Chapter 2

Level sets and shape optimization

In the previous chapter I reviewed the problems with modeling salt and how level sets are an ideal tool for addressing some of those challenges. In this chapter I explain the basics of level sets and how they apply to the shape optimization problem of finding a salt boundary that minimizes the FWI objective function. I derive the gradient and demonstrate updating on a simple 2D model. Last, I derive the Hessian equations and demonstrate updating on 2D models with a comparison between the full Hessian and the Gauss-Newton Hessian approaches.

LEVEL SETS AS A TOOL

Basic definitions

A level set is a contour of a higher dimensional surface, ϕ. One can use the level set as an elegant tool for keeping track of boundaries as they change form. If one considers a 2D level set being the contour of a 3D implicit surface, then the spatial domain of the 2D level set can be defined as $\Theta \subset \mathbf{R^2}$ with elements $\chi \in \Theta$. A salt body Ω is simply a subset of the 2D domain ($\Omega \subset \Theta$) such that:

$$\Omega = \{\chi \mid \phi(\chi, \tau) > 0\},$$

where τ indicates the 'time' axis along which the evolution steps progress ($\tau = 0$ is the initial iteration). As such, for a single updating step (along τ), our current salt body Ω evolves to Ω'. I further define the boundary of the salt body as Γ (see Figure

2.1), such that:

$$\Gamma = \{\chi \mid \phi(\chi, \tau) = 0\}.$$

I define a point along the boundary curve to be:

$$\chi_\Gamma \in \Gamma.$$

With this definition of the boundary points, the level set of ϕ that represents the salt body boundary can be described as:

$$\phi(\chi_\Gamma, \tau) = 0.$$

While this derivation uses a 2D salt model for simplicity, a 3D salt body would instead have a four-dimensional implicit surface.

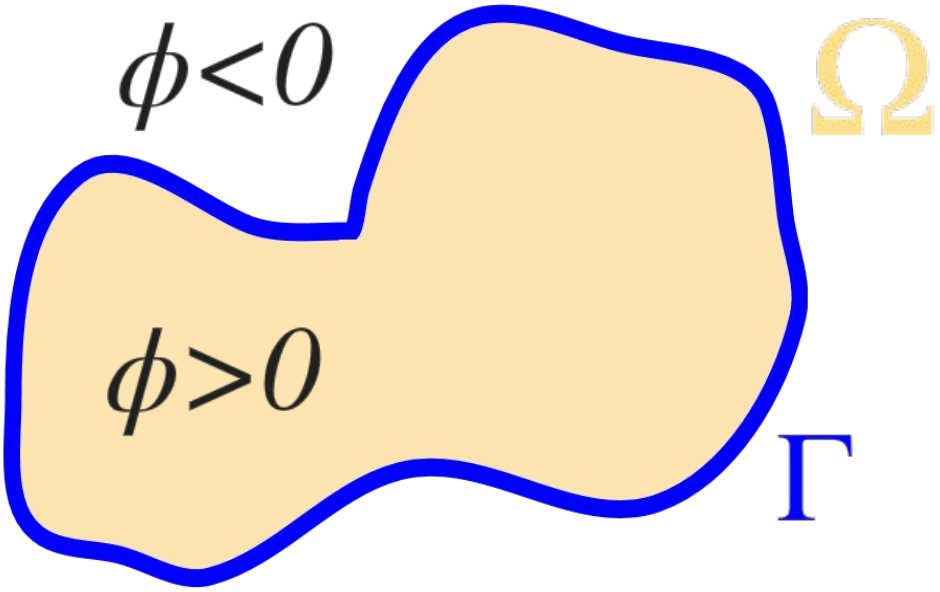


Figure 2.1: Diagram explaining the salt body domain (Ω), the ϕ values inside and outside of the salt, and the salt boundary (Γ). [**NR**] chapter2/. levelset-domain

Shape optimization derivation

With a clear understanding of level sets being established, the first step of the shape optimization derivation is to define the objective function we wish to minimize. I choose a variation of the FWI objective function

$$\psi(\mathbf{p}) = \frac{1}{2}||\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}||_2^2, \tag{2.1}$$

Here $\mathbf{F}$ is the forward acoustic wave propagator which includes a source function, wave propagation, and sampling based on a receiver geometry. Our seismic velocity model is $\mathbf{m}$, and $\mathbf{d_{obs}}$ is the observed pressure data. I deviate from the traditional FWI objective function by parameterizing $\mathbf{m}$ in terms of $\mathbf{p}$. Here I define $\mathbf{p} = [\boldsymbol{\phi} \quad \mathbf{b}]^T$, with $\boldsymbol{\phi}$ as the implicit surface function and $\mathbf{b}$ as the background velocity function, both varying across the full spatial domain Θ. In this section, I intend to take the derivative of equation 2.1 with respect to the underlying parameters $\mathbf{b}$ and $\boldsymbol{\phi}$. Since the chain rule applies, I first take the derivative with respect to $\mathbf{m}$ and then $\mathbf{p}$. I start by expanding our definition of the objective function:

$$\psi(\mathbf{p}) = \frac{1}{2}||\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}||_2^2 \tag{2.2}$$

$$= \frac{1}{2}(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}})^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.3}$$

$$= \frac{1}{2}(\mathbf{F}(\mathbf{m}(\mathbf{p}))^T\mathbf{F}(\mathbf{m}(\mathbf{p})) - 2\mathbf{F}(\mathbf{m}(\mathbf{p}))^T\mathbf{d_{obs}} + \mathbf{d_{obs}}^T\mathbf{d_{obs}}), \tag{2.4}$$

I then take the first derivative:

$$\frac{d\psi}{d\mathbf{p}} = \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T \mathbf{F}(\mathbf{m}(\mathbf{p})) - \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T \mathbf{d_{obs}} \tag{2.5}$$

$$= \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T (\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.6}$$

$$= \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}). \tag{2.7}$$

Here one recognizes $\frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T$ as the familiar adjoint Born operator ($\mathbf{B}^T$) used in standard FWI (Virieux and Operto (2009)). However, we must continue to expand our gradient to be defined in terms of our underlying parameters $\mathbf{b}$ and $\boldsymbol{\phi}$ (contained in vector $\mathbf{p}$). This requires us to state our model space clearly:

$$\mathbf{m}(\mathbf{p}) = \mathbf{m}(\boldsymbol{\phi}, \mathbf{b}) = H(\boldsymbol{\phi})(\mathbf{c}_{\text{salt}} - \mathbf{b}) + \mathbf{b}, \tag{2.8}$$

where $H(\circ)$ is the Heaviside function. Because I assume a homogeneous salt body, $\mathbf{c}_{\text{salt}}$ is a constant salt-velocity vector, typically about 4500m/s. Figure 2.2 explains this equation as the sum of a salt overlay created from $\boldsymbol{\phi}$ with the background velocity model $\mathbf{b}$. Note that the background velocity (shown in Figure 2.2c) is defined even

in the region under the salt footprint. That way if the salt shrinks to a smaller size or otherwise retracts inward, the background velocity is already defined and does not need to be interpolated.

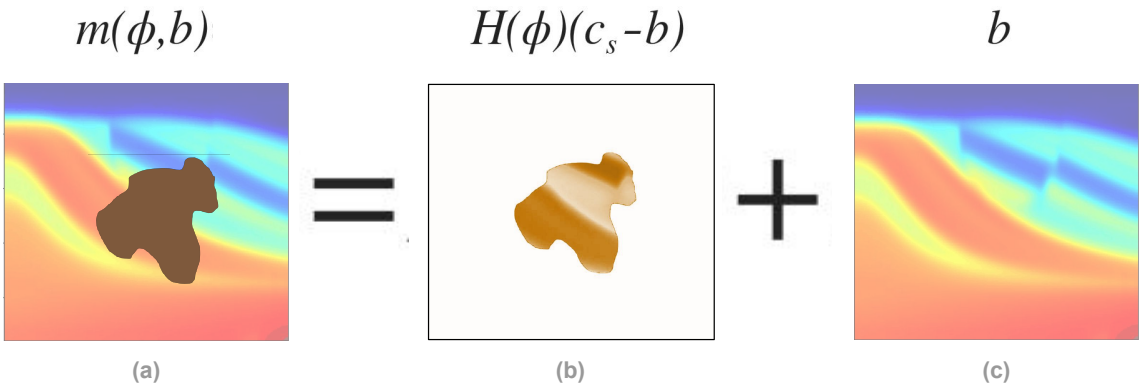


Figure 2.2: a) Full velocity model. b) Salt overlay. c) Background velocity. [**NR**] chapter2/. model-components

I expand the definition in equation 2.8 with a Taylor series as:

$$\mathbf{m_1} = \mathbf{m_0} + \frac{\partial \mathbf{m}}{\partial \boldsymbol{\phi}}\Big|_{\mathbf{m_0}} \triangle \boldsymbol{\phi} + \frac{\partial \mathbf{m}}{\partial \mathbf{b}}\Big|_{m_0} \triangle \boldsymbol{b} + ... \tag{2.9}$$

This approximation is only valid when the Taylor series converges with the addition of increasingly higher order terms. For the Heaviside function, this is not the case, since the function is not differentiable in its original form. For this reason, I use a smoothed approximation of the Heaviside function, such as:

$$\widetilde{H}(\phi) = \frac{1}{2}\left[1 + \frac{2}{\pi}\arctan(\frac{\pi\phi}{\epsilon})\right]. \tag{2.10}$$

An advantage of using equation 2.10 as a Heaviside approximation is that the derivative is non-zero everywhere. One downside is that the support of the function is infinite, and it will only approach $+1$ or 0 at $+\inf$ or $-\inf$. Further, the derivative of this function is very high near the zero crossing point. Kadu et al. (2017b) introduces an alternate approximation that has compact support in the region surrounding the zero-crossing:

$$\widetilde{H}_\epsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\epsilon, \\ \frac{1}{2}\left[1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right] & \text{if } -\epsilon \leq \phi \leq \epsilon, \\ 1 & \text{if } \phi > \epsilon. \end{cases} \tag{2.11}$$

Besides being able to capture +1 and 0 values exactly, the derivative of this approximation varies much less (see curves in Figure 2.3) and better balances weighting to the full boundary region when applying level set updates. By substituting this formulation of the Heaviside function in equation 2.8, I can now truncate the series in equation 2.9 and ignore higher order terms. This creates a linear approximation for the perturbation of the velocity model $\mathbf{m}$ with respect to $\boldsymbol{\phi}$ and $\mathbf{b}$:

$$\triangle \boldsymbol{m} \approx \frac{\partial \mathbf{m}(\boldsymbol{\phi}_o, \mathbf{b}_o)}{\partial \boldsymbol{\phi}} \triangle \boldsymbol{\phi} + \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \boldsymbol{b_o})}{\partial \mathbf{b}} \triangle \boldsymbol{b}. \tag{2.12}$$

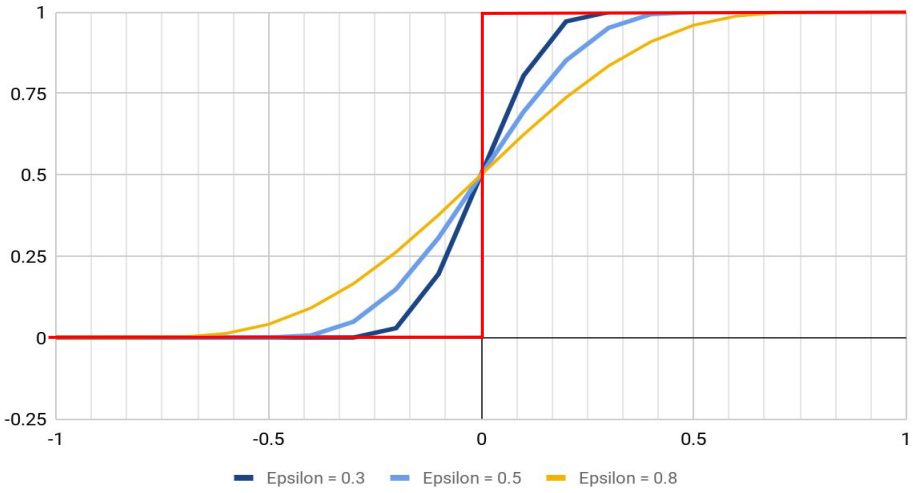


Figure 2.3: Curves of Heaviside approximations based on equation 2.11 for varying values of ϵ, and true Heaviside function (red). Note that within $\{-\epsilon, \epsilon\}$, the slope of each curve (its derivative) is relatively constant. [**ER**] chapter2/. heaviside-approx-chart

This can be written as a matrix operation:

$$\triangle \boldsymbol{m} \approx \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \begin{bmatrix} \triangle \boldsymbol{\phi} \\ \triangle \boldsymbol{b} \end{bmatrix}$$
$$\triangle \boldsymbol{m} \approx \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \triangle \boldsymbol{p},$$

where I define operator $\mathbf{D}$ as:

$$\begin{aligned} \mathbf{D} &= \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{\delta}(\boldsymbol{\phi_o})(\mathbf{c}_{\text{salt}} - \mathbf{b_o}) & \mathbf{I} - \widetilde{H}_\epsilon(\boldsymbol{\phi_o}) \end{bmatrix}. \end{aligned} \tag{2.13}$$

Here, $\widetilde{H_\epsilon}$ is the Heaviside approximation (equation 2.11), $\mathbf{I}$ is the identity matrix, $\widetilde{\delta}$ is the derivative of $\widetilde{H_\epsilon}$, $\mathbf{b}$ is the background velocity ($\mathbf{b_0}$ is fixed), $\boldsymbol{\phi}$ is the implicit surface ($\boldsymbol{\phi_0}$ is fixed), and $\mathbf{c}_{salt}$ is the constant salt velocity. Its functional form, $\widetilde{\delta}(\cdot)$ approximates an impulse function at $\phi = 0$, and its application acts as a selector for the salt boundary. Because of this, the operator $\mathbf{D}$ ultimately scales and masks the parameter fields $\triangle\boldsymbol{\phi}$ and $\triangle\boldsymbol{b}$.

This new approximation of the perturbation can be combined in our velocity model with equation 2.7 to get:

$$\frac{d\psi}{d\mathbf{p}} = \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.14}$$
$$\approx \mathbf{D}^T\mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.15}$$
$$\approx \mathbf{D}^T\mathbf{B}^T \triangle \boldsymbol{d}. \tag{2.16}$$

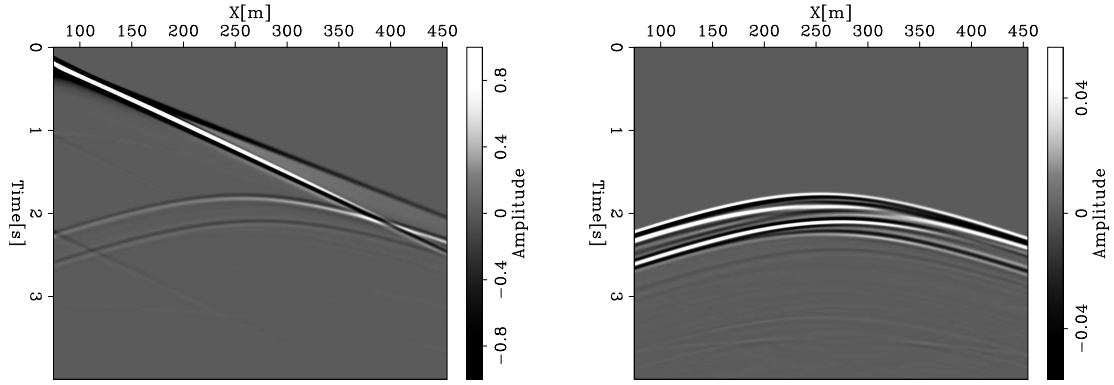
SIMPLE 2D SYNTHETIC MODEL DEMONSTRATION

In order to demonstrate first order (gradient) updating on ϕ (assuming $\mathbf{b}$ is constant for simplicity), I apply the inversion workflow shown in Algorithm 1 on a 2D circular salt model example (Figure 2.8(a)), with an outward normal perturbation (the initial salt is larger than the true model). This creates an error in the modeled data (Figure 2.4(a)) and thus a data space residual (Figure 2.4(b)). Note that these figures show

the top and bottom salt events between about 1.7-2.5 seconds.

Algorithm 1 Steepest descent updating algorithm

procedure LEVELSETINVERSION-ORDER1(d_{obs},ϕ_0)
  for i in (1,N) **do**
    $d_{\text{syn}}(i) = \text{F}(\phi_{i-1})$
    $\text{residual}(i) = d_{\text{obs}} - d_{\text{syn}}(i)$
    $\text{gradient}(i) = D^T B^T \text{residual}(i)$
    $\alpha = \text{linesearch}(\text{gradient}(i))$
    $\phi_i = \phi_{i-1} - \alpha \cdot \text{gradient}(i)$
  end for
  Return $m(\phi_N)$
end procedure



(a) (b)

Figure 2.4: a) Observed data. b) Data residuals. Observed data and residuals are a single shot gather created from the small initial salt example (Figure 2.12(a)). [**ER**] chapter2/. obsdata-sample-small,residuals-sample-small

I calculate the search direction for $\boldsymbol{\phi}$ (Figure 2.6) by back-propagating the data residual and cross-correlating it with the source wavefield as per the Born approximation imaging condition (Figure 2.5), followed by applying the **D** operator from equation 2.13. In this case, the search direction is negative near the perturbation, since it wants to decrease what is ultimately a positive velocity error. This search direction pushes a decrease in the value of the implicit surface (compare Figures 2.7(a) and 2.7(b)). This decrease draws the zero-level set deeper so that it is in closer alignment with the true salt boundary. Applying the approximate Heaviside function (equation 2.11) to this updated implicit surface (Figure 2.7(b)) gives us a new model that is closer in form to the true model (Figure 2.8(b)). This process can be continued, making iterative updates to the implicit surface and as a result, the velocity

model itself.

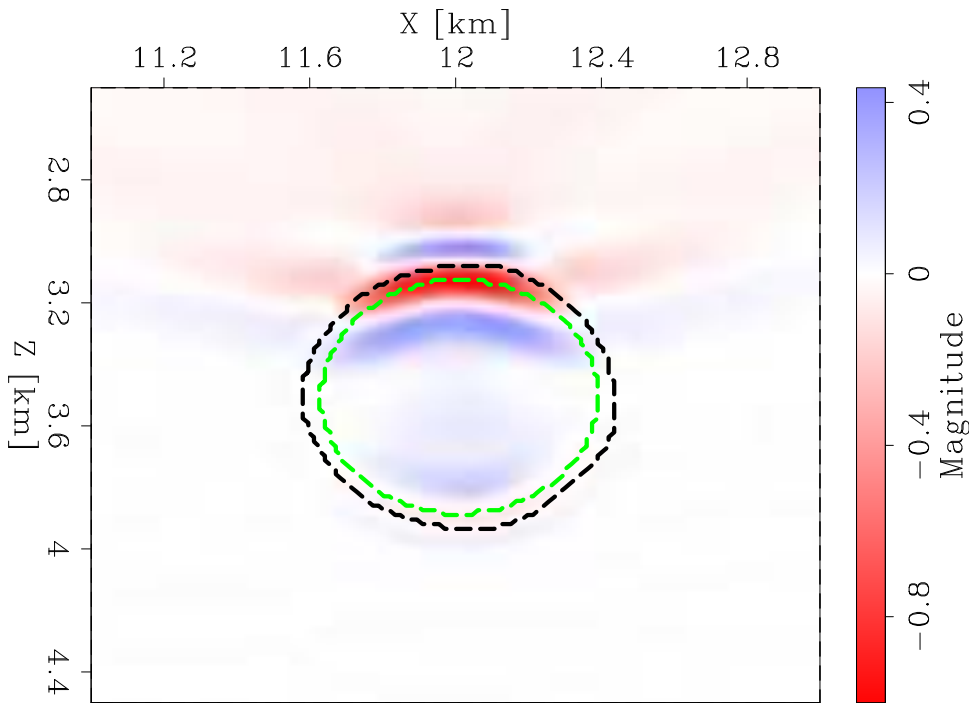


Figure 2.5: Search direction from adjoint Born image of back-propagated residuals used in conventional FWI. True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. rtmGrad0-big

For an initial model where the salt circle is too small (Figure 2.12(a)), the adjoint Born component of the search direction (Figure 2.9) is positive, since it is trying to increase the model velocity to that of the salt. The full search direction for $\boldsymbol{\phi}$ gives a positive perturbation of the implicit surface (Figure 2.10). This update raises the implicit surface (compare Figures 2.11(a) and 2.11(b)), moving the salt boundary upwards to correct for the deep boundary discrepancy (Figure 2.12(b)).

INTRODUCING THE HESSIAN OPERATOR

We can use the Newton method (equation 2.17) to perform the inversion, which helps remove the effect of our operator from the gradient, improves the search direction, and subsequently speeds up convergence. This is done by inverting the Hessian ($\mathbf{H}$) of the objective function we are minimizing and applying it to the negative of our gradient ($\mathbf{g}$) to find the search direction $\triangle\mathbf{m}$:

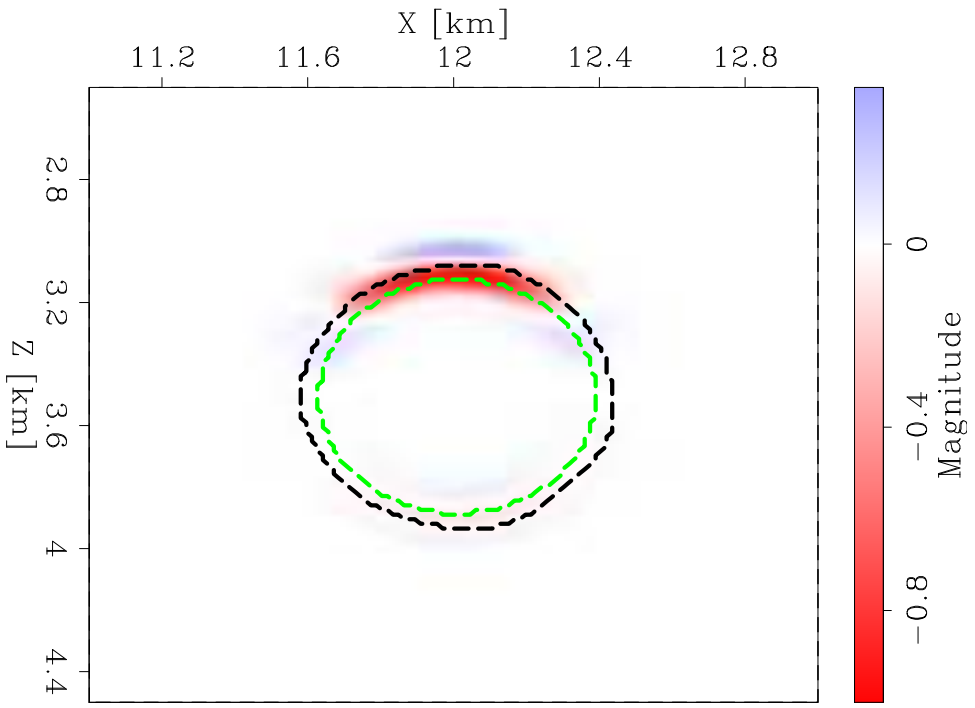


Figure 2.6: Implicit surface search direction ($\triangle\boldsymbol{\phi}$). True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. phiGrad0-big

$$\triangle \mathbf{m} = -\mathbf{H}^{-1}\mathbf{g}. \tag{2.17}$$

For our case, we can represent the Hessian as the second derivative (equation 2.18) of the FWI objective function (equation 2.1):

$$\begin{aligned} \frac{\delta \psi}{\delta \mathbf{m}} &= \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}^T}(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) \\ \frac{\delta}{\delta \mathbf{m}}\frac{\delta \psi}{\delta \mathbf{m}} &= \frac{\delta}{\delta \mathbf{m}}\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) \\ \frac{\delta^2 \psi}{\delta \mathbf{m}^2} &= \frac{d^2\mathbf{F}(\mathbf{m})}{d\mathbf{m}^2}^T(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) + \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}. \end{aligned} \tag{2.18}$$

However, this second derivative contains a term based on the data residuals. This term can be expensive to compute and is often neglected, so the Gauss-Newton approximation is used instead:

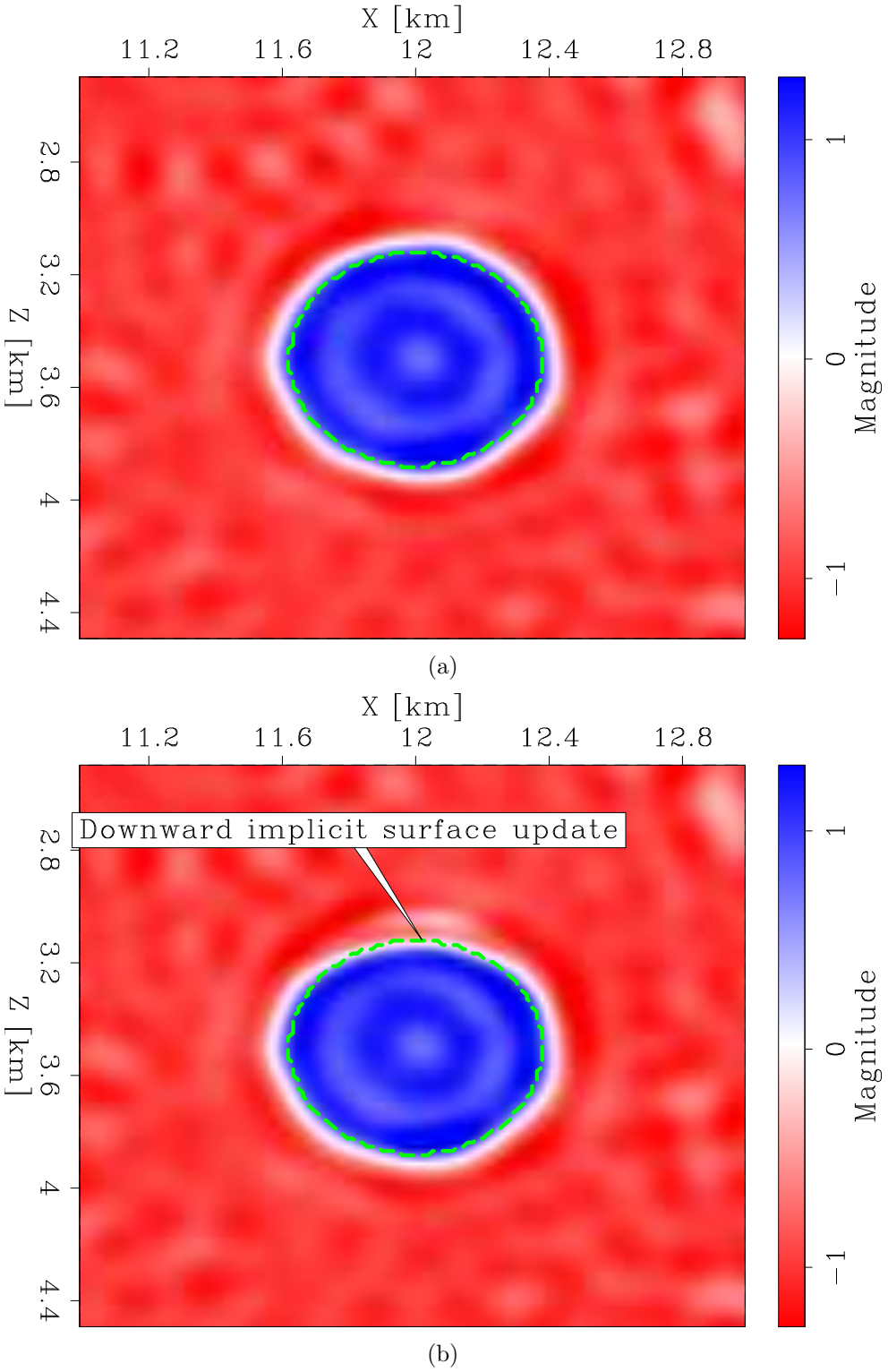


Figure 2.7: Implicit surface (ϕ) model a) before, and b) after updating with search direction (Figure 2.6). True salt extent shown as green dashed line. [**CR**] chapter2/. initphi-big,nextphi-big

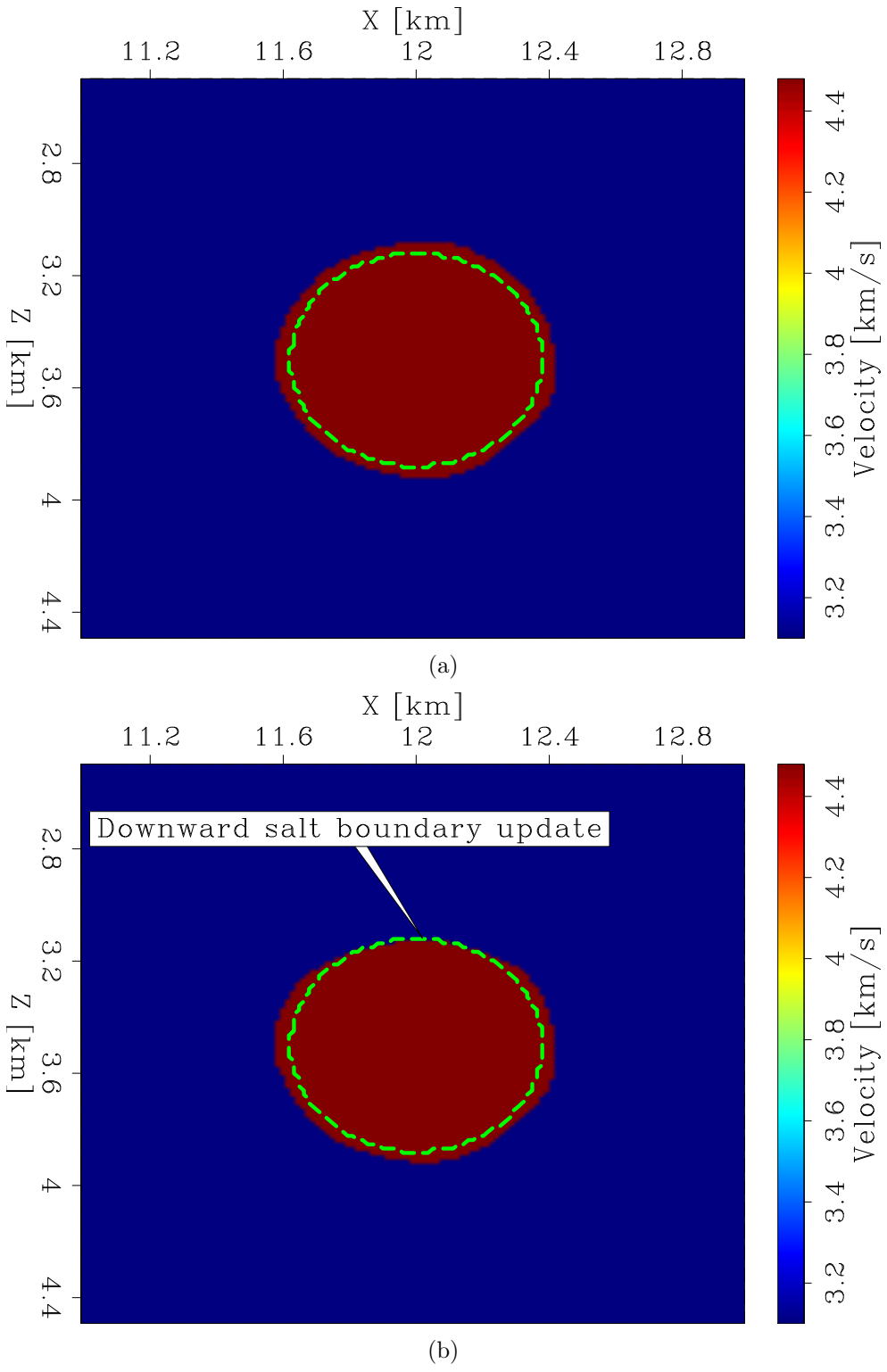


Figure 2.8: Velocity model (**m**) a) before, and b) after updating. True salt extent shown as green dashed line. [**CR**] chapter2/. initmodel-big,nextmodel-big

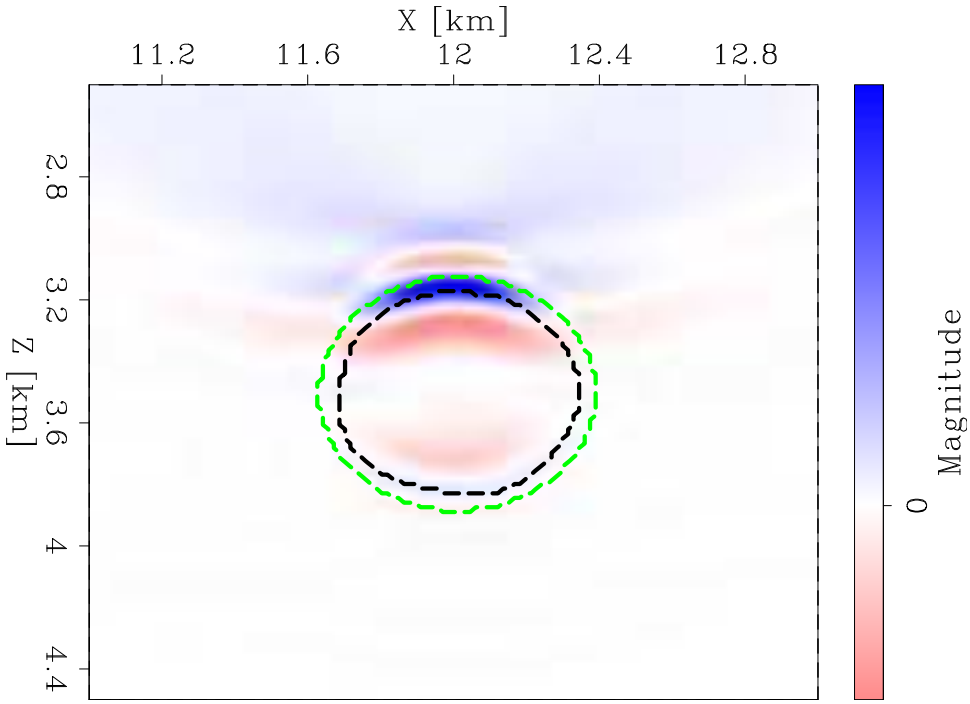


Figure 2.9: Search direction from adjoint Born image of back-propagated residuals used in conventional FWI. True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. rtmGrad0-small

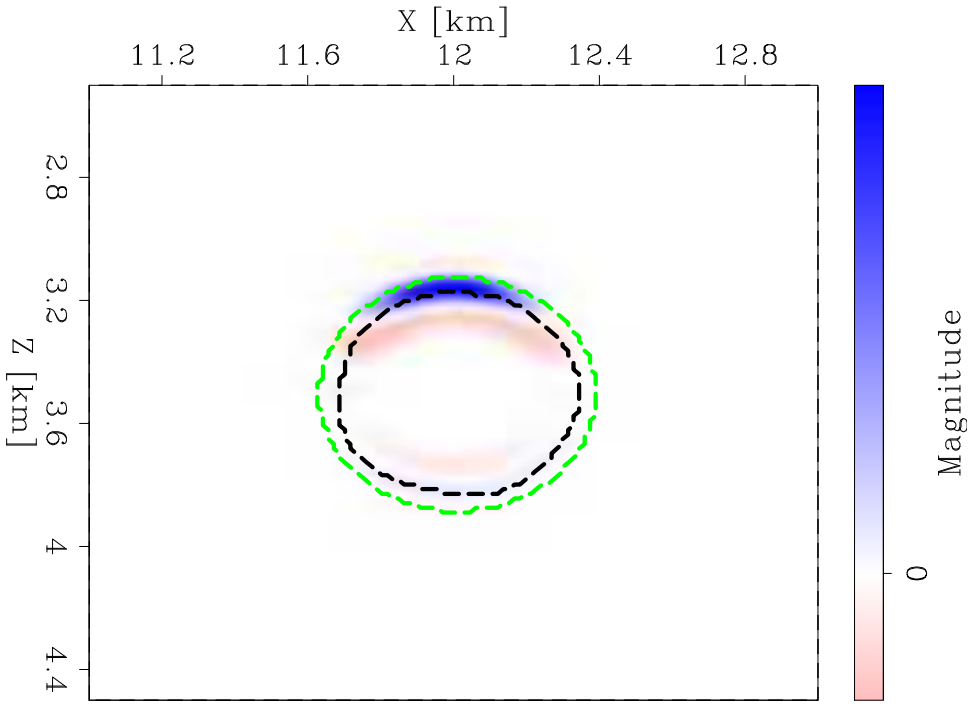


Figure 2.10: Implicit surface search direction ($\triangle\boldsymbol{\phi}$). True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. phiGrad0-small

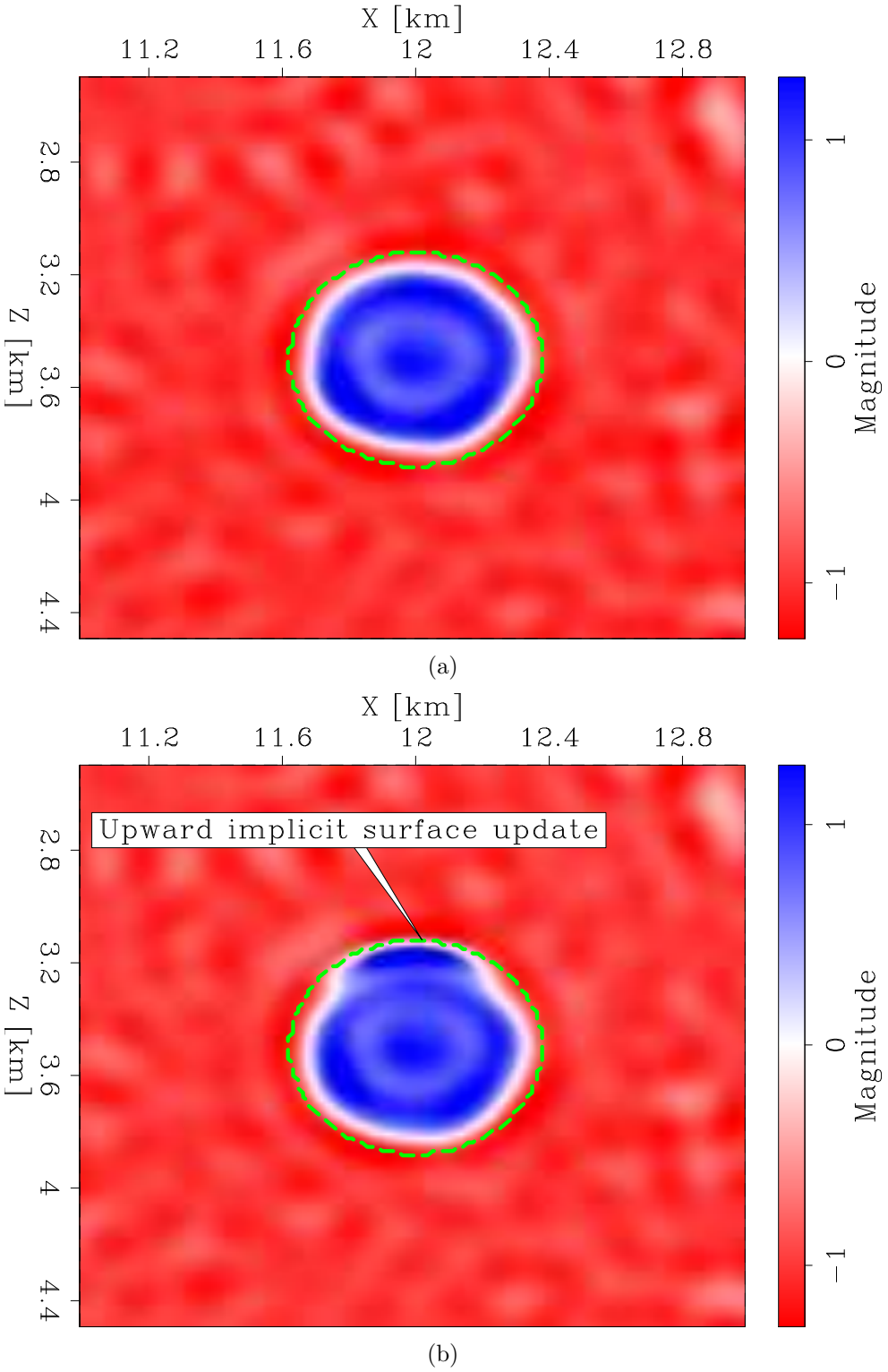


Figure 2.11: Implicit surface (ϕ) model a) before, and b) after updating with search direction (Figure 2.10). True salt extent shown as green dashed line. [**CR**] chapter2/. initphi-small,nextphi-small

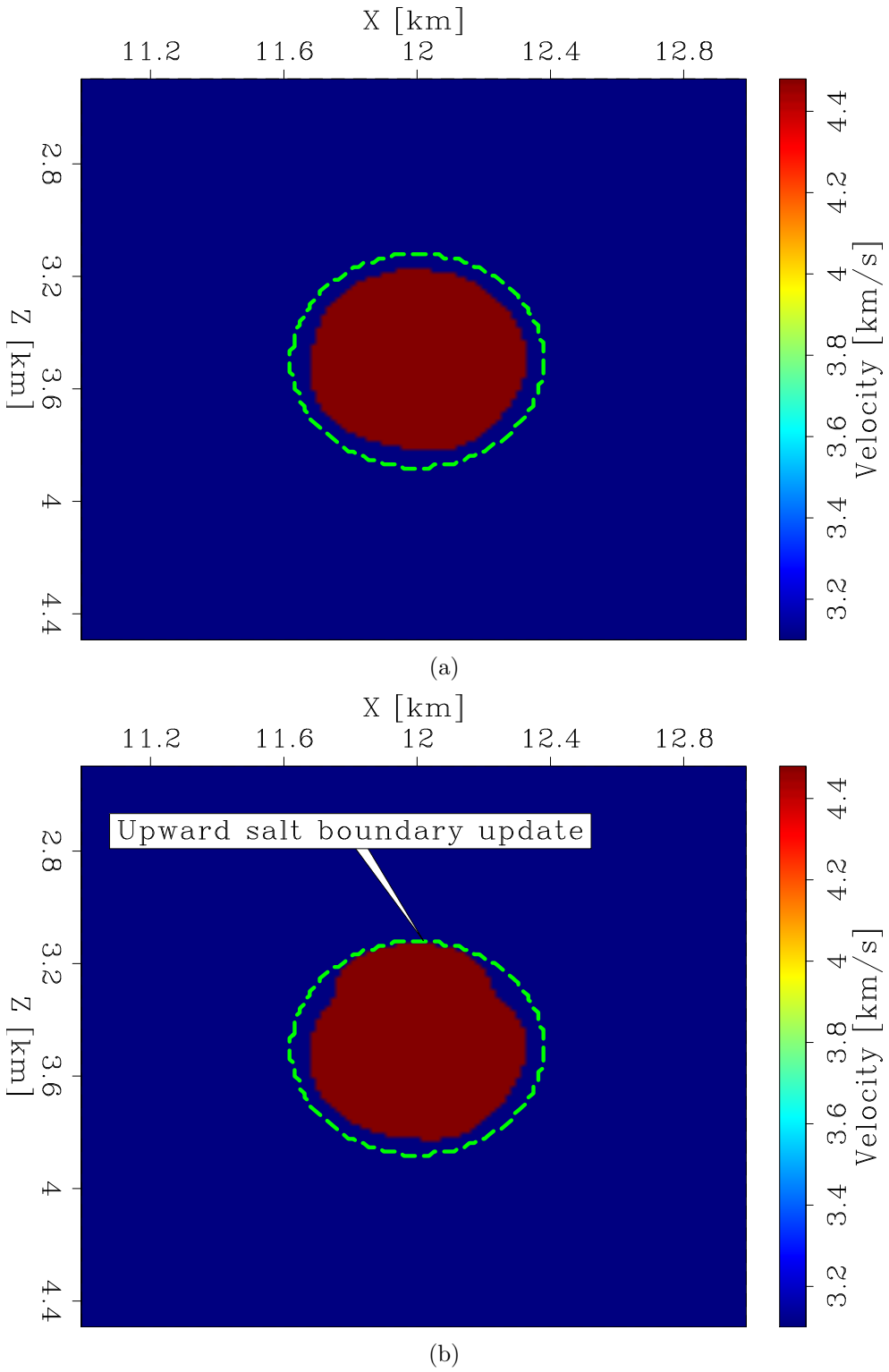


Figure 2.12: Velocity model (**m**) a) before, and b) after updating. True salt extent shown as green dashed line. [**CR**] chapter2/. initmodel-small,nextmodel-small

$$\frac{\delta^2\psi}{\delta\mathbf{m}^2} = \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}} + \frac{d^2\mathbf{F}(\mathbf{m})}{d\mathbf{m}^2}^T (\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}})$$
$$\mathbf{H_{GN}} = \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}} \tag{2.19}$$

While using this approximation is cheaper to compute, it is not as accurate. Fichtner (2010) and Biondi et al. (2015) show that the residual term of the FWI objective function Hessian can be found by computing WEMVA-like operations on the data-space residuals. By comparison, the Gauss-Newton approximation of the FWI objective function is simply the forward Born operator followed by the adjoint Born operator. An inherent limitation of the Born operator is that it only models first-order reflections, and so double-scattered energy such as waves that bounce inside canyons or salt bodies (Figure 2.13) can be spatially misplaced in the search direction when using the Gauss-Newton Hessian. One would expect to gain a better search direction for canyon and salt-type models by using the full Hessian instead. For this reason, I investigate using the full Hessian on 2D synthetic models.

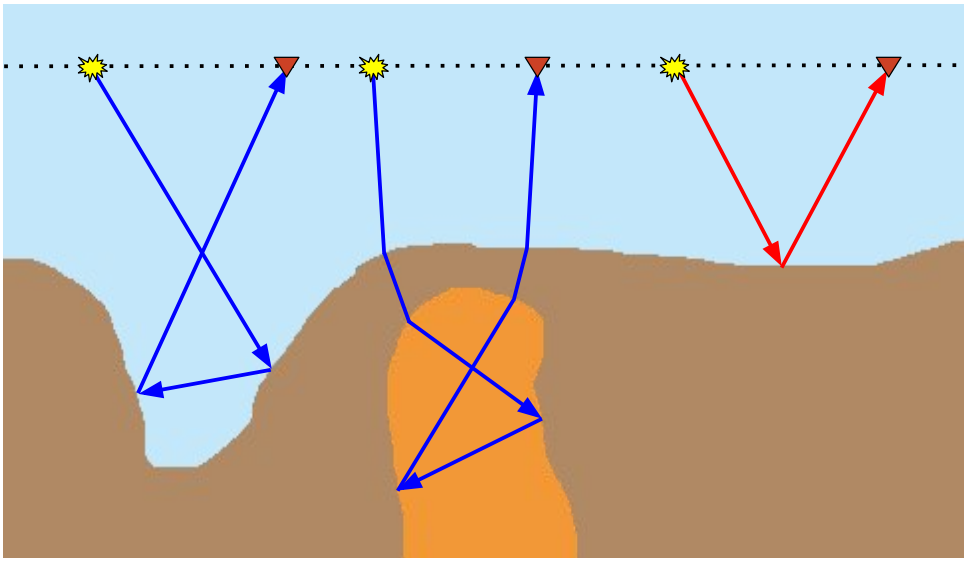


Figure 2.13: Second order reflection ray paths in a canyon and inside a salt body (blue). First order reflection off water bottom (red). [**NR**] chapter2/. reflection-order-diagram

HESSIAN INVERSION COMPARISONS

As discussed in the last section, the full Hessian is potentially better than the Gauss-Newton Hessian at finding a search direction for models that contain salt canyons. I test this hypothesis by creating a synthetic salt canyon true model, and then perturb it to create a starting model so that a true search direction is known. Then I run separate inversions with the full and Gauss Newton Hessians to compare search direction results. Furthermore, I first test by perturbing only one side of the canyon, and then test by perturbing both sides. This allows us to see how the starting velocity model affects the efficacy of the Gauss-Newton and full Hessians relative to one another.

Single canyon perturbation example

For the first example, I select an upper canyon portion of the synthetic Sigsbee model (Figure 2.14). I perturb the left hand side of the canyon (Figure 2.15) to create secondary scattering against the opposite canyon wall (shown in Figure 2.16). I use an evenly spaced acquisition geometry of 38 shots and 230 receivers, and perform modeling using absorbing boundary conditions and a Ricker wavelet with a central frequency of 15 Hz. For both the single and double perturbation cases, I assume that $\triangle \mathbf{b} = 0$, and so invert for a model defined as $\triangle \mathbf{p} = \triangle \boldsymbol{\phi}$ (inverting only for the implicit surface update, not the background velocity update).

Double canyon perturbation example

For the second example, I use the same true model as before (Figure 2.14). However, this time I perturb both the left and right hand sides of the canyon (Figure 2.17). This offers further complexity to the secondary scattering of the model (shown in Figure 2.18). The same acquisition geometry and wavelet were used.

Benefits of the full Hessian

When comparing the results of the Gauss-Newton (Figure 2.19) and the full Hessian results (Figure 2.20) from the double canyon perturbation model, one can see a slight improvement in the focusing of the energy in the full Hessian example. This improved

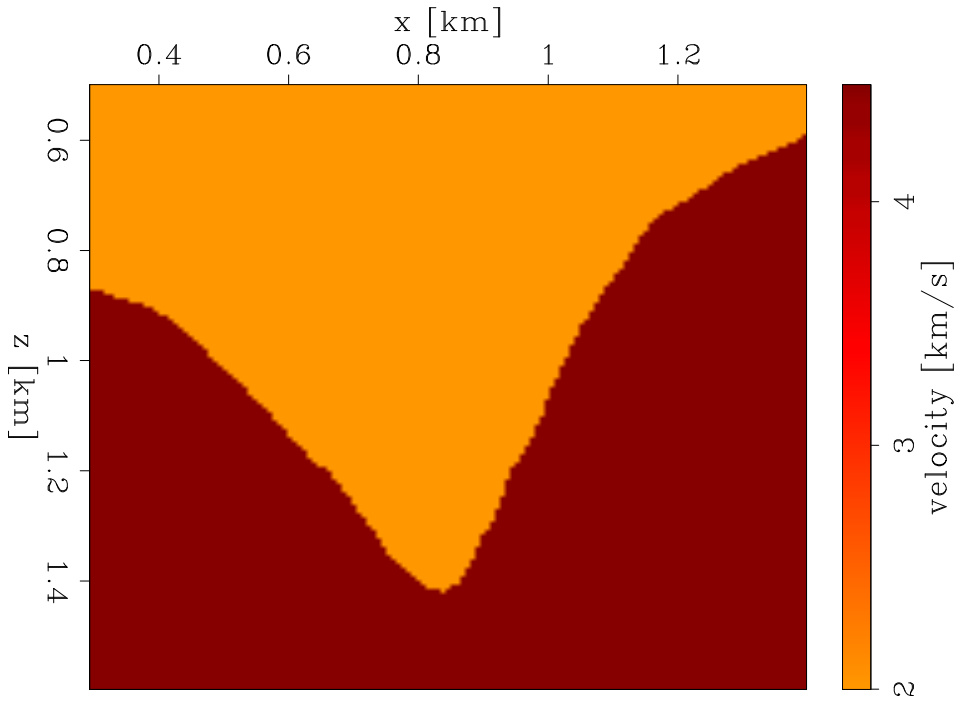


Figure 2.14: The canyon portion of the Sigsbee model that was used. [**ER**] chapter2/. single-guess

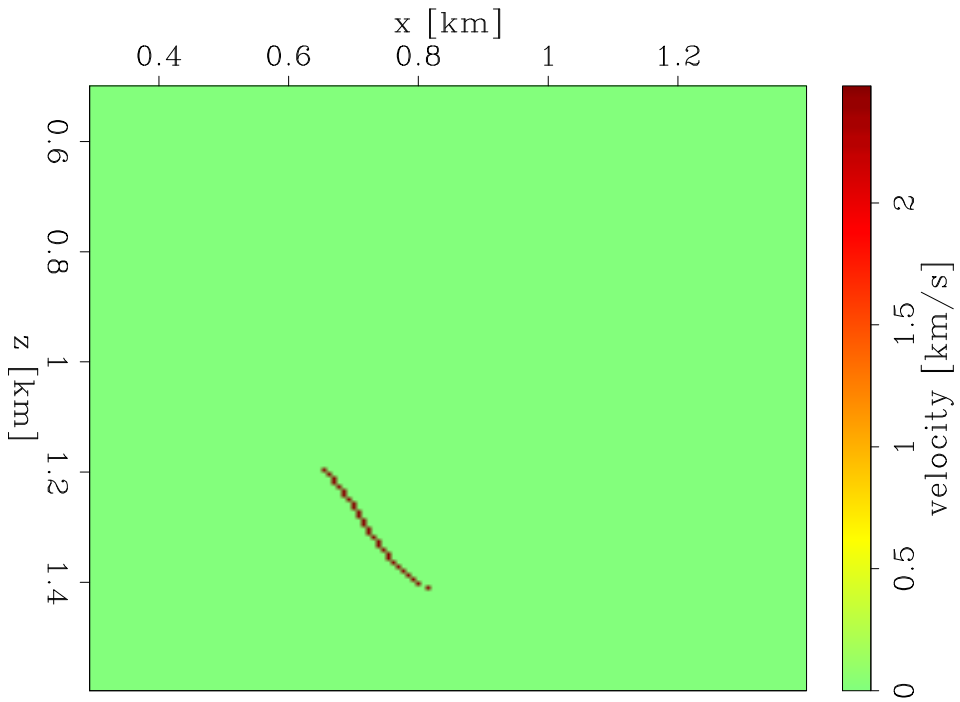


Figure 2.15: The single canyon perturbation ($\triangle \mathbf{m}_{\text{actual}}$) of the Sigsbee model. [**ER**] chapter2/. single-pert

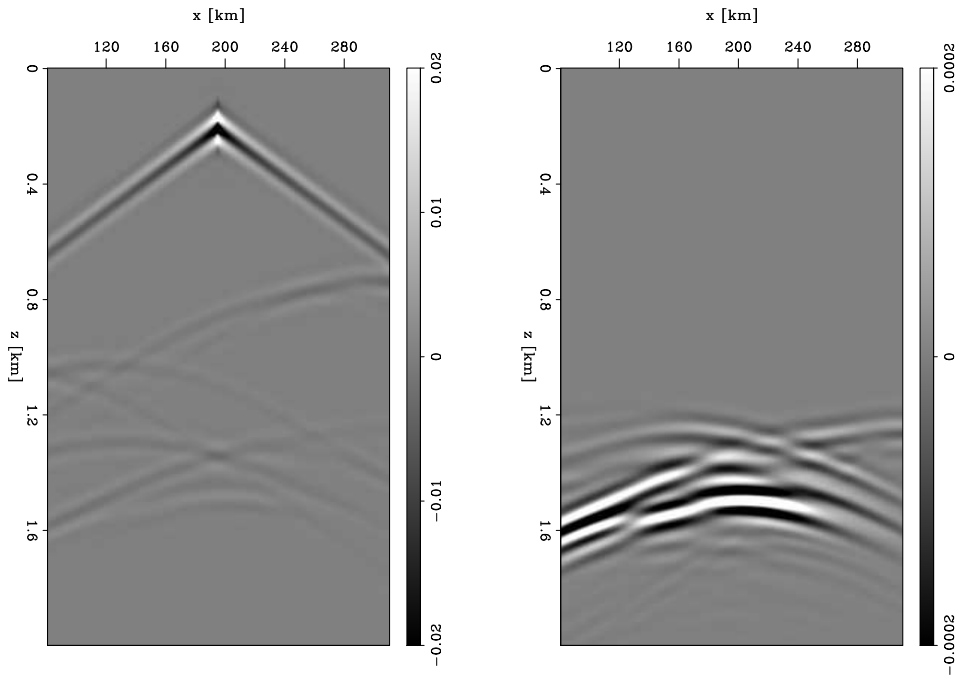


Figure 2.16: Data generated on the true model ($\mathbf{d_{obs}}$) from the center shot (left). The data residual ($\triangle\mathbf{d}$) for the same center shot (right) generated by differencing $\mathbf{d_{obs}}$ with data generated from the single perturbation model. [**CR**] chapter2/. centershot-analysis-single

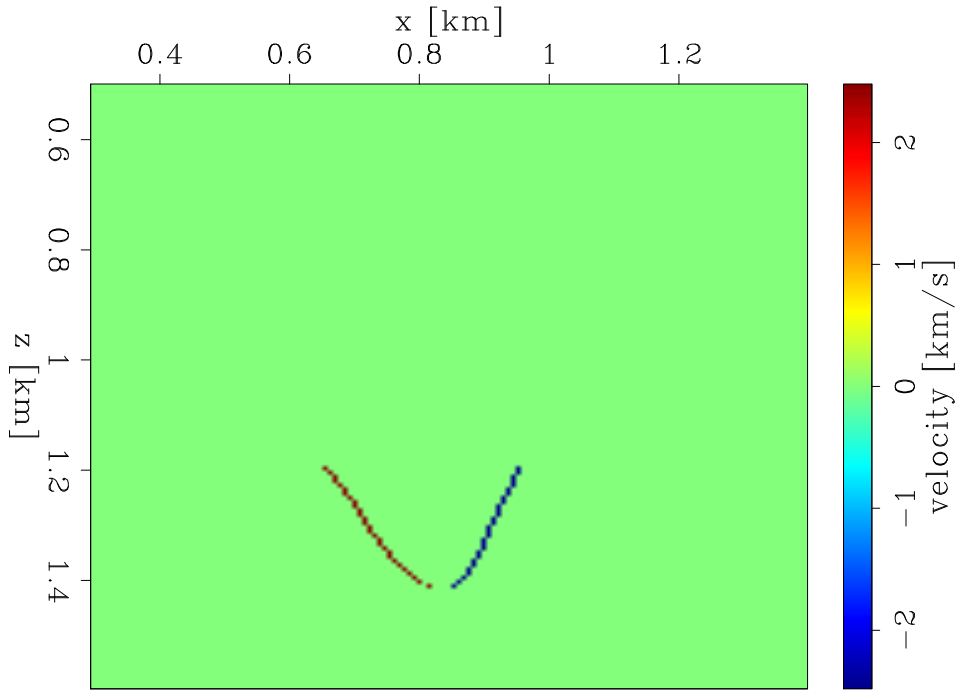


Figure 2.17: The double canyon perturbation ($\triangle \mathbf{m}_{\text{actual}}$) of the Sigsbee model. [**ER**] chapter2/. double-pert

search direction should lead to better convergence in the greater non-linear inversion scheme. When comparing against the steepest descent search direction for the double perturbation case (Figure 2.21), one can see that the search directions by either type of Hessian inversion create significant improvements. The improvement of the full Hessian versus the Gauss-Newton Hessian that is seen in the double perturbation case is minimal compared to the improvement that the Gauss-Newton Hessian has over the steepest descent direction (Figure 2.21). Similarly in the single perturbation case, the search direction from inverting the Gauss-Newton Hessian (Figure 2.22) is significantly better than the steepest descent search direction (Figure 2.23).

Limitations

However, the single perturbation case demonstrates that the advantages of the full Hessian are not realized for all models, since the Hessian is model dependent. The single perturbation example results are much different with regards to the full Hessian

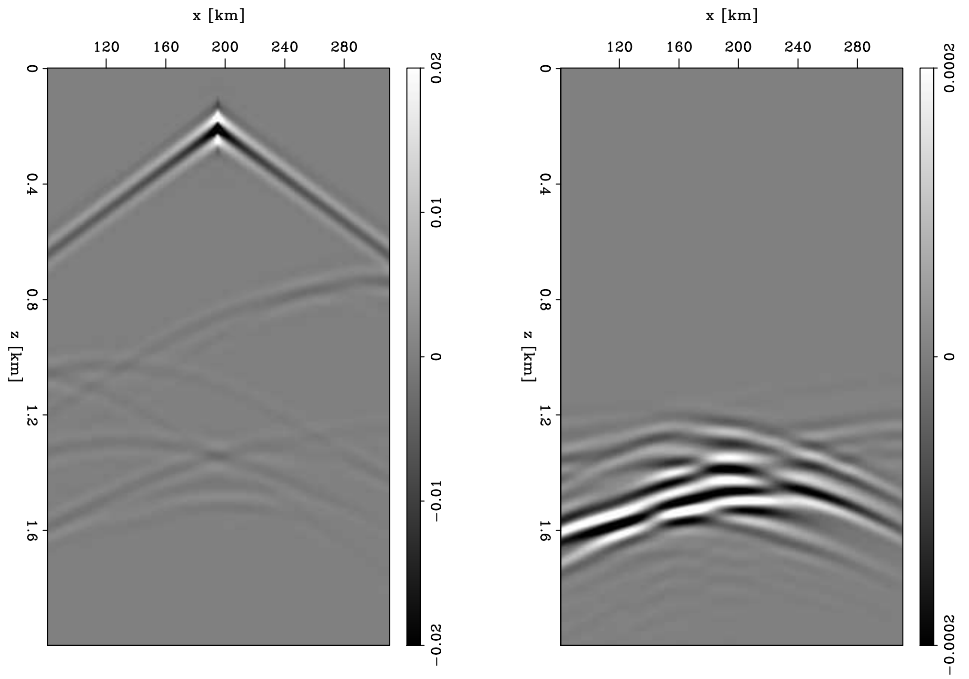


Figure 2.18: Data generated on the true model ($\mathbf{d_{obs}}$) from the center (left). The data residual ($\triangle\mathbf{d}$) for the same center shot (right) generated by differencing $\mathbf{d_{obs}}$ with data generated from the double perturbation model. [**CR**] chapter2/. centershot-analysis-double

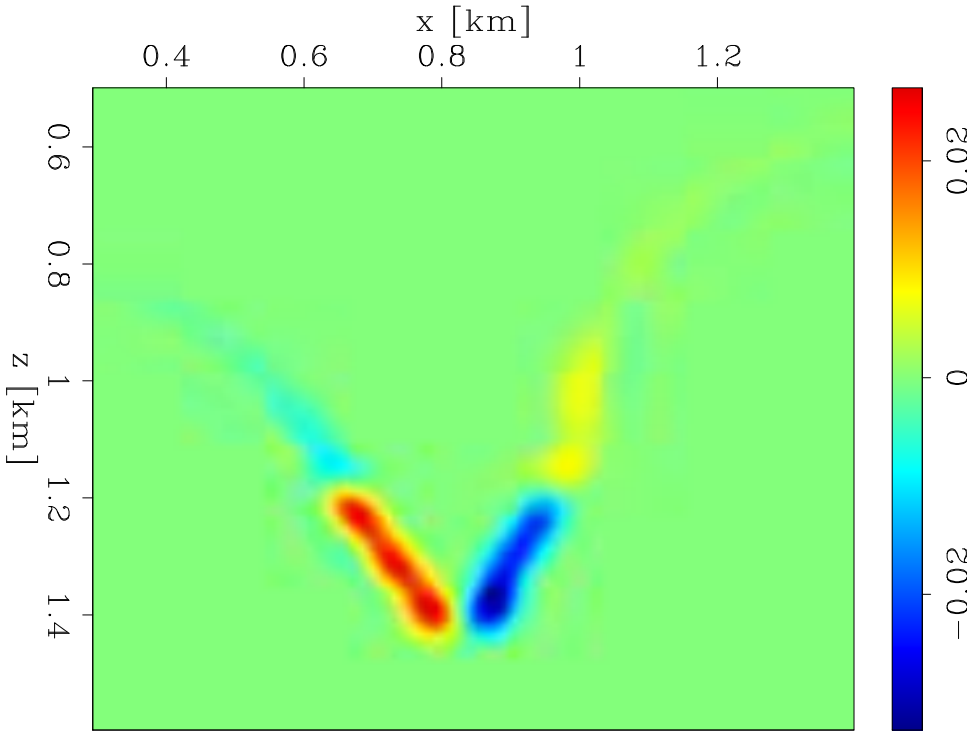


Figure 2.19: The Newton search direction ($\triangle \boldsymbol{\phi}$) using the inverted Gauss-Newton approximation of the Hessian on the double perturbation model. [**CR**] chapter2/. double-final-gn

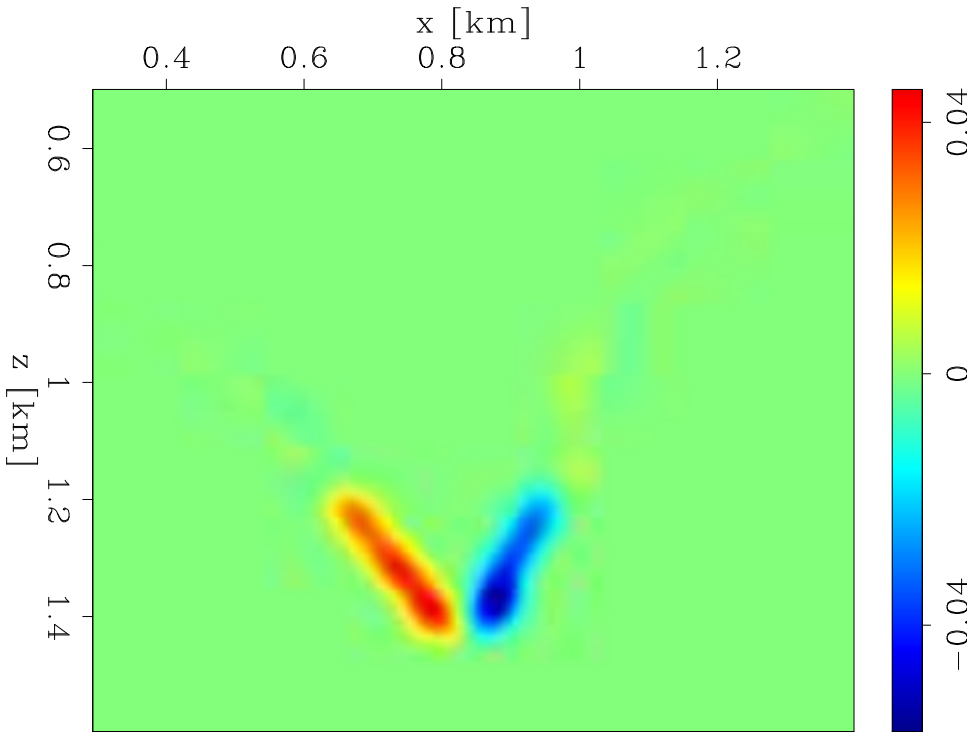


Figure 2.20: The Newton search direction ($\triangle\boldsymbol{\phi}$) using the inverted full Hessian on the double perturbation model. [**CR**] chapter2/. double-final-full

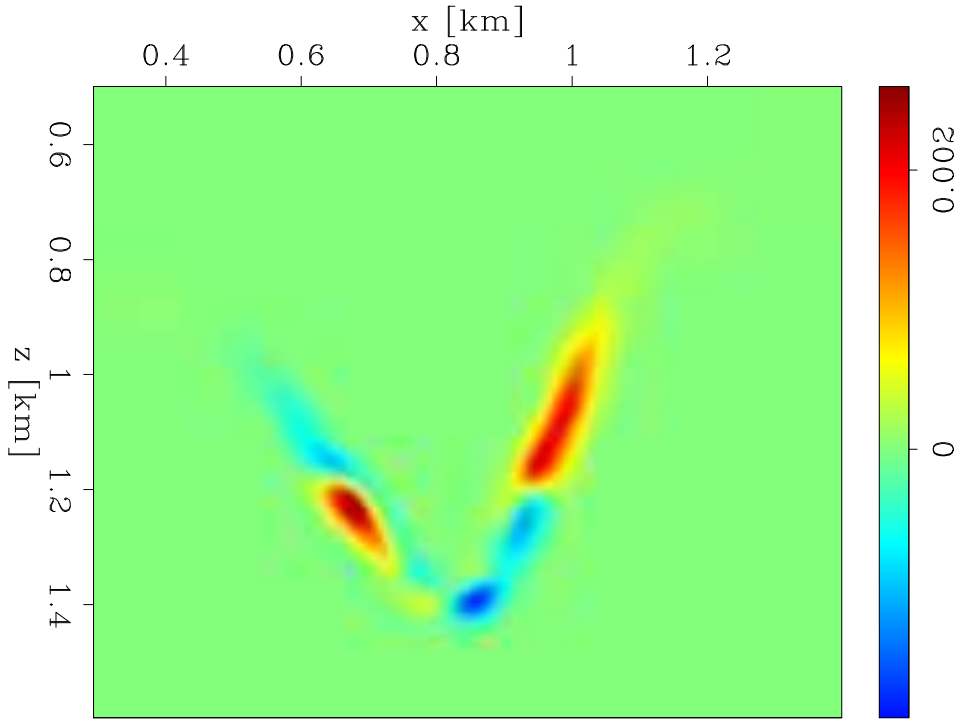


Figure 2.21: The steepest descent search direction ($\triangle\boldsymbol{\phi}$) (negative of the FWI gradient) for the double perturbation model. [**CR**] chapter2/. double-negative-gradient

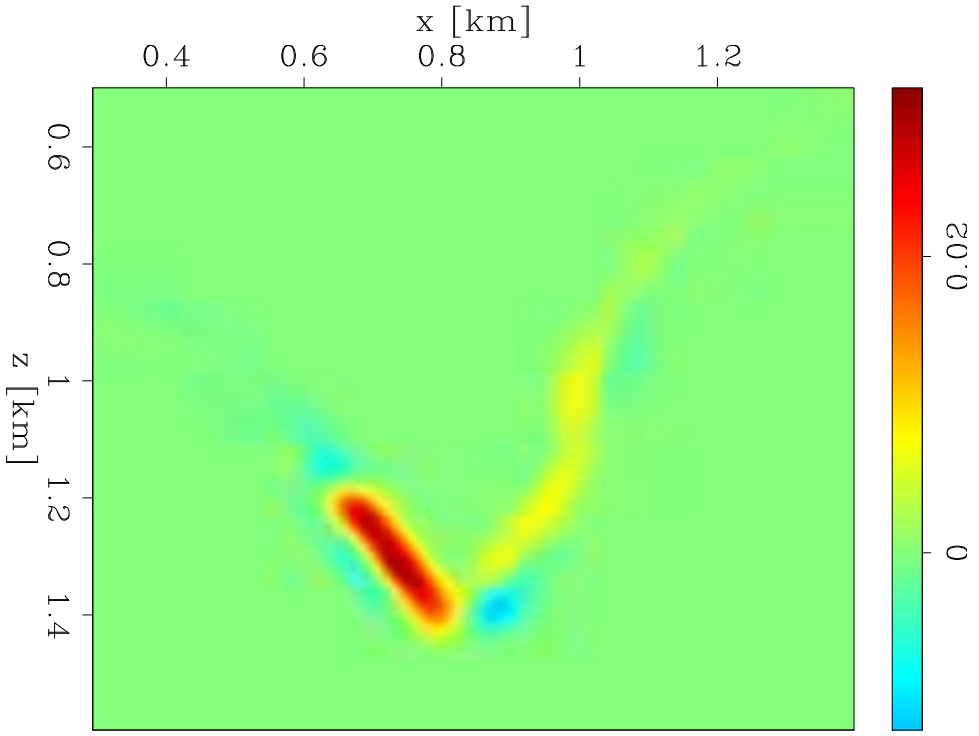


Figure 2.22: The Newton search direction ($\triangle \boldsymbol{\phi}$) using the inverted Gauss-Newton approximation of the Hessian on the single perturbation model. [**CR**] chapter2/. single-final-gn

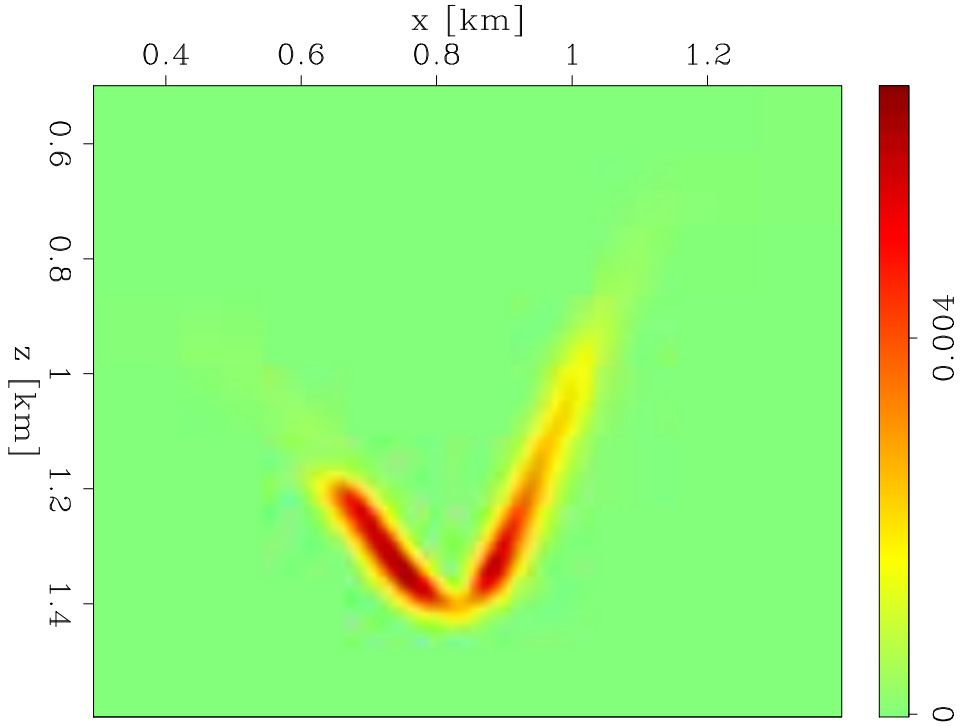


Figure 2.23: The steepest descent search direction ($\triangle\phi$) (negative of the FWI gradient) for the single perturbation model. [**CR**] chapter2/. single-negative-gradient

search direction. While the Gauss-Newton Hessian system inversion rapidly converges (Figures 2.22 and 2.25(a)), the full Hessian inversion becomes unstable part way through (Figures 2.24 and 2.25(b)). Because the full Hessian operator is not inherently positive semi-definite like the Gauss-Newton Hessian, it may have negative eigenvalues, which can lead to instability during inversion. This was very likely the case in the single canyon perturbation example. On the other hand, with the double perturbation example we get stable convergence using either method (compare Figures 2.26(a) and 2.26(b)).

One of the few feasible methods for enforcing our operator to be positive semi-definite is to use the Levenberg-Marquardt (1963) method of regularizing the operator with a scaled identity matrix:

$$\hat{\mathbf{H}}_{\mathbf{full}} = \mathbf{H}_{\mathbf{full}} + \alpha \mathbf{I} \tag{2.20}$$

However, in order to use this method properly, the correct scaling of the identity

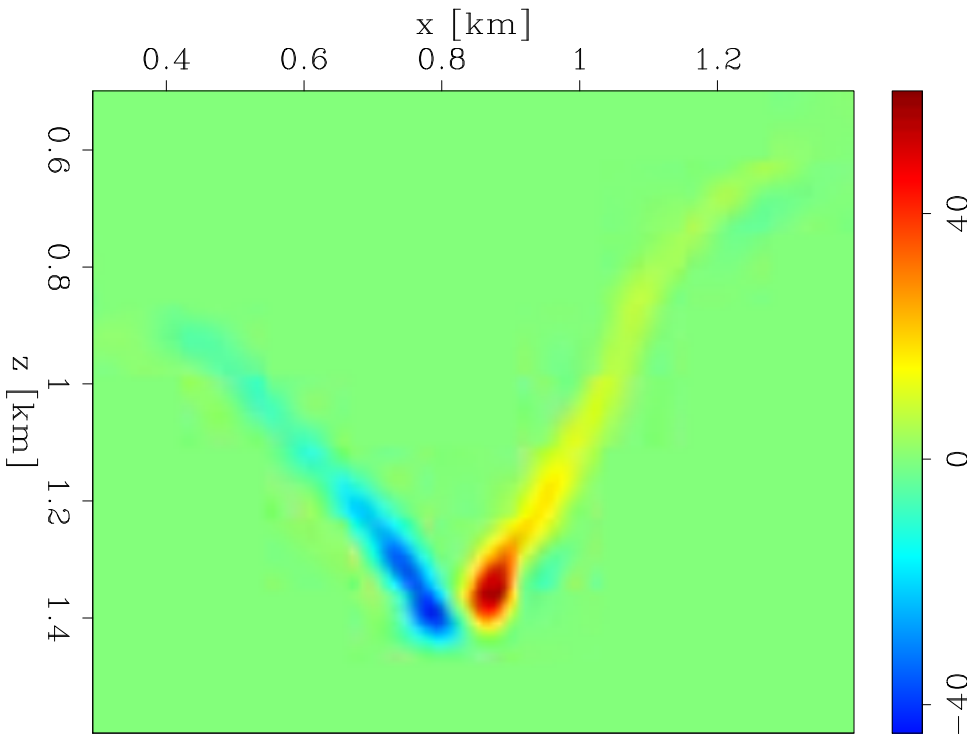


Figure 2.24: The Newton search direction using the inverted full Hessian on the single perturbation model. [**CR**] chapter2/. single-final-full

matrix must be found. If too large of a scaling is selected, the operator becomes more like a scaled identity matrix, negating the potential benefit of inverting the full Hessian system to begin with. If the scaling is too small, the system will still be ill-conditioned, and prone to instability as observed earlier. The ideal scaling is slightly more than the value of the most negative eigenvalue of the operator. This makes the operator positive definite. Since our model (and as a result, our Hessian) is very large, it is impractical to store or factorize the Hessian matrix to determine the most negative eigenvalue through traditional non-iterative linear algebra methods.

Power Iteration Method

The most practical way to find the best scaling is by using the power iteration method outlined in Larson (2009) to find the maximum absolute-valued eigenvalue (positive in the case shown for Figure 2.27). After this has been found, one shifts the diagonal of the operator by the negative of this value, and then repeat the power iterations to find a new maximum absolute-valued eigenvalue. The difference between this value and

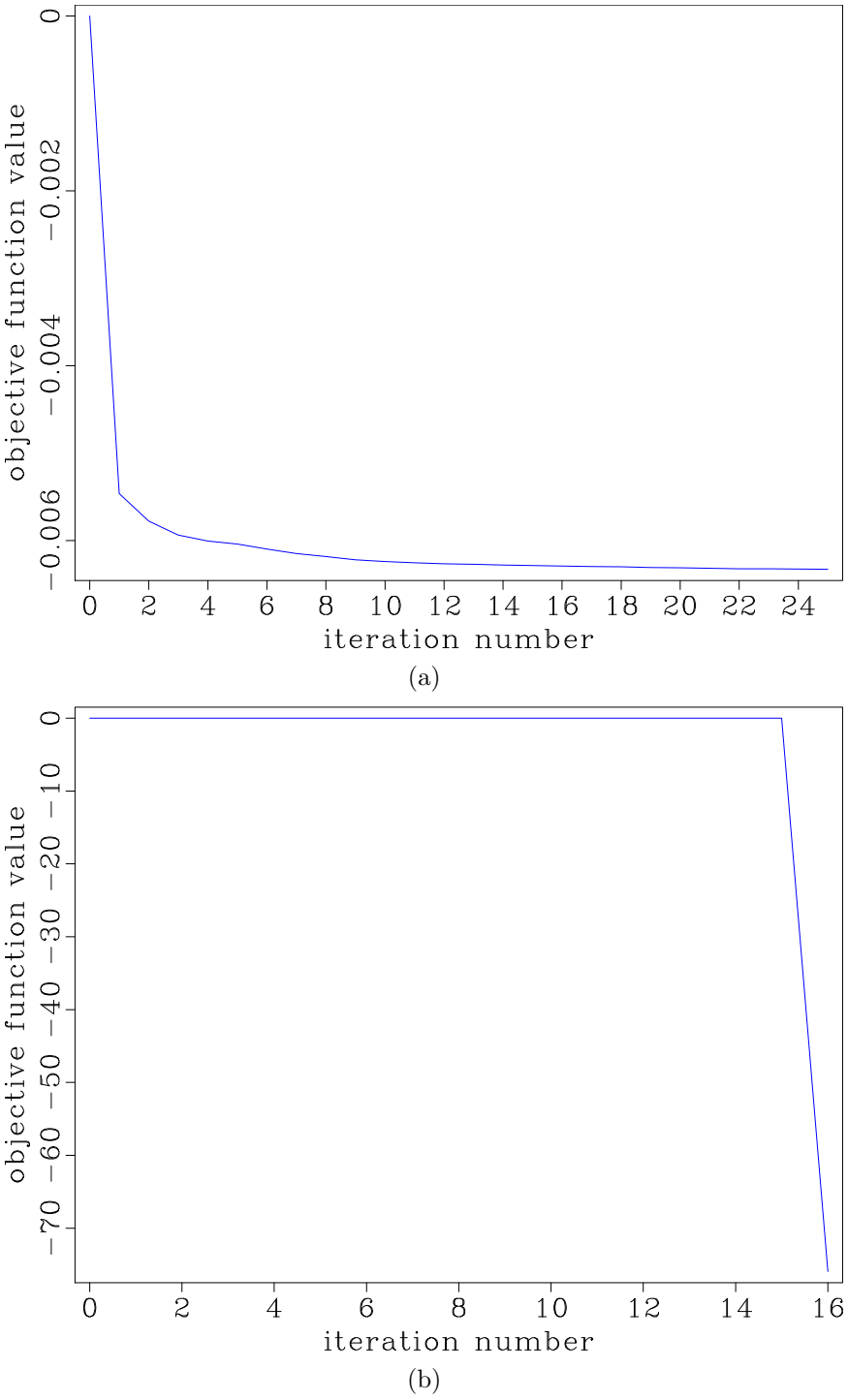
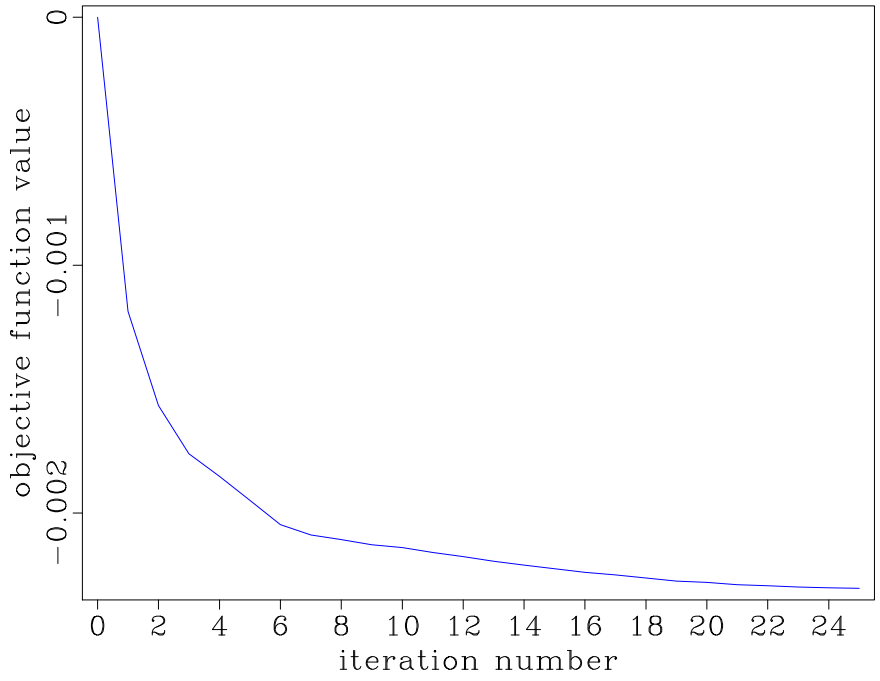
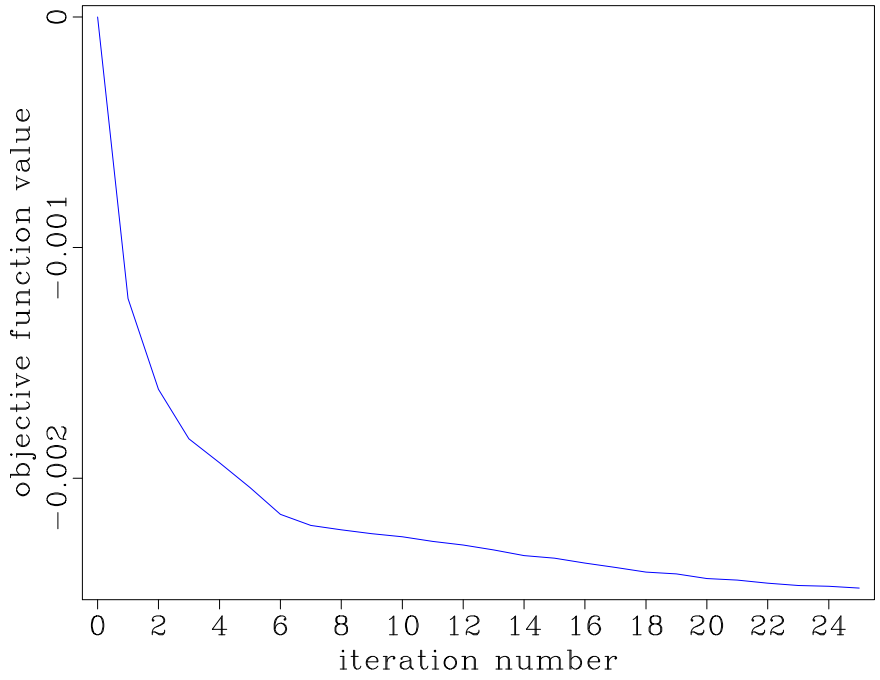


Figure 2.25: The objective functions from the Hessian inversions using the single canyon perturbation model. Note, the values are negative because a conjugate gradient (CG) solver was used instead of a CG least-squares solver. a) Gauss-Newton Hessian. b) Full Hessian. [**CR**] chapter2/. objfunc-single-gn,objfunc-single-full



(a)



(b)

Figure 2.26: The nearly identical objective functions from the Hessian inversions using the double canyon perturbation model. Note, the values are negative because a conjugate gradient (CG) solver was used instead of a CG least-squares solver. a) Gauss-Newton Hessian. b) Full Hessian. [**CR**] chapter2/. objfunc-double-gn,objfunc-double-full

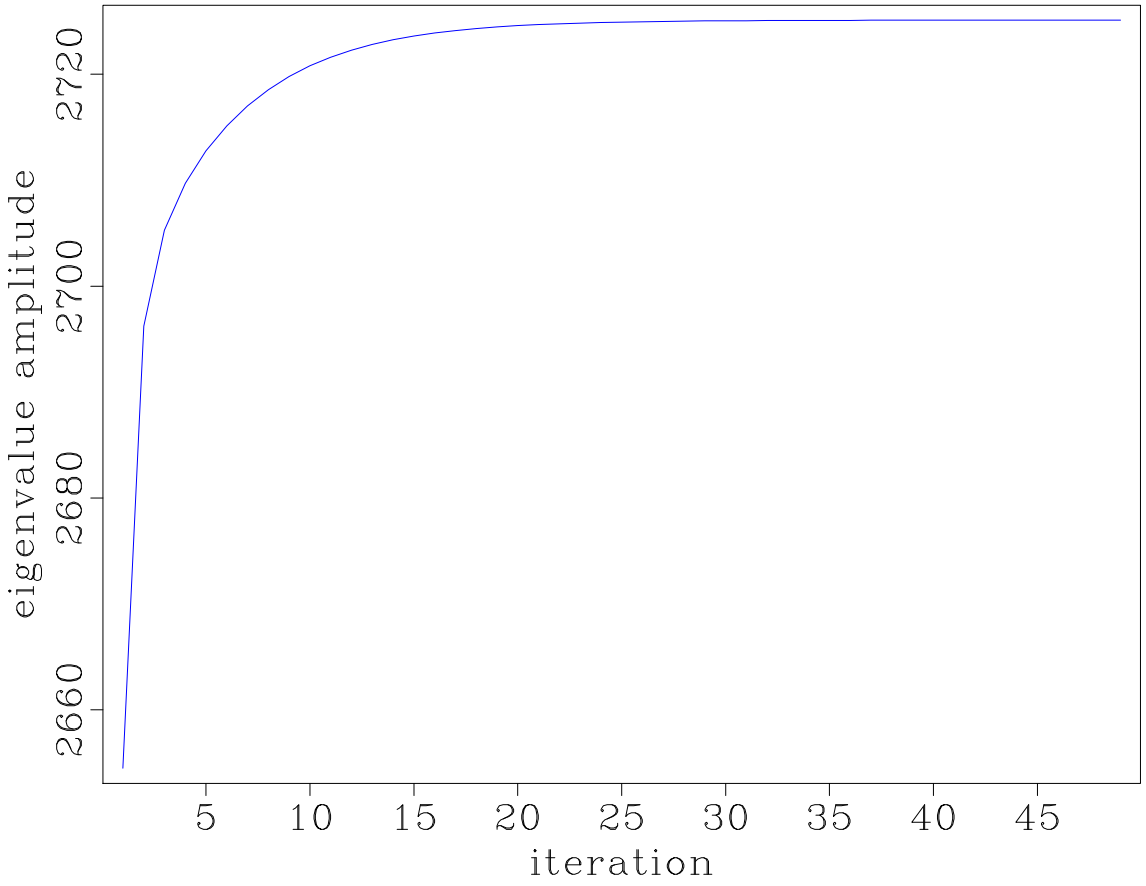


Figure 2.27: The power iteration curve showing the progressing approximation of the maximum absolute value eigenvalue of the full Hessian operator used on the single canyon perturbation model. [**CR**] chapter2/. powerit1

the first one derived is the magnitude of the most negative eigenvalue. I experimented with this method, but found the benefits of this effort to be minimal, and at notable computational cost. Figure 2.27 shows that in practice at least 25 iterations (and so $\sim$ 25 forward full Hessian operator applications) were necessary for each of the two power iteration searches. Once these searches were complete and a proper shift was found, I found that the results of using this Levenberg-Marquardt shift were almost imperceptible from the Gauss-Newton results. Furthermore, since the Hessian operator is model-dependent (and so changes with each outer loop iteration of FWI), these power iteration steps would need to be performed each time the Newton system was inverted.

CONCLUSIONS

Level set concepts can be combined with the FWI objective function to create a shape optimization scheme that allows an elegant way to address the problem of finding an optimal salt body boundary. My demonstrations on simple 2D examples illustrate the relationship between the FWI and implicit surface search directions, and the effect that updating this new model space ultimately has on the velocity model. When I investigate the use of the inverse Hessian to refine the search direction, I find that the Gauss-Newton Hessian approximation is sufficient to improve convergence. However, the theoretically more accurate full Hessian gives mixed results which depend on the model. Robust inversion can only be assured by performing more computation (power iterations) to find an optimal correction for the diagonal elements of the operator. For this reason, I find that the impracticalities of maintaining stability in the full Hessian inversion outweigh the potential benefits from using it. While the Gauss-Newton Hessian is less accurate than the full Hessian, its inversion is stable, and at far less cost.