Preconditioned least-squares reverse-time migration using random phase encoding

Ali Almomin

ABSTRACT

Least-squares reverse-time migration (LSRTM) provides very accurate images of the subsurface. However, the computational cost of this technique is extremely high. One way to reduce that cost is to encode the sources using a random phase function and create a "super source". This encoding method introduces crosstalk artifacts that require averaging several realizations of the random encodings to suppress. I compare the convergence rates of the conventional and phase-encoded LSRTM for a fixed-spread geometry and show that the performance gain for the phase-encoded LSRTM far exceeds the loss due to the additional realizations. I also reduce the inversion cost by using the diagonal of the Hessian matrix as a preconditioner to the gradient. I also compare the convergence rates of different encoding methods used to estimate the true Hessian matrix. Then, I introduce a new source-based Hessian approximation and compare it to the other methods of approximating the Hessian matrix. Finally, I show the effect of each preconditioner on the LSRTM inversion. Results from the Marmousi synthetic model show that, for the same cost, preconditioning with the source-based Hessian gives the most accurate results.

INTRODUCTION

Reverse-time migration (RTM) uses the full wave equation to image the subsurface with high accuracy. However, RTM images suffer from several operator artifacts such as low-frequency noise, decreased resolution due to squaring the wavelet, and imbalanced amplitudes. These artifacts appear because, by migrating the data, we apply only the adjoint of the linear modeling operator, as opposed to its inverse. The imaging operator can be inverted in several ways, including iterative least-squares inversion. Although inversion can remove these artifacts, it is not widely used because the computational cost of applying the forward and adjoint operators in each iteration is extremely high.

Several methods have been proposed to reduce the computational cost of LSRTM. One of these methods is to reduce the data size by encoding the sources to create a super source (Morton and Ober, 1998; Jing et al., 2000; Romero et al., 2000; Sun et al., 2002). This technique has also been applied to full waveform inversion (Krebs et al., 2009; Gao et al., 2010; Boonyasiriwat and Schuster, 2010; Ben-hadj ali et al.,

2011). The cost of applying the forward or adjoint of the modeling operator becomes independent of the number of sources, which greatly reduces the computational cost of the inversion. On the other hand, combining the sources causes crosstalk artifacts in the estimated image. These artifacts can be suppressed by changing the encoding function over iterations. Romero et al. (2000) and Krebs et al. (2009) showed that one-sample random phase encoding gives the best convergence rate.

In this paper, I compare the convergence rate of conventional LSRTM to phase-encoded LSRTM to test whether the computational reduction justifies the additional realizations of the encoding function. I first measure the norm of the model error of the two models. Then, I measure the norm of the model error after processing the estimated model at each iteration. The processing steps, which are a low-cut filter and an automatic gain control (AGC), are used to reduce the low-frequency noise and amplitude imbalance in order to measure the error in the image resolution.

The convergence rate of LSRTM inversion can be accelerated by preconditioning the gradient with the diagonal of the Hessian matrix. The Hessian matrix can be estimated by applying the encoding function either to the receiver side only or to both the receiver and source sides (Tang, 2009). A cheaper approximation to the Hessian matrix is the source intensity function, which ignores the receiver side of the Hessian matrix (Tang and Lee, 2010). The source intensity function can also be encoded along the source axis. I compare the convergence rates of these methods and also introduce a new approximation to the Hessian matrix that is based on the blended source-wavefield only. Finally, I test each of these preconditioners in the LSRTM and compare the results as a function of their cost.

METHOD

Least-squares RTM

The two-way wave equation is linearized over the slowness squared as follows:

$$s^2(\mathbf{x}) = b(\mathbf{x}) + m(\mathbf{x}), \tag{1}$$

where s is the slowness, b is the background model, which is a smooth version of the slowness squared, m is the model, and $\mathbf{x}$ is the model coordinate. Then, the Green's functions that satisfy the acoustic wave equation using the background model are defined as follows:

$$\left(\nabla^2 + \omega^2 b(\mathbf{x})\right) G_0(\mathbf{x}, \mathbf{x}_s, \omega) = -\delta(\mathbf{x} - \mathbf{x}_s), \tag{2}$$

$$\left(\nabla^2 + \omega^2 b(\mathbf{x})\right) G_0(\mathbf{x}, \mathbf{x}_r, \omega) = -\delta(\mathbf{x} - \mathbf{x}_r), \tag{3}$$

where G_0 is the Green's function, $\mathbf{x}_s$ and $\mathbf{x}_r$ are the source and receiver coordinates, and ω is frequency. The forward modeling operator is defined using the Green's functions as follows:

$$d(\mathbf{x}_r, \mathbf{x}_s, \omega) = -\omega^2 f(\omega) \sum_{\mathbf{x}} G_0(\mathbf{x}, \mathbf{x}_s, \omega) G_0(\mathbf{x}, \mathbf{x}_r, \omega) m(\mathbf{x}), \tag{4}$$

where d is the surface data and f is the source function. The forward modeling operator $\mathbf{F}$ can be written in matrix form as follows:

$$\mathbf{d} = \mathbf{F}\mathbf{m}. \tag{5}$$

We now define the objective function J as:

$$J(\mathbf{m}) = \|\mathbf{F}\mathbf{m} - \mathbf{d}_{\text{obs}}\|_2^2, \tag{6}$$

where $\mathbf{d}_{\text{obs}}$ is the observed surface data. The quadratic objective function J can be minimized iteratively using the following scheme (Claerbout and Fomel, 2011):

$$\begin{aligned}
&\mathbf{r} \longleftarrow \mathbf{F}\mathbf{m} - \mathbf{d}_{\text{obs}} \\
&\text{iterate } \{ \\
&\qquad \Delta\mathbf{m} \longleftarrow \mathbf{F}^*\mathbf{r} \\
&\qquad \Delta\mathbf{r} \longleftarrow \mathbf{F}\Delta\mathbf{m} \\
&\qquad (\mathbf{m}) \longleftarrow \text{stepper}(\mathbf{m}, \mathbf{r}, \Delta\mathbf{m}, \Delta\mathbf{r}) \\
&\qquad \}
\end{aligned}$$

The * indicates the adjoint operator. The cost of each iteration equals the cost of the forward and adjoint operators. The stepper function is either steepest-descent or conjugate gradient.

There are several encoding functions that can be used in LSRTM (Perrone and Sava, 2009; Godwin and Sava, 2011). However, a single-sample random phase function gives the best convergence results (Romero et al., 2000; Krebs et al., 2009). This encoding function results in crosstalk artifacts in the estimated models. These artifacts are reduced by averaging several realizations of the encoding function. The source-side encoding function α is defined as follows (Tang, 2009):

$$\alpha(\mathbf{x}_s, p_s) = \frac{1}{\sqrt{N_{\text{realize}}}}\gamma(\mathbf{x}_s, p_s), \tag{7}$$

where p_s is the realization index, N_{realize} is the number of realizations, and γ is a random sequence of signs (i.e. +1 and -1). The encoding function is used to blend the observed data as follows:

$$\widetilde{d}_{\text{obs}}(\mathbf{x}_r, p_s, \omega) = \sum_{\mathbf{x}_s} \alpha(\mathbf{x}_s, p_s) d_{\text{obs}}(\mathbf{x}_r, \mathbf{x}_s, \omega). \tag{8}$$

Similarly, the same encoding function is used to blend the modeled data:

$$S(\mathbf{x}, p_s, \omega) = \sum_{\mathbf{x}_s} \alpha(\mathbf{x}_s, p_s) f(\omega) G_0(\mathbf{x}, \mathbf{x}_s, \omega), \tag{9}$$

where S is the blended source wavefield. Due to the linearity of the wave equation, this wavefield can be simply computed by simultaneously injecting the source functions

at different locations after multiplying by the proper weight. Once S is computed, the blended forward modeling operator can defined as follows:

$$\widetilde{d}(\mathbf{x}_r, p_s, \omega) = -\omega^2 \sum_{\mathbf{x}} S(\mathbf{x}, p_s, \omega) G_0(\mathbf{x}, \mathbf{x}_r, \omega) m(\mathbf{x}), \tag{10}$$

where $\widetilde{}$ is used to indicate blending. The blended forward modeling operator $\widetilde{\mathbf{F}}$ can also be expressed in matrix form:

$$\widetilde{\mathbf{d}} = \widetilde{\mathbf{F}}\mathbf{m}. \tag{11}$$

Finally, the objective function of the blended operator can be written as follows:

$$\widetilde{J}(\mathbf{m}) = \|\widetilde{\mathbf{F}}\mathbf{m} - \widetilde{\mathbf{d}}_{\text{obs}}\|_2^2. \tag{12}$$

Notice that the objective functions changes between realizations, since the encoding function changes. This change of the objective function requires a modification in the minimization scheme as follows:

```
iterate {
        r~ <-- F~ m - d~_obs
        Δm <-- F~* r~
        Δr~ <-- F~ Δm
        m <-- stepper(m, Δm, Δr~)
        }
```

There are two changes in the minimization scheme of the blended objective function compared to the conventional one. First, the computation of the residual is moved inside the loop, because the encoding function changes in each iteration. This change adds the cost of a forward modeling operator to each iteration. Second, the stepper algorithm can only be steepest-descent if the step size is determined with linear optimization. Otherwise, a non-linear conjugate gradient can be performed, requiring a line search in each iteration. In this paper I present only the result of using steepest-descent stepper, because the iteration cost is consistent.

Hessian Estimation

For any linear operator $\mathbf{F}$, the Hessian matrix is computed as follows:

$$H(\mathbf{x}, \mathbf{y}) = \mathbf{F}^*(\mathbf{y})\mathbf{F}(\mathbf{x}), \tag{13}$$

where $\mathbf{x}$ and $\mathbf{y}$ are model coordinates. There are several ways to utilize the Hessian matrix in the inversion process, but I will focus on using its diagonal as a preconditioner to the gradient:

$$\mathbf{s}_k = \frac{\mathbf{g}_k}{\text{diag}\{\mathbf{H}\} + \epsilon\mathbf{I}}, \tag{14}$$

where $\mathbf{g}_k$ is gradient at the k^{th} iteration, $\mathbf{s}_k$ is the preconditioned search direction, $\mathbf{I}$ is an identity operator, and ϵ is a constant used to stabilize the division. For the modeling operator, the diagonal of the Hessian matrix can be written as follows:

$$\text{diag}\{\mathbf{H}\} = D(\mathbf{x}) = \sum_{\omega} \omega^4 |f(\omega)|^2 \sum_{\mathbf{x}_s} |G_0(\mathbf{x}, \mathbf{x}_s, \omega)|^2 \sum_{\mathbf{x}_r} |G_0(\mathbf{x}, \mathbf{x}_r, \omega)|^2. \tag{15}$$

Unlike with forward and adjoint modeling operators, the computations must be done on each source-receiver pair separately. As with LSRTM, the expense of this operation can be reduced by encoding the source or receiver side, or both sides. I first define a receiver-side encoding function β as follows:

$$\beta(\mathbf{x}_r, p_r) = \frac{1}{\sqrt{N_{\text{realize}}}} \gamma(\mathbf{x}_r, p_r), \tag{16}$$

where p_r is the realization index, and the other variables are the same as in the encoding function α. I now define an encoded receiver wavefield:

$$R(\mathbf{x}, p_r, \omega) = \sum_{\mathbf{x}_r} \beta(\mathbf{x}_r, p_r) G_0(\mathbf{x}, \mathbf{x}_r, \omega). \tag{17}$$

This encoding results in the receiver-side blended function $\widetilde{\mathbf{D}}$, which can be written as follows:

$$\widetilde{D}(\mathbf{x}, p_r) = \sum_{\omega} \omega^4 |f(\omega)|^2 \sum_{\mathbf{x}_s} |G_0(\mathbf{x}, \mathbf{x}_s, \omega) R(\mathbf{x}, p_r, \omega)|^2. \tag{18}$$

The cost of one realization of the function $\widetilde{\mathbf{D}}$ is equivalent to an unblended migration of all the shots. Additionally, the source side can also be blended:

$$\widetilde{\widetilde{D}}(\mathbf{x}, p_s, p_r) = \sum_{\omega} \omega^4 |S(\mathbf{x}, p_s, \omega) R(\mathbf{x}, p_r, \omega)|^2. \tag{19}$$

The cost of one realization of the function $\widetilde{\widetilde{\mathbf{D}}}$ is equivalent to migrating one shot only. However, the additional blending results in more crosstalk. Hence, the function $\widetilde{\widetilde{\mathbf{D}}}$ requires more realizations to reduce the crosstalk artifacts than does the function $\widetilde{\mathbf{D}}$.

Although the cost of computing the function $\mathbf{D}$ can be reduced with blending, additional propagation(s) are still required to compute the receiver side. Therefore, preconditioning with the source intensity function $\mathbf{D}_S$ can be done by ignoring the receiver side of the Hessian matrix. The source intensity function can be written as follows:

$$D_S(\mathbf{x}) = \sum_{\omega} \omega^4 |f(\omega)|^2 \sum_{\mathbf{x}_s} |G_0(\mathbf{x}, \mathbf{x}_s, \omega)|^2. \tag{20}$$

The previous formulation computes the exact source function in one iteration of LSRTM. However, if the inversion is done with the blended operator, the source intensity function can be computed using the blended source wavefield as follows:

$$\widetilde{D}_S(\mathbf{x}, p_s) = \sum_{\omega} \omega^4 |S(\mathbf{x}, \mathbf{p}_s, \omega)|^2. \tag{21}$$

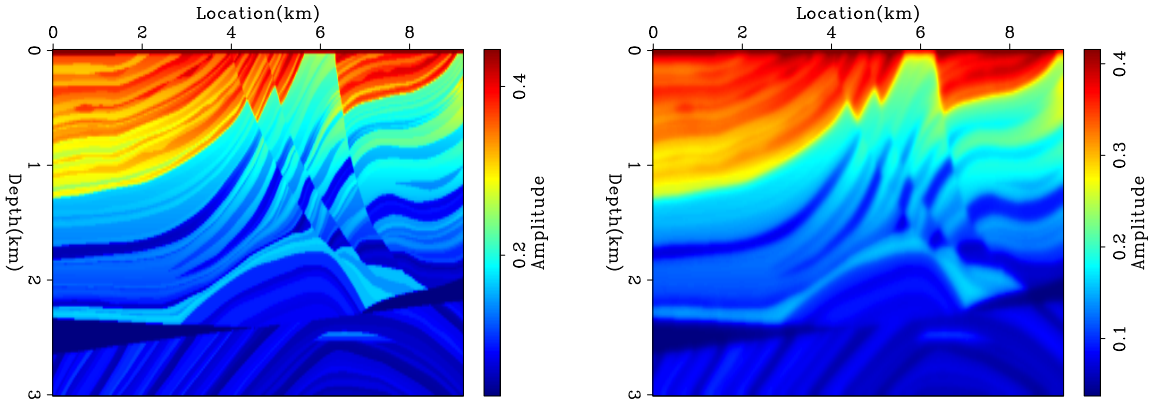
By comparing equation (21) to equation (19), we can see that ignoring the receiver side in the Hessian matrix can be physically interpreted as having receivers everywhere in the subsurface. As a result, the source intensity function overestimates the Hessian matrix. Therefore, I propose a better approximation to the Hessian matrix by using the blended source wavefiled to approximate the receiver-side. This source-based Hessian can be written as follows:

$$\widetilde{D}_{SS}(\mathbf{x}, p_s) = \sum_{\omega} \omega^4 |S(\mathbf{x}, \mathbf{p}_s, \omega)|^4. \tag{22}$$

The function $\widetilde{\mathbf{D}}_{SS}$ approximates the receiver wavefield by the source wavefield. Physically, this assumes that the receivers exist at the same location as the sources. This is a better approximation than the original source intensity function, especially for the fixed-spread geometry. This formulation requires no additional propagation if the source side is blended. However, there are two sources of error in equation (22). First, the receiver spacing could be different than the source spacing, even if their spreads cover the same area. Second, the receiver side should have a different encoding function than the source side. These errors will prevent the source-based Hessian from approaching the true Hessian matrix, regardless of the number of realizations.

SYNTHETIC EXAMPLES

A decimated Marmousi model is used for the synthetic examples. The sampling for both spatial axes is 20 m. A Ricker wavelet with a fundamental frequency of 15 Hz and temporal sampling of 1.5 ms is used to model the data. The receiver spacing is 20 m, and the source spacing is 100 m. Born modeling with a time-domain finite-difference propagator was used in both the observed and the calculated data.



(a) (b)

Figure 1: (a) Marmousi slowness squared and (b) the background model. [**ER**]

Figures 1(a) and 1(b) show the Marmousi slowness squared and the background model, respectively. The reflectivity model, obtained by subtracting the Marmousi slowness squared from the background model, is shown in Figure 2.

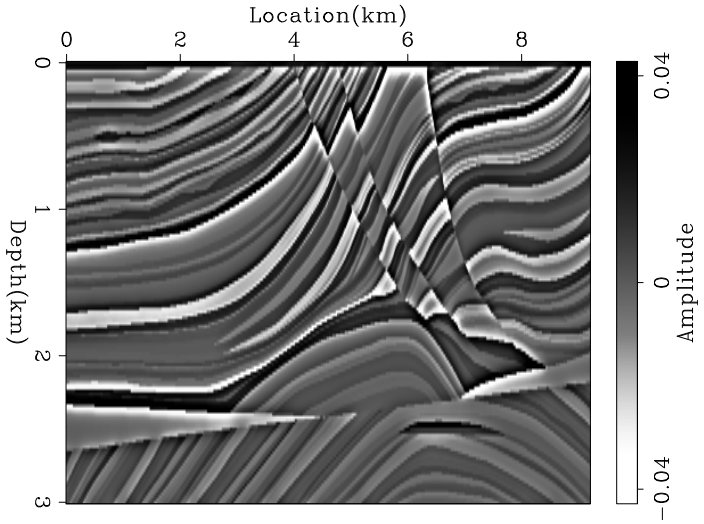
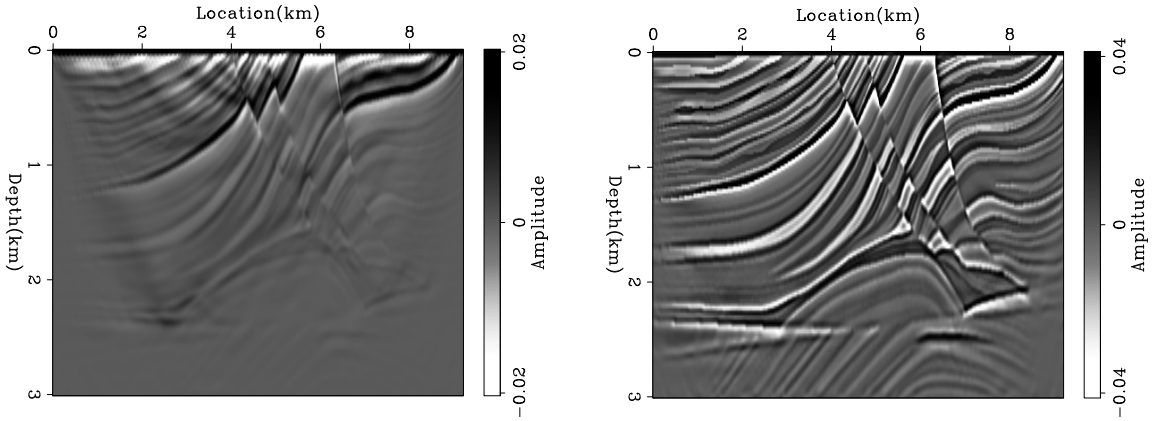


Figure 2: The Marmousi reflectivity model. [**ER**]

First, I ran a conventional LSRTM without any preconditioning for 50 iterations. The result of the fifth iteration is shown in Figure 3(a). This result is dominated by low-frequency noise as well as strong imbalance in amplitudes of the model. Next, I repeated the experiment, this time blending the sources. Figure 3(b) shows the result after 310 iterations, which is equivalent in cost to five iterations of unblended LSRTM. By comparing Figures 3(a) and 3(b), we see the blended LSRTM gives much better results.

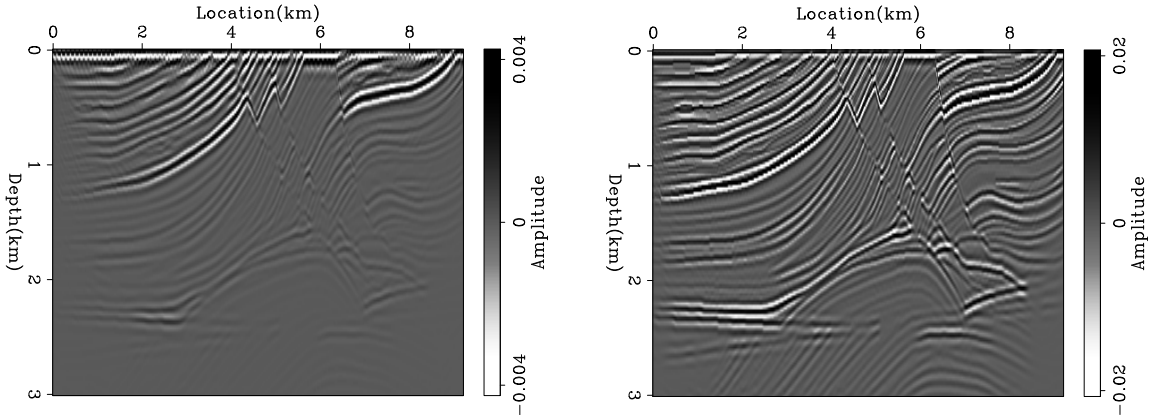


(a) (b)

Figure 3: (a) Conventional LSRTM after 5 iterations and (b) blended LSRTM after 310 iterations. [**ER**]

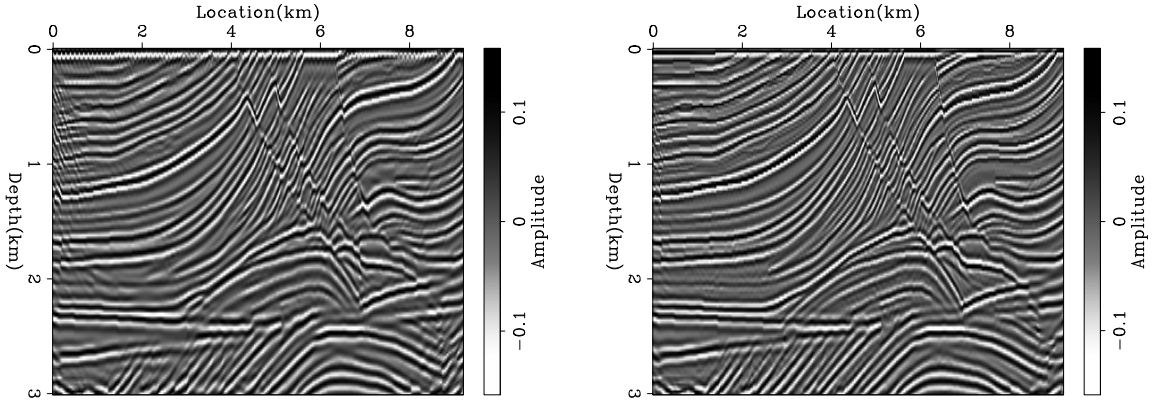
In order to determine whether the difference between the two results is only caused by the low frequency noise, I applied a low-cut filter to both results, as shown in Figures 4(a) and 4(b). Next, I further processed the results to remove the amplitude imbalance by applying an AGC, as shown in Figures 5(a) and 5(b). For comparison,

Figures 6(a) and 6(b) show the true reflectivity model after applying the same processing steps. We can see that the blended LSRTM gives sharper and more accurate images even after processing.



(a) (b)

Figure 4: The result of applying a low-cut filter to Figures 3(a) and 3(b). [**ER**]

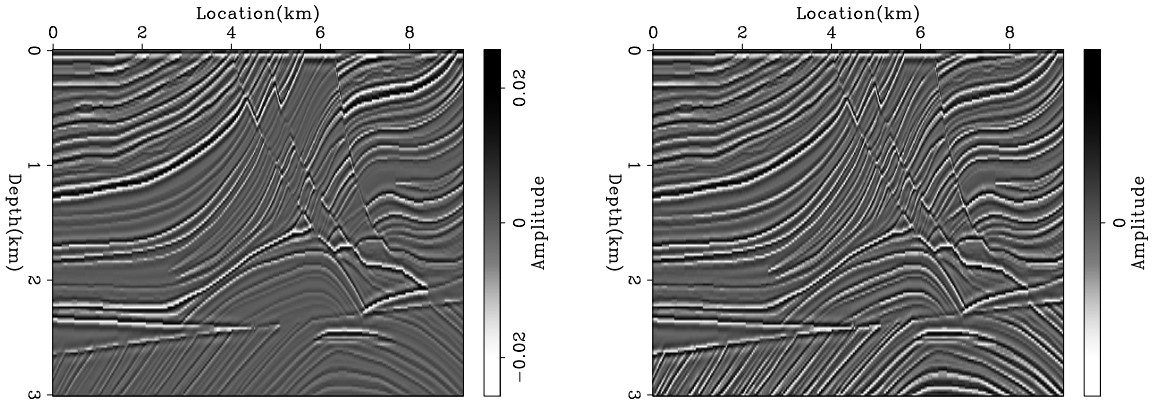


(a) (b)

Figure 5: The result of applying an AGC to Figures 4(a) and 4(b). [**ER**]

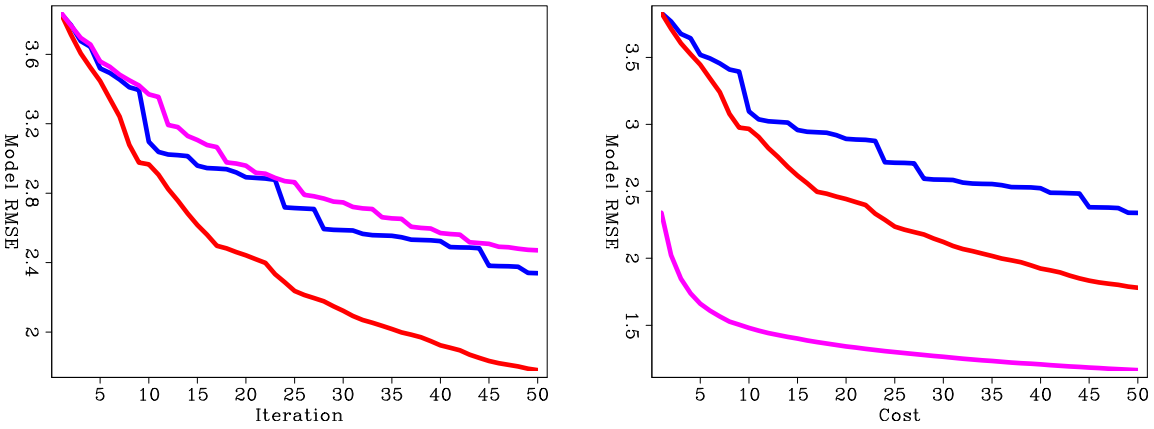
For a more accurate measure of error, I computed the RMS error between the true reflectivity model and the result of each iteration before processing. Figure 7(a) shows three curves: the unblended LSRTM using steepest-descent stepper (LSRTM-SD), the unblended LSRTM using conjugate-direction stepper (LSRTM-CD), and blended LSRTM using steepest-descent stepper (B-LSRTM). It is interesting to see that the blended LSRTM is converging at a similar rate to the unblended LSRTM with the steepest-descent stepper, although their costs are very different. This seems to indicate that while suppressing the crosstalk, the blended LSRTM is also inverting the operator without much loss of efficiency. The only advantage for unblended LSRTM seems to result from having a better stepper.

Figure 7(b) shows the RMS error curves as a function of cost. The cost unit is equivalent to a conventional RTM of all the shots. This Figure clearly shows that for



(a) (b)

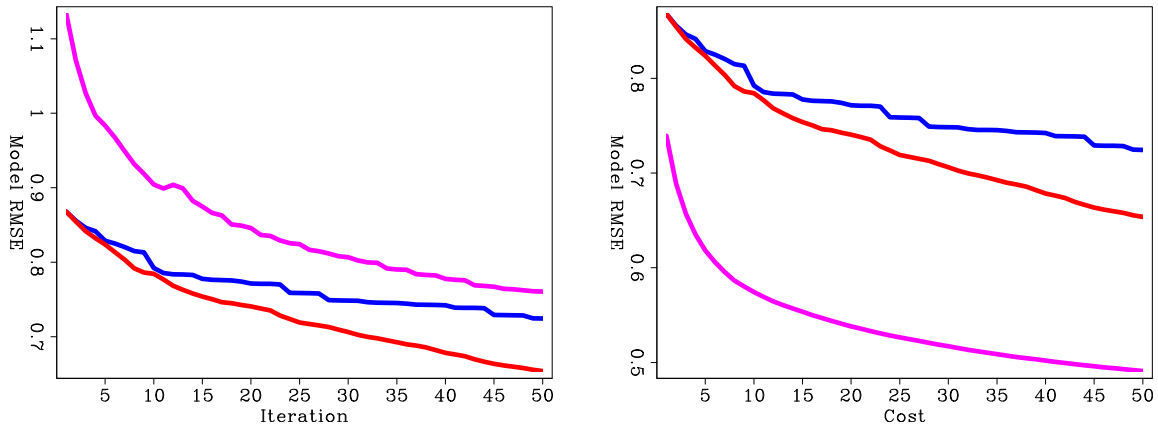
Figure 6: The result of applying (a) a low-cut filter and (b) an AGC to Figure 2. [**ER**]



(a) (b)

Figure 7: The model RMS error versus (a) iteration and (b) migration cost. LSRTM-SD is blue, SLRTM-CD is red, and B-LSRTM is purple. [**CR**]

the same cost, the B-LSRTM gives much better results than the conventional LSRTM regardless of the stepper algorithm.



(a) (b)

Figure 8: The same as Figures 7(a) and 7(b) but after processing the inversion results with a low-cut filter and AGC. LSRTM-SD is blue, SLRTM-CD is red, and B-LSRTM is purple. [**CR**]

Figure 8(a) is the same as Figure 7(a), but after processing the inversion results. The B-LSRTM seems to have a much larger initial error because the crosstalk artifacts, which are high frequency, are amplified by applying a low-cut filter. However, by comparing the results at the same cost, as shown in Figure 8(b), the B-LSRTM is still superior to the other unblended LSRTM inversions.

Next, I computed the exact diagonal of the Hessian matrix. Figures 9(a), 9(b), and 9(c) shows the contribution from three different shots, and Figure 9(d) shows the total diagonal of the Hessian matrix. I then tested the convergence rates of four preconditioners: the Hessian matrix with receiver-side blending (HRB), the Hessian matrix with source- and receiver-side blending (HSRB), the source intensity function with source-side blending (SISB), and the source-based Hessian with source-side blending (SHSB).

Figures 10(a) and 10(b) show the RMS error of each preconditioner compared to the exact Hessian diagonal versus iteration and versus cost, respectively. Both HRB and HSRB approach the true Hessian diagonal, but encoding both sides gives a faster convergence rate per cost. The SISB converges the fastest, but it converges to a solution with a large error. On the other hand, SHSB converges to a much better solution and at a similar rate to SISB. The preconditioners after 50 migrations equivalent cost are shown in Figures 11(a), 11(b), 11(c), and 11(d).

Finally, I ran three LSRTM inversions using HRSB, SISB, and SHSB as preconditioners and compared them to the unpreconditioned B-LSRTM. Figure 12(a) shows the model RMS error of the four curves as a function of iteration. The three preconditioned curves seems to converge at a simiar rate per iteration. However, the convergence rates per cost are different, as shown in Figure 12(b). Although SHSB

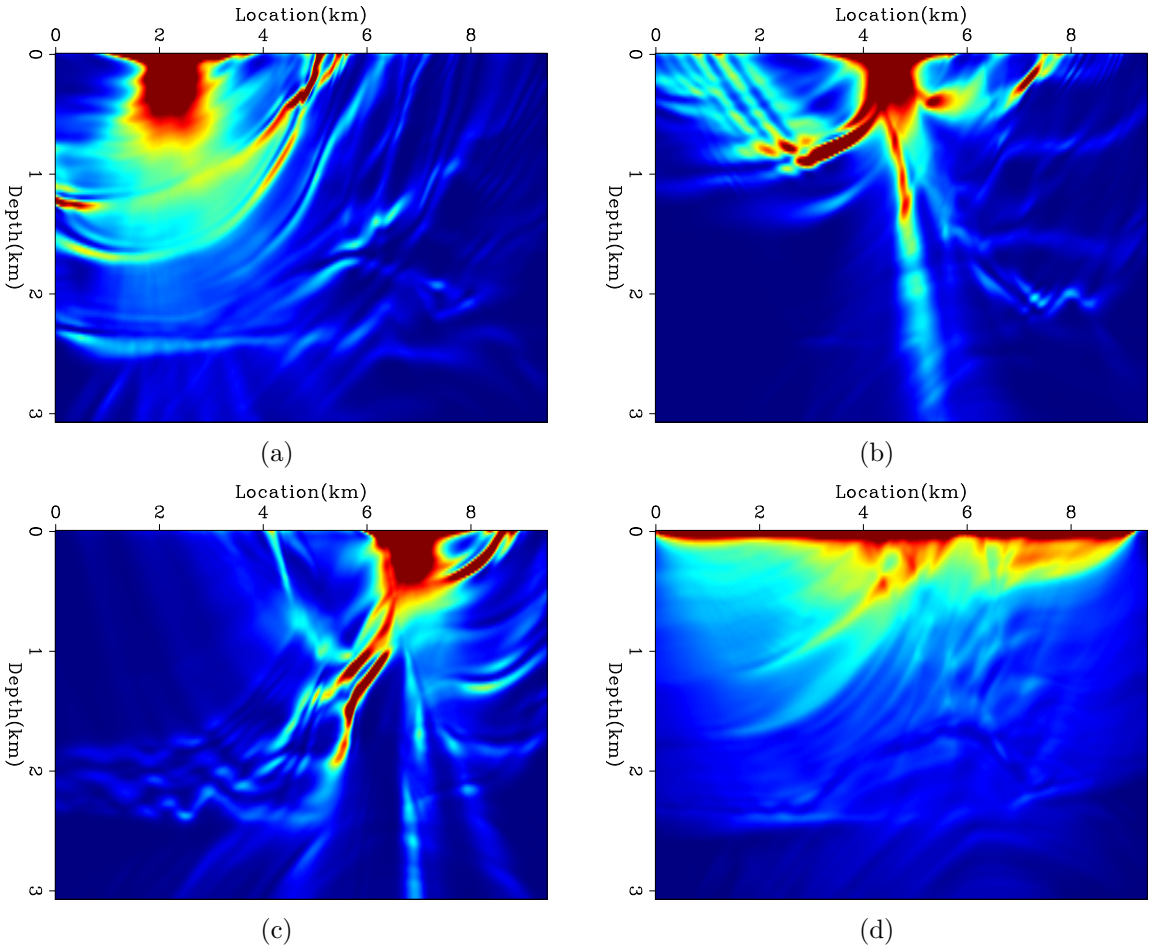


Figure 9: (a), (b), and (c) show the contributions of three different shots and (d) shows the total diagonal of the Hessian matrix. [**CR**]

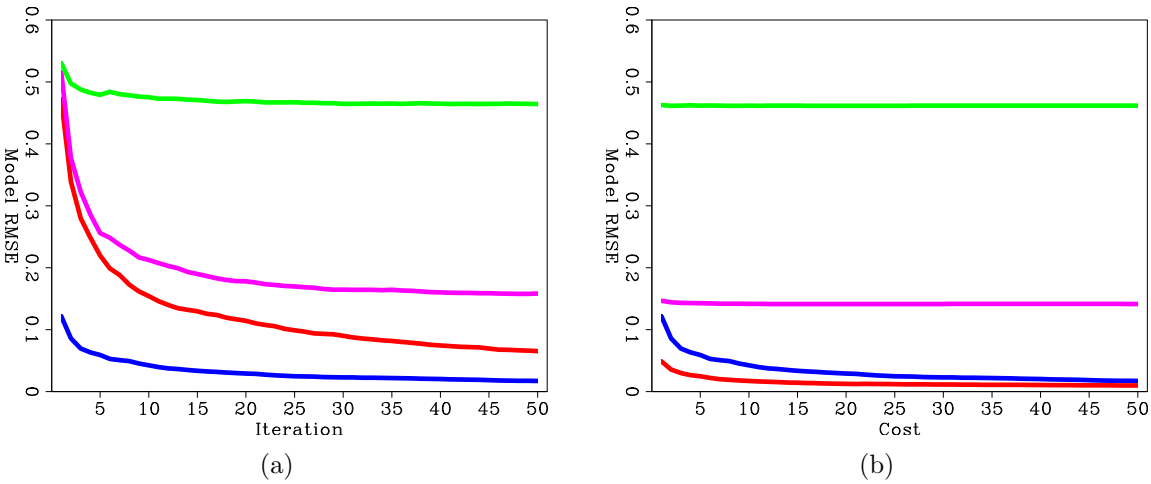


Figure 10: The model RMS error of (a) every iterations/realization and (b) every migration cost. HRB is blue, HSRB is red, SISB is green, SHSB and is purple. [**CR**]

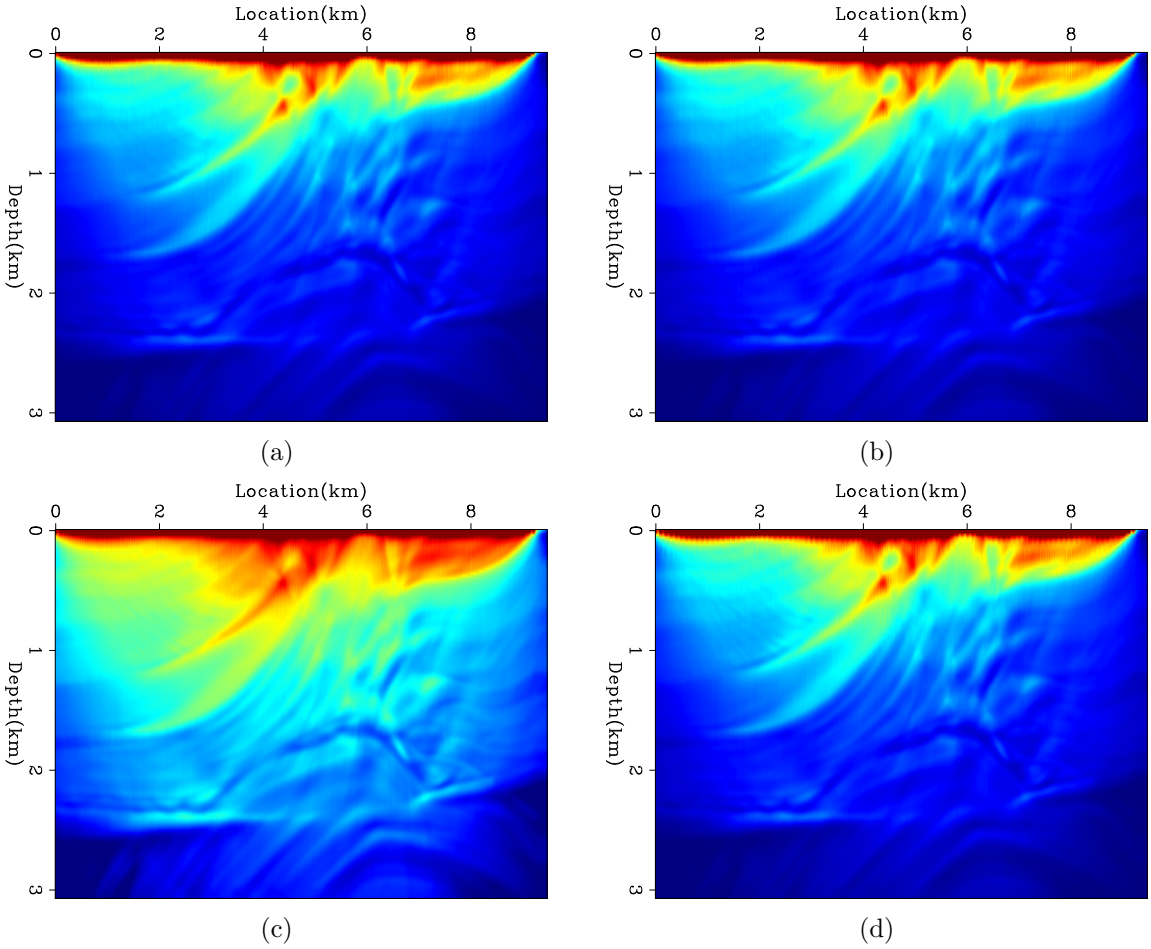


Figure 11: The results after 50 migrations equivalent cost of (a) HRB, (b) HSRB, (c) SISB, and (d) SHSB. [**CR**]

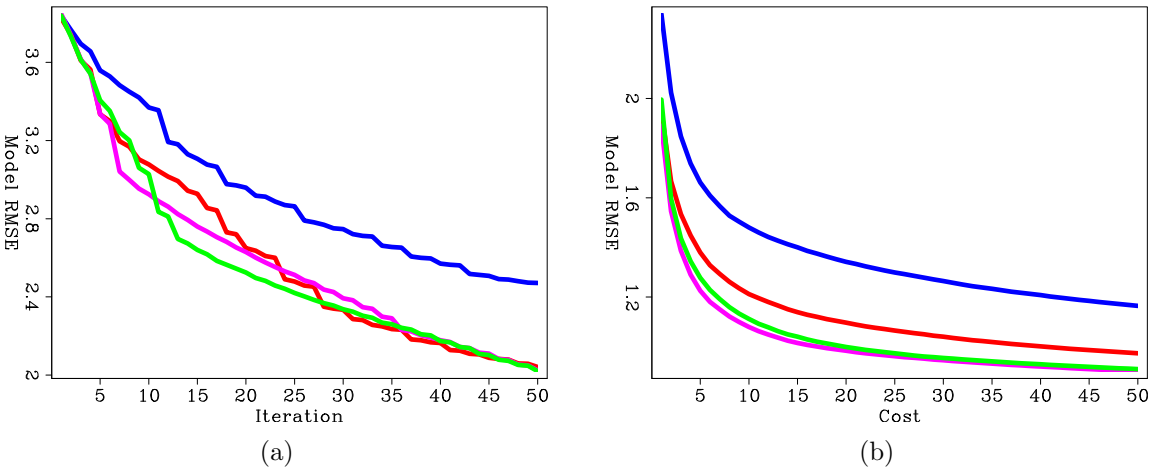


Figure 12: The model RMS error versus (a) iteration and (b) migration cost. No preconditioning is blue, HRSB is green, SISB is red, SHSB and is purple. [**CR**]

had larger error in estimating the Hessian matrix, it resulted in the best convergence rate for LSRTM due to its cheaper cost. The second-best method was HRSB.

As with the previous LSRTM, I processed the results of preconditioned LSRTM and compared it to the processed, true reflectivity. Figures 13(a) and 13(b) show the model RMS error versus iteration and cost, respectively. Processing the models did not change the order of the curves, but, as expected, it reduced the difference between them in the early iterations.

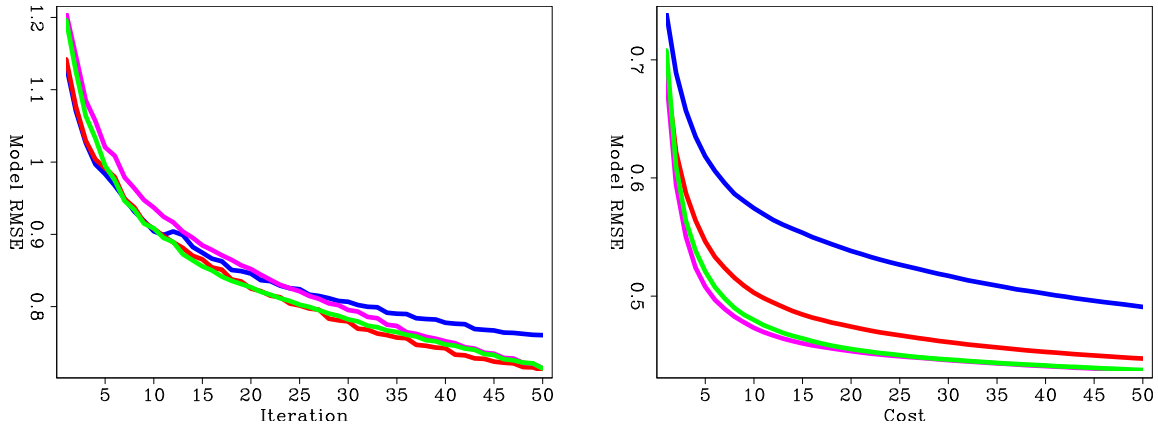


Figure 13: The same as Figures 12(a) and 12(b) but after processing the inversion results with a low-cut filter and AGC. No preconditioning is blue, HRSB is green, SISB is red, SHSB and is purple. [**CR**]

DISCUSSION AND CONCLUSIONS

By examining the results using the Marmousi model, we see that encoding the sources in LSRTM is more efficient than the conventional method. This is true despite the fact that each iteration requires an additional forward operator and the stepper is not as efficient.

When estimating the Hessian matrix, encoding both the source and receiver sides is more efficient than encoding the receiver side only. Moreover, the source-based Hessian is more accurate than the conventional source intensity function. As a preconditioner, the source-based Hessian is the most efficient due to its cheaper computational cost compared to estimating both sides of the Hessian matrix. However, using this source-based Hessian in different acquisition geometries needs to be further tested.

REFERENCES

Ben-hadj ali, H., S. Operto, and J. Virieux, 2011, An efficient frequency-domain full waveform inversion method using simultaneous encoded sources: Geophysics, **76**,

R109–R124.

Boonyasiriwat, C. and G. T. Schuster, 2010, 3D Multisource Full-Waveform Inversion using Dynamic Random Phase Encoding: SEG Technical Program Expanded Abstracts, 1044–1049.

Claerbout, J. F. and S. Fomel, eds., 2011, Image estimation by example: Geophysical soundings image construction.

Gao, F., A. Atle, and P. Williamson, 2010, Full waveform inversion using deterministic source encoding: SEG Technical Program Expanded Abstracts, 1013–1017.

Godwin, J. and P. C. Sava, 2011, A comparison of shot-encoding schemes for wave-equation migration: SEG Technical Program Expanded Abstracts, 32–36.

Jing, X., C. J. Finn, T. a. Dickens, and D. E. Willen, 2000, Encoding multiple shot gathers in prestack migration: SEG Technical Program Expanded Abstracts, 786–789.

Krebs, J. R., J. E. Anderson, D. Hinkley, R. Neelamani, S. Lee, A. Baumstein, and M.-D. Lacasse, 2009, Fast full-wavefield seismic inversion using encoded sources: Geophysics, **74**, WCC177–WCC188.

Morton, S. A. and C. C. Ober, 1998, Faster shot-record depth migration using phase encoding: SEG Technical Program Expanded Abstracts.

Perrone, F. and P. C. Sava, 2009, Comparison of shot encoding functions for reverse-time migration: SEG Technical Program Expanded Abstracts, 2980–2984.

Romero, L. a., D. C. Ghiglia, C. C. Ober, and S. A. Morton, 2000, Phase encoding of shot records in prestack migration: Geophysics, **65**, 426–436.

Sun, P., S. Zhang, and F. Liu, 2002, Prestack migration of areal shot records with phase encoding: SEG Technical Program Expanded Abstracts, 1172–1175.

Tang, Y., 2009, Target-oriented wave-equation least-squares migration/inversion with phase-encoded Hessian: Geophysics, **74**, WCA95–WCA107.

Tang, Y. and S. Lee, 2010, Preconditioning full waveform inversion with phase-encoded Hessian: SEG Technical Program Expanded Abstracts, 1034–1038.