Chapter 3

Radial basis functions for model sparsity

In the previous chapter, I reviewed the advantages of using the inverse Hessian to improve the search direction used in the FWI problem. In this chapter, I first discuss how re-parameterizing the implicit surface to a sparse domain with Radial Basis Functions (RBFs) may improve the theoretical convergence rate of the Hessian inversion. Next, I demonstrate how RBFs can give a satisfactory representation of a dense implicit surface when suitable parameters are chosen. Last, I give a comparison of level set inversions showcasing the improved convergence rate that is gained by using RBFs.

MOTIVATION

Any uniform increase in the 3D model size increases the number of model parameters by $O(n_x n_y n_z)$. When using a Newton method as described in equation 2.18, we make use of a Hessian that (in matrix form) has $(n_x n_y n_z)^2$ elements. As the model expands to 3D, this quickly becomes intractable to store in memory or even on disk. In practice, we overwhelmingly prefer to solve the inverse Hessian system (equation 2.18) using the conjugate gradient method, which only requires forward applications of the Hessian operator (in our case, the Gauss-Newton Hessian). Regardless, the upper limit of the number of iterations needed to converge fully using the conjugate-gradient method is inversely proportional to the number of model parameters (Aster

et al., 2013). As such, our desire to improve the rate of convergence when solving the Newton system motivates us to reduce the number of model parameters in some manner.

For 3D seismic wave propagation, the coarseness of spatial sampling is based on constraints imposed by the numerical dispersion and numerical stability inherent in our finite differencing scheme. For the purpose of wave modeling, our spatial resolution will generally be higher than the features we can robustly resolve from the data. We can typically use uniform down-sampling methods on the update gradient to create a smaller model space without significantly affecting our end results (Cox and Verschuur, 2001). One of the disadvantages of using down-sampling to reduce the model space is that it uniformly reduces resolution. This runs counter to our interest in having high resolution in specific areas of interest, like the salt boundaries. For this reason, any application of down-sampling usually gives less than satisfactory results for inversion resolution.

In the level set problem, we are most interested in areas around and within the salt boundaries. The implicit surface tracking our salt boundary doesn't need to be represented by a fine grid across the whole domain like the one used for wave propagation. However, aggressive down-sampling schemes will quickly start to deteriorate the information that we wish to keep. What is needed is a method that allows the flexibility of spatially varying resolution.

RADIAL BASIS FUNCTION IMPLEMENTATION

One way to achieve model space reduction is with a basis function that lets us use fewer model parameters to describe a larger spatial area. Radial basis functions are a simple and effective kernel to use. We can separately scale and then sum an aggregation of RBFs to approximate a dense implicit surface, with the approximation accuracy related to the shape of the RBF kernel and the number of RBF kernels used. Kadu et al. (2017a) propose an implementation like this which replaces a dense Cartesian grid parametrization of the implicit surface $\boldsymbol{\phi}$ with a surface described as an aggregate of evenly spaced RBFs:

$$\boldsymbol{\phi}(\boldsymbol{\lambda};\epsilon) = \sum_{i}^{N_\lambda} \lambda_i \exp^{-(\epsilon \mathbf{r})^2}. \tag{3.1}$$

In this formulation, $\boldsymbol{\lambda}$ is the new (sparse) model parameter vector, N_λ is the length of $\boldsymbol{\lambda}$, $\mathbf{r}$ is radial distance from the ith RBF center, and ϵ controls the sharpness of the RBF taper (constant for all i). In their approach, Kadu et al. (2017a) chose a uniform distance between each RBF center location beforehand based on the resolution they desired.

However, there are only a few areas where one really wants high resolution (namely, the salt edge where we expect boundary movement). The resolution achieved with the sparse parametrization is primarily based on how many RBFs are used to describe a particular area (RBF density). Because the salt evolves from an initial boundary, the further from this initial boundary a model region is, the less likely that it will be updated, which means low RBF density can be justified. On the other hand, regions close to the current salt boundary are more likely to update, justifying higher RBF density. Therefore, I introduce the idea of spatially varying the density of the RBF centers to represent the implicit surface in a sparse fashion (as shown in Figure 3.1). This allows clustering the center locations of the RBFs in areas I expect to see updating occur, while using a lower density in regions where I don't expect updating (see Figure 3.2). This is a more efficient way to distribute RBF positions, resulting in fewer RBF parameters needed to attain high resolution around the salt boundary than if I used the regular RBF spacing described in Kadu et al. (2017a).

For the new representation of $\boldsymbol{\phi}$ described in equation 3.1, the operator $\mathbf{D}$ must be modified to account for the additional linear transformation:

$$\begin{aligned}\mathbf{D} &= \left[\frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \boldsymbol{\phi}}\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\lambda}} \quad \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \mathbf{b}}\right] \\ &= \left[\widetilde{\delta}(\boldsymbol{\phi_o})(\mathbf{c_s}-\mathbf{b})\exp^{-(\epsilon \mathbf{r})^2} \qquad \mathbf{I} - \widetilde{H}(\boldsymbol{\phi_o})\right].\end{aligned} \tag{3.2}$$

In this formulation, $\widetilde{H}$ is the Heaviside approximation (equation 2.12), $\mathbf{I}$ is the identity matrix, $\widetilde{\delta}$ is the derivative of $\widetilde{H}$, $\mathbf{r}$ is radial distance from an RBF center, $\mathbf{b}$ is the background velocity ($\mathbf{b_0}$ is fixed), $\boldsymbol{\phi}$ is the implicit surface ($\boldsymbol{\phi_0}$ is fixed), and $\mathbf{c_s}$ is the constant salt velocity. Further, the model space has also changed:

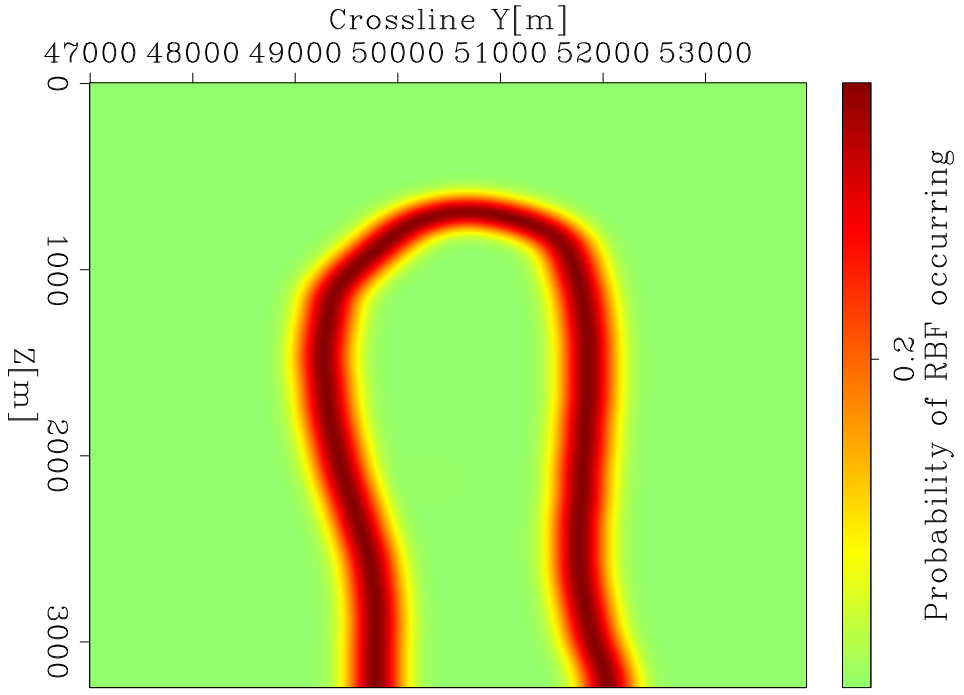


Figure 3.1: The probability distribution used to randomly position the radial basis function centers in Figure 3.2. [**ER**] chapter3/. centers-dist

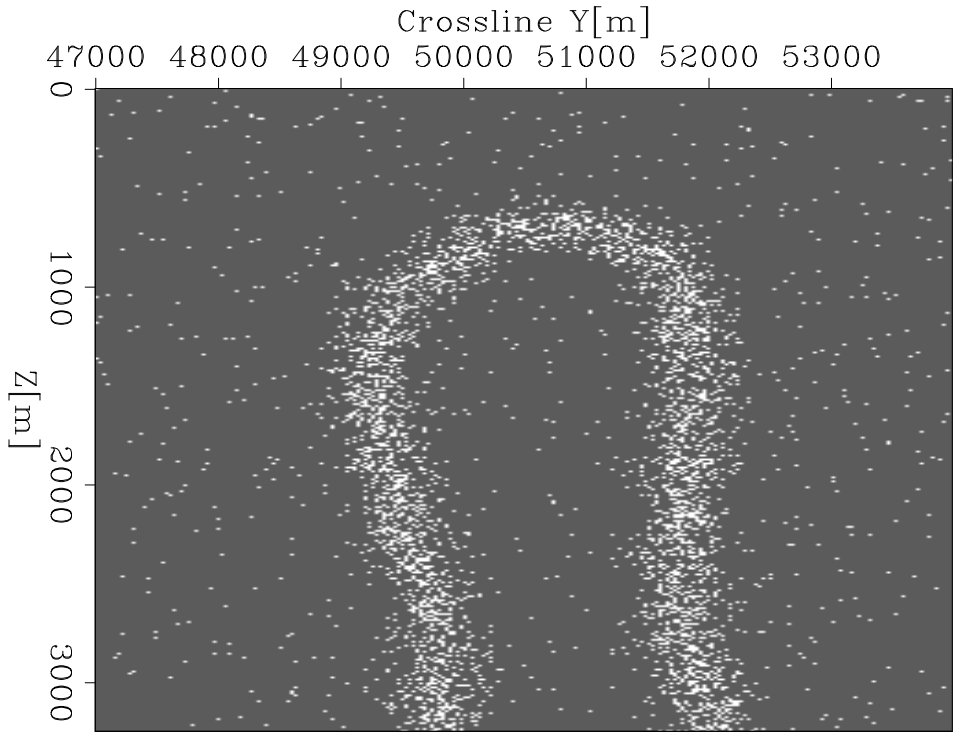


Figure 3.2: Center positions for radial basis functions used to construct the implicit surface. [**ER**] chapter3/. centers

$$\triangle \mathbf{p} = \begin{bmatrix} \triangle \boldsymbol{\lambda} \\ \triangle \mathbf{b} \end{bmatrix}.$$

When I apply the operator $\mathbf{D}$ or $\mathbf{D^T}$, I consider the locations of the RBF centers to be fixed throughout the inversion (for example, as shown in Figure 3.2).

Computational considerations

The forward application of the $\mathbf{D}$ operator derived earlier now includes a $\exp^{-(\epsilon \mathbf{r})^2}$ term. This means that for each element λ_i in the model vector, I scale and sum a Gaussian function to the aggregated surface, $\boldsymbol{\phi}$. If I compute $\exp^{-(\epsilon \mathbf{r})^2}$ over the full model, then the $\mathbf{D}$ operator becomes expensive to apply on a large 3D spatial domain, since the algorithm would loop over the full model space for each RBF. One observation that I leverage is that the value of the radial basis function decreases significantly at high values of r. Furthermore, I scale and sum the same Gaussian each time (ϵ is fixed). Based on this, I pre-compute the radial basis function $\exp^{-(\epsilon \mathbf{r})^2}$ just once over a region where its value is actually significant. Naturally, the size of this region is based on the taper of the RBF, which is governed by ϵ. By choosing ϵ well, and then pre-computing the corresponding Gaussian function over a limited region, the RBF summation computation is greatly simplified, and increases the speed of applying $\mathbf{D}$ or $\mathbf{D^T}$ significantly.

EXAMPLES OF RBF FITTING

In order to show how RBFs can accurately represent a salt body with far fewer parameters than a dense representation, I demonstrate their use on a Gulf of Mexico velocity model provided by Shell. I choose a section of the velocity model that has a notable salt protrusion in it as an example (Figure 3.3). Beginning with this model, I build a probability density map that favors placing RBF centers near the original picked boundary (Figure 3.1), with less likelihood further from that boundary. From this, I generate random RBF positions (Figure 3.2). Using these RBF centers, I then perform a non-linear conjugate gradient inversion to find the proper weighting of the

RBF kernels in order to best fit the starting model salt shape (Figure 3.4(b)), which is built from an initial implicit surface (Figure 3.4(a)). The inversion finds sparse parameters that create a new implicit surface (Figure 3.5(a)), which looks different from the one used to build the fitting model (Figure 3.4(a)). However, when I apply the Heaviside function to either of these to create the salt, we can see that the sparse parameters create a model with a good fit (compare Figures 3.4(b) and 3.5(b)). The inversion converges relatively quickly (Figure 3.6), but naturally has some unresolved residual since the RBF parametrization is sparse and cannot match the dense salt model guess perfectly (see Figure 3.7(a)).

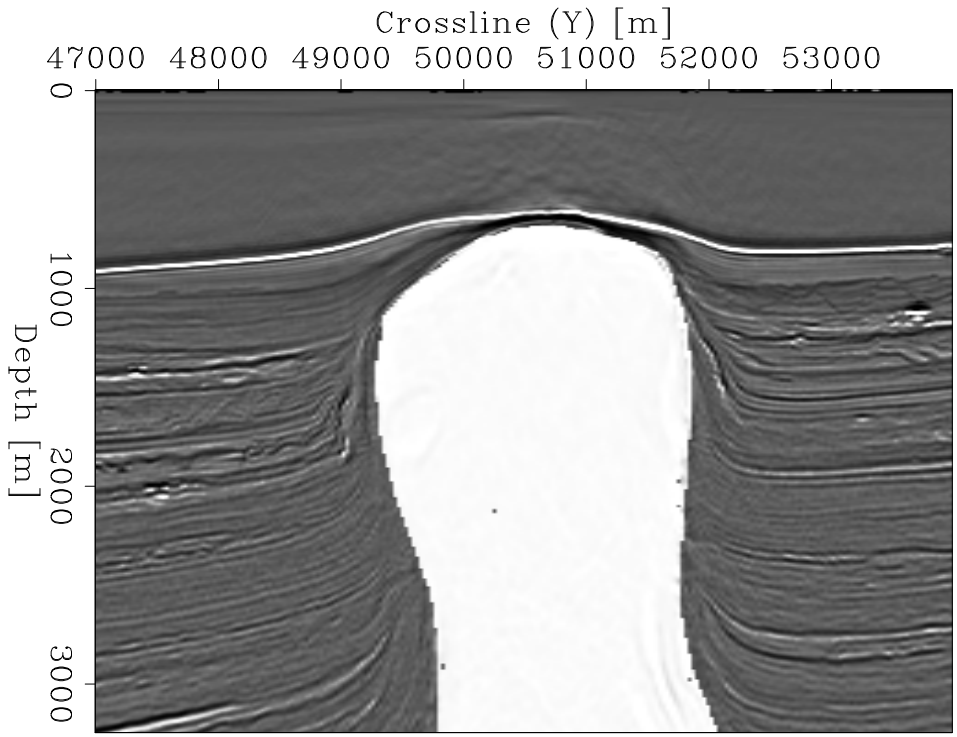
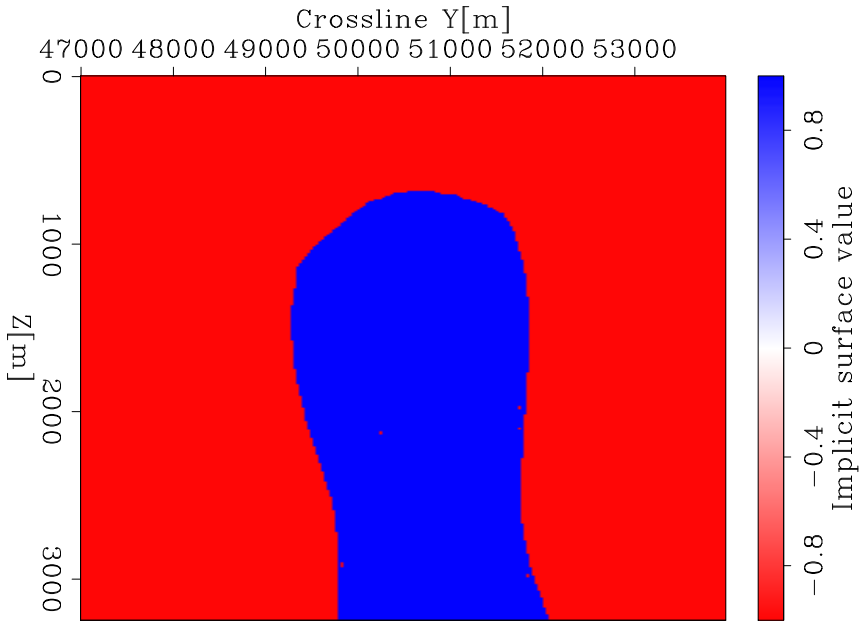
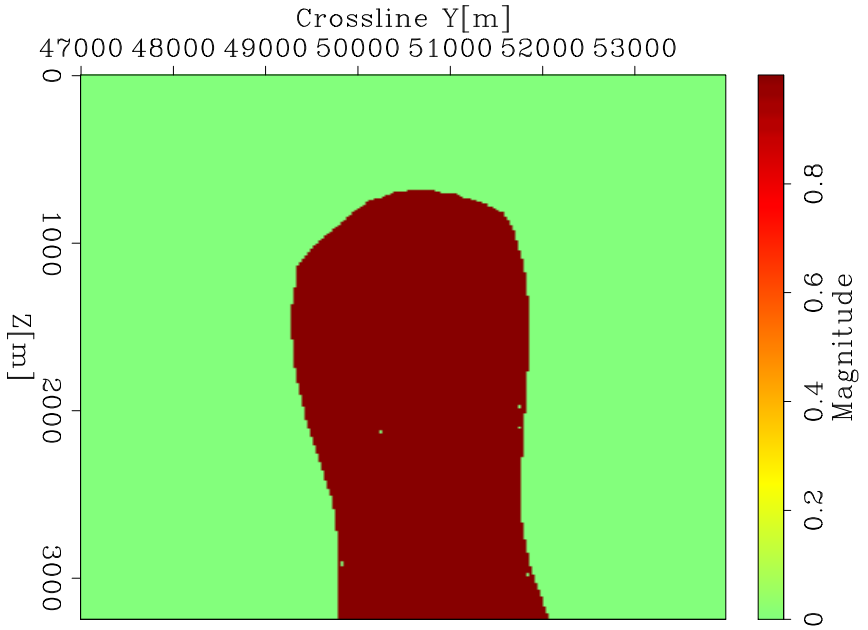


Figure 3.3: Salt model used by Shell (white) overlaid on the corresponding RTM image. [**ER**] chapter3/. both

Figure 3.7(a) shows that the matching model and the resulting inverted model after Heaviside function application are quite similar. However, if we choose ϵ and the corresponding RBF footprint poorly, we wont be able to represent the original salt model as well as we could. When ϵ is too high, the RBF decays quickly, resulting in a model that is less smooth. Alternatively, when ϵ is too low, the RBF decays slowly and creates a model that is too smooth. Figure 3.7(b) shows a case where these parameters were chosen poorly, while Figure 3.7(a) does a much better job in

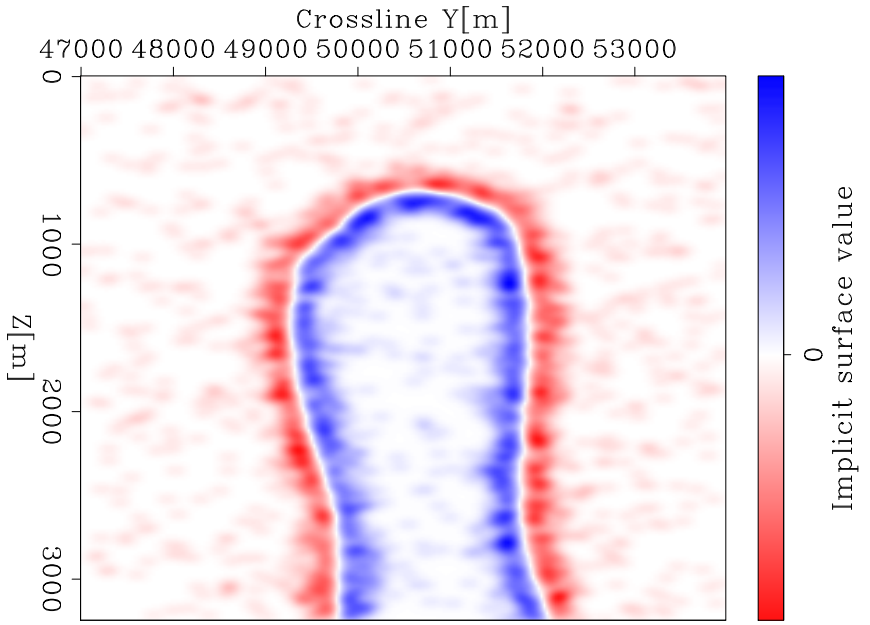


(a)

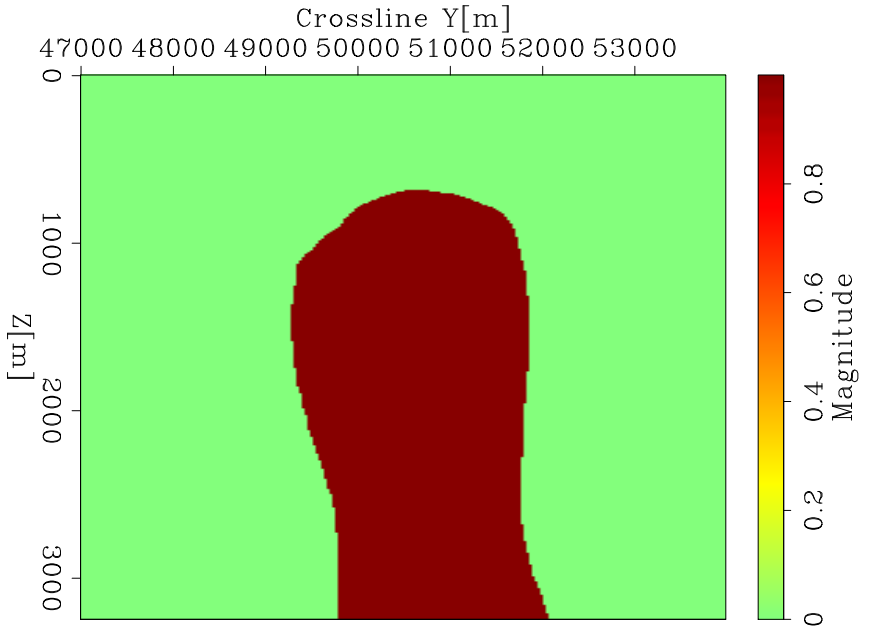


(b)

Figure 3.4: a) Implicit surface $\boldsymbol{\phi}_o$ that creates the salt body shape (b) that my inversion tries to match. b) is the result of applying the Heaviside function to $\boldsymbol{\phi}_o$. chapter3/. matching-phi,matching-phi-Heavi



(a)



(b)

Figure 3.5: a) Implicit surface $\boldsymbol{\phi}_{\text{final}}$ created from final inverted RBF parameters ($\boldsymbol{\phi}_{\text{final}} = \mathbf{D}\boldsymbol{\lambda}_{\text{final}}$); b) the result of applying the Heaviside function to $\boldsymbol{\phi}_{\text{final}}$. chapter3/. resulting-phi,resulting-phi-Heavi

both the salt center region as well as the boundary.

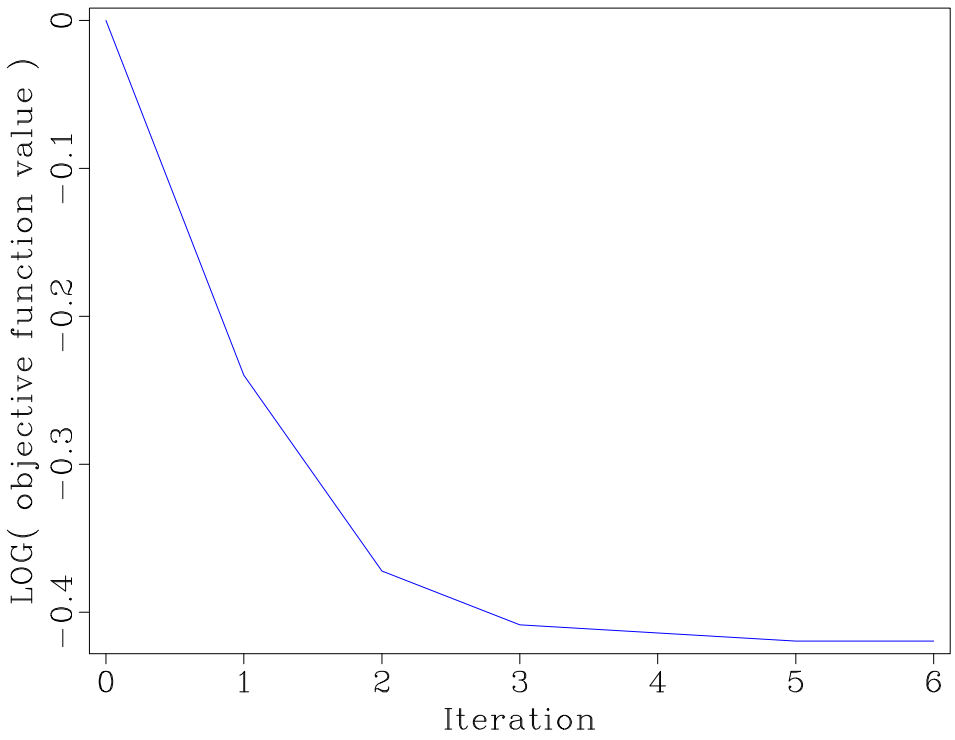
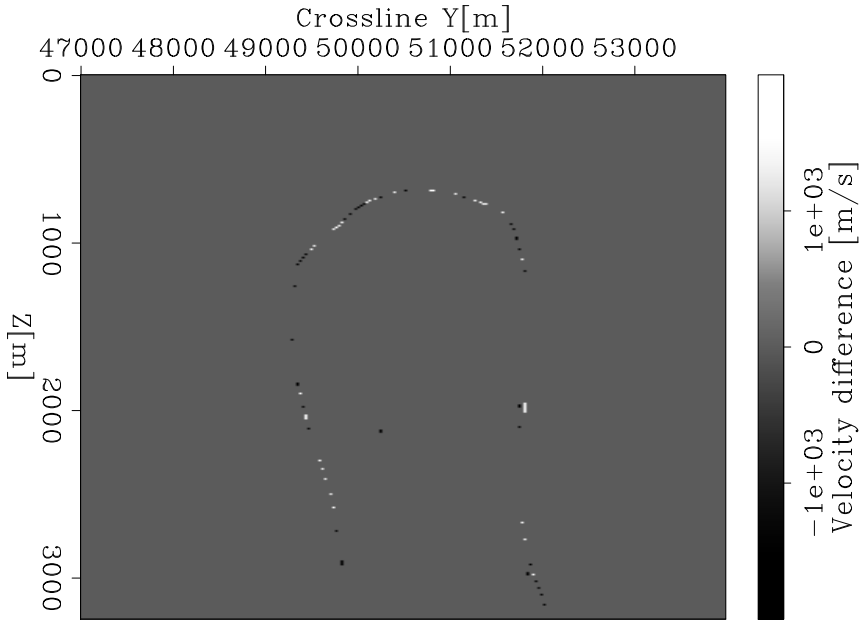


Figure 3.6: Log of normalized objective function from the non-linear inversion used to find the RBF parameters used in Figures 3.5(a), 3.5(b) and 3.7(a). [**ER**] chapter3/. objfunc

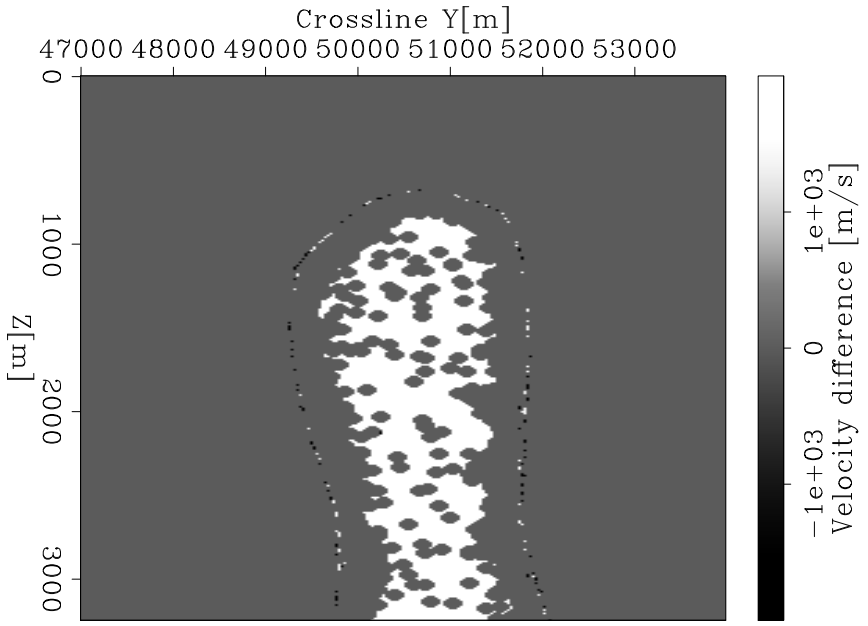
COMPARISON OF SPARSE (RBF) AND DENSE MODEL INVERSIONS

In order to illustrate the improved rate of convergence gained by using a sparse RBF model, I demonstrate on a 2D synthetic inversion example. The goal here is to show how RBFs affect the greater shape optimization problem by comparing the inversion convergence rates between a sparse RBF parametrization and a dense non-RBF parametrization.

I choose a 'true' model similar to Figure 3.3 that I wish to invert for (Figure 3.8) that has an inclusion close to the edge. This model is chosen to show how the RBF inversion can invert for a more unusual model geometry. I begin with a model containing a much smaller inclusion (Figure 3.9). Just as in chapter two (Algorithm 1), the inversion workflow has an outer loop where I do non-linear modeling to find the



(a)



(b)

Figure 3.7: Differences between original salt model and the resulting model produced by the RBF representation. (a) shows difference of fitted salt model using $\epsilon = 0.25$ value, while (b) shows difference of fitted salt model using $\epsilon = 2.25$ value. Both cases used 98% fewer model parameters than the original full-grid scheme, and both used the same RBF positions. Background velocity is 2.5 km/s and salt velocity is 4.5 km/s. [**ER**] chapter3/. rbfinv-diff-full,rbfinv-diff-sparse

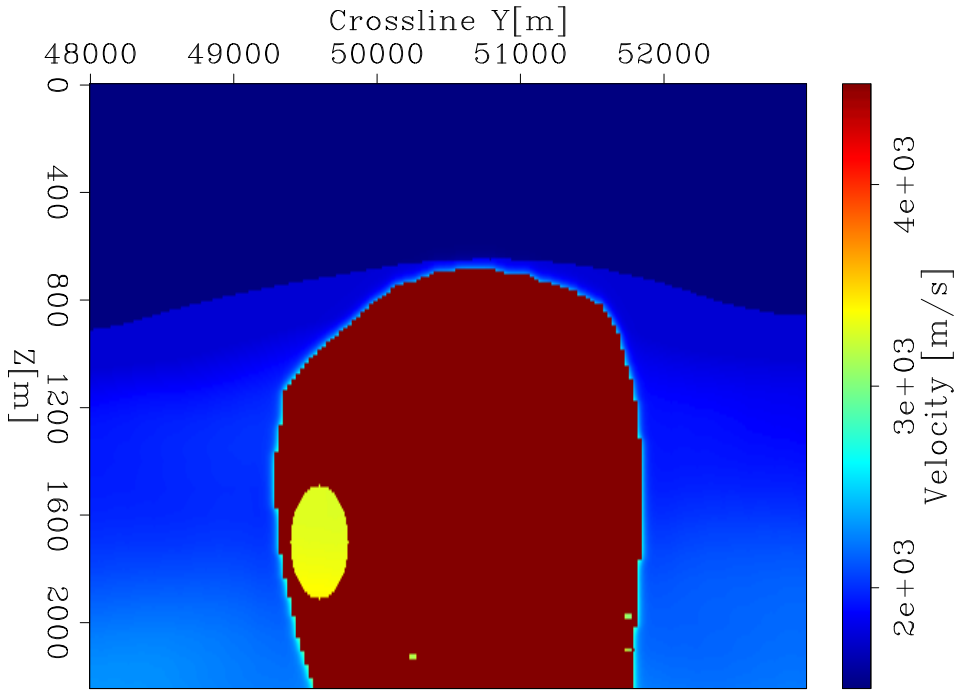


Figure 3.8: True velocity model that was used to synthetically generate the 'observed' data. [**ER**] chapter3/. true-model-rbf-inversion

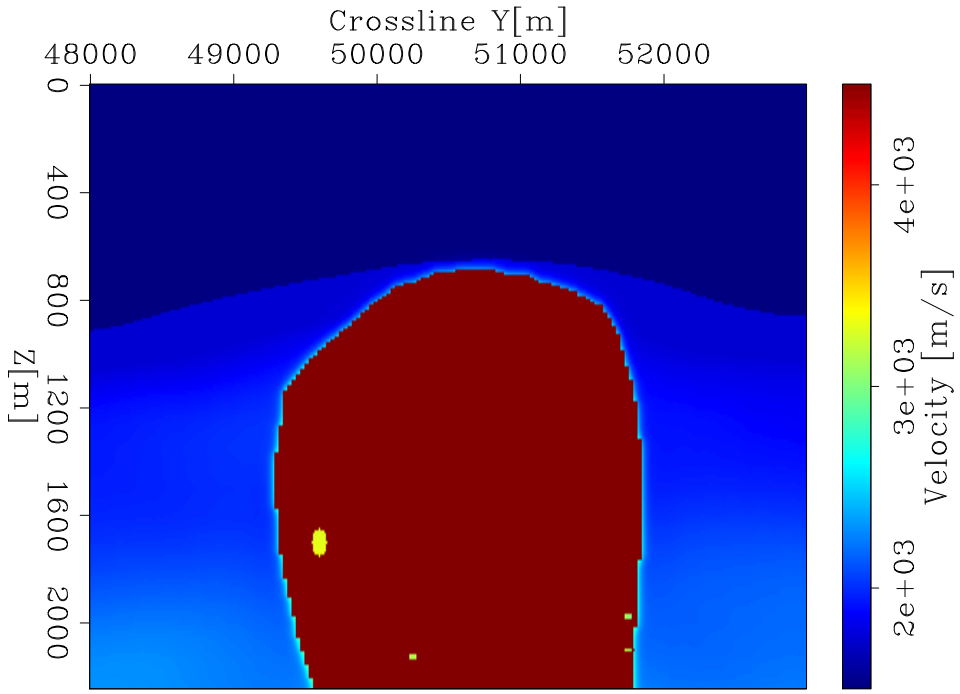


Figure 3.9: Starting velocity model. Note the inclusion is much smaller than in the true model (Figure 3.8). [**ER**] chapter3/. start-model-rbf-inversion

residual and use the adjoint Born operator to find the gradient. However, following this I now have an inner loop using iterative methods to invert the Hessian and find the search direction from the gradient (solving equation 2.18), before performing the linesearch. In these examples, the inner loop uses a Gauss-Newton Hessian, and I compare using a dense model (no RBF parametrization) to using a sparse model (with RBF parametrization). The RBF model has only 7% of the model points that the dense model has. I perform the same number of iterations (20) for each inner-loop linear inversion of the Gauss-Newton Hessian using a conjugate gradient solver. For the first Hessian inversion, the sparse RBF parameterization (Figure 3.10) converges faster than the dense model example (Figure 3.11). Note that the objective function curves become negative since I use a conjugate gradient (CG) solver based on equation 3.3:

$$\min \psi(\triangle \mathbf{m}) = \frac{1}{2} \triangle \mathbf{m}^\mathbf{T} \mathbf{H} \triangle \mathbf{m} - \mathbf{g}^\mathbf{T} \triangle \mathbf{m}, \tag{3.3}$$

instead of a conjugate gradient least-squares (CGLS) solver (based on equation 3.4):

$$\min \psi(\triangle \mathbf{m}) = \frac{1}{2} ||\mathbf{H} \triangle \mathbf{m} - \mathbf{g}||_2^2. \tag{3.4}$$

Because the Hessian is a symmetric operator, I can take advantage of using the CG solver instead of the more expensive CGLS solver. However, the residual is not squared as in CGLS, so the objective function can take negative values.

I find that after 14 outer loop (non-linear) iterations the sparse (RBF) inversion converges, while the dense (non-RBF) inversion objective function is still descending after 40 iterations (Figure 3.12). The normalized model norm for the sparse inversion also reaches a lower value (Figure 3.13), while the non-RBF inversion actually increases instead. By representing the dense model sparsely with RBFs, I reduce the number of parameters as well as create a smoother equivalent update in the dense model space (with smoothness based on the ϵ used). In this sense, the RBFs act somewhat like a regularization. Both these factors contribute to the improved convergence rate that I find when using RBFs in the inversion. When comparing the inverted model results of each parametrization, one can clearly see that the RBF approach (Figure 3.14) provides a superior result to the dense model space parametrization (Figure 3.15).

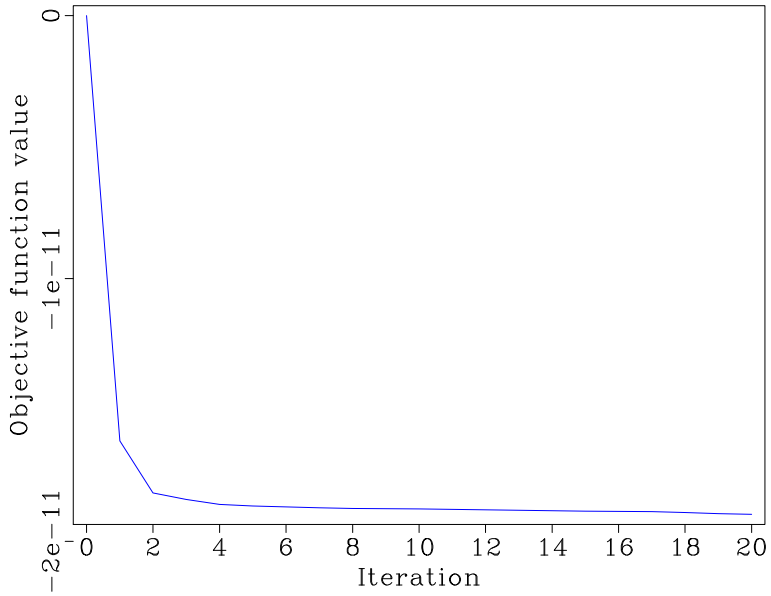


Figure 3.10: Objective function from the first inner-loop Gauss Newton inversion of the sparse (RBF) model system. [**CR**] chapter3/. GNinversion-rbf-objfunc-0

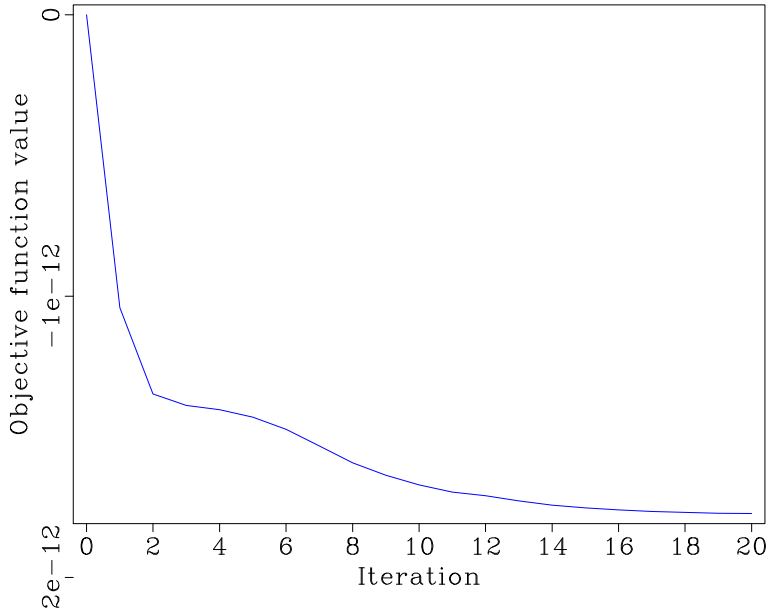


Figure 3.11: Objective function from the first inner-loop Gauss Newton inversion of the dense (non-RBF) model system. [**CR**] chapter3/. GNinversion-norbf-objfunc-0

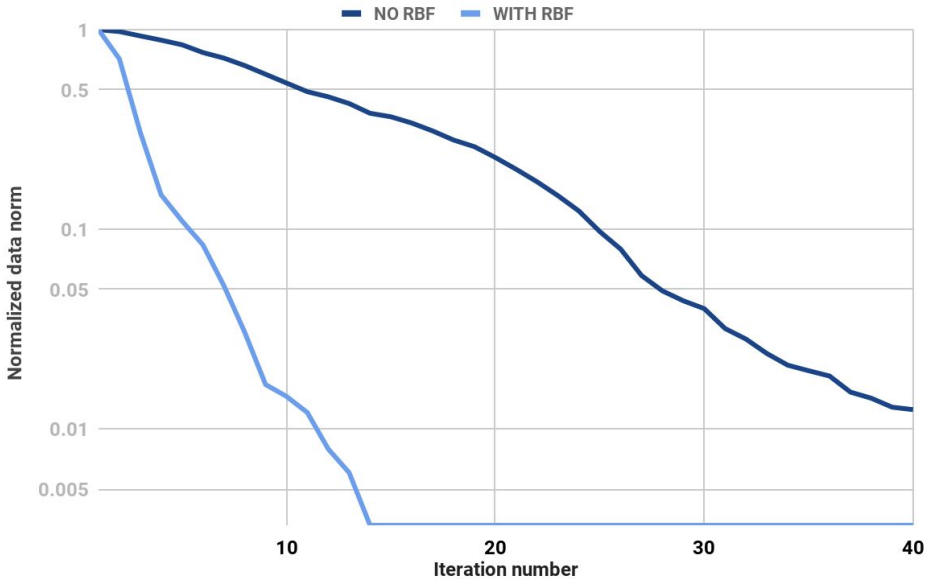


Figure 3.12: Objective function (data norm) comparison of the outer-loop inversion for RBF and non-RBF parameterized model approaches. [**CR**] chapter3/. RBF-vs-noRBF-dataNorm

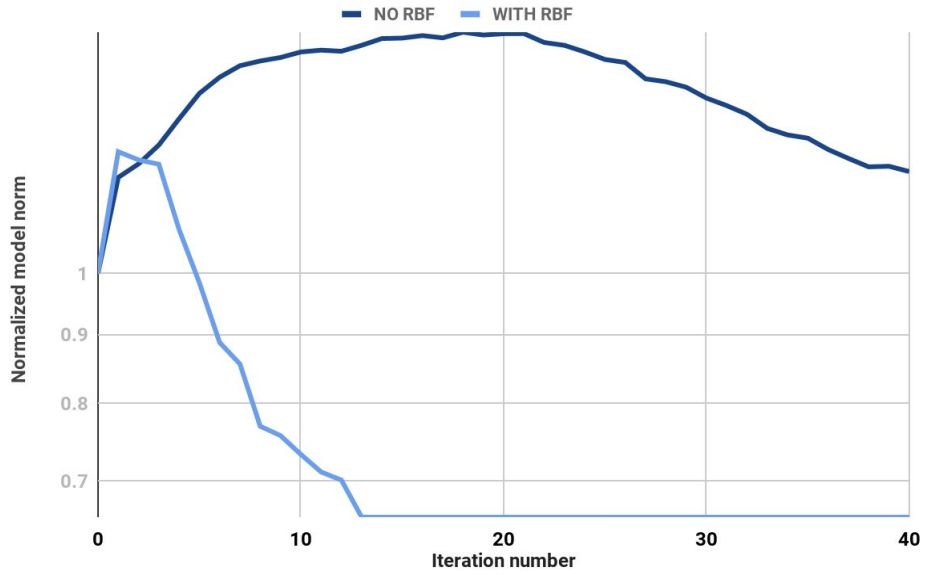


Figure 3.13: Model norm comparison of the outer-loop inversion for RBF and non-RBF parameterized model approaches. [**CR**] chapter3/. RBF-vs-noRBF-modelNorm

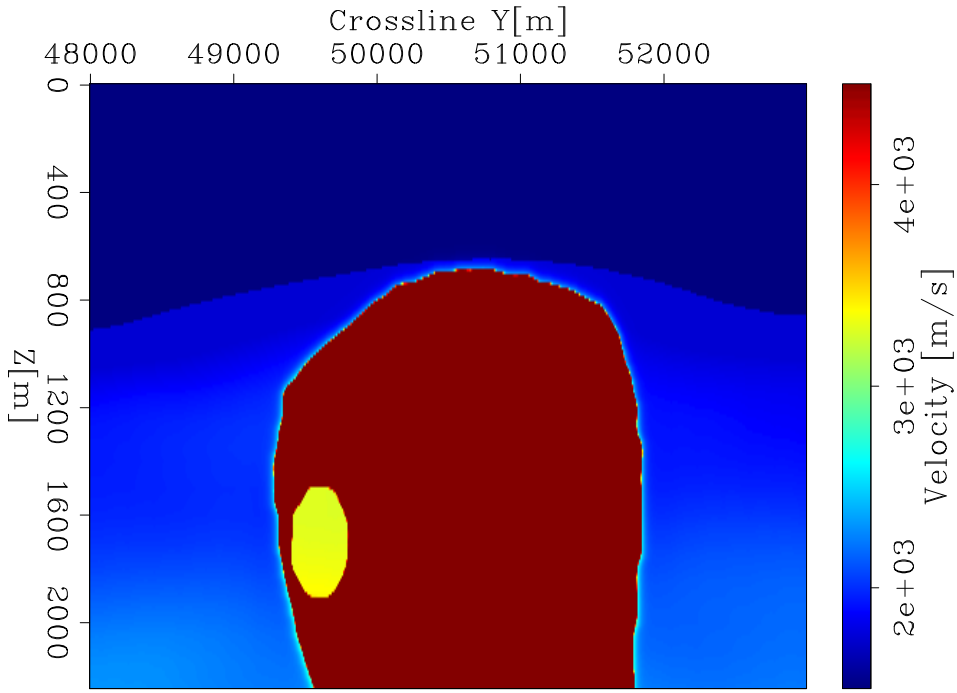


Figure 3.14: Velocity model after 30 outer loop (non-linear) iterations using RBF parametrization. chapter3/. model-30-RBF-inversion

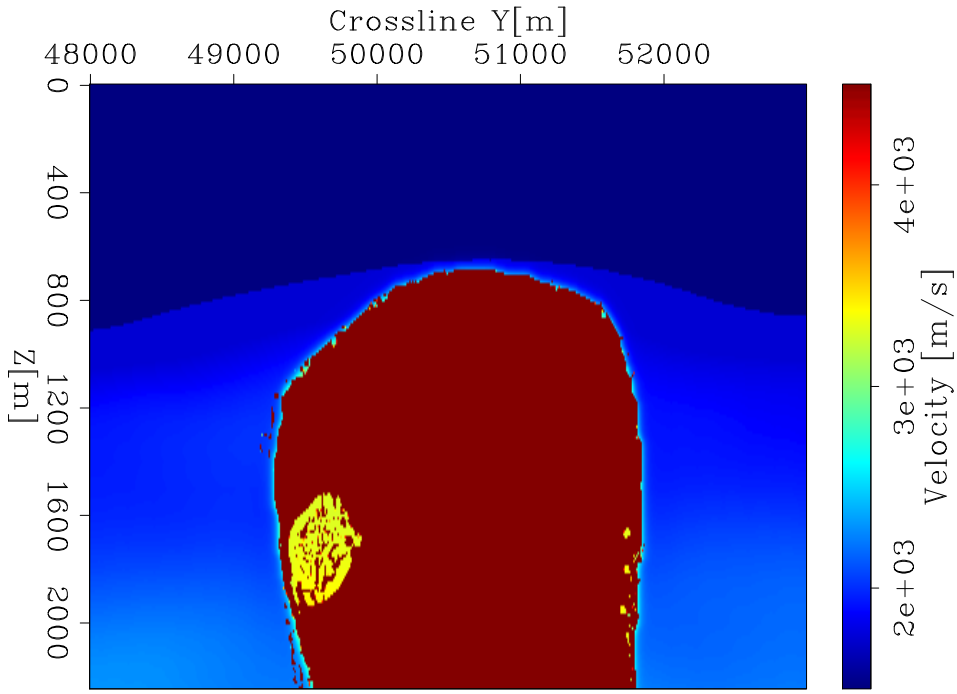


Figure 3.15: Velocity model after 30 outer loop (non-linear) iterations without RBF parametrization. chapter3/. model-30-noRBF-inversion

CONCLUSIONS

The speed at which one can invert the Hessian system and find the search direction is sensitive to the number of parameters in the model, and using 3D spatial models requires a large number of model parameters for wave propagation. However, I can sparsely represent this dense model using radial basis functions to achieve significant parameter reduction. I show that this representation allows me to accurately depict the dense model, and that steps can be taken to make this transform computationally efficient. Finally, when I compare inversions using a sparse representation (RBFs) versus a dense representation, I find that the sparse model inversion provides a better outcome and a faster convergence rate.