CWP-885

Preconditioned 3D Least-Squares RTM with Non-Stationary Matching Filters

Antoine Guitton[1]

[1] *Center for Wave Phenomena, Colorado School of Mines, Golden, Colorado, USA*

ABSTRACT

I approximate the inverse Hessian $(\mathbf{L}'\mathbf{L})^{-1}$, where $\mathbf{L}$ is a Born modeling operator and $\mathbf{L}'$ is the corresponding adjoint RTM operator, with non-stationary matching filters. These non-stationary filters can be seen as a low-rank approximation of the true inverse Hessian. For 3D least-squares imaging, I use these matching filters to precondition the inversion of prestack seismic data and observe a significant convergence speed-up.

Key words: RTM, Preconditioning, 3D, Matching Filters

1 INTRODUCTION

Assuming that the background velocity model is accurate enough, least-squares imaging improves the resolution of seismic images (deconvolution effect), the illumination of deep reflectors, and the amplitude fidelity (Kuhl and Sacchi, 2003; Clapp, 2005). In addition, migration artifacts due to the acquisition geometry are reduced (Nemeth et al., 1999). With two-way operators, back-scattering noise is also attenuated. However, these important features come at an increased cost, especially in 3D, where many iterations involving computer-intensive migration and demigration steps are needed to best fit the recorded prestack data.

Because least-squares imaging is expensive, many cost-saving strategies are possible if we recognize that the benefits of least-squares inversion comes from the effects of the inverse Hessian operator on the migrated images. Acknowledging this fact, we can either try to approximate the inverse Hessian without running any inversion (Rickett, 2003; Guitton, 2004), speed-up the inversion by designing preconditioning operators (Aoki and Schuster, 2009; Hou and Symes, 2015, 2016), decrease the size of the problem using a target-oriented solution (Tang, 2009), or design pseudo-unitary migration operators (Zhang et al., 2014).

My goal in this paper is to speed-up the inversion by designing a preconditioning operator based on the computation of matching filters. These filters operate in the model space, are non-stationary (vary in x,y and z directions), and approximate the effects of the inverse Hessian. Estimating these filters is straightforward and is explained in details in Guitton (2004). The novelty here is that I am using these filters to precondition the inversion for 3D least-squares reverse-time migration (LSRTM). On a synthetic 3D data example, I am showing a significant speed-up where three to four iterations of the preconditioned inversion yields significantly better results than fifteen iterations of a non-preconditioned one. On a 3D field data example, the cost saving is about a factor two: with 3D data, these are important cost reduction numbers. Compared to other preconditioning techniques such as those using deblurring filters (Aoki and Schuster, 2009), or approximate inverses (Hou and Symes, 2015), matching filters are simple to implement, robust and easily generalizable to more complicated imaging operators (e.g., elastic, anisotropic, etc...).

In the first part of this paper, I introduce the methodologies for the filter estimation and preconditioning parts. In the second section, I illustrate the proposed methodology on a 3D synthetic example and prove that preconditioning with matching filters increases convergence speed. Finally, I apply this methodology to an OBC dataset from the North Sea.

2 THEORY

This section describes the estimation of the non-stationary filters first, and then how these filters can be used to precondition LSRTM. In this paper, a migrated image corresponds to an acoustic velocity perturbation $\mathbf{m}$ according to the following decomposition of the velocity model $\mathbf{m_t}$:

$$\mathbf{m_t} = \mathbf{m_b} + \mathbf{m} \tag{1}$$

where $\mathbf{m_b}$ is the background velocity (i.e., migration velocity). Throughout this paper, I call abusively $\mathbf{m}$ a reflectivity model as well because migrations are usually interpreted that way.

2.1 Filter estimation

In a first step, we need to estimate the non-stationary matching filters. I will first describe how we can compute them and will give some implementation details that help reducing the non-stationary convolution footprint on the processed images.

Let's define $\mathbf{L}$ as a Born modeling operator using a two-way scalar wave equation (velocity only) and $\mathbf{L}'$ as its adjoint (i.e., reverse time migration operator). First, I estimate a migrated image $\mathbf{m_1}$ with RTM using the input data $\mathbf{d}$:

$$\mathbf{m_1} = \mathbf{L}'\mathbf{d} \tag{2}$$

From $\mathbf{m_1}$, I compute a second image $\mathbf{m_2}$ going through a full loop of demigration/migration steps as follows:

$$\mathbf{m_2} = \mathbf{L}'\mathbf{L}\mathbf{m_1} \tag{3}$$

Therefore, $\mathbf{m_2}$ is the result of the migration of the remodeled data. Having the two migration results $\mathbf{m_1}$ and $\mathbf{m_2}$, I estimate a bank of non-stationary filters $\mathbf{a}$ minimizing

$$f(\mathbf{a}) = \|\mathbf{M_2}\mathbf{a} - \mathbf{m_1}\|_2^2 \tag{4}$$

where $\mathbf{M_2}$ is the operator representation of the non-stationary convolution with the remigrated image $\mathbf{m_2}$ (see implementation details later in the text.) The least-squares solution $\hat{\mathbf{a}}$ is given by the normal equations

$$\hat{\mathbf{a}} = (\mathbf{M_2}'\mathbf{M_2})^{-1}\mathbf{M_2}'\mathbf{m_1} \tag{5}$$

showing that the filters are obtained by the correlation of the two migrated images divided by the autocorrelation of $\mathbf{m_2}$. In practice, a conjugate-direction solver (Claerbout and Fomel, 2014) is used to estimate the filter coefficients. Now, from equation (3), we have

$$\mathbf{m_1} = (\mathbf{L}'\mathbf{L})^{-1}\mathbf{m_2} \tag{6}$$

and from equation (4)

$$\mathbf{m_1} \approx \mathbf{M_2}\hat{\mathbf{a}} \tag{7}$$

which shows that the estimated filters $\hat{\mathbf{a}}$ approximate the inverse Hessian $(\mathbf{L}'\mathbf{L})^{-1}$. In other words we have:

$$(\mathbf{M_2}'\mathbf{M_2})^{-1}\mathbf{M_2}'\mathbf{M_1} \approx (\mathbf{L}'\mathbf{L})^{-1} \tag{8}$$

with $\mathbf{M_1}$ the operator representation of the non-stationary convolution with the migrated image $\mathbf{m_1}$. Using the definition of $\mathbf{m_2}$ we end-up with

$$(\mathbf{M_1}'\mathbf{L}'\mathbf{L}\mathbf{L}'\mathbf{L}\mathbf{M_1})^{-1}\mathbf{M_1}'\mathbf{L}'\mathbf{L}\mathbf{M_1} \approx (\mathbf{L}'\mathbf{L})^{-1} \tag{9}$$

as well. These filters can be seen as low rank approximations of the Hessian. In equation (9) we can see the influence of $\mathbf{m_1}$ in the approximation of the inverse Hessian: it appears twice in the numerator and denominator. I illustrated in Guitton (2004) how these filters can efficiently approximate the inverse Hessian for a 2D, one-way source-receiver migration operator. Recently, a similar data-space approach has been proposed by Khalil et al. (2016) and Wang et al. (2016). However, their method approximates $\mathbf{L}\mathbf{L}'$ and can't be directly used for the inversion of over-determined systems. In addition, the matching can be quite expensive to do in the data space. Finally, with field data acquired on an irregular grid, interpolation might be needed before the filter-estimation step.

I present in the next sections some implementation details that improve the convergence of the filter estimation step.

2.2 Non-stationary filtering without edges

I now give some implementation details that improve convergence during the filter estimation step and decrease artifacts. The non-stationary convolution involved in equation (4) is quite expensive in 3D and in practice, a filter is kept constant within a small cube, or patch. A code to compute the non-stationary convolution operator, where the adjoint becomes the filter estimation step is:

```
do iy = 1, size(y)
   ip = aa%pch(iy)
   lag => aa%hlx(ip)%lag
   do ia = 1, size(lag)
      ix = iy - lag(ia)
      if (adj) then
        a(ia,ip)=a(ia,ip)+y(iy)*x(ix)
      else
        y(iy)=y(iy)+a(ia,ip)*x(ix)
      end if
    end do
end do
```

In the code above, `y` is the non-stationary convolution output (output of forward operator), `x` the operator and `a` the unknown filter coefficients (output of adjoint operator) and `ip` is the patch index number. For one filter per output point, the index `ip` goes from `1` to `size(y)`. For one filter for the whole dataset, the index `ip` is constant and we end up with a more familiar stationary convolution. There are more details in this code involving Helical boundary conditions. I refer the reader to Claerbout and Fomel (2014) for a more complete description.

One issue with this implementation of the non-stationary convolution is that we end up with one filter per patch in the output vector space `y`. In other words, a filter is only applied to a patch in the output space. Doing so, because filters are different from patch to patch, we might see the patching's footprint in `y` (artifacts). Another consequence is that convergence to estimate `a` will slow down because the inversion has to fight against these artifacts that contaminate the residual. If we are expecting abrupt changes in the output space, then the code for non-stationary filtering above would be fine. For migrated images, where more smoothness is desired, a different implementation of the non-stationary convolution is needed.

There are two ways to correct for this. One is to add a regularization term in the filter estimation step to penalize strong variations between neighboring filters. This has also the advantage of helping the inversion which becomes massively under-determined with so many coefficients to estimate in the non-stationary case; however, this increases the cost of each iteration as well. Another way is to modify the inner part of the convolution above so that instead of having one filter per output patch in y we have one filter per input patch in x:

```
if (adj) then
   a(ia,ip)=a(ia,ip)+y(ix)*x(iy)
else
   y(ix)=y(ix)+a(ia,ip)*x(iy)
end if
```

What I have done here is quite simple: for the forward and adjoint operations, I merely swapped the indexes iy and ix and kept everything else the same (remember that y and x have the same size). Now, we have a filter constant in a patch in the input space. In other words, I transposed the non-stationary operator. It turns out that this simple change of variable attenuates the patching artifacts and improves convergence for the filter estimation step. If you do this, you have to make sure that your other codes using these filters are consistent with this convention. The reader is encouraged to read Margrave (1998) for a detailed discussion of these two implementations of non-stationary filtering.

2.3 Preconditioned inversion

Now that I have estimated the non-stationary filters $\hat{\mathbf{a}}$, I can use them to precondition the least-squares inversion. In a nutshell, using a conjugate-direction solver, the non-preconditioned inversion works as shown in Algorithm 1, where $\mathbf{d}$ is the data vector, $\mathbf{rd}$ the data residual, $\mathbf{x}$ the model vector (to be estimated), $\mathbf{L}$ the linear operator ($\mathbf{L}'$ being its adjoint), and cg() computes an update for $\mathbf{rd}$ and $\mathbf{x}$. For the preconditioned case, we end up with Algorithm 2 where $\mathbf{A}$ is the operator representation of the non-stationary convolution with $\hat{\mathbf{a}}$.

Algorithm 1 Simple Solver

```
1: $\mathbf{r_d} \leftarrow \mathbf{Lx} - \mathbf{d}$
2: for iter do
3:     $\mathbf{g} \leftarrow \mathbf{L}'\mathbf{r_d}$
4:     $\mathbf{gg} \leftarrow \mathbf{Lg}$
5:     $(\mathbf{r_d}, \mathbf{x}) \leftarrow \mathtt{cg}(\mathbf{r_d}, \mathbf{x}, \mathbf{g}, \mathbf{gg})$
6: end for
```

Algorithm 2 Preconditioned Solver

```
   $\mathbf{r_d} \leftarrow \mathbf{Lx} - \mathbf{d}$
2: for iter do
       $\mathbf{g} \leftarrow \mathbf{AL}'\mathbf{r_d}$
4:     $\mathbf{gg} \leftarrow \mathbf{Lg}$
       $(\mathbf{r_d}, \mathbf{x}) \leftarrow \mathtt{cg}(\mathbf{r_d}, \mathbf{x}, \mathbf{g}, \mathbf{gg})$
6: end for
```

The implementation in Algorithm 2 is similar to the weighted CG algorithm of Hou and Symes (2016) where the non-stationary convolution is used in the definition of the model-space inner product. Note that the updating step cg() in both algorithms only involves inner products in the data space and are not directly affected by $\mathbf{A}$. The preconditioned algorithm converges if $\mathbf{A}$ is positive definite. By testing this condition at each iteration (i.e., $\mathbf{z}'\mathbf{Az} > \mathbf{0}$ for any $\mathbf{z}$) , I came to conclude that $\mathbf{A}$ always is.

In the next section, I present LSRTM results for 3D synthetic and field data examples with and without preconditioning and show that the matching filters improve the convergence of the inversion significantly.

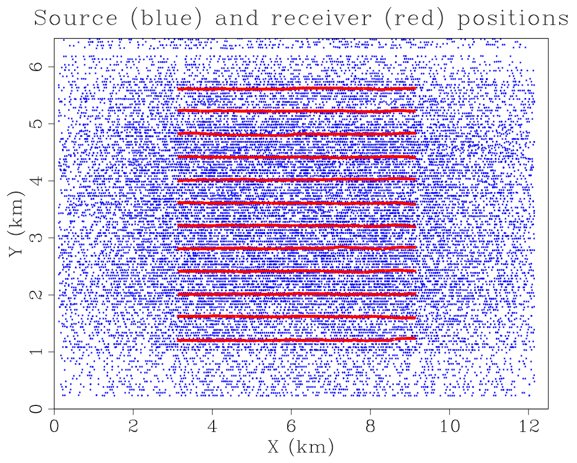
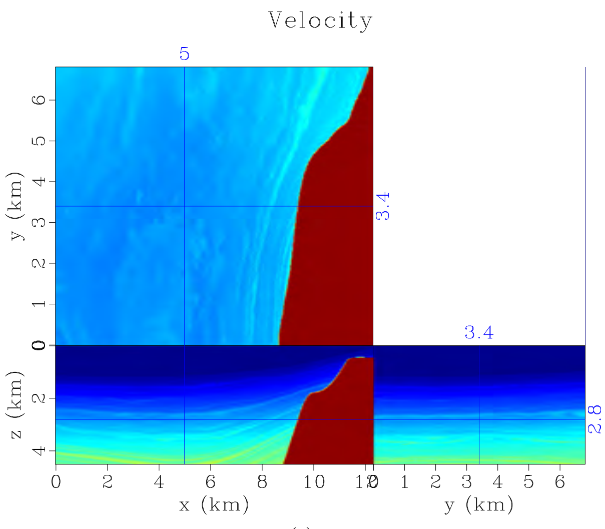


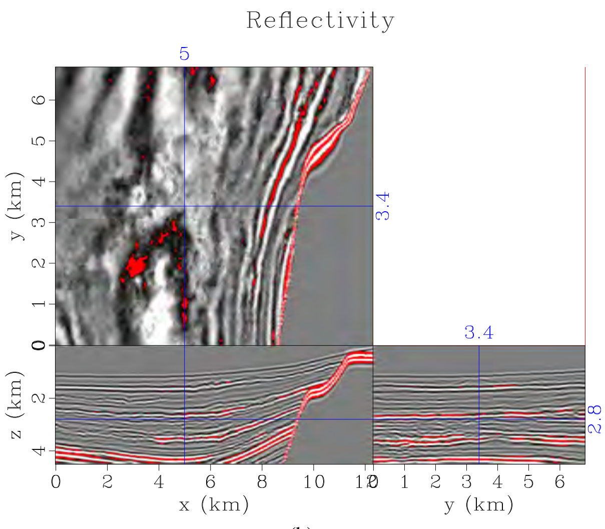
Figure 1. Source (blue) and receiver positions (red) for the OBC dataset

3 A 3D SYNTHETIC DATA EXAMPLE

For this 3D synthetic example, I use a subset of the 3D SEAM model for the reflectivity and velocity models and opt for an OBC-style acquisition geometry. The shot/receiver geometry is similar to the one used in the 3D field OBC example that I will show in the next section and is displayed in Figure 1. In this experiment, I have twelve parallel receiver lines every 400 m. at a depth of around 90 m. In the field, the acquisition is done for one cable-pair at a time, where the shots are covering a 12 km by 2.4 km rectangle above the two receiver-line positions before the cables are moved by 800 m. For this experiment, I model and migrate 288 3D receiver gathers, with around 4200 traces (shot positions) per gather. I generate Born-modeled data using the velocity and perturbation models of Figures 2(a) and 2(b), respectively. There is a salt-body on the right side of the model. The maximum frequency of the data is 20 Hz. I estimate the reflectivity model in two steps. First, I apply a first-order vertical derivative to the velocity model to obtain a high-wavenumber reflectivity model. Then, I smooth the high-wavenumber model to obtain Figure 2(b). I use this last model as the true perturbation model to generate the synthetic Born-modeled data, while Figure 2(a) constitutes the background velocity field. Therefore the goal of LSRTM is to recover an image as close as possible to Figure 2(b). Now, I describe the different stages of the approximate inverse Hessian computation.

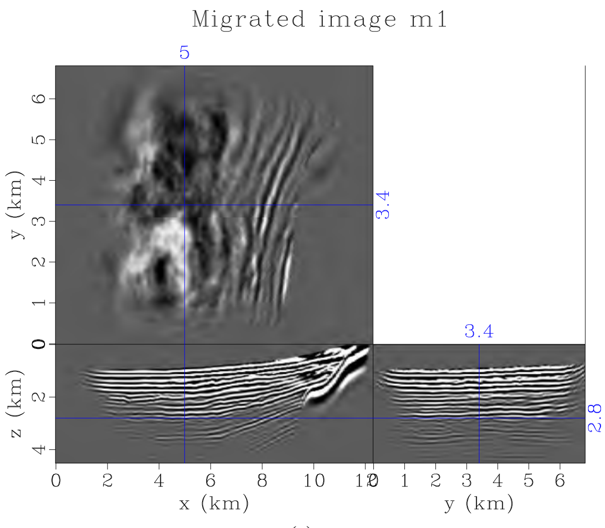


(a)

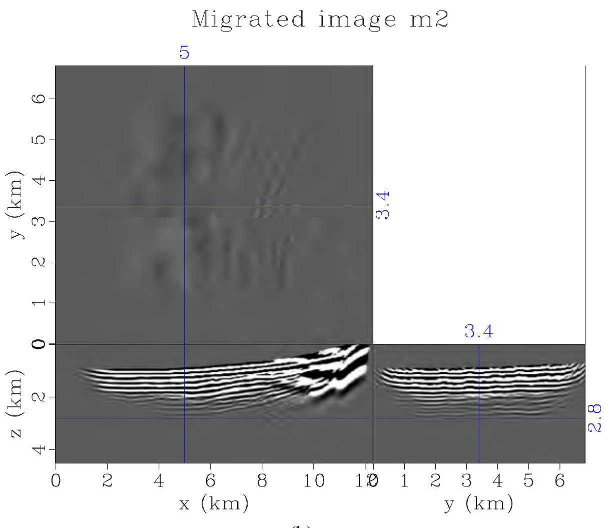


(b)

Figure 2. (a) velocity and (b) reflectivity models used to generate the Born data. In (b), clipped values are showed in red.



(a)



(b)

Figure 3. RTM images (a) $\mathbf{m_1} = \mathbf{L}'\mathbf{d}$ and (b) $\mathbf{m_2} = \mathbf{L}'\mathbf{L}\mathbf{m_1}$ needed to estimate the filters. Note the amplitude decrease in the deepest parts of $\mathbf{m_2}$ compared with $\mathbf{m_1}$.

3.1 Computing $\mathbf{m_1}$ and $\mathbf{m_2}$

The first step consists in computing a migrated image $\mathbf{m_1}$ shown in Figure 3(a) and a re-migrated image $\mathbf{m_2} = \mathbf{L}'\mathbf{L}\mathbf{m_1}$ shown in Figure 3(b). Because the Born-modeling operator $\mathbf{L}$ is not unitary, the re-migrated image $\mathbf{m_2}$ is significantly different from $\mathbf{m_1}$ with much lower amplitudes in the deepest parts of the model. This decrease of amplitudes is also visible comparing the true reflectivity model in Figure 2(b) and the migrated image $\mathbf{m_1}$ in Figure 3(a): the matching filters will compensate for these discrepancies. Also, note that the RTM images were processed to remove the back-scattering noise resulting from the sharp velocity contrast where salt is present. This processing step is necessary to focus the filter estimation on the reflectors only.

3.2 Computing the 3D matching filters

The second step consists in estimating 3D matching filters to minimize the objective function in equation (4). I estimate 3D filters of dimensions 15x15x15 which, admittedly, might be too many coefficients given that one of the main impact of the inverse Hessian on the migrated images is an amplitude balancing effect. I keep these filters constant within a patch of 5x5x5 samples, thus minimizing the numbers of filter coeffi-

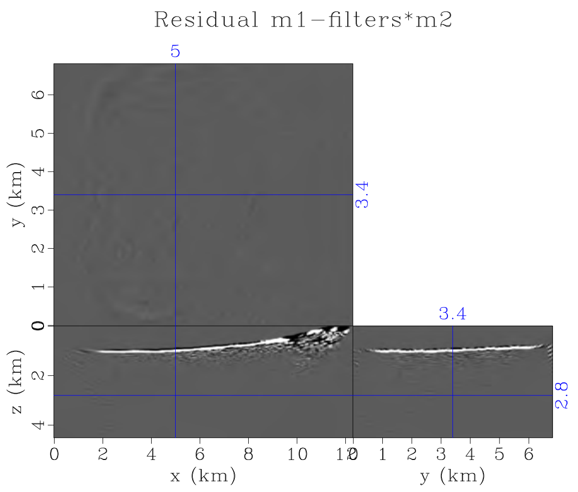


Figure 4. Residual $\mathbf{m1} - \mathbf{M_2 a}$ after 400 iterations. The difference is very small proving that the filters are approximating the inverse Hessian well.

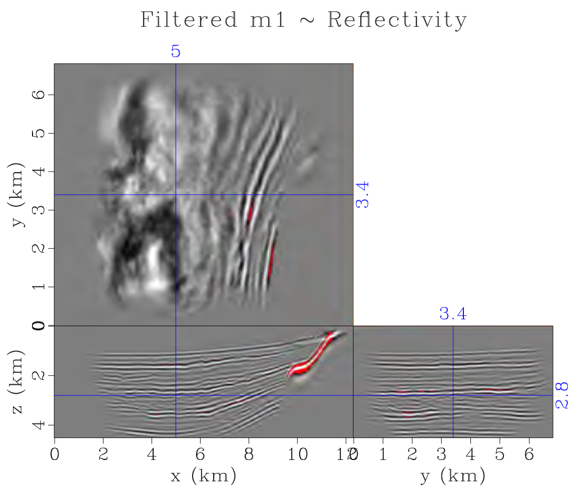


Figure 6. Improved migration $\mathbf{a} * \mathbf{m_1}$. The clip value of Figure 2(b) is used here for display so that both images are directly comparable.

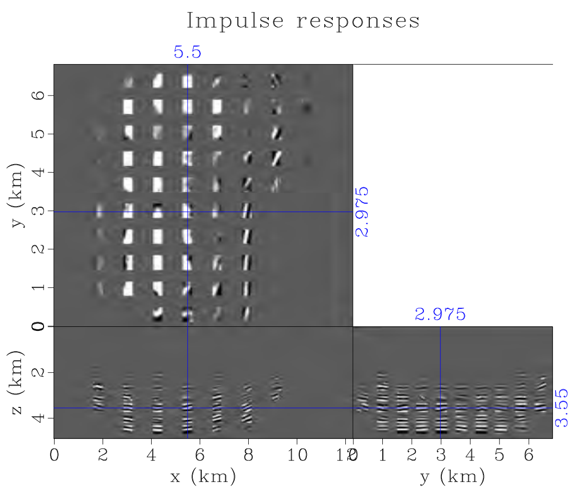


Figure 5. Impulse responses at few locations of the model. Overall and as expected, the filter coefficients get stronger with depth.

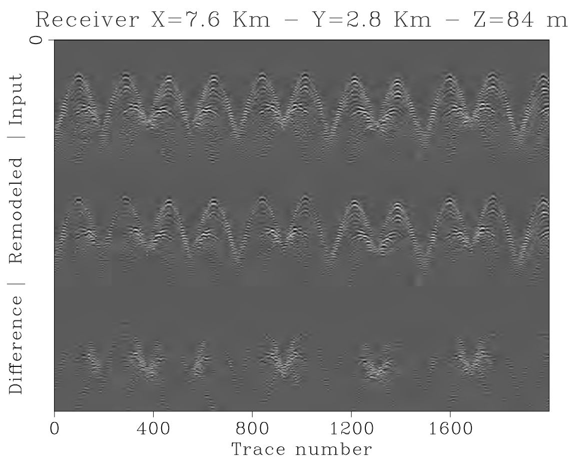


Figure 7. Comparison between the "observed" data (estimated from the true models in Figures 2(a) and 2(b)), modeled data (estimated from the improved RTM result of Figure 6), and data residual (bottom) for one receiver gather.

cients to estimate. The number of patches is 40 in the z direction, 100 in x, and 50 in y, for a total of 675×10^6 filter coefficients to estimate. Because the filters are constant within a patch in the input space and not in the output space, no regularization across filters is applied.

I iterate 400 times, which yields a small residual (Figure 4). I show in Figure 5 a subset of the non-stationary filters applied to equally spaced spikes (i.e., impulse responses). One of the first striking feature is the increase of amplitude with depth, as expected. We can also notice that the filters conform to the geology.

Having these filters, it is easy to compute an improved RTM image that, hopefully, will look closer to the true reflectivity function in Figure 2(b). I show in Figure 6 such a result. Comparing with Figure 3(a), we see the amplitude-balancing effect of the filters. Comparing with Figure 2(b), i.e. the true reflectivity model, we notice that the filters indeed help recovering it very well. To better assess the quality of the inverse Hessian approximation, I model data from Figure 6 and compare them in Figure 7 with the "observed data" derived from the models in Figures 2(a) and 2(b). The non-stationary filters help recovering most of the initial reflectivity: the difference between the observed and modeled data is indeed small. Having a good approximation of $(\mathbf{L}'\mathbf{L})^{-1}$, I show in the next section how the filters help speeding-up LSRTM.

3.3 3D LSRTM Inversion

Now, I am using these filters to improve the convergence of LSRTM. With LSRTM, given some prestack data $\mathbf{d}$, we mini-

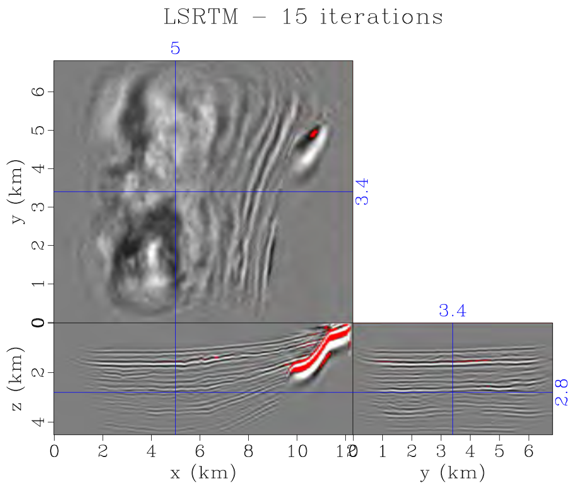


Figure 8. LSRTM result after 15 iterations. The amplitudes are matching those of the true model in Figure 2(b) well, except in the deepest parts where the amplitudes are still too weak. Compare also with the simple filtering result of Figure 6.

mize

$$g(\mathbf{m}) = \|\mathbf{Lm} - \mathbf{d}\|_2^2 \tag{10}$$

where $\mathbf{L}$ is the acoustic Born modeling operator of equation (2) and $\mathbf{m}$ the reflectivity model to be estimated. The adjoint $\mathbf{L}'$ is the RTM operator without model extensions and/or anisotropy. But first, I run fifteen iterations of LSRTM without any preconditioning (Algorithm 1). The final image, using the same clip values as in Figure 2(b), is shown in Figure 8. For the inversion, I scale the gradient at each iteration by the receiver illumination map. In addition, back scattering noise is attenuated applying a mild high-pass filter. In equation, the gradient is computed as follows:

$$\nabla g(\mathbf{m}) = \mathbf{WBL}'\mathbf{r_d}, \tag{11}$$

where $\mathbf{r_d}$ is the data-space residual, $\mathbf{B}$ is the bandpass operator, and $\mathbf{W}$ is the illumination-compensation operator (diagonal). The amplitude fidelity is improved after inversion. Comparing with the filtering result of Figure 6, I notice that the deepest parts of the model (z $>$ 2km) are still lacking strength: more iterations and/or a better illumination-compensation scheme (e.g., incorporating the source illumination as well) would help. However, LSRTM is able to extend the image on the edges of the model better than the simple filtering approach. I show in Figure 9 a comparison between a shot and the corresponding residual after 15 iterations. Except for the salt events, visible as strong diffractions, and for the latest arrivals, corresponding to the not-so-well recovered deep events, the matching is pretty good.

Then, I use the filters estimated in the previous section to precondition the inversion as explained in Algorithm 2. At each iteration, I run a simple test to verify that $\mathbf{A}$ (i.e., the convolution with the matching filters) is positive definite by making sure that $(\mathbf{L}'\mathbf{r_d})'\mathbf{A}(\mathbf{L}'\mathbf{r_d}) > 0$. In essence, I make sure that the gradient vector is still pointing at a descent direction.

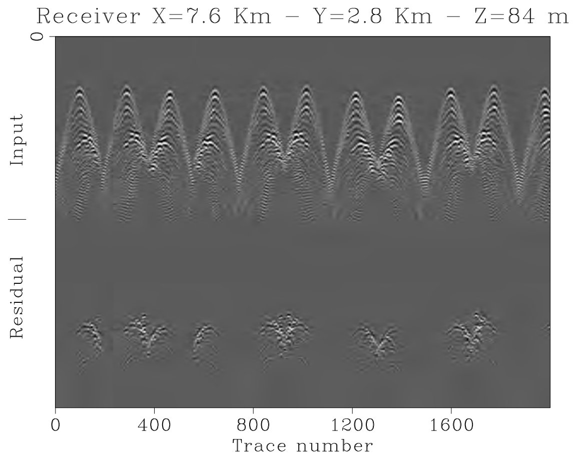


Figure 9. Comparison between the input data (top) and the residual after 15 iterations of LSRTM.

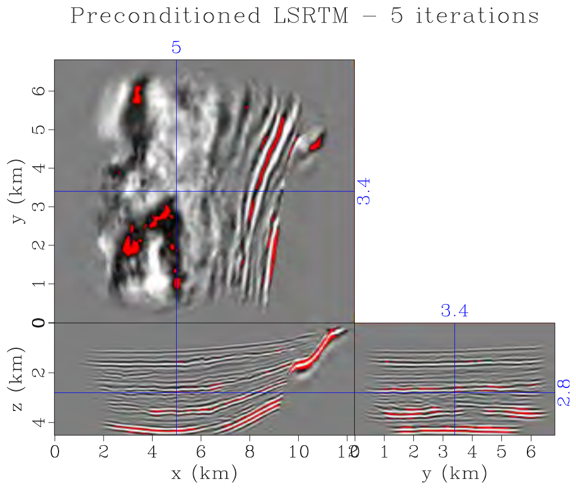


Figure 10. Preconditioned LSRTM with non-stationary 3D matching filters: the amplitude fidelity is improved with far fewer iterations.

Now, the gradient is computed as follows :

$$\nabla g(\mathbf{m}) = \mathbf{ABL}'\mathbf{r_d}, \tag{12}$$

which is very similar to equation (11) where $\mathbf{W}$ is replaced by the non-stationary convolution operator $\mathbf{A}$. I show the inversion results after 5 iterations in Figure 10. The estimated reflectivity looks very close to the true model in Figure 2(b): notice how similar the clipped values are. Because the amplitudes are so weak on the edges in $\mathbf{m_1}$ and $\mathbf{m_2}$ (and thus not well covered by our filters, as seen in Figure 5), the preconditioned inversion doesn't do as well as the standard LSRTM in these areas. However, after two or three iterations of preconditioned LSRTM, I could switch to LSRTM to recover them (a strategy I use with the field data example in the next section). Finally, I show in Figure 11 convergence plots for the two methods: the preconditioned inversion converges in basically two to three iterations. Notice how the objective function for the standard LSRTM has a lower value than the precondi-

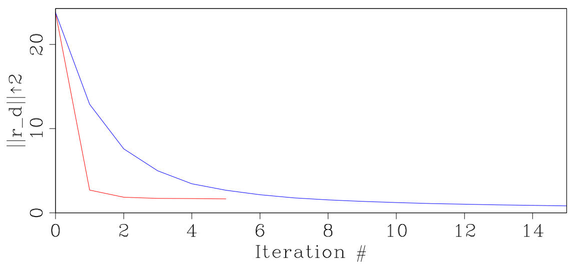


Figure 11. Evolution of the objective function $g(\mathbf{m})$ with iterations for (red) the preconditioned inversion and (blue) standard LSRTM.

tioned one at late iterations. These small differences stem from the edges of the model where the standard LSRTM performs better. Where it matters, i.e. in the middle of the model, the preconditioned inversion outperforms standard LSRTM.

In the next section, I apply the preconditioned LSRTM method to a North-Sea OBC dataset from the Volve field (Szydlik et al., 2007).

4 A 3D OBC FIELD DATA EXAMPLE FROM THE NORTH SEA

Now, I apply 3D LSRTM with and without preconditioning to a field dataset from the North Sea shot over the Volve field. The acquisition geometry is similar to the one used in the synthetic example. The maximum frequency of the data is again 20 Hz. This area has a relatively strong anisotropy and δ, ϵ models are available, but I don't include them in my computations. The velocity model in Figure 14(a) shows the top and base chalk layer very well between z=2.5 km and z=4 km: its proper delineation is one of the main target of the imaging process.

I display some observed traces for one receiver gather in two figures. For t<2 s., Figure 12 shows the early arrivals. For t>2 s. Figure 13 shows the late arrivals. In both figures, the top row corresponds to the observed data, while the second and third rows correspond to the residuals after LSRTM with and without preconditioning, respectively. In the observed data, there are strong amplitudes at early times due to source effects: quasi-mono-frequency events with water velocities are present everywhere and seem to over-shadow reflections. At later times, remaining slow converted waves are visible. I show in Figure 13 the top and base chalk reflections.

First, I compute the RTM image shown in Figure 14(b): the top panel shows an interesting depth-slice at 1.225 km where channel structures are present. We notice weak reflections in the deepest parts of the model below the chalk and strong migration artifacts on the edges of the image (as seen in the corner of the depth slice). Then, I run 15 iterations of LSRTM and display the final image in Figure 14(c). The first thing to notice are the stronger reflectors below the chalk layer. Then we also see the more balanced amplitudes in the depth slice and the attenuation of the migration artifacts on the edges. Although not easily noticeable, the resolution of the reflectors above chalk has increased a lot: a strong black reflector above the chalk layer at z≈2 km appears.

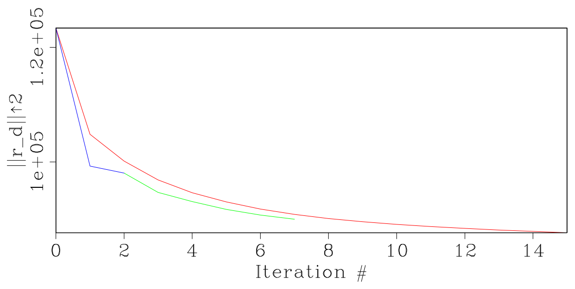


Figure 16. Evolution of the objective function $g(\mathbf{m})$ with iterations for (blue) the preconditioned inversion, (red) standard LSRTM without preconditioning, and (green) LSRTM without preconditioning using Figure 15(a) as a starting guess.

Next, after estimating 3D matching filters from the migration result of Figure 14(b) and the re-migrated image (not shown here), I run only two iterations of preconditioned LSRTM. Note that due to weak reflections in the deepest parts of the model, the size of each filter is now 21x21x21 (see the Discussion section for more details.) It turns out that after only two iterations, the residual stops decreasing, not improving the image further. The result is shown in Figure 15(a). We notice most of the beneficial aspects of LSRTM with only two iterations (i.e. decreasing artifacts, improved illumination). However, looking in more details, we don't see a significant improvement in resolution: I will try to explain this later. Overall on this example, I estimate that we get about 85% of the LSRTM benefits for 15% of the cost. Comparing the residuals for both LSRTM with 15 iterations without preconditioning and LSRTM with 2 iterations with preconditioning for one receiver gather, we notice that the data fit is slightly better with the LSRTM without preconditioning at early times (Figures12), while the preconditioning result is slightly better at later times (Figure 13). This difference is due to the strong amplitude correction of the filters early on in the deepest parts of the model.

Finally, to get all the benefits of LSRTM, I use the result of Figure 15(a) as the starting model for five more iterations without preconditioning of LSRTM. I show the final model in Figure 15(b). Now, this last result is much closer to Figure 14(c) for half the computing cost. The arrow labeled "1" in Figure 15(b) shows a reflector that was not visible with preconditioning only in Figure 15(a). Arrow "2" also shows in a depth-slice the reduction in artifacts and improved illumination. Figure 16 shows the convergence plots for all three results (LSRTM without preconditioning, LSRTM with preconditioning, LSRTM without preconditioning using the preconditioned result as a starting guess). The convergence is not as good as for the synthetic case due mostly to noise in the data (high amplitude, coherent) and inaccuracies of the velocity model. For the same reasons, the dramatic speed-up we could witness using the preconditioning scheme for the synthetic example is reduced with this field dataset.

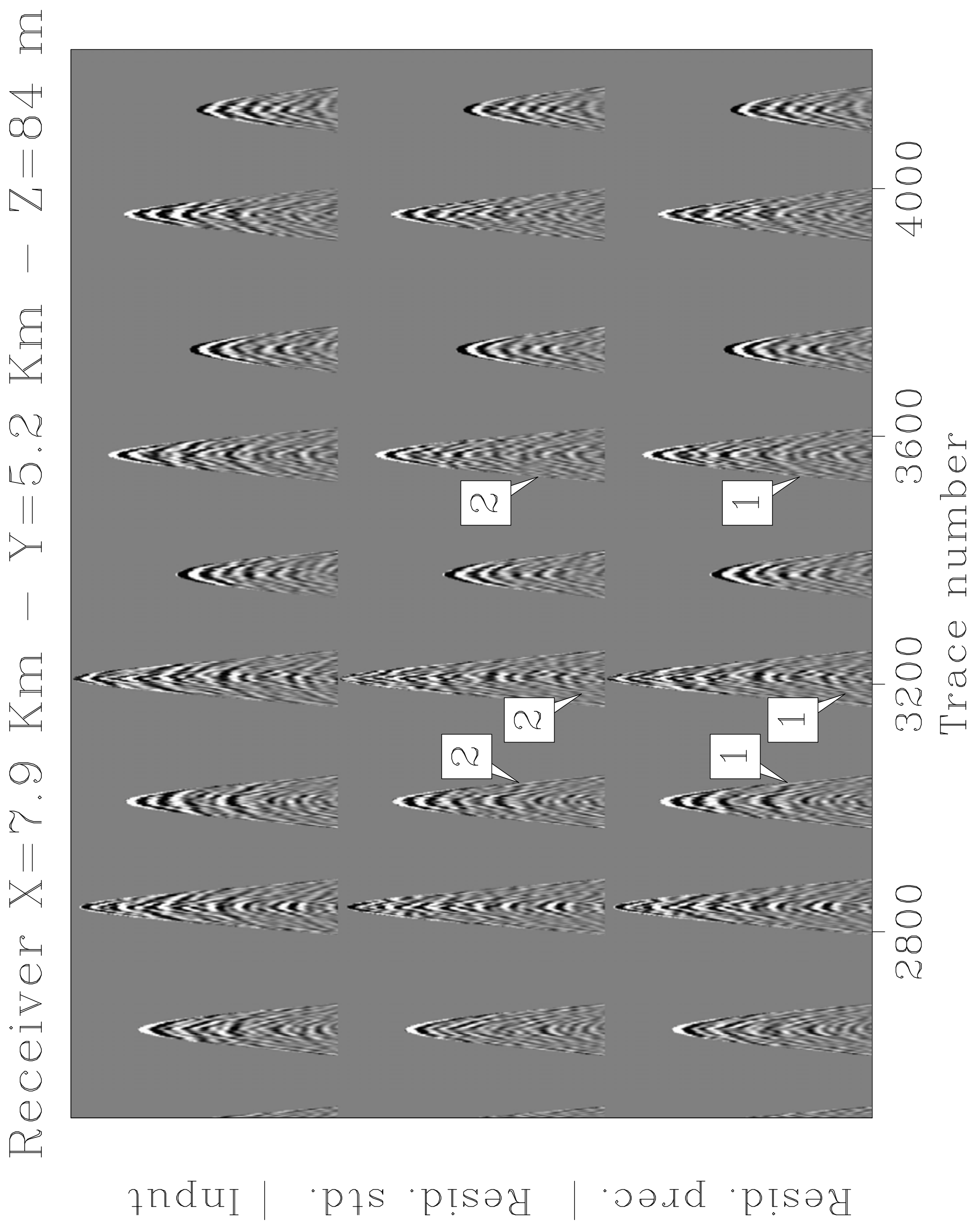


Figure 12. Top part of one receiver gather (0.6 s.$<$ t $<$ 2 s). Top: input gather. Middle: Residual after 15 iterations of LSRTM. Bottom: Residual after 2 iterations of LSRTM with preconditioning. The arrows show events that are better matched after 15 iterations without preconditioning (marked as 2), than after 2 iterations with (marked as 1).

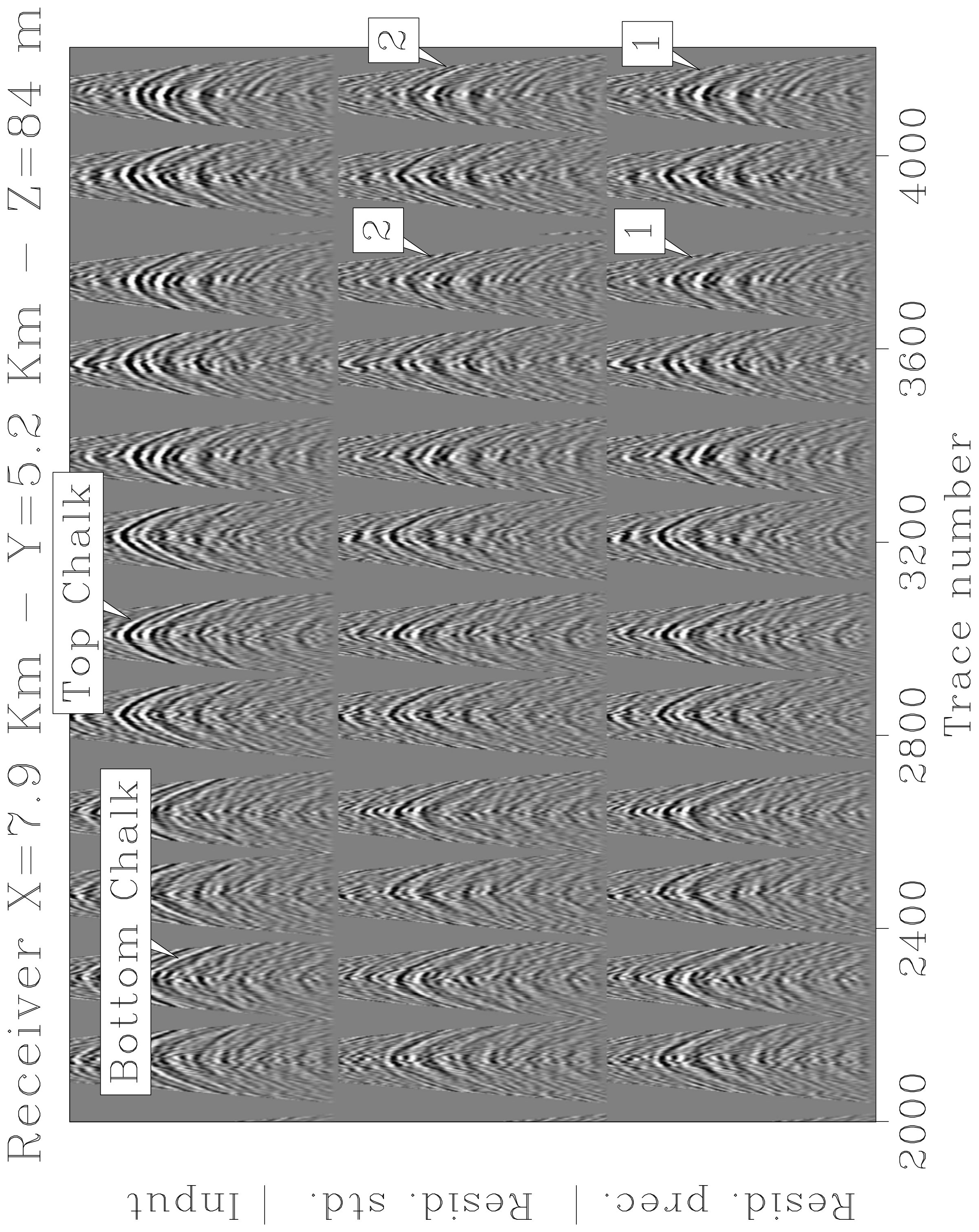
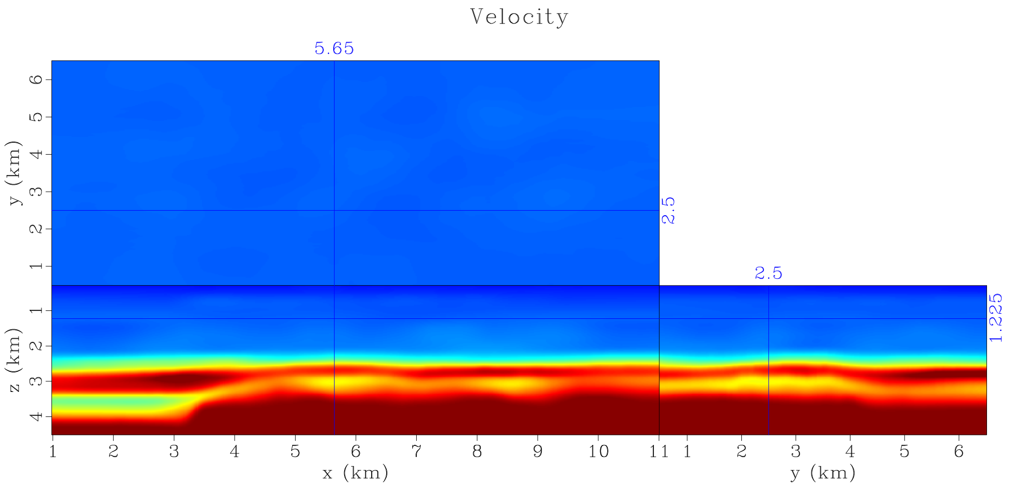
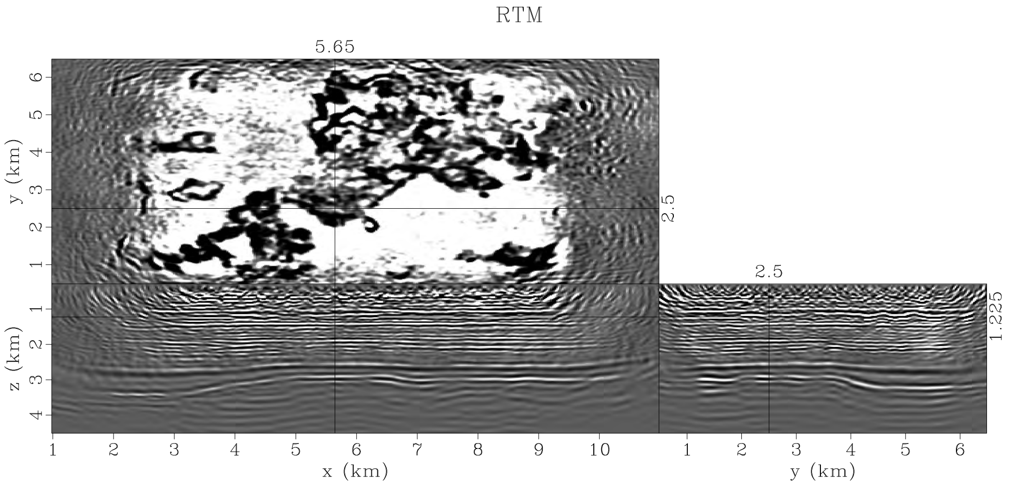


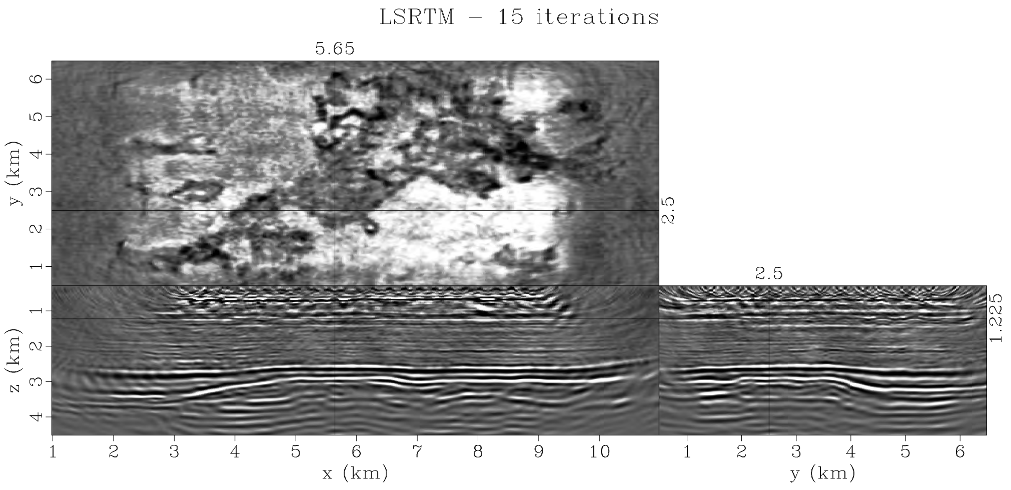
Figure 13. Bottom part of one receiver gather (t > 2 s). Top: input gather. Middle: Residual after 15 iterations of LSRTM. Bottom: Residual after 2 iterations of LSRTM with preconditioning. The arrows show events that are better matched after 2 iterations with preconditioning (marked as 1), than after 15 iterations without (marked as 2).



(a)

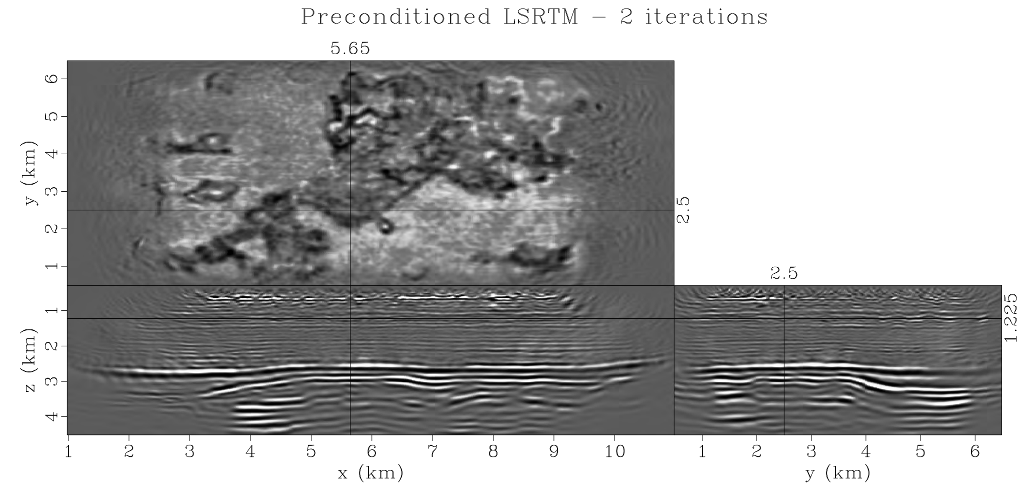


(b)

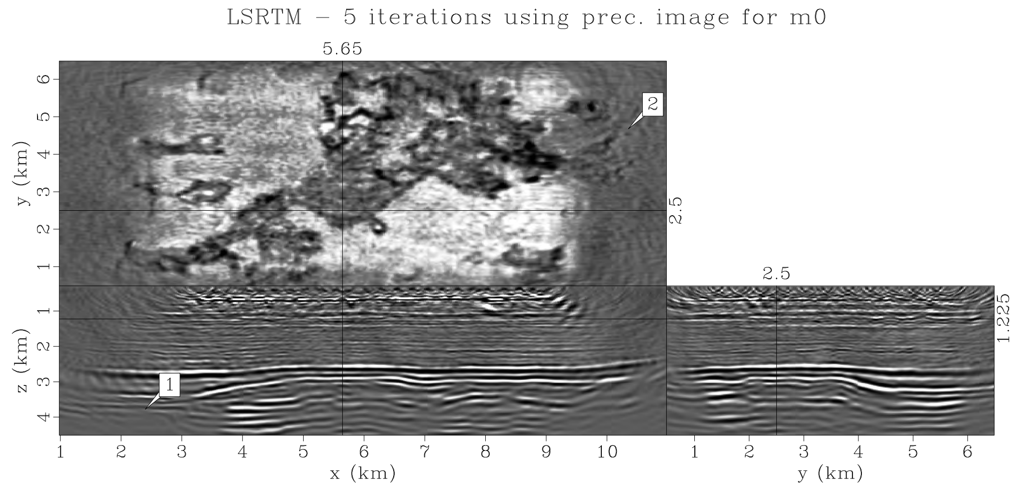


(c)

Figure 14. (a) Velocity model for the North Sea dataset. The chalk layer is clearly visible between z=2.5 km and z=4 km as a strong velocity contrast. (b) RTM and (c) LSRTM (15 iterations) images. The depth slices show the amplitude balancing effect of LSRTM. With LSRTM, artifacts are attenuated on the edges of the model and resolution is improved.



(a)



(b)

Figure 15. A comparison between (a) the LSRTM image with preconditioning after two iterations, and (b) the LSRTM image without preconditioning after five iterations using (a) as the starting model.

5 DISCUSSION

Overall, the methodology works very well: non-stationary matching filters provide a cheap and efficient way to approximate the inverse Hessian in 3D. The convergence of LSRTM is improved dramatically with a synthetic test case, and substantially with a field data example. Switching from preconditioned to un-preconditioned LSRTM seems to provide all the benefits of LSRTM for about half the cost.

Many interesting avenues of research are left untouched. First, effects of the wrong velocity on the filter estimation process and preconditioned LSRTM need to be investigated: clearly, the field data example tends to prove that noise in the data and velocity errors have effects that need to be better understood. Second, it would be very beneficial to use this technique with extended images where the cost of inversion is quite higher than with a zero subsurface-offset LSRTM (Hou and Symes, 2016). Recently, Khalil et al. (2016) showed an implementation of matching filters in the data space. A combination of both ideas (data and model space filters) might improve convergence even further. Finally, studying the influence of $\mathbf{m_1}$ on the inverse Hessian approximation could improve the method: starting from spikes, flat layers, etc..., might prove useful in building better filters.

There are implementation details that need to be worked on as well. The filter estimation process can be expensive. For shallow target, I observe that filters can be quite small. For deeper reflectors and not-so-well illuminated targets (like below the chalk layer in the North Sea dataset), bigger and more expensive filters are needed. A filter which size would change with position would help mitigating the computing costs.

One missing feature of the preconditioned LSRTM with matching filters is the increase of resolution. This should come at no surprise to us: because I use a convolution in equation (4) to match a lower resolution image $\mathbf{m_2}$ to a higher resolution image $\mathbf{m_1}$, we can't recover the high wavenumbers. In other words, when I apply the filters $\hat{\mathbf{a}}$ to the gradient in Algorithm 2, I also bandpass it. Therefore, I steer the solution to a smoother version of the true model. To mitigate this lack of resolution improvement, I offer three solutions. First, I can stop using the filters, as done with the Volve dataset and iterate a few times to recover the high wavenumbers. Second, I can use different filter sizes for each LSRTM iteration. This would come at the extra cost of estimating more filter banks. Finally, I can add a regularization operator to the LSRTM objective function that will boost the high wavenumbers. This can be done using a sparsity-promoting functional such as the ℓ^1 norm or Cauchy function, for instance. All three solutions must be appraised not only on the quality of the final image, but also on the convergence improvement. My goal, after all, is to speed-up inversion, not to slow it down.

These issues set aside, one of the main benefits of the proposed approach is that it works for any imaging operator. Including anisotropic or elastic effects is trivial because the filter estimation process remains the same: most of the physical (and geometrical) aspects of wave propagation are encapsulated in the two images we are trying to match.

6 CONCLUSION

3D matching filters can approximate the inverse Hessian of the least-squares RTM problem. Without any inversion and by simply applying the filters to the RTM image, noticeable improvements in terms of illumination and artifacts reduction are obtained. Using these filters in an inversion scheme to precondition the gradient further improves the image and convergence properties. On a 3D synthetic example, the convergence speed-up is significant ($\sim \times 5$) while a field data example shows a more modest but still welcome improvement ($\sim \times 2$).

7 ACKNOWLEDGMENTS

I would like to thank the sponsors of the Center for Wave Phenomena for their financial support. I would also like to thank Marianne Houbiers at Statoil for help and feedback with the North-Sea dataset. I thank Statoil ASA and the Volve license partners ExxonMobil E&P Norway AS and Bayerngas Norge AS, for the release of the Volve data. The views expressed in this paper are the views of the author and do not necessarily reflect the views of Statoil ASA and the Volve field license partners.

REFERENCES

Aoki, N., and G. T. Schuster, 2009, Fast least-squares migration with a deblurring filter: Geophysics, **74**, WCA83–WCA93.

Claerbout, J., and S. Fomel, 2014, Geophysical Imake Estimation by Example: http://sepwww.stanford.edu/data/media/public/sep//prof/gee/book-sep.pdf.

Clapp, M. L., 2005, Imaging under salt: Illumination compensation by regularized inversion: PhD thesis, Stanford University.

Guitton, A., 2004, Amplitude and kinematic corrections of migrated images for nonunitary imaging operators: Geophysics, **69**, 1017–1024.

Hou, J., and W. W. Symes, 2015, An approximate inverse to the extended born modeling operator: Geophysics, **80**, R331–R349.

———, 2016, Accelerating extended least-squares migration with weighted conjugate gradient iteration: Geophysics, **81**, S165–S179.

Khalil, A., H. Hoeber, G. Roberts, and F. Perrone, 2016, An alternative to least-squares imaging using data-domain matching filters: SEG Technical Program Expanded Abstracts, 4188–4192.

Kuhl, H., and M. D. Sacchi, 2003, Least-squares wave-equation migration for AVP/AVA inversion: Geophysics, **68**, 262–273.

Margrave, G. F., 1998, Theory of nonstationary linear filtering in the fourier domain with application to timevariant filtering: Geophysics, **63**, 244–259.

Nemeth, T., C. Wu, and G. T. Schuster, 1999, Least-squares migration of incomplete reflection data: Geophysics, **64**, 208–221.

Rickett, J. E., 2003, Illumination-based normalization for wave-equation depth migration: Geophysics, **68**, 1371–1379.

Szydlik, T., P. Smith, S. Way, L. Aamodt, and C. Friedrich, 2007, 3D PP/PS prestack depth migration on the volve field: First Break, **25**, 43–47.

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

Wang, P., A. Gomes, Z. Zhang, and M. Wang, 2016, Least-squares rtm: Reality and possibilities for subsalt imaging: SEG Technical Program Expanded Abstracts, 4204–4209.

Zhang, Y., A. Ratcliffe, G. Roberts, and L. Duan, 2014, Amplitude-preserving reverse time migration: From reflectivity to velocity and impedance inversion: Geophysics, **79**, S271–S283.