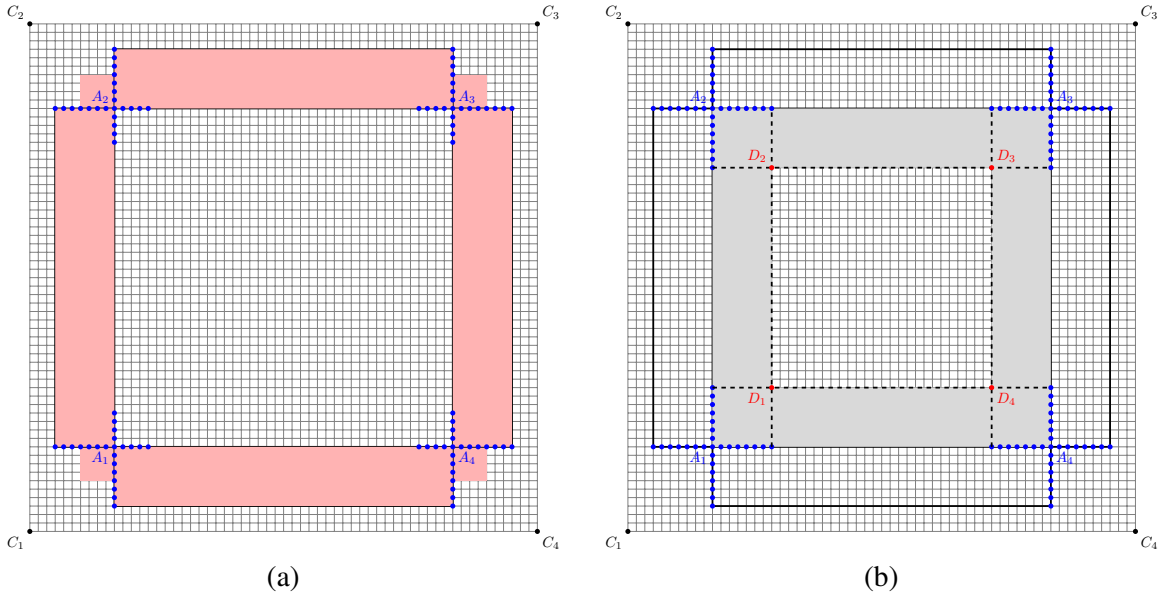


Figure 5: A 2-D sketch of required points for boundary saving for staggered grid finite difference: (a) Saving the points outside the model (red region). (b) Effective boundary, saving the points inside the model zone (gray region).

Table 3: Storage requirement for different saving strategy

Boundary saving scheme	Saving amount (Unit: Bytes)
Conventional saving strategy	$4nt \cdot nz \cdot nx$
Dussaud's: regular grid	$4nt \cdot 2N(nz + nx)$
Effective boundary: regular grid	$4nt \cdot (2N(nz + nx) - 4N^2)$
Effective boundary: staggered grid	$4nt \cdot (2(2N - 1)(nz + nx) - 4(2N - 1)^2)$

Assume the forward modeling is performed nt steps using the floating point format on the computer. The saving amount will be multiplied by $nt \cdot \text{sizeof(float)} = 4nt$. Table 3 lists this memory requirement for different boundary saving strategies.

In principle, the proposed effective boundary saving will reduce $4nt \cdot 4N^2$ bytes for regular grid finite difference, compared with the method of Dussaud et al. (2008). The storage requirement of staggered grid based effective boundary saving is about $(2N - 1)/N$ times of that in the regular grid finite difference, by observing $2N \ll nb \ll nx, nz$. For the convenience of practical implementation, the four corners can be saved twice so that the saving burden of the effective boundary saving has no difference with the method of Dussaud et al. (2008) in regular grid finite difference. Since the saving burden for staggered grid finite difference has not been touched in Dussaud et al. (2008), it is still of special value to minimize its storage requirement for GPU computing.

GPU IMPLEMENTATION USING CPML BOUNDARY CONDITION

CPML boundary condition

To combine the absorbing effects into the acoustic equation, CPML boundary condition is such a nice way that we merely need to combine two convolutional terms into the above equations:

$$\begin{cases} \dfrac{\partial^2 p}{\partial t^2} = v^2 \left(Px + Pz\right) \\ Px = \partial_x Ax + \Psi_x \\ Pz = \partial_z Az + \Psi_z \\ Ax = \partial_x p + \Phi_x \\ Az = \partial_z p + \Phi_z \end{cases} \tag{13}$$

where Ψ_x, Ψ_z are the convolutional terms of Ax and Az; Φ_x, Φ_z are the convolutional terms of Px and Pz. These convolutional terms can be computed via the following relation:

$$\begin{cases} \Psi_x^n = b_x \Psi_x^{n-1} + (b_x - 1)\partial_x^{n+1/2} Ax \\ \Psi_z^n = b_z \Psi_z^{n-1} + (b_z - 1)\partial_z^{n+1/2} Az \\ \Phi_x^n = b_x \Phi_x^{n-1} + (b_x - 1)\partial_x^{n-1/2} p \\ \Phi_z^n = b_z \Phi_z^{n-1} + (b_z - 1)\partial_z^{n-1/2} p \end{cases} \tag{14}$$

where $b_x = e^{-d(x)\Delta t}$ and $b_z = e^{-d(z)\Delta t}$. In the absorbing layers, the damping parameter $d(u)$ we used is (Collino and Tsogka, 2001):

$$d(u) = d_0 (\frac{u}{L})^2, d_0 = -\frac{3v}{2L} \ln(R), \tag{15}$$

where L indicates the PML thickness; u represent the distance between current position (in PML) and PML inner boundary. R is always chosen as $10^{-3} \sim 10^{-6}$. For more details about the derivation of CPML, the interested readers are referred to Collino and Tsogka (2001) and Komatitsch and Martin (2007). The implementation of CPML boundary condition is easy to carry out: in each iteration the wavefield extrapolation is performed according to the first equation in (13); it follows by adding the convolutional terms in terms of (14).

Memory manipulation

Consider the Marmousi model (size=751x2301) and the Sigsbee model (size=1201x3201). Assume $nt = 10000$ and the finite difference of order $2N = 8$. Conventionally, one have to store 64.4 GB for Marmousi model and 143.2 GB for Sigsbee model on the disk of the computer. Using the method of Dussaud et al. (2008) or regular grid based effective boundary saving, the storage requirement will be greatly reduced, about 0.9

GB and 1.3 GB for the two models. Staggered grid finite difference is preferable due to higher accuracy, however, the saving amount of effective boundary needs 1.6 GB and 2.3 GB for the two models, much larger than regular grid. Besides the additional variable allocation, the storage requirement may still be a bottleneck to save all boundaries on GPU to avert the CPU saving and data exchange for low-level hardware, even if we are using effective boundary saving.

Fortunately, page-locked (also known as pinned) host memory provides us a practical solution to mitigate this conflict. Zero-copy system memory has identical coherence and consistency to global memory. Copies between page-locked host memory and device memory can be performed concurrently with kernel execution (Nvidia, 2011). * Therefore, we store a certain percentage of effective boundary on the page-locked host memory, and the rest on device. A reminder is that overuse of the pinned memory may degrade the bandwidth performance.

Code organization

Allowing for the GPU block alignment, the thickness of CPML boundary is chosen to be 32. Most of the CUDA kernels are configured with a block size 16x16. Some special configurations are related to the initialization and calculation of CPML boundary area. The CPML variables are initialized along x and z axis with CUDA kernels `cuda_init_abcz(...)` and `cuda_init_abcx(...)`. When `device_alloc(...)` is invoked to allocate memory, there is a variable `phost` to control the percentage of the effective boundary saved on host and device memory by calling the function `cudaHostAlloc(...)`. A pointer is referred to the pinned memory via `cudaHostGetDevicePointer(...)` The wavelet is generated on device using `cuda_ricker_wavelet(...)` with a dominant frequency `fm` and delayed wavelength. Adding a shot can be done by a smooth bell transition `cuda_add_bellwlt(...)`. We implement RTM (of order `NJ=2, 4, 6, 8, 10`) with forward and backward propagation functions `step_forward(...)` and `step_backward(...)`, in which the shared memory is also used for faster computation. The cross-correlation imaging of each shot is done by `cuda_cross_correlate(...)`. The final image can be obtained by stacking the images of many shots using `cuda_imaging(...)`. Most of the low-frequency noise can be removed by applying the muting function `cuda_mute(...)` and the Laplacian filtering `cuda_laplace_filter(...)`.

NUMERICAL EXAMPLES

Exact reconstruction

To make sure that the proposed effective boundary saving strategy does not introduce any kind of error/artifacts for the source wavefield, the first example is designed using

*Generally, a computer has same or larger amount of resource on host compared with GDDR memory on device.

a constant velocity model: velocity=2000 m/s, $nz = nx = 320$, $\Delta z = \Delta x = 5m$. The source position is set at the center of the model. The modeling process is performed $nt = 1000$ time samples. We record the modeled wavefield snap at $k = 420$ and $k = 500$, as shown in the top panels of Figure 6. The backward propagation starts from $k = 1000$ and ends up with $k = 1$. In the backward steps, the reconstructed wavefield at $k = 500$ and $k = 420$ are also recorded, shown in the bottom panels of Figure 6. We also plot the wavefield in the boundary zone in both two panels. Note that the correctness of the wavefield in the original model zone is guaranteed while the wavefield in the boundary zone does not need to be correct.

Marmousi model

The second example is GPU-based RTM for Marmousi model (Figure 7) using our effective boundary saving. The spatial sampling interval is $\Delta x = \Delta z = 4m$. 51 shots are deployed. In each shot, 301 receivers are placed in the split shooting mode. The parameters we use are listed as follows: $nt = 13000$, $\Delta t = 0.3$ ms. Due to the limited resource on our computer, we store 65% boundaries using page-locked memory. Figure 8 gives the resulting RTM image after Laplacian filtering. As shown in the figure, RTM with the effective boundary saving scheme produces excellent image: the normalized cross-correlation imaging condition greatly improves the deeper parts of the image due to the illumination compensation. The events in the central part of the model, the limits of the faults and the thin layers are much better defined.

Sigsbee model

The last example is Sigsbee model shown in Figure 9. The spatial interval is $\Delta x = \Delta z = 25m$. 55 shots are evenly distributed on the surface of the model. We still perform $nt = 13000$ time steps for each shot (301 receivers). Due to the larger model size, 75% boundaries have to be stored with the aid of pinned memory. Our RTM result is shown in Figure 10. Again, the resulting image obtained by normalized cross-correlation imaging condition exhibits better resolution for the edges of the salt body and the diffraction points. Some events in the image using normalized cross-correlation imaging condition are more visible, while they have a much lower amplitude or are even completely lost in the image of cross-correlation imaging condition.

CONCLUSION AND DISCUSSION

In this paper, we introduce the effective boundary saving strategy for GPU-based RTM imaging. Compared with the method of Dussaud et al. (2008), the saving amount of effective boundary with regular grid finite difference scheme is slightly

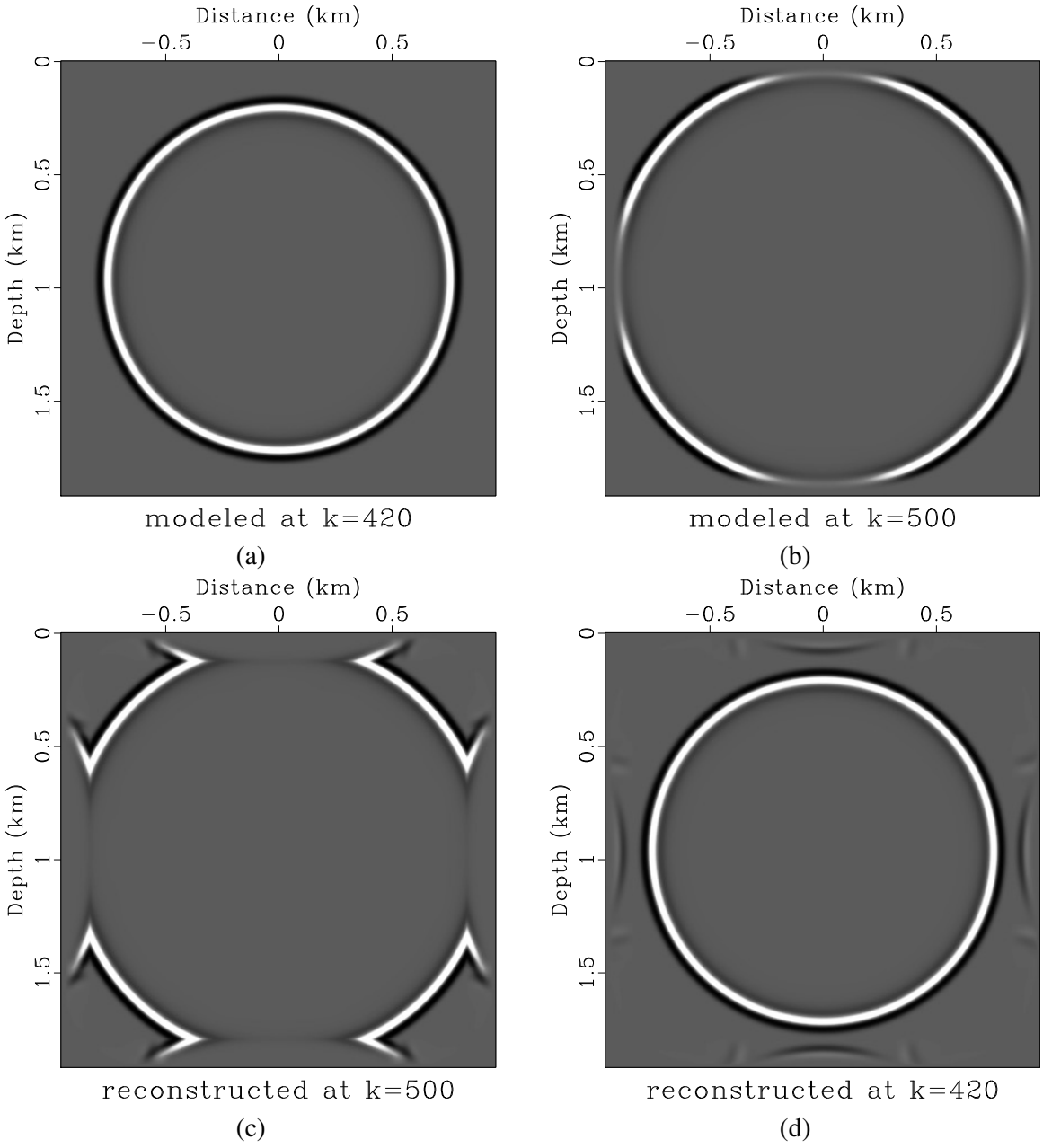


Figure 6: The wavefield snaps with a constant velocity model: velocity=2000 m/s, $nz = nx = 320$, $\Delta z = \Delta x = 5m$, source at the center. The forward modeling is conducted with $nt = 1000$ time samples. (a–b) Modeled wavefield snaps at $k = 420$ and $k = 500$. The backward propagation starts from $k = 1000$ and ends at $k = 1$. (c–d) Reconstructed wavefield snaps at $k = 500$ and $k = 420$. Note the correctness of the wavefield in the original model zone is guaranteed while the wavefield in the boundary zone may be incorrect (32 layers of the boundary on each side are also shown in the figure).

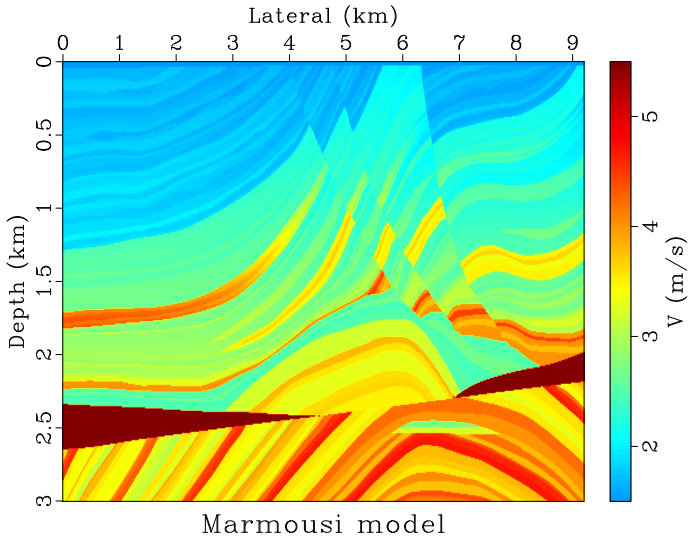


Figure 7: The Marmousi velocity model. gpurtm/marmousi/ marmousi

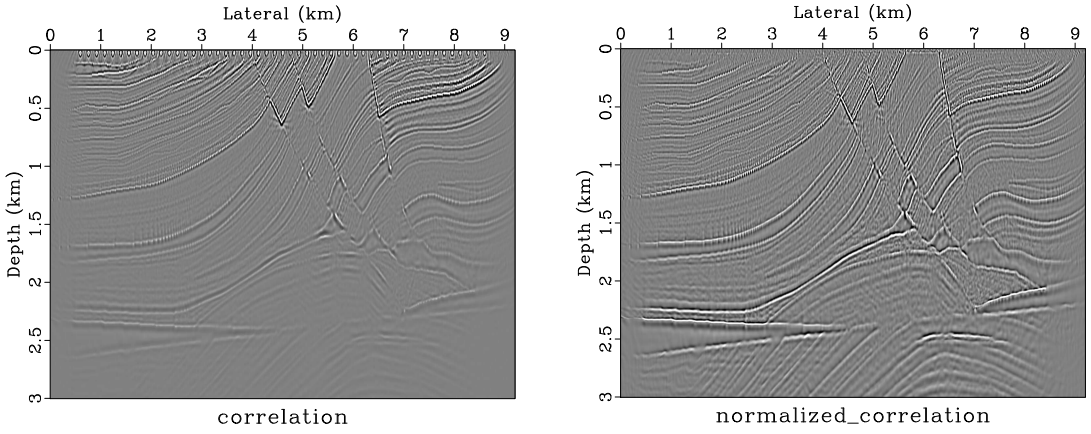


Figure 8: RTM result of Marmousi model using effective boundary saving scheme (staggered grid finite difference). (a) Result of cross-correlation imaging condition. (b) Result of normalized cross-correlation imaging condition. gpurtm/marmousi/ mimag1,mimag2

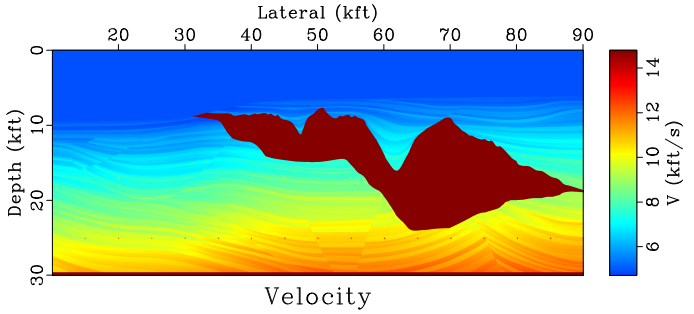


Figure 9: The Sigsbee velocity model. gpurtm/sigsbee/ sigsbee

reduced. The RTM storage of effective boundary saving for staggered finite difference is first explored, and then implemented with CPML boundary condition. We demonstrate the validity of effective boundary saving strategy by numerical test and imaging of benchmark models.

The focus of this paper is RTM implementation using effective boundary saving in staggered grid instead of GPU acceleration. A limitation of this work is that the numerical examples are generated with NVS5400M GPU on a laptop (compute capability 2.1, GDDR3). It is easy to do performance analysis for different dataset size and higher stencil orders if the latest GPU card and CUDA driver are available. It is also possible to obtain improved speedup by incorporating MPI with GPU programming using advanced clusters with larger GDDR memory (Komatitsch et al., 2010a; Suh et al., 2010) or FPGA optimization (Fu and Clapp, 2011; Medeiros et al., 2011). Unfortunately, higher stencil orders of staggered grid RTM using effective boundary implementation in 3D is still a problem. 3D RTM using the 2nd order regular grid finite difference with Clayton and Enquist boundary condition (only 1 layer on each side to save) needs tens of GBs (Liu et al., 2013b). It implies that 3D RTM with higher stencil orders will definitely exceed the memory bound of current and next generation GPUs. For GPU implementation of 3D RTM, the practical way is using the random boundary condition (Liu et al., 2013a) or saving on the disk. A deeper discussion of the practical issues for GPU implementation of RTM can be found in Liu et al. (2012a).

ACKNOWLEDGMENTS

The work of the first author is supported by China Scholarship Council during his visit to The University of Texas at Austin. This work is sponsored by National Science Foundation of China (No. 41390454). We wish to thank Sergey Fomel and two anonymous reviewers for constructive suggestions, which lead to massive amount of revision and improvement in this paper. The code of even-order GPU-based prestack RTM (combined with CPML boundary condition) using effective boundary saving strategy is available alongside this paper. The RTM examples are reproducible with

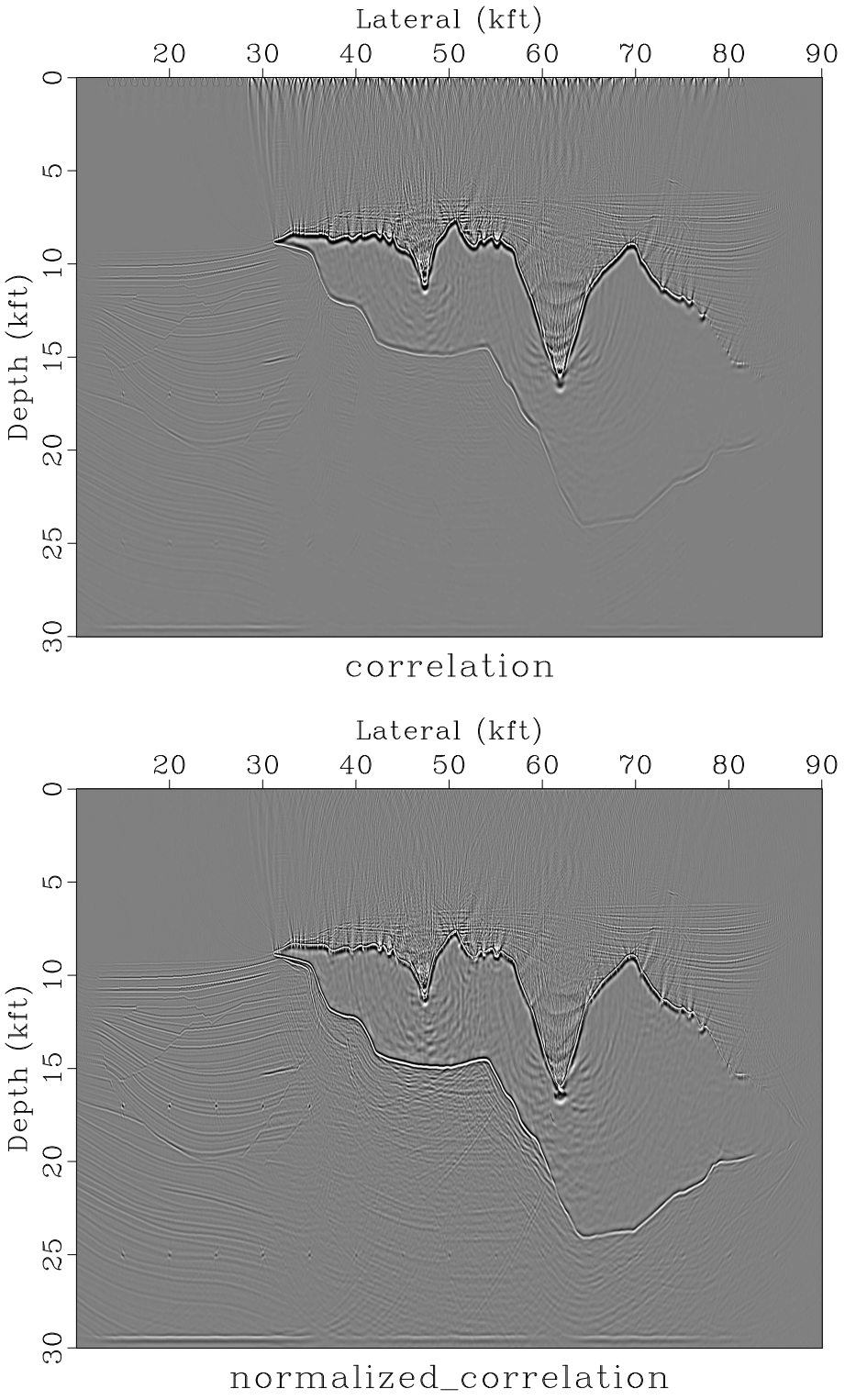


Figure 10: RTM result of Sigsbee model using effective boundary saving scheme (staggered grid finite difference). (a) Result of cross-correlation imaging condition. (b) Result of normalized cross-correlation imaging condition. gpurtm/sigsbee/ simag1,simag2

the help of Madagascar software package (Fomel et al., 2013).

REFERENCES

Abdelkhalek, R., H. Calandra, O. Coulaud, J. Roman, and G. Latu, 2009, Fast seismic modeling and reverse time migration on a gpu cluster: International Conference on High Performance Computing & Simulation, HPCS'09., IEEE, 36–43.

Baysal, E., D. D. Kosloff, and J. W. Sherwood, 1983, Reverse time migration: Geophysics, **48**, 1514–1524.

Biondi, B., 2006, 3d seismic imaging: Society of Exploration Geophysicists.

Boonyasiriwat, C., G. Zhan, M. Hadwiger, M. Srinivasan, and G. Schuster, 2010, Multisource reverse-time migration and full-waveform inversion on a gpgpu: Presented at the 72nd EAGE Conference & Exhibition.

Cerjan, C., D. Kosloff, R. Kosloff, and M. Reshef, 1985, A nonreflecting boundary condition for discrete acoustic and elastic wave equations: Geophysics, **50**, 705–708.

Clapp, R. G., 2009, Reverse time migration with random boundaries: 79th Annual International Meeting, SEG Expanded Abstracts, 2809–2813.

Clapp, R. G., H. Fu, and O. Lindtjorn, 2010, Selecting the right hardware for reverse time migration: The Leading Edge, **29**, 48–58.

Collino, F., and C. Tsogka, 2001, Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media: Geophysics, **66**, 294–307.

Dussaud, E., W. W. Symes, P. Williamson, L. Lemaistre, P. Singer, B. Denel, and A. Cherrett, 2008, Computational strategies for reverse-time migration: SEG Annual meeting.

Foltinek, D., D. Eaton, J. Mahovsky, P. Moghaddam, and R. McGarry, 2009, Industrial-scale reverse time migration on gpu hardware: Presented at the 2009 SEG Annual Meeting.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8.

Fornberg, B., 1988, Generation of finite difference formulas on arbitrarily spaced grids: Mathematics of computation, **51**, 699–706.

Fu, H., and R. G. Clapp, 2011, Eliminating the memory bottleneck: an fpga-based solution for 3d reverse time migration: Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays, ACM, 65–74.

Guitton, A., B. Kaelin, and B. Biondi, 2006, Least-squares attenuation of reverse-time-migration artifacts: Geophysics, **72**, S19–S23.

Guo, M., Y. Chen, and H. Wang, 2013, The application of gpu-based tti rtm in a complex area with shallow gas and fault shadow-a case history: Presented at the 75th EAGE Conference & Exhibition.

Hussain, T., M. Pericas, N. Navarro, and E. Ayguadé, 2011, Implementation of a reverse time migration kernel using the hce high level synthesis tool: International Conference on Field-Programmable Technology (FPT), IEEE, 1–8.

Ji, Q., S. Suh, and B. Wang, 2012, Iterative velocity model building using gpu based layer-stripping tti rtm, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–5.

Kim, Y., Y. Cho, U. Jang, and C. Shin, 2013, Acceleration of stable {TTI} p-wave reverse-time migration with {GPUs}: Computers & Geosciences, **52**, 204 – 217.

Komatitsch, D., G. Erlebacher, D. Göddeke, and D. Michéa, 2010a, High-order finite-element seismic wave propagation modeling with mpi on a large gpu cluster: Journal of Computational Physics, **229**, 7692–7714.

Komatitsch, D., and R. Martin, 2007, An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation: Geophysics, **72**, SM155–SM167.

Komatitsch, D., D. Michéa, G. Erlebacher, and D. Göddeke, 2010b, Running 3d finite-difference or spectral-element wave propagation codes 25x to 50x faster using a gpu cluster: Presented at the 72nd EAGE Conference & Exhibition.

Leader, C., and R. Clapp, 2012, Least squares reverse time migration on gpus-balancing io and computation: Presented at the 74th EAGE Conference & Exhibition.

Lin, C., and H. Wang, 2012, Application of gpus in seismic depth migration: Presented at the 74th EAGE Conference & Exhibition.

Liu, G., Y. Liu, L. Ren, and X. Meng, 2013a, 3d seismic reverse time migration on gpgpu: Computers & Geosciences, **59**, 17 – 23.

Liu, H., R. Ding, L. Liu, and H. Liu, 2013b, Wavefield reconstruction methods for reverse time migration: Journal of Geophysics and Engineering, **10**, 015004.

Liu, H., B. Li, H. Liu, X. Tong, Q. Liu, X. Wang, and W. Liu, 2012a, The issues of prestack reverse time migration and solutions with graphic processing unit implementation: Geophysical Prospecting, **60**, 906–918.

Liu, H., H. Liu, X. Shi, R. Ding, and J. Liu, 2012b, Gpu based pspi one-way wave high resolution migration, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–5.

Liu, H.-w., H. Liu, X.-L. Tong, and Q. Liu, 2012c, A fourier integral algorithm and its gpu/cpu collaborative implementation for one-way wave equation migration: Computers & Geosciences, **45**, 139–148.

Liu, W., T. Nemeth, A. Loddoch, J. Stefani, R. Ergas, L. Zhuo, B. Volz, O. Pell, and J. Huggett, 2009, Anisotropic reverse-time migration using co-processors: Presented at the SEG Houston International Exposition. SEG.

McMechan, G., 1983, Migration by extrapolation of time-dependent boundary values: Geophysical Prospecting, **31**, 413–420.

Medeiros, V., R. Rocha, A. Ferreira, J. Correia, J. Barbosa, A. Silva-Filho, M. Lima, R. Gandra, and R. Bragança, 2011, Fpga-based accelerator to speed-up seismic applications: 2011 Simpasio em Sistemas Computacionais (WSCAD-SSC), IEEE, 9–9.

Michéa, D., and D. Komatitsch, 2010, Accelerating a three-dimensional finite-difference wave propagation code using gpu graphics cards: Geophysical Journal International, **182**, 389–402.

Micikevicius, P., 2009, 3d finite difference computation on gpus using cuda: Pro-

ceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, ACM, 79–84.

Nvidia, C., 2011, Nvidia cuda c programming guide.

Suh, S., and B. Wang, 2011, Expanding domain methods in gpu based tti reverse time migration, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 3460–3464.

Suh, S. Y., A. Yeh, B. Wang, J. Cai, K. Yoon, and Z. Li, 2010, Cluster programming for reverse time migration: The leading edge, **29**, 94–97.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, SM213–SM221.

Weiss, R. M., and J. Shragge, 2013, Solving 3d anisotropic elastic wave equations on parallel gpu devices: Geophysics, **78**, F7–F15.

Ying, S., T. Dong-sheng, and K. Xuan, 2013, Denoise investigation on prestack reverse time migration based on gpu/cpu collaborative parallel accelerating computation: Fifth International Conference on Computational and Information Sciences (ICCIS), IEEE, 30–33.

Yoon, K., C. Shin, S. Suh, L. R. Lines, and S. Hong, 2003, 3d reverse-time migration using the acoustic wave equation: An experience with the seg/eage data set: The Leading Edge, **22**, 38–41.

A graphics processing unit implementation of time-domain full-waveform inversion

Pengliang Yang, Jinghuai Gao*, and Baoli Wang† **

ABSTRACT

The graphics processing unit (GPU) has become a popular device for seismic imaging and inversion due to its superior speedup performance. In this paper we implement GPU-based full waveform inversion (FWI) using the wavefield reconstruction strategy. Because the computation on GPU is much faster than CPU-GPU data communication, in our implementation the boundaries of the forward modeling are saved on the device to avert the issue of data transfer between host and device. The Clayton-Enquist absorbing boundary is adopted to maintain the efficiency of GPU computation. A hybrid nonlinear conjugate gradient algorithm combined with the parallel reduction scheme is utilized to do computation in GPU blocks. The numerical results confirm the validity of our implementation.

INTRODUCTION

The classical time-domain full waveform inversion (FWI) was originally proposed by Tarantola (1984) to refine the velocity model by minimizing the energy in the difference between predicted and observed data in the least-squares sense (Symes, 2008). It was further developed by Tarantola (1986) with applications to elastic cases (Pica et al., 1990). After Pratt et al. (1998) proposed frequency domain FWI, the multiscale inversion became an area of active research, and provided a hierarchical framework for robust inversion. The Laplace-domain FWI and the Laplace-Fourier domain variant have also been developed by Shin and Cha (2008, 2009). Until now, building a good velocity model is still a challenging problem and attracts increasing effort of geophysicists (Virieux and Operto, 2009).

There are many drawbacks in FWI, such as the non-linearity, the non-uniqueness of the solution, as well as the expensive computational cost. The goal of FWI is to match the synthetic and the observed data. The minimization of the misfit function is essentially an iterative, computationally intensive procedure: at each iteration one has to calculate the gradient of the objective function with respect to the model parameters by cross correlating the back propagated residual wavefield with the corresponding forward propagated source wavefield. The forward modeling itself demands

***e-mail:** ypl.2100@gmail.com, jhgao@mail.xjtu.edu

large computational efforts, while back propagation of the residual wavefield has large memory requirements to access the source wavefield.

Recent advances in computing capability and hardware makes FWI a popular research subject to improve velocity models. As a booming technology, graphics processing unit (GPU) has been widely used to mitigate the computational drawbacks in seismic imaging (Micikevicius, 2009; Yang et al., 2014) and inversion (Boonyasiriwat et al., 2010; Shin et al., 2014), due to its potential gain in performance. One key problem for GPU implementation is that the parallel computation is much faster while the data communication between host and device always takes longer time. In this paper we report a 2D implementation of GPU-based FWI using a wavefield reconstruction strategy. The boundaries of the forward modeling are saved on the device to avert the issue of CPU-GPU data transfer. Shared memory on the GPU is used to speedup the modeling computation. A hybrid nonlinear conjugate gradient method is adopted in the FWI optimization. In each iteration, a Gaussian shaping step is employed to remove noise in the computed gradient. We demonstrate the validity and the relatively superior speedup of our GPU implementation of FWI using the Marmousi model.

FWI AND ITS GPU IMPLEMENTATION

FWI: data mismatch minimization

In the case of constant density, the acoustic wave equation is specified by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} - \nabla^2 p(\mathbf{x},t;\mathbf{x}_s) = f_s(\mathbf{x},t;\mathbf{x}_s). \tag{1}$$

where we have set $f_s(\mathbf{x},t;\mathbf{x}_s) = f(t')\delta(\mathbf{x}-\mathbf{x}_s)\delta(t-t')$. According to the above equation, a misfit vector $\Delta\mathbf{p} = \mathbf{p}_{cal} - \mathbf{p}_{obs}$ can be defined by the differences at the receiver positions between the recorded seismic data $\mathbf{p}_{obs}$ and the modeled seismic data $\mathbf{p}_{cal} = \mathbf{f}(\mathbf{m})$ for each source-receiver pair of the seismic survey. Here, in the simplest acoustic velocity inversion, $\mathbf{f}(\cdot)$ indicates the forward modeling process while $\mathbf{m}$ corresponds to the velocity model to be determined. The goal of FWI is to match the data misfit by iteratively updating the velocity model. The objective function taking the least-squares norm of the misfit vector $\Delta\mathbf{p}$ is given by

$$E(\mathbf{m}) = \frac{1}{2}\Delta\mathbf{p}^\dagger\Delta\mathbf{p} = \frac{1}{2}\|\mathbf{p}_{cal}-\mathbf{p}_{obs}\|^2 = \frac{1}{2}\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t|p_{cal}(\mathbf{x}_r,t;\mathbf{x}_s)-p_{obs}(\mathbf{x}_r,t;\mathbf{x}_s)|^2 \tag{2}$$

where ns and ng are the number of sources and geophones, $\dagger$ denotes the adjoint operator (conjugate transpose). The recorded seismic data is only a small subset of the whole wavefield at the locations specified by sources and receivers.

The gradient-based minimization method updates the velocity model according to a descent direction $\mathbf{d}_k$:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \alpha_k \mathbf{d}_k. \tag{3}$$

where k denotes the iteration number. By neglecting the terms higher than the 2nd order, the objective function can be expanded as

$$E(\mathbf{m}_{k+1}) = E(\mathbf{m}_k + \alpha_k \mathbf{d}_k) = E(\mathbf{m}_k) + \alpha_k \langle \nabla E(\mathbf{m}_k), \mathbf{d}_k \rangle + \frac{1}{2}\alpha_k^2 \mathbf{d}_k^\dagger \mathbf{H}_k \mathbf{d}_k, \tag{4}$$

where $\mathbf{H}_k$ stands for the Hessian matrix; $\langle \cdot, \cdot \rangle$ denotes inner product. Differentiation of the misfit function $E(\mathbf{m}_{k+1})$ with respect to α_k gives

$$\alpha_k = -\frac{\langle \mathbf{d}_k, \nabla E(\mathbf{m}_k) \rangle}{\mathbf{d}_k^\dagger \mathbf{H}_k \mathbf{d}_k} = -\frac{\langle \mathbf{d}_k, \nabla E(\mathbf{m}_k) \rangle}{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{J}_k \mathbf{d}_k \rangle} = \frac{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{p}_{obs} - \mathbf{p}_{cal} \rangle}{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{J}_k \mathbf{d}_k \rangle}, \tag{5}$$

in which we use the approximate Hessian $\mathbf{H}_k := \mathbf{H}_a = \mathbf{J}_k^\dagger \mathbf{J}_k$ and $\nabla_{\mathbf{m}} E = \mathbf{J}^\dagger \Delta \mathbf{p}$, according to equation (72). A detailed derivation of the minimization process is given in Appendix A.

Nonlinear conjugate gradient method

The conjugate gradient (CG) algorithm decreases the misfit function along the conjugate gradient direction:

$$\mathbf{d}_k = \begin{cases} -\nabla E(\mathbf{m}_0), & k = 0 \\ -\nabla E(\mathbf{m}_k) + \beta_k \mathbf{d}_{k-1}, & k \geq 1 \end{cases} \tag{6}$$

There are a number of ways to compute β_k. We use a hybrid a hybrid scheme combing Hestenes-Stiefel method and Dai-Yuan method (Hager and Zhang, 2006)

$$\beta_k = \max(0, \min(\beta_k^{HS}, \beta_k^{DY})). \tag{7}$$

in which

$$\begin{cases} \beta_k^{HS} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1}) \rangle}{\langle \mathbf{d}_{k-1}, \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1}) \rangle} \\ \beta_k^{DY} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k) \rangle}{\langle \mathbf{d}_{k-1}, \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1}) \rangle} \end{cases} \tag{8}$$

This provides an automatic direction reset while avoiding over-correction of β_k in conjugate gradient iteration. It reduces to steepest descent method when the subsequent search directions lose conjugacy. The gradient of the misfit function w.r.t. the model is given by (Bunks et al., 1995)

$$\nabla E_{\mathrm{m}} = \frac{2}{v^3(\mathbf{x})} \sum_{r=1}^{ng} \sum_{s=1}^{ns} \int_0^{t_{\max}} \frac{\partial^2 p_{cal}(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2} p_{res}(\mathbf{x}_r, t; \mathbf{x}_s) \mathrm{d}t \tag{9}$$

where $p_{res}(\mathbf{x}, t; \mathbf{x}_s)$ is the back propagated residual wavefield, see the Appendix B and C for more details. A Gaussian smoothing operation plays an important role in removing the noise in the computed gradient. A precondition is possible by normalizing the gradient by the source illumination which is the energy of forward wavefield accounting for geometrical divergence (Gauthier et al., 1986; Bai et al., 2014):

$$\nabla E(\mathrm{m}_k) = \frac{\nabla E_{\mathrm{m}}}{\sqrt{\sum_{s=1}^{ns} \int_0^{t_{\max}} p_{cal}^2(x, t; x_s)\mathrm{d}t + \gamma^2}} \tag{10}$$

where γ is a stability factor to avoid division by zero. To obtain a reasonable step size α_k in equation (85), we estimate a small step length ϵ proposed by Pica et al. (1990):

$$\max(\epsilon|\mathbf{d}_k|) \leqslant \frac{\max(|\mathbf{m}_k|)}{100}. \tag{11}$$

and the Taylor approximation

$$\mathbf{J}_k\mathbf{d}_k \approx \frac{\mathbf{f}(\mathbf{m}_k + \epsilon\mathbf{d}_k) - \mathbf{f}(\mathbf{m}_k)}{\epsilon} \tag{12}$$

We summarize the FWI flowchart in Figure 1.

Wavefield reconstruction via boundary saving

One key problem of GPU-based implementations of FWI is that the computation is always much faster than the data transfer between the host and device. Many researchers choose to reconstruct the source wavefield instead of storing the modeling time history on the disk, just saving the boundaries (Dussaud et al., 2008; Yang et al., 2014). For $2N$-th order finite difference, regular grid scheme needs to save N points on each side (Dussaud et al., 2008), while staggered-grid scheme required at least $2N-1$ points on each side (Yang et al., 2014). In our implementation, we use 2nd order regular grid finite difference because FWI begins with a rough model and velocity refinement is mainly carried out during the optimization. Furthermore, high-order finite differences and staggered-grid schemes do not necessarily lead to FWI converge to an accurate solution while requiring more compute resources. A key observation for wavefield reconstruction is that one can reuse the same template by exchanging the role of p^{k+1} and p^{k-1}. In other words, for forward modeling we use

$$p^{k+1} = 2p^k - p^{k-1} + v^2\Delta t^2\nabla^2 p^k. \tag{13}$$

while for backward reconstruction we use

$$p^{k-1} = 2p^k - p^{k+1} + v^2\Delta t^2\nabla^2 p^k. \tag{14}$$

The wavefield extrapolation can be stepped efficiently via pointer swap, i.e.,

$$\begin{aligned} &for\ ix, iz... \quad p_0(:) = 2p_1(:) - p_0(:) + v^2(:)\Delta t^2\nabla^2 p_1(:) \\ &ptr = p_0; p_0 = p_1; p_1 = ptr; //\text{swap pointer} \end{aligned} \tag{15}$$

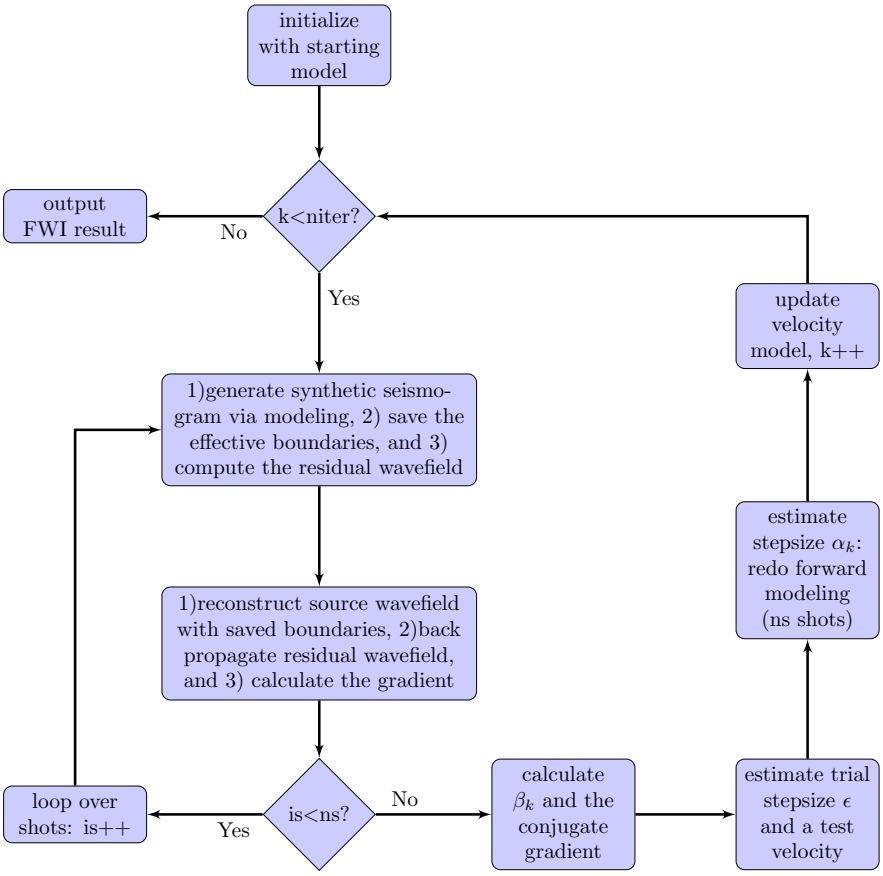


Figure 1: Backward reconstruction can be realized using the saved boundaries. Note that no absorbing boundary condition is applied on the top boundary of the model in the forward modeling.

where $(:) = [ix, iz]$, p_0 and p_1 are p^{k+1}/p^{k-1} and p^k, respectively.

Note that all the computation is done on GPU blocks. In our codes, the size of the block is set to be 16x16. We replicate the right- and bottom-most cols/rows enough times to bring the total model size up to an even multiple of block size. As shown in Figure 2, the whole computation area is divided into 16x16 blocks. For each block, we use a 18x18 shared memory array to cover all the grid points in this block. It implies that we add a redundant point on each side, which stores the value from other blocks, as marked by the window in Figure 2. When the computation is not performed for the interior blocks, special care needs to be paid to the choice of absorbing boundary condition (ABC) in the design of FWI codes. Allowing for efficient GPU implementation, we use the 45^o Clayton-Enquist ABC proposed in Clayton and Engquist (1977) and Engquist and Majda (1977). For the left boundary, it is

$$\frac{\partial^2 p}{\partial x \partial t} - \frac{1}{v}\frac{\partial^2 p}{\partial t^2} = \frac{v}{2}\frac{\partial^2 p}{\partial z^2} \tag{16}$$

which requires only one layer to be saved on each side for wavefield reconstruction. The equations for right and bottom boundary can also be written in a similar way. To simulate free surface boundary condition, no ABC is applied to the top boundary. The same technique has been adopted by Liu et al. (2013) for reverse time migration. We believe its application to FWI is valuable and straightforward.

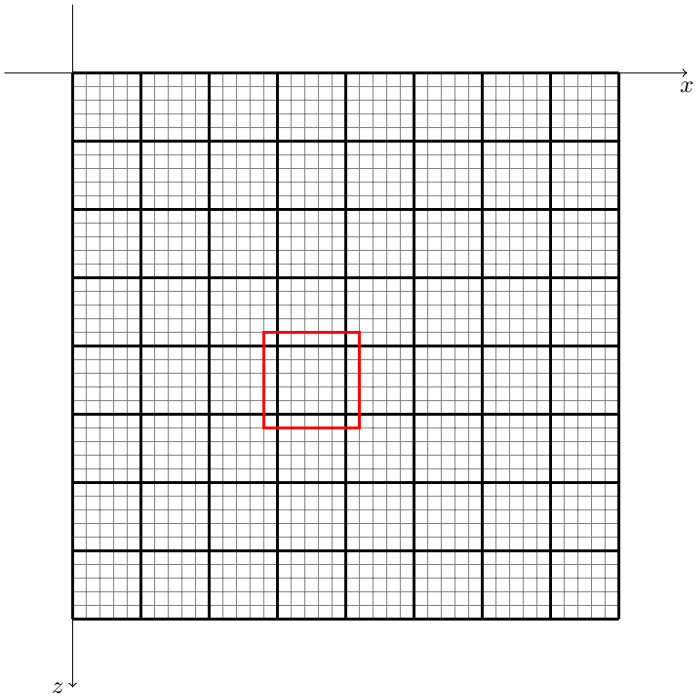


Figure 2: 2D blocks in GPU memory. The marked window indicates that the shared memory in every block needs to be extended on each side with halo ghost points storing the grid value from other blocks.

Parallel reduction on CUDA blocks

Recognizing that hardware serializes divergent thread execution within the same warp, but all threads within a warp must complete execution before that warp can end, we use a parallel reduction technique to find the maximum of the model vector $\mathbf{m}_k$ and the descent vector $\mathbf{d}_k$, as well as summation for the inner product in the numerator and the denominator of α_k. A sequential addressing scheme is utilized because it is free of conflict (Harris et al., 2007). As shown in Figure 3, parallel reduction approach builds a summation tree to do stepwise partial sums. In each level half of the threads will perform reading from global memory and writing to shared memory. The required number of threads will decrease to be half of previous level. It reduces the serial computational complexity from $O(N)$ to $O(\log_2(N))$: In each step many threads perform computation simultaneously, leading to low arithmetic intensity. In this way, we expect a significant improvement in computational efficiency.

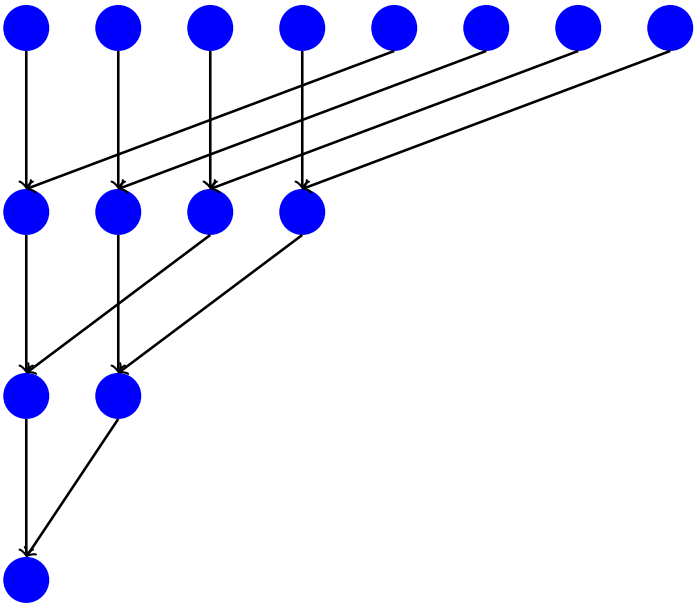


Figure 3: Parallel reduction on GPU block. It reduces a serial computational complexity $O(N)$ to be $O(\log_2(N))$ steps: in each step many threads perform computation simultaneously, leading to low arithmetic intensity.

NUMERICAL RESULTS

Exact reconstruction with saved boundaries

Since we are advocating the wavefield reconstruction method in FWI, the foremost thing is to demonstrate that the boundary saving strategy does not introduce any kind of errors or artifacts for the wavefield to be reconstructed. To attain this goal, we design a constant velocity model: velocity=2000 m/s, $nz = nx = 200$, $\Delta z = \Delta x = 5$

m. A 15 Hz Ricker wavelet is taken as the source and is placed at the center of the model. We do the modeling process for 1000 steps with time interval $\Delta t = 1$ ms. We record the modeled wavefield snap at 0.28 s and 0.4 s, as shown in the top panels of Figure 10. The figure shows that at time 0.4 s, the wavefield has already spread to the boundaries which absorb most of the reflection energy. In the backward steps, the reconstructed shot snaps at 0.4 s and 0.28 s are also recorded, shown in the bottom panels of Figure 10. As can be seen from the figure, the backward reconstruction starts from the boundaries (bottom left) and gradually recovers the interior wavefield (bottom right).

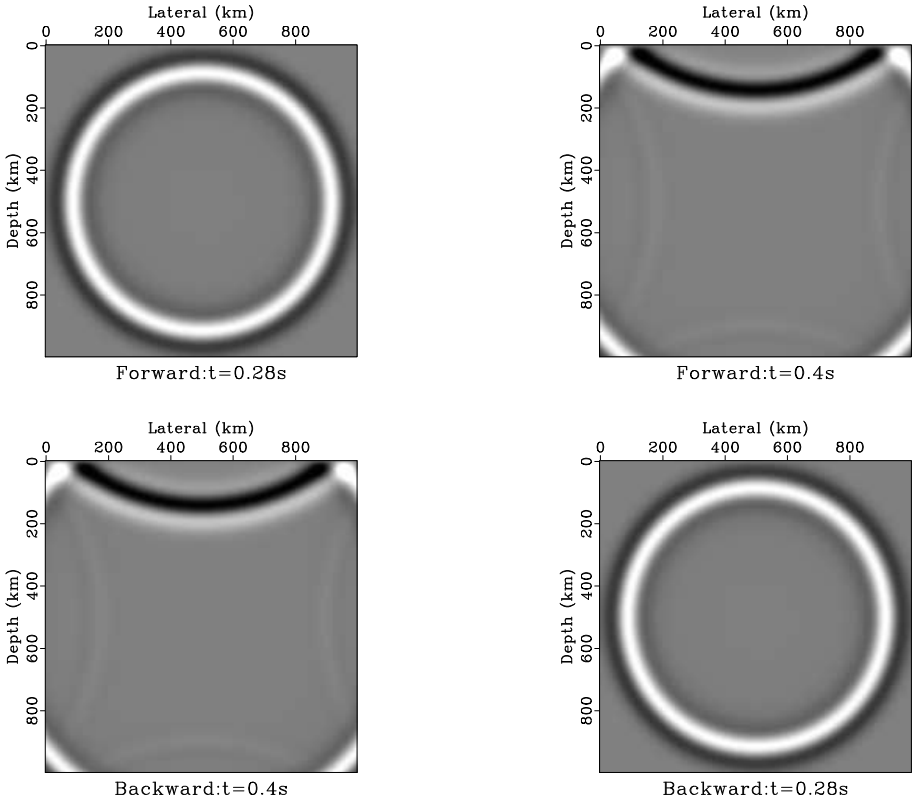


Figure 4: Backward reconstruction can be realized using the saved boundaries. Note that no absorbing boundary condition is applied on the top boundary of the model in the forward modeling. gpufwi/fbrec/ fb

Speedup performance

The acceleration of GPU implementation on advanced computer hardware is a key concern of many researchers. There are many factors which may accelerate the FWI computation. Compared with saving the wavefield on disk, wavefield reconstruction will accelerate the GPU computing because no CPU-GPU data transfer is needed any more. The parallel reduction to find the maxium value of model vector $\mathbf{m}_k$ and descent direction vector $\mathbf{d}_k$ is another factor to speedup the FWI computation.

However, among these factors, the forward modeling takes most of the computing time. Each iteration needs four times of forward modeling: two of them are for sources and receivers; one is performed for wavefield reconstruction and gradient calculation, and another one is to estimate the step length α_k. Therefore, we only focus on the speedup obtained in the forward modeling procedure.

To do the performance analysis, we run the sequential implementation CPU code and parallel multi-thread GPU code of forward modeling for 1000 time steps. We estimate the average time cost of 5 shots for different data sizes. Because the GPU block size is set to be 16x16. To make the comparison fair, we generate test models whose size is of multiple 16x16 blocks. The size of the test model is choosen to be $nx \cdot nz$, $nx = nz = i \cdot 160$, where $i = 1, \ldots, 7$ is an integer. We only have a NVS5400 GPU card (compute capability 2.1, GDDR3) run on a laptop. Even so, compared with sequential implementation on host, we still achieve approximately 5.5–6 times speedup on the GPU device, as shown in Figure 5.

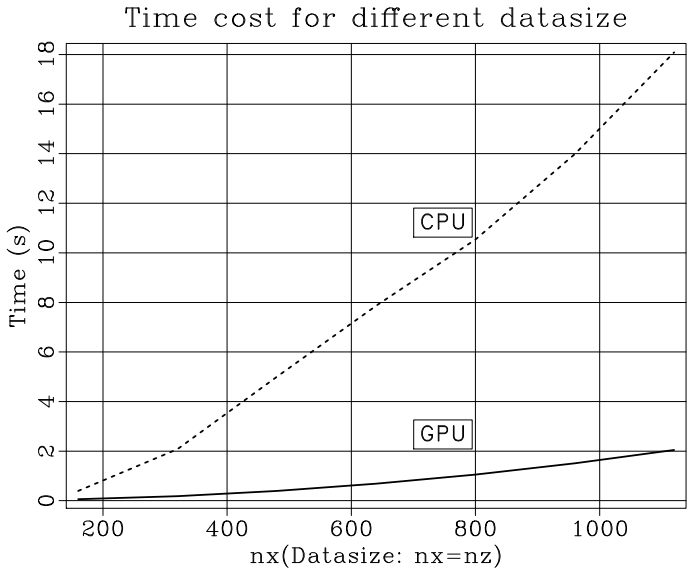


Figure 5: Comparison of the time cost for CPU- vs. GPU implementation under different model sizes with one shot, 1000 time steps of forward modeling. gpufwi/speedup/ timecost

Marmousi model

We use the Marmousi model for the benchmark test, as shown in the top panel of Figure 4. FWI tacitly requires a good starting model incorporated with low frequency information. 21 shots are deployed as the observations in the FWI, while 3 of them are shown in Figure 5. We use a starting model (bottom panel of Figure 4) obtained by smoothing the original model 20 times with a 5x5 window.

The FWI is carried out for 300 iterations. A 10 Hz Ricker wavelet is deployed in our modeling and inversion. We record all the updated velocity to make sure the velocity refinement is going on during the iterative procedure. The updated velocity model at iterations 1, 20, 50, 100, 180 and 300 is displayed in Figure 6. Figure 7 describes the decreasing misfit function in iterations. As can be seen from the Figures 6 and 7, the velocity model changes significantly at the early stage. Later iterations in FWI make some improvement on small details for the velocity model. More iterations will refine the model further, however, gaining less and less improvement.

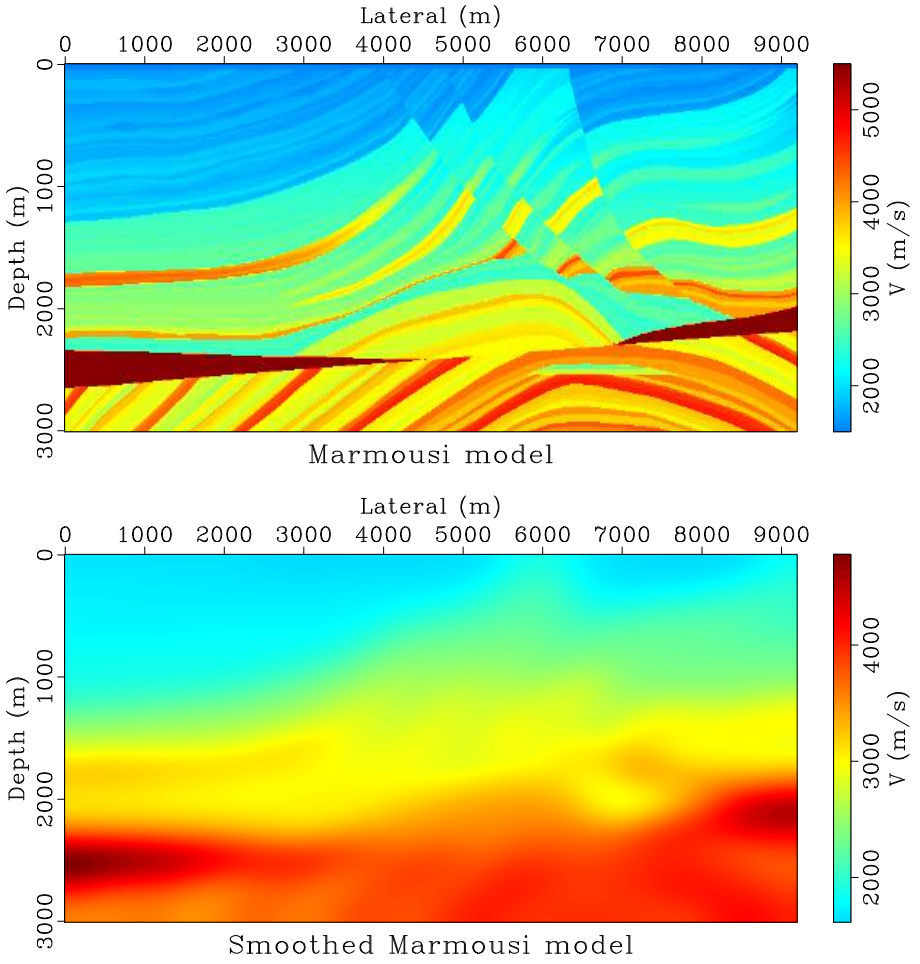


Figure 6: Top: The original Marmousi is downsampled by a factor of 3 along depth and lateral direction. The shots are generated according to the subsampled Marmousi model. Bottom: The starting model of FWI for Marmousi model, which is obtained by smoothing the original model 20 times with a 5x5 window. gpufwi/marmtest/ marm

CONCLUSION

We have implemented GPU-based FWI using the wavefield reconstruction strategy, which averts the issue of CPU-GPU data transfer. The Clayton-Enquist absorbing

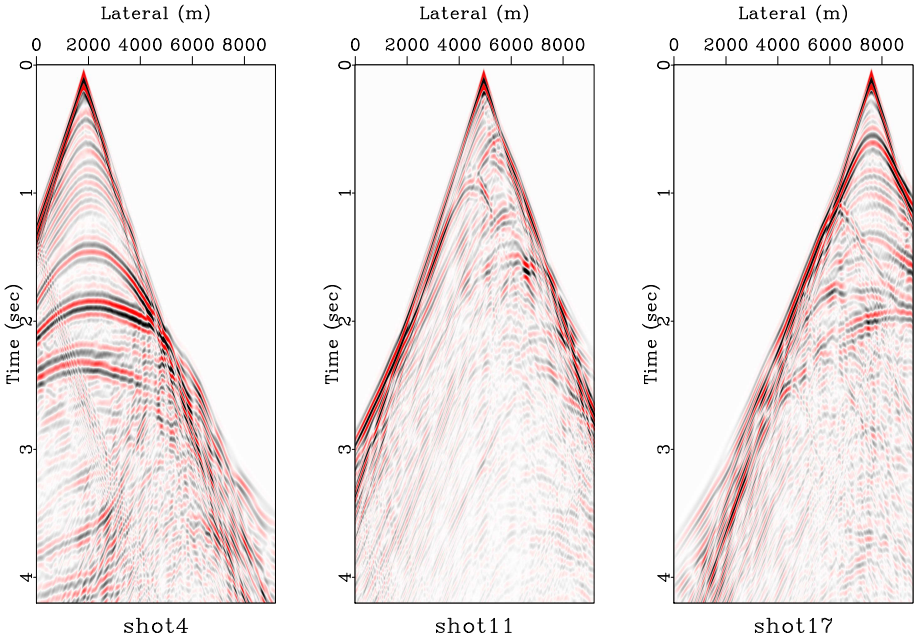


Figure 7: 21 shots were deployed in the FWI. Here, shots 4, 11 and 17 are shown from left to right. gpufwi/marmtest/ shotsnap

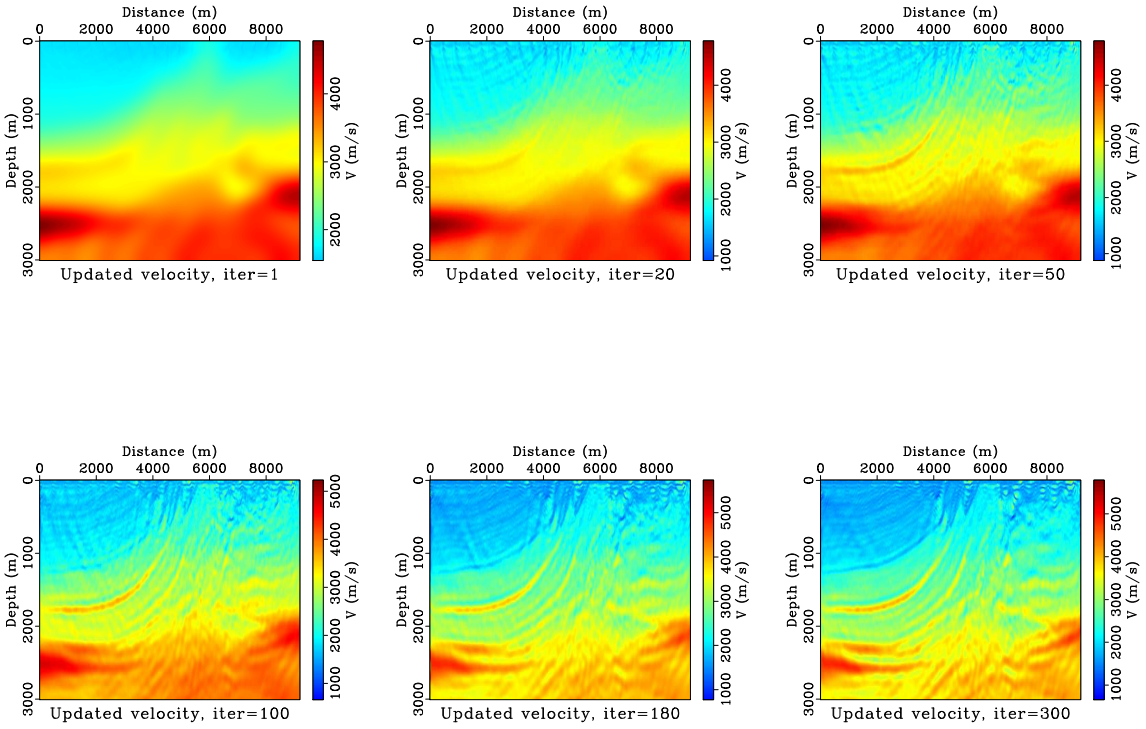


Figure 8: The updated velocity model at iterations 1, 20, 50, 100, 180 and 300. gpufwi/marmtest/ vsnap

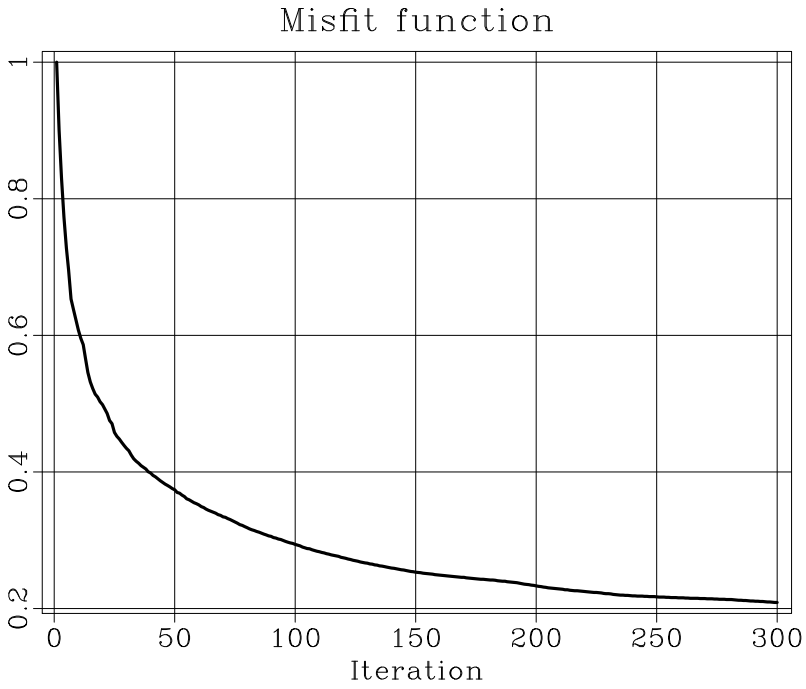


Figure 9: The misfit function decreases with iteration. gpufwi/marmtest/ objs

boundary was utilized to maintain the efficiency of GPU computation. A hybrid nonlinear conjugate gradient method combined with parallel reduction technique was adopted in the FWI optimization. The validity of our implementation for GPU-based FWI was demonstrated using a numerical test.

DISCUSSION

It is important to point out that FWI can be accelerated in many ways. A good choice of preconditioning operator may lead to fast convergence rate and geologically consistent results (Virieux and Operto, 2009; Ayeni et al., 2009; Guitton et al., 2012). Multishooting and source encoding method is also a possible solution for accelerating FWI (Schiemenz and Igel, 2013; Moghaddam et al., 2013). These techniques can be combined with GPU implementation (Wang et al., 2011). There are many reports advocating their acceleration performance based on particular GPU hardware. These reports may be out of date soon once the more powerful and advanced GPU product are released. Although the speedup performance of our implementation may be a little poor due to our hardware condition, we believe that it is useful to give readers the implementation code to do performance analysis using their own GPU cards. The current GPU-based FWI implementation parallelizes the forward modeling process which makes it possible to run FWI on a single node and low-level GPU condition even for a laptop. However, it is completely possible to obtain higher speedup performance using the latest, high performance GPU products, and further parallelize the code on

multi-GPU architectures using message passing interface (MPI) programming.

ACKNOWLEDGMENTS

The work of the first author is supported by China Scholarship Council during his visit to Bureau of Economic Geology, The University of Texas at Austin. This work is sponsored by National Science Foundation of China (No. 41390454). Thanks go to IFP for the Marmousi model. We wish to thank Sergey Fomel for valuable help to incorporate the codes into Madagascar software package (Fomel et al., 2013) (`http://www.ahay.org`), which makes all the numerical examples reproducible. The paper is substantially improved according to the suggestions of Joe Dellinger, Robin Weiss and two other reviewers.

APPENDIX A

MISFIT FUNCTION MINIMIZATION

Here, we mainly follow the delineations of FWI by Pratt et al. (1998) and Virieux and Operto (2009).The minimum of the misfit function $E(\mathbf{m})$ is sought in the vicinity of the starting model $\mathbf{m}_0$. The FWI is essentially a local optimization. In the framework of the Born approximation, we assume that the updated model $\mathbf{m}$ of dimension M can be written as the sum of the starting model $\mathbf{m}_0$ plus a perturbation model $\Delta\mathbf{m}$: $\mathbf{m} = \mathbf{m}_0 + \Delta\mathbf{m}$. In the following, we assume that $\mathbf{m}$ is real valued.

A second-order Taylor-Lagrange development of the misfit function in the vicinity of $\mathbf{m}_0$ gives the expression

$$E(\mathbf{m}_0+\Delta\mathbf{m}) = E(\mathbf{m}_0)+\sum_{i=1}^{M}\frac{\partial E(\mathbf{m}_0)}{\partial m_i}\Delta m_i+\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j}\Delta m_i \Delta m_j+O(||\Delta\mathbf{m}||^3) \tag{A-1}$$

Taking the derivative with respect to the model parameter m_i results in

$$\frac{\partial E(\mathbf{m})}{\partial m_i} = \frac{\partial E(\mathbf{m}_0)}{\partial m_i} + \sum_{j=1}^{M}\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_j \partial m_i}\Delta m_j, i = 1, 2, \ldots, M. \tag{A-2}$$

Equation (A-2) can be abbreviated as

$$\frac{\partial E(\mathbf{m})}{\partial \mathbf{m}} = \frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} + \frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}\Delta\mathbf{m} \tag{A-3}$$

Thus,

$$\Delta\mathbf{m} = -\left(\frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}\right)^{-1}\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} \tag{A-4}$$

where

$$\nabla E_{\mathbf{m}} = \frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} = \left[\frac{\partial E(\mathbf{m}_0)}{\partial m_1}, \frac{\partial E(\mathbf{m}_0)}{\partial m_2}, \ldots, \frac{\partial E(\mathbf{m}_0)}{\partial m_M}\right]^T \tag{A-5}$$

and

$$\mathbf{H} = \frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2} = \left(\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j}\right) \tag{A-6}$$

$\nabla E_{\mathbf{m}}$ and $\mathbf{H}$ are the gradient vector and the Hessian matrix, respectively.

$$\nabla E_{\mathbf{m}} = \nabla E(\mathbf{m}) = \frac{\partial E(\mathbf{m})}{\partial \mathbf{m}} = \mathtt{Re}\left[\left(\frac{\partial \mathbf{f}(\mathbf{m})}{\partial \mathbf{m}}\right)^{\dagger} \Delta\mathbf{p}\right] = \mathtt{Re}\left[\mathbf{J}^{\dagger}\Delta\mathbf{p}\right] \tag{A-7}$$

where $\mathtt{Re}$ takes the real part, and $\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{m})}{\partial \mathbf{m}}$ is the Jacobian matrix, i.e., the sensitivity or the Frchet derivative matrix.

In matrix form

$$\mathbf{H} = \frac{\partial^2 E(\mathbf{m})}{\partial \mathbf{m}^2} = \mathtt{Re}\left[\mathbf{J}^{\dagger}\mathbf{J}\right] + \mathtt{Re}\left[\frac{\partial \mathbf{J}^T}{\partial \mathbf{m}^T}(\Delta\mathbf{p}^*, \Delta\mathbf{p}^*, \ldots, \Delta\mathbf{p}^*)\right]. \tag{A-8}$$

In the Gauss-Newton method, this second-order term is neglected for nonlinear inverse problems. In the following, the remaining term in the Hessian, i.e., $\mathbf{H}_a = \mathtt{Re}[\mathbf{J}^{\dagger}\mathbf{J}]$, is referred to as the approximate Hessian. It is the auto-correlation of the derivative wavefield. Equation (68) becomes

$$\Delta\mathbf{m} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} = -\mathbf{H}_a^{-1}\mathtt{Re}[\mathbf{J}^{\dagger}\Delta\mathbf{p}]. \tag{A-9}$$

To guarantee the stability of the algorithm (avoiding the singularity), we may use $\mathbf{H} = \mathbf{H}_a + \eta\mathbf{I}$, leading to

$$\Delta\mathbf{m} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} = -(\mathbf{H}_a + \eta\mathbf{I})^{-1}\mathtt{Re}\left[\mathbf{J}^{\dagger}\Delta\mathbf{p}\right]. \tag{A-10}$$

Alternatively, the inverse of the Hessian in equation (68) can be replaced by $\mathbf{H} = \mathbf{H}_a \approx \mu\mathbf{I}$, leading to the gradient or steepest-descent method:

$$\Delta\mathbf{m} = -\mu^{-1}\nabla E_{\mathbf{m}} = -\alpha\nabla E_{\mathbf{m}} = -\alpha\mathtt{Re}\left[\mathbf{J}^{\dagger}\Delta\mathbf{p}\right]. \tag{A-11}$$

where $\alpha = \mu^{-1}$.

APPENDIX B

FRÉCHET DERIVATIVE

Recall that the basic acoustic wave equation reads

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2} - \nabla^2 p(\mathbf{x}, t; \mathbf{x}_s) = f_s(\mathbf{x}, t; \mathbf{x}_s).$$

The Green's function $G(\mathbf{x}, t; \mathbf{x}_s, t')$ is defined by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 G(\mathbf{x}, t; \mathbf{x}_s, t')}{\partial t^2} - \nabla^2 G(\mathbf{x}, t; \mathbf{x}_s, t') = \delta(\mathbf{x} - \mathbf{x}_s)\delta(t - t'). \tag{B-1}$$

Thus the integral representation of the solution can be given by (Tarantola, 1984)

$$\begin{aligned} p(\mathbf{x}_r, t; \mathbf{x}_s) &= \int_V \mathrm{d}\mathbf{x} \int \mathrm{d}t' G(\mathbf{x}_r, t; \mathbf{x}, t') f(\mathbf{x}, t'; \mathbf{x}_s) \\ &= \int_V \mathrm{d}\mathbf{x} \int \mathrm{d}t' G(\mathbf{x}_r, t - t'; \mathbf{x}, 0) f(\mathbf{x}, t'; \mathbf{x}_s) (\text{Causility of Green}'\text{s function}) \\ &= \int_V \mathrm{d}\mathbf{x} G(\mathbf{x}_r, t; \mathbf{x}, 0) * f(\mathbf{x}, t; \mathbf{x}_s) \end{aligned} \tag{B-2}$$

where $*$ denotes the convolution operator.

A perturbation $v(\mathbf{x}) \to v(\mathbf{x}) + \Delta v(\mathbf{x})$ will produce a field $p(\mathbf{x}, t; \mathbf{x}_s) + \Delta p(\mathbf{x}, t; \mathbf{x}_s)$ defined by

$$\frac{1}{(v(\mathbf{x}) + \Delta v(\mathbf{x}))^2}\frac{\partial^2 [p(\mathbf{x}, t; \mathbf{x}_s) + \Delta p(\mathbf{x}, t; \mathbf{x}_s)]}{\partial t^2} - \nabla^2 [p(\mathbf{x}, t; \mathbf{x}_s) + \Delta p(\mathbf{x}, t; \mathbf{x}_s)] = f_s(\mathbf{x}, t; \mathbf{x}_s) \tag{B-3}$$

Note that

$$\frac{1}{(v(\mathbf{x}) + \Delta v(\mathbf{x}))^2} = \frac{1}{v^2(\mathbf{x})} - \frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} + O(\Delta^2 v(\mathbf{x})) \tag{B-4}$$

Equation (94) subtracts equation (91), yielding

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 \Delta p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2} - \nabla^2 \Delta p(\mathbf{x}, t; \mathbf{x}_s) = \frac{\partial^2 [p(\mathbf{x}, t; \mathbf{x}_s) + \Delta p(\mathbf{x}, t; \mathbf{x}_s)]}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} \tag{B-5}$$

Using the Born approximation, equation (96) becomes

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 \Delta p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2} - \nabla^2 \Delta p(\mathbf{x}, t; \mathbf{x}_s) = \frac{\partial^2 p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} \tag{B-6}$$

Again, based on integral representation, we obtain

$$\Delta p(\mathbf{x}_r, t; \mathbf{x}_s) = \int_V \mathrm{d}\mathbf{x} G(\mathbf{x}_r, t; \mathbf{x}, 0) * \frac{\partial^2 p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})}. \tag{B-7}$$

APPENDIX C

GRADIENT COMPUTATION

In terms of equation (64),

$$\begin{aligned}\frac{\partial E(\mathbf{m})}{\partial m_i} &= \frac{1}{2}\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)(p_{cal}-p_{obs})^* + \left(\frac{\partial p_{cal}}{\partial m_i}\right)^*(p_{cal}-p_{obs})\right] \\ &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)^*\Delta p\right](\Delta p = p_{cal}-p_{obs}) \\ &= \mathtt{Re}\left[\left(\frac{\partial \mathbf{p}_{cal}}{\partial m_i}\right)^\dagger \Delta\mathbf{p}\right] = \mathtt{Re}\left[\left(\frac{\partial \mathbf{f}(\mathbf{m})}{\partial m_i}\right)^\dagger \Delta\mathbf{p}\right], i = 1,2,\ldots,M.\end{aligned} \tag{C-1}$$

According to the previous section, it follows that

$$\frac{\partial p_{cal}}{\partial v_i(\mathbf{x})} = \int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,t;\mathbf{x},0)*\ddot{p}(\mathbf{x},t;\mathbf{x}_s)\frac{2}{v^3(\mathbf{x})} = \int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,t;\mathbf{x},0)*\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}. \tag{C-2}$$

The convolution guarantees

$$\int \mathrm{d}t[g(t)*f(t)]h(t) = \int \mathrm{d}t f(t)[g(-t)*h(t)]. \tag{C-3}$$

Then, equation (71) becomes

$$\begin{aligned}\frac{\partial E(\mathbf{m})}{\partial m_i} &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)^*\Delta p\right](\Delta p = p_{cal}-p_{obs}) \\ &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t\mathtt{Re}\left[\left(\int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,t;\mathbf{x},0)*\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)\right] \\ &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*\left(\int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,-t;\mathbf{x},0)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)\right)\right] \\ &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*\left(\int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,0;\mathbf{x},t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)\right)\right] \\ &= \sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}} \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^* p_{res}(\mathbf{x}_r,t;\mathbf{x}_s)\right]\end{aligned} \tag{C-4}$$

where $p_{res}(\mathbf{x},t;\mathbf{x}_s)$ is a time-reversal wavefield produced using the residual $\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)$ as the source. As follows from reciprocity theorem,

$$p_{res}(\mathbf{x},t;\mathbf{x}_s) = \int_V \mathrm{d}\mathbf{x}G(\mathbf{x}_r,0;\mathbf{x},t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s) = \int_V \mathrm{d}\mathbf{x}G(\mathbf{x},0;\mathbf{x}_r,t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s). \tag{C-5}$$

satisfying

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p_{res}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} - \nabla^2 p_{res}(\mathbf{x},t;\mathbf{x}_s) = \Delta p(\mathbf{x}_r,t;\mathbf{x}_s). \tag{C-6}$$

It is noteworthy that an input f and the system impulse response function g are exchangeable in convolution. That is to say, we can use the system impulse response function g as the input, the input f as the impulse response function, leading to the same output. In the seismic modeling and acquisition process, the same seismogram can be obtained when we shoot at the receiver position $\mathbf{x}_r$ when recording the seismic data at position $\mathbf{x}$.

REFERENCES

Ayeni, G., Y. Tang, B. Biondi, et al., 2009, Joint preconditioned least-squares inversion of simultaneous source time-lapse seismic data sets: Presented at the 2009 SEG Annual Meeting.

Bai, J., D. Yingst, R. Bloor, and J. Leveille, 2014, Viscoacoustic waveform inversion of velocity structures in the time domain: Geophysics, **79**, R103–R119.

Boonyasiriwat, C., G. Zhan, M. Hadwiger, M. Srinivasan, and G. Schuster, 2010, Multisource reverse-time migration and full-waveform inversion on a GPGPU: Presented at the 72nd EAGE Conference & Exhibition.

Bunks, C., F. M. Saleck, S. Zaleski, and G. Chavent, 1995, Multiscale seismic waveform inversion: Geophysics, **60**, 1457–1473.

Clayton, R., and B. Engquist, 1977, Absorbing boundary conditions for acoustic and elastic wave equations: Bulletin of the Seismological Society of America, **67**, 1529–1540.

Dussaud, E., W. W. Symes, P. Williamson, L. Lemaistre, P. Singer, B. Denel, and A. Cherrett, 2008, Computational strategies for reverse-time migration: SEG Annual meeting.

Engquist, B., and A. Majda, 1977, Absorbing boundary conditions for numerical simulation of waves: Proceedings of the National Academy of Sciences, **74**, 1765–1766.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8.

Gauthier, O., J. Virieux, and A. Tarantola, 1986, Two-dimensional nonlinear inversion of seismic waveforms: Numerical results: Geophysics, **51**, 1387–1403.

Guitton, A., G. Ayeni, and E. Díaz, 2012, Constrained full-waveform inversion by model reparameterization 1: Geophysics, **77**, R117–R127.

Hager, W. W., and H. Zhang, 2006, A survey of nonlinear conjugate gradient methods: Pacific journal of Optimization, **2**, 35–58.

Harris, M., et al., 2007, Optimizing parallel reduction in cuda: NVIDIA Developer Technology, **2**, 45.

Liu, H., R. Ding, L. Liu, and H. Liu, 2013, Wavefield reconstruction methods for reverse time migration: Journal of Geophysics and Engineering, **10**, 015004.

Micikevicius, P., 2009, 3D finite difference computation on GPUs using CUDA: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, ACM, 79–84.

Moghaddam, P. P., H. Keers, F. J. Herrmann, and W. A. Mulder, 2013, A new optimization approach for source-encoding full-waveform inversion: Geophysics, **78**, R125–R132.

Pica, A., J. Diet, and A. Tarantola, 1990, Nonlinear inversion of seismic reflection data in a laterally invariant medium: Geophysics, **55**, 284–292.

Pratt, G., C. Shin, et al., 1998, Gauss–newton and full newton methods in frequency–space seismic waveform inversion: Geophysical Journal International, **133**, 341–362.

Schiemenz, A., and H. Igel, 2013, Accelerated 3-D full-waveform inversion using simultaneously encoded sources in the time domain: application to valhall ocean-bottom cable data: Geophysical Journal International, **195**, 1970–1988.

Shin, C., and Y. H. Cha, 2008, Waveform inversion in the Laplace domain: Geophysical Journal International, **173**, 922–931.

———, 2009, Waveform inversion in the Laplace-Fourier domain: Geophysical Journal International, **177**, 1067–1079.

Shin, J., W. Ha, H. Jun, D.-J. Min, and C. Shin, 2014, 3D laplace-domain full waveform inversion using a single GPU card: Computers & Geosciences, **67**, 1–13.

Symes, W. W., 2008, Migration velocity analysis and waveform inversion: Geophysical Prospecting, **56**, 765–790.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

———, 1986, A strategy for nonlinear elastic inversion of seismic reflection data: Geophysics, **51**, 1893–1903.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, WCC1–WCC26.

Wang, B., J. Gao, H. Zhang, W. Zhao, et al., 2011, CUDA-based acceleration of full waveform inversion on GPU: Presented at the 2011 SEG Annual Meeting, Society of Exploration Geophysicists.

Yang, P., J. Gao, and B. Wang, 2014, RTM using effective boundary saving: A staggered grid GPU implementation: Computers & Geosciences, **68**, 64 – 72.

Seislet-based morphological component analysis using scale-dependent exponential shrinkage

Pengliang Yang and Sergey Fomel†**

ABSTRACT

Morphological component analysis (MCA) is a powerful tool used in image processing to separate different geometrical components (cartoons and textures, curves and points etc). MCA is based on the observation that many complex signals may not be sparsely represented using only one dictionary/transform, however can have sparse representation by combining several over-complete dictionaries/transforms. In this paper we propose seislet-based MCA for seismic data processing. MCA algorithm is reformulated in the shaping-regularization framework. Successful seislet-based MCA depends on reliable slope estimation of seismic events, which is done by plane-wave destruction (PWD) filters. An exponential shrinkage operator unifies many existing thresholding operators and is adopted in scale-dependent shaping regularization to promote sparsity. Numerical examples demonstrate a superior performance of the proposed exponential shrinkage operator and the potential of seislet-based MCA in application to trace interpolation and multiple removal.

INTRODUCTION

A wide range of applications have been carried out by solving a series of linear inverse problems and using the fact that numerous varieties of signals can be sparsely represented with an appropriate dictionary, namely a certain kind of transform bases. Under the dictionary, the signals have fewer non-zeros of the representation coefficients. However, many complex signals are usually linear superposition of several elementary signals, and cannot be efficiently represented using only one dictionary. The concept of morphological diversity was therefore proposed by Starck et al. (2004, 2005) to combine several dictionaries for sparse representations of signals and images. Then, the signal is considered as a superposition of several morphological components. One has to choose a dictionary whose atoms match the shape of the geometrical structures to sparsify, while leading to a non-sparse (or at least not as sparse) representation of the other signal content. That is the essence of so-called morphological component analysis (MCA) (Starck et al., 2004, 2007; Woiselle et al., 2011).

Seislet transform and seislet frame are useful tools for seismic data compression and sparse representation (Fomel and Liu, 2010). Seislets are constructed by apply-

***e-mail:** ypl.2100@gmail.com, jhgao@mail.xjtu.edu

ing the wavelet lifting scheme (Sweldens, 1998) along the spatial direction, taking advantage of the prediction and update steps to characterize local structure of the seismic events. In the seislet transform, the locally dominant event slopes are found by plane-wave destruction (PWD), which is implemented using finite difference stencils to characterize seismic images by a superposition of local plane waves (Claerbout, 1992). By increasing the accuracy and dip bandwidth of PWD, Fomel (2002) demonstrated its competitive performance compared with prediction error filter (PEF) in the applications to fault detection, data interpolation, and noise attenuation. PWD keeps the number of adjustable parameters to a minimum, endows the estimated quantity with a clear physical meaning of the local plane-wave slope, and gets rid of the requirement of local windows in PEF. Recently, Chen et al. (2013a,b) accelerated the computation of PWD using an analytical estimator and improved its accuracy.

In this paper, we propose seislet-based MCA for seismic data processing. We reformulate MCA algorithm in the shaping-regularization framework (Fomel, 2007, 2008). Successful seislet-based MCA depends on reliable slope estimation of seismic events, which can be done by plane-wave destruction (PWD) filtering. Due to the special importance of an effective shrinkage or thresholding function in sparsity-promoting shaping optimization, we propose a scale-dependent exponential shrinkage operator, which can flexibly approximate many well-known existing thresholding functions. Synthetic and field data examples demonstrate the potential of seislet-based MCA in the application to trace interpolation and multiple removal.

MCA WITH SCALE-DEPENDENT SHAPING REGULARIZATION

Analysis-based iterative thresholding

A general inverse problem combined with a priori constraint $R(x)$ can be written as an optimization problem

$$\min_x \frac{1}{2}\|d_{obs} - Fx\|_2^2 + \lambda R(x), \tag{1}$$

where x is the model to be inverted, and d_{obs} is the observations. To solve the problem with sparsity constraint $R(x) = \|x\|_1$, the iterative shrinkage-thresholding (IST) algorithm has been proposed (Daubechies et al., 2004), which can be generally formulated as

$$x^{k+1} = T_\lambda(x^k + F^*(d_{obs} - Fx^k)), \tag{2}$$

where k denotes the iteration number; and F^* indicates the adjoint of F. $T_\lambda(x)$ is an element-wise shrinkage operator with threshold λ:

$$T_\lambda(x) = (t_\lambda(x_1), t_\lambda(x_2), \ldots, t_\lambda(x_m))^T, \tag{3}$$

in which the soft thresholding function (Donoho, 1995) is

$$t_\lambda(u) = \mathrm{Soft}_\lambda(u) = \begin{cases} u - \lambda\frac{u}{|u|}, & |u| > \lambda, \\ 0, & |u| \le \lambda. \end{cases} = u. * \max(1 - \frac{\lambda}{|u|}, 0). \tag{4}$$

Allowing for the missing elements in the data, the observations are connected to the complete data via the relation

$$d_{obs} = Md = M\Phi x = Fx, F = M\Phi. \tag{5}$$

where M is an acquisition mask indicating the observed and missing values. Assume Φ is a tight frame such that $\Phi^*\Phi = \mathrm{Id}$, $x = \Phi^*\Phi x = \Phi^* d$. It leads to

$$\begin{aligned} d^{k+1} &= \Phi x^{k+1} \\ &= \Phi T_\lambda(\Phi^* d^k + (M\Phi)^*(d_{obs} - Md^k)) \\ &= \Phi T_\lambda(\Phi^* d^k + \Phi^*(M^* d_{obs} - M^* M d^k)) \\ &= \Phi T_\lambda(\Phi^*(d^k + (d_{obs} - Md^k))), \end{aligned} \tag{6}$$

in which we use $M^* = M = (M)^2$ and $Md_{obs} = M^2 d = Md$. Now we define a residual term as $r^k = d_{obs} - Md^k$, thus Eq. (6) results in

$$\begin{cases} r^k \leftarrow d_{obs} - Md^k \\ d^{k+1} \leftarrow \Phi T_\lambda(\Phi^*(d^k + r^k)), \end{cases} \tag{7}$$

which is equivalent to solving

$$\min_d \frac{1}{2}\|d_{obs} - Md\|_2^2 + \lambda R(\Phi^* d). \tag{8}$$

Note that Eq. (8) analyzes the target unknown d directly, without resort to x and $d = \Phi x$. Eq. (6) is referred to as the analysis formula (Elad et al., 2007). In this paper, we used the analysis formula because it directly addresses the problem in the data domain for the convenience of interpolation and signal separation.

Understanding iterative thresholding as shaping regularization

Note that at each iteration soft thresholding is the only nonlinear operation corresponding to the ℓ_1 constraint for the model x, i.e., $R(x) = \|x\|_1$. Shaping regularization (Fomel, 2007, 2008) provides a general and flexible framework for inversion without the need for a specific penalty function $R(x)$ when a particular kind of shaping operator is used. The iterative shaping process can be expressed as

$$x^{k+1} = S(x^k + B(d_{obs} - Fx^k)), \tag{9}$$

where the shaping operator S can be a smoothing operator (Fomel, 2007), or a more general operator even a nonlinear sparsity-promoting shrinkage/thresholding operator (Fomel, 2008). It can be thought of a type of Landweber iteration followed by

projection, which is conducted via the shaping operator S. Instead of finding the formula of gradient with a known regularization penalty, we have to focus on the design of shaping operator in shaping regularization. In gradient-based Landweber iteration the backward operator B is required to be the adjoint of the forward mapping F, i.e., $B = F^*$; in shaping regularization however, it is not necessarily required. Shaping regularization gives us more freedom to choose a form of B to approximate the inverse of F so that shaping regularization enjoys faster convergence rate in practice. In the language of shaping regularization, the updating rule in Eq. (7) becomes

$$\begin{cases} r^k \leftarrow d_{obs} - Md^k, \\ d^{k+1} \leftarrow \Phi S(\Phi^*(d^k + r^k)), \end{cases} \tag{10}$$

where the backward operator is chosen to be the inverse of the forward mapping.

MCA using sparsity-promoting shaping

MCA considers the complete data d to be the superposition of several morphologically distinct components: $d = \sum_{i=1}^{N} d_i$. For each component d_i, MCA assumes there exists a transform Φ_i which can sparsely represent component d_i by its coefficients α_i ($\alpha_i = \Phi_i^* d_i$ should be sparse), and can not do so for the others. Mathematically,

$$\min_{\{d_i\}} \sum_{i=1}^{N} R(\Phi_i^* d_i), \text{subject to } d_{obs} = M \sum_{i=1}^{N} d_i. \tag{11}$$

The above problem can be rewritten as

$$\min_{\{d_i\}} \frac{1}{2} \left\| d_{obs} - M \sum_{i=1}^{N} d_i \right\|_2^2 + \lambda \sum_{i=1}^{N} R(\Phi_i^* d_i). \tag{12}$$

We prefer to rewrite Eq. (12) as

$$\begin{aligned} \min_{\{d_i\}} \frac{1}{2} \left\| \left(d_{obs} - M \sum_{j \neq i} d_j \right) - M d_i \right\|_2^2 \\ + \lambda R(\Phi_i^* d_i) + \lambda \sum_{j \neq i} R(\Phi_j^* d_j). \end{aligned} \tag{13}$$

Thus, optimizing with respect to d_i leads to the analysis IST shaping as Eq. (9). At the kth iteration, optimization is performed alternatively for many components using the block coordinate relaxation (BCR) technique (Bruce et al., 1998): for the ith component d_i^k, $i = 1, \ldots, N$: $\Phi \leftarrow \Phi_i$, $d^k \leftarrow d_i^k$, $d^{k+1} \leftarrow d_i^{k+1}$, $d_{obs} \leftarrow d_{obs} - M \sum_{j \neq i} d_j^k$, yields the residual term $r^k = d_{obs} - M \sum_{i=1}^{N} d_i^k$ and the updating rule

$$\begin{cases} r^k \leftarrow d_{obs} - M \sum_{i=1}^{N} d_i^k \\ d_i^{k+1} \leftarrow \Phi_i S(\Phi_i^*(d_i^k + r^k)). \end{cases} \tag{14}$$

The final output of the above algorithm are the morphological components $\hat{d}_i, i = 1, \ldots, N$. The complete data can then be reconstructed via $\hat{d} = \sum_{i=1}^{N} \hat{d}_i$. This is the main principle of the so-called MCA-based inpainting algorithm (Elad et al., 2005).

SEISLET-BASED MCA SPARSIFIED WITH SCALE-DEPENDENT EXPONENTIAL SHRINKAGE

Seislet transform and local slope estimation

Seislet transform and seislet frame were proposed by Fomel and Liu (2010) for seismic data compression and sparse representation. Seislets are constructed by applying the wavelet lifting scheme (Sweldens, 1998) along the local slope direction. For each level of lifting decomposition, seismic data is split into even and odd parts (**e** and **o**). Then the prediction and update step follows to obtain the detail difference/residual **d** and smooth information **s**:

$$\mathbf{d} = \mathbf{e} - P[\mathbf{o}], \mathbf{s} = \mathbf{e} + U[\mathbf{d}]. \tag{15}$$

Recognizing that seismic data can be organized as collections of traces or records, Fomel and Liu (2010) suggest prediction of one seismic trace or record from its neighbors and update of records on the next scale to follow structural features in seismic data. In the Z-transform notation, the simplest Haar prediction filter can be written as

$$P(Z) = Z, \tag{16}$$

and the linear interpolation prediction filter is

$$P(Z) = (Z + 1/Z)/2. \tag{17}$$

Successful prediction and update play a key role for local slope estimation. By modifying the biorthogonal wavelet construction, the prediction and update operators for a simple seislet transform are defined as

$$\begin{aligned} P[\mathbf{e}]_k &= (S_k^+[\mathbf{e}_{k-1}] + S_k^-[\mathbf{e}_k])/2, \\ U[\mathbf{r}]_k &= (S_k^+[\mathbf{r}_{k-1}] + S_k^-[\mathbf{r}_k])/4, \end{aligned} \tag{18}$$

where S_k^+ and S_k^- are the operators that predict a trace from its left and right neighbors, corresponding to shifting seismic events in terms of their local slopes. The job of local slope estimation can be done by PWD filters. Particularly, it is possible to obtain two or more dips with the help of PWD filters to capture different geometrical components of seismic data. The estimation of slopes involves a least-square optimization problem to be solved (Fomel, 2002), leading to extra computation. It is important to point out that besides PWD, there are other approaches to estimating dips of seismic data, i.e., local slant stack (Ottolini, 1983) and volumetric scan (Marfurt, 2006). However, PWD implements slope estimation through prediction and therefore is appropriate for use with the seislet transform.

Sparsifying MCA with exponential shrinkage shaping

The IST algorithm used by MCA requires soft thresholding function to filter out the unwanted small values. Besides soft thresholding (Donoho, 1995), many other

shrinkage functions can also be applied to obtain possibly better sparseness. One particular choice is hard thresholding:

$$t_\lambda(u) = \text{Hard}_\lambda(u) = u. * (|u| > \lambda ? 1 : 0). \tag{19}$$

where $(\cdot)?A : B$ frequency used hereafter is an if-else judgment in C-code style: The expression equals A if the statement $(\cdot)$ is true, and B otherwise.

Another choice is Stein thresholding (Peyre, 2010; Mallat, 2009):

$$t_\lambda(u) = \text{Stein}_\lambda(u) = u. * \max\left(1 - (\frac{\lambda}{|u|})^2, 0\right). \tag{20}$$

Stein thresholding does not suffer from the bias of soft thresholding, that is,

$$|\text{Stein}_\lambda(u) - u| \to 0, |\text{Soft}_\lambda(u) - u| \to \lambda, \text{if } u \to \infty. \tag{21}$$

Recent advances in nonconvex optimization (Chartrand, 2012; Voronin and Chartrand, 2013; Chartrand and Wohlberg, 2013) show that the shrinkage operator in IST algorithm (Eq. (2)) can be generalized to a p-quasinorm ($0 < p \leq 1$) thresholding operator T_λ, in which

$$t_\lambda(u) = \text{pThresh}_{\lambda,p}(u) = u. * \max\left(1 - (\frac{\lambda}{|u|})^{2-p}, 0\right). \tag{22}$$

A special case is that of $p = 1$, which corresponds to the soft thresholding operator exactly.

Most of these shrinkage functions interpolate between the hard and soft thresholders. It is tempting for us to design a more general shrinkage function to sparsify the transform domain coefficients in shaping regularized MCA. One possibility is multiplying an exponential factor on the elements of original data:

$$t_\lambda(u) = u. * \exp(-(\frac{\lambda}{|u|})^{2-p}). \tag{23}$$

Based on Taylor series, this operator in Eq. (23) enjoys some useful properties:

- It is valuable to point out that the exponential shrinkage can be considered as a smooth ℓ_0 constraint (Mohimani et al., 2009; Gholami and Hosseini, 2011). For $|u| >> \lambda$, it is a good approximation of the p-thresholding operator in Eq. (22), and does not suffer the bias when $p \neq 1$. It reduces to Stein thresholding operator for $p = 0$ and soft thresholding for $p = 1$.

$$\begin{aligned} t_{\lambda,0}(u) &= u. * \exp(-(\frac{\lambda}{|u|})^2) \approx u. * (1 - (\frac{\lambda}{|u|})^2), \\ t_{\lambda,1}(u) &= u. * \exp(-(\frac{\lambda}{|u|})) \approx u. * (1 - (\frac{\lambda}{|u|})). \end{aligned} \tag{24}$$

- It is free of non-differentiable singularity at the thresholding point λ. The transition between the small values and the large values is smoothly stretched. Due to the exponential factor less than 1 ($\exp(-(\frac{\lambda}{|u|})^{2-p}) < 1$), this operator will slightly decrease the data amplitude, even for $|u| < \lambda$.
- Besides the threshold λ, we have another independent parameter p which can be flexibly chosen to achieve better performance.

In the language of shaping regularization, shrinkage-based shaping operator S is equivalent to multiplying the coefficient vector x by a diagonal weighting matrix W to in the sense that

$$S(x) = Wx, \tag{25}$$

where

$$\mathrm{diag}(W_{ii}) = \begin{cases} 1 - \frac{\lambda}{|x_i|} > 0?1:0, & \text{Hard} \\ \max(1 - \frac{\lambda}{|x_i|}, 0), & \text{Soft} \\ \max(1 - \left(\frac{\lambda}{|x_i|}\right)^2, 0), & \text{Stein} \\ \max(1 - \left(\frac{\lambda}{|x_i|}\right)^{2-p}, 0), & \text{pThresh} \\ \exp(-(\frac{\lambda}{|x_i|})^{2-p}), & \text{Exponential} \end{cases} \tag{26}$$

For the convenience of comparison, we plot these thresholding operators in Figure 1.

Note that we are using seislet transform which has different scales for signal representation. Usually, large scales of seislet coefficients corresponds to unpredictable noise, while most of the important information gets transformed into smaller scales. We design a scale-dependent diagonal weighting operator:

$$\mathrm{diag}(W_{ii}) = s(x_i) < s_0?1:0, \tag{27}$$

where s_0 is user-defined scale, while $s(x_i)$ is the scale that the coefficient x_i correspond to. Putting all things together, in the MCA shaping regularization we are using a scale-dependent exponential shrinkage operator which is a composite operator cascaded with a scale-muting operator W_s and an exponential weighting operator W_{exp}:

$$S(x) = W_{exp} W_s x. \tag{28}$$

The use of scale-dependent exponential shrinkage offers easy control on the separation of the signal components we would like to capture. It is interesting to mention that under the Fourier basis, the scale-muting operator W_s becomes a frequency mask (it behaves like a selective hard thresholding), which can be employed to remove the groundroll in application to seismic interpolation (Gholami, 2014).

By incorporating PWD dip estimation and scale-dependent exponential shrinkage shaping, we summarize the proposed seislet-based MCA algorithm as Algorithm MCA. Seislet transforms associated with different dips form a combined seislet frame (Fomel and Liu, 2010). The threshold in each iteration can be determined with a

predefined percentile according to Hoare's algorithm. Shrinkage operator plays the role of crosstalk removal in MCA algorithm, as explained in more detail in Appendix A.

Figure 1: A schematic plot of the shrinkage operators,$\lambda = 1$

Algorithm 1 Seislet-based MCA algorithm

Input: Observed seismic image d_{obs}; sampling mask M; iteration number $niter$; shrinkage shaping parameter p, seislet transform Φ_i associated with the ith estimated slope.

Output: Separated seismic component d_i.

Initialize: $d_i^{(k)} \leftarrow 0$, $i = 1 \dots N$;
for $k = 1 \dots niter$ **do**
  $r^{(k)} \leftarrow d_{obs} - M \sum_{i=1}^{N} d_i^{(k)}$;
  for $i = 1 \dots N$ **do**
    $x_i^{(k)} \leftarrow \Phi_i^*(d_i^{(k)} + r_i^{(k)})$;
    Estimate shaping parameters λ and s_0;
    $d_i^{(k)} \leftarrow \Phi_i S(x_i^k)$;
  end for
end for

NUMERICAL EXAMPLES

Trace interpolation

Interpolation of random missing traces is an important task in seismic data processing. Unlike most studies using only one transform, we consider the seismic data having

two different components, which can be characterized by a seislet frame composed of seislet transforms associated with two different slopes. PWD filter is utilized to estimate the two dip fields (Fomel, 2002). As shown in Figure 2, the complete data is decimated with a random eliminating rate 25%. 10 iterations are carried out to separate these components. The estimated dips (Figure 3) by PWD indicate that the two separated components exhibit different modes: component 2 has positive and negative dips, corresponding to seismic diffractions, while the events of component 1 are more consistent (most values of the dip are positive). The summation of the two components gives a reasonable interpolation result (right panel of Figure 4.) For comparison, we define the signal-to-noise ratio as $\mathrm{SNR} = 10\log_{10}(\frac{|s|^2}{|s-\hat{s}|^2})$ to quantify the reconstruction performance. The resulting SNRs using exponential shrinkage, soft, hard, and generalized p-thresholding are 11.98 dB, 11.29 dB, 5.37 dB and 8.94 dB, respectively. It shows that the proposed exponential shrinkage outperforms the existing methods in MCA interpolation. It is important to point out that approximating the ℓ_0 and ℓ_1 minimization in a smooth constraint has already been validated in seismic interpolation applications by Cao et al. (2011). The use of exponential shrinkage enriches the sparsity-promoting shaping operator and extends the smooth constraint to MCA approach.

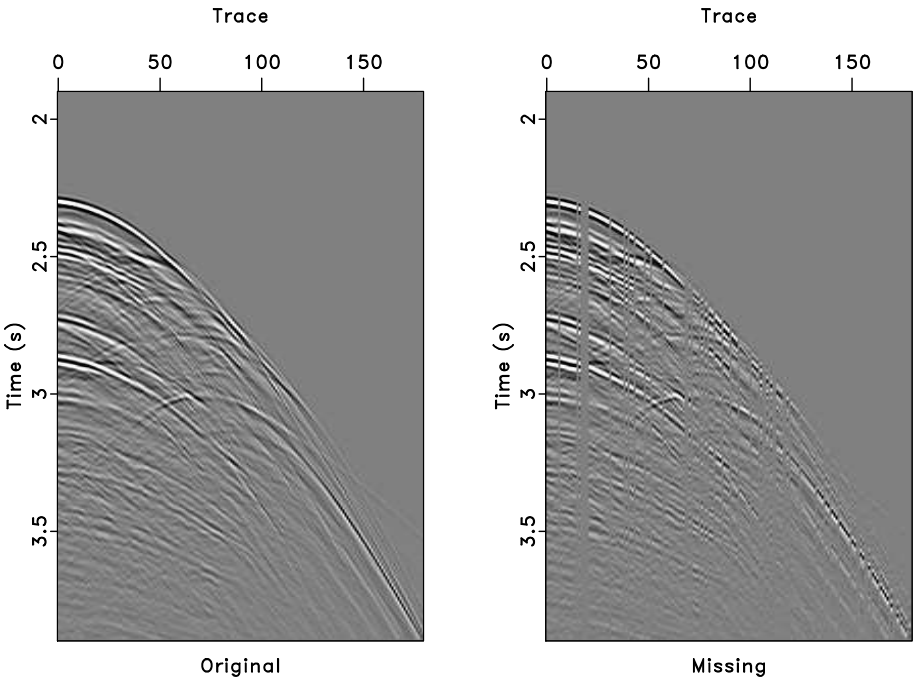


Figure 2: The observed seismic data (right) to be interpolated is obtained by 25% random eliminating the complete data (left). mcaseislet/interp/ data

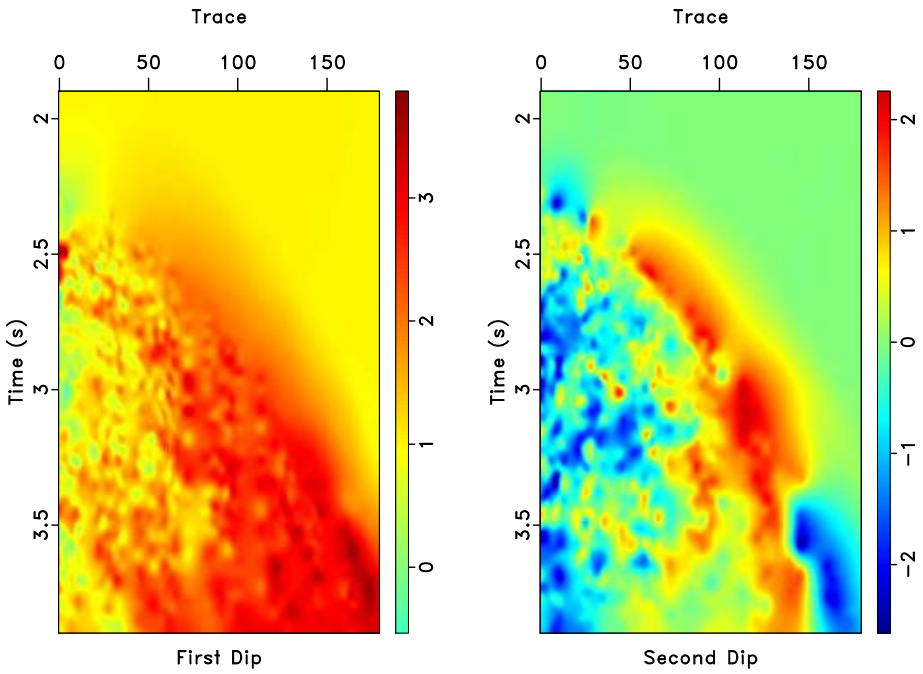


Figure 3: PWD estimated dips: dip estimation for component 1 (Left) are positive-valued, while dip for component 2 includes negative and positive values. mcaseislet/interp/ dips

Multiple removal

Our second example is the separation of primaries and multiples for the field CMP gather shown in Figure 5 (Fomel and Guitton, 2006). In the case of signal separation, the mask operator M becomes an identity. The multiples are predicted using surface-related multiple elimination (SRME). Even though SRME fails to predict the correct amplitudes, however, the resulting prediction helps PWD to extract the dominant slopes of multiple events. Before applying the seislet-based MCA method, it is important to point out that the iteratively reweighted least-squares (IRLS) method using the model precondition is another way for sparsity-enforced separation (Daubechies et al., 2010). Thus, we compared multiple removal by two different methods: seislet-based IRLS method and the proposed seislet-based MCA method. For comparison, the separated primaries and multiples using different methods are plotted in Figures 6 and 7. Visually, the seislet-based IRLS method and the seislet-based MCA method output similar primaries, which can also be seen clearly from the velocity scans of the primaries shown in Figure 8. To further confirm our conclusion, we draw the velocity scans of predicted multiples in Figure 9. The panels of velocity scan in Figures 8 and 9 demonstrate that using MCA, the primaries correctly correspond to high velocity part while the multiples are associated with low velocity part. Figure 9 shows that seislet-based MCA outperforms the seislet-based IRLS method in the locations A and B in the velocity scan panel due to the nice match of the corresponding semblance

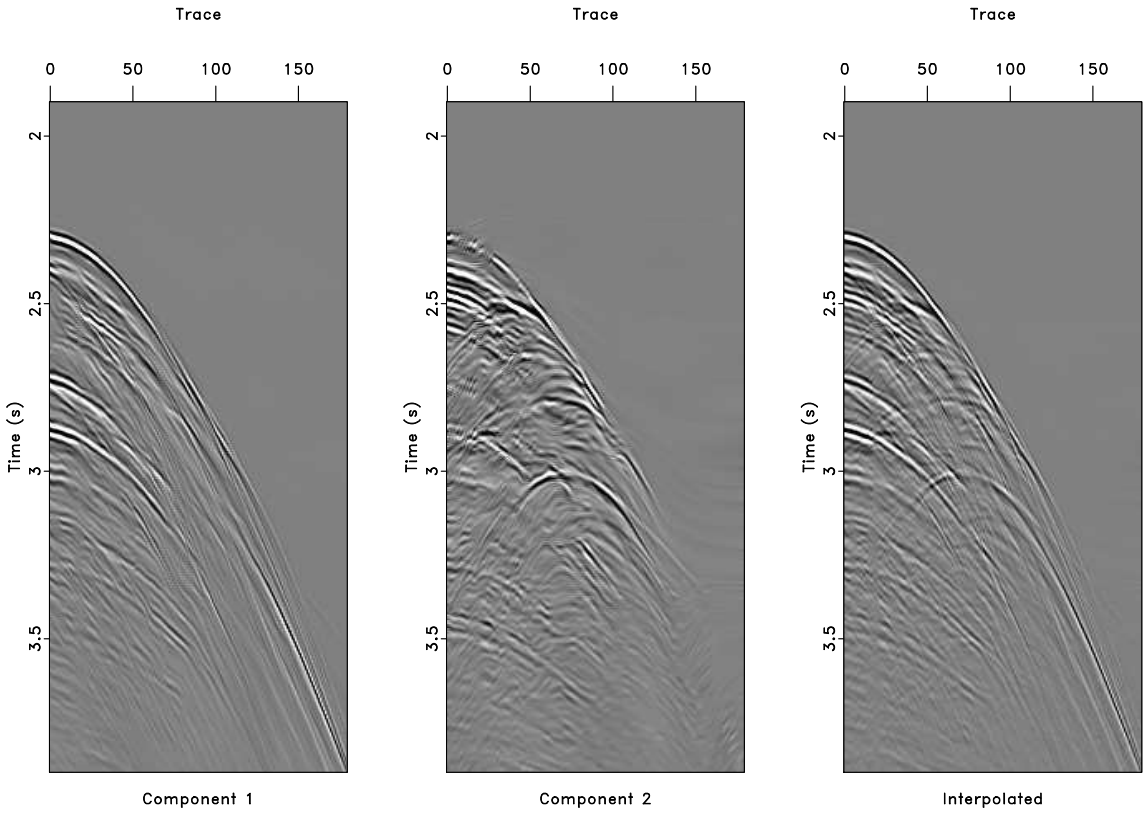


Figure 4: From left to right: MCA reconstructed component 1, component 2 and the final interpolated data. mcaseislet/interp/ interp

scan of the original data; at locations C and D, the energy of multiples obtained by seislet-MCA has less leakage, compared to the seislet-IRLS method. Note that the seislet-based MCA algorithm only uses 15 iterations to obtain the best separation effect, while the number of iterations for seislet-IRLS method is 1000. Therefore, seislet-based MCA is very efficient to demultiple.

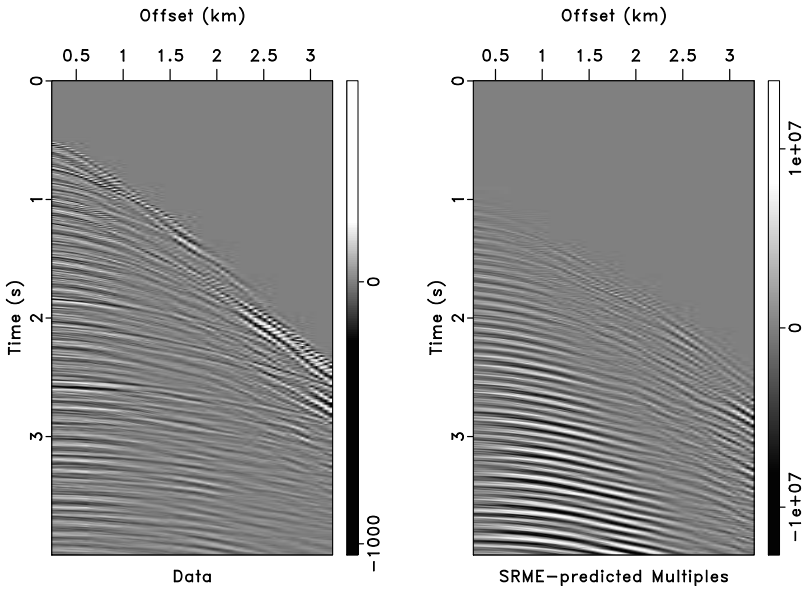


Figure 5: The field CMP data (left) and SRME predicted multiples (right). The amplitudes of SRME prediction needs to be corrected. mcaseislet/sep2/ srme

CONCLUSION AND DISCUSSION

We have developed a seislet-based MCA method for seismic data processing. PWD filter can be utilized to estimate the slopes of seismic data. An exponential shrinkage function is introduced to diversify the capability of sparsity-promoting shaping operator. The proposed seislet-based MCA using scaled-dependent shaping regularization is promising in the application to seismic trace interpolation, and multiple removal. The numerical results reveal that the exponential shrinkage operator in sparsity-promoting shaping regularization plays an extremely important role in successful seislet-based MCA separation, superior to many existing thresholding operators. The additional parameter p provides us more flexibility for approximating many existing shrinkage operators to achieve better separation performance. Meanwhile, it is free of non-differential singularity and unifies many existing shrinkage operators.

Seislet-MCA using PWD-based dip estimation is of special physical meaning for geophysical data in seismic processing, while the sparse dictionaries reported in Starck et al. (2004) are useful in image processing but lacking seismic attributes. However, a computational expensive optimization problem using least-squares minimization, which is not involved in the method of Starck et al. (2004), has to be solved to estimate

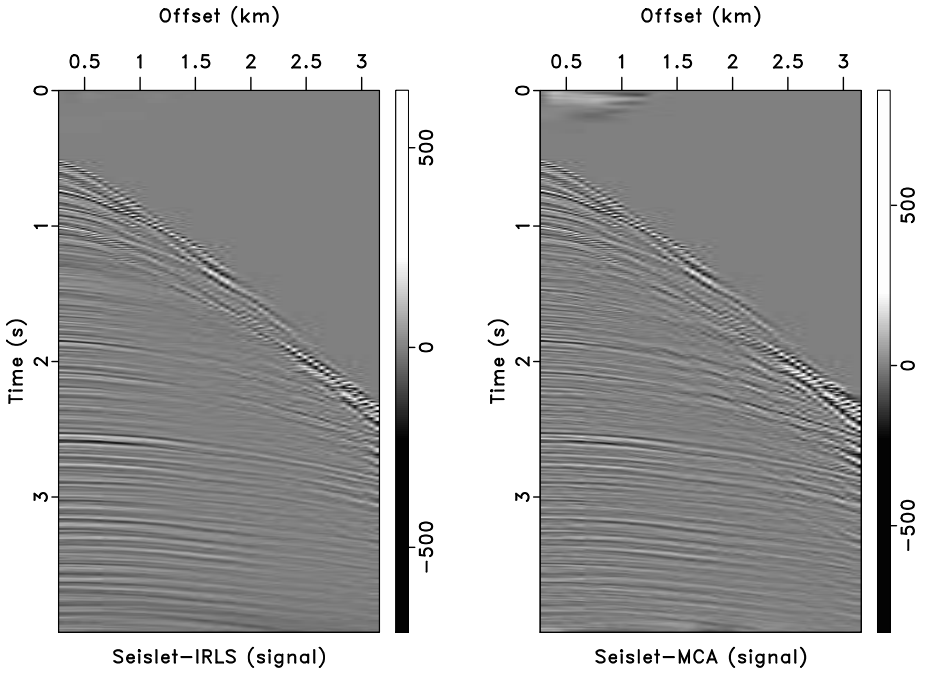


Figure 6: Separated primaries using seislet-IRLS (1000 iterations) and seislet-based MCA (15 iterations). mcaseislet/sep2/ signal

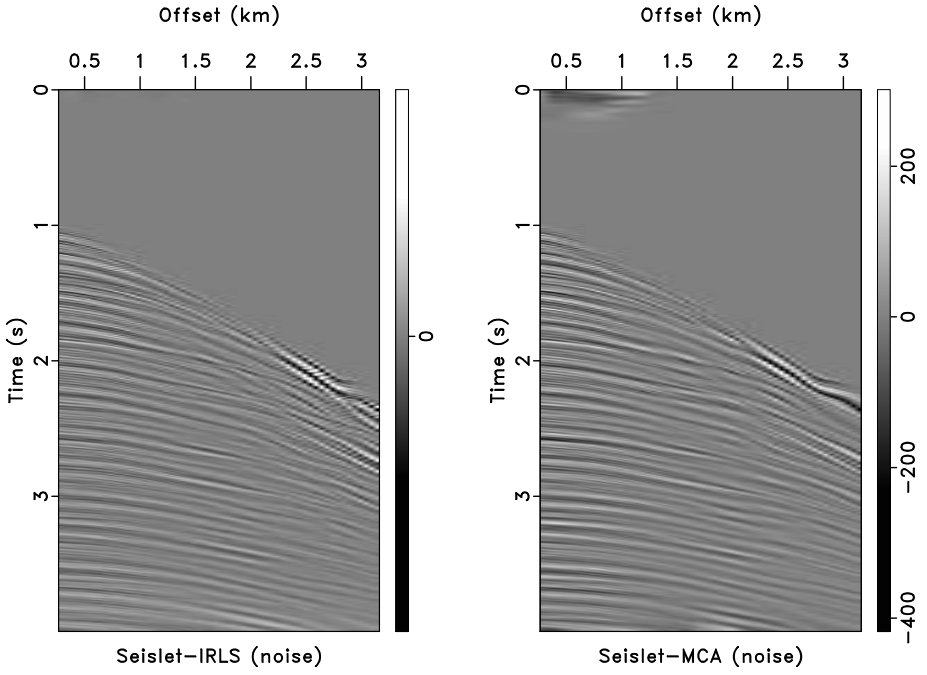


Figure 7: Separated multiples using seislet-IRLS (1000 iterations) and seislet-based MCA (15 iterations). mcaseislet/sep2/ nois

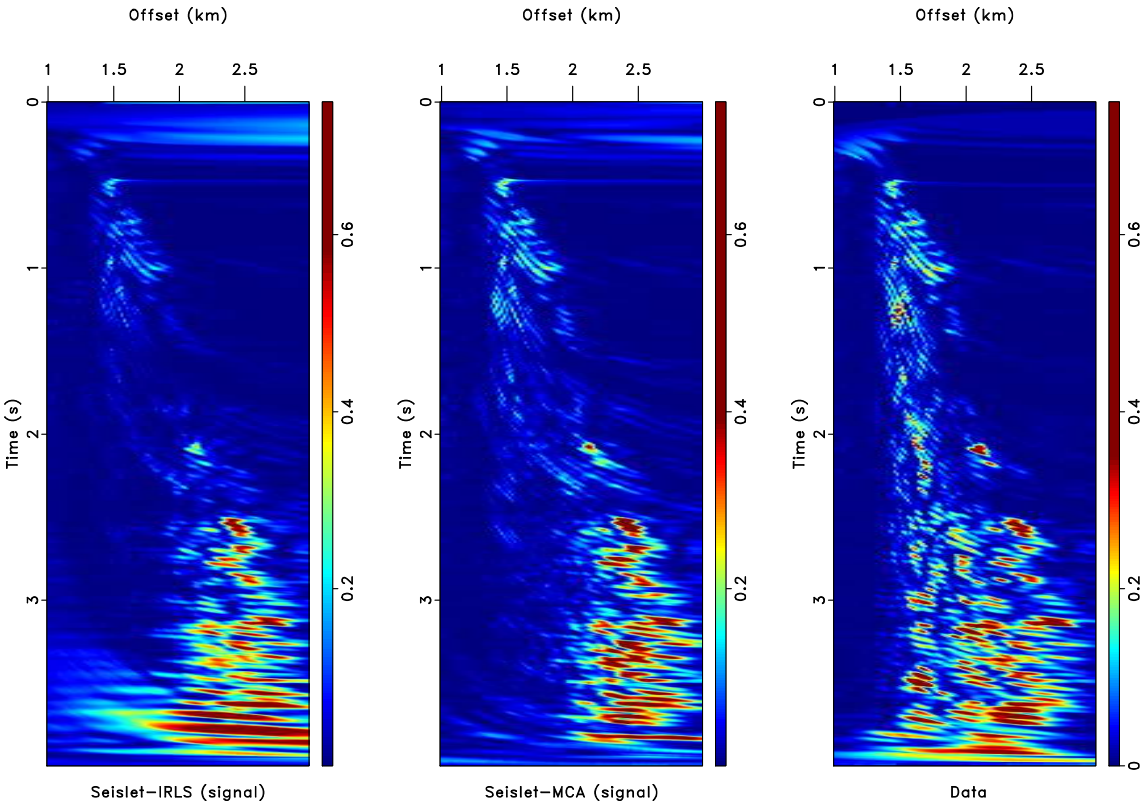


Figure 8: Velocity scan for primaries obtained by seislet-IRLS and seislet-based MCA (first two panels, from left to right). The velocity scan of original data is shown in the last panel for comparison. mcaseislet/sep2/ vsignal

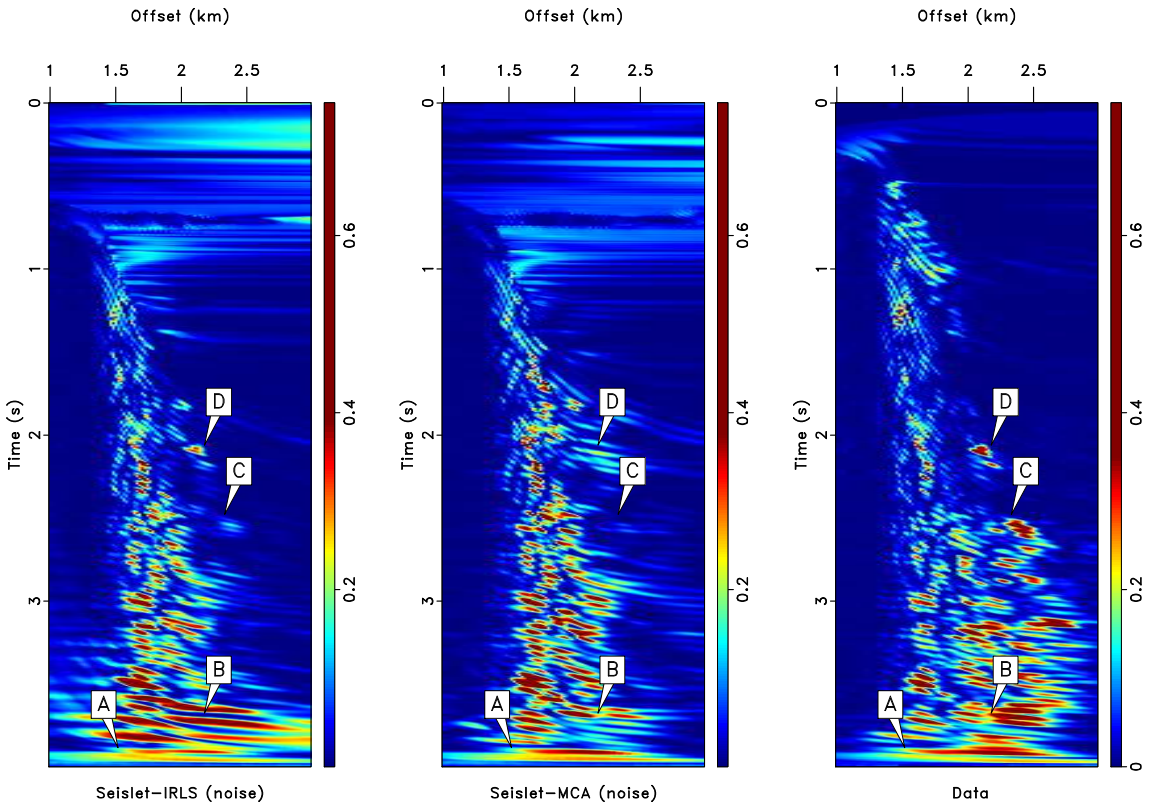


Figure 9: Velocity scan for multiples obtained by seislet-IRLS and seislet-based MCA (first two panels, from left to right). The velocity scan of original data is shown in the last panel for comparison. Seislet-based MCA outperforms seislet-based IRLS method in the locations A and B in the velocity scan panel due to the nice match of the corresponding semblance scan of the original data; at locations C and D, the energy of multiples obtained by seislet-MCA has less leakage, compared to seislet-IRLS method. mcaseislet/sep2/ vnois

the slope fields before applying our seislet-based method. Besides the computational expensive slope estimation, the proposed method is very efficient for interpolation and separation. The method fails to interpolate the missing traces when the random decimating rate is larger than 70% for 2D seismic data, which honors the necessity of high dimensional data reconstruction using 3D seislet transform. Although numerically working well, up to now we have no theoretical convergence proof of the nonlinear shaping algorithm, and it remains an open problem for future works.

ACKNOWLEDGMENTS

The work of the first author was supported by the China Scholarship Council (No. 201306280082) during his visit to the Bureau of Economic Geology and the University of Texas at Austin. We would like to thank Wenchao Chen and Jianwei Ma for valuable suggestions. Reproducible examples are created using the Madagascar software package (Fomel et al., 2013).

CONNECTIONS BETWEEN SEISLET FRAME AND SEISLET-MCA ALGORITHM

The complete data d is regarded to be superposition of several different geometrical components, and each component can be sparely represented using a seislet dictionary Φ_i, i.e.,

$$\begin{aligned} d &= \sum_{i=1}^{N} d_i = \sum_{i=1}^{N} \Phi_i \alpha_i \\ &= [\Phi_1, \Phi_2, \cdots, \Phi_N] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} \end{aligned} \tag{A-1}$$

where $F = [\Phi_1, \Phi_2, \cdots, \Phi_N]$ is a combined seislet dictionary (i.e. seislet frame), and the backward operator is chosen to be

$$B = \frac{1}{N} \begin{bmatrix} \Phi_1^* \\ \Phi_2^* \\ \vdots \\ \Phi_N^* \end{bmatrix} \tag{A-2}$$

in the sense that

$$FB = \frac{1}{N} \sum_{i=1}^{N} \Phi_i \Phi_i^* = \mathrm{Id}. \tag{A-3}$$

The difference between seislet-MCA algorithm and seislet frame minimization is the use of BCR technique (Bruce et al., 1998): We sparsify one component while keeping

all others fixed. At the $k+1$-th iteration applying the backward operator on the i-th component leads to

$$\tilde{\alpha}_i^{k+1} = \alpha_i^k + \sum_{i=1}^{N} \Phi_i^* r^k = \alpha_i^k + r_i^j + \sum_{j \neq i} \Phi_i^* r_j^k \tag{A-4}$$

where the terms $\sum_{j \neq i} \Phi_i^* r_j^k$ are the crosstalk between the i-th component and the others. An intuitive approach to filter out the undesired crosstalk is shrinkage/thresholding. The proposed exponential shrinkage provides us a flexible control on the performance of the shrinkage/thresholding operator.

REFERENCES

Bruce, A., S. Sardy, and P. Tseng, 1998, Block coordinate relaxation methods for nonparamatric signal denoising: Proceedings of SPIE, **3391**, 75–86.

Cao, J., Y. Wang, J. Zhao, and C. Yang, 2011, A review on restoration of seismic wavefields based on regularization and compressive sensing: Inverse Problems in Science and Engineering, **19**, 679–704.

Chartrand, R., 2012, Nonconvex splitting for regularized low-rank + sparse decomposition: IEEE Transactions on Signal Processing, **60**, 5810–5819.

Chartrand, R., and B. Wohlberg, 2013, A nonconvex admm algorithm for group sparsity with sparse groups: Presented at the Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Chen, Z., S. Fomel, and W. Lu, 2013a, Accelerated plane-wave destruction: Geophysics, **78**, V1–V9.

———, 2013b, Omnidirectional plane-wave destruction: Geophysics, **78**, V171–V179.

Claerbout, J. F., 1992, Earth soundings analysis: Processing versus inversion: Blackwell Scientific Publications Cambridge, Massachusetts, USA, **6**.

Daubechies, I., M. Defrise, and C. De Mol, 2004, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint: Commun. Pure Appl. Math., **57**, 1413–1457.

Daubechies, I., R. DeVore, M. Fornasier, and S. Gunturk, 2010, Iteratively reweighted least squares minimization for sparse recovery: Commun. Pure Appl. Math., **63**.

Donoho, D., 1995, De-noising by soft-thresholding: IEEE Transactions on Information Theory, **41**, 613–627.

Elad, M., P. Milanfar, and R. Rubinstein, 2007, Analysis versus synthesis in signal priors: Inverse Probl., **23**, 947–968.

Elad, M., J. Starck, P. Querre, and D. L. Donoho, 2005, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA): Applied and Computational Harmonic Analysis, **19**, 340 – 358.

Fomel, S., 2002, Applications of plane-wave destruction filters: Geophysics, **67**, 1946–1960.

———, 2007, Shaping regularization in geophysical-estimation problems: Geophysics, **72**, R29–R36.

———, 2008, Nonlinear shaping regularization in geophysical inverse problems: SEG Annual Meeting, 2046–2051.

Fomel, S., and A. Guitton, 2006, Regularizing seismic inverse problems by model reparameterization using plane-wave construction: Geophysics, **71**, A43–A47.

Fomel, S., and Y. Liu, 2010, Seislet transform and seislet frame: Geophysics, **75**, V25–V38.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8.

Gholami, A., 2014, Non-convex compressed sensing with frequency mask for seismic data reconstruction and denoising: Geophysical Prospecting, **62**, 1389–1405.

Gholami, A., and S. Hosseini, 2011, A general framework for sparsity-based denoising and inversion: IEEE Transactions on Signal Processing, **59**, 5202–5211.

Mallat, S., 2009, A wavelet tour of signal processing, 3rd ed.: Academic Press.

Marfurt, K. J., 2006, Robust estimates of 3D reflector dip and azimuth: Geophysics, **71**, P29–P40.

Mohimani, G. H., M. Babaie-Zadeh, and C. Jutten, 2009, A fast approach for overcomplete sparse decomposition based on smoothed l-0 norm: IEEE Transactions on Signal Processing, **57**, 289–301.

Ottolini, R., 1983, Signal/noise separation in dip space: Stanford Exploration Project: SEP report, **3**, 143–149.

Peyre, G., 2010, Advanced image, signal and surface processing.

Starck, J., M. Elad, and D. Donoho, 2004, Redundant multiscale transforms and their application for morphological component separation: Advances in Imaging and Electron Physics, **132**, 287–348.

———, 2005, Image decomposition via the combination of sparse representations and a variational approach: IEEE Trans. Image Process., **14**, 1570–1582.

Starck, J., J. Fadili, and F. Murtagh, 2007, The undecimated wavelet decomposition and its reconstruction: IEEE Trans. Image Process., **16**, 297–309.

Sweldens, W., 1998, The lifting scheme: A construction of second generation wavelets: SIAM Journal on Mathematical Analysis, **29**, 511–546.

Voronin, S., and R. Chartrand, 2013, A new generalized thresholding algorithm for inverse problems with sparsity constraints: 38th International Conference on Acoustics, Speech, and Signal Processing, IEEE, 1636–1640.

Woiselle, A., J. Starck, and J. Fadili, 2011, 3-D data denoising and inpainting with the low-redundancy fast curvelet transform: J. Math. Imaging Vis., **39**, 121–139.

A numerical tour of wave propagation

Pengliang Yang[*]

ABSTRACT

This tutorial is written for beginners as an introduction to basic wave propagation using finite difference method, from acoustic and elastic wave modeling, to reverse time migration and full waveform inversion. Most of the theoretical delineations summarized in this tutorial have been implemented in Madagascar with Matlab, C and CUDA programming, which will benefit readers' further study.

BASIC WAVE EQUATION

Define $\mathbf{x} = (x, y, z)$, time t, the s-th energy source function $\tilde{S}(\mathbf{x}, t; \mathbf{x}_s)$, pressure $p(\mathbf{x}, t; \mathbf{x}_s)$, particle velocity $\mathbf{v}(\mathbf{x}, t)$, material density $\rho(\mathbf{x})$, the bulk modulus $\kappa(\mathbf{x})$. Now we have

- Newton's law

$$\rho(\mathbf{x}) \frac{\partial \mathbf{v}(\mathbf{x}, t; \mathbf{x}_s)}{\partial t} = \nabla p(\mathbf{x}, t; \mathbf{x}_s). \tag{1}$$

- Constitutive law

$$\frac{1}{\kappa(\mathbf{x})} \frac{\partial p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t} = \nabla \cdot \mathbf{v}(\mathbf{x}, t; \mathbf{x}_s) + \tilde{S}(\mathbf{x}, t; \mathbf{x}_s). \tag{2}$$

Acoustic wave equation

Acoustics is a special case of fluid dynamics (sound waves in gases and liquids) and linear elastodynamics. Note that elastodynamics is a more accurate representation of earth dynamics, but most industrial seismic processing based on acoustic model. Recent interest in quasiacoustic anisotropic approximations to elastic P-waves.

Assume $\tilde{S}(\mathbf{x}, t; \mathbf{x}_s)$ is differentiable constitutive law w.r.t. time t. Substituting Eq. (2) into the differentiation of Eq. (1) gives

$$\frac{1}{\kappa(\mathbf{x})} \frac{\partial^2 p(\mathbf{x}, t; \mathbf{x}_s)}{\partial t^2} = \nabla \cdot \left(\frac{1}{\rho(\mathbf{x})} \nabla p(\mathbf{x}, t; \mathbf{x}_s) \right) + \frac{\partial \tilde{S}(\mathbf{x}, t; \mathbf{x}_s)}{\partial t}. \tag{3}$$

[*]**e-mail:** ypl.2100@gmail.com

We introduce $v(\mathbf{x}) = \sqrt{\kappa(\mathbf{x})/\rho(\mathbf{x})}$ (compressional p-wave velocity):

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = \rho(\mathbf{x})\nabla\cdot\left(\frac{1}{\rho(\mathbf{x})}\nabla p(\mathbf{x},t;\mathbf{x}_s)\right) + \rho(\mathbf{x})\frac{\partial \tilde{S}(\mathbf{x},t;\mathbf{x}_s)}{\partial t} \tag{4}$$

Under constant density condition, we obtain the 2nd-order equation

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = \nabla^2 p(\mathbf{x},t;\mathbf{x}_s) + f_s(\mathbf{x},t;\mathbf{x}_s) \tag{5}$$

where $\nabla^2 = \nabla\cdot\nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}$, $f_s(\mathbf{x},t;\mathbf{x}_s) = \rho(\mathbf{x})\frac{\partial \tilde{S}(\mathbf{x},t;\mathbf{x}_s)}{\partial t}$. In 2D case, it is

$$\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = v^2(\mathbf{x})\left(\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial z^2} + \frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial x^2}\right) + f_s(\mathbf{x},t;\mathbf{x}_s). \tag{6}$$

A shot of acoustic wavefield obtained at t=0.35s with 4-th order finite difference scheme and the sponge absorbing boundary condition is shown in Figure 1, where the source is put at the center of the model. For 3D, it becomes

$$\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = v^2(\mathbf{x})\left(\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial z^2} + \frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial x^2} + \frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial y^2}\right) + f_s(\mathbf{x},t;\mathbf{x}_s) \tag{7}$$

Similarly, we put the source at the center of a 3D volume (size=100x100x100), performed the modeling for 300 steps in time and recorded the corresponding wavefield at kt=250, see Figure 2.

The above spatial operator is spatially homogeneous. This isotropic formula is simple and easy to understand, and becomes the basis for many complicated generalizations in which the anisotropy may come in. In 2D case, the elliptically-anisotropic wave equation reads

$$\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} = v_1^2(\mathbf{x})\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial z^2} + v_2^2(\mathbf{x})\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial x^2} + f_s(\mathbf{x},t;\mathbf{x}_s) \tag{8}$$

Here, I use the Hess VTI model shown in Figure 3a and Figure 3b. We perform 1000 steps of modeling with time interval $\Delta t = 0.001s$, and capture the wavefield at $t = 0.9s$, as shown in Figure 4.

Elastic wave equation

In elastic wave equation, the modulus $\kappa(\mathbf{x})$ corresponds to two Lame parameters: $\lambda + 2\mu = \rho v_p^2$ and $u = \rho v_s^2$, in which v_p and v_s denote the P- and S-wave velocity. The

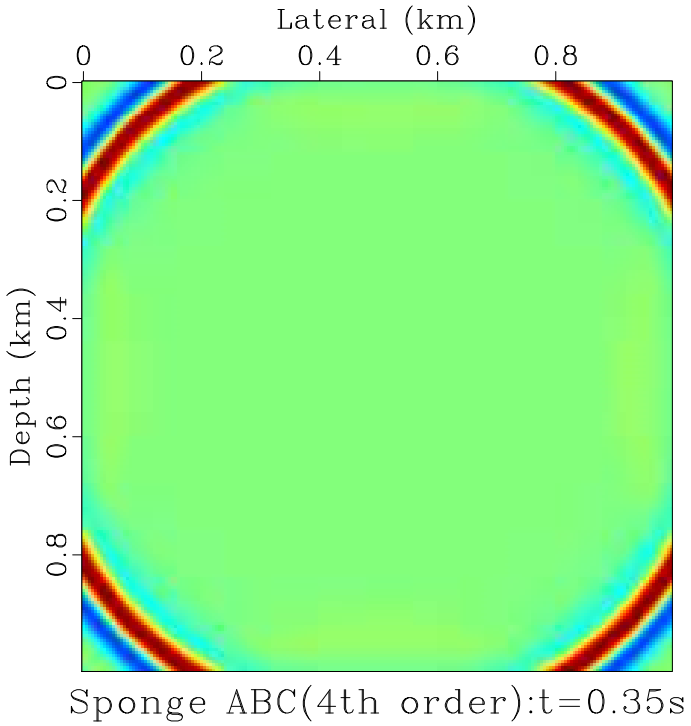


Figure 1: A snap of acoustic wavefield obtained at t=0.35s with 4-th order finite difference scheme and the sponge absorbing boundary condition. primer/testfd2d/ snapfd2d

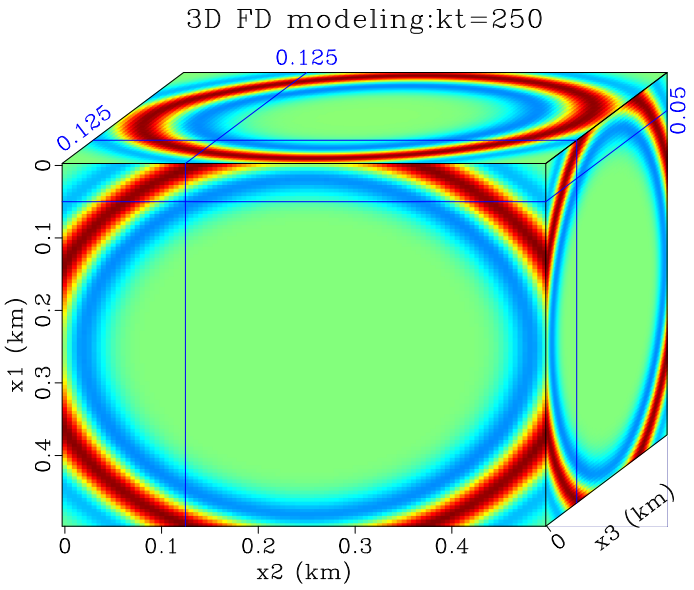


Figure 2: A wavefield snap recorded at kt=250, 300 steps modeled. primer/testfd3d/ snapfd3d

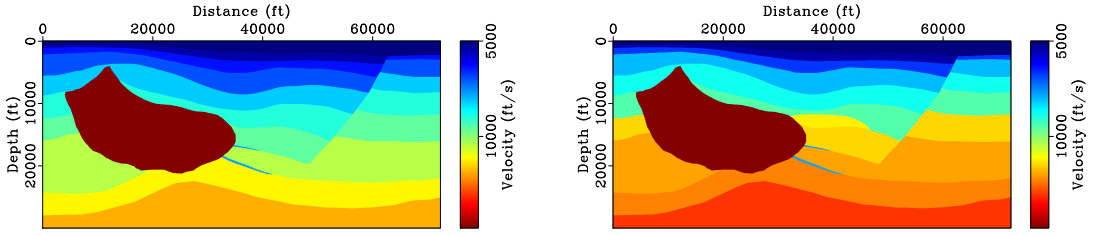


Figure 3: Two velocity components of Hess VTI model primer/testaniso/ vp,vx

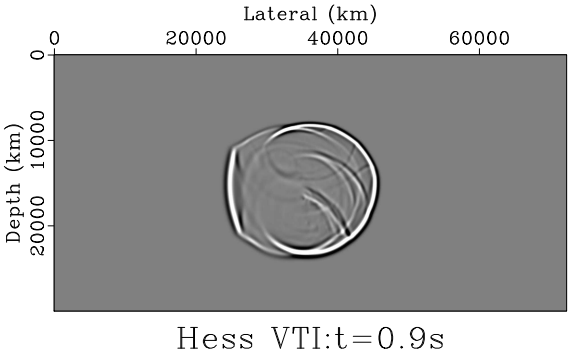


Figure 4: Wavefield at $kt = 0.9s$, 1000 steps of modeling with time interval $\Delta t = 0.001s$ performed. primer/testaniso/ snapaniso

elastic wave equation can be written as

$$\begin{cases} \frac{\partial v_x}{\partial t} = \frac{1}{\rho}(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial x}) \\ \frac{\partial v_z}{\partial t} = \frac{1}{\rho}(\frac{\partial \tau_{zx}}{\partial z} + \frac{\partial \tau_{zz}}{\partial z}) \\ \frac{\partial \tau_{xx}}{\partial t} = (\lambda + 2\mu)\frac{\partial v_x}{\partial x} + \lambda \frac{\partial v_z}{\partial z} \\ \frac{\partial \tau_{zz}}{\partial t} = \lambda \frac{\partial v_x}{\partial x} + (\lambda + 2\mu)\frac{\partial v_z}{\partial z} \\ \frac{\partial \tau_{xz}}{\partial t} = \mu \frac{\partial v_x}{\partial x} + \mu \frac{\partial v_z}{\partial z} \end{cases} \tag{9}$$

where τ_{ij} (sometimes σ_{ij}) is stress, v_i is particle velocity, $i, j = x, z$. We display the 2 components of elastic wave propagation at $kt = 270$, $nt = 300$ modeled with $\Delta t = 0.001$ in Figure 5, in which the grid size is 200x200, the spatial interval is $\Delta x = \Delta z = 5m$, and the velocities are chosen to be $Vp = 2km/s$, $Vs = Vp/\sqrt{2}$.

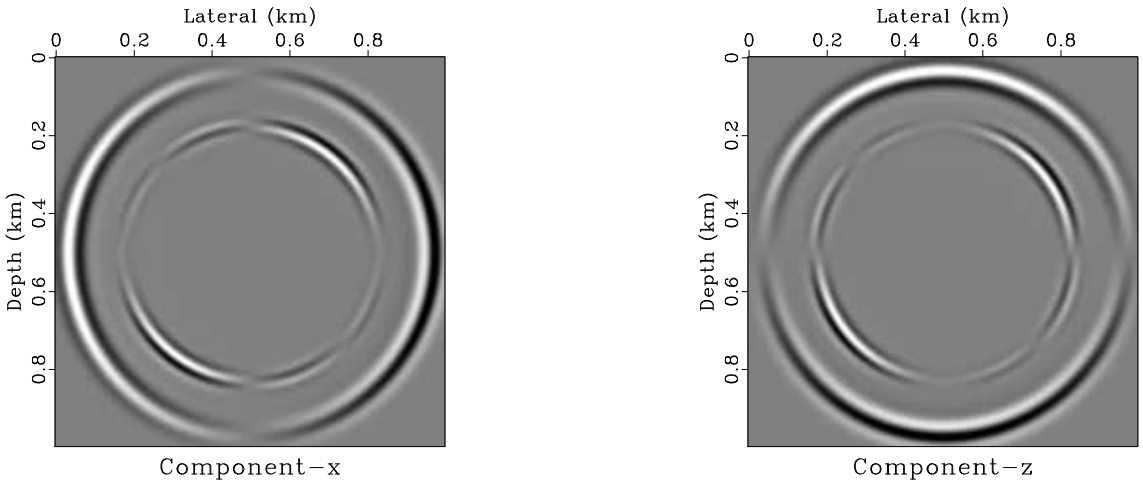


Figure 5: Two components of elastic wave propagation at $kt = 270$, $nt = 300$ modeled with $\Delta t = 0.001$. Grid size=200x200,$\Delta x = \Delta z = 5m$, $Vp = 2km/s$, $Vs = Vp/\sqrt{2}$ primer/testelastic2d/ elasticxz

FORWARD MODELING

Taylor and Páde expansion

The Taylor expansion of a function $f(x+h)$ at x is written as

$$f(x+h) = f(x) + \frac{\partial f(x)}{\partial x}h + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2}h^2 + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3}h^3 + \dots. \tag{10}$$

A popular example is

$$(1+x)^\alpha = 1 + \alpha x + \frac{\alpha(\alpha-1)}{2!}x^2 + \ldots + \frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!}x^n + \dots. \tag{11}$$

Here we mainly consider the following expansion formula:

$$(1-x)^{\frac{1}{2}} = 1 - \frac{1}{2}x - \frac{1}{8}x^2 - \frac{1}{16}x^3 - \frac{5}{128}x^4 - \ldots, |x| < 1. \tag{12}$$

The Páde expansion of Eq. (12) follows from expansion in continuous fractions:

$$(1-x)^{\frac{1}{2}} = 1 - \cfrac{x/2}{1 - \cfrac{x/4}{1 - \cfrac{x/4}{\dots}}} \tag{13}$$

I provide an informal derivation:

$$y=(1-x)^{\frac{1}{2}}\Rightarrow x=1-y^2=(1-y)(1+y)\Rightarrow 1-y=\frac{x}{1+y}$$
$$y=1-\frac{x}{1+y}=1-\frac{x}{1+(1-\frac{x}{1+y})}=1-\frac{x/2}{1-\frac{x/2}{1+y}}$$
$$=1-\frac{x/2}{1-\frac{x/2}{2-\frac{x}{1+y}}}=1-\frac{x/2}{1-\frac{x/4}{1-\frac{x/2}{1+y}}}=\dots$$

The 1st-order Pade expansion is:

$$(1-x)^{\frac{1}{2}}=1-\frac{x}{2} \tag{14}$$

The 2nd-order Pade expansion is:

$$(1-x)^{\frac{1}{2}}=1-\frac{x/2}{1-\frac{x}{4}}. \tag{15}$$

And the 3rd-order one is:

$$(1-x)^{\frac{1}{2}}=1-\frac{x/2}{1-\frac{x/4}{1-\frac{x}{4}}}. \tag{16}$$

Approximate the wave equation

The innovative work was done by John Claerbout, and is well-known as 15° wave equation to separate the up-going and down-going waves (Claerbout, 1971, 1986).

Eliminating the source term, the Fourier transform of the scalar wave equation (Eq. (5)) can be specified as:

$$\frac{\omega^2}{v^2}=k_x^2+k_z^2. \tag{17}$$

The down-going wave equation in Fourier domain is

$$k_z=\sqrt{\frac{\omega^2}{v^2}-k_x^2}=\frac{\omega}{v}\sqrt{1-\frac{v^2k_x^2}{\omega^2}}. \tag{18}$$

Using the different order Pade expansions, we have:

$$\begin{cases} 1\text{st}-\text{order}: k_z=\dfrac{\omega}{v}\left(1-\dfrac{v^2k_x^2}{2\omega^2}\right) \\ 2\text{nd}-\text{order}: k_z=\dfrac{\omega}{v}\left(1-\dfrac{\frac{2v^2k_x^2}{\omega^2}}{4-\frac{v^2k_x^2}{\omega^2}}\right) \\ 3\text{rd}-\text{order}: k_z=\dfrac{\omega}{v}\left(1-\dfrac{\frac{v^2k_x^2}{2\omega^2}-\frac{v^4k_x^4}{8\omega^4}}{1-\frac{v^2k_x^2}{2\omega^2}}\right) \\ 4\text{th}-\text{order}: k_z=\dfrac{\omega}{v}\left(1-\dfrac{\frac{v^2k_x^2}{2\omega^2}-\frac{v^4k_x^4}{4\omega^4}}{1-\frac{3v^2k_x^2}{4\omega^2}+\frac{v^4k_x^4}{16\omega^4}}\right) \end{cases} \tag{19}$$

The corresponding time domain equations are:

$$\begin{cases} \text{1st}-\text{order}: \dfrac{\partial^2 p}{\partial t\,\partial z} + \dfrac{v}{2}\dfrac{\partial^2 p}{\partial x^2} - \dfrac{1}{v}\dfrac{\partial^2 p}{\partial t^2} = 0, (\text{the well}-\text{known } 15^\circ \text{ wave equation}) \\ \text{2nd}-\text{order}: \dfrac{\partial^3 p}{\partial t^2 \partial z} - \dfrac{v^2}{4}\dfrac{\partial^3 p}{\partial x^2 \partial z} - \dfrac{1}{v}\dfrac{\partial^3 p}{\partial t^3} + \dfrac{3v}{4}\dfrac{\partial^3 p}{\partial x^2 \partial t} = 0, (45^\circ \text{ wave equation}) \\ \text{3rd}-\text{order}: \dfrac{\partial^4 p}{\partial t^3 \partial z} - \dfrac{v^2}{2}\dfrac{\partial^4 p}{\partial x^2 \partial t \partial z} - \dfrac{1}{v}\dfrac{\partial^4 p}{\partial t^4} + v\dfrac{\partial^4 p}{\partial x^2 \partial t^2} - \dfrac{v^3}{8}\dfrac{\partial^4 p}{\partial x^4} = 0 \\ \text{4th}-\text{order}: \dfrac{\partial^5 p}{\partial t^4 \partial z} - \dfrac{3v^2}{4}\dfrac{\partial^5 p}{\partial x^2 \partial t^2 \partial z} - \dfrac{v^4}{16}\dfrac{\partial^5 p}{\partial x^4 \partial z} + \dfrac{1}{v}\dfrac{\partial^5 p}{\partial t^5} + \dfrac{5v}{4}\dfrac{\partial^5 p}{\partial x^2 \partial t^3} + \dfrac{5v^3}{16}\dfrac{\partial^5 p}{\partial t \partial x^4} = 0 \end{cases} \tag{20}$$

Absorbing boundary condition (ABC)

Clayton-Enquist boundary condition

To simulate the wave propagation in the infinite space, the absorbing boundary condition (ABC), namely the proximal approximation (PA) boundary condition, was proposed in Clayton and Engquist (1977) and Engquist and Majda (1977). The basic idea is to use the wave equation with opposite direction at the boundary. Take the bottom boundary as an example. Here, allowing for the incident wave is downgoing, we use the up-going wave equation at the bottom boundary. Using a model including 3 layers (Figure 1), Figure 2 displays 10 shots data volume obtained the Clayton-Enquist boundary condition.

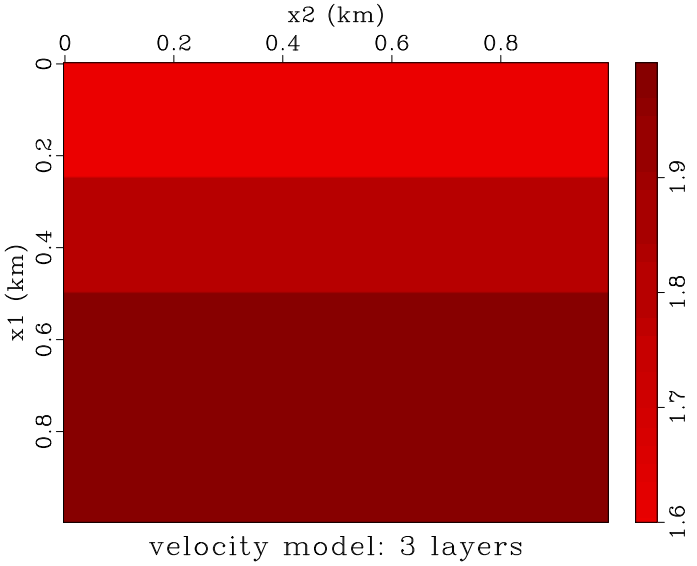


Figure 6: Velocity model: 3 layers primer/modeling2d/ vel

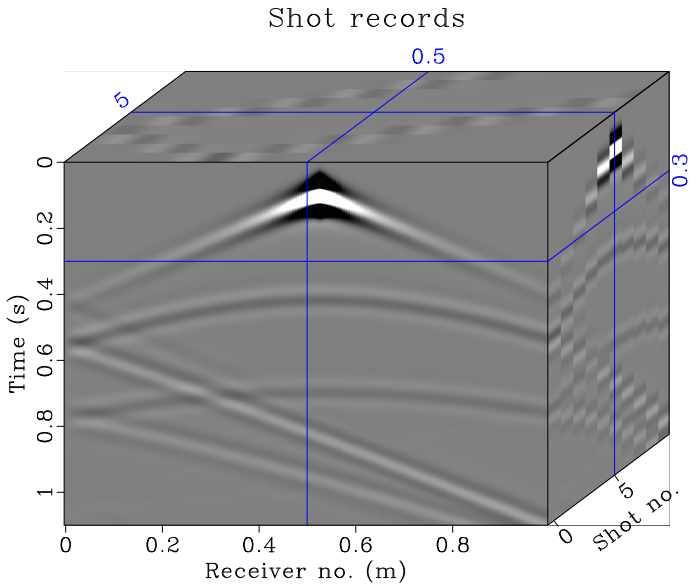


Figure 7: 10 shots data volume obtained using the Clayton-Enquist boundary condition. primer/modeling2d/ shots

Sponge ABC

The sponge ABC was proposed by Cerjan et al. (1985). The principle is very simple: attenuating the refections exponentially in the extended artificial boundary (Figure A-1) area by multiplying a factor less $d(u)$ than 1. Commonly, we use the factor

$$d(u) = \exp(-[0.015 * (nb - i)]^2), u = x, z(i\Delta x \text{ or } i\Delta z) \tag{21}$$

where nb is the thickness of the artificial boundary on each side of the model. Usually, we choose it to be $nb = 20 \sim 30$. The sponge ABC can be easily applied to a wide range of wave propagation problems, including some governing wave equations for complicated medium.

Perfectly Matched Layer (PML)

The PML ABC was proposed in electromagnetics computation (Berenger, 1994). In seismic wave propagration community, two versions of PML boundary condition have been developed: the split PML (SPML) and nonsplit PML (NPML).

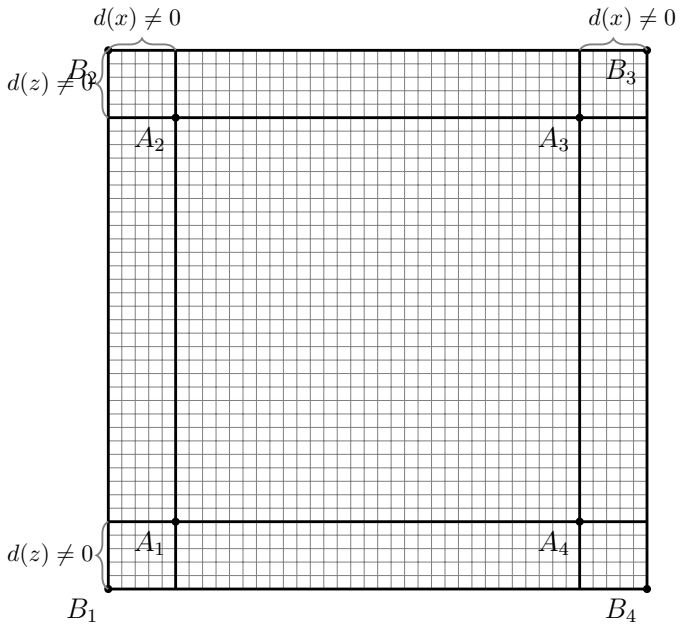


Figure 8: A schematic diagram of extended artificial boundary area. $A_1A_2A_3A_4$ is the original model zone, which is extended to be $B_1B_2B_3B_4$ with artificial boundary. In the extended bounary area, the attenuation coeffcient $d(u) \neq 0$; In the model zone $A_1A_2A_3A_4$, $d(u) = 0$, $u = x, z$.

Split PML (SPML) for acoustics

It is possible for us to split the wave field into two components: x-component p_x and z-component p_z (Carcione et al., 2002). Then the acoustic wave equation becomes

$$\begin{cases} p = p_x + p_z \\ \dfrac{\partial p_x}{\partial t} = v^2 \dfrac{\partial v_x}{\partial x} \\ \dfrac{\partial p_z}{\partial t} = v^2 \dfrac{\partial v_z}{\partial z} \\ \dfrac{\partial v_x}{\partial t} = \dfrac{\partial p}{\partial x} \\ \dfrac{\partial v_z}{\partial t} = \dfrac{\partial p}{\partial z} \end{cases} \tag{22}$$

To absorb the boundary reflection, by adding the decaying coefficients $d(u)$ the SPML governing equation can be specified as (Collino and Tsogka, 2001)

$$\begin{cases} p = p_x + p_z \\ \dfrac{\partial p_x}{\partial t} + d(x)p_x = v^2\dfrac{\partial v_x}{\partial x} \\ \dfrac{\partial p_z}{\partial t} + d(z)p_z = v^2\dfrac{\partial v_z}{\partial z} \\ \dfrac{\partial v_x}{\partial t} + d(x)v_x = \dfrac{\partial p}{\partial x} \\ \dfrac{\partial v_z}{\partial t} + d(z)v_z = \dfrac{\partial p}{\partial z} \end{cases} \tag{23}$$

where $d(x)$ and $d(z)$ are the ABC coefficients designed to attenuate the reflection in the boundary zone, see Figure A-1. There exists many forms of ABC coefficients function. In the absorbing layers, we use the following model for the damping parameter $d(x)$ (Collino and Tsogka, 2001):

$$d(u) = d_0(\frac{u}{L})^2, d_0 = -\frac{3v}{2L}\ln(R), u = x, z \tag{24}$$

where L indicates the PML thinkness; x represents the distance between current position (in PML) and PML inner boundary. R is always chosen as $10^{-3} \sim 10^{-6}$. It is important to note that the same idea can be applied to elastic wave equation (Collino and Tsogka, 2001). The split version of wave equation is very suitable for the construction of seismic Poynting vector. A straightforward application is the angle gather extration using Poynting vector, see Section .

A numerical example of SPML using 8th order staggered finite difference scheme is given in Figure 9.

Nonsplit Convolutional-PML (CPML) for acoustics

Another approach to improve the behavior of the discrete PML at grazing incidence consists in modifying the complex coordinate transform used classically in the PML to introduce a frequency-dependent term that implements a Butterworth-type filter in the layer. This approach has been developed for Maxwells equations named convolutional PML (CPML) (Roden and Gedney, 2000) or complex frequency shifted-PML (CFS-PML). The key idea is that for waves whose incidence is close to normal, the presence of such a filter changes almost nothing because absorption is already almost perfect. But for waves with grazing incidence, which for geometrical reasons do not penetrate very deep in the PML, but travel there a longer way in the direction parallel to the layer, adding such a filter will strongly attenuate them and will prevent them from leaving the PML with significant energy .

Define $Ax = \frac{\partial p}{\partial x}$, $Az = \frac{\partial p}{\partial z}$. Then the acoustic wave equation reads

$$\frac{\partial^2 p}{\partial t^2} = v^2\left(\frac{\partial Ax}{\partial x} + \frac{\partial Az}{\partial z}\right).$$

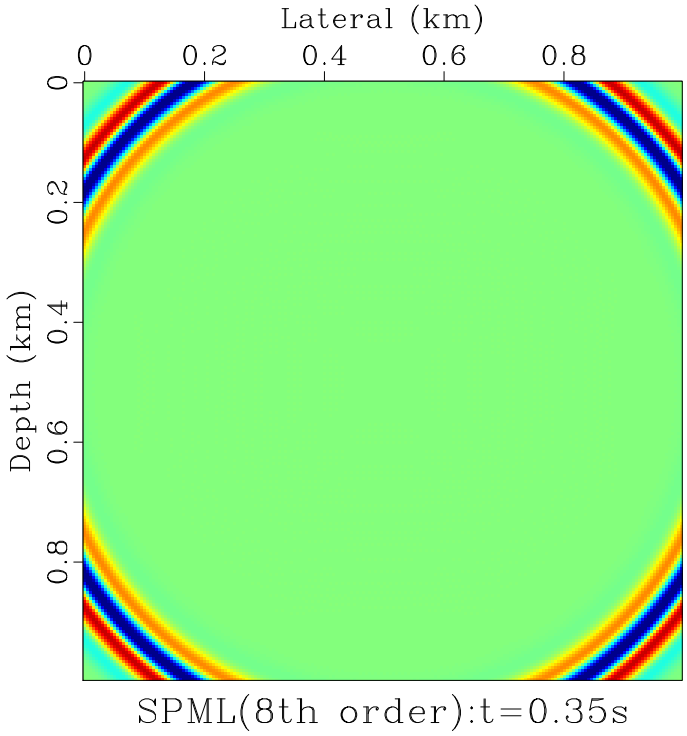


Figure 9: Wavefield snap of SPML with 8th order finite difference primer/testspml/ snapspml

To combine the absorbing effects into the acoustic equation, we merely need to combine two convolution terms into the above equations:

$$\begin{cases} \dfrac{\partial^2 p}{\partial t^2} = v^2 \left(Px + Pz\right) \\ Px = \dfrac{\partial Ax}{\partial x} + \Psi_x \\ Pz = \dfrac{\partial Az}{\partial z} + \Psi_z \\ Ax = \dfrac{\partial p}{\partial x} + \Phi_x \\ Az = \dfrac{\partial p}{\partial z} + \Phi_z \end{cases} \tag{25}$$

where Ψ_x, Ψ_z are the convolution terms of Ax and Az; Φ_x, Φ_z are the convolution terms of Px and Pz. These convolution terms can be computed via the following relation:

$$\begin{cases} \Psi_x^n = b_x \Psi_x^{n-1} + (b_x - 1)\partial_x^{n-1/2} Ax \\ \Psi_z^n = b_z \Psi_z^{n-1} + (b_z - 1)\partial_z^{n-1/2} Az \\ \Phi_x^n = b_x \Phi_x^{n-1} + (b_x - 1)\partial_x^{n-1/2} P \\ \Phi_z^n = b_z \Phi_z^{n-1} + (b_z - 1)\partial_z^{n-1/2} P \end{cases} \tag{26}$$

where $b_x = e^{-d(x)\Delta t}$ and $b_z = e^{-d(z)\Delta t}$. More details about the derivation of C-PML, the interested readers are referred to Collino and Tsogka (2001) and Komatitsch and Martin (2007).

Nonsplit PML (NPML) for elastics

The nonsplit PML for elastic implementation is

$$\begin{cases} \rho\dfrac{\partial v_x}{\partial t} = (\dfrac{\partial \tau_{xx}}{\partial x} + \dfrac{\partial \tau_{xz}}{\partial z}) - \Omega_{xx} - \Omega_{xz} \\ \rho\dfrac{\partial v_z}{\partial t} = (\dfrac{\partial \tau_{xz}}{\partial x} + \dfrac{\partial \tau_{zz}}{\partial z}) - \Omega_{zx} - \Omega_{zz} \\ \dfrac{\partial \tau_{xx}}{\partial t} = (\lambda+2\mu)\dfrac{\partial v_x}{\partial x} + \lambda\dfrac{\partial v_z}{\partial z} - (\lambda+2\mu)\Psi_{xx} - \lambda\Psi_{zz} \\ \dfrac{\partial \tau_{zz}}{\partial t} = \lambda\dfrac{\partial v_x}{\partial x} + (\lambda+2\mu)\dfrac{\partial v_z}{\partial z} - \lambda\Psi_{xx} - (\lambda+2\mu)\Psi_{zz} \\ \dfrac{\partial \tau_{xz}}{\partial t} = \mu\dfrac{\partial v_x}{\partial x} + \mu\dfrac{\partial v_z}{\partial z} - \mu\Psi_{zx} - \mu\Psi_{xz} \end{cases} \tag{27}$$

where the auxiliary variables are governed via the following relation

$$\begin{cases} \dfrac{\partial \Omega_{xx}}{\partial t} + d(x)\Omega_{xx} = d(x)\dfrac{\partial \tau_{xx}}{\partial x}, \dfrac{\partial \Omega_{xz}}{\partial t} + d(z)\Omega_{xz} = d(z)\dfrac{\partial \tau_{xz}}{\partial z} \\ \dfrac{\partial \Omega_{zx}}{\partial t} + d(x)\Omega_{zx} = d(x)\dfrac{\partial \tau_{xz}}{\partial x}, \dfrac{\partial \Omega_{zz}}{\partial t} + d(z)\Omega_{zz} = d(z)\dfrac{\partial \tau_{zz}}{\partial z} \\ \dfrac{\partial \Psi_{xx}}{\partial t} + d(x)\Psi_{xx} = d(x)\dfrac{\partial v_x}{\partial x}, \dfrac{\partial \Psi_{xz}}{\partial t} + d(z)\Psi_{xz} = d(z)\dfrac{\partial v_x}{\partial z} \\ \dfrac{\partial \Psi_{zx}}{\partial t} + d(x)\Psi_{zx} = d(x)\dfrac{\partial v_z}{\partial x}, \dfrac{\partial \Psi_{zz}}{\partial t} + d(z)\Psi_{zz} = d(z)\dfrac{\partial v_z}{\partial z} \end{cases} \tag{28}$$

Discretization

The Taylor series expansion of a function $f(x)$ can be written as

$$\begin{cases} f(x+h) = f(x) + \dfrac{\partial f(x)}{\partial x}h + \dfrac{1}{2!}\dfrac{\partial^2 f(x)}{\partial x^2}h^2 + \dfrac{1}{3!}\dfrac{\partial^3 f(x)}{\partial x^3}h^3 + \dots \\ f(x-h) = f(x) - \dfrac{\partial f(x)}{\partial x}h + \dfrac{1}{2!}\dfrac{\partial^2 f(x)}{\partial x^2}h^2 - \dfrac{1}{3!}\dfrac{\partial^3 f(x)}{\partial x^3}h^3 + \dots \end{cases} \tag{29}$$

It leads to

$$\begin{cases} \dfrac{f(x+h)+f(x-h)}{2} = f(x) + \dfrac{1}{2!}\dfrac{\partial^2 f(x)}{\partial x^2}h^2 + \dfrac{1}{4!}\dfrac{\partial^4 f(x)}{\partial x^4}h^4 + \dots \\ \dfrac{f(x+h)-f(x-h)}{2} = \dfrac{\partial f(x)}{\partial x}h + \dfrac{1}{3!}\dfrac{\partial^3 f(x)}{\partial x^3}h^3 + \dfrac{1}{5!}\dfrac{\partial^5 f(x)}{\partial x^5}h^5 + \dots \end{cases} \tag{30}$$

Let $h = \Delta x/2$. This implies

$$\begin{cases} \dfrac{\partial f(x)}{\partial x} = \dfrac{f(x+\Delta x/2) - f(x-\Delta x/2)}{\Delta x} + O(\Delta x^2) \\ f(x) = \dfrac{f(x+\Delta x/2) + f(x-\Delta x/2)}{2} + O(\Delta x^2) \end{cases} \tag{31}$$

Higher-order approximation of staggered-grid finite difference

To approximate the 1st-order derivatives as accurate as possible, we express it in the following

$$\begin{aligned} \frac{\partial f}{\partial x} =& a_1 \frac{f(x+\Delta x/2) - f(x-\Delta x/2)}{\Delta x} + \\ & a_2 \frac{f(x+3\Delta x/2) - f(x-3\Delta x/2)}{3\Delta x} + \\ & a_3 \frac{f(x+5\Delta x/2) - f(x-5\Delta x/2)}{5\Delta x} + \cdots \\ =& c_1 \frac{f(x+\Delta x/2) - f(x-\Delta x/2)}{\Delta x} + \\ & c_2 \frac{f(x+3\Delta x/2) - f(x-3\Delta x/2)}{\Delta x} + \\ & c_3 \frac{f(x+5\Delta x/2) - f(x-5\Delta x/2)}{\Delta x} + \cdots \end{aligned} \tag{32}$$

where $c_i = a_i/(2i-1)$. Substituting the $f(x+h)$ and $f(x-h)$ with (29) for $h = \Delta x/2, 3\Delta x/2, \ldots$ results in

$$\begin{aligned} \frac{\partial f}{\partial x} =& c_1 \left(\Delta x \frac{\partial f}{\partial x} + \frac{1}{3}(\frac{\Delta x}{2})^2 \frac{\partial^3 f}{\partial x^3} + \cdots \right) / \Delta x \\ & + c_2 \left(3\Delta x \frac{\partial f}{\partial x} + \frac{1}{3}(\frac{3\Delta x}{2})^2 \frac{\partial^3 f}{\partial x^3} + \cdots \right) / \Delta x \\ & + c_3 \left(5\Delta x \frac{\partial f}{\partial x} + \frac{1}{3}(\frac{5\Delta x}{2})^2 \frac{\partial^3 f}{\partial x^3} + \cdots \right) / \Delta x + \ldots \\ =& (c_1 + 3c_2 + 5c_3 + 7c_4 + \cdots) \frac{\partial f}{\partial x} \\ & + \frac{\Delta x^2}{3 \cdot 2^2} (c_1 + 3^3 c_2 + 5^3 c_3 + 7^3 c_4 + \cdots) \frac{\partial^3 f}{\partial x^3} \\ & + \frac{\Delta x^4}{3 \cdot 2^4} (c_1 + 3^5 c_2 + 5^5 c_3 + 7^5 c_4 + \cdots) \frac{\partial^5 f}{\partial x^5} + \cdots \\ =& (a_1 + a_2 + a_3 + a_4 + \cdots) \frac{\partial f}{\partial x} \\ & + \frac{\Delta x^2}{3 \cdot 2^2} (a_1 + 3^2 a_2 + 5^2 a_3 + 7^2 a_4 + \cdots) \frac{\partial^3 f}{\partial x^3} \\ & + \frac{\Delta x^4}{3 \cdot 2^4} (a_1 + 3^4 a_2 + 5^4 a_3 + 7^4 a_4 + \cdots) \frac{\partial^5 f}{\partial x^5} + \cdots \end{aligned} \tag{33}$$

Thus, taking first N terms means

$$\begin{cases} a_1+a_2+a_3+\cdots+a_N=1 \\ a_1+3^2a_2+5^2a_3+\cdots+(2N-1)^2a_N=0 \\ a_1+3^4a_2+5^4a_3+\cdots+(2N-1)^4a_N=0 \\ \cdots \\ a_1+3^{2N-2}a_2+5^{2N-2}a_3+\cdots)+(2N-1)^{2N-2}a_N=0 \end{cases} \tag{34}$$

In matrix form,

$$\underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1^2 & 3^2 & \dots & (2N-1)^2 \\ \vdots & & \ddots & \vdots \\ 1^{2N-2} & 3^{2N-2} & \dots & (2N-1)^{2N-2} \end{bmatrix}}_{\mathbf{V}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\mathbf{b}} \tag{35}$$

The above matrix equation is Vandermonde-like system: $\mathbf{Va}=\mathbf{b}$, $\mathbf{a}=(a_1,a_2,\dots,a_N)^T$. The Vandermonde matrix

$$\mathbf{V}=\begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \\ \vdots & & \ddots & \vdots \\ x_1^{N-1} & x_2^{N-1} & \dots & x_N^{N-1} \end{bmatrix} \tag{36}$$

in which $x_i=(2i-1)^2$, has analytic solutions. $\mathbf{Va}=\mathbf{b}$ can be solved using the specific algorithms, see Bjorck (1996). And we obtain

$$\frac{\partial f}{\partial x}=\frac{1}{\Delta x}\sum_{i=1}^{N}c_i(f(x+i\Delta x/2)-f(x-i\Delta x/2)+O(\Delta x^{2N}) \tag{37}$$

The MATLAB code for solving the $2N$-order finite difference coefficients is provided in the following.

```
function c=staggered_fdcoeff(NJ)
% Computing 2*N-order staggered-grid FD coefficients (NJ=2N)
% Example:
%       format long
%       NJ=10;
%       c=staggered_fdcoeff(NJ)

N=NJ/2;
x=zeros(N,1);
b=zeros(N,1);     b(1)=1;
c=b;
for k=1:N
```

```
    x(k)=(2*k-1)^2;
end

for k=1:N-1
    for i=N:-1:k+1
        b(i)=b(i)-x(k)*b(i-1);
    end
end

for k=N-1:-1:1
    for i=k+1:N
        b(i)=b(i)/(x(i)-x(i-k));
    end
    for i=k:N-1
        b(i)=b(i)-b(i+1);
    end
end
for k=1:N
    c(k)=b(k)/(2*k-1);
end
```

In general, the stability of staggered-grid difference requires that

$$\Delta t \max(v)\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}} \leq \frac{1}{\sum_{i=1}^{N}|c_i|}. \tag{38}$$

Define $C = \frac{1}{\sum_{i=1}^{N}|c_i|}$. Then, we have

$$\begin{cases} N=1, & C=1 \\ N=2, & C=0.8571 \\ N=3, & C=0.8054 \\ N=4, & C=0.7774 \\ N=5, & C=0.7595 \end{cases}$$

In the 2nd-order case, numerical dispersion is limited when

$$\max(\Delta x, \Delta z) < \frac{\min(v)}{10 f_{\max}}. \tag{39}$$

The 4th-order dispersion relation is:

$$\max(\Delta x, \Delta z) < \frac{\min(v)}{5 f_{\max}}. \tag{40}$$

Discretization of SPML

Take

$$\frac{\partial p_x}{\partial t}+d(x)p_x=v^2\frac{\partial v_x}{\partial x}$$

for an example. Using the 2nd-order approximation in Eq. (31), we expand it at the time $(k+\frac{1}{2})\Delta t$ and the point $[ix\Delta x, iz\Delta z]$

$$\frac{p_x^{k+1}[ix,iz]-p_x^k[ix,iz]}{\Delta t}+d[ix]\frac{p_x^{k+1}[ix,iz]+p_x^k[ix,iz]}{2}=v^2[ix,iz]\frac{v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz]-v_x^{k+\frac{1}{2}}[ix-\frac{1}{2},iz]}{\Delta x} \tag{41}$$

That is to say,

$$p_x^{k+1}[ix,iz]=\frac{1-0.5\Delta td[ix]}{1+0.5\Delta td[ix]}p_x^k[ix,iz]+\frac{1}{1+0.5\Delta td[ix]}\frac{\Delta tv^2[ix,iz]}{\Delta x}(v_x^{k+1\frac{1}{2}}[ix+\frac{1}{2},iz]-v_x^{k+\frac{1}{2}}[ix-\frac{1}{2},iz]) \tag{42}$$

At time $k\Delta t$ and $[ix+\frac{1}{2},iz]$, we expand

$$\frac{\partial v_x}{\partial t}+d(x)v_x=\frac{\partial p}{\partial x}$$

as

$$\frac{v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz]-v_x^{k-\frac{1}{2}}[ix+\frac{1}{2},iz]}{\Delta t}+d[ix]\frac{v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz]+v_x^{k-\frac{1}{2}}[ix+\frac{1}{2},iz]}{2}=\frac{p^k[ix+1,iz]-p^k[ix,iz]}{\Delta x} \tag{43}$$

Thus, we have

$$v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz]=\frac{1-0.5\Delta td[ix]}{1+0.5\Delta td[ix]}v_x^{k-\frac{1}{2}}[ix+\frac{1}{2},iz]+\frac{1}{1+0.5\Delta td[ix]}\frac{\Delta t}{\Delta x}(p^k[ix+1,iz]-p^k[ix,iz]) \tag{44}$$

In summary,

$$\begin{cases} v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz] = \frac{1-0.5\Delta td[ix]}{1+0.5\Delta td[ix]} v_x^{k-\frac{1}{2}}[ix+\frac{1}{2},iz] + \\ \frac{1}{1+0.5\Delta td[ix]}\frac{\Delta t}{\Delta x}(p^k[ix+1,iz]-p^k[ix,iz]) \\ v_z^{k+\frac{1}{2}}[ix,iz+\frac{1}{2}] = \frac{1-0.5\Delta td[iz]}{1+0.5\Delta td[iz]} v_z^{k-\frac{1}{2}}[ix,iz+\frac{1}{2}] + \\ \frac{1}{1+0.5\Delta td[iz]}\frac{\Delta t}{\Delta z}(p^k[ix,iz+1]-p^k[ix,iz]) \\ p_x^{k+1}[ix,iz] = \frac{1-0.5\Delta td[ix]}{1+0.5\Delta td[ix]} p_x^k[ix,iz] + \\ \frac{1}{1+0.5\Delta td[ix]}\frac{\Delta t v^2[ix,iz]}{\Delta x}(v_x^{k+\frac{1}{2}}[ix+\frac{1}{2},iz]-v_x^{k+\frac{1}{2}}[ix-\frac{1}{2},iz]) \\ p_z^{k+1}[ix,iz] = \frac{1-0.5\Delta td[iz]}{1+0.5\Delta td[iz]} p_z^k[ix,iz] + \\ \frac{1}{1+0.5\Delta td[iz]}\frac{\Delta t v^2[ix,iz]}{\Delta z}(v_z^{k+\frac{1}{2}}[ix,iz+\frac{1}{2}]-v_z^{k+\frac{1}{2}}[ix,iz-\frac{1}{2}]) \\ p^{k+1}[ix,iz] = p_x^{k+1}[ix,iz]+p_z^{k+1}[ix,iz] \end{cases} \tag{45}$$

If we define:

$$b' = \frac{1-0.5\Delta td}{1+0.5\Delta td}, b = \exp(-\Delta td) \tag{46}$$

we can easily find that b' is a good approximation of b up to 2nd order, allowing for the 2nd order Pade expansion:

$$\exp(z) \approx \frac{1+0.5z}{1-0.5z} \tag{47}$$

Then, we have

$$1-b \approx 1-b' = \frac{\Delta td}{1+0.5\Delta td} \tag{48}$$

Discretization of NPML

Note that all sub-equations can be formulated in the following form:

$$\frac{\partial f}{\partial t} + df = \gamma. \tag{49}$$

The analytic solution of this equation is

$$f = -\frac{1}{d}e^{-dt} + \frac{1}{d}\gamma \tag{50}$$

In discrete form,

$$\begin{aligned} f(k\Delta t) &= -\frac{1}{d}e^{-dk\Delta t} + \frac{1}{d}\gamma, \\ f((k+1)\Delta t) &= -\frac{1}{d}e^{-dt}e^{-dk\Delta t} + \frac{1}{d}\gamma. \end{aligned} \tag{51}$$

Thus,

$$f((k+1)\Delta t) = e^{-d\Delta t}f(k\Delta t) + \frac{1}{d}(1-e^{-d\Delta t})\gamma \tag{52}$$

For $\frac{\partial \Omega_{xx}}{\partial t} + d(x)\Omega_{xx} = d(x)\frac{\partial \tau_{xx}}{\partial x}$, $\gamma = d(x)\frac{\partial \tau_{xx}}{\partial x}$, the update rule becomes

$$\Omega_{xx}^{k+1} = e^{-d(x)\Delta t}\Omega_{xx}^{k} + (1-e^{-d(x)\Delta t})\frac{\partial \tau_{xx}^{k+1/2}}{\partial x} = b_x\Omega_{xx}^{k} + (1-b_x)\frac{\partial \tau_{xx}^{k+1/2}}{\partial x} \tag{53}$$

where $b_x = e^{-d(x)\Delta t}$ and $b_z = e^{-d(z)\Delta t}$. Ω_{xx}, Ω_{xz}, Ω_{zx}, Ω_{zz}, Ψ_{xx}, Ψ_{xz}, Ψ_{zx} and Ψ_{zz} can be obtained in the same way:

$$\begin{cases} \Omega_{xx}^{k+1} = b_x\Omega_{xx}^{k} + (1-b_x)\frac{\partial \tau_{xx}^{k+1/2}}{\partial x}, \Omega_{xz}^{k+1} = b_z\Omega_{xz}^{k} + (1-b_z)\frac{\partial \tau_{xz}^{k+1/2}}{\partial z} \\ \Omega_{zx}^{k+1} = b_x\Omega_{zx}^{k} + (1-b_z)\frac{\partial \tau_{xz}^{k+1/2}}{\partial x}, \Omega_{zz}^{k+1} = b_z\Omega_{zz}^{k} + (1-b_z)\frac{\partial \tau_{zz}^{k+1/2}}{\partial z} \\ \Psi_{xx}^{k+1} = b_x\Psi_{xx}^{k} + (1-b_x)\frac{\partial v_{x}^{k+1/2}}{\partial x}, \Psi_{xz}^{k+1} = b_z\Psi_{xz}^{k} + (1-b_z)\frac{\partial v_{x}^{k+1/2}}{\partial z} \\ \Psi_{zx}^{k+1} = b_x\Psi_{zx}^{k} + (1-b_x)\frac{\partial v_{z}^{k+1/2}}{\partial x}, \Psi_{zz}^{k+1} = b_z\Psi_{zz}^{k} + (1-b_z)\frac{\partial v_{z}^{k+1/2}}{\partial z} \end{cases} \tag{54}$$

As can be seen from Eq. (27), we only need to subtract the reflection part Ω and Ψ after global updating (Eq. (9)). We summarize this precedure as follows:

Step 1: Perform the computation of Eq. (9) in whole area;

Step 2: In PML zone, subtract decaying parts according to Eq. (27).

REVERSE TIME MIGRATION (RTM)

Brief overview

One-way equation based imaging techniques are inadequate to obtain accurate images in complex media due to propagation direction changes in the background model (Biondi, 2006). These approaches are extremely limited when handling the problems of turning waves in the model containing sharp wave-speed contrasts and steeply dipping reflectors. As an advanced imaging technology without dip and extreme lateral velocity limitation, reverse time migration (RTM) was proposed early (Baysal

et al., 1983; McMechan, 1983), but not practical in terms of stringent computation and memory requirement. However, it gained increasingly attention in recent years due to the tremendous advances in computer capability. Until recently, 3D prestack RTM is now feasible to obtain high fidelity images (Yoon et al., 2003; Guitton et al., 2006).

RTM implementation

RTM can be carried out as follows: (1) forward-extrapolating the source wavefield,(2) backward-extrapolating the receiver wavefield, both explicitly in time, and (3) apply an imaging condition.

Imaging condition

The cross-correlation imaging condition can be expressed as

$$I(\mathbf{x}) = \sum_{s=1}^{ns} \int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x}, t; \mathbf{x}_s) p_g(\mathbf{x}, t; \mathbf{x}_g) \tag{55}$$

where $I(\mathbf{x})$ is the migration image value at point $\mathbf{x}$; and $p_s(\mathbf{x}, t)$ and $p_g(\mathbf{x}, t)$ are the forward and reverse-time wavefields at point $\mathbf{x}$. With illumination compensation, the cross-correlation imaging condition is given by

$$I(\mathbf{x}) = \sum_{s=1}^{ns} \frac{\int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x}, t; \mathbf{x}_s) p_g(\mathbf{x}, t; \mathbf{x}_g)}{\int_0^{t_{\max}} \mathrm{d}t p_s(\mathbf{x}, t; \mathbf{x}_s) p_s(\mathbf{x}, t; \mathbf{x}_s) + \sigma^2} \tag{56}$$

in which σ^2 is chosen small to avoid being divided by zeros.

There exists a better way to carry out the illumination compensation, as suggested by Guitton et al. (2007)

$$I(\mathbf{x}) = \sum_{s=1}^{ns} \frac{\int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x}, t; \mathbf{x}_s) p_g(\mathbf{x}, t; \mathbf{x}_g)}{\langle \int_0^{t_{\max}} \mathrm{d}t p_s(\mathbf{x}, t; \mathbf{x}_s) p_s(\mathbf{x}, t; \mathbf{x}_s) \rangle_{x,y,z}} \tag{57}$$

where $\langle \rangle_{x,y,z}$ stands for smoothing in the image space in the x, y, and z directions.

Yoon et al. (2003) define the seismic Poynting vector as

$$\mathbf{S} = \mathbf{v}p = \nabla p \frac{\mathrm{d}p}{\mathrm{d}t} p = (v_x p, v_z p). \tag{58}$$

Here, we denote S_s and S_r as the source wavefield and receiver wavefield Poynting vector. As mentioned before, boundary saving with split PML is a good scheme for the computation of Poynting vector, because p and (v_x, v_z) are available when

backward reconstructing the source wavefield. The angle between the incident wave and the reflected wave can then be obtained:

$$\gamma = \arccos \frac{\mathbf{S}_s \cdot \mathbf{S}_r}{|\mathbf{S}_s||\mathbf{S}_r|} \tag{59}$$

The incident angle (or reflective angle) is half of γ, namely,

$$\theta = \frac{\gamma}{2} = \frac{1}{2} \arccos \frac{\mathbf{S}_s \cdot \mathbf{S}_r}{|\mathbf{S}_s||\mathbf{S}_r|} \tag{60}$$

Using Poynting vector to confine the spurious artefacts, Yoon and Marfurt (2006) propose a hard thresholding scheme to weight the imaging condition:

$$I(\mathbf{x}) = \sum_{s=1}^{ns} \frac{\int_0^{t_{\max}} \mathrm{d}t \sum_{g=1}^{ng} p_s(\mathbf{x}, t; \mathbf{x}_s) p_g(\mathbf{x}, t; \mathbf{x}_g) W(\theta)}{\int_0^{t_{\max}} \mathrm{d}t p_s(\mathbf{x}, t; \mathbf{x}_s) p_s(\mathbf{x}, t; \mathbf{x}_s) + \sigma^2} \tag{61}$$

where

$$W(\theta) = \begin{cases} 1 & \theta < \theta_{\max} \\ 0 & otherwise \end{cases} \tag{62}$$

Costa et al. (2009) modified the weight as

$$W(\theta) = \cos^3(\frac{\theta}{2}). \tag{63}$$

These approaches are better for eliminating the backward scattering waves in image.

Computation strategies and boundary saving

There are several possible ways to do RTM computation. The simplest one may be just storing the forward modeled wavefields on the disk, and reading them for imaging condition in the backward propagation steps. This approach requires frequent disk I/O and has been replaced by wavefield reconstruction method. The so-called wavefield reconstruction method is a way to recover the wavefield via backward reconstructing or forward remodeling, using the saved wavefield shots and boundaries. It is of special value for GPU computing because saving the data in device variables eliminates data transfer between CPU and GPU. By saving the last two wavefield snaps and the boundaries, one can reconstruct the wavefield of every time step, in time-reversal order. The checkpointing technique becomes very useful to further reduce the storage (Symes, 2007; Dussaud et al., 2008). It is also possible to avert the issue of boundary saving by applying the random boundary condition, which may bring some noises in the migrated image (Clapp, 2009; Clapp et al., 2010; Liu et al., 2013b,a).

Yang et al. (2014) proposed an effective boundary for regular and staggered-grid finite differences. In the case of regular grid finite difference of order $2N$, we need

to save N points on each inner side in the model zone to reconstruct the wavefield. For staggered grid finite difference of order $2N$, we need to save $2N-1$ points on each inner in the model for perfect reconstruction. The concept of effective boundary saving does not depends on C or GPU implementation. However, it is of special value for GPU implemenation, because it eliminates the CPU-GPU data transfer for boundary saving. An example of effective boundary saving for regular grid finite difference is given in Figure 10. The imaging examples using effective boundary saving with staggered-grid finite difference can be found in the next section.

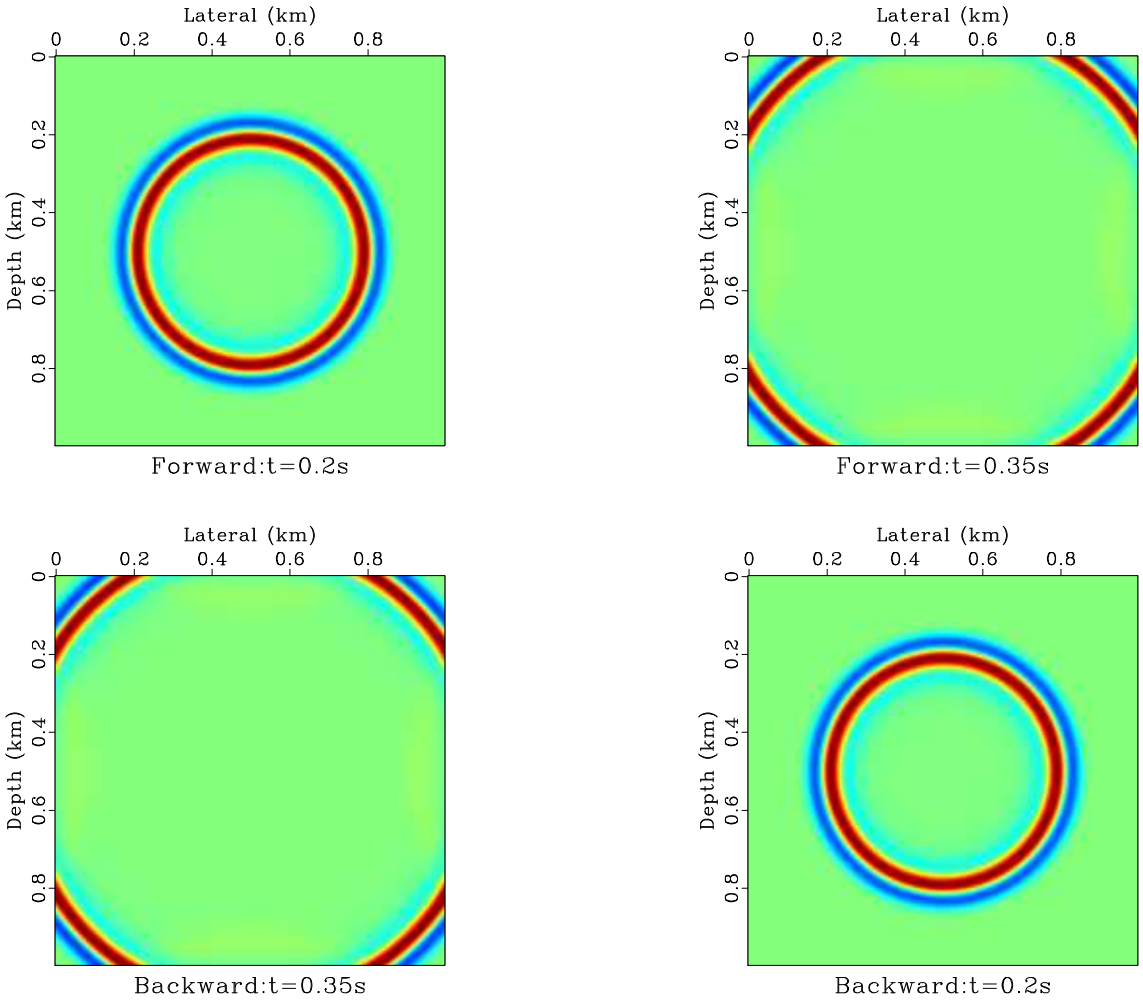


Figure 10: The forward modeled wavefield can be exactly reconstructed using effective boundary saving. primer/testeb/ fb

Numerical examples

I show my GPU-based RTM result for two benchmark models: Marmousi model (Versteeg, 1994) and Sigsbee model (DiMarco et al., 2001). Here, I use CPML boundary

condition to obtain high quality imaging result.

The Marmousi model is shown in Figure 11. The spatial sampling interval is $\Delta x = \Delta z = 4m$. 51 shots are deployed. In each shot, 301 receivers are placed in the split shooting mode. The parameters we use are listed as follows: $nt = 13000$, $\Delta t = 0.3$ ms. Due to the limited resource on our computer, we store 65% boundaries using page-locked memory. Figure 12a and 12b give the resulting RTM image after Laplacian filtering. As shown in the figure, RTM with the effective boundary saving scheme produces excellent image: the normalized cross-correlation imaging condition greatly improves the deeper parts of the image due to the illumination compensation. The events in the central part of the model, the limits of the faults and the thin layers are much better defined.

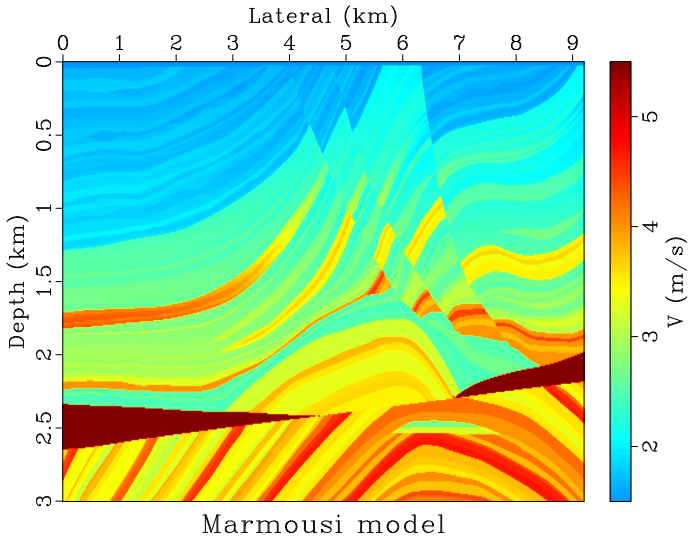


Figure 11: The Marmousi velocity model. primer/marmousi/ marmousi

The Sigsbee model is shown in Figure 13. The spatial interval is $\Delta x = \Delta z = 25m$. 55 shots are evenly distributed on the surface of the model. We still perform $nt = 13000$ time steps for each shot (301 receivers). Due to the larger model size, 75% boundaries have to be stored with the aid of pinned memory. Our RTM results are shown in Figure 14a and 14b. Again, the resulting image obtained by normalized cross-correlation imaging condition exhibits better resolution for the edges of the salt body and the diffraction points. Some events in the image using normalized cross-correlation imaging condition are more visible, while they have a much lower amplitude or are even completely lost in the image of cross-correlation imaging condition.

FULL WAVEFORM INVERSION (FWI)

Time domain FWI was proposed by Tarantola (1984), and developed in Tarantola (1986); Pica et al. (1990). Later, frequency domain FWI was proposed by Pratt et al.

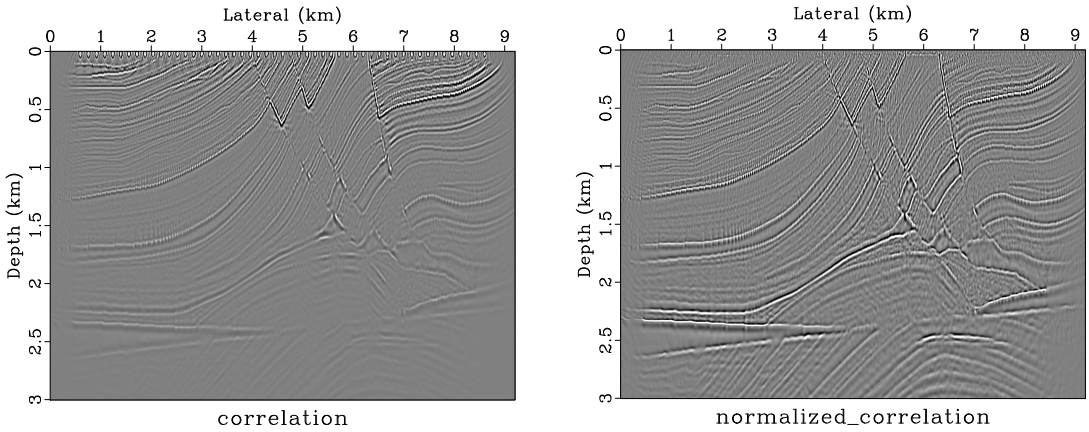


Figure 12: RTM result of Marmousi model using effective boundary saving scheme (staggered grid finite difference). (a) Result of cross-correlation imaging condition. (b) Result of normalized cross-correlation imaging condition. primer/marmousi/ mimag1,mimag2

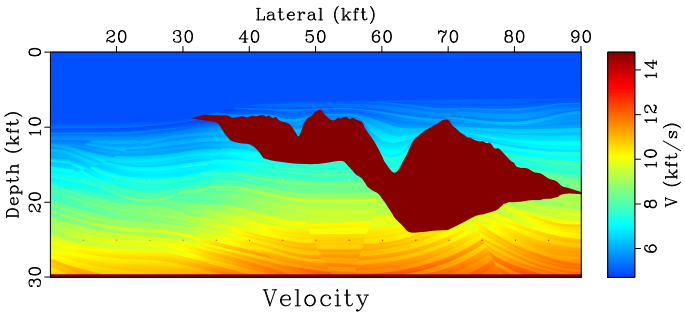


Figure 13: The Sigsbee velocity model. primer/sigsbee/ sigsbee

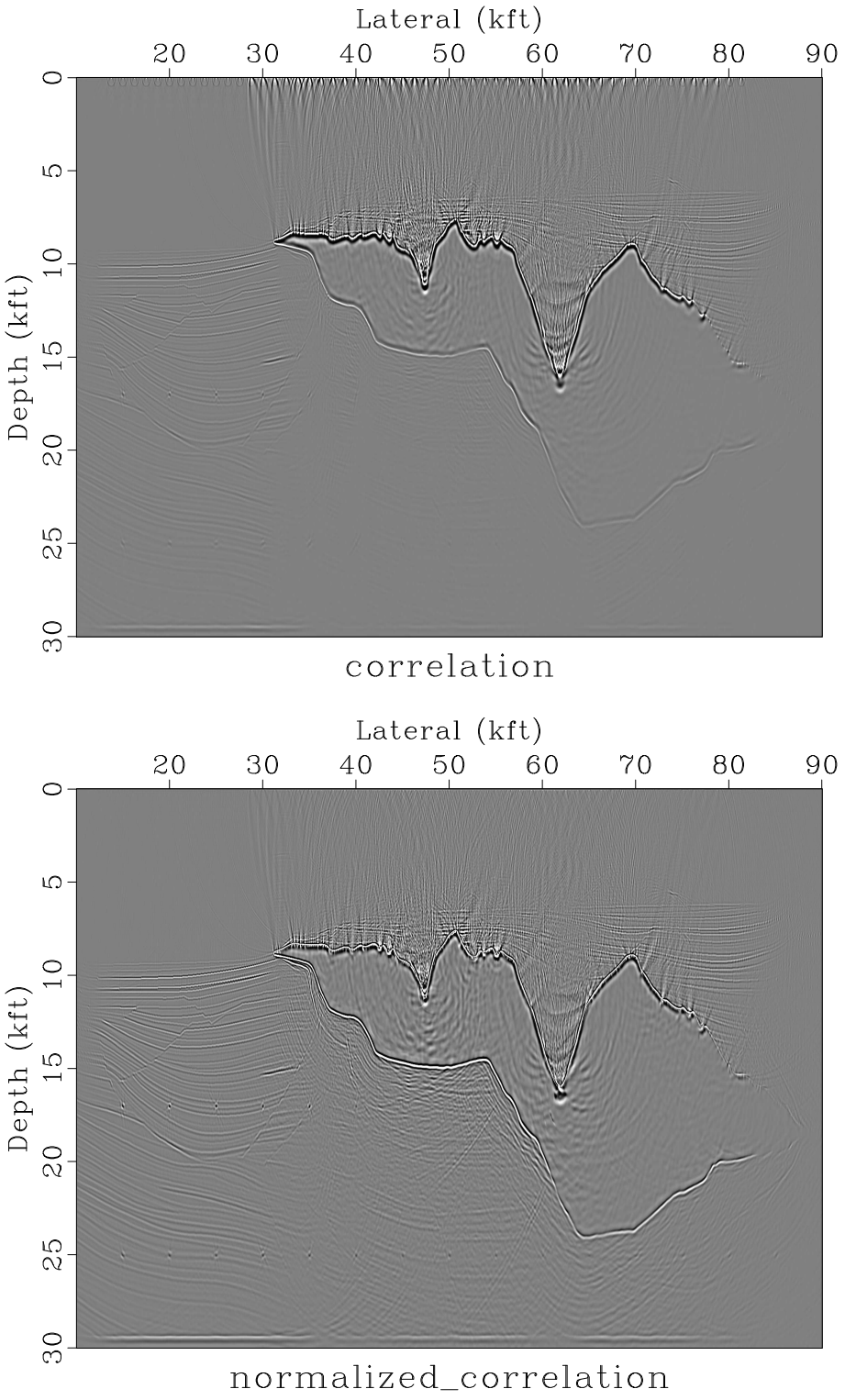


Figure 14: RTM result of Sigsbee model using effective boundary saving scheme (staggered grid finite difference). (a) Result of cross-correlation imaging condition. (b) Result of normalized cross-correlation imaging condition. primer/sigsbee/ simag1,simag2

(1998). Actually, many authors call it full waveform tomography. (tomography=fwi, imaging=migration) Here, we mainly follow two well-documented paper Pratt et al. (1998) and Virieux and Operto (2009). We define the misfit vector $\Delta\mathbf{p} = \mathbf{p}_{cal} - \mathbf{p}_{obs}$ by the differences at the receiver positions between the recorded seismic data $\mathbf{p}_{obs}$ and the modelled seismic data $\mathbf{p}_{cal} = \mathbf{f}(\mathbf{m})$ for each source-receiver pair of the seismic survey. Here, in the simplest acoustic velocity inversion, $\mathbf{m}$ corresponds to the velocity model to be determined. The objective function taking the least-squares norm of the misfit vector $\Delta\mathbf{p}$ is given by

$$E(\mathbf{m}) = \frac{1}{2}\Delta\mathbf{p}^{\dagger}\Delta\mathbf{p} = \frac{1}{2}\Delta\mathbf{p}^{T}\Delta\mathbf{p}^{*} = \frac{1}{2}\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_{0}^{t_{\max}} \mathrm{d}t|p_{cal}(\mathbf{x}_r, t; \mathbf{x}_s) - p_{obs}(\mathbf{x}_r, t; \mathbf{x}_s)|^2 \tag{64}$$

where ns and ng are the number of sources and geophones, $\dagger$ denotes the adjoint and $*$ the complex conjugate, while $\mathbf{f}(\cdot)$ indicates the forward modeling of the wave propagation. The recorded seismic data is only a small subset of the whole wavefield.

The minimum of the misfit function $E(\mathbf{m})$ is sought in the vicinity of the starting model $\mathbf{m}_0$. FWI is essentially a local optimization. In the framework of the Born approximation, we assume that the updated model $\mathbf{m}$ of dimension M can be written as the sum of the starting model $\mathbf{m}_0$ plus a perturbation model $\Delta\mathbf{m}$: $\mathbf{m} = \mathbf{m}_0 + \Delta\mathbf{m}$. In the following, we assume that $\mathbf{m}$ is real valued.

A second-order Taylor-Lagrange development of the misfit function in the vicinity of $\mathbf{m}_0$ gives the expression

$$E(\mathbf{m}_0+\Delta\mathbf{m}) = E(\mathbf{m}_0)+\sum_{i=1}^{M}\frac{\partial E(\mathbf{m}_0)}{\partial m_i}\Delta m_i+\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j}\Delta m_i \Delta m_j + O(||\Delta\mathbf{m}||^3) \tag{65}$$

Taking the derivative with respect to the model parameter m_i results in

$$\frac{\partial E(\mathbf{m})}{\partial m_i} = \frac{\partial E(\mathbf{m}_0)}{\partial m_i} + \sum_{j=1}^{M}\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_j \partial m_i}\Delta m_j, i = 1, 2, \ldots, M. \tag{66}$$

Briefly speaking, it is

$$\frac{\partial E(\mathbf{m})}{\partial \mathbf{m}} = \frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} + \frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}\Delta\mathbf{m} \tag{67}$$

Thus,

$$\Delta\mathbf{m} = -\left(\frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}\right)^{-1}\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} \tag{68}$$

where

$$\nabla E_{\mathbf{m}} = \frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} = \left[\frac{\partial E(\mathbf{m}_0)}{\partial m_1}, \frac{\partial E(\mathbf{m}_0)}{\partial m_2}, \ldots, \frac{\partial E(\mathbf{m}_0)}{\partial m_M}\right]^T \tag{69}$$

and

$$\mathbf{H}=\frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}=\left[\frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j}\right]=\begin{bmatrix} \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_1^2} & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_1 m_2} & \cdots & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_1 m_M} \\ \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_2 m_1} & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_2^2} & \cdots & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_2 m_M} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_M m_1} & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_M m_2} & \cdots & \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_M^2} \end{bmatrix}. \tag{70}$$

$\nabla E_{\mathbf{m}}$ and $\mathbf{H}$ are the gradient vector and the Hessian matrix, respectively.

The Newton, Gauss-Newton, and steepest-descent methods

In terms of Eq. (64),

$$\begin{aligned}\frac{\partial E(\mathbf{m})}{\partial m_i}&=\frac{1}{2}\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)(p_{cal}-p_{obs})^*+\left(\frac{\partial p_{cal}}{\partial m_i}\right)^*(p_{cal}-p_{obs})\right]\\ &=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)^*\Delta p\right](\Delta p=p_{cal}-p_{obs})\\ &=\mathtt{Re}\left[\left(\frac{\partial \mathbf{p}_{cal}}{\partial m_i}\right)^\dagger\Delta\mathbf{p}\right]=\mathtt{Re}\left[\left(\frac{\partial \mathbf{f}(\mathbf{m})}{\partial m_i}\right)^\dagger\Delta\mathbf{p}\right],i=1,2,\ldots,M.\end{aligned} \tag{71}$$

That is to say,

$$\nabla E_{\mathbf{m}}=\nabla E(\mathbf{m})=\frac{\partial E(\mathbf{m})}{\partial \mathbf{m}}=\mathtt{Re}\left[\left(\frac{\partial \mathbf{f}(\mathbf{m})}{\partial \mathbf{m}}\right)^\dagger\Delta\mathbf{p}\right]=\mathtt{Re}\left[\mathbf{J}^\dagger\Delta\mathbf{p}\right] \tag{72}$$

where $\mathtt{Re}$ takes the real part, and $\mathbf{J}=\frac{\partial \mathbf{p}_{cal}}{\partial \mathbf{m}}=\frac{\partial \mathbf{f}(\mathbf{m})}{\partial \mathbf{m}}$ is the Jacobian matrix, i.e., the sensitivity or the Fréchet derivative matrix.

Differentiation of the gradient expression (71) with respect to the model parameters gives the following expression for the Hessian $\mathbf{H}$:

$$\begin{aligned}\mathbf{H}_{i,j}&=\frac{\partial^2 E(\mathbf{m})}{\partial m_i\partial m_j}=\frac{\partial}{\partial m_j}\left(\frac{\partial E(\mathbf{m})}{\partial m_i}\right)=\frac{\partial}{\partial m_j}\mathtt{Re}\left[\left(\frac{\partial \mathbf{p}_{cal}}{\partial m_i}\right)^\dagger\Delta\mathbf{p}\right]\\ &=\frac{\partial}{\partial m_j}\mathtt{Re}\left[\left(\frac{\partial \mathbf{p}_{cal}}{\partial m_i}\right)^T\Delta\mathbf{p}^*\right]=\mathtt{Re}\left[\frac{\partial}{\partial m_j}\left(\frac{\partial \mathbf{p}_{cal}}{\partial m_i}\right)^T\Delta\mathbf{p}^*\right]+\mathtt{Re}\left[\frac{\partial \mathbf{p}_{cal}^\dagger}{\partial m_i}\frac{\partial \mathbf{p}_{cal}}{\partial m_j}\right]\end{aligned} \tag{73}$$

In matrix form

$$\mathbf{H}=\frac{\partial^2 E(\mathbf{m})}{\partial \mathbf{m}^2}=\mathtt{Re}\left[\mathbf{J}^\dagger\mathbf{J}\right]+\mathtt{Re}\left[\frac{\partial \mathbf{J}^T}{\partial \mathbf{m}^T}(\Delta\mathbf{p}^*,\Delta\mathbf{p}^*,\ldots,\Delta\mathbf{p}^*)\right]. \tag{74}$$

In many cases, this second-order term is neglected for nonlinear inverse problems. In the following, the remaining term in the Hessian, i.e., $\mathbf{H}_a = \mathtt{Re}[\mathbf{J}^\dagger\mathbf{J}]$, is referred to as the approximate Hessian. It is the auto-correlation of the derivative wavefield. Eq. (68) becomes

$$\Delta\mathbf{m} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} = -\mathbf{H}_a^{-1}\mathtt{Re}[\mathbf{J}^\dagger\Delta\mathbf{p}]. \tag{75}$$

The method which solves equation (74) when only $\mathbf{H}_a$ is estimated is referred to as the Gauss-Newton method. To guarantee th stability of the algorithm (avoiding the singularity), we can use $\mathbf{H} = \mathbf{H}_a + \eta\mathbf{I}$, leading to

$$\Delta\mathbf{m} = -\mathbf{H}^{-1}\nabla E_{\mathbf{m}} = -(\mathbf{H}_a + \eta\mathbf{I})^{-1}\mathtt{Re}\left[\mathbf{J}^\dagger\Delta\mathbf{p}\right]. \tag{76}$$

Alternatively, the inverse of the Hessian in Eq. (68) can be replaced by $\mathbf{H} = \mathbf{H}_a = \mu\mathbf{I}$, leading to the gradient or steepest-descent method:

$$\Delta\mathbf{m} = -\mu^{-1}\nabla E_{\mathbf{m}} = -\alpha\nabla E_{\mathbf{m}} = -\alpha\mathtt{Re}\left[\mathbf{J}^\dagger\Delta\mathbf{p}\right], \alpha = \mu^{-1}. \tag{77}$$

At the k-th iteration, the misfit function can be presented using the 2nd-order Taylor-Lagrange expansion

$$E(\mathbf{m}_{k+1}) = E(\mathbf{m}_k - \alpha_k\nabla E(\mathbf{m}_k)) = E(\mathbf{m}_k) - \alpha_k\langle\nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k)\rangle + \frac{1}{2}\alpha_k^2\nabla E(\mathbf{m}_k)^\dagger\mathbf{H}_k\nabla E(\mathbf{m}_k). \tag{78}$$

Setting $\frac{\partial E(\mathbf{m}_{k+1})}{\partial\alpha_k} = 0$ gives

$$\alpha_k = \frac{\nabla E(\mathbf{m}_k)^\dagger\nabla E(\mathbf{m}_k)}{\nabla E(\mathbf{m}_k)^\dagger\mathbf{H}_k\nabla E(\mathbf{m}_k)} \overset{\mathbf{H}_k := \mathbf{H}_a = \mathbf{J}_k^\dagger\mathbf{J}_k}{=} \frac{\nabla E(\mathbf{m}_k)^\dagger\nabla E(\mathbf{m}_k)}{\langle\mathbf{J}_k\nabla E(\mathbf{m}_k), \mathbf{J}_k\nabla E(\mathbf{m}_k)\rangle} \tag{79}$$

Conjugate gradient (CG) implementation

The gradient-like method can be summarized as

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \alpha_k\mathbf{d}_k. \tag{80}$$

The conjugate gradient (CG) algorithm decreases the misfit function along the conjugate gradient direction:

$$\mathbf{d}_k = \begin{cases} -\nabla E(\mathbf{m}_0), & k = 0 \\ -\nabla E(\mathbf{m}_k) + \beta_k\mathbf{d}_{k-1}, & k \geq 1 \end{cases} \tag{81}$$

There are many ways to compute β_k:

$$\begin{cases} \beta_k^{HS} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1})\rangle}{\langle \mathbf{d}_{k-1}, \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1})\rangle} \\ \beta_k^{FR} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k)\rangle}{\langle \nabla E(\mathbf{m}_{k-1}), \nabla E(\mathbf{m}_{k-1})\rangle} \\ \beta_k^{PRP} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1})\rangle}{\langle \nabla E(\mathbf{m}_{k-1}), \nabla E(\mathbf{m}_{k-1})\rangle} \\ \beta_k^{CD} = -\dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k)\rangle}{\langle \mathbf{d}_{k-1}, \nabla E(\mathbf{m}_{k-1})\rangle} \\ \beta_k^{DY} = \dfrac{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k)\rangle}{\langle \mathbf{d}_{k-1}, \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1})\rangle} \end{cases} \tag{82}$$

To achieve best convergence rate, in practice we suggest to use a hybrid scheme combing Hestenes-Stiefel and Dai-Yuan:

$$\beta_k = \max(0, \min(\beta_k^{HS}, \beta_k^{DY})). \tag{83}$$

Iterating with Eq. (80) needs to find an appropriate α_k. Here we provide two approaches to calculate α_k.

- Approach 1: Currently, the objective function is

$$E(\mathbf{m}_{k+1}) = E(\mathbf{m}_k + \alpha_k \mathbf{d}_k) = E(\mathbf{m}_k) + \alpha_k \langle \nabla E(\mathbf{m}_k), \mathbf{d}_k\rangle + \frac{1}{2}\alpha_k^2 \mathbf{d}_k^\dagger \mathbf{H}_k \mathbf{d}_k. \tag{84}$$

Setting $\frac{\partial E(\mathbf{m}_{k+1})}{\partial \alpha_k} = 0$ gives

$$\alpha_k = -\frac{\langle \mathbf{d}_k, \nabla E(\mathbf{m}_k)\rangle}{\mathbf{d}_k^\dagger \mathbf{H}_k \mathbf{d}_k} \overset{\mathbf{H}_k := \mathbf{H}_a = \mathbf{J}_k^\dagger \mathbf{J}_k}{=} -\frac{\langle \mathbf{d}_k, \nabla E(\mathbf{m}_k)\rangle}{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{J}_k \mathbf{d}_k\rangle}. \tag{85}$$

- Approach 2: Recall that

$$\mathbf{f}(\mathbf{m}_k + \alpha_k \mathbf{d}_k) = \mathbf{f}(\mathbf{m}_k) + \frac{\partial \mathbf{f}(\mathbf{m}_k)}{\partial \mathbf{m}} \mathbf{d}_k + O(||\mathbf{d}_k||^2) = \mathbf{f}(\mathbf{m}_k) + \alpha_k \mathbf{J}_k \mathbf{d}_k + O(||\mathbf{d}_k||^2). \tag{86}$$

Using the 1st-order approximation, we have

$$\begin{aligned} E(\mathbf{m}_{k+1}) &= \frac{1}{2}||\mathbf{f}(\mathbf{m}_k + \alpha_k \mathbf{d}_k) - \mathbf{p}_{obs}||^2 \\ &\approx \frac{1}{2}||\mathbf{f}(\mathbf{m}_k) + \alpha_k \mathbf{J}_k \mathbf{d}_k - \mathbf{p}_{obs}||^2 = \frac{1}{2}||\mathbf{f}(\mathbf{m}_k) - \mathbf{p}_{obs} + \alpha_k \mathbf{J}_k \mathbf{d}_k||^2 \\ &= E(\mathbf{m}) + \alpha_k \langle \mathbf{J}_k \mathbf{d}_k, \mathbf{f}(\mathbf{m}_k) - \mathbf{p}_{obs}\rangle + \frac{1}{2}\alpha_k^2 \langle \mathbf{J}_k \mathbf{d}_k, \mathbf{J}_k \mathbf{d}_k\rangle. \end{aligned} \tag{87}$$

Setting $\frac{\partial E(\mathbf{m}_{k+1})}{\partial \alpha_k} = 0$ gives

$$\alpha_k = \frac{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{p}_{obs} - \mathbf{f}(\mathbf{m}_k)\rangle}{\langle \mathbf{J}_k \mathbf{d}_k, \mathbf{J}_k \mathbf{d}_k\rangle}. \tag{88}$$

In fact, Eq. (88) can also be obtained from Eq. (85) in terms of Eq. (72): $\nabla E_{\mathbf{m}} = \mathbf{J}^{\dagger}\Delta\mathbf{p}$.

In terms of Eq. (86), the term $\mathbf{J}_k\mathbf{d}_k$ is computed conventionally using a 1st-order-accurate finite difference approximation of the partial derivative of $\mathbf{f}$:

$$\mathbf{J}_k\mathbf{d}_k = \frac{\mathbf{f}(\mathbf{m}_k + \epsilon\mathbf{d}_k) - \mathbf{f}(\mathbf{m}_k)}{\epsilon} \tag{89}$$

with a small parameter ϵ. In practice, we chose an ϵ such that

$$\max(\epsilon|\mathbf{d}_k|) \leqslant \frac{\max(|\mathbf{m}_k|)}{100}. \tag{90}$$

Fréchet derivative

Recall that the basic acoustic wave equation can be specified as

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} - \nabla^2 p(\mathbf{x},t;\mathbf{x}_s) = f_s(\mathbf{x},t;\mathbf{x}_s). \tag{91}$$

where $f_s(\mathbf{x},t;\mathbf{x}_s) = f(t')\delta(\mathbf{x}-\mathbf{x}_s)\delta(t-t')$. The Green's function $\Gamma(\mathbf{x},t;\mathbf{x}_s,t')$ is defined by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 \Gamma(\mathbf{x},t;\mathbf{x}_s,t')}{\partial t^2} - \nabla^2 \Gamma(\mathbf{x},t;\mathbf{x}_s,t') = \delta(\mathbf{x}-\mathbf{x}_s)\delta(t-t'). \tag{92}$$

Thus the integral representation of the solution can be given by

$$\begin{aligned} p(\mathbf{x}_r,t;\mathbf{x}_s) &= \int_V \mathrm{d}\mathbf{x}\int \mathrm{d}t'\Gamma(\mathbf{x}_r,t;\mathbf{x},t')f(\mathbf{x},t';\mathbf{x}_s) \\ &= \int_V \mathrm{d}\mathbf{x}\int \mathrm{d}t'\Gamma(\mathbf{x}_r,t-t';\mathbf{x},0)f(\mathbf{x},t';\mathbf{x}_s)(\text{Causility of Green}'\text{s function}) \\ &= \int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,t;\mathbf{x},0) * f(\mathbf{x},t;\mathbf{x}_s) \end{aligned} \tag{93}$$

where $*$ denotes the convolution operator.

A perturbation $v(\mathbf{x}) \to v(\mathbf{x}) + \Delta v(\mathbf{x})$ will produce a field $p(\mathbf{x},t;\mathbf{x}_s) + \Delta p(\mathbf{x},t;\mathbf{x}_s)$ defined by

$$\frac{1}{(v(\mathbf{x})+\Delta v(\mathbf{x}))^2}\frac{\partial^2[p(\mathbf{x},t;\mathbf{x}_s)+\Delta p(\mathbf{x},t;\mathbf{x}_s)]}{\partial t^2} - \nabla^2[p(\mathbf{x},t;\mathbf{x}_s)+\Delta p(\mathbf{x},t;\mathbf{x}_s)] = f_s(\mathbf{x},t;\mathbf{x}_s) \tag{94}$$

Note that

$$\frac{1}{(v(\mathbf{x})+\Delta v(\mathbf{x}))^2} = \frac{1}{v^2(\mathbf{x})} - \frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} + O(\Delta^2 v(\mathbf{x})) \tag{95}$$

Eq. (94) subtracts Eq. (91), yielding

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2\Delta p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}-\nabla^2\Delta p(\mathbf{x},t;\mathbf{x}_s)=\frac{\partial^2[p(\mathbf{x},t;\mathbf{x}_s)+\Delta p(\mathbf{x},t;\mathbf{x}_s)]}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} \tag{96}$$

Using the Born approximation, Eq. (96) becomes

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2\Delta p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}-\nabla^2\Delta p(\mathbf{x},t;\mathbf{x}_s)=\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})} \tag{97}$$

Again, based on integral representation, we obtain

$$\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)=\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,t;\mathbf{x},0)*\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2\Delta v(\mathbf{x})}{v^3(\mathbf{x})}. \tag{98}$$

Gradient computation

According to the previous section, it follows that

$$\frac{\partial p_{cal}}{\partial v_i(\mathbf{x})}=\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,t;\mathbf{x},0)*\ddot{p}(\mathbf{x},t;\mathbf{x}_s)\frac{2}{v^3(\mathbf{x})}=\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,t;\mathbf{x},0)*\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}. \tag{99}$$

The convolution guarantees

$$\int \mathrm{d}t[g(t)*f(t)]h(t)=\int \mathrm{d}t f(t)[g(-t)*h(t)]. \tag{100}$$

Then, Eq. (71) becomes

$$\begin{aligned}
\frac{\partial E(\mathbf{m})}{\partial m_i}&=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int \mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial p_{cal}}{\partial m_i}\right)^*\Delta p\right](\Delta p=p_{cal}-p_{obs})\\
&=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}}\mathrm{d}t\mathtt{Re}\left[\left(\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,t;\mathbf{x},0)*\frac{\partial^2 p(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)\right]\\
&=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}}\mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*(\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,-t;\mathbf{x},0)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s))\right]\\
&=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}}\mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^*(\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,0;\mathbf{x},t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s))\right]\\
&=\sum_{r=1}^{ng}\sum_{s=1}^{ns}\int_0^{t_{\max}}\mathrm{d}t\mathtt{Re}\left[\left(\frac{\partial^2 p_{cal}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2}\frac{2}{v^3(\mathbf{x})}\right)^* p_{res}(\mathbf{x}_r,t;\mathbf{x}_s)\right]
\end{aligned} \tag{101}$$

where $p_{res}(\mathbf{x},t;\mathbf{x}_s)$ is a time-reversal wavefield produced using the residual $\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)$ as the source. It follows from reciprocity theorem

$$p_{res}(\mathbf{x},t;\mathbf{x}_s)=\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x}_r,0;\mathbf{x},t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s)=\int_V \mathrm{d}\mathbf{x}\Gamma(\mathbf{x},0;\mathbf{x}_r,t)*\Delta p(\mathbf{x}_r,t;\mathbf{x}_s). \tag{102}$$

satisfying

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p_{res}(\mathbf{x},t;\mathbf{x}_s)}{\partial t^2} - \nabla^2 p_{res}(\mathbf{x},t;\mathbf{x}_s) = \Delta p(\mathbf{x}_r,t;\mathbf{x}_s). \tag{103}$$

It is noteworthy that an input f and the system impulse response function g are exchangeable in convolution. That is to say, we can use the system impulse response function g as the input, the input f as the impulse response function, leading to the same output. In the seismic modeling and acquisition process, the same seismogram can be obtained when we shot at the receiver position $\mathbf{x}_r$ when recording the seismic data at the position $\mathbf{x}$.

Numerical results

I use the Marmousi model for the benchmark test, as shown in the top panel of Figure 4. FWI tacitly requires a good starting model incorporated with low frequency information. Therefore, we use a starting model (bottom panel of Figure 4) obtained by smoothing the original model 20 times with a 5x5 window.

The FWI is carried out for 300 iterations. We record all the updated velocity to make sure the velocity refinement is going on during the iterative procedure. The updated velocity model at the iteration 1, 20, 50, 100, 180 and 300 are displyed in Figure 6. Figure 7 presents the decreasing misfit function in iterations. As can be seen from the Figures 6 and 7, the velocity model changes significantly at the early stage. Later iterations in FWI make some improvement on small details for the velocity model.

ACKNOWLEDGEMENT

I would like to thank to Sergey Fomel for the encouragement and the correction of this work, which leads to significant improvement of the tutorial. Special thanks to Baoli Wang, who helps me a lot in GPU programming.

REFERENCES

Baysal, E., D. D. Kosloff, and J. W. Sherwood, 1983, Reverse time migration: Geophysics, **48**, 1514–1524.

Berenger, J.-P., 1994, A perfectly matched layer for the absorption of electromagnetic waves: Journal of computational physics, **114**, 185–200.

Biondi, B., 2006, 3d seismic imaging: Society of Exploration Geophysicists.

Bjorck, A., 1996, Numerical methods for least squares problems: Society for Industrial and Applied Mathematics.

Carcione, J. M., G. C. Herman, and A. Ten Kroode, 2002, Seismic modeling: Geophysics, **67**, 1304–1325.

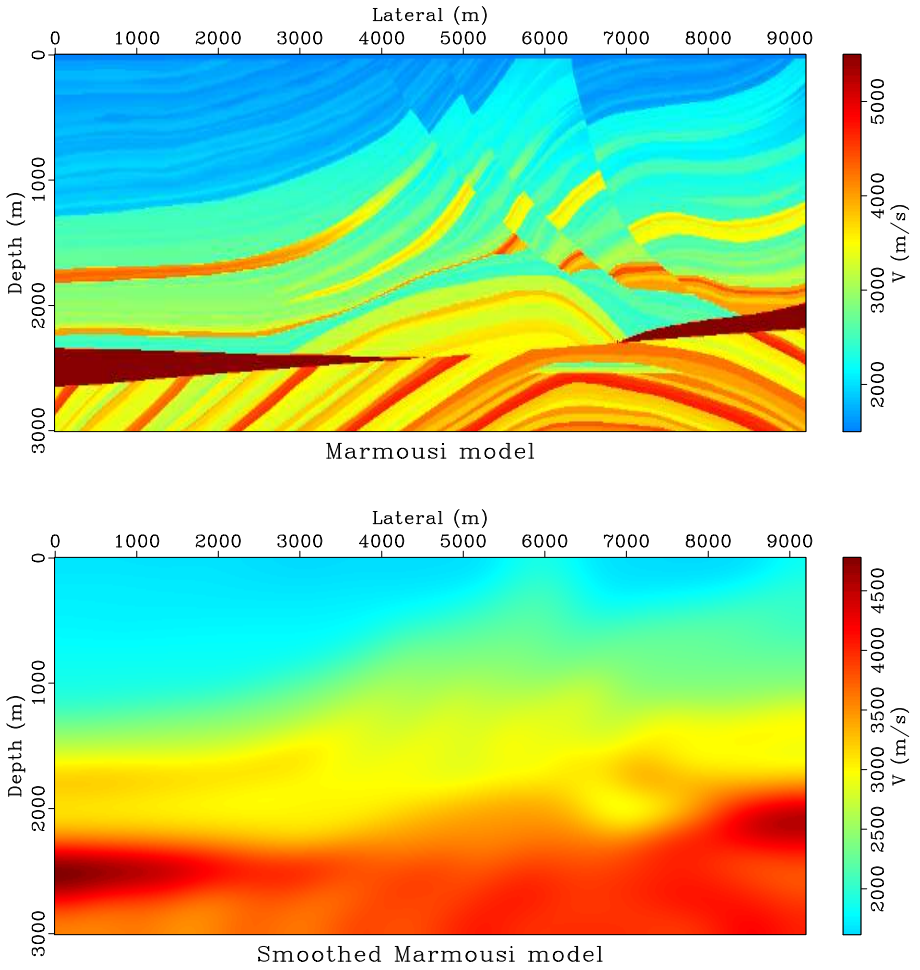


Figure 15: Top: The original Marmousi is downsampled with a factor of 3 along depth and lateral direction. The shots are generated according to the subsampled Marmousi model. Bottom: The starting model of FWI for Marmousi model, which is obtained by smoothing the original model 20 times with a 5x5 window. primer/marmfwi/ marm

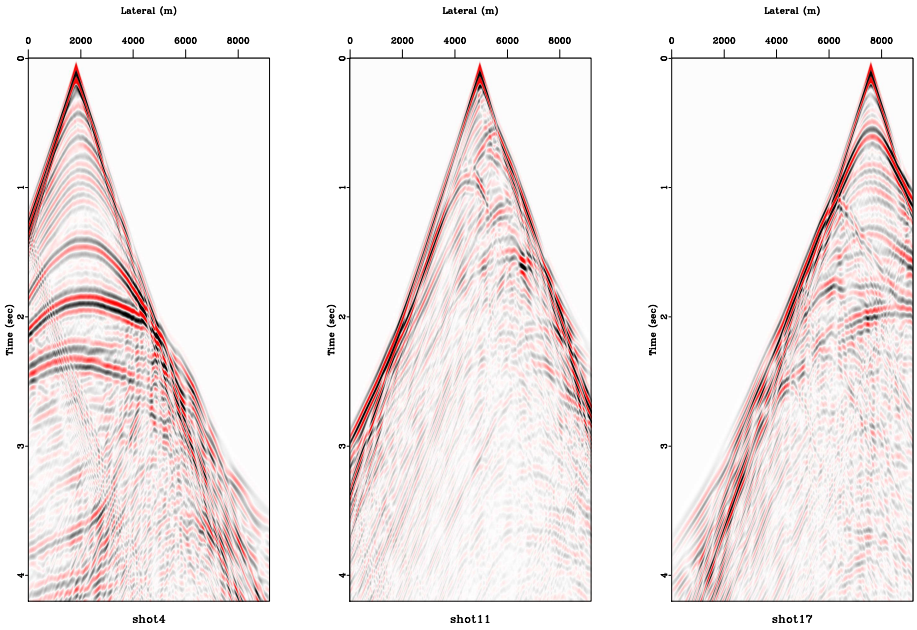


Figure 16: 21 shots were deployed in the FWI. Here, shot 4, 11 and 17 are shown from left to right primer/marmfwi/ shotsnap

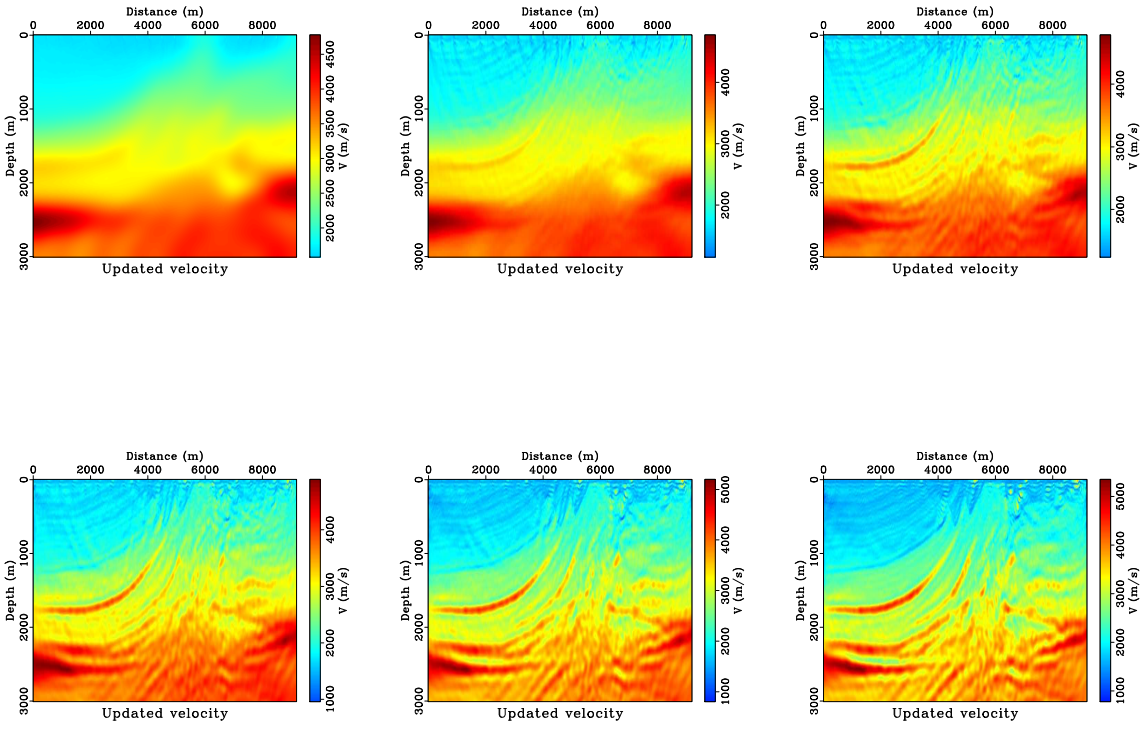


Figure 17: The updated velocity model at the iteration 1, 20, 50, 100, 180 and 300. primer/marmfwi/ vsnap

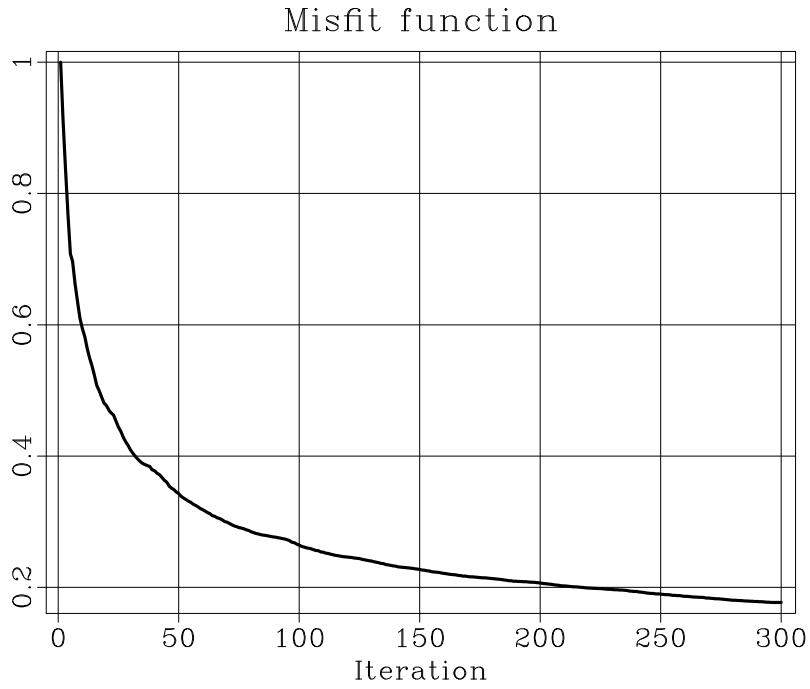


Figure 18: The misfit function decreases with the iterations. primer/marmfwi/ objs

Cerjan, C., D. Kosloff, R. Kosloff, and M. Reshef, 1985, A nonreflecting boundary condition for discrete acoustic and elastic wave equations: Geophysics, **50**, 705–708.

Claerbout, J., 1986, Imaging the earth's interior: Geophysical Journal of the Royal Astronomical Society, **86**, 217–217.

Claerbout, J. F., 1971, Toward a unified theory of reflector mapping: Geophysics, **36**, 467–481.

Clapp, R. G., 2009, Reverse time migration with random boundaries: 79th Annual International Meeting, SEG Expanded Abstracts, 2809–2813.

Clapp, R. G., H. Fu, and O. Lindtjorn, 2010, Selecting the right hardware for reverse time migration: The Leading Edge, **29**, 48–58.

Clayton, R., and B. Engquist, 1977, Absorbing boundary conditions for acoustic and elastic wave equations: Bulletin of the Seismological Society of America, **67**, 1529–1540.

Collino, F., and C. Tsogka, 2001, Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media: Geophysics, **66**, 294–307.

DiMarco, S., R. Reid, A. Jochens, W. Nowlin Jr, M. Howard, et al., 2001, General characteristics of currents in the deepwater gulf of mexico: Presented at the Offshore Technology Conference, Offshore Technology Conference.

Dussaud, E., W. W. Symes, P. Williamson, L. Lemaistre, P. Singer, B. Denel, and A. Cherrett, 2008, Computational strategies for reverse-time migration: SEG Annual meeting.

Engquist, B., and A. Majda, 1977, Absorbing boundary conditions for numerical

simulation of waves: Proceedings of the National Academy of Sciences, **74**, 1765–1766.

Guitton, A., B. Kaelin, and B. Biondi, 2006, Least-squares attenuation of reverse-time-migration artifacts: Geophysics, **72**, S19–S23.

Guitton, A., A. Valenciano, D. Bevc, and J. Claerbout, 2007, Smoothing imaging condition for shot-profile migration: Geophysics, **72**, S149–S154.

Komatitsch, D., and R. Martin, 2007, An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation: Geophysics, **72**, SM155–SM167.

Liu, G., Y. Liu, L. Ren, and X. Meng, 2013a, 3d seismic reverse time migration on gpgpu: Computers & Geosciences, **59**, 17 – 23.

Liu, H., R. Ding, L. Liu, and H. Liu, 2013b, Wavefield reconstruction methods for reverse time migration: Journal of Geophysics and Engineering, **10**, 015004.

McMechan, G., 1983, Migration by extrapolation of time-dependent boundary values: Geophysical Prospecting, **31**, 413–420.

Pica, A., J. Diet, and A. Tarantola, 1990, Nonlinear inversion of seismic reflection data in a laterally invariant medium: Geophysics, **55**, 284–292.

Pratt, G., C. Shin, et al., 1998, Gauss–newton and full newton methods in frequency–space seismic waveform inversion: Geophysical Journal International, **133**, 341–362.

Roden, J. A., and S. D. Gedney, 2000, Convolutional pml (cpml): An efficient fdtd implementation of the cfs-pml for arbitrary media: Microwave and optical technology letters, **27**, 334–338.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, SM213–SM221.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

———, 1986, A strategy for nonlinear elastic inversion of seismic reflection data: Geophysics, **51**, 1893–1903.

Versteeg, R., 1994, The marmousi experience: Velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927–936.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, WCC1–WCC26.

Yang, P., J. Gao, and B. Wang, 2014, {RTM} using effective boundary saving: A staggered grid {GPU} implementation: Computers & Geosciences.

Yoon, K., and K. J. Marfurt, 2006, Reverse-time migration using the poynting vector: Exploration Geophysics, **37**, 102–107.

Yoon, K., C. Shin, S. Suh, L. R. Lines, and S. Hong, 2003, 3d reverse-time migration using the acoustic wave equation: An experience with the seg/eage data set: The Leading Edge, **22**, 38–41.

Fourier pseudo spectral method for attenuative simulation with fractional Laplacian

Pengliang Yang[*]

ABSTRACT

This tutorial is devoted to pseudo-spectral method (PSM) by reorganizing the course material from Prof. Heiner Igel and a paper by J.M. Carcione, to illustrate how to implement viscoacoustic wave simulation based on fractional Laplacian operator.

WHAT IS A PSEUDO-SPECTRAL METHOD?

Spectral solutions to time-dependent PDEs are formulated in the frequency-wavenumber domain and solutions are obtained in terms of spectra (e.g. seismograms). This technique is particularly interesting for geometries where partial solutions in the $\omega - k$ domain can be obtained analytically (e.g. for layered models).

In the pseudo-spectral approach - in a finite-difference like manner - the PDEs are solved pointwise in physical space ($x - t$). However, the space derivatives are calculated using orthogonal functions (e.g. Fourier Integrals, Chebyshev polynomials). They are either evaluated using matrix-matrix multiplications, fast Fourier transform (FFT), or convolutions.

Let us start with the 1-D acoustic wave equation.

$$\frac{1}{v^2}\partial_{tt}p = \rho\partial_x\left(\frac{1}{\rho}\partial_x p\right) + f \tag{1}$$

Omitting the source term, we may discretize the wave equation using standard centered finite difference for time stepping as

$$\frac{p^{n+1} - 2p^n + p^{n-1}}{\rho c^2 \Delta t^2} = \partial_x\left(\frac{1}{\rho}\partial_x p\right) \tag{2}$$

where we use the notation $p(n\Delta t) := p^n$. Thus, we have the following evolution scheme

$$p^{n+1} = 2p^n - p^{n-1} + \rho c^2 \Delta t^2 \partial_x\left(\frac{1}{\rho}\partial_x p\right) \tag{3}$$

where the space derivatives will be calculated using the Fourier transform.

[*]**e-mail:** ypl.2100@gmail.com

COMPUTING DERIVATIVES USING FOURIER TRANSFORM

The Fourier transform of a function $f(x)$ and the inverse are define respectively

$$\tilde{f}(\omega) = F[f] = \int f(x) e^{-i\omega x} \mathrm{d}x \tag{4}$$

and

$$f(x) = F^{-1}[f] = \frac{1}{2\pi} \int \tilde{f}(\omega) e^{i\omega x} \mathrm{d}\omega \tag{5}$$

where $\tilde{f}$ is the Fourier transform of f. In spatial dimension, ω corresponds to the wavenumber k. Therefore we have

$$\partial_x f = \partial_x \left(\frac{1}{2\pi} \int \tilde{f} e^{ik_x x} \mathrm{d}x \right) = \frac{1}{2\pi} \int \underline{ik_x \tilde{f}} e^{ik_x x} \mathrm{d}x \tag{6}$$

and

$$\partial_{xx} f = \partial_{xx} \left(\frac{1}{2\pi} \int \tilde{f} e^{ik_x x} \mathrm{d}x \right) = \frac{1}{2\pi} \int \underline{(ik_x)^2 \tilde{f}} e^{ik_x x} \mathrm{d}x \tag{7}$$

The term $\partial_x \left(\frac{1}{\rho} \partial_x p \right)$ will be calculated by the following precedure:

$$\begin{aligned}
p &\xrightarrow{F} \tilde{p} \\
&\xrightarrow{ik_x} ik_x \tilde{p} \\
&\xrightarrow{F^{-1}} \partial_x p = F^{-1}[ik_x \tilde{p}] \\
&\xrightarrow{\frac{1}{\rho}} \frac{1}{\rho} F^{-1}[ik_x \tilde{p}] \\
&\xrightarrow{F} F[\tfrac{1}{\rho} F^{-1}[ik_x \tilde{p}]] \\
&\xrightarrow{ik_x} ik_x F[\tfrac{1}{\rho} F^{-1}[ik_x \tilde{p}]] \\
&\xrightarrow{F^{-1}} F^{-1}[ik_x F[\tfrac{1}{\rho} F^{-1}[ik_x \tilde{p}]]]
\end{aligned} \tag{8}$$

Let us conduct a 2-D numerical simulation of acoustic wave equation with constant density. The equation is then

$$\partial_{tt} p = c^2 (\partial_{xx} + \partial_{zz}) p \tag{9}$$

Based upon Fourier transform for spatial axis, we know the right hand side of the above equation corresponds to

$$-(k_x^2 + k_z^2)\tilde{p} \tag{10}$$

Thus, we have the following time evolution

$$p^{n+1} = 2p^n - p^{n-1} + \Delta t^2 c^2 F^{-1}[-(k_x^2 + k_z^2) F p^n] \tag{11}$$

FRACTIONAL LAPLACIAN

According to Carcione (2010), the uniform-density pressure formulation is

$$\partial_t^2 p = \omega_0^{2-2\beta} c^{2\beta} (\partial_x^2 + \partial_z^2)^\beta + s \tag{12}$$

where s is the body force per unit; the order β is $1 \leq \beta \leq 2$. When $\beta \to 2$, it introduces stronger attenuation. Regarding the fractional order of the Laplacian operator, we may be able to update the wavefield by

$$p^{n+1} = 2p^n - p^{n-1} + \Delta t^2 c^2 F^{-1}[(-k_x^2 - k_z^2)^\beta F p^n] \tag{13}$$

Another choice to perform fractional order wave simulation is

$$\rho(\partial_x^\beta \rho^{-1} \partial_x^\beta + \partial_z^\beta \rho^{-1} \partial_z^\beta) \tag{14}$$

which implies the following wavefield extrapolation

$$p^{n+1} = 2p^n - p^{n-1} + \Delta t^2 c^2 F^{-1}[(-1)^\beta (k_x^{2\beta} + k_z^{2\beta}) F p^n] \tag{15}$$

with constant density ρ.

A snapshot using the code provided in the appendix is shown in Figure 1, in which we use the sponge boundary condition.

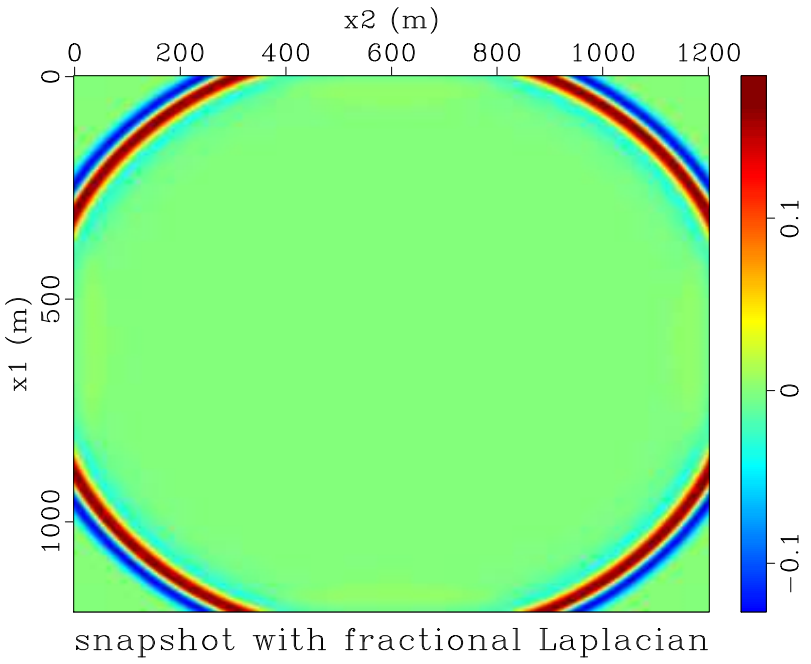


Figure 1: A snapshot of 2-D acoustic propagation using PSM method fraclap/ps2d/ snapshot

CONCLUSION

The Fourier method can be considered as the limit of the finite-difference method as the length of the operator tends to the number of points along a particular dimension. The space derivatives are calculated in the wavenumber domain by multiplication of the spectrum with ik. The inverse Fourier transform results in an exact space derivative up to the Nyquist frequency. The use of Fourier transform imposes some constraints on the smoothness of the functions to be differentiated. Discontinuities lead to Gibbs' phenomenon. As the Fourier transform requires periodicity this technique is particular useful where the physical problems are periodical (e.g. angular derivatives in cylindrical problems).

By introducing fractional order of the Laplacian operator, we are able to perform wave simulation in attenuative medium. The computation of fractional Laplacian can be easily carried out by using Fourier pseudo spectral method, which is computational accurate and efficient enough without any additional storage when considering a constant Q model.

ABSORBING BOUNDARY CONDITION

One of the easiest absorbing boundary condtion (ABC) is sponge (or referred to as Gaussian taper) boundary condition proposed by Cerjan et al. (1985). The principle is very simple: Attenuating the refections exponentially in the extended artificial boundary (Figure A-1) area by multiplying a factor less $d(u)$ than 1, i.e., $d(u) = \exp(-[\alpha(nb-i)]^2), u = x, z(i\Delta x$ or $i\Delta z)$ where nb is the thickness of the artificial boundary on each side of the model. A usual choice is $\alpha = 0.015$ for $nb = 20 \sim 30$ absorbing layers. The sponge ABC can be easily applied to a wide range of wave propagation problems, including some governing wave equations for complicated medium.

REFERENCES

José M Carcione. A generalization of the fourier pseudospectral method. *Geophysics*, 75(6):A53–A56, 2010.

C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50(4):2117–2131, 1985.

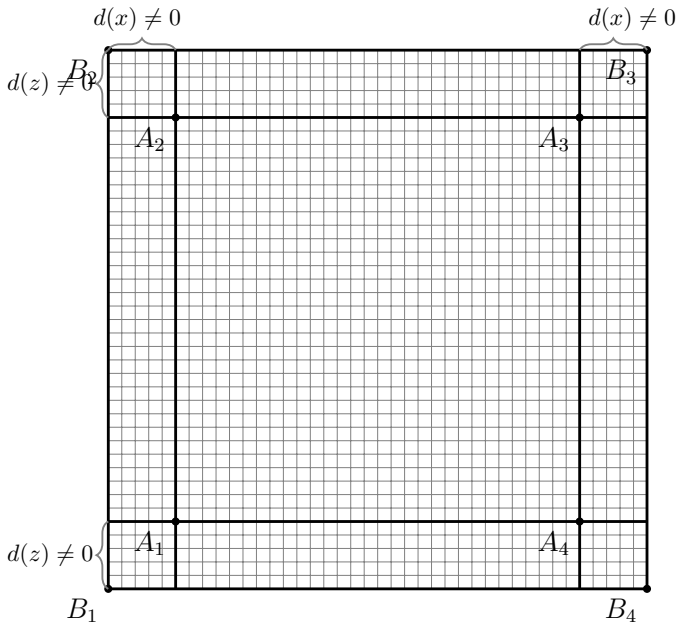


Figure A-1: A schematic diagram of extended artificial boundary area. $A_1A_2A_3A_4$ is the original model zone, which is extended to be $B_1B_2B_3B_4$ with artificial boundary. In the extended bounary area, the attenuation coeffcient $d(u) \neq 0$; In the model zone $A_1A_2A_3A_4$, $d(u) = 0$, $u = x, z$.

From modeling to full waveform inversion: A hands-on tour using Madagascar

Pengliang Yang
ISTerre - Univ. Grenoble Alpes
*E-mail: ypl.2100@gmail.com**

ABSTRACT

This tutorial is devoted to Madagascar school 2016 Zurich. In this tutorial, there are two aspects we would like to explore:

- Madagascar functionality, which is the tool. We may consider Madagascar to facilitate our research, from the numerical test to publication.
- Scientific aspects, which are the key things we care. Even though we are playing a game with simple exercise, we have to think about the scientific enhancement/improvement to polish the techniques used in modeling and inversion applications.

INTRODUCTION

As a brief introduction, I want to emphasize several points from my own understanding:

- Do you really need Madagascar? Yes because you will benefit a lot from it. Of course no if you are able to manage things in your own way, even better than Madagascar.
- Assuming we need Madagascar hereafter. In what aspects we can benefit from it?
- Is it a responsibility to contribute your papers, in particular your codes? Definitely no, especially when your research is sponsored by others while a permission public release is not accessible. You may want to be selfish: "I only want to use the codes from others instead of sharing mine with the community". No objection: Some people are doing things like that.
- Keep in mind Madagascar is not the goal, it is just a tool to share your research progress with others. If you are ready to be open, why not a contributor?

*e-mail: ypl.2100@gmail.com

PRELIMINARY

In this section, I provide some fundamental gadgets to help the beginners run Madagascar with ease.

Reproduce numerical examples and papers using SConstruct

Processing workflow:

- `Fetch`(data_file,dir,ftp_server_info): download data_file from a specific directory dir of an FTP server
- `Flow`(targets,sources,commands): generate target[s] from source[s] using command[s]
- `Plot`(intermediate_plot[,source],plot_command): generate intermediate_plot in the working directory
- `Result`(plot,intermediate_plots,combination) generate a final plot in Fig folder of the working directory
- `End`(): collect default targets

Run scons for your computation

scons run an SConstruct to generate data

scons view view the results from an SConstruct

scons -c clean the local directory, delete all target files

pscons parallel execution of an SConstruct

Paper Sconstruct imports Python packages for processing TeX files:

```
1 from rsf.tex import *
2 End(name,lclass,options,use)
```

- name - name of the root tex file to build, paper.tex.
- lclass - name of the LaTeX class file to use.
- options - document options for LaTeX class file.
- use - names of LaTeX packages to import during compilation.

To generate your paper including numerical examples:

- `sftour scons lock`: lock the results from an SConstruct
- `scons read/paper.read`: look at the generated paper in pdf
- `scons -c` remove all intermediate files

Some of the most useful commands

Plotting the figures

sfgraph create line plots, or scatter plots

sfgrey create raster plots or 2D image plots

sfgrey3 create 3D image plots of panels (or slices) of a 3D cube

sfwiggle plot data with wiggly traces

Look up data attributes and data processing:

sfin check the layout of the data, number of points in each dimension, sampling intervals in each axis, labels, units, ...

sfattr check statistical properties: covariable, rms, mean, minimum and maximum etc

sfwindow window or select part of data

sfadd add two dataset with scaling factors

sfmath mathematical operations for the data: log, sin, tan, exp, ...

sfsmooth smoothing the data using triangular window (repeating many time to approximate Gaussian)

Image format: Vplot

- suffix '.vpl', vectorized image→scaling without loss of quality
- convert to be `pdf/eps/png/jpeg/tiff/mpeg/...`
- how: `vpconvert format=pdf fig.vpl`

FORWARD MODELING

The wave equation we consider in this course material

$$(\frac{1}{v^2}\partial_t^2 - \nabla^2)p = f \tag{1}$$

Omitting the source, extrapolate your wavefield:

$$p^{k+1} = 2p^k - p^{k-1} + \Delta t^2 v^2 \nabla^2 p^k \tag{2}$$

where

$$\nabla^2 p = \frac{p[ix][iz+1] - 2p[ix][iz] + p[ix][iz-1]}{\Delta z^2} + \frac{p[ix-1][iz] - 2p[ix][iz] + p[ix-1][iz]}{\Delta x^2} \tag{3}$$

The Clayton-Enquist absorbing boundary condition (ABC) (Clayton and Engquist, 1977)

$$\text{left boundary}: \frac{\partial^2 p}{\partial x \partial t} - \frac{1}{v}\frac{\partial^2 p}{\partial t^2} = \frac{v}{2}\frac{\partial^2 p}{\partial z^2} \tag{4}$$

The codes in every time step looks like

```
//dtz=dt/dz; dtx=dt/dx
 void step_forward(float **p0, float **p1, float **p2,
      float **vv, float dtz, float dtx, int nz, int nx)
/*< forward modeling step, Clayton-Enquist ABC incorporated >*/
{
    int ix,iz;
    float v1,v2,diff1,diff2;

    for (ix=0; ix < nx; ix++)
    for (iz=0; iz < nz; iz++)
    {
    v1=vv[ix][iz]*dtz; v1=v1*v1;
    v2=vv[ix][iz]*dtx; v2=v2*v2;
    diff1=diff2=-2.0*p1[ix][iz];
    diff1+=(iz-1>=0)?p1[ix][iz-1]:0.0;
    diff1+=(iz+1<nz)?p1[ix][iz+1]:0.0;
    diff2+=(ix-1>=0)?p1[ix-1][iz]:0.0;
    diff2+=(ix+1<nx)?p1[ix+1][iz]:0.0;
    diff1*=v1;
    diff2*=v2;
    p2[ix][iz]=2.0*p1[ix][iz]-p0[ix][iz]+diff1+diff2;
    }

/*
    // top boundary
```

```
    iz=0;
    for (ix=1; ix < nx-1; ix++) {
        v1=vv[ix][iz]*dtz;
        v2=vv[ix][iz]*dtx;
        diff1=  (p1[ix][iz+1]-p1[ix][iz])-
                (p0[ix][iz+1]-p0[ix][iz]);
        diff2=   p1[ix-1][iz]-2.0*p1[ix][iz]+p1[ix+1][iz];
        diff1*=v1;
        diff2*=0.5*v2*v2;
        p2[ix][iz]=2.0*p1[ix][iz]-p0[ix][iz]+diff1+diff2;
    }
*/
    /* bottom boundary */
    iz=nz-1;
    for (ix=1; ix < nx-1; ix++) {
        v1=vv[ix][iz]*dtz;
        v2=vv[ix][iz]*dtx;
        diff1=-(p1[ix][iz]-p1[ix][iz-1])+
               (p0[ix][iz]-p0[ix][iz-1]);
        diff2=p1[ix-1][iz]-2.0*p1[ix][iz]+p1[ix+1][iz];
        diff1*=v1;
        diff2*=0.5*v2*v2;
        p2[ix][iz]=2.0*p1[ix][iz]-p0[ix][iz]+diff1+diff2;
    }

    /* left boundary */
    ix=0;
    for (iz=1; iz <nz-1; iz++){
        v1=vv[ix][iz]*dtz;
        v2=vv[ix][iz]*dtx;
        diff1=p1[ix][iz-1]-2.0*p1[ix][iz]+p1[ix][iz+1];
        diff2=(p1[ix+1][iz]-p1[ix][iz])-
              (p0[ix+1][iz]-p0[ix][iz]);
        diff1*=0.5*v1*v1;
        diff2*=v2;
        p2[ix][iz]=2.0*p1[ix][iz]-p0[ix][iz]+diff1+diff2;
    }

    /* right boundary */
    ix=nx-1;
    for (iz=1; iz <nz-1; iz++){
        v1=vv[ix][iz]*dtz;
        v2=vv[ix][iz]*dtx;
        diff1=p1[ix][iz-1]-2.0*p1[ix][iz]+p1[ix][iz+1];
        diff2=-(p1[ix][iz]-p1[ix-1][iz])+
```

```
                (p0[ix][iz]-p0[ix-1][iz]);
            diff1*=0.5*v1*v1;
            diff2*=v2;
            p2[ix][iz]=2.0*p1[ix][iz]-p0[ix][iz]+diff1+diff2;
        }
}
```

Write your own code and run it as a test

You have to

1. create a user directory in /RSFSRC/user/dirname, where dirname is the directory name, for example, 'pyang'.
2. copy a SConstruct from other existing users, and use it as a template to create your own things. For example, take /RSFSRC/user/psava/SConstruct. Assume you are going to do C programming to generate a target executable `sfmodeling2d`. You need to empty all other code list:

```
import os, sys

try:
    import bldutil
    glob_build = True # scons command launched in RSFSRC
    srcroot = '../..'
    Import('env bindir libdir pkgdir')
    env = env.Clone()
except:
    glob_build = False # scons in the local directory
    srcroot = os.environ.get('RSFSRC', '../..')
    sys.path.append(os.path.join(srcroot,'framework'))
    import bldutil
    env = bldutil.Debug() # Debugging flags for compilers
    bindir = libdir = pkgdir = None
    SConscript(os.path.join(srcroot,'api/c/SConstruct'))

targets = bldutil.UserSconsTargets()

# C mains
targets.c = '''
modeling2d
'''

```

```
25      targets.build_all(env, glob_build, srcroot,
26                        bindir, libdir, pkgdir)
```

3. Use text editor (emacs, gedit, ...) to create a file Mmodeling2d.c (which will generate the target `sfmodeling2d`, `M` will be automatically replaced by `sf`). In Mmodeling2d.c, start your codes by including the RSF header file: `#include <rsf.h>`, which defines many useful interfaces/subroutines for the convenience of data I/O (including parameters and files)

   ```
   sf_input()/sf_output()
   sf_histint()/sf_histfloat()
   sf_getint()/sf_getfloat()
   ```

   and memory allocation for the variables

   ```
   sf_intalloc()/sf_floatalloc()
   sf_intalloc2()/sf_floatalloc2()
   ...
   ```

   which will be used frequently when coding with Madagascar.

4. specify your input and output files, and initialize Madagascar:

```
1  sf_file vinit, shots;     /* input and output file*/
2  sf_init(argc,argv);       /* initialize Madagascar */
3  vinit=sf_input ("in");    /* initial velocity model in m/s */
4  shots=sf_output("out");   /* output image */
```

   Here the input file `vinit` is the velocity model, while the output `shots` is a shot gather (or many shots) collected at many receivers for different sources.

5. read the parameters from the input file using the interfaces Madagascar prepared: `sf_hist*(),sf_get*()`

```
1  /* get parameters for forward modeling */
2  if (!sf_histint(vinit,"n1",&nz)) sf_error("no n1");
3  if (!sf_histint(vinit,"n2",&nx)) sf_error("no n2");
4  if (!sf_histfloat(vinit,"d1",&dz)) sf_error("no d1");
5  if (!sf_histfloat(vinit,"d2",&dx)) sf_error("no d2");
6
7  if (!sf_getfloat("fm",&fm)) fm=10;
8  /* dominant freq of ricker */
9  if (!sf_getfloat("dt",&dt)) sf_error("no dt");
```

```
/* time interval */
if (!sf_getint("nt",&nt))    sf_error("no nt");
/* total modeling time steps */
if (!sf_getint("ns",&ns))    sf_error("no ns");
/* total shots */
if (!sf_getint("ng",&ng))    sf_error("no ng");
/* total receivers in each shot */
if (!sf_getint("jsx",&jsx))    sf_error("no jsx");
/* source x-axis  jump interval  */
if (!sf_getint("jsz",&jsz))    jsz=0;
/* source z-axis jump interval  */
if (!sf_getint("jgx",&jgx))    jgx=1;
/* receiver x-axis jump interval */
if (!sf_getint("jgz",&jgz))    jgz=0;
/* receiver z-axis jump interval */
if (!sf_getint("sxbeg",&sxbeg))    sf_error("no sxbeg");
/* x-begining index of sources, starting from 0 */
if (!sf_getint("szbeg",&szbeg))    sf_error("no szbeg");
/* z-begining index of sources, starting from 0 */
if (!sf_getint("gxbeg",&gxbeg))    sf_error("no gxbeg");
/* x-begining index of receivers, starting from 0 */
if (!sf_getint("gzbeg",&gzbeg))    sf_error("no gzbeg");
/* z-begining index of receivers, starting from 0 */
if (!sf_getbool("csdgather",&csdgather)) csdgather=false;
/* default, common shot-gather; if n, record at every point*/
```

6. specify the parameters for the output file using the interfaces Madagascar prepared: `sf_put*()`

```
/* put the labels, legends and parameters in output */
sf_putint(shots,"n1",nt);
sf_putint(shots,"n2",ng);
sf_putint(shots,"n3",ns);
sf_putfloat(shots,"d1",dt);
sf_putfloat(shots,"d2",jgx*dx);
sf_putfloat(shots,"o1",0);
sf_putstring(shots,"label1","Time");
sf_putstring(shots,"label2","Lateral");
sf_putstring(shots,"label3","Shot");
sf_putstring(shots,"unit1","sec");
sf_putstring(shots,"unit2","m");
sf_putfloat(shots,"amp",amp);
sf_putfloat(shots,"fm",fm);
sf_putint(shots,"ng",ng);
sf_putint(shots,"szbeg",szbeg);
```

```
sf_putint(shots,"sxbeg",sxbeg);
sf_putint(shots,"gzbeg",gzbeg);
sf_putint(shots,"gxbeg",gxbeg);
sf_putint(shots,"jsx",jsx);
sf_putint(shots,"jsz",jsz);
sf_putint(shots,"jgx",jgx);
sf_putint(shots,"jgz",jgz);
sf_putint(shots,"csdgather",csdgather?1:0);
```

7. allocate memory for the arries, format: `sf_floatalloc2(n1,n2)`

```
/* allocate the variables */
wlt=(float*)malloc(nt*sizeof(float));
bndr=(float*)malloc(nt*(2*nz+nx)*sizeof(float));
dobs=(float*)malloc(ng*nt*sizeof(float));
trans=(float*)malloc(ng*nt*sizeof(float));
vv=sf_floatalloc2(nz, nx);
p0=sf_floatalloc2(nz, nx);
p1=sf_floatalloc2(nz, nx);
p2=sf_floatalloc2(nz, nx);
sxz=(int*)malloc(ns*sizeof(int));
gxz=(int*)malloc(ng*sizeof(int));
```

8. do your own computation (forward simulation) as usual. The whole time stepping looks like

```
  memset(p0[0],0,nz*nx*sizeof(float));
  memset(p1[0],0,nz*nx*sizeof(float));
  memset(p2[0],0,nz*nx*sizeof(float));
  /* forward modeling */
  for(it=0; it<nt; it++)
  {
    add_source(p1, &wlt[it], &sxz[is], 1, nz, true);
    step_forward(p0, p1, p2, vv, dtz, dtx, nz, nx);
    ptr=p0; p0=p1; p1=p2; p2=ptr;
    record_seis(&dobs[it*ng], gxz, p0, ng, nz);
  }
```

9. free the variables

```
/* free the variables */
free(sxz);
free(gxz);
free(bndr);
free(dobs);
```

```
free(trans);
free(wlt);
free(*vv); free(vv);
free(*p0); free(p0);
free(*p1); free(p1);
free(*p2); free(p2);
```

10. Finally, you end up with a complete code /RSFSRC/user/pyang/Mmodeling2d.c. You can go into directory RSFSRC, compile and install the target executable `sfmodeling2d`:

    ```
    cd $RSFSRC
    scons install
    ```

    If there exists any error in your code, you will get the reporting message in the terminal.

A 2-D Modeling experiment using SConstruct

1. Invoke RSF module to create a experiment environment.

   ```
   from rsf.proj import *

   End()
   ```

   Almost every SConstruct for numerical test has to start with `from rsf.proj import *` and ends with `End()`.

2. In between, add several lines to create a very simple velocity model including 3 layers using `Flow(target,source,command)`. The command here is `sfmath`. One may type the command name '`sfmath`' to look up the manual in the terminal.

3. Performing 2-D forward simulation with 1 point source (a Ricker wavelet) by specifying the parameters for the command `sfmodeling2d`. Again, one may type the command name to look up the manual in the terminal if you cannot keep everything in mind.

4. Polish your resulting figures. Help yourself by self-documentation of the command, type the command name `sfgrey` or `man sfgrey` [†] in the terminal. For example, try color style: `color=g bartype=h`

The final SConstruct looks like

[†]Keep in mind the initial '`sf`' has to be included when looking for the functionality of the command, although it can be omitted in SConstruct.

```
from rsf.proj import *

Flow('vel0',None,
     '''
     math output=1.6 n1=50 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')

Flow('vel1',None,
     '''
     math output=1.8 n1=50 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')
Flow('vel2',None,
     '''
     math output=2.0 n1=100 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')
Flow('vel',['vel0','vel1','vel2'], 'cat axis=1 ${SOURCES[1:3]}')

Result('vel',
       '''
       grey title="velocity model: 3 layers"
       color=j scalebar=y bartype=v
       ''')

Flow('shots','vel',
     '''
     modeling2d nt=1100 dt=0.001 ns=10 ng=200
     sxbeg=5 szbeg=2 jsx=20 jsz=0
     gxbeg=0 gzbeg=3 jgx=1 jgz=0
     ''')

Result('shots',
     '''
       byte allpos=n gainpanel=all |
       grey3 flat=n frame1=300 frame2=100 frame3=5
       label1=Time unit1=s
       label2="Receiver no." label3="Shot no."
       title="Shot records" point1=0.8 point2=0.8
     ''')

Plot('shots','grey title=Shots',view=1)

```

```
45 #use sfwindow to select 5-th shot and display it using sfgrey
46
47
48 End()
```

We obtain the velocity model in Figure 1 and the shot gathers in Figure 2. To have a look at the movie by looping over each shot gather, run `scons`.

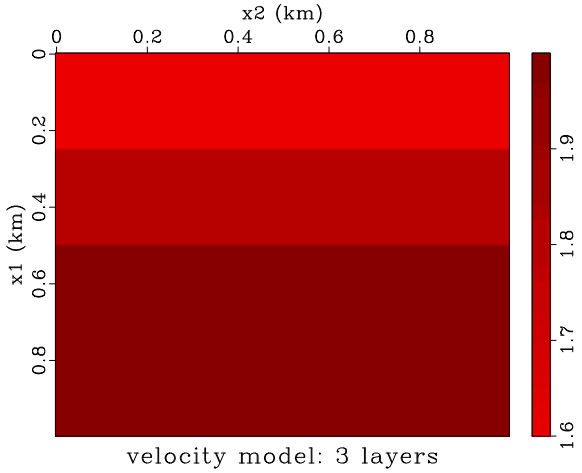


Figure 1: A 3-layer velocity model modeling2fwi/modeling2d/ vel

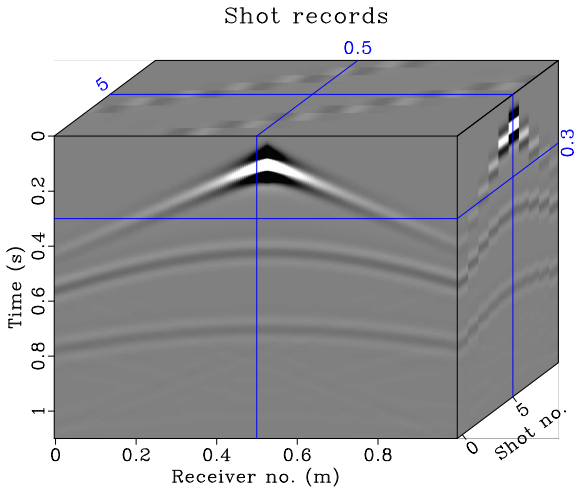


Figure 2: Shot gather modeling2fwi/modeling2d/ shots

Further exercises

How to:

1. higher accuracy in space → higher order FD stencil → Fourier pseudospectral method (Carcione, 2010), see the code /RSFSRC/user/pyang/Mps2d.c?

2. implement sponge/Gaussian taper boundary condition (Cerjan et al., 1985), PML (Komatitsch and Martin, 2007)? (Yang, 2014)

3. increase temporal discretization accuracy: leap-frog → Runge-Kutta scheme?

4. locate your source and receiver position at arbitrary position: Kaiser windowed sinc interpolation (Hicks, 2002)

FULL WAVEFORM INVERSION

FWI is a nonlinear iterative minimization process by matching the waveform between the synthetic data and the observed seismograms (Tarantola, 1984; Virieux and Operto, 2009). In least-squares sense, the misfit functional of FWI reads

$$C(m) = \frac{1}{2}\|R_r p - d\|^2, \tag{5}$$

where m is the model parameter (i.e. the velocity) in model space; R_r is a restriction operator mapping the wavefield onto the receiver locations; $d := d(x_r, t)$ is the observed seismogram at receiver location x_r while $p := p(x, t)$ is the synthetic wavefield whose adjoint wavefield $\bar{p}$ is given by

$$(\frac{1}{v^2}\partial_t^2 - \nabla)\bar{p} = -\frac{\partial C}{\partial p} = -R_r^\dagger(R_r p - d) \tag{6}$$

which indicates that the adjoint wave equation is exactly the same as the forward wave equation except that the adjoint source is data residual backprojected into the wavefield. In each iteration the model has to be updated following a Newton descent direction Δm^k

$$m^{k+1} = m^k + \gamma_k \Delta m^k, \tag{7}$$

with a stepsize γ_k. Away from the sources ($f = 0$), the gradient can be computed by

$$\nabla C = -\frac{2}{v^3}\int_T \mathrm{d}t\bar{p}\partial_t^2 p = -\frac{2}{v}\int_T \mathrm{d}t\bar{p}\nabla^2 p \tag{8}$$

Reconstruct your source/incident wavefield backwards in time

As one can see from above, the computation of FWI gradient requires simultaneously accessing the source/incident wavefield and the adjoint wavefield at each time step. To achieve this goal, we may store the absorbing boundary at each time step when doing forward simulation, and then reconstruct the incident wavefield backwards in

time using the stored boundary. According to equation (2), the reconstruction is easy

$$p^{k-1} = 2p^k - p^{k+1} + \Delta t^2 v^2 \nabla^2 p^k \tag{9}$$

Therefore, we may employ the same subroutine by switching the role of wavefield p^{k+1} and p^{k-1}. The elements in the boundary does not follow the above equation but we can store it in forward simulation and re-inject them for backward reconstruction.

1. Redo forward modeling using the same model while storing the boundary at each time step

```
for(it=0; it<nt; it++){
  add_source(p1, &wlt[it], &sxz[is], 1, nz, true);
  step_forward(p0, p1, p2, vv, dtz, dtx, nz, nx);
  ptr=p0; p0=p1; p1=p2; p2=ptr;
  rw_bndr(&bndr[it*(2*nz+nx)], p0, nz, nx, true);
  record_seis(&dobs[it*ng], gxz, p0, ng, nz);

  if(it==kt){
    sf_floatwrite(p0[0],nz*nx, check1);
  }
}
```

2. reverse propagate the incident wavefield backwards from final snapshot and stored boundary: at each time step, inject the corresponding boundary

```
ptr=p0; p0=p1; p1=ptr;
for(it=nt-1; it>-1; it--){
  rw_bndr(&bndr[it*(2*nz+nx)], p1, nz, nx, false);

  if(it==kt){
    sf_floatwrite(p1[0],nz*nx, check2);
  }
  step_backward(p0, p1, p2, vv, dtz, dtx, nz, nx);
  add_source(p1, &wlt[it], &sxz[is], 1, nz, false);
  ptr=p0; p0=p1; p1=p2; p2=ptr;
}
```

3. check whether you perfectly reconstruct your incident wavefield at any time step `kt`, as shown in Figure 3.

The final SConstruct looks like

```
from rsf.proj import *

Flow('vel0',None,
```

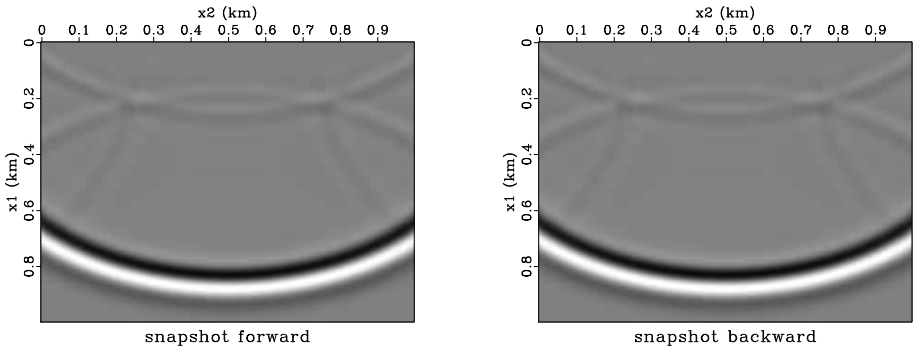


Figure 3: The backward reconstructed wavefield is the same as the incident wavefield modeling2fwi/fbrec2d/ check

```
        ''')

Math(output='1.6 n1=50 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')

Flow('vel1',None,
     ''')
     math output=1.8 n1=50 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')
Flow('vel2',None,
     ''')
     math output=2.0 n1=100 n2=200 d1=0.005 d2=0.005
     label1=x1 unit1=km label2=x2 unit2=km
     ''')
Flow('vel',['vel0','vel1','vel2'], 'cat axis=1 ${SOURCES[1:3]}')


Flow('shot check1 check2','vel',
     ''')
     sffbrec2d check1=${TARGETS[1]} check2=${TARGETS[2]}
     csdgather=n fm=15 dt=0.001 ns=1 ng=200 nt=1100 ng=200 kt=550
     sxbeg=100 szbeg=2 jsx=37 jsz=0
     gxbeg=0 gzbeg=1 jgx=1 jgz=0
     ''')

Result('shot','grey gainpanel=all title=shot')


Flow('diff','check1 check2','sfadd ${SOURCES[1]} scale=1,-1')
```

```
34
35
36 Plot('check1','grey title="snapshot forward" scalebar=y')
37 Plot('check2','grey title="snapshot backward" scalebar=y')
38 Plot('diff','grey title="difference" scalebar=y')
39
40 Result('check','check1 check2 diff','SideBySideIso')
41
42 End()
```

Do a synthetic FWI test using Marmousi model

It is convenient to perform adjoint simulation when reconstructing the incident wavefield backwards. The computation of FWI gradient can be done on the fly by adding several lines:

```
1  memset(sp0[0], 0, nz*nx*sizeof(float));
2  memset(sp1[0], 0, nz*nx*sizeof(float));
3  for(it=0; it<nt; it++)
4  {
5    add_source(sp1, &wlt[it], &sxz[is], 1, nz, true);
6    step_forward(sp0, sp1, sp2, vv, dtz, dtx, nz, nx);
7    ptr=sp0; sp0=sp1; sp1=sp2; sp2=ptr;
8    rw_bndr(&bndr[it*(2*nz+nx)], sp0, nz, nx, true);
9
10   record_seis(dcal, gxz, sp0, ng, nz);
11   cal_residuals(dcal, &dobs[it*ng], &derr[is*ng*nt+it*ng], ng);
12 }
13
14 ptr=sp0; sp0=sp1; sp1=ptr;
15 memset(gp0[0], 0, nz*nx*sizeof(float));
16 memset(gp1[0], 0, nz*nx*sizeof(float));
17 for(it=nt-1; it>-1; it--)
18 {
19   rw_bndr(&bndr[it*(2*nz+nx)], sp1, nz, nx, false);
20   step_backward(illum, lap, sp0, sp1, sp2, vv, dtz, dtx, nz, nx);
21   add_source(sp1, &wlt[it], &sxz[is], 1, nz, false);
22
23   add_source(gp1, &derr[is*ng*nt+it*ng], gxz, ng, nz, true);
24   step_forward(gp0, gp1, gp2, vv, dtz, dtx, nz, nx);
25
26   cal_gradient(g1, lap, gp1, nz, nx);
27   ptr=sp0; sp0=sp1; sp1=sp2; sp2=ptr;
28   ptr=gp0; gp0=gp1; gp1=gp2; gp2=ptr;
```

```
}
```

Of course, to apply the steepest descent method for minimization of the misfit function for waveform inversion, a step length has to be determined at each iteration. You may use the estimation proposed by Pica et al. (1990,in the appendix) in your FWI code as `RSFSRC/user/pyang/Mfwi2d.c`.

The workflow for synthetic FWI test based on Marmousi model follows the several steps:

1. Obtain the Marmousi velocity model, which can be downloaded by `Fetch('marmvel.hh','marm')` in SConstruct, as shown in the top panel of Figure 4.
2. We may start to generate the observed seismograms/shots using resampled Marmousi, as shown in Figure 5.
3. By smoothing the true model using triangular window many times, an initial model plotted in the bottom panel of Figure 4 is generated to do the FWI test.
4. Using the observed seismograms/shots from true model, we start the inversion with the rough initial model.
5. You may appreciate your the inverted velocity during the iterations in Figure 6, as well as the variations of the misfit function in Figure 7.

A complete SConstruct for the above workflow appears in the following:

```
from rsf.proj import *

# marmvel.hh contains Marmousi model which can
# be downloaded from the server using Fetch.
Fetch('marmvel.hh','marm')

Flow('vel','marmvel.hh',
        '''
        dd form=native | window j1=8 j2=8 | sfsmooth rect1=3 rect2=3|
        put label1=Depth  unit1=m label2=Lateral unit2=m
        ''')
Plot('vel',
        '''
        grey color=j mean=y title="Marmousi model"
        scalebar=y bartype=v barlabel="V"
        barunit="m/s" screenratio=0.45 color=j labelsz=10 titlesz=12
        ''')

Flow('shots','vel',
        '''
```

```
        sfmodeling2d csdgather=n fm=4 amp=1 dt=0.0015 ns=7 ng=288 nt=2800
        sxbeg=4 szbeg=2 jsx=45 jsz=0 gxbeg=0 gzbeg=3 jgx=1 jgz=0
        ''')
Plot('shots','grey color=g title=shot label2= unit2=',view=1)


Plot('shot1','shots',
     'window n3=1 f3=0| grey title=shot1 label2=Lateral unit2=m')
Plot('shot3','shots',
     'window n3=1 f3=2| grey title=shot3 label2=Lateral unit2=m')
Plot('shot5','shots',
     'window n3=1 f3=4| grey title=shot5 label2=Lateral unit2=m')
Plot('shot7','shots',
     'window n3=1 f3=6| grey title=shot7 label2=Lateral unit2=m')
Result('shotsnap','shot1 shot3 shot5 shot7',
       'SideBySideAniso',vppen='txscale=2.')

# smoothed velocity model
Flow('smvel','vel','smooth repeat=6  rect1=8 rect2=10')
Plot('smvel',
     '''
     grey title="Initial model" wantitle=y allpos=y color=j
     pclip=100 scalebar=y bartype=v barlabel="V" barunit="m/s"
         screenratio=0.45 color=j labelsz=10 titlesz=12
     ''' )

Result('marm','vel smvel','TwoRows')

# use the over-smoothed model as initial model for FWI
Flow('vsnaps grads objs illums','smvel shots',
        '''
        sffwi2d shots=${SOURCES[1]}
        grads=${TARGETS[1]} objs=${TARGETS[2]}
        illums=${TARGETS[3]} niter=10 precon=y rbell=1
        ''')
Result('vsnaps',
        '''
        grey title="Updated velocity" allpos=y color=j pclip=100
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')
Plot('vsnap1','vsnaps',
        '''
        window n3=1|grey title="Updated velocity, iter=1"
        allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
```

```
        ''')
Plot('vsnap2','vsnaps',
        '''
        window n3=1 f3=1|grey title="Updated velocity, iter=2"
        allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')
Plot('vsnap4','vsnaps',
        '''
        window n3=1 f3=3|grey title="Updated velocity, iter=4"
        allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')

Plot('vsnap6','vsnaps',
        '''
        window n3=1 f3=5|grey title="Updated velocity, iter=6"
         allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')
Plot('vsnap8','vsnaps',
        '''
        window n3=1 f3=7|grey title="Updated velocity, iter=8"
        allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')
Plot('vsnap10','vsnaps',
        '''
        window n3=1 f3=9|grey title="Updated velocity, iter=10"
        allpos=y color=j pclip=100 labelsz=10 titlesz=12
        scalebar=y bartype=v barlabel="V" barunit="m/s"
        ''')

Result('vsnap','vsnap1 vsnap2 vsnap4 vsnap6 vsnap8 vsnap10',
       'TwoRows')

Result('objs',
        '''
        sfput n2=1 label1=Iteration unit1= unit2= label2= |
        graph title="Misfit function" dash=0 plotfat=5
        grid=y yreverse=n
        ''')

End()
```

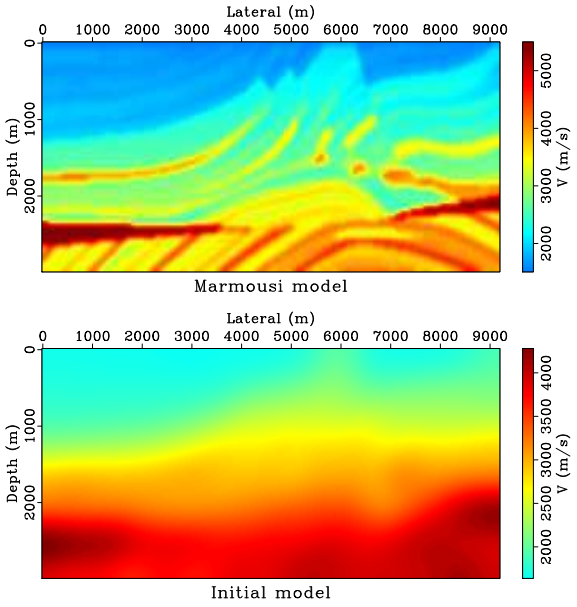


Figure 4: Top: True Marmousi model; bottom: Initial model for FWI modeling2fwi/marmtest/ marm

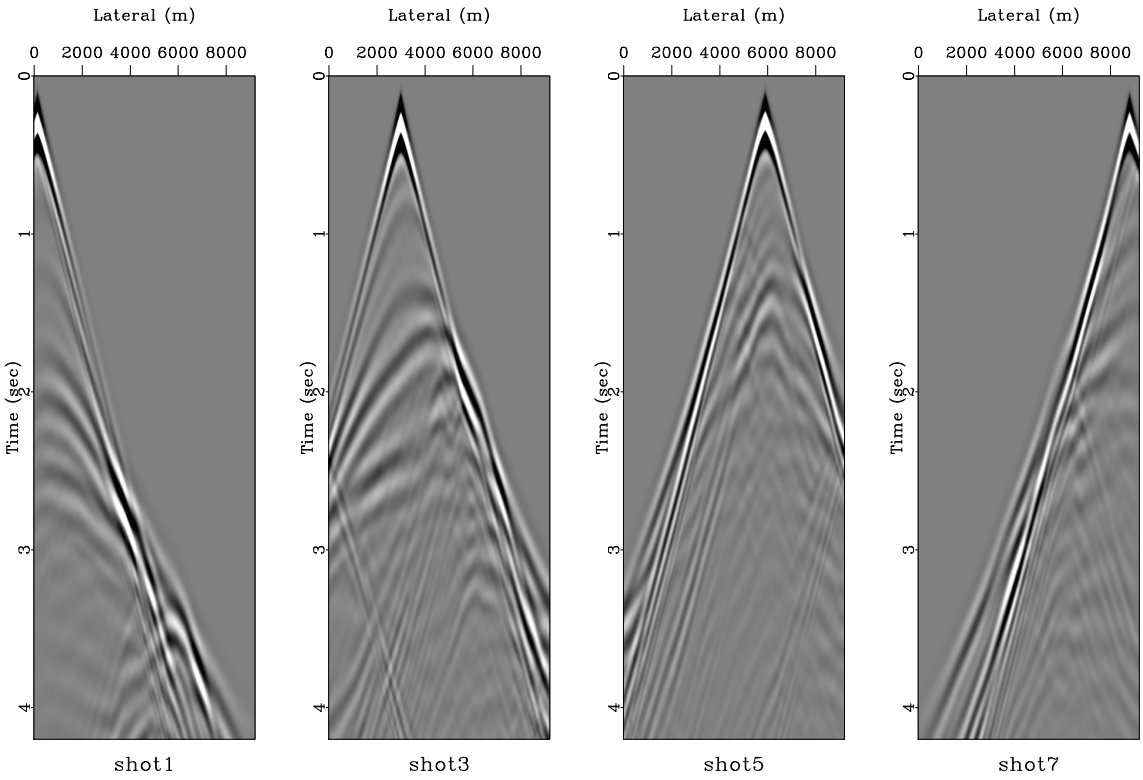


Figure 5: Shots from true Marmousi model modeling2fwi/marmtest/ shotsnap

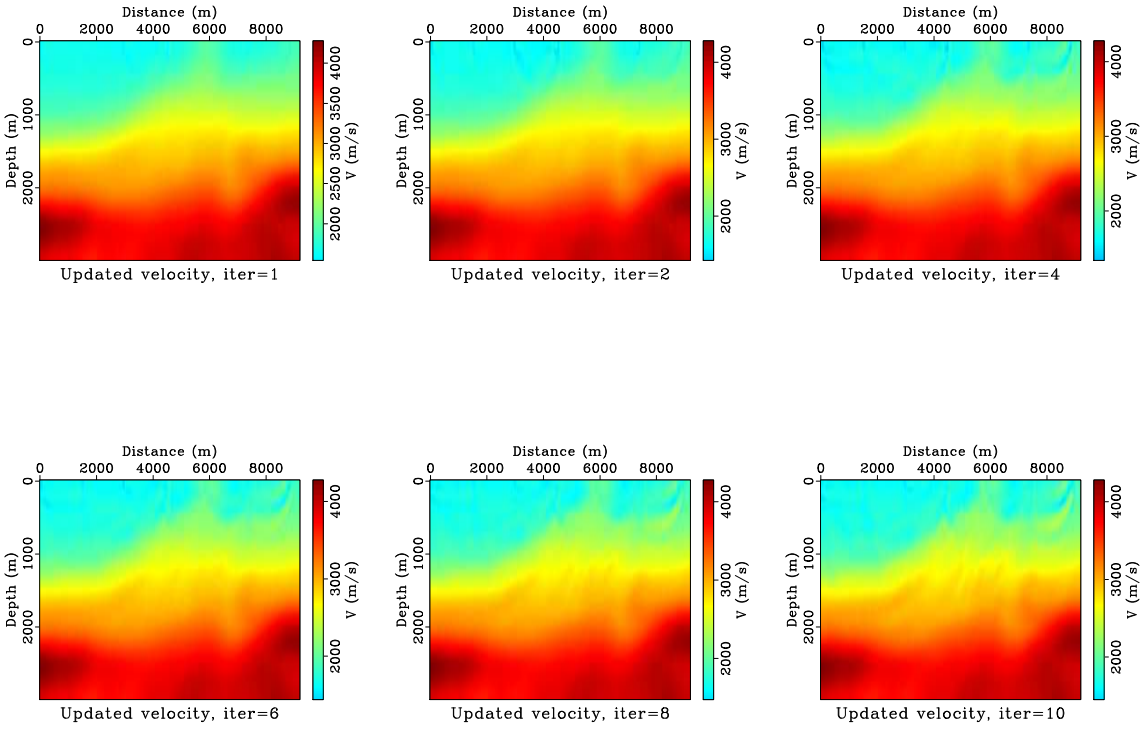


Figure 6: Inverted velocity during iterations modeling2fwi/marmtest/ vsnap

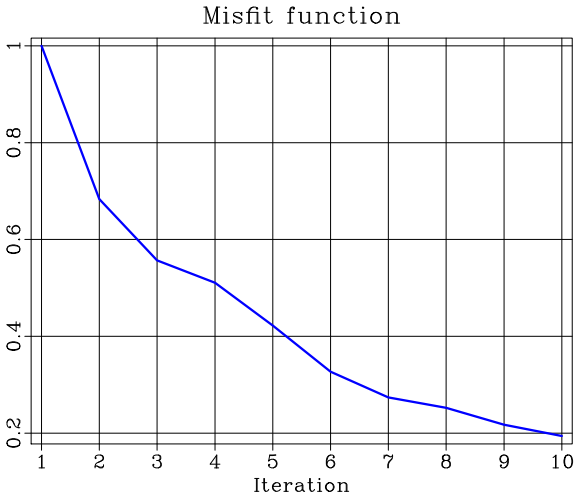


Figure 7: The misfit function decreases during iterations modeling2fwi/marmtest/ objs

Your exercise

- It works so slow! How to speeed up? → CUDA (Yang et al., 2015,Mgpufwi.cu) + MPI high performance computing (Jeff)?

- How to improve the poor resulting velocity model in Figure 6? A better initial model by less smoothing? Increase the number of iterations? Estimate a good step length to satisfy the Wolf condition? Estimate Hessian → Truncated newton (Métivier et al., 2014)+ Source encoding+ Good preconditioning?

- Derive the adjoint equation for first order wave equation system

$$\begin{cases} \partial_t p = \kappa \nabla \cdot \mathbf{v} + f_p \\ \rho \partial_t \mathbf{v} = \nabla p \end{cases} \tag{10}$$

  where $\kappa = \rho v^2$.

- Write a forward simulation code based on sponge boundary condition using the above system

- Derive your gradient expressions for velocity and density

- code your multiparameter FWI for inverting v and ρ

FURTHER THINKING

- The storage complexity using regular grid FD and staggered grid FD stencil?

- How to reduce the storage requirement for a 3-D volume? CFL → Nyquist: decimation+ interpolation (Yang et al., 2016c,d)

- How to derive the adjoint state equation? → Lagrange multiplier+cost function (Plessix, 2006)? What is the adjoint state equation for 1st order acoustic wave equation, viscoacoustic system, viscoelastic system (Yang et al., 2016a)?

- Is it possible to do reverse propagation in attenuating medium? How to handle instability? Binomial checkpointing→ CARFS (checkpointing-assisted reverse-forward simulation) (Yang et al., 2016b)?

- What if FWI using other norms/misfit function C? Only change the adjoint source $\frac{\partial C}{\partial p}$?

CONCLUSION

1. No answer sheet for your exercises!
2. Too many open questions in FWI: good initial model? Misfit function immune to cycle-skipping issue? Better preconditioning? Inverting attenuating?
3. FWI is a research field waiting for your addition!

REFERENCES

Carcione, J. M. (2010). A generalization of the fourier pseudospectral method. *Geophysics*, 75(6):A53–A56.

Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M. (1985). A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50(4):2117–2131.

Clayton, R. and Engquist, B. (1977). Absorbing boundary conditions for acoustic and elastic wave equations. *Bulletin of the Seismological Society of America*, 67:1529–1540.

Hicks, G. J. (2002). Arbitrary source and receiver positioning in finite-difference schemes using Kaiser windowed sinc functions. *Geophysics*, 67:156–166.

Komatitsch, D. and Martin, R. (2007). An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5):SM155–SM167.

Métivier, L., Bretaudeau, F., Brossier, R., Operto, S., and Virieux, J. (2014). Full waveform inversion and the truncated Newton method: quantitative imaging of complex subsurface structures. *Geophysical Prospecting*, 62:1353–1375.

Pica, A., Diet, J. P., and Tarantola, A. (1990). Nonlinear inversion of seismic reflection data in laterally invariant medium. *Geophysics*, 55(3):284–292.

Plessix, R. E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503.

Tarantola, A. (1984). Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49(8):1259–1266.

Virieux, J. and Operto, S. (2009). An overview of full waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26.

Yang, P. (2014). A numerical tour of wave propagation. Technical report, Xi'an Jiaotong University.

Yang, P., Brossier, R., Métivier, L., and Virieux, J. (2016a). A systematic formulation of 3D multiparameter full waveform inversion in viscoelastic medium. *submitted to Geophysical Journal International*.

Yang, P., Brossier, R., Métivier, L., and Virieux, J. (2016b). Wavefield reconstruction in attenuating media: A checkpointing-assisted reverse-forward simulation method. *submitted to Geophysics*.

Yang, P., Brossier, R., and Virieux, J. (2016c). Downsampling plus interpolation for wavefield reconstruction by reverse propagation. In *78th EAGE Conference & Exhibition Expanded Abstracts*, number SBT5 08.

Yang, P., Brossier, R., and Virieux, J. (2016d). Wavefield reconstruction from significantly decimated boundaries. *Geophysics, Accepted*.

Yang, P., Gao, J., and Wang, B. (2015). A graphics processing unit implementation of time-domain full-waveform inversion. *Geophysics*, 80(3):F31–F39.