Chapter 2

Prediction-error filters and interpolation

This chapter is a review of interpolating seismic data using prediction-error filters (PEFs) (Claerbout, 1992, 2004; Crawley, 2000), a process performed in two steps. The first is to estimate a PEF on fully-sampled *training data*, which ideally have the same autocorrelation as the data we wish to interpolate. This estimation minimizes the squared prediction error to find the set of nontrivial filter coefficients that most effectively predicts the training data. Convolving the PEF with the training data produces an output with an approximately white spectrum. A PEF, with its relatively small number of coefficients, represents useful information obtained from training data such as the amplitude spectra and dip information (in more than one dimension), while ignoring the amplitude scaling and phase of the data. A multi-dimensional PEF can accurately predict both multiple conflicting dips and spatially-aliased data. The PEF can have as many dimensions as the data; moreover, as the dimensionality of the data increases, prediction using a PEF with correspondingly higher dimensionality results in improved prediction capacity.

Once this PEF has been obtained from *training data*, it is then used in the second step wherein the missing data are estimated using the PEF. I define this result as

the *interpolated data*, that are composed of the original *sampled data* and the created *interpolated values*. Since the PEF embodies the amplitude spectrum of the training data, the missing data can be estimated by minimizing the output from convolution of the known PEF with the final model, the interpolated data. This is again posed as a least-squares problem, in which the known data are held fixed so that only the missing data are allowed to vary, and is solved using a conjugate-direction method.

A multi-dimensional PEF is wholly dependent upon its training data, in particular its multi-dimensional autocorrelation. The examples in this chapter show that, when the fully-sampled *original data* including the missing samples to be interpolated, are used as training data, the PEF is able to accurately interpolate the data. When the training data are less than ideal, the quality of the interpolation is compromised.

This interpolation using even ideal training data is degraded when a multi-dimensional PEF is estimated on data containing slopes that vary as a function of space. This failure is caused by the assumption of stationarity, that all events are planar, which is violated when the local autocorrelation of the data varies as a function of space. I initially address this problem by breaking both the PEF estimation and the interpolation into smaller overlapping patches that are assumed to contain locally planar features. After each patch is interpolated independently, the patches are reassembled with appropriate weighting for overlapping portions of the patches to produce the final output. While this result is marginally better than that from using the purely stationary approach, it is far from ideal, and many parameters are required, such as the number of patches and the amount of overlap. Instead, I use a single spatially-variable, nonstationary PEF (Crawley, 2000) on the entire dataset. For the nonstationary model data tested, the nonstationary PEF predicts nearly all coherent energy in the test data, and accurately reconstructs missing data. This due to both the larger number of filter coefficients possible with a nonstationary PEF and applying the filter on the entire interpolated data simultaneously.

PREDICTION-ERROR FILTER ESTIMATION

This section reviews how to estimate a forward prediction-error filter from fully-sampled training data (Robinson and Treitel, 1967; Claerbout, 1976; Yilmaz, 1987). Consider first a one-dimensional example. A PEF captures the inverse amplitude spectrum of the training data and, when convolved with the training data, produces an output that is increasingly uncorrelated as the size of the PEF increases.

A forward linear prediction filter, p_i, of n_p points predicts values, $\hat{d}_j$, based on previous inputs, $d_{j-i}, i = 1, \ldots, n_\mathrm{p}$, so that for all $j = 1, \ldots, n_\mathrm{d}$

$$\hat{d}_j = \sum_{i=1}^{n_\mathrm{p}} d_{j-i} p_i. \tag{2.1}$$

The difference between the actual data and the predicted estimate is the prediction-error series r_j, for all $j = 1, \ldots, n_\mathrm{d}$

$$r_j = d_j - \hat{d}_j = d_j - \sum_{i=1}^{n_\mathrm{p}} d_{j-i} p_i. \tag{2.2}$$

This data-prediction problem can be rephrased as a data-whitening one; as the prediction improves, the residual decreases and becomes increasingly random. Equation 2.2 can be expressed in terms of vectors, denoted with bold lower-case letters, and matrices, denoted by bold upper-case letters, wherein we now define an n_f-element prediction-error filter $\mathbf{f} = [f_1, f_2, f_3, \ldots, f_{n_\mathrm{f}}]^T$ with $f_1 = 1$ fixed and set the remaining $n_p = n_f - 1$ coefficients as the negative of the prediction filter coefficients so that $\mathbf{f} = [1, -p_1, -p_2, \ldots, -p_{n_\mathrm{p}}]^T$. We also use the n_d-element data vector $\mathbf{d} = [d_1, d_2, \ldots, dn_\mathrm{d}]^T$ to create the $n_\mathrm{d} \times n_\mathrm{p}$-element data convolution matrix, $\mathbf{D}$. The

D matrix contains rows of shifted copies of the data vector,

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ d_1 & 0 & 0 & \cdots & 0 \\ d_2 & d_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n_\mathrm{p}} & d_{n_\mathrm{p}-1} & d_{n_\mathrm{p}-2} & \cdots & d_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_{n_\mathrm{d}-2} \\ 0 & 0 & 0 & \cdots & d_{n_\mathrm{d}-1} \\ 0 & 0 & 0 & \cdots & d_{n_\mathrm{d}} \end{bmatrix}. \tag{2.3}$$

The matrix **D** is multiplied with the n_p unknown elements in the prediction-error filter vector **Kf** to produce the $n_\mathrm{d} + n_\mathrm{p}$-element output residual (prediction-error) vector $\mathbf{r}_\mathrm{d}$,

$$\mathbf{r}_\mathrm{d} = \mathbf{DKf} + \mathbf{d}. \tag{2.4}$$

The subscript in $\mathbf{r}_\mathrm{d}$ denotes that the residual pertains to the fit of the data. In order to isolate the *unknown* filter coefficients from the leading unity value of the prediction-error filter **f**, we have introduced a diagonal $n_\mathrm{p} \times n_\mathrm{f}$ matrix **K** that selects the unknown filter coefficients, so that

$$\mathbf{K} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

In equation 2.4, the **DKf** term creates a vector of the negative of the predicted values and the **d** vector contains the original data that when combined create the prediction error, $\mathbf{r}_\mathrm{d}$. We now minimize this prediction error by finding the unknown coefficients of **f** that minimize the L_2 norm of the residual, $\|\mathbf{r}_\mathrm{d}\|^2$, given by

$$\|\mathbf{r}_\mathrm{d}\|^2 = \mathbf{r}_\mathrm{d}^\dagger \mathbf{r}_\mathrm{d} = (\mathbf{DKf} + \mathbf{d})^\dagger (\mathbf{DKf} + \mathbf{d}) \tag{2.6}$$

where the symbol † denotes the adjoint or complex-conjugate transpose of a matrix. We minimize equation 2.6 by expanding this expression and setting the derivatives with respect to the unknown filter values $\mathbf{Kf}$ to zero:

$$\frac{\partial \|\mathbf{r}_\mathrm{d}\|^2}{\partial \mathbf{Kf}} = \mathbf{0} = \mathbf{K}^\dagger\mathbf{D}^\dagger\mathbf{DKf} + \mathbf{K}^\dagger\mathbf{D}^\dagger\mathbf{d}. \tag{2.7}$$

We then move the second term to one side to obtain

$$\mathbf{D}^\dagger\mathbf{DKf} = -\mathbf{D}^\dagger\mathbf{d} \tag{2.8}$$

which is the same as equation B-78 in Yilmaz (1987), where the matrix $\mathbf{D}^\dagger\mathbf{D}$ is an autocorrelation matrix, which has Toeplitz structure, and $\mathbf{D}^\dagger\mathbf{d}$ is a vector of the second and subsequent autocorrelation lags for this ungapped filter. Further simplifying the expression and isolating the $\mathbf{Kf}$ term on the left side of equation 2.7 gives

$$\mathbf{Kf} = -(\mathbf{D}^\dagger\mathbf{D})^{-\mathbf{1}}\mathbf{D}^\dagger\mathbf{d}, \tag{2.9}$$

the least-squares formula for the unknown filter coefficients. For solving large systems of equations, such as those we deal with, the method of conjugate directions is particularly efficient (Hestenes and Stiefel, 1952; Shewchuk, 1994; Claerbout, 2004). The conjugate-directions algorithm requires a minimal amount of memory, as it requires only the data vector $\mathbf{d}$ in order to apply the matrices $\mathbf{D}$ and $\mathbf{D}^\dagger$, instead of holding the autocorrelation matrix in memory.

Note three of the important properties of prediction-error filters, proved elsewhere in the considerable literature (Wiener, 1964; Robinson and Treitel, 1967; Burg, 1975; Claerbout, 1976, 1992). First, the prediction-error filter is minimum-delay, so that the PEF yields stable deconvolution. Second, the PEF is dependent solely upon the autocorrelation of the training data, not the phase, polarity, or amplitude scale of the data. Finally, the output of the PEF is uncorrelated with itself for all lags in both directions, meaning that the output spectrum is white.

Having now reviewed 1D prediction-error filter estimation, I next show how the

1D construct can be used with helical coordinates (Claerbout, 1998) to represent a PEF and data in any number of dimensions, demonstrating this on 2D data.

MULTI-DIMENSIONAL FILTERING IN THE HELICAL COORDINATE

The content of this section is a condensation of material found elsewhere (Claerbout, 1998, 2004). The helical coordinate allows implementation of multi-dimensional convolution of multi-dimensional signals as a one-dimensional operation. This coordinate, the *helical* coordinate, can also be used to explain the particular shape of a multi-dimensional prediction-error filter. With use of the helical coordinate, I estimate a two-dimensional PEF on two-dimensional synthetic data.

The helical coordinate is a way of expressing multi-dimensional operators as one-dimensional. Take two-dimensional data, padded by zeros along the first axis,

$$\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline 1 & 2 \\ \hline 1 & 4 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \tag{2.10}$$

and perform a 2D autocorrelation to produce

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 4 & 9 & 2 \\ \hline 6 & 22 & 6 \\ \hline 2 & 9 & 4 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} . \tag{2.11}$$

We perform a helical transform on the original two-dimensional data by appending each column of numbers on the 1-axis of 2.10 and writing it as the one-dimensional vector,

| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 2 | 4 | 0 | 0 | 0 |. (2.12)

When we autocorrelate this 1D vector, we get the one-dimensional output,

| 0 | 0 | 0 | 4 | 6 | 2 | 0 | 0 | 0 | 0 | 9 | 22 | 9 | 0 | 0 | 0 | 0 | 2 | 6 | 4 | 0 | 0 | 0 |. (2.13)

By unwrapping this 1D autocorrelation back to two dimensions, we obtain the two-dimensional autocorrelation, 2.11, of the original data. In this way, we have performed a two-dimensional autocorrelation by doing a one-dimensional operation.

This same approach can be applied in any number of dimensions, with the length of the one-dimensional helical vector equal to the total number of elements in the entire multi-dimensional dataset. This method of transforming multi-dimensional autocorrelations to 1D autocorrelations applies as well to multi-dimensional convolution and deconvolution (Claerbout, 1998). In practice, the edge effects introduced by this transform will depend on which axes are wrapped using the helix, but can be mostly eliminated with adequate zero-padding.

The multi-dimensional equivalent of the PEF described in equation 2.9 has a particular shape, shown in Figure 2.1. This shape should ideally cover as much of the space as possible while remaining causal in all dimensions. In one dimension, this forward prediction-error filter has a leading unity value, i.e. the predicted point, that is based upon previous inputs along the solitary axis. In two dimensions, the filter is shown in Figure 2.1b. This 2D filter ideally spans previous columns as well as previous values on the column with the leading unity value, i.e. the predicted point. The leading value is not at the corner of the filter because we wish to predict the leading value by using as much of the previous data as possible, so not including data before the leading unity value on previous columns would result in a poorer prediction. In three dimensions this combination of causality and the desire to use as many previous inputs as possible produces the right-panel of Figure 2.1, where

along the third axis all planes before the leading predicted value are included in the prediction as well as all previous columns in the plane containing the predicted value.

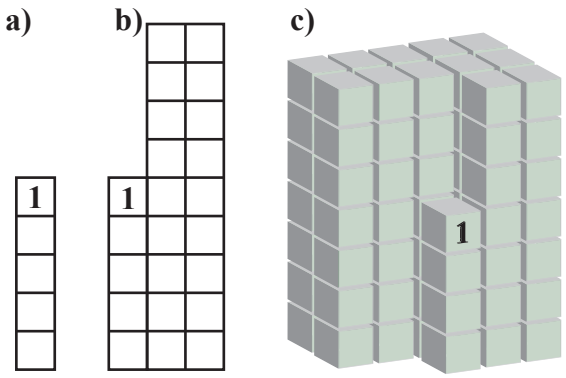

Figure 2.1: Examples of the structure of 1D, 2D, and 3D PEFs: a) five term 1D PEF; b) 23-term 2D PEF; c) 88-term 3D PEF. **NR** PEF/. helix

Problems with this method of performing correlation and convolution come from the boundaries between traces in the helical coordinate, which can be eliminated by use of adequate zero-padding. Helical boundary effects cause wrapping much like from a Fourier transform, except that the wrapping occurs on the subsequent trace attached to the helical trace instead of the same trace. This happens when the earlier lags of the filter lie on the original trace while later lags of the filter lie on the next trace wrapped on the helix. For convolution, this can be avoided by padding the end of the traces by the length of the filter, but for deconvolution the padding must be much greater, as the response of the filter is much longer than the length of the filter.

PEF ESTIMATION ON SYNTHETIC DATA

Let us illustrate PEF estimation on two-dimensional synthetic data that contain simultaneous planar events with two distinct slopes. The synthetic data in Figure 2.2a were created by first filtering a field of 256×256 normally distributed random numbers with a two-dimensional dip filter. This is repeated with a different slope for a second set of random numbers. Then each set of these data is independently

bandpass filtered such that the planar events that slope downward to the right have a higher passband than do the planar events sloping downward to the left. These two data sets are then summed.

These synthetic data are used as the input or training data, $\mathbf{d}$, in the PEF estimation (equation 2.9). Figure 2.2 shows the data $\mathbf{d}$ and residuals or prediction-errors, $\mathbf{r}_\mathrm{d}$, in equation 2.4 of both a single 10-element 1D column PEF estimated on the first trace and a 10×3 2D PEF estimated on the entire data set respectively, obtained by solving equation 2.9. The 2D PEF has 24 free coefficients, so the conjugate-direction solver was iterated until theoretical convergence at 24 iterations. As the data convolution matrix $\mathbf{D}$ has roughly 256×256 rows, this problem is overdetermined for the 24 unknowns.

The residual power, $\|\mathbf{r}_\mathrm{d}\|^2$, of the 1D PEF estimation, equivalent to convolution of the PEF with the training data, is reduced by 93 percent relative to that of the input data, or the data convolved with the initial guess of $\mathbf{Kf} = \mathbf{0}$, with the leading unity value fixed. Noticeable coherent dipping energy exists in the residual, however, so the output is not completely uncorrelated. The 2D PEF residual is still lower, with a 30 percent reduction compared to the 1D PEF, but, more important, it has an uncorrelated residual. The line graph in Figure 2.3 shows the residual norm as a function of iteration number in the conjugate-direction process. The 2D PEF has both a lower overall residual norm than that of the 1D PEF and more rapid convergence. The higher-dimensional PEF was better capable of capturing the entire inverse spectrum of the data. In particular, the three columns in the 2D PEF are able to capture two slopes, whereas the single column 1D PEF is unable to capture any slope. The conjugate-direction solver converged in fewer iterations (less than 10) than the theoretical number of 24, the total number of unknowns. Since the input data were generated consisted of events with just two constant slopes, it is not surprising that the PEF performed so well. If these data contained more slopes, a PEF with more columns would be required to capture the more complicated information in the data, with one additional column needed for each additional slope. Next, we review how to interpolate missing data using a PEF that has been acquired on training data.

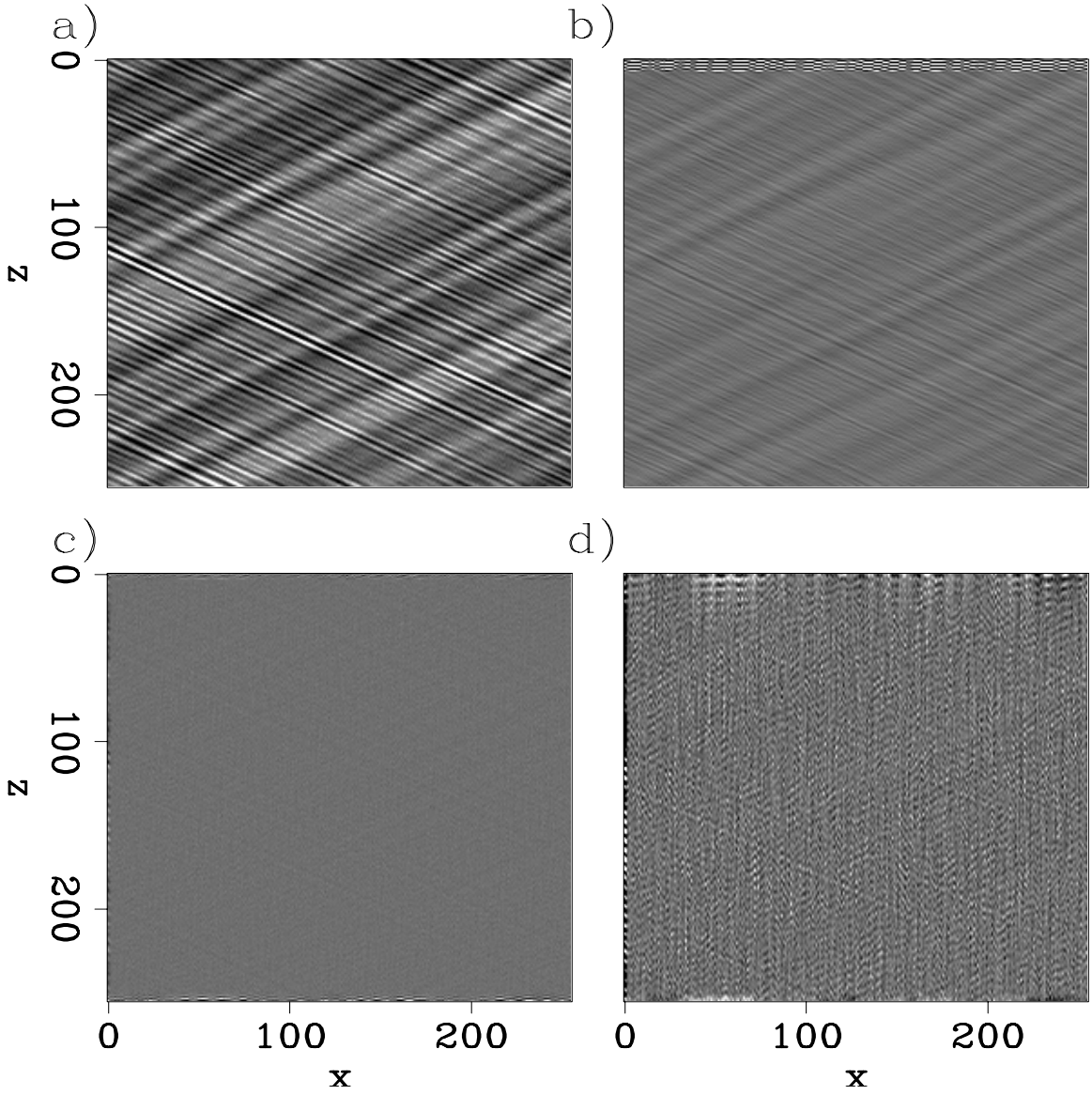

Figure 2.2: Estimation of a 2D PEF on 2D synthetic data with two dips. (a): original input training data; (b): residual ($\mathbf{r}_\mathrm{d}$) after PEF estimation using a single 10×1 vertical filter; (c): residual ($\mathbf{r}_\mathrm{d}$) for a 10×3 filter; (d): residual in (c), divided by the 1D PEF from (a) to highlight the spatial coherency of the residual. All images have the same amplitude scale. The 2D PEF more accurately predicts the input data. **ER** PEF/. planeest

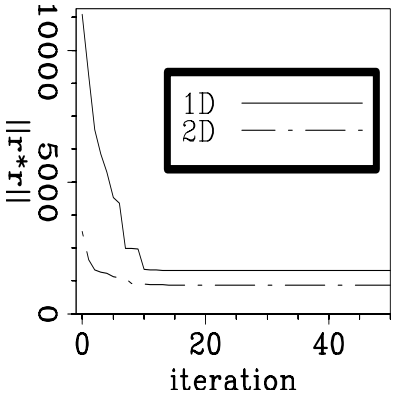

Figure 2.3: Residual power as a function of conjugate direction iteration for a 10-element 1D PEF and a 24-element 2D PEF. The 2D PEF converges more quickly and has a lower residual. **ER** PEF/. planeestcv

I then demonstrate the process for the same two-slope example, now with incomplete sampling.

INTERPOLATION WITH A PREDICTION-ERROR FILTER

Having seen how to compute a multi-dimensional prediction-error filter, let us now see how that filter is used to fill in missing data. Letting both the input sampled data and the output interpolated model be on the same grid, I again phrase the problem as a least-squares one solved using the method of conjugate-directions.

This interpolation method has two desired goals. The first is to honor the samples in the data, considering them as known and fixed in the interpolation process. The second goal is to create interpolated samples that, after combining with the sampled data, have the same multi-dimensional amplitude spectrum as the training data used in the PEF computation. Suppose for now that the fully-sampled original data are used as training data for the PEF. Then those same original data are sampled to produce under-sampled data with missing samples that will be interpolated using the previously-obtained PEF. To achieve the two desired goals we minimize the convolution of that PEF with all values, both sampled and not, of the interpolation result while forcing all sampled data points in the output interpolation to match the sampled data.

We start with a 1D example and an equation similar to equation 2.4, but now apply the known PEF as a convolutional matrix, $\mathbf{F}$, multiplied with the desired interpolated model $\mathbf{m}$ to produce a residual error in the interpolated model, $\mathbf{r}_\mathrm{m}$, which is therefore given by

$$\mathbf{r}_\mathrm{m} = \mathbf{F}\mathbf{m}. \tag{2.14}$$

Here $\mathbf{F}$ is the convolutional matrix containing the PEF that was computed in the previous step and is now held fixed. The $n_\mathrm{d} \times 1$ output interpolated model $\mathbf{m}$ consists of the fixed input (known) values and the values to be interpolated (unknown), and the subscript in $\mathbf{r}_\mathrm{m}$ distinguishes this residual from the residual in the previous data-fitting step.

We wish to find a least-squares solution for the interpolated values of $\mathbf{m}$, with the $\mathbf{F}$ matrix known. We first break up the $\mathbf{m}$ vector into unknown and known values using matrices $\mathbf{J}$ and $\mathbf{L}$, respectively. These matrices each have n_d columns and have a combined total of n_d rows. This approach is somewhat similar in action to that of the $\mathbf{K}$ matrix in the previous step, which isolated the leading unity value of the PEF. We then break up the $\mathbf{F}$ matrix into two narrower matrices, $\mathbf{F}_\mathrm{u}$ and $\mathbf{F}_\mathrm{k}$ that operate on the unknown and known values of $\mathbf{m}$, so we can rewrite equation 2.14 as

$$\mathbf{r}_\mathrm{m} = \mathbf{F}_\mathrm{u}\mathbf{J}\mathbf{m} + \mathbf{F}_\mathrm{k}\mathbf{L}\mathbf{m}. \tag{2.15}$$

Taking the L_2 norm of equation 2.14, differentiating with respect to the unknown values $\mathbf{J}\mathbf{m}$, and setting these equations to zero, gives

$$\mathbf{0} = \frac{\partial \|\mathbf{r}_\mathrm{m}\|^2}{\partial \mathbf{J}\mathbf{m}} = \mathbf{F}_\mathrm{u}^\dagger \mathbf{F}_\mathrm{u}\mathbf{J}\mathbf{m} + \mathbf{F}_\mathrm{u}^\dagger \mathbf{F}_\mathrm{k}\mathbf{L}\mathbf{m}. \tag{2.16}$$

The first term of this equation is the PEF convolved with the unknown output model points, since the known model points are removed, while the second term is the PEF convolved with the known points of the output model, since the unknown model points are in the first term. The second term in equation 2.16 is fixed throughout the minimization process, so we write it as a single known quantity $\mathbf{r}_0$ and replace it in

equation 2.16, giving

$$\mathbf{0} = \mathbf{F}_{\mathrm{u}}^{\dagger}\mathbf{F}_{\mathrm{u}}\mathbf{J}\mathbf{m} + \mathbf{F}_{\mathrm{u}}^{\dagger}\mathbf{r}_0. \tag{2.17}$$

We can then rearrange terms to place the unknown model points $\mathbf{J}\mathbf{m}$ on the left side of the equation, to get

$$\mathbf{J}\mathbf{m} = -(\mathbf{F}_{\mathrm{u}}^{\dagger}\mathbf{F}_{\mathrm{u}})^{-1}\mathbf{F}_{\mathrm{u}}^{\dagger}\mathbf{r}_0. \tag{2.18}$$

We solve these normal equations using a conjugate direction solver as in the solution to equation 2.7. If the norm of the residual in equation 2.16 was exactly zero, the value of $\mathbf{F}_{\mathrm{u}}\mathbf{J}\mathbf{m}$ would be the negative of $\mathbf{F}_{\mathrm{k}}\mathbf{L}\mathbf{m} = \mathbf{r}_0$. I typically use as the starting guess zeroes for the unknown values so that the $\mathbf{F}_{\mathrm{u}}\mathbf{J}\mathbf{m}$ term would also be zero and the residual would be $-\mathbf{r}_0$, which would be far from an optimal solution. As the conjugate direction solver iterates, the solution for $\mathbf{J}\mathbf{m}$ changes the residual series from $-\mathbf{r}_0$ to something much closer to zeroes.

The previous theory was described for one dimension, but equation 2.18 can be extended to multiple dimensions using the helical coordinate, as was described for the PEF estimation process. Here the multi-dimensional PEF would be applied to the multi-dimensional data using helical convolution to produce a multi-dimensional output.

Interpolation example on synthetic data

Having defined the second step of the interpolation procedure, using a PEF estimated from training data to create missing data, let us now test this second step on a sampled version of the 2D synthetic data from Figure 2.2 in the previous section. We first compute the PEF from training data, which are the same fully-sampled data, shown in Figure 2.4a. We sample these data in a checkerboard-like pattern, so that half of the data is missing. Four 32×32 point squares are sampled along each axis, as shown in Figure 2.4b, with the missing data values set to zero.

The sampled version of the data, with zeroes in the place of missing data, is the fixed $\mathbf{L}\mathbf{m}$ term in equation 2.16, meaning that the $\mathbf{J}\mathbf{m}$ term in this case is all

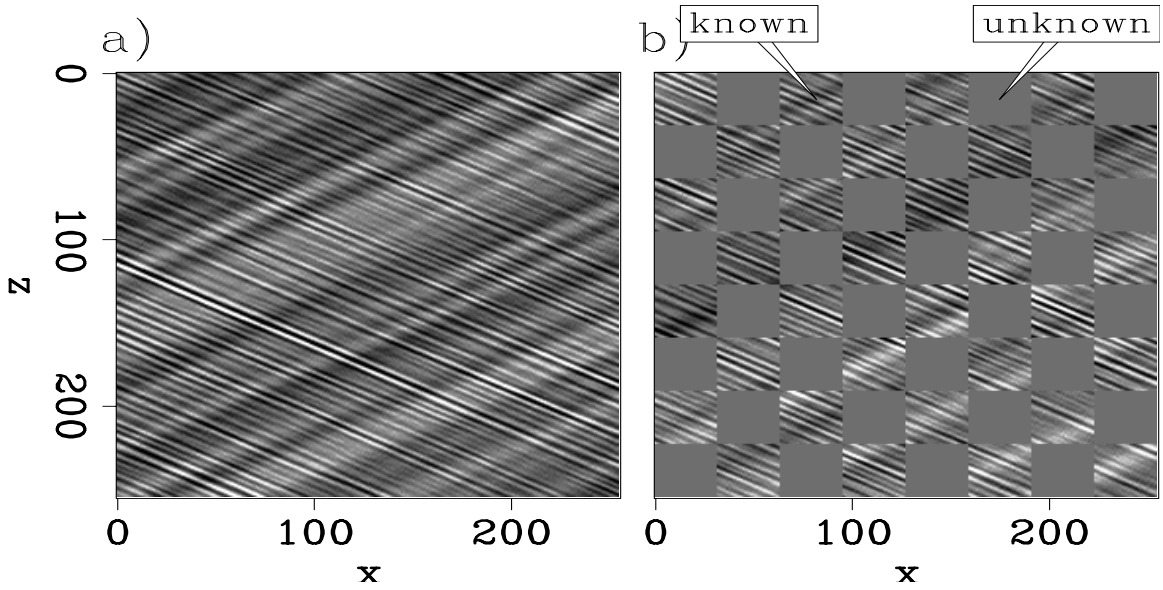

Figure 2.4: Synthetic 2D data. (a): original data, identical to that in Figure 2.2a. (b): the original data with 50 percent of the values missing in a checkerboard pattern of 32×32 cells. **ER** PEF/. dataholeann

zeroes. When the fully-sampled data are filtered by the 10×3 two-dimensional PEF estimated in the previous section, the filtered output is largely uncorrelated, as shown in Figure 2.5a, which has the amplitude scale magnified by a factor of 20 compared to that in Figure 2.4a. On close inspection the filtered output does show remnants of the two slopes in the input data, particularly for the higher-frequency event sloping downward to the right. The sampled data, filtered with the same two-dimensional PEF and shown in Figure 2.5b, have the same largely random character as do the filtered fully-sampled data in Figure 2.5a within the sampled regions, but near the boundaries between the sampled and unknown portions of the data in the filtered output amplitudes are relatively large and are correlated. Figure 2.5b can also be viewed as an image of $\mathbf{r}_0$. Solving equation 2.17 should produce an interpolated output that when filtered will be much more like Figure 2.5a than Figure 2.5b because the interpolated data should be the negative of the correlated (and therefore predictable) portions of $\mathbf{r}_0$.

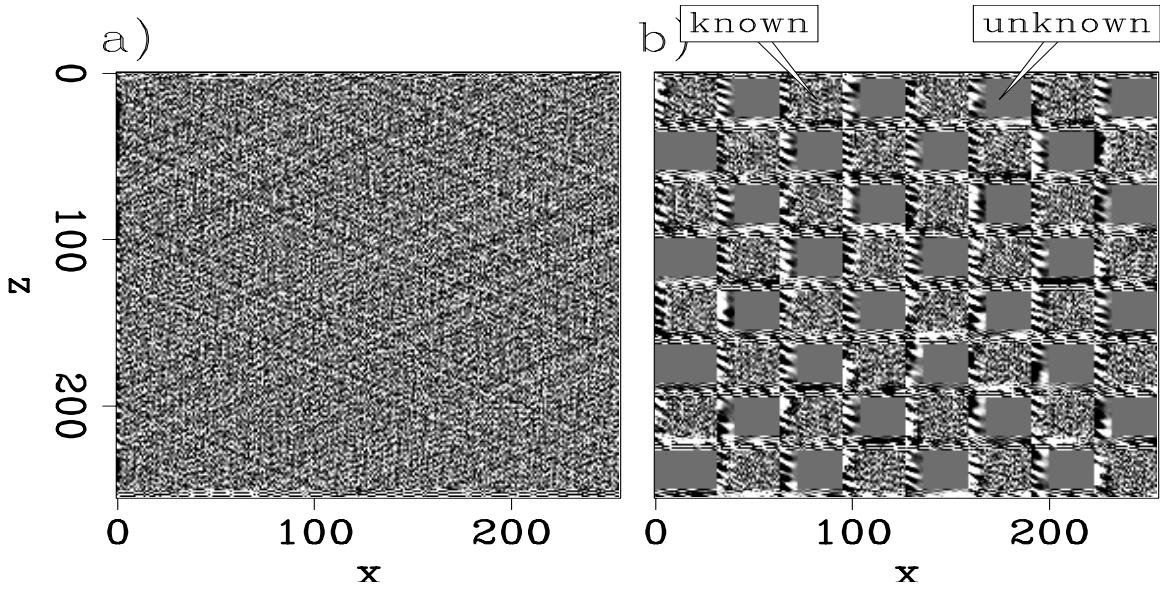

Figure 2.5: Synthetic 2D data shown in Figure 2.4 after filtering with a 10×10 PEF. (a): filtered original data. (b): filtered sampled data, $\mathbf{r}_0$ in equation 2.18. The amplitudes are magnified by a factor of 20 compared to those in Figure 2.4. The boundaries between sampled and unknown data have a large prediction error, because the drop from known data to the zeros not match values that would be predicted by the PEF. **ER** PEF/. dataholefiltann

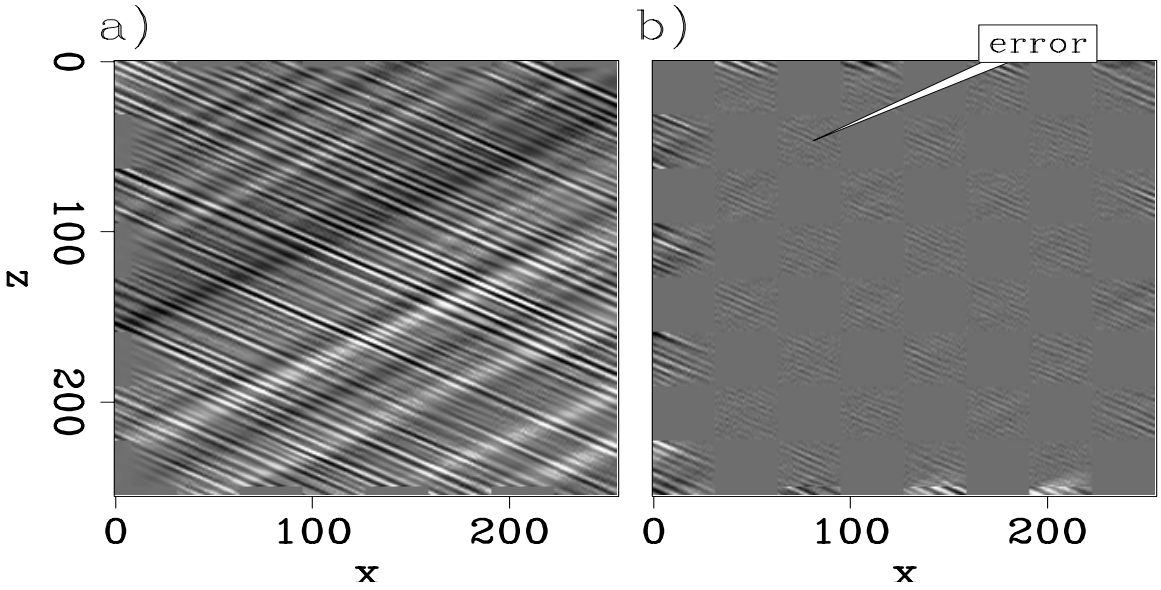

Figure 2.6: 2D Synthetic data in Figure 2.4 interpolated using a 2D PEF. (a): interpolated result. (b): difference between interpolated and original fully-sampled data. The interpolation captures the slopes of the training data, with the amplitudes under-predicted at the edges of the data. **ER** PEF/. dataholeinterpann

I stopped the conjugate-direction solver for $\mathbf{m}$ in equation 2.18 after 300 iterations. If this interpolation problem had been solved with explicitly constructed matrices, it would have been much more expensive because the $\mathbf{F}_{\mathrm{u}}^{\dagger}\mathbf{F}_{\mathrm{u}}$ matrix is sparse, with over one billion elements of mostly zeroes. The iterative approach is, in comparison, much more efficient, allowing solution of much larger problems.

The interpolation result is shown in Figure 2.6a, while Figure 2.6b shows the error between the interpolated result on the left and the original fully-sampled data in Figure 2.4a. The interpolated data have slopes that match those of the planar events in the fully-sampled data, and the locations of the portions that have been interpolated are not obvious. The difference panel in Figure 2.6b shows large differences near the edges of the image and relatively small differences in the interpolated regions in the interior of the image. The errors around the edges of the image that were interpolated arise because the PEF is extrapolating instead of interpolating the data there. In the

interior of the image we see that amplitudes of interpolated values are slightly underpredicted by up to roughly 5 percent. The reason for the slight amplitude differences between the interpolated and original data away from the boundaries is more easily seen by filtering the data and the difference with the PEF; the results are shown in Figure 2.7.

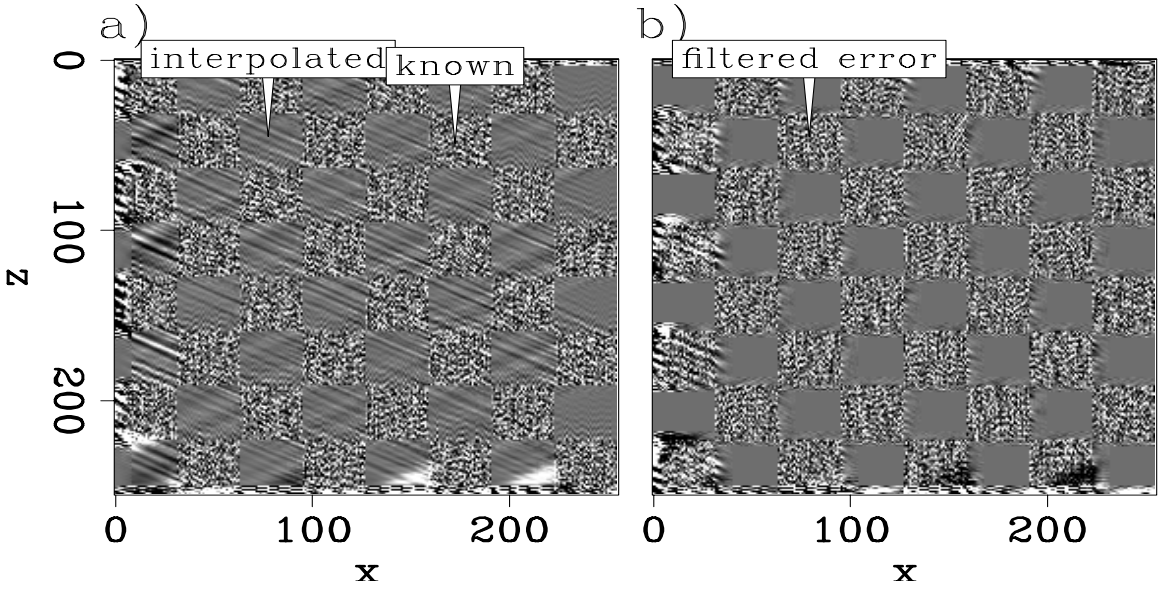

Figure 2.7: Interpolation and interpolation error, the difference between the interpolated data and the originally fully-sampled data, filtered with the PEF. (a): filtered interpolation result. (b): filtered difference between interpolation and original fully-sampled data. The random noise present in the filtered original data (Figure 2.5a) is not present at the interpolated values. **ER** PEF/. dataholeinterpfiltann

The large coherent output near the boundaries in the filtered sampled data (Figure 2.5b) are gone, and the random noise present in the filtered original data (Figure 2.5a) is not present in the interpolated values of Figure 2.7a. This lack of noise in the interpolated values signifies the slightly lower amplitude in the result, which is equivalent to random numbers filtered by the PEF. This random noise is not predictable and hence is not introduced to the interpolation, as the starting solution of the solver is zero-values, so this unpredictable random noise that is present in the original data would increase the residual and hence is not introduced. The amplitude of this result

could be corrected by introducing random noise into the filtered residual in the interpolated areas. Different realizations of random noise would produce different results, providing equally-probable interpolation results (Clapp, 2001).

All of the results here are predicated on having correct training data. In this stationary example, if there were sufficient contiguous data adjacent to the missing data, they could be used as training data. In this case, the checkerboard pieces are too small to estimate a PEF on but examples elsewhere (Claerbout, 2004) show how this is done.

Next we examine what happens when the training data used in the first step of PEF estimation has an autocorrelation that differs from that of the ideal data, i.e., the data in 2.4a from which we had obtained the checkerboard sampled data in Figure 2.4b.

Interpolation with imperfect training data

Given that the fully-sampled data here were provided as training data to the PEF estimation, and that the data consisted of planar features with just two slopes, it might be no surprise that the interpolation result in the previous section was so accurate. Now we examine what happens when the training data for the PEF differs from the ideal. Two of the ways the training data can differ from the ideal are in phase and amplitude scale. Suppose first, however, they have statistically the same (scaled) two-dimensional autocorrelation. Figure 2.8a is the original sampled data, 2.8b contains the training data that have been generated with the same dip filters and bandpass filtering as those for the fully-sampled data used to create Figure 2.8a, but different random numbers were used before filtering, so the phases of the planar events are different. The data have also been amplified by a factor of 100. We first estimate a PEF on the imperfect training data in Figure 2.8b, and then use the PEF to interpolate the checkerboard holes in 2.8a. The interpolated result in Figure 2.8c shows that these differences in the training data used to estimate the PEF in equation 2.9 made little difference in the PEF and less than 5 percent difference on

the interpolated result. The PEF is insensitive to both the phase and the amplitude of the data; it depends on only the normalized autocorrelation of the data.

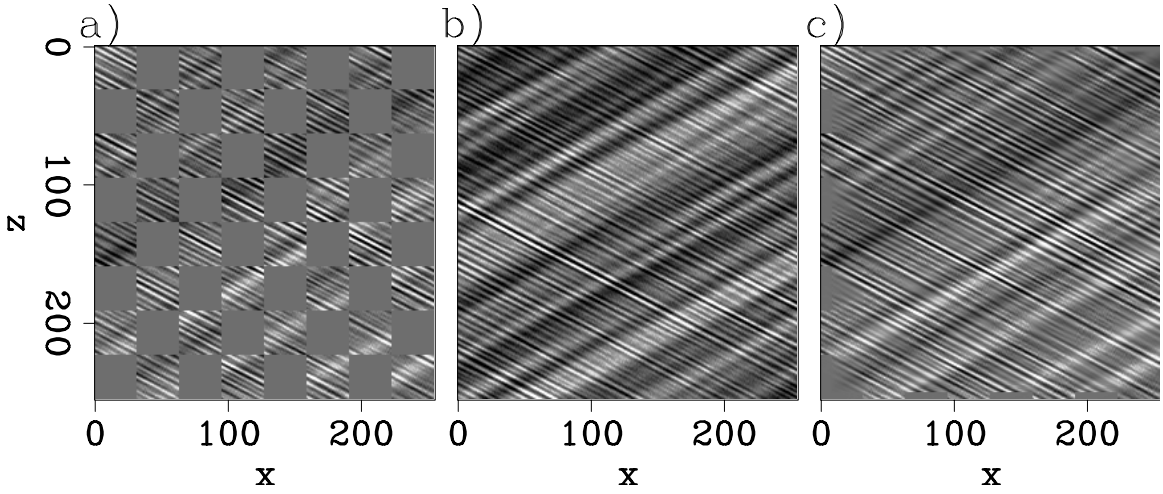

Figure 2.8: Interpolation with training data with different amplitude scale and phase. (a): sampled data. (b): training data with different phase and 100 times the amplitude of the ideal training data set in Figure 2.4a. (c): interpolated result. The interpolation is almost identical to that in Figure 2.6a based on using ideal training data. **ER** PEF/. statphasepeffill

Figure 2.9b contains training data that have two slopes that differ slightly from those in the previous training data as well as from those in the sampled data. The slope of features that are downward to the right is 15 degrees greater than in the data to be interpolated, and that of features that are downward to the left is 15 degrees less. Using these data as the training data for a PEF that is used to again interpolate the sampled data in Figure 2.9a produces the result in Figure 2.9c. The lower-frequency event (downward to the left) is acceptably interpolated, while the higher frequency one is not properly interpolated, as seen in the comparison with the result in Figure 2.8c. Use of an erroneous slope in the training data causes less degradation of the interpolation for the lower-frequency data.

Finally, suppose the slopes present in the training data do not at all match those in the original data that were sampled. The training data shown in Figure 2.10b do not in the least exemplify the original data in Figure 2.4a that have been sampled into the

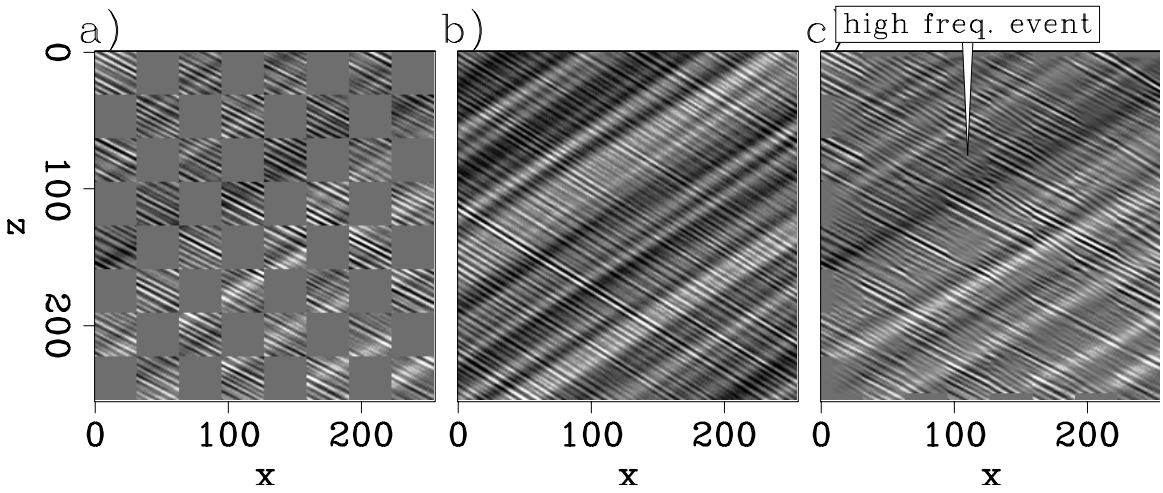

Figure 2.9: Interpolation with training data with different amplitude scale, phase, and slopes. (a): sampled data. (b): training data with slopes 15 degrees less than and greater than the original slopes. (c): interpolated result. The interpolation of the low-frequency planar event is reasonably good, but that of the higher-frequency event is not. **ER** PEF/. statmediumpeffillann

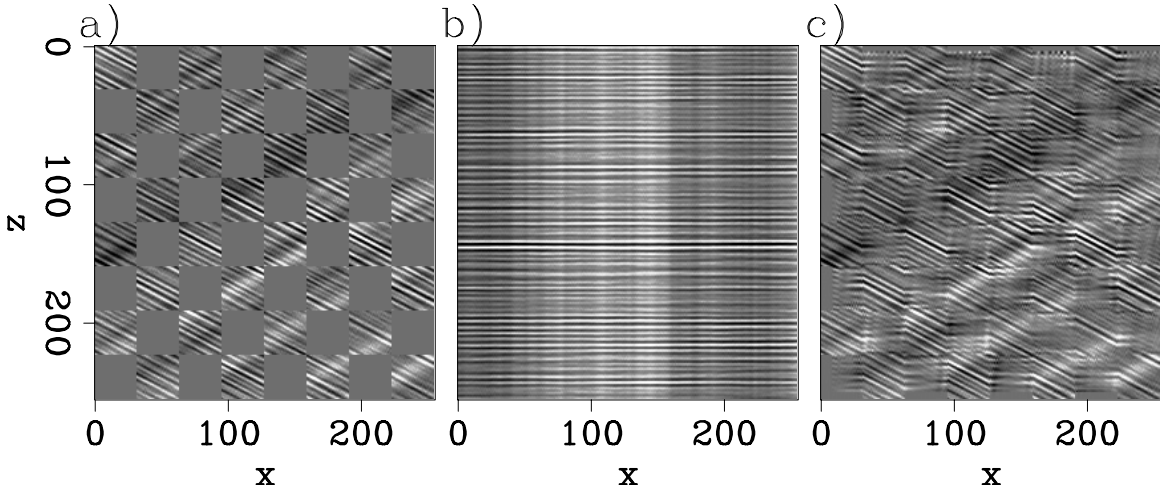

Figure 2.10: Interpolation with training data with slopes that are wildly different from those in the ideal training data. (a): sampled data. (b): training data with vertical and horizontal slopes. (c): interpolated result. The interpolation is hopelessly incorrect. **ER** PEF/. statbadpeffill

checkerboard pattern of gaps seen in Figure 2.10a. As a result, the PEF is significantly altered, and, when applied in the second step, produces the faulty patchwork quilt interpolation shown in Figure 2.10c. The interpolated data match not at all the fully-sampled data, and are obviously incorrect. While the training data can differ in amplitude and phase from those in the data that need to be interpolated, they must reasonably reflect the character of the data, in particular their multi-dimensional autocorrelation.

In summary, data are interpolated as two linear least-squares problems. We first capture the inverse multivariate amplitude spectrum of the training data in a compact prediction-error filter, then use that PEF to filter the missing data while simultaneously matching the data at known locations. The method of conjugate directions is used to solve both problems, largely because of the relatively small memory requirements of the method: only the data vector rather than the full matrix is held in memory to apply forward and adjoint convolution. This interpolation method is successful on the test data comprised of multiple stationary dips as long as the training data reasonably mimic the true data. Next, I show how this method can fail when the dip of the data varies as a function of position, even when the training data are ideal.

NONSTATIONARY PREDICTION-ERROR FILTERS

In the previous section, I used a PEF to interpolate a combination of planar events with constant slopes. Seismic data are not composed solely of planar events, but instead of various curved ones that gradually change in slope as a function of position. An example of this is the three-dimensional quarter-dome synthetic data (Claerbout, 2004) shown in Figure 2.12a. The three joined panels in the frame are the three unfolded faces of a (x, y, z) data cube, with the top panel a depth slice (z is constant), the right panel a cross-line section (x is constant), and the left panel an in-line section (y is constant). The lines on each of the panels correspond to the locations where the other slices intersect the cube. The data contain horizontal layers at shallow

depths, an anticline structure at middle depths, and layers with constant dip below. The anticline structure has both dips that are gradually varying in the upper-right portions of the depth sections, and rapidly varying toward the left.

We next see the action of the PEF-based approach on the data shown in the Figure 2.12a. I first estimate a $5 \times 5 \times 5$ 3D PEF on the $200 \times 100 \times 50$ data by using 113 iterations (equal to the number of unknown filter coefficients) of a conjugate-direction solver for equation 2.9. The residual of this process, obtained by convolving the PEF with the training data, is shown in Figure 2.12b, where I divided the residual by a 1D PEF estimated on the data to highlight errors in spatial prediction. Based on the small relative amplitudes in the filtered residual, we judge that the PEF accurately predicts both the shallow horizontal layers and the deep constant-slope layers, but is unsuccessful in interpolating the anticline structure at intermediate depths. The slowly varying dips toward the right of the depth slices are slightly more accurately predicted than are the rapidly varying ones toward the left.

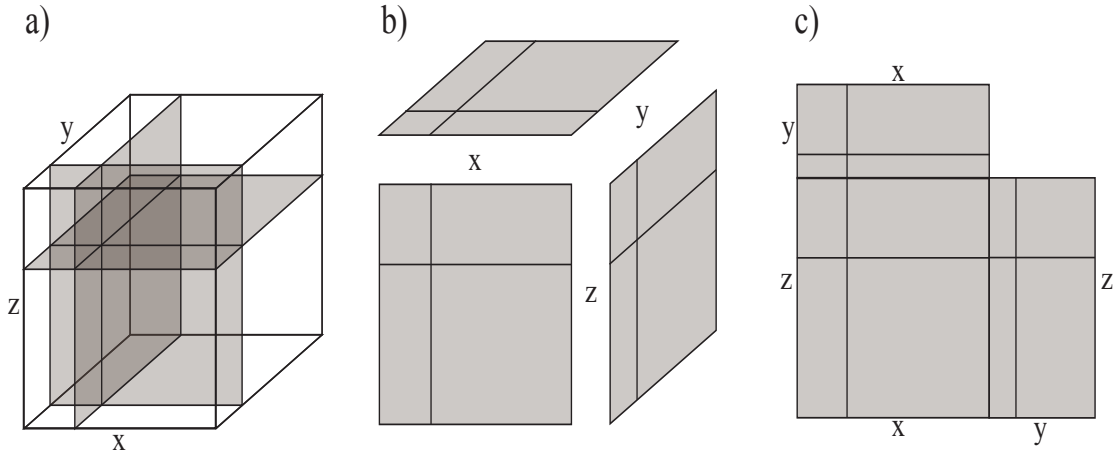

Figure 2.11: A schematic of the 'cubeplot' figures used throughout this thesis. Three input planes from within a cube (a) are separated (b) and placed side-by-side, where the lines on each image denote the intersections of the other slices through that image.
NR PEF/. cubeplot

With a PEF estimated on the fully-sampled data, we next attempt to use that PEF to interpolate a sampled version of the quarter-dome data. Figure 2.13a shows

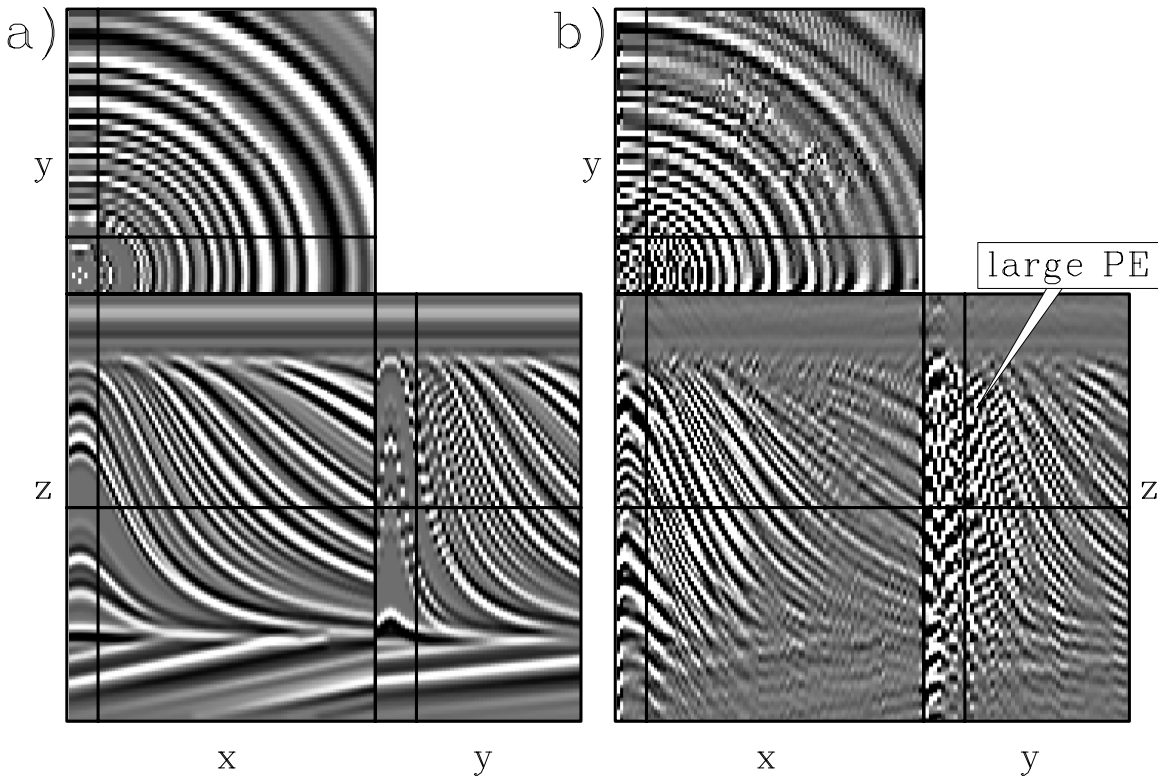

Figure 2.12: Estimating a PEF on the three-dimensional quarter-dome data. (a): original fully-sampled data. (b): filtered residual (i.e. prediction-error) from convolving a PEF with the original data, then dividing by a 1D PEF estimated on (a). Only the stationary upper and lower energy is predicted. **ER** PEF/. qdomestatestann

a quite poorly sampled version of the data in Figure 2.12a wherein a small percentage of the traces in the full data set (only 20 percent) were sampled at random. We then use the PEF estimated from the fully-sampled data to interpolate the missing data, using equation 2.18. The interpolation result is shown in Figure 2.13b. Interpolating the extremely poorly sampled data with the PEF gives a result that generally does reflect the structure of the ideal training data used, but nevertheless is severely flawed as representing the original data before having been sampled. The areas with rapidly varying dips are especially poorly interpolated, but the interpolation is flawed as well over the more stationary portions of the data. The failure does not reside in the design of the PEF, as evidenced by the small PEF residual in Figure 2.13b. Despite the use of the ideal training data (Figure 2.12a), the assumption of data stationarity (i.e., a single autocorrelation for the entire data set) is inadequate for these poorly-sampled data. Two ways of dealing with this nonstationarity are (1) treat the data in (overlapping) patches small enough that events are approximately linear within them, and (2) solve for a single nonstationary PEF. Both are described next.

One way to deal with curved data is to consider the data as a collection of small overlapping patches, each patch small enough so that the curved events appear to be straight within a patch. Using this approach, the data in Figure 2.12 are broken up into smaller overlapping patches. By assuming that the data in each of these patches are stationary, we apply the same stationary interpolation approach described previously on each patch independently, with a different PEF for each patch. We then reassemble the patches to form the interpolated output. Figure 2.14 shows this approach. Figure 2.14a is a checkerboard plot depicting the size of the 216 overlapping $50 \times 25 \times 15$ patches on the $200 \times 100 \times 50$ data, the patches were alternating in black or white, with the gray indicating where the patches overlap. Figure 2.14b is the residual of the $5 \times 5 \times 5$ PEF estimation performed on 216 patches that are solved separately and then reassembled. The patches are padded on all axes to account for PEF edge effects, with the padding discarded before reassembly. The estimation is better, but only marginally so, than the stationary result shown in Figure 2.12b, but again produces the largest residual in the region with the most spatial variability, and the boundaries between patches.

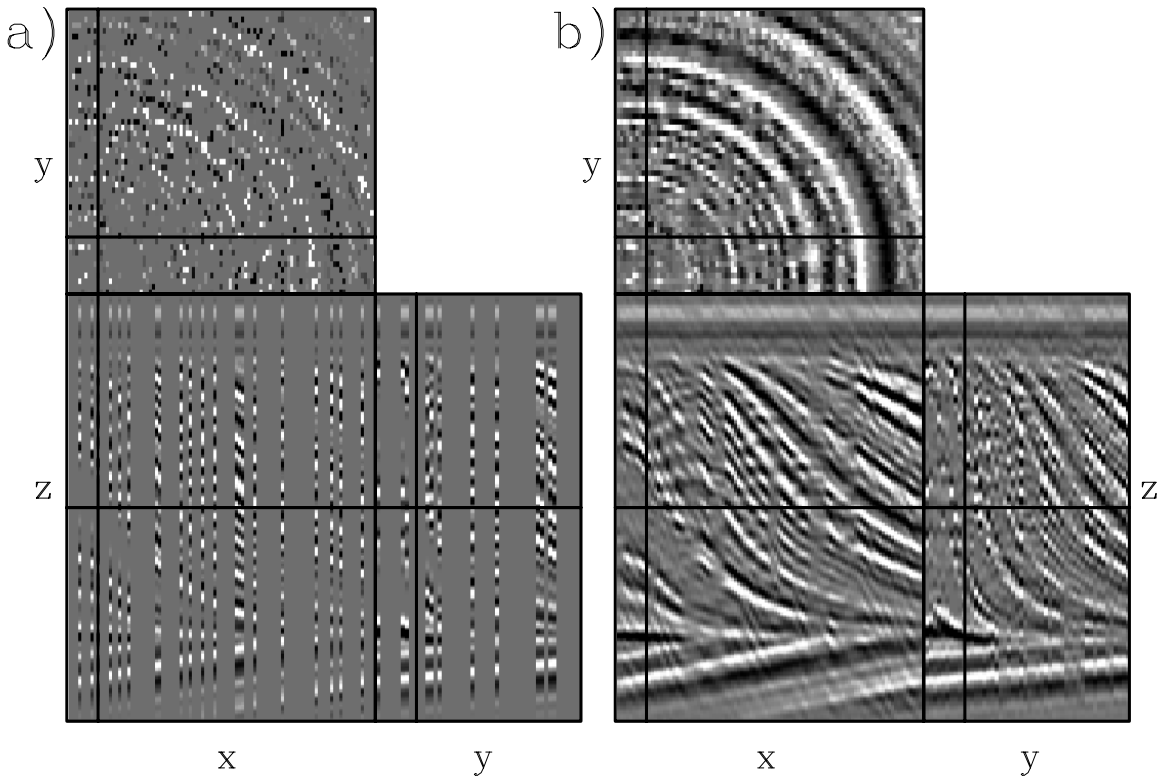

Figure 2.13: Interpolating the quarter-dome data from Figure 2.12 with a 3D PEF. (a): randomly-sampled traces with only 20 percent of data present. (b): data from (a) interpolated with a 3D PEF estimated on the fully-sampled data. **ER** PEF/. qdomestatfillann

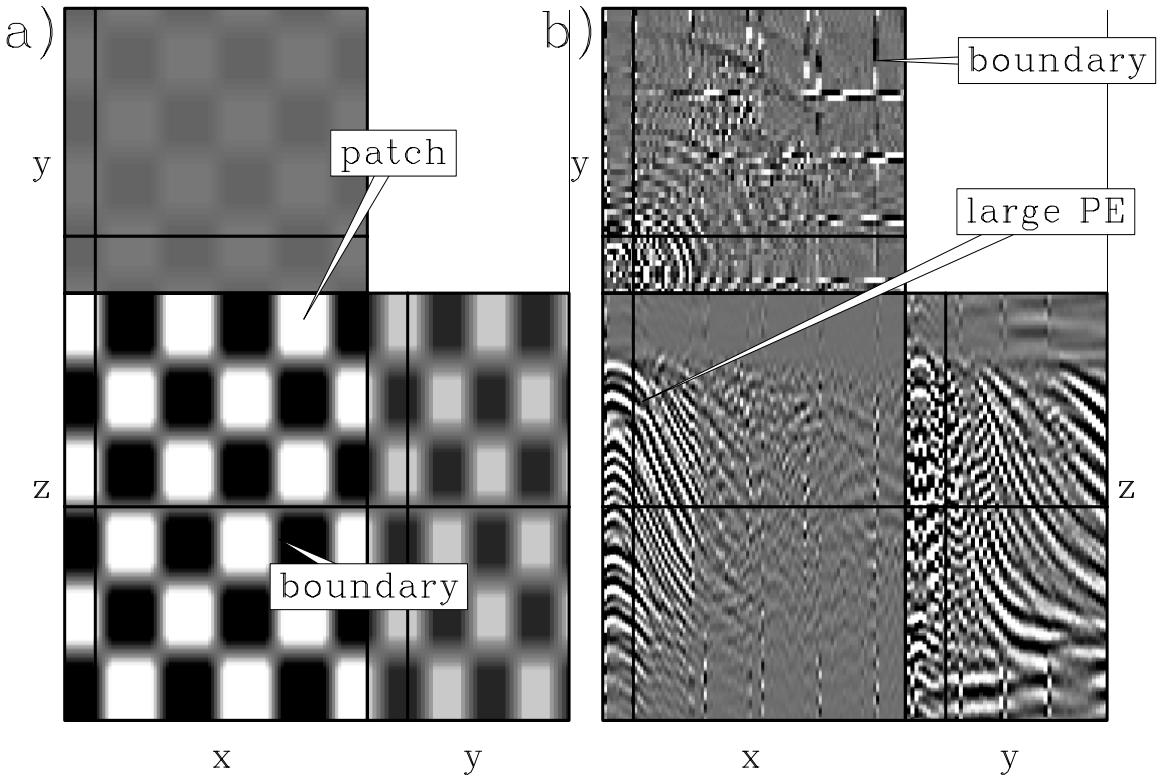

Figure 2.14: Multiple PEFs estimated on the quarter-dome synthetic in patches. (a): a representation of the size of the patches used, depicted by alternating black and white patches and gray representing overlap between patches. (b): patched filtered residual of PEF estimation for the 216 different patches PEFs shown in (a). The patched approach is only slightly better than a single PEF for the entire data set. It still performs poorly where the slope changes most rapidly. **CR** PEF/. patchpefestann

The PEFs estimated on the fully-sampled data in Figure 2.12a are now used to interpolate the missing data, with the PEF for each patch used to interpolate within that same sampled patch. In Figure 2.15b, the interpolation result, shows that this method performs not much better than did that of using a single PEF (Figure 2.13b). It still fails wherever the data sampling was poorest. This is for two main reasons. First, the patches are too large for the rapidly changing slopes at the peak of the anticline. The obvious solution to this problem would be to reduce the patch size, but the patches need to be large enough to provide enough fitting equations for the PEF estimation and to span large regions of missing data. This is related to the second problem, the boundary problems that appear during the second step, equation 2.18. In the previous stationary example in Figure 2.6b the interpolated data were satisfactory but were poor near the boundaries where the result was extrapolated instead of interpolated. The relatively small patches used result in many more boundaries where the PEF does not perform well. These problems are largely solved by using a smoothly nonstationary prediction-error filter (Clapp et al., 1999; Crawley, 2000), with filter coefficients that vary as a function of space. I describe this nonstationary PEF approach next.

Nonstationary PEF estimation

Use of a single PEF for the entire data set does not adequately describe data with locally changing slopes, while using many PEFs for smaller patches in separate problems partially addresses the changing slopes but has issues with the size of the patch and boundary effects. I estimate a nonstationary prediction-error filter by solving a single least-squares problem, in which instead of estimating a series of prediction-error filters that are each tied to a specific patch of data, I estimate one large nonstationary prediction-error filter that varies smoothly as a function of space to account for the varying slopes present in the data. The equations used to estimate a nonstationary PEF have a form similar to that of the stationary case in equation 2.4 except that the structure of the matrices involved differ, as described in this synopsis of Guitton (2003). The PEF vector $\mathbf{f}$, comprised of the unknown filter coefficients and the

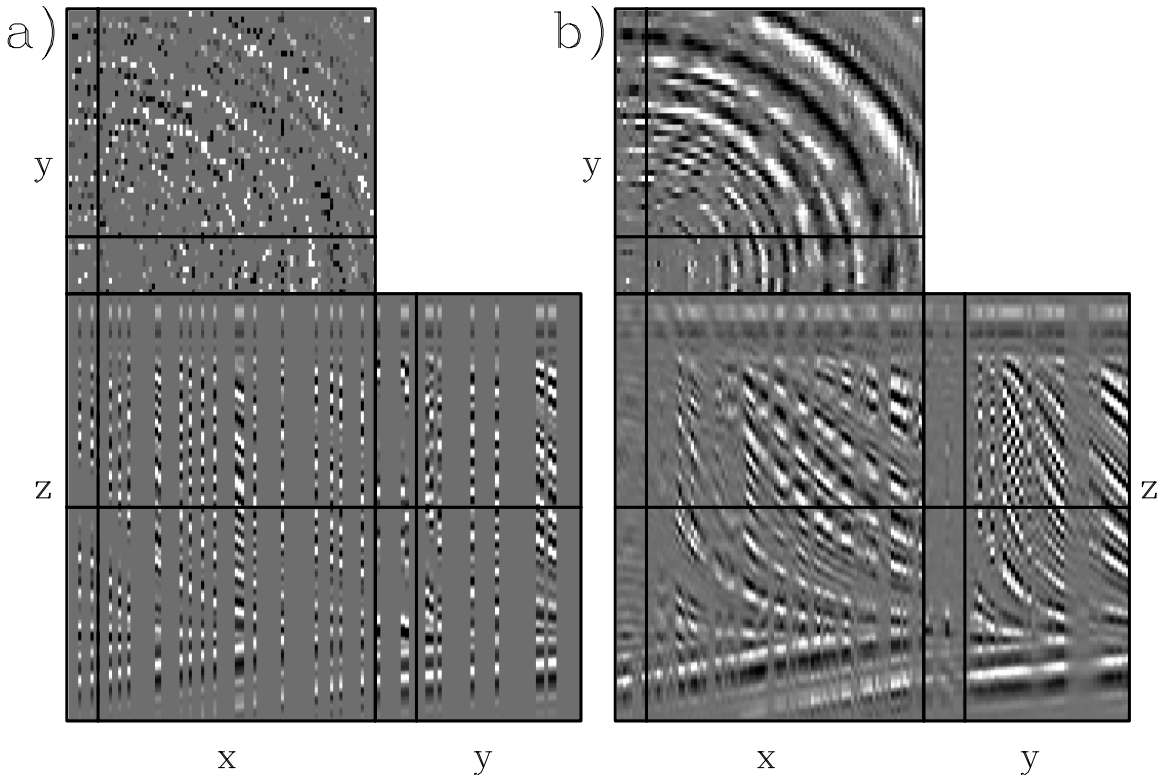

Figure 2.15: The quarter-dome synthetic interpolated in patches. The three faces correspond to three slices through the 3D cube. (a): input sampled data. (b): the interpolated result. The result is still poor due to the rapidly changing dips combined with the sparse sampling of the data, which increased boundary issues from the patches, even with overlap. **CR** PEF/. patchpeffillann

leading unity value is now the nonstationary filter, $\mathbf{f}_{\rm ns}$, containing a separate series of filter coefficients optimized for each data point. Other matrices involved in nonstationary PEF estimation also follow this convention of the $_{\rm ns}$ subscript indicating the nonstationary counterpart to the stationary PEF problem. The filter vector $\mathbf{f}_{\rm ns}$, instead of having $n_{\rm f}$ filter coefficients, now contains $n_{\rm f} \times n_{\rm d}$ coefficients, where $n_{\rm f}$ is the number of multi-dimensional filter coefficients and $n_{\rm d}$ is the total number of data points. The vector $\mathbf{f}_{\rm ns}$ is structured as a concatenation of the filters for each data point (Margrave, 1998), so

$$\mathbf{f}_{\rm ns} = \left(\begin{array}{c} \mathbf{f}_1 \\ \hline \mathbf{f}_2 \\ \hline \vdots \\ \hline \mathbf{f}_{n_{\rm d}} \end{array} \right) \quad \text{with } \mathbf{f}_k = \left(\begin{array}{c} 1 \\ f_{2,k} \\ f_{3,k} \\ \vdots \\ f_{n_{\rm f},k} \end{array} \right). \tag{2.19}$$

Here the horizontal lines distinguish the distinct sets of PEF coefficients for each data point. The first subscript of f is the coefficient index for the output data point, and the second subscript labels the output data point. The definition of the residual we wish to minimize for the nonstationary filter is similar to those of the stationary case in equation 2.4,

$$\mathbf{r}_{\rm d} = \mathbf{DKf} + \mathbf{d}, \tag{2.20}$$

so that

$$\begin{aligned} \min_{\mathbf{f}_{\rm ns}} & \|\mathbf{r}_{\rm d}\|^2 + \epsilon^2 \|\mathbf{r}_{\rm f}\|^2 \\ \mathbf{r}_{\rm d} &= \mathbf{D}_{\rm ns}\mathbf{K}_{\rm ns}\mathbf{f}_{\rm ns} + \mathbf{d} \\ \mathbf{r}_{\rm f} &= \mathbf{R}\mathbf{K}_{\rm ns}\mathbf{f}_{\rm ns}. \end{aligned} \tag{2.21}$$

Following the same differentiation with respect to unknown filter values as in the stationary case, this system may be rewritten in a single line as

$$\mathbf{K}_{\rm ns}\mathbf{f}_{\rm ns} = -(\mathbf{D}_{\rm ns}^\dagger \mathbf{D}_{\rm ns} + \epsilon^2 \mathbf{R}^\dagger \mathbf{R})^{-1} \mathbf{D}_{\rm ns}^\dagger \mathbf{d} \tag{2.22}$$

The first equation in 2.21 denotes that we wish to minimize both residuals $\mathbf{r}_\mathrm{d}$ and $\mathbf{r}_\mathrm{f}$, where $\mathbf{r}_\mathrm{f}$ is scaled by a trade-off parameter, ϵ. The second equation in 2.21 is similar to equation 2.4. Now a third equation for the model residual, $\mathbf{r}_\mathrm{f}$, is added to regularize the problem, since the number of unknown filter coefficients has increased from n_p to $n_\mathrm{p} \times n_\mathrm{d}$ in equation 2.21 while the number of fitting equations is still roughly n_d, making the problem under-determined and in need of regularization.

Just as for the nonstationary filter vector, the nonstationary convolution matrix $\mathbf{D}_\mathrm{ns}$ is also n_d times larger than its stationary counterpart, so that it can map the larger number of nonstationary filter coefficients ($n_\mathrm{p} \times n_\mathrm{d}$) to the output space of length n_d, giving a matrix of $n_\mathrm{d} \times (n_\mathrm{p} \times n_\mathrm{d})$. $\mathbf{D}_\mathrm{ns}$, like $\mathbf{f}_\mathrm{ns}$, is a concatenation of component matrices, $\mathbf{D}^i_\mathrm{ns}$, each corresponding a convolution matrix for the i^{th} output point, so

$$\mathbf{D}_\mathrm{ns} = \left[\begin{array}{c|c|c|c} \mathbf{D}^1_\mathrm{ns} & \mathbf{D}^2_\mathrm{ns} & \dots & \mathbf{D}^{n_\mathrm{d}}_\mathrm{ns} \end{array} \right], \tag{2.23}$$

$$\mathbf{D}_\mathrm{ns} = \left[\begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & \cdots & 0 \\ d_2 & d_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n_\mathrm{p}} & d_{n_\mathrm{p}-1} & \cdots & d_1 \end{pmatrix} \right], \tag{2.24}$$

where each sub-matrix $(\mathbf{D}^1_\mathrm{ns}, \mathbf{D}^2_\mathrm{ns}, \dots, \mathbf{D}^{n_\mathrm{d}}_\mathrm{ns})$ has $n_\mathrm{d} \times n_\mathrm{p}$ elements. Each sub-matrix $\mathbf{D}^i_\mathrm{ns}$ in $\mathbf{D}_\mathrm{ns}$ contains only one row of nonzero elements, so that only the first row of $\mathbf{D}^1_\mathrm{ns}$ is nonzero, only the second row of $\mathbf{D}^2_\mathrm{ns}$ is nonzero, and so on. This maps each series of data points to a set of nonstationary filter coefficients for each output point. In reality, I choose the nonstationary PEF, $\mathbf{f}_\mathrm{ns}$, such that it does not have unique coefficients for each output data point, but instead has the same coefficients over a small region in order to reduce the memory requirement. For example, reusing the same filter coefficients for two adjacent output data points would reduce the size of the $\mathbf{f}_\mathrm{ns}$ vector by half. This alters the matrix $\mathbf{D}_\mathrm{ns}$ so that the component matrices $\mathbf{D}^i_\mathrm{ns}$ contain two rows of nonzero coefficients instead of one, and there would be half as many component matrices. The matrix $\mathbf{K}$ that constrains the first filter coefficient to 1 in equation 2.4 is replaced by a $\mathbf{K}_\mathrm{ns}$ that is tailored to isolate the constrained

coefficients of the much larger nonstationary filter vector. The matrix now is

$$\mathbf{K}_{\text{ns}} = \left(\begin{array}{c|c|c|c} \mathbf{M} & \mathbf{0} & \mathbf{0} & \cdots \\ \hline \mathbf{0} & \mathbf{M} & \mathbf{0} & \cdots \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{M} & \cdots \\ \hline \vdots & \vdots & \vdots & \ddots \end{array} \right), \tag{2.25}$$

where

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{2.26}$$

Each component matrix $\mathbf{M}$ has $n_{\text{p}} \times n_{\text{f}}$ elements and corresponds to a set of PEF coefficients for a single data point or small region. There are n_{d} of these component matrices on the diagonal of the $\mathbf{K}_{\text{ns}}$ matrix, for a total size of $(n_{\text{d}} n_{\text{p}}) \times (n_{\text{d}} n_{\text{f}})$ elements. Multiplying the removed first row with the leading unity value in $\mathbf{K}_{\text{ns}}$ with $\mathbf{D}_{\text{ns}}$ produces a data vector of length n_{d}, as in the stationary case, and separates the unknown nonstationary PEF coefficients from the known leading-unity values.

Now let us focus on the second term of equation 2.21, the regularization term. I regularize the filter by minimizing the differences between the filter coefficients of filters that are adjacent in space. That is, when we look at the vector $\mathbf{f}_{\text{ns}}$, the elements in consideration are n_{p} values apart along the vector, the distance between horizontal lines in the description of $\mathbf{f}_{\text{ns}}$ in equation 2.19. When written for a 1-D example the matrix $\mathbf{R}$ is

$$\mathbf{R} = \left(\begin{array}{c|c|c|c} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \cdots \\ \hline \mathbf{0} & \mathbf{I} & -\mathbf{I} & \cdots \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\ \hline \vdots & \vdots & \vdots & \ddots \end{array} \right), \tag{2.27}$$

where $\mathbf{I}$ is an $n_{\text{p}} \times n_{\text{p}}$ identity matrix, making $\mathbf{R}$ an $n_{\text{d}} n_{\text{p}} \times n_{\text{d}} n_{\text{p}}$ matrix. This can be visualized in one dimension as organizing PEF coefficients along both a lag axis and a position axis and applying a derivative filter along the position axis separately for each lag. In higher dimensions the derivative is replaced by a multi-dimensional Laplacian,

and can be visualized as organizing the filter coefficients onto the additional spatial axes and applying the multi-dimensional Laplacian filter along all of the spatial axes for each lag. In matrix form, the higher-dimensional case would look like the matrix in equation 2.27, but with additional off-diagonal terms corresponding to the adjacent filter coefficients in other spatial axes.

Once the nonstationary PEF has been estimated, it can be applied in the same manner as for the stationary PEF in equation 2.16. Now, however, the convolution with the PEF is nonstationary so the $\mathbf{F}$ matrices are replaced with the nonstationary convolution matrix $\mathbf{F}_{\text{ns}}$, with coefficients $f_{i,j}$ indexed by both space i and filter lag j, written as

$$\mathbf{F}_{\text{ns}} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots \\ f_{1,1} & 1 & 0 & 0 & \cdots \\ f_{2,2} & f_{2,1} & 1 & 0 & \cdots \\ f_{3,3} & f_{3,2} & f_{3,1} & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{2.28}$$

The matrix $\mathbf{F}_{\text{ns}}$ is a compact $n_{\text{d}} \times n_{\text{u}}$ matrix, as it is applied to the n_{u}-element unknown data and outputs a filtered version of that n_{d}-element data. This goes into an equation similar to equation 2.16, but with a nonstationary $\mathbf{F}_{\text{ns}}$ matrix in place of the stationary $\mathbf{F}$ matrix to produce

$$\mathbf{r}_{\text{m}} = \mathbf{F}_{\text{ns}}\mathbf{J}\mathbf{m} + \mathbf{r}_0. \tag{2.29}$$

This can again be expressed in a form similar to equation 2.18, giving

$$\mathbf{J}\mathbf{m} = -(\mathbf{F}_{\text{ns}}^{\dagger}\mathbf{F}_{\text{ns}})^{-1}\mathbf{F}_{\text{ns}}\mathbf{r}_0. \tag{2.30}$$

Now that both stages of the interpolation have been described, this time with a non-stationary prediction-error filter, we apply the method to the quarter-dome synthetic data.

Nonstationary prediction-error filter interpolation: 3D example

Figure 2.16 shows nonstationary PEF estimation, again on the quarter-dome synthetic that proved problematic for both a single PEF and a series of PEFs applied in patches. The 3D nonstationary PEF has $5 \times 5 \times 5$ elements, and varies every 5^{th} data point on the depth axis and every 2^{nd} data point on the other axes, giving a total of almost 3 million unique filter coefficients, slightly less than triple the number of fitting equations for these data.

The pattern in Figure 2.16b shows the regions over which the PEF coefficients are constant, in this case five points along the depth axis and two points along the other two axes. Figure 2.16c is the data residual $\mathbf{r}_{\text{d}}$ of the nonstationary PEF estimation in equation 2.21, that is, the convolution of the nonstationary PEF with the fully-sampled training data. The data are almost perfectly predicted. The conjugate-direction solver used to solve this problem converges quickly, as shown in Figure 2.16d, which plots the norm of the residual of equation 2.21 as a function of iteration number. The solution converges in less than 100 iterations instead of the theoretical guarantee of nearly $20,000$ times that number, the total number of unknowns.

Figure 2.17a shows the same sampled data seen in Figure 2.13a, and Figure 2.17b shows the interpolated result, the solution of equation 2.29. The difference between the interpolated data and the original data is shown in Figure 2.17c. Here the only significant errors are near the highly nonstationary left side of the input data.

The interpolation result is significantly better than either the stationary PEF result in Figure 2.13b or the result obtained with patching in Figure 2.15b. This is explained by both the number of filter coefficients estimated and the lack of lower amplitudes associated with the boundaries between patches. For the stationary case 113 filter coefficients were estimated; the patch-based case used approximately $44,000$ filter coefficients, while the nonstationary case has over 2 million coefficients. Figure 2.18 is an example where a nonstationary PEF was estimated with the same number of filter coefficients as the case in Figure 2.15. We can see that the interpolation result

in Figure 2.18a is much better than the result in Figure 2.15a, and the differences between the interpolated data and the original fully-sampled data are again where the data is the most nonstationary, where using additional filter coefficients improves the result.

The convergence of the PEF interpolation, shown in Figure 2.17d shows that the conjugate-direction algorithm again requires a small number of iterations to converge. The value of using the method of conjugate directions for this problem is clear when we consider that the PEF estimation matrix, if actually constructed, would have approximately 8×10^{12} elements, almost all of which are zero. The matrix in the second step of the interpolation would also be large. Instead in both of these problems only a few copies of the data and the filter are needed, and the results converge after fewer than 100 iterations in both problems.

The examples shown in this chapter have involved real-valued data, but all the theory is applicable as well to complex data and complex-valued prediction-error filters. The only adjustment is to ensure that the adjoint matrices are the complex-conjugate transpose of the original matrices.

The assumption made in most of the interpolation examples in this chapter is a strong and unrealistic one: that the answer is already known; that is, the training data are essentially that data we would have if they contained no gaps. What I have shown is that a nonstationary PEF is able to reconstruct missing data nearly perfectly if adequate training data for the PEF are used. This means that the training data have a local autocorrelation that matches to (within a scale factor) an acceptable extent that of the data that need to be interpolated. The following chapters deal with practical examples where ideal training data are not available, so other training data must be used.

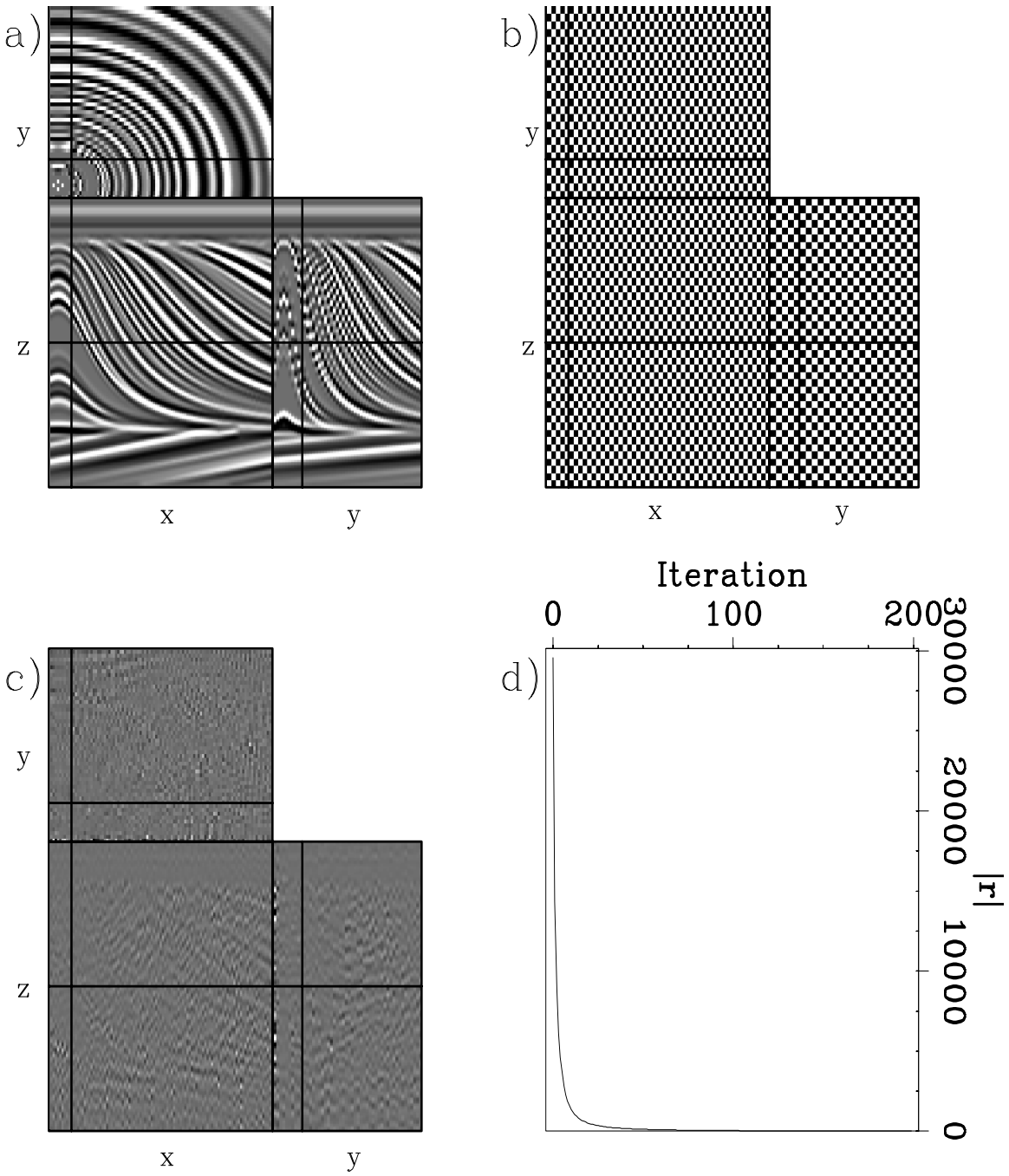

Figure 2.16: Estimation of a nonstationary PEF on the quarter-dome synthetic. (a): original data. (b): diagram of non-stationarity of filter coefficients. (c): data residual of PEF estimation after division by a 1D PEF estimated on (a) To emphasize differences. (d): convergence of PEF estimation. **ER** PEF/. nspefestann

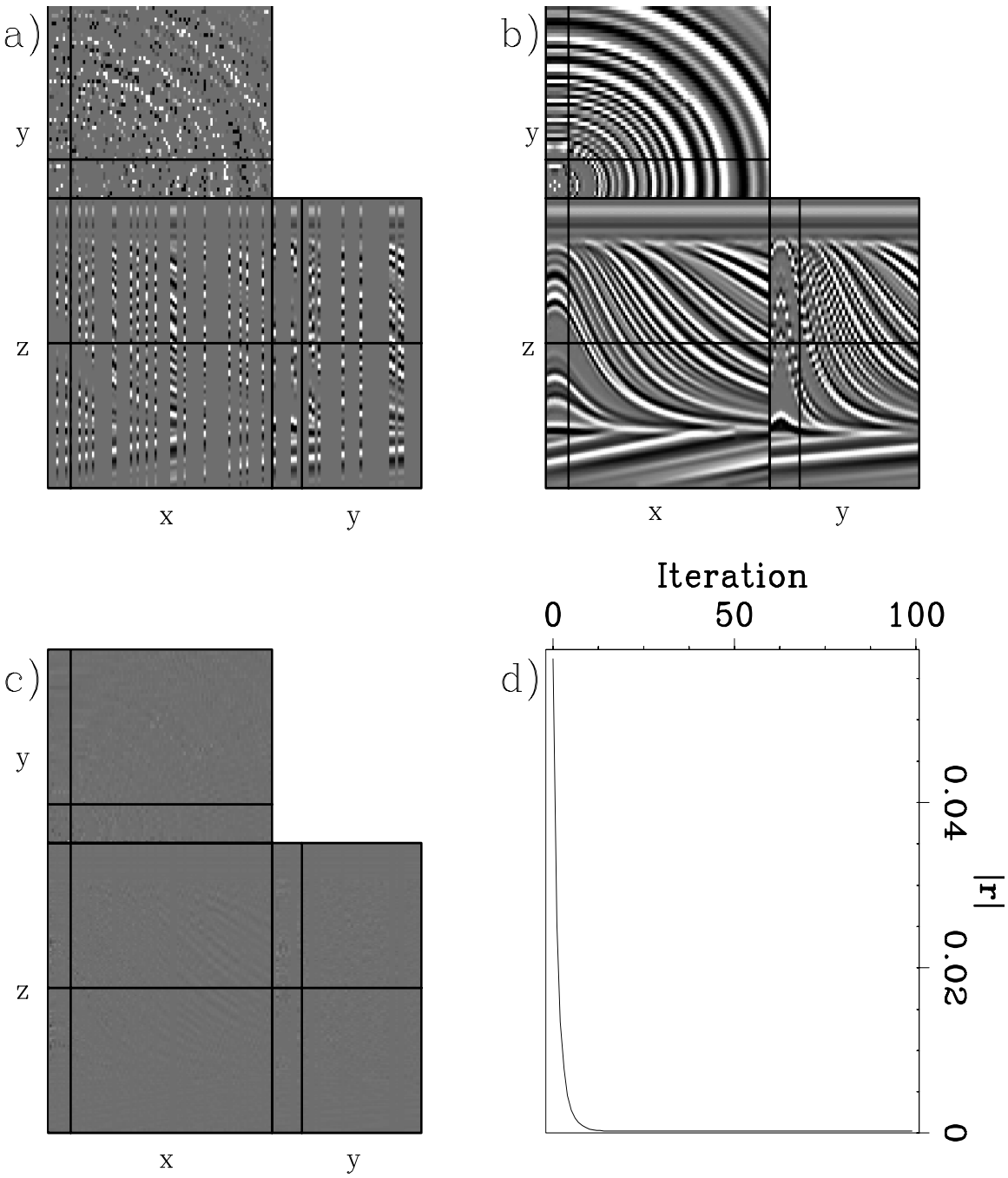

Figure 2.17: Interpolation of the quarter-dome with a nonstationary PEF. (a): Sampled quarter-dome data. (b): Interpolated result. (c): difference between interpolated result and original data. (d): convergence of interpolation of missing data. **ER** PEF/. nspeffillann

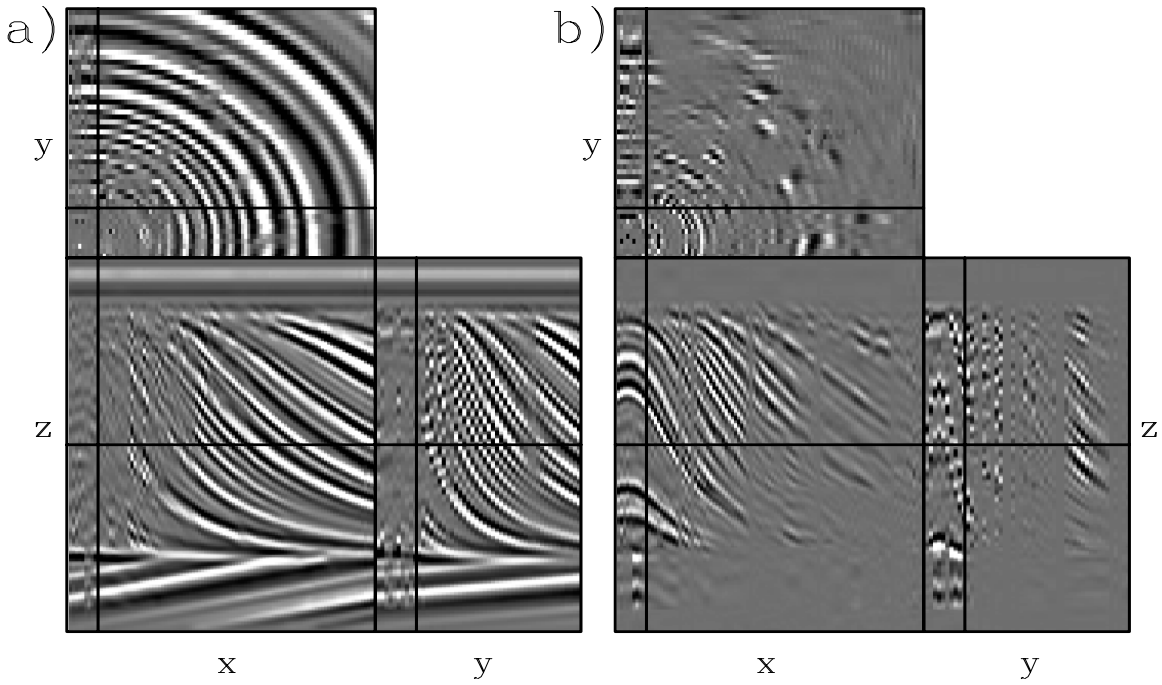

Figure 2.18: Interpolation of the quarter-dome with a nonstationary PEF with the same number of coefficients as the patched case. (a): interpolated data. (b): difference between interpolated result and original data. The interpolation is much better than the patched case but is worse than the smoothly-nonstationary result. **ER** PEF/. patchnsfillann