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

[a]Published in Geophysics, 80, no. 3, F31-F39, (2015)

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

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

parameters by cross correlating the back propagated residual wavefield with the corresponding forward propagated source wavefield. The forward modeling itself demands 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 (A-7). 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 (5), 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.,

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

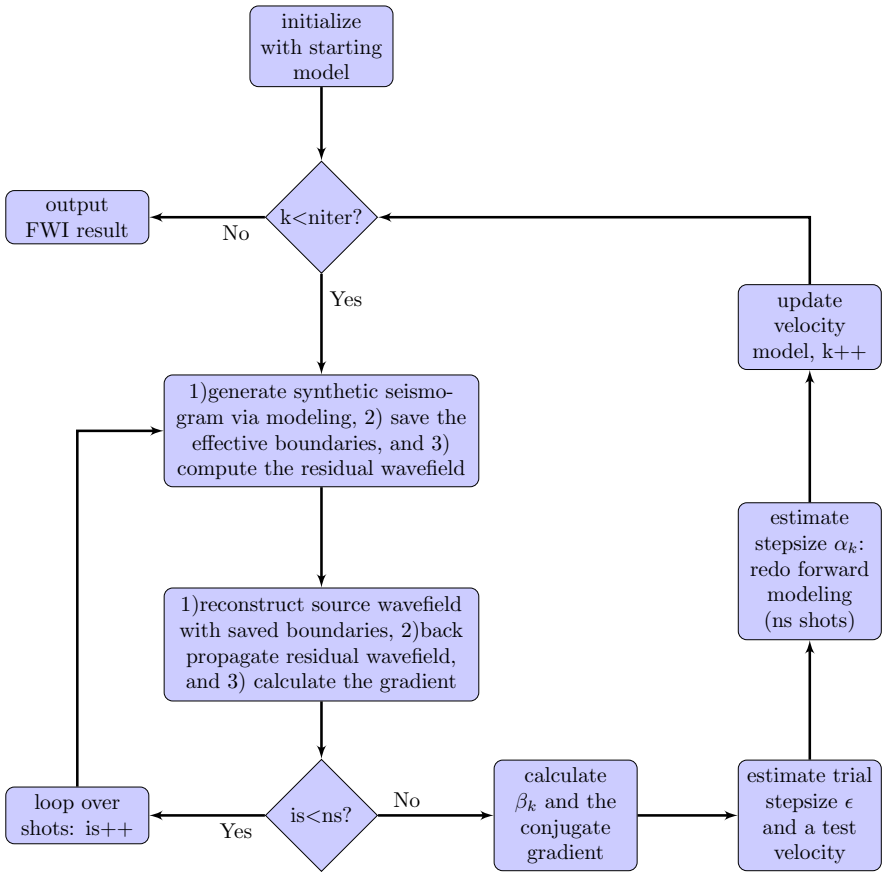

Figure 1: Backward reconstruction can be realized using the saved boundaries. Note that no absorbing boundary condition is applied on the top boundary of the model in the forward modeling.

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.

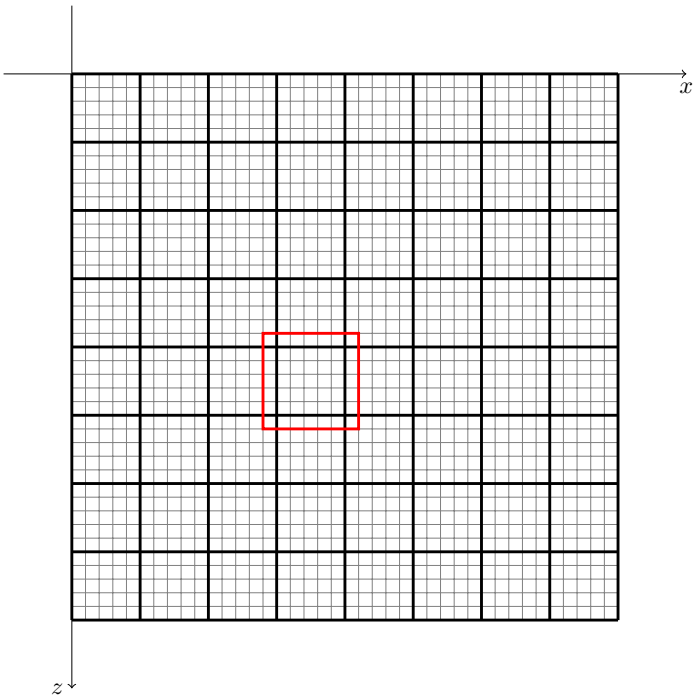

Figure 2: 2D blocks in GPU memory. The marked window indicates that the shared memory in every block needs to be extended on each side with halo ghost points storing the grid value from other blocks.

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.

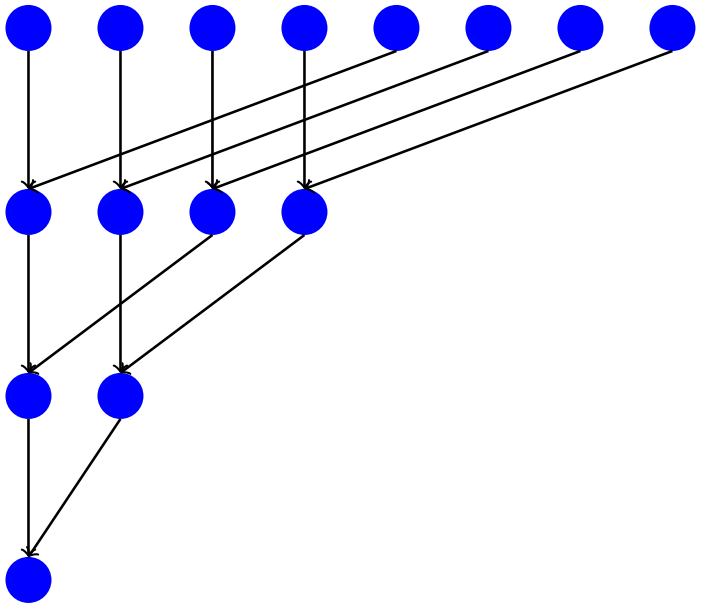

Figure 3: Parallel reduction on GPU block. It reduces a serial computational complexity $O(N)$ to be $O(\log_2(N))$ steps: in each step many threads perform computation simultaneously, leading to low arithmetic intensity.

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 4. 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 4. As can be seen from the figure, the backward reconstruction starts from the boundaries (bottom left) and gradually recovers the interior wavefield (bottom right).

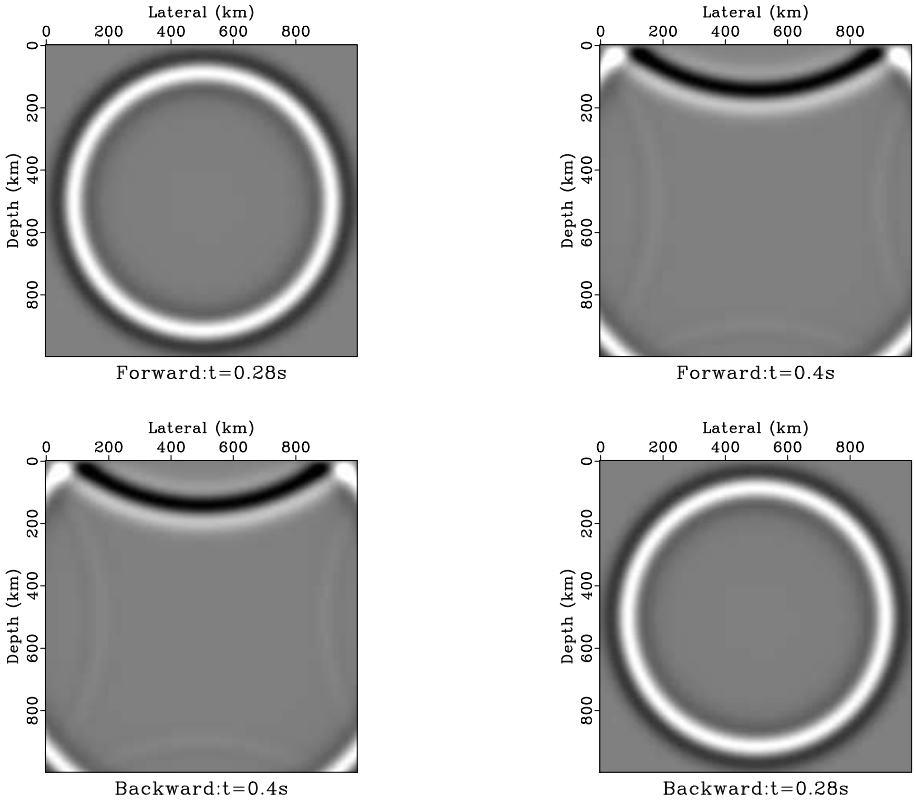

Figure 4: Backward reconstruction can be realized using the saved boundaries. Note that no absorbing boundary condition is applied on the top boundary of the model in the forward modeling.

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.

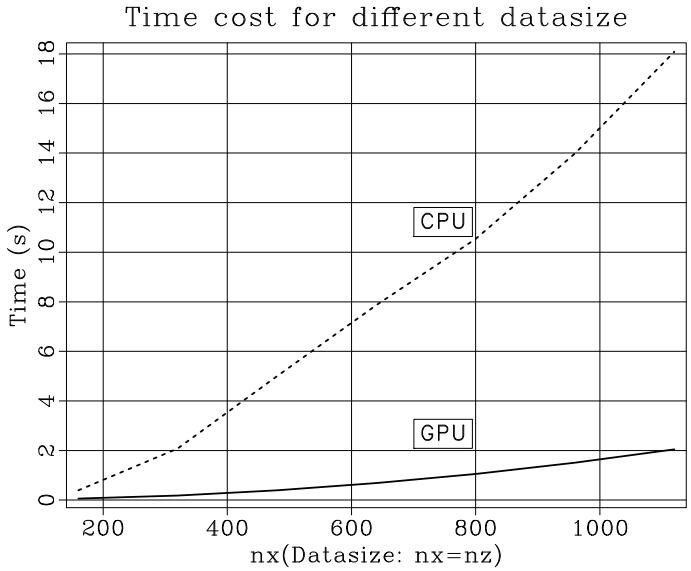

Figure 5: Comparison of the time cost for CPU- vs. GPU implementation under different model sizes with one shot, 1000 time steps of forward modeling.

Marmousi model

We use the Marmousi model for the benchmark test, as shown in the top panel of Figure 6. 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 7. We use a starting model (bottom panel of Figure 6) 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 8. Figure 9 describes the decreasing misfit function in iterations. As can be seen from the Figures 8 and 9, 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.

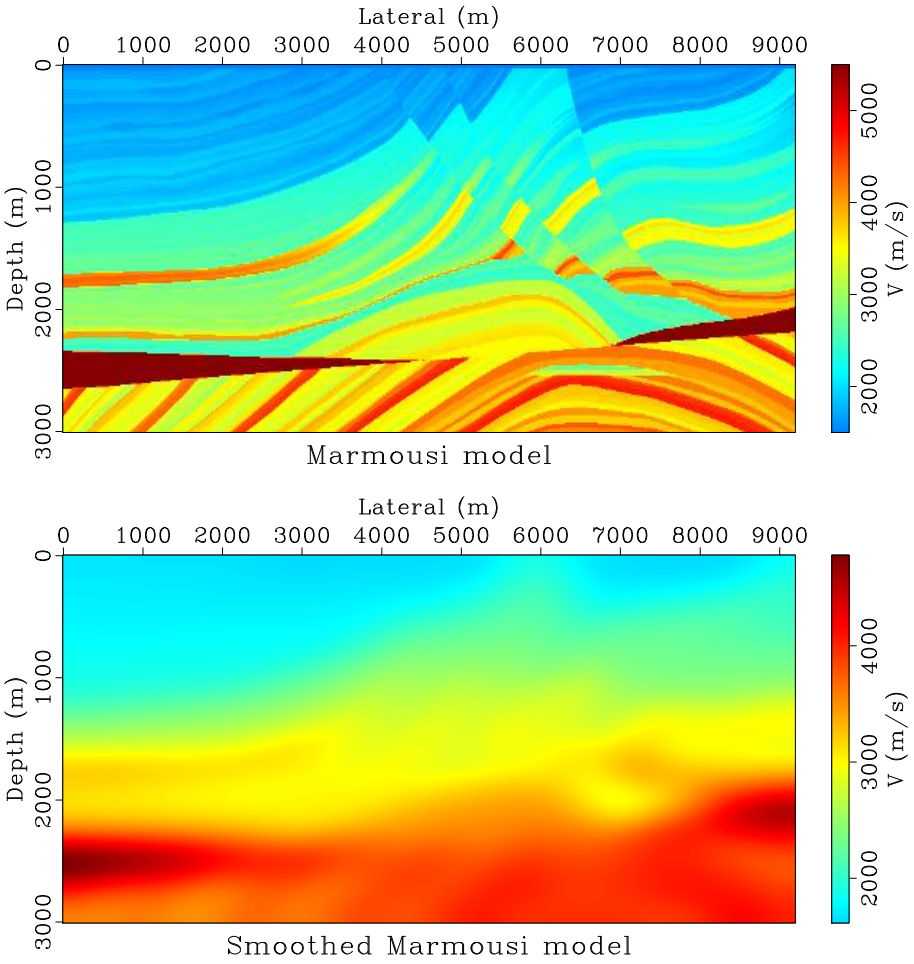

Figure 6: Top: The original Marmousi is downsampled by a factor of 3 along depth and lateral direction. The shots are generated according to the subsampled Marmousi model. Bottom: The starting model of FWI for Marmousi model, which is obtained by smoothing the original model 20 times with a 5x5 window.

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 boundary was utilized to maintain the efficiency of GPU computation. A hybrid nonlinear conjugate gradient method combined with parallel reduction technique was

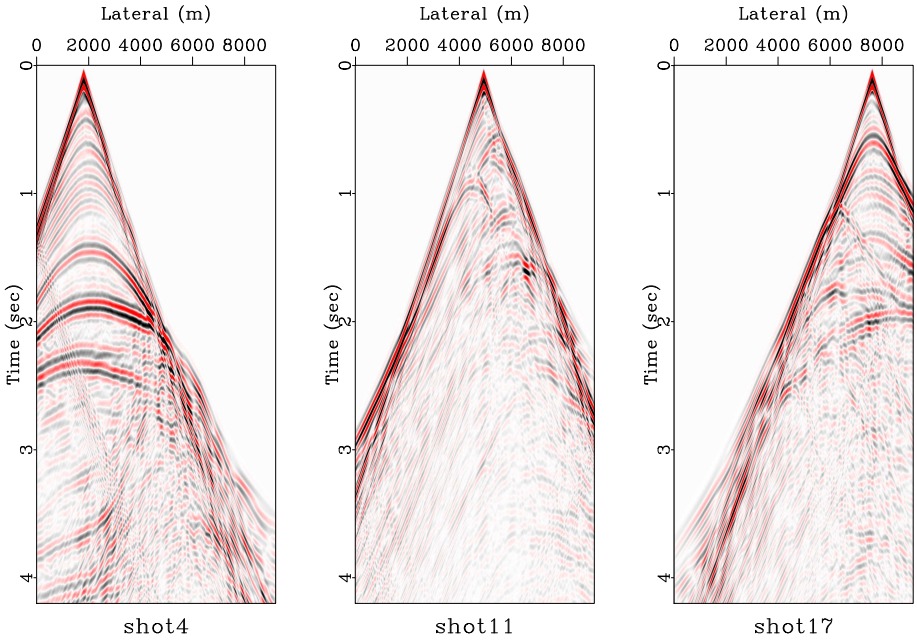

Figure 7: 21 shots were deployed in the FWI. Here, shots 4, 11 and 17 are shown from left to right.

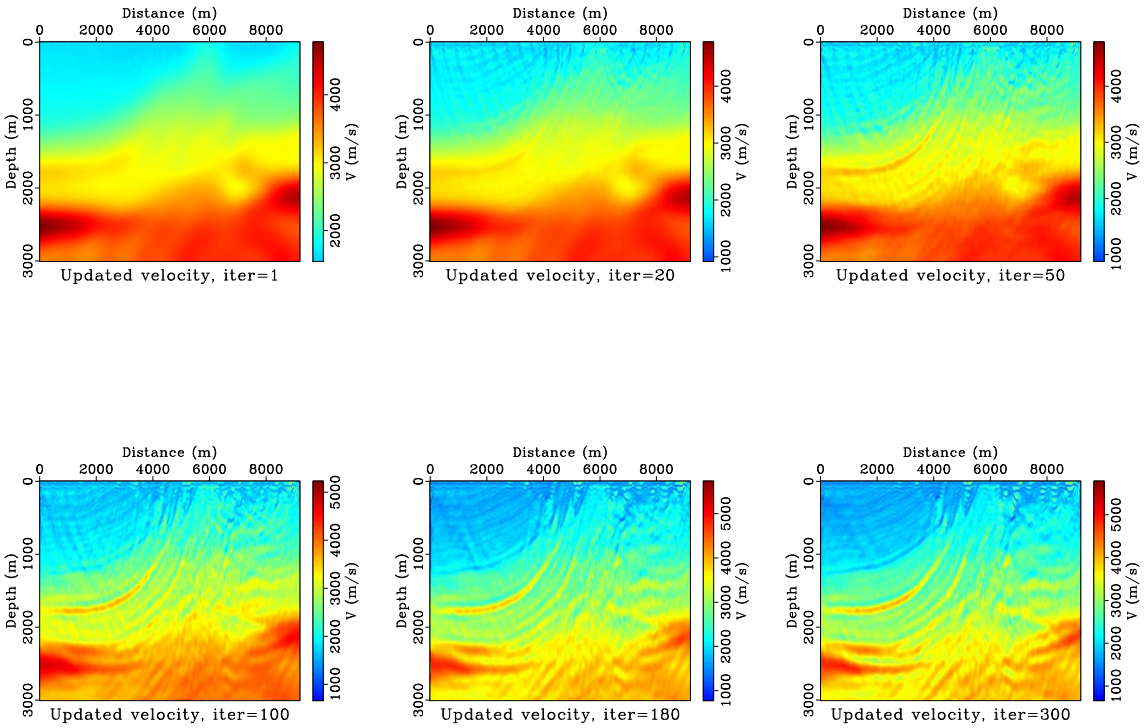

Figure 8: The updated velocity model at iterations 1, 20, 50, 100, 180 and 300.

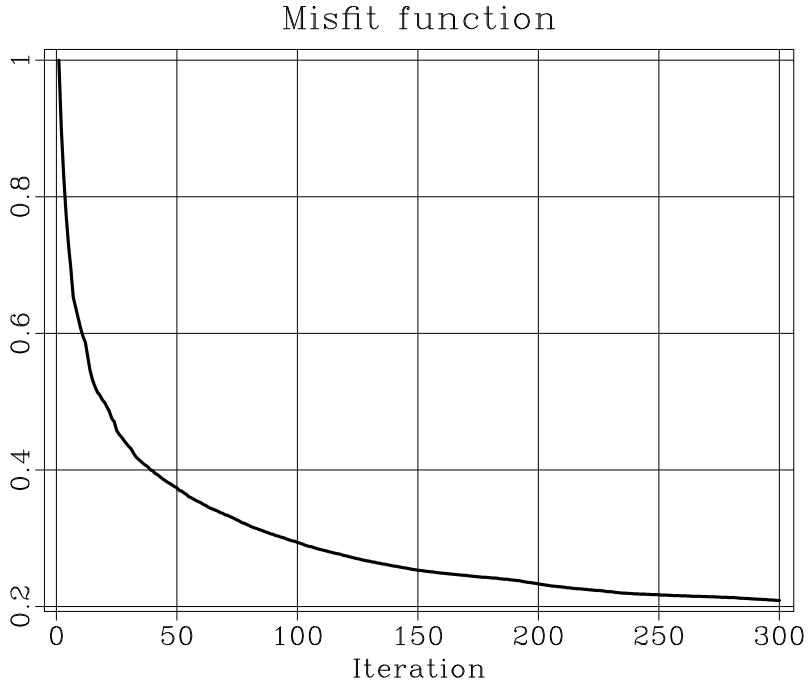

Figure 9: The misfit function decreases with iteration.

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 (A-4) 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 (A-4) 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 (B-3) subtracts equation (1), 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 (B-5) 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 (2),

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

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 (C-1) 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.