Chapter 5

Interpolation in the frequency-space domain with nonstationary PEFs

Spitz (1991) demonstrates that data can be interpolated in the f-x domain using a spatial prediction filter on each frequency. Unlike Fourier transform-based methods that typically require unaliased data and dip-based methods that perform relatively poorly when multiple events with highly differing dips are present, this method's ability to interpolate multiple aliased dips has been demonstrated successfully. The original Spitz approach of interpolation has been extended from two to three dimensions (Wang, 2002), but the method continues to be applied in a patch-based approach with events assumed to be locally planar. In this chapter, I use the approximation in the Spitz method to interpolate by integer factors, producing an output with two, three, or four times the sampling density along each interpolated axis. I do this by using nonstationary prediction-error filters on frequency slices in two, three, four, and five dimensions.

Following the approach in Chapter 2, interpolating by an integer factor presents the same problem for PEF estimation as does that of irregular sampling in Chapter

3: the interleaved traces cause all rows of the convolutional matrix used in estimating the PEF to contain unknown data and become unusable. When the data are in time and space, the PEF coefficients can be spread out so that columns of the PEF are spaced by the interpolation factor, and the filter coefficients are also spaced along the time axis by the same factor (Claerbout, 1992). This method of spacing of the filter, which can be thought of as subsampling in both space and time, treats the lower frequencies as higher ones, so care must be taken to ensure that this does not time-alias the data.

Once the t-x PEF has been estimated at this spacing, the spacing operation is reversed and the PEF is used to interpolate the data. Because of this spacing the spectral information captured along the time axis of a t-x PEF at this coarser sampling is different from the finer sampling where the PEF is applied to interpolate the data, so the frequencies appear to be doubled (or tripled in the case of a factor of three interpolation). This potential pitfall of capturing incorrect or temporally aliased information is avoided in an f-x approach, since, in an f-x approach, the PEF operates only over space and not time or frequency. Because of this, no characterization of the frequency content is made in the f-x approach, only that obtained by the spatial autocorrelation of the data at each frequency for each of the independent PEFs.

I apply the assumption behind Spitz interpolation, that the wavenumber spectrum of coarser-sampled lower frequencies is the same as that of the finer-sampled higher frequencies, to estimate a non-stationary prediction-error filter. I use this filter to interpolate regularly-sampled data on a frequency-by-frequency basis. By using a nonstationary PEF, the more rapidly varying slopes are better interpolated than with a spatial-patch-based approach, but patching is still required on the time axis. I use multidimensional filters to interpolate in two, three, four, and five dimensions simultaneously, and test this method on synthetic plane-wave examples, the quarter-dome synthetic, 3D prestack synthetic data, and finally 3D prestack field data. For the prestack examples, a three-dimensional interpolation performs best for inline source interpolation, a 4D interpolation being less desirable because of the small number

of points along the fourth axis. This changes with crossline receiver cable interpolation and field data, where the data are most robustly interpolated with a full 4D approach. The 4D approach produces a larger improvement in the field-data example, which I attribute to the presence of localized noise that dominates lower-dimensional approaches. Finally, I iteratively interpolate the 3D prestack field data to the extent necessary for 3D surface-related multiple prediction, and examine the deterioration of the result as progressively more data need to be interpolated.

PLANE WAVES IN FREQUENCY AND SPACE

Seismic data are typically oversampled along the time axis but undersampled along the other axes. Seismic data are also composed of a superposition of (locally) planar events, such as that in Figure 5.1a. Applying a temporal Fourier transform to this planar event results in a complex exponential in space that varies as a function of frequency, with the real and imaginary parts of this event shown in Figures 5.1b and 5.1c, respectively. Both higher frequencies and steeper slopes result in higher wavenumbers.

This dependence of wavenumber on frequency and slope is predictable for a planar event. Starting with a planar event Fourier transformed and sampled in frequency and x, decimating for each frequency, f, in the x-direction by a factor p will produce a vector with the same wavenumber as that for the fully sampled plane wave at a frequency pf. Figures 5.2a-d are examples of this, Figure 5.2a is a repeat of Figure 5.1b, showing the real part of the lower 128 frequencies of 64 traces of a single plane wave, this time with the x-axes along the ordinate. Figure 5.2b is this same plane wave, but with every second trace removed leaving a total of 32 traces, and the frequency range is half that of Figure 5.2a, so each of the 128 frequencies in Figure 5.2b is half that of the corresponding frequency in Figure 5.2a. This same trend is repeated with $p = 3$ in Figure 5.2c and $p = 4$ in Figure 5.2d. I obtained the finer sampling in frequency by zero-padding the time axis by a factor of two, three or four before the Fourier transformation. The wavenumbers of the corresponding frequency

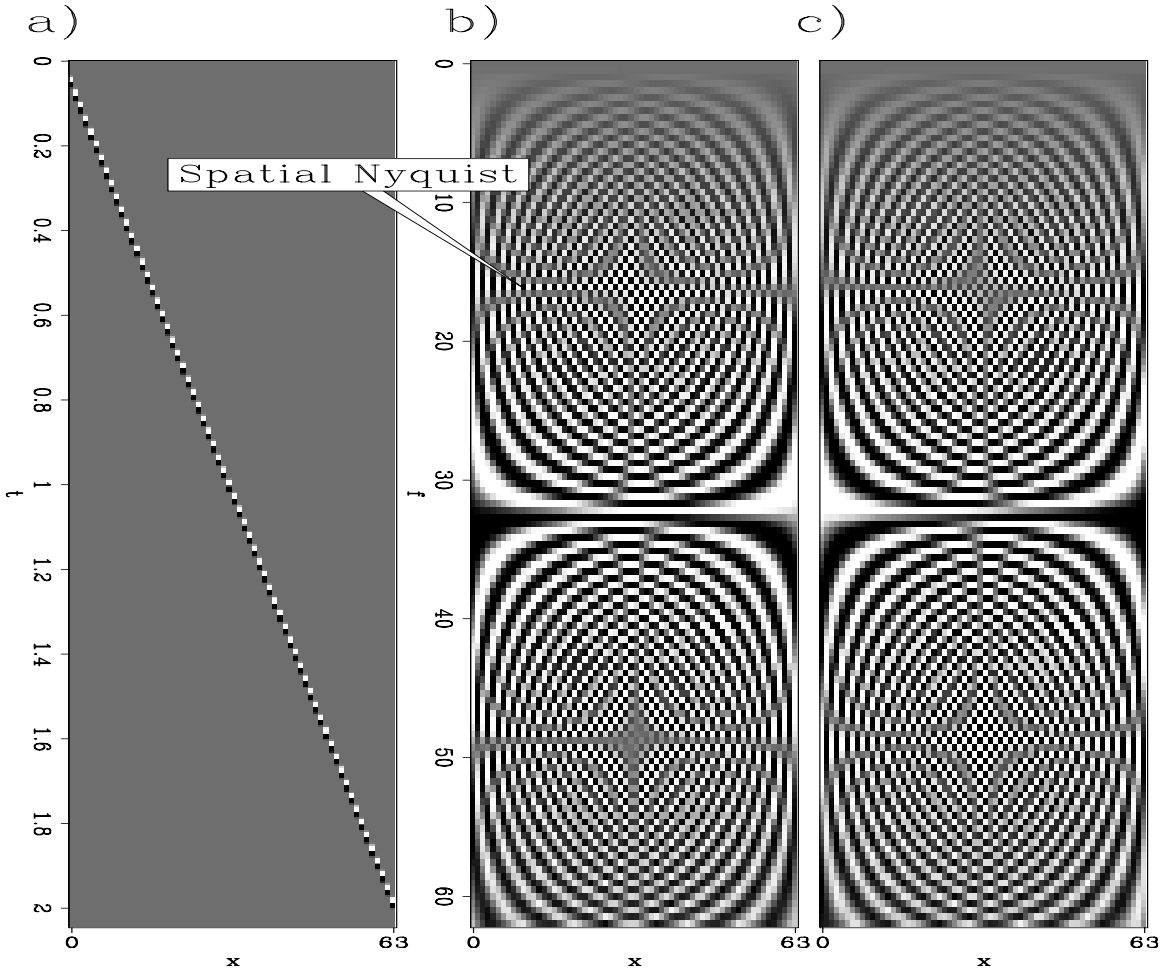

Figure 5.1: A spatially-aliased plane wave in t-x, generated with a Ricker wavelet with a central frequency of 55 Hz and a slope of 650 m/s sampled 64 times (every 20 m) shown in time and space in (a). The real (b) and imaginary (c) values of the positive frequencies show how the spatial wavenumber increases with frequency until the spatial Nyquist is reached at 17.5 Hz, and then becomes aliased. The aliasing is also apparent in t-x. **ER** fxNS/. 1planeann

slices of the different spatial samplings are the same, although both the phase and amplitude of these frequency slices differ from that at a higher frequency.

I estimate a PEF on these lower-frequency data, exploiting the fact that the PEF is insensitive to the phase and amplitude-scale differences associated with the frequency change, but the similar wavenumber spectra. I then use this PEF to interpolate the missing data at a higher spatial sampling rate and frequency.

F-X INTERPOLATION WITH A MULTIDIMENSIONAL PREDICTION-ERROR FILTER

Recall the PEF-estimation equation from Chapter 2,

$$\mathbf{Kf} = -(\mathbf{D}^\dagger\mathbf{D})^{-\mathbf{1}}\mathbf{D}^\dagger\mathbf{d}. \tag{5.1}$$

The unknown filter coefficients $\mathbf{Kf}$ are estimated from fully-sampled training data, $\mathbf{d}$, and a convolutional matrix, $\mathbf{D}$, containing the training data. The interpolation goal is to increase the density of spatial sampling by an integer factor, p. Using the Spitz approximation, the training data (with the original spatial sampling rate) have a frequency that is $1/p$ of the desired output frequency. Since these data are in frequency and not time, $\mathbf{d}, \mathbf{D}$, and $\mathbf{Kf}$ are all complex-valued. The length of the PEF correlates with the number of simultaneous dips that can be interpolated, or, equivalently, the number of complex sinusoids that can be predicted by the PEF coefficients. A two-term 1D PEF can capture one sinusoid, while a three-term PEF can capture two, and so on. Expanding this analogy to higher dimensions is straightforward. In the helical coordinate, equation 5.1 can be used to solve for a one-dimensional filter (for 2D interpolation), a two-dimensional filter (for 3D interpolation) and so on.

Once this PEF has been estimated on the lower-frequency training data, it can then be used to interpolate the missing samples, as in equation 2.18 (repeated here as equation 5.2),

$$\mathbf{Jm} = -(\mathbf{F}_{\mathrm{u}}^\dagger\mathbf{F}_{\mathrm{u}})^{-\mathbf{1}}\mathbf{F}_{\mathrm{u}}^\dagger\mathbf{r}_0, \tag{5.2}$$

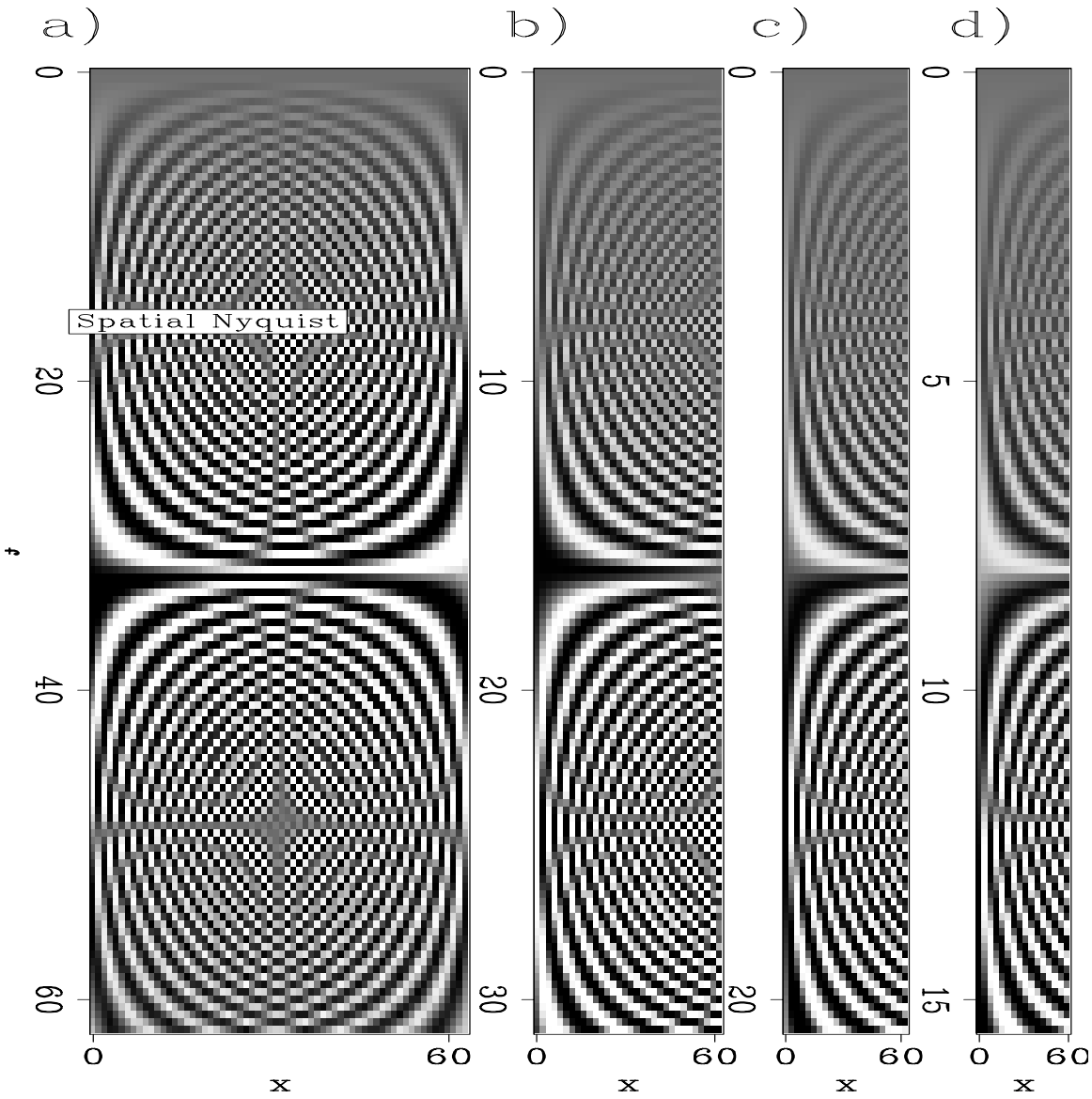

Figure 5.2: A plane wave in f-x with four different sampling rates. a) original f-x data, with Δx and Δf. b) resampled to $\frac{\Delta f}{2}$ (showing the lower half of frequencies) and $2\Delta x$. c) $\frac{\Delta f}{3}$ (showing the lower third of frequencies) and $3\Delta x$. d) $\frac{\Delta f}{4}$ (showing the lower quarter of frequencies) and $4\Delta x$. The spatial wavenumber is the same at each frequency, but the phase is different. **ER** fxNS/. 1planetrickann

where the unknown values $\mathbf{Jm}$ for a single frequency are determined by a convolutional matrix, $\mathbf{F}_{\mathrm{u}}$, containing the PEF estimated from the lower-frequency training data, as well as the known data convolved with the PEF, $\mathbf{r}_0$. Again, the data consist of a single output frequency that is p times more densely sampled in space than is the input, and each frequency is solved as an independent problem.

Multidimensional plane-wave interpolation

Using filters of one dimension less than the data means that in practice a larger amount of data can be simultaneously interpolated compared to that in t-x interpolation methods wherein the dimensionality of the filter should (at least) operate over the dimensions containing holes. For large problems where the data must be broken up into chunks because of memory limitations, this means that fewer chunks are required, and the chunks can either cover larger regions of the same axes interpolated in t-x or additional dimensions can be added to the interpolation, if available. Because 3D prestack reflection seismic data contain five axes: time, two source coordinates, and two receiver coordinates, I construct a five-dimensional $256 \times 30 \times 30 \times 30 \times 30$ hypercube containing three plane waves, shown in slices of all combinations of axes in Figure 5.3. Fourier transformation of the data yields 128 frequency slices (excluding the symmetric negative frequencies), each of dimension $30 \times 30 \times 30 \times 30$.

In order to interpolate these data by a factor of two along all four spatial axes, I generate 128 frequency slices, each with half the frequency of the corresponding slice of the original data, extending the data by a factor of two by zero-padding prior to applying the temporal Fourier transform. I then estimate 128 unique four-dimensional $3 \times 3 \times 3 \times 3$ prediction-error filters required to interpolate the 128 input frequency slices on a $60 \times 60 \times 60 \times 60$ grid. This produces a total of roughly 26 GB of data for even this small five-dimensional example, shown in Figure 5.4. This example would be impossible to solve at once in t-x, as it requires 128 times more memory than that in f-x. The interpolation correctly interpolates these data along all four spatial axes, several of which were severely aliased. Because of the high dimensionality of

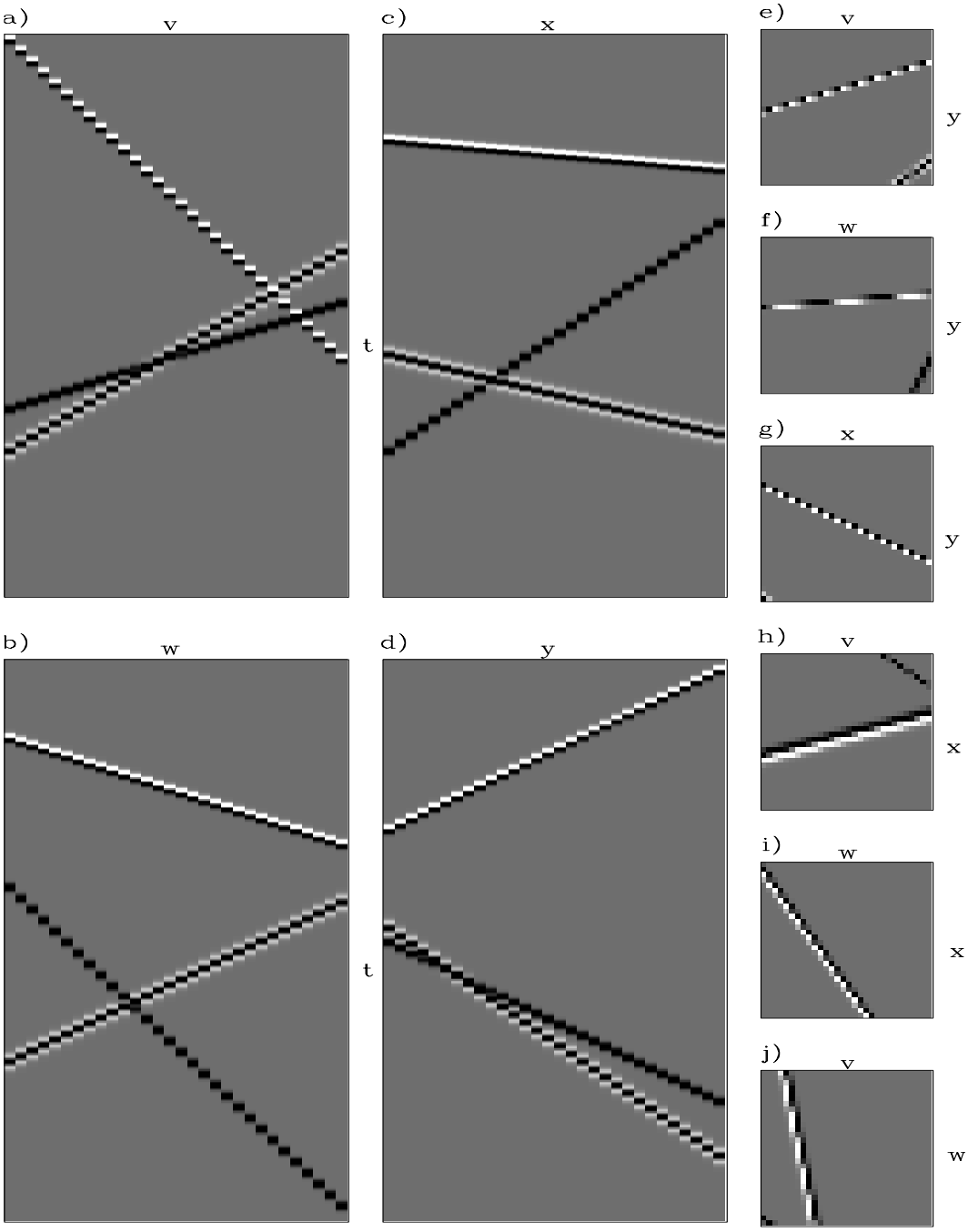

Figure 5.3: Three plane waves in five arbitrary dimensions, four spatial and time (v,w,x,y,t). The ten slices correspond to combinations of all of the axes: a) v,t; b) w,t; c) x,t; d) y,t; e) v,y; f) w,y; g) x,y; h) v,x; i) w,x; j) v,w. The plane waves are severely aliased along many of the axes. **ER** fxNS/. planesinann

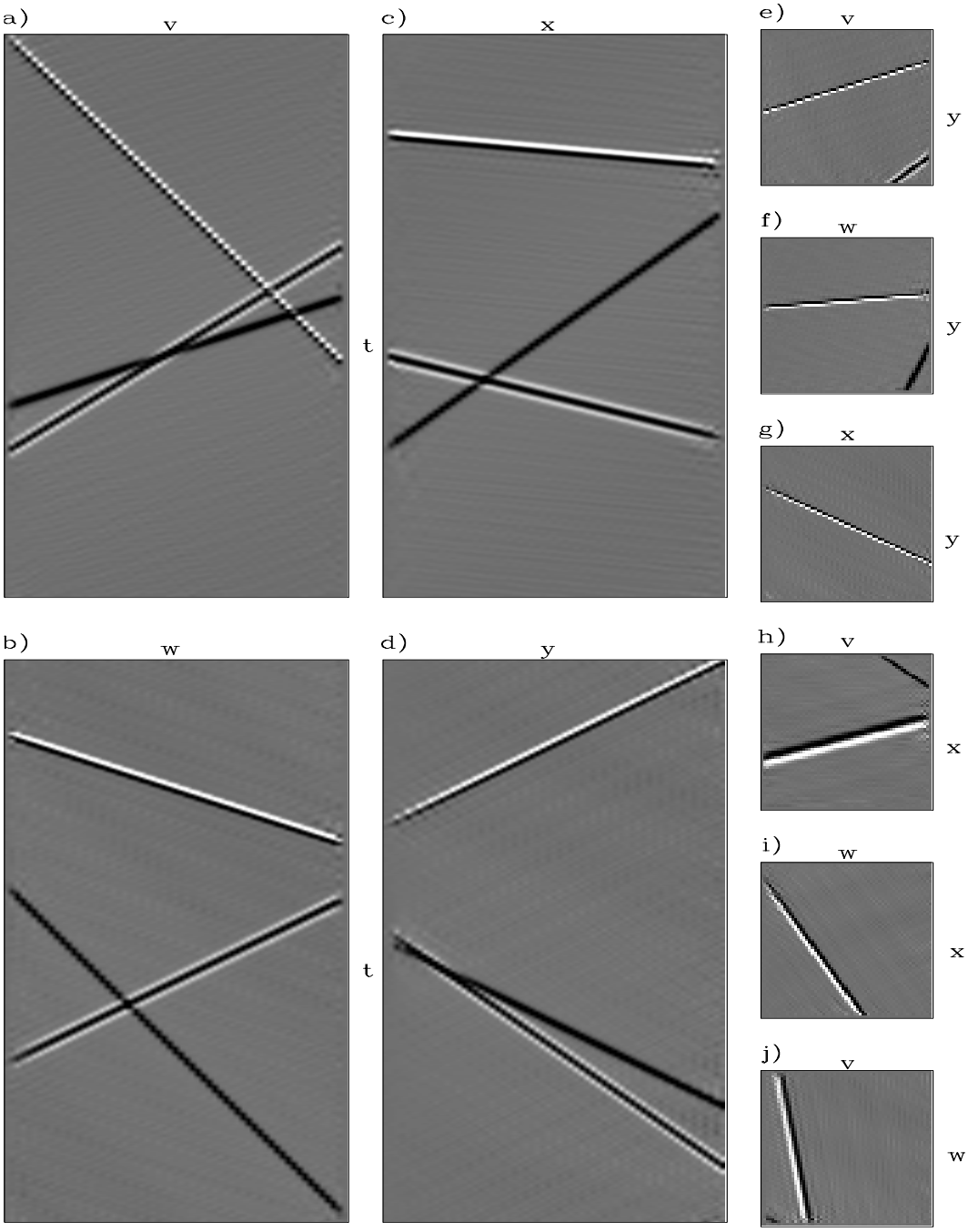

Figure 5.4: Interpolation of the data in Figure 5.3 using 128 four-dimensional PEFs. The aliased data are properly interpolated along all axes, and the amount of data has been increased by a factor of 16. **ER** fxNS/. planesinterpann

the problem, the interpolation by a factor of 16 was successful, whereas a factor of 16 interpolation along a single axis would be compromised because only the lowest 1/16 frequencies would be used.

The situation for this interpolation was ideal in that the events being interpolated were planar, although interesting in places, and were predictable in f-$\mathbf{x}$ with a single prediction-error filter at each frequency. For field data, which contain curved events, instead of using this stationary approach in overlapping patches in all dimensions, I opt for a single spatially-variable PEF for each frequency. Because the filters cannot vary in time, I apply this approach in time patches that have been Fourier transformed.

NONSTATIONARY F-X INTERPOLATION

Unlike the previous example where the data are composed of planar events, seismic data are composed mainly of curved ones. Hyperbolic features, however, can be considered as locally planar events that have a slope varying as a function of position and time. Figure 5.5a shows a simple hyperbola in space and time. Figure 5.5b is the real portion of the same data Fourier transformed from time to frequency. Each frequency slice is like a chirp function with a wavenumber that decreases as the apex of the hyperbola is approached and slope decreases. It then increases in both wavenumber and slope as the position moves toward each asymptote of the hyperbola. Figure 5.5c is this same hyperbola with half the spatial sampling and double the frequency sampling. Although not obvious to the eye, the wavenumbers present in the subsampled data are the same as those in the original data; thus the same low-frequency assumption that works for plane waves also works for waves with spatially varying slope.

Returning to equation 2.22,

$$\mathbf{K}_{\text{ns}}\mathbf{f}_{\text{ns}} = -(\mathbf{D}_{\text{ns}}^{\dagger}\mathbf{D}_{\text{ns}} + \epsilon^2\mathbf{R}^{\dagger}\mathbf{R})^{-1}\mathbf{D}_{\text{ns}}^{\dagger}\mathbf{d}, \tag{5.3}$$

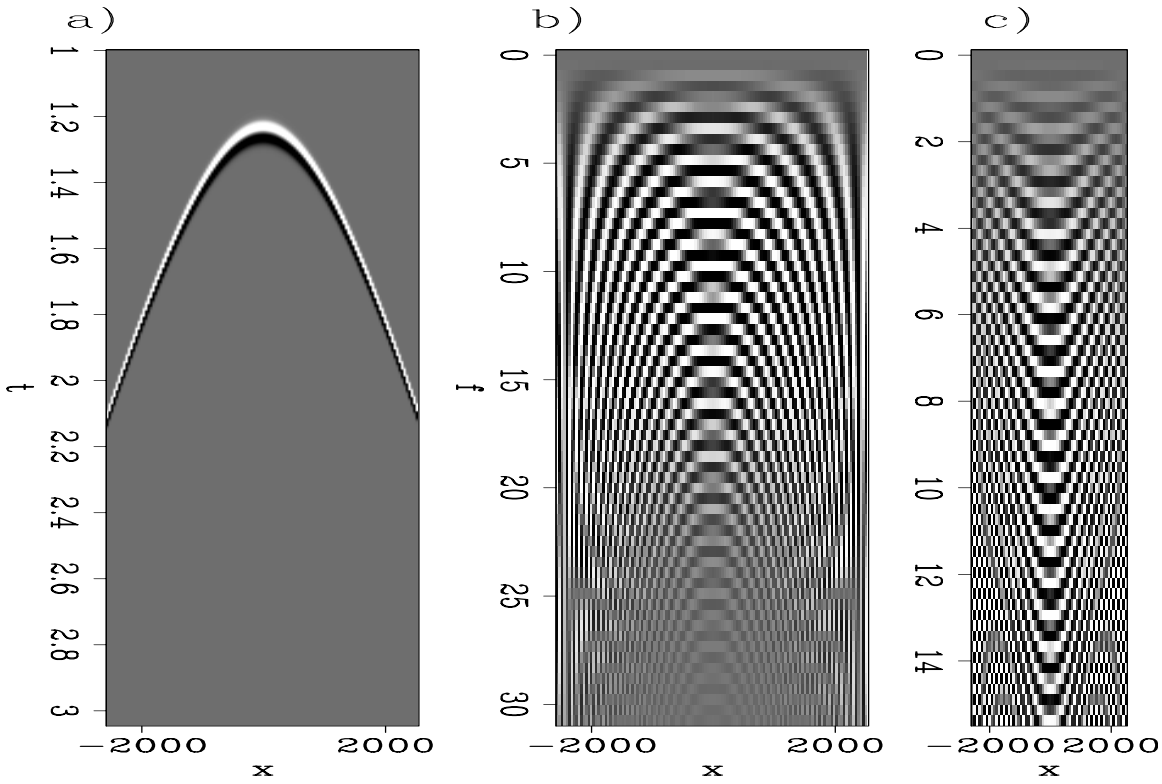

Figure 5.5: A single hyperbola in t-x and f-x: a) a hyperbola in t-x; b) the real part of the same hyperbola in f-x; c) the real part of the same data with half the spatial samples removed and the lower half of frequencies shown. For each frequency, the local wavenumber content is the same in both b and c. **ER** fxNS/. hyperbola

recall that unknown nonstationary PEF coefficients, $\mathbf{K}_{\text{ns}}\mathbf{f}_{\text{ns}}$, can be estimated from training data, $\mathbf{d}$, a nonstationary data-convolution matrix, $\mathbf{D}_{\text{ns}}$, that is a function of the training data, and a regularization operator $\mathbf{R}$ with a scaling factor ϵ.

Since the goal of this interpolation is to increase spatial sampling by a factor p for any given frequency, the training data are again the input data that we wish to interpolate, but at a frequency p times lower than the frequency we are interpolating. These complex-valued training data are used to estimate a set of complex-valued nonstationary PEF coefficients using equation 5.3.

Once I estimate this nonstationary PEF, I then use it to solve equation 2.30, repeated here as equation 5.4,

$$\mathbf{Jm} = -(\mathbf{F}_{\text{ns}}^{\dagger}\mathbf{F}_{\text{ns}})^{-1}\mathbf{F}_{\text{ns}}\mathbf{r}_0, \tag{5.4}$$

interpolating the unknown data, which are the $p-1$ samples between each pair of known values on each axis of the frequency slice. Here, the unknown values of the frequency slice, $\mathbf{Jm}$, are estimated from the known values convolved with the PEF, $\mathbf{r}_0$, and a convolutional matrix $\mathbf{F}_{\text{ns}}$, created with the nonstationary PEF estimated in equation 5.4. Note that the size of the interpolated data is greater by a factor of p along each interpolated axis than that of the data used to estimate the nonstationary PEF. The $\mathbf{F}_{\text{ns}}$ matrix must therefore be constructed so that one local set of filter coefficients is repeated p times along each axis within this matrix when compared to a matrix for data that are the size of the training data.

QUARTER-DOME SYNTHETIC EXAMPLE

The quarter-dome synthetic has been used in both Chapters 2 and 3; it has both events with stationary slopes and those with a range of variable slopes. Let us now sample the data with half the original sampling along each of the two spatial axes and attempt to interpolate to the original sampling. We first pad the input $256 \times 50 \times 25$ data by a factor of two in time and apply a Fourier transform to create frequency

slices, each with one-half the frequency of the original data. Next, we estimate a 3×3 nonstationary PEF that varies every fifth point in both spatial dimensions for each of the 128 frequency slices of the training data by using 100 iterations of a conjugate-direction solver on equation 5.3. Once these nonstationary PEFs have been estimated, We can then use them in equation 5.4 to interpolate the data onto the larger 100×50 slices.

Figure 5.6a shows the original data, and the sampled data (with zeroes in the place of unsampled cells) are shown in Figure 5.6b. As a comparison, I use stationary PEFs to solve the same problem, with a 3×3 stationary PEF used for each frequency. This approach produces the result in Figure 5.6c. Although the upper and lower parts of the model, which are stationary, are correctly interpolated, the steep slopes that vary out-of-plane in the center of the model are aliased when using stationary PEFs. This stationary approach can also be applied in patches, wherein the model is broken up into many overlapping patches and each problem is solved independently. Figure 5.6d is a result of this approach, where the $200 \times 50 \times 25$ input data were broken up into 2160 overlapping patches of $32 \times 15 \times 15$. This result is somewhat better than that in Figure 5.6c, with most of the data properly interpolated except for the most variable regions to the near-left of the cube, also seen in the lower-left of the depth slice.

Applying the nonstationary f-x approach described in equations 5.3 and 5.4 to the data in Figure 5.6b results in Figure 5.6e. While the highly variable regions are reasonably well interpolated, obvious errors appear in the upper and lower stationary regions of the data, seen in the crossline slice on the right panel. Large amplitude errors exist in the flat regions correctly interpolated by all other methods. This method creates spatially variable PEFs, but assumes stationarity in time, that is, that a PEF for each frequency is appropriate at all times. This is incorrect in this example, as the slope varies as a function of position **and** time.

Patching was used along all axes to produce Figure 3.7d, which adds considerable expense. When nonstationary PEFs are used, I use this approach but only along the time axis, considerably reducing the added cost. I break up the time axis of Figure

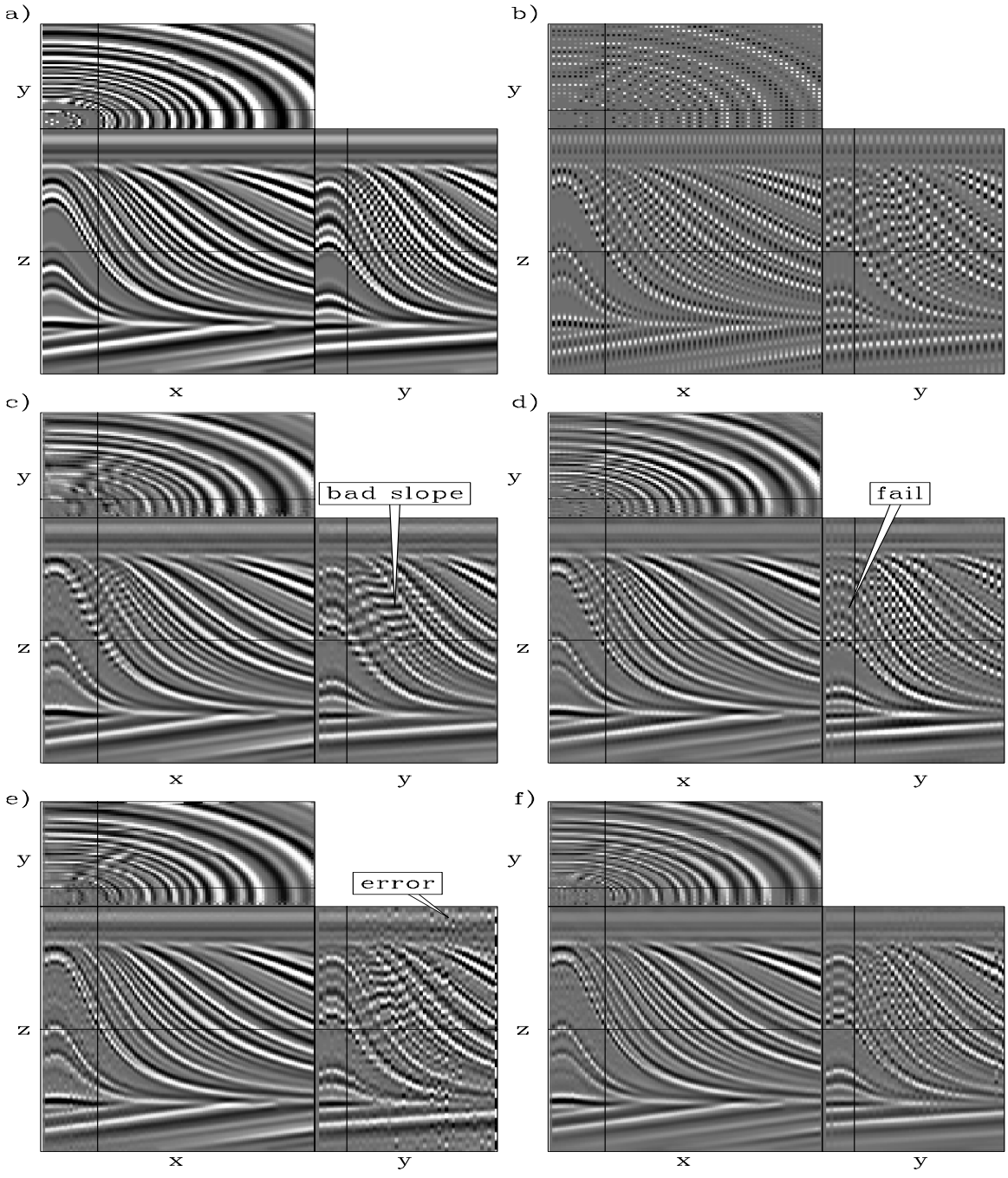

Figure 5.6: Interpolation of the quarter-dome synthetic. a) original data. b) data sub-sampled by a factor of two on each spatial axis. c) Stationary f-x interpolation. d) f-x interpolation in patches. e) Nonstationary f-x interpolation. f) Nonstationary f-x interpolation in time patches. The nonstationary f-x interpolation in time patches performs slightly better than the patched approach, most notably in the crossline, and is more than an order of magnitude faster. **ER** fxNS/. qdomefxann

5.6b into 30 overlapping windows of 32 points each and then use the same nonstationary filters used to generate Figure 5.6e. This time, however, I do so independently on each frequency for each time window, producing the result in Figure 5.6f. This result couples the patch-based approach along the Fourier-transformed axis with spatially variable PEFs. The interpolation in Figure 5.6f is arguably better than the result of patching along all axes in Figure 5.6d, most notably in the highly-variable region, and is also more than ten times faster to compute. This is because the lack of overlap in space between patches causes the amount of data shuffling to be reduced as well as the overall amount of data interpolated. This bodes well for higher dimensions, as higher-dimensional PEFs can be used and the massive amount of data duplication required for the overlapping spatial patches for each additional dimension is eliminated in the interpolation.

This quarter-dome example, repeatedly used in this thesis, is well served with spatially variable PEFs used in time patches. The computation is relatively fast, and the result compares well with that of a more expensive traditional approach of patching along all axes.

SYNTHETIC 3D MARINE DATA EXAMPLES

I now test this method on a more difficult synthetic dataset containing many diffractions, courtesy of ExxonMobil. A schematic of the model used to create these data is shown in Figure 5.7a. A prism containing hundreds of point diffractors is placed below a horizontal water-bottom, with reflectors beneath the water-bottom placed at a reflection time slightly after the arrival of the diffracted multiples. The data were modeled analytically (Baumstein and Farrington, 2006), with the many interfering point diffractors creating a cloud of coherent noise over the reflector of interest. The shifted apexes of the multiples as well as the out-of-plane multiples are problems not properly addressed with the conventional high-resolution radon or two-dimensional surface-related multiple-elimination methods.

The geometry of this synthetic dataset is based upon a modern conventional 3D

marine acquisition (Figure 5.7b). The minimum and maximum inline offsets are 100 m and 5225 m, respectively, with 410 receivers along a cable at 12.5 m intervals. The 550 m crossline aperture consists of 12 cables separated by 50 m. The 320 inline sources are acquired in a flip-flop fashion, with the inline spacing between two flip sources or two flop sources at 37.5 m, while the crossline separation between the flip and flop sources is 25 m, and the crossline separation between adjacent sail lines is 200 m. A summary of the sampling along with the desired output sampling for 3D SRME is shown in Table 5.1.

Table 5.1: Sampling of the synthetic marine data.

Axis	Δ_{recorded}	Δ_{desired}
h_x	12.5 m	12.5 m
h_y	50 m	12.5 m
s_x	37.5 m	12.5 m
s_y	200 m	12.5 m

As seen in Table 5.1, the inline sources need to be interpolated by a factor of three and the crossline receivers by a factor of four for ideal sampling. I first attempt to interpolate the data in the inline source direction by a factor of three by interpolating simultaneously in two, three, and four dimensions. I compare these results in different sections along all of the axes of this four-dimensional result; a 3D interpolation that does not include the crossline offset axes produces the best result. I then interpolate receiver cables in the crossline direction, varying the dimensionality of the filter, and again find that a 3D approach outperforms a 4D approach.

Synthetic data: Inline source interpolation

The out-of-plane diffractors in Figure 5.7a create both a cloud of diffractions beneath the water-bottom reflection and a set of diffracted multiples that are even more complicated, shown in the constant-offset section on the front face in Figure 5.8a. These

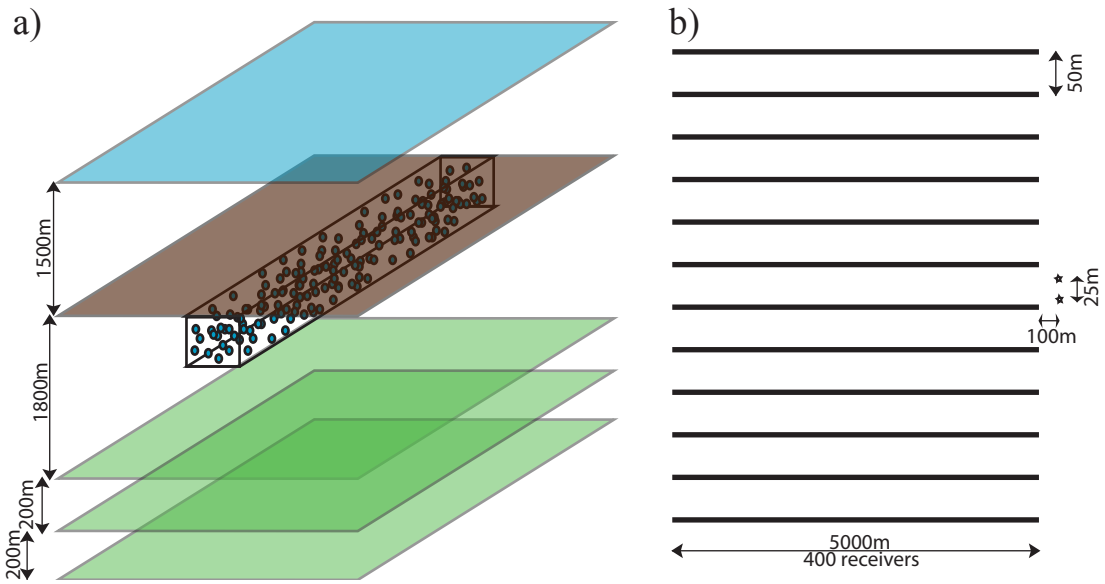

Figure 5.7: Schematic of the synthetic dataset: a) shows the model, a prism-shaped region below the water bottom filled with point diffractors, below which lie three reflectors whose primary reflection times are nearly the same times as those of water-bottom multiples; b) is a plan view showing the acquisition schematic, with twelve receiver cables and flip-flop sources. **NR** fxNS/. exxon

diffractions and diffracted multiples also appear in the shot gather in Figure 5.8b. The crossline variability of this noise can be seen in the time slices in both Figures 5.8a and 5.8b, as well as in the crossline offset sections on both figures.

For interpolation along the inline source axis, I test four different approaches using nonstationary PEFs in frequency and space. All of these approaches use equation 5.3 to estimate the nonstationary PEF and equation 5.2 to interpolate the data with the PEF; the only differences in the equations are the domain in which both the PEF and interpolation take place and the dimensionality of the Laplacian filter used in the regularization matrix, $\mathbf{R}$. I use a conjugate-direction solver in all cases, with 30 iterations for each nonstationary PEF estimation and 60 iterations for each interpolation. I perform these tests on a single sail line from these data, extracting only the flip sources, for a total of 160 sources and 410 receivers on each of the 12 receiver cables. As pre-processing for these tests, I first perform a NMO correction with water

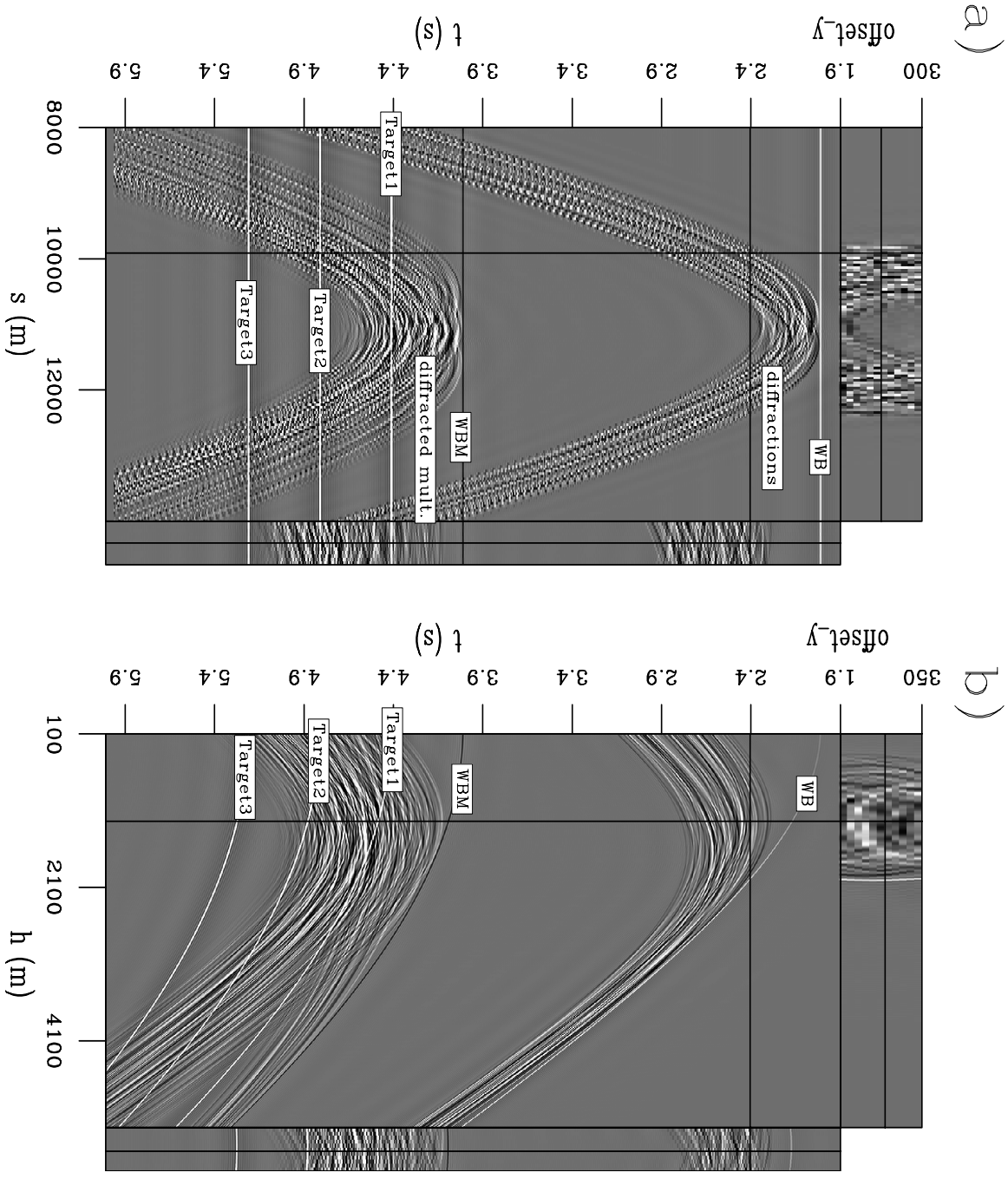

Figure 5.8: Synthetic dataset. a) a cube with a constant-offset section on the front face, crossline offsets on the side face, and a time slice through this cube on the top; b) another cube showing a single shot with inline offsets on the front face, crossline offsets on the side face, and a time slice on top. Events are aliased along both the inline source and crossline offset axes. **ER** fxNS/. exxoninann

velocity to partially flatten the data along offset as an attempt to reduce the amount of energy that crosses over the patches on the time axis that I next create, breaking up the 1024-element time axis into 40 time patches, each 64-elements long. I then apply a Fourier transformation along the now 64-element patched-time axis to produce 32 frequency slices (ignoring symmetric negative frequencies), each of which has 40 time patches, producing a cube that has $160 \times 410 \times 12 \times 40 \times 32$ complex elements, with a desired output three times the size along the first (inline source) axis.

The simplest nonstationary PEF-based interpolation would be to use a 1D PEF that operates over only the source axis. This results in $(410 \times 12 \times 40 \times 32)$ interpolation problems in which a one-dimensional PEF is first estimated on lower-frequency data and is then used to interpolate the sampled data to a higher spatial sampling rate. I estimate a six-term PEF that varies every fifth point along the source axis, for a total of 160 unknown filter coefficients for each problem. In total I estimate the same number of unknown filter coefficients as points in the input data: roughly one billion complex elements.

I next use a more complicated approach by increasing the interpolation to three dimensions. One way to do this is by estimating a PEF over both the interpolated inline source axis and the inline offset axis. Now each individual problem is a two-dimensional problem in which 2D PEFs are estimated on lower-frequency, coarser-sampled training data in inline source and inline offset. These PEFs are then applied to a higher-frequency output with denser sampling along both axes. I subsample the training data by a factor of three along the inline offset axis prior to estimating the PEF, and then I estimate a 6×6 PEF that varies every fifth sample along each axis of the training data, meaning every 15 samples in the interpolated result. These $23,332$ unknowns are estimated for each of the $12 \times 40 \times 32$ problems for a total of roughly 360 million complex PEF coefficients.

Another domain for 3D interpolation of inline sources is in inline source and crossline offset. Here I estimate a 6×3 PEF that varies every fifth point along the inline source direction and does not vary along the crossline offset direction. I solve for the 480 complex PEF coefficients for each of the $410 \times 40 \times 32$ problems for roughly

250 million unknown filter coefficients. Since there are few crossline offsets, I do not subsample the training data along this axis; instead I interpolate both inline sources and crossline offsets, followed by subsampling along the crossline offset axis to return to the original crossline offset sampling. Otherwise, the subsampling would leave only four samples along the crossline offset axis.

Finally, using all of the data in the sail line in a 4D interpolation, I estimate a $6 \times 6 \times 3$ PEF, over inline source, inline offset, and crossline offset, that varies every fifth point along both the inline source and inline offset axes, and every third point along the crossline offset axis. This is repeated for each of the 32 frequencies of the 40 time windows for a total of roughly 430 million complex filter coefficients. I again subsample the training data along the densely-sampled inline offset axis and subsample the output data along the crossline offset axis to remove the currently undesired interpolated crossline offsets.

In all of these cases, once the data have been interpolated frequency by frequency and patch by patch, they are transformed back from frequency to time. I then reassemble the time patches with a triangular weighting in overlapping areas to ensure a smooth transition from one patch to the next. Because the PEFs do not operate over the time axis that I reassemble from patches, a simple triangular weighting suffices because no edge effects from the filter need to be accounted for. I now compare these four different approaches by interpolating the entirety of the sail line and comparing the results. For the four axes, six potential slices can be shown: time and inline offset, time and crossline offset, time and inline source, inline offset and crossline offset, crossline offset and inline source, and inline source and inline offset. Because of the relatively small number of samples in the crossline direction, I now show only results in the inline direction, although crossline information has been used in two of the cases.

Figure 5.9 shows an example of a constant inline offset section from the sixth cable of the data using the four different approaches. This image contains both the known data at every third sample along the inline source axis and the two interpolated traces between each known trace. I have zoomed in on a section of the diffracted multiples

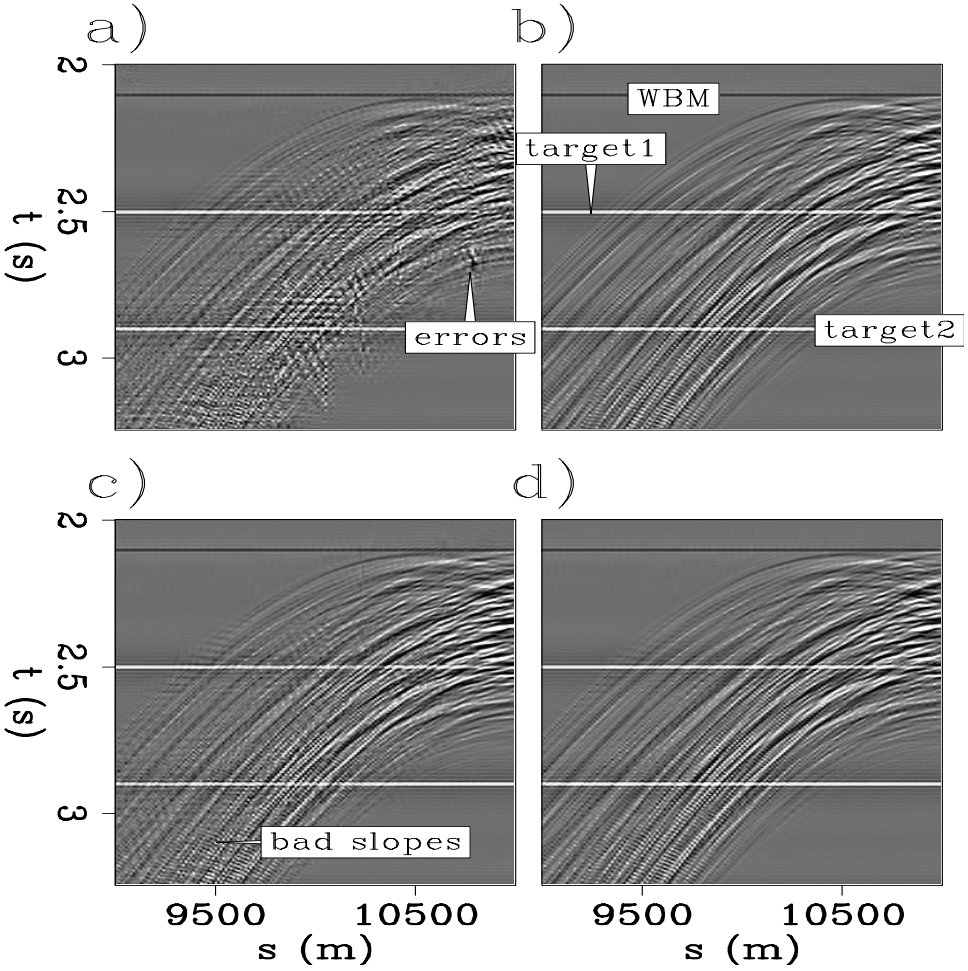

Figure 5.9: A zoomed-in comparison of inline source interpolation along constant-offset sections at 1137.5 m using different PEF dimensions. a) 2D interpolation along constant inline offset sections. b) 3D interpolation along inline source, inline offset cubes. c) 3D interpolation along inline source, crossline offset cubes. d) 4D interpolation along inline source, inline offset and crossline offset hypercubes. Based on the resulting continuity between sources the inline 3D and 4D interpolations perform better than the 2D or 3D crossline interpolations. **ER** fxNS/. exxoncoffinterpcoffcompann

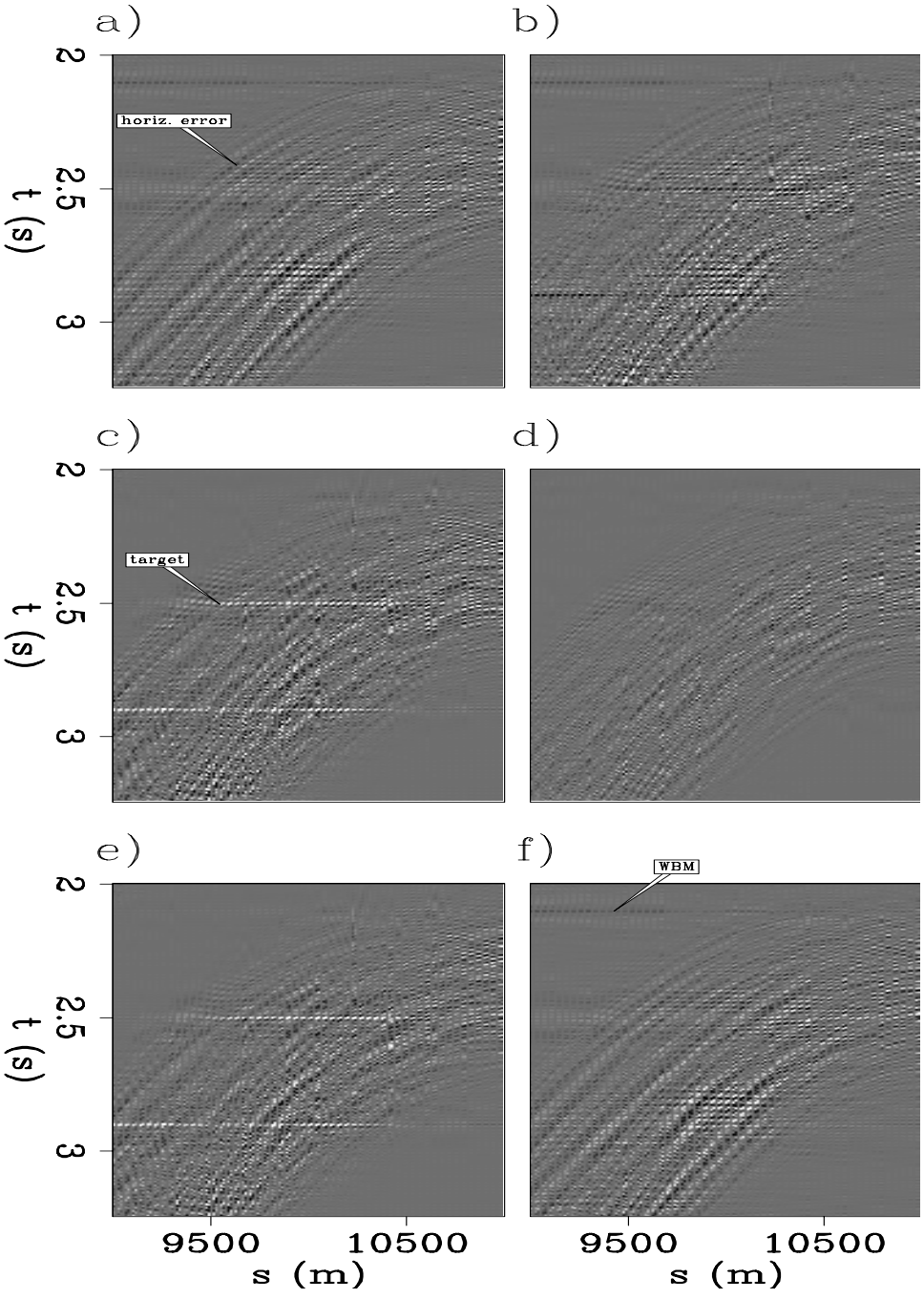

Figure 5.10: Differences between the plots in Figure 5.9. a) 2D vs. 3D inline interpolation; b) 2D vs. 3D crossline interpolation; c) 3D inline vs. 3D crossline interpolation; d) 3D inline vs. 4D interpolation; e) 3D crossline vs. 4D interpolation; f) 2D vs. 4D interpolation. The 2D results have horizontal errors, the 3D crossline result underpredicts the target reflectors, and the 3D inline and 4D interpolations both perform similarly well and show few differences. **ER** fxNS/. exxonsourcecoffdiffann

that shows the largest differences between the four approaches. Figure 5.9a shows the result of a 2D interpolation. While the horizontal reflectors and horizontal water-bottom multiple are easily interpolated, the previously-aliased diffracted multiples are only partially interpolated: the outer diffractions have been interpolated with some success, but the inner, more dense part of the cloud of diffractions contain obvious errors. Horizontal energy appears before the second flat reflection at 2.8 s, and several traces contain errors with energy present after the cloud of diffractions. Including the inline source axis in the PEF estimation produces the result in Figure 5.9b. This result is a large improvement over the 2D result, as the diffractors appear to be continuous, with no errors detectable. Using the crossline offset axis instead of the inline offset axis in the interpolation produces Figure 5.9c. The addition of the 12-point crossline offset axis improves the result from the 2D example in Figure 5.9a, as the errors on small groups of traces are not present.

I reexamine these results by taking differences between the results in Figure 5.9 and amplifying the results, shown in Figure 5.10. The differences featuring the 2D interpolation show both incorrect (flat) slopes surrounding the target reflectors, as in Figures 5.10a, 5.10b, and 5.10f. Meanwhile, the 3D crossline interpolation incorrectly interpolates the target reflectors, as in Figures 5.10b, 5.10c, and 5.10d. The 3D inline and 4D interpolations have few differences, as shown in Figure 5.10d.

However, the 3D crossline offset, inline source interpolation is clearly worse than the 3D inline interpolation, as the original data are visible in the cloud of the diffractions, particularly at the rightmost half of the image. Finally, the 4D interpolation appears to be comparable to the 3D inline result. The locations of the recorded data are not obvious, and no visible errors exist. While it is difficult to discern the 4D from the 3D inline interpolation, the 2D result is clearly the worst, and the 3D interpolation is superior when including the inline offset axis instead of the crossline offset axis in the interpolation. Next, we examine these same results along the inline offset axis.

Figure 5.11 shows these same interpolations, but this time when viewed along the inline offset axis for a single interpolated shot, so in this figure all of the visible data

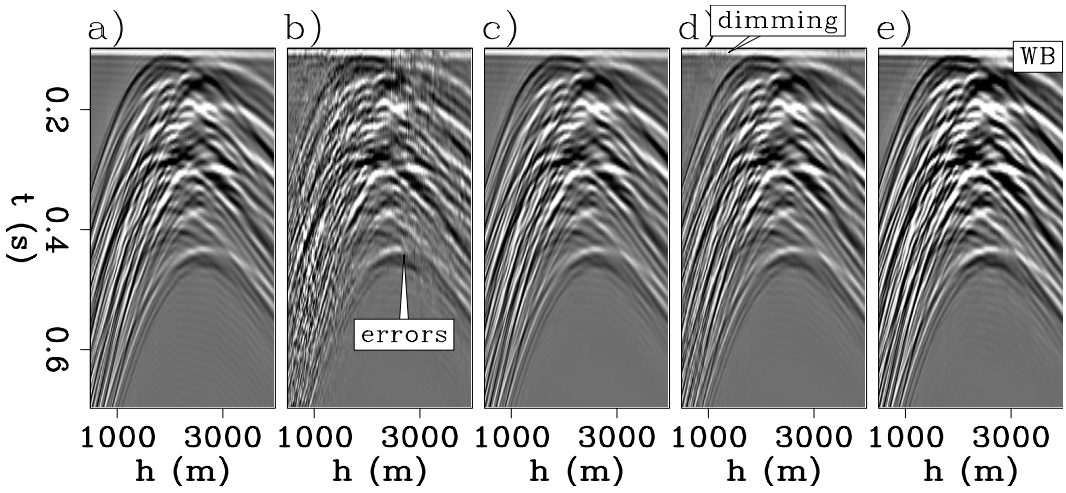

Figure 5.11: One cable from an interpolated inline source at 11706.5 m. a) original recorded shot located adjacent to the interpolated shots in b-e. b) 2D interpolation along constant inline offset sections. c) 3D interpolation along inline source, inline offset cubes. d) 3D interpolation along inline source, crossline offset cubes. e) 4D interpolation along inline source, inline offset and crossline offset hypercubes. The 3D inline and 4D interpolations again produce the best results. **ER** fxNS/. exxoncoffinterpshotcompann

have been created by interpolation. For reference, the recorded shot nearest to the interpolated sources in Figures 5.11b-e is shown in Figure 5.11a. Again, the figure shows a close-up of an interesting region of the results below the water-bottom reflection near the apexes of the diffractions. The 2D interpolation in Figure 5.11b, again shows considerable variability from trace to trace. I attribute this first to each trace having been created from an independent problem, and second that the parameters used in the interpolation (size and variability of the PEFs, amount of regularization of the PEF, and number of iterations) were optimized for a single output constant-offset section and were then applied to the entire dataset. The 3D interpolations again show much more continuity than does the 2D interpolation, with the 3D inline source, inline offset interpolation in Figure 5.11c giving a more continuous result than that of the 3D inline source, crossline offset interpolation. This is most obvious at early times around the water-bottom reflection. The difference between the two 3D results is that each trace in Figure 5.11d is solved as an independent problem, while

all inline offsets in Figure 5.11c are obtained from the solution to a single problem. The differences between the full 4D result in Figure 5.11e, and the 3D inline source, inline-offset result are minimal.

Finally, I examine a time slice from slightly before the first subsurface reflector in an inline offset, inline source position cube, for all of the results in Figure 5.12. In all of these images, the ordinate axis has been interpolated by a factor of three. The 2D interpolation in Figure 5.12a shows a problem that appears as a sinusoidal event along the inline offset axis at roughly 10000 m on the inline source axis, where the flanks of some of the diffracted multiples intersect the time slice. This is the result of variations in the amplitude of the interpolated sources. Other problems also exist with this result, such as the poorly interpolated flanks of other diffracted multiples at the top of the image, where incorrect data are created outside of the flanks of the multiples. The flanks of the diffractors at the bottom-left of the image also appear speckled where the original data are obvious as the higher-amplitude speckles, where this sinusoidal pattern is also present.

The results in Figures 5.12b and 5.12c show that the increase from 2D to 3D interpolation improves the result considerably, again especially in the inline offset, inline source case. The crossline offset, inline source interpolation in Figure 5.12c improves upon the 2D interpolation below the cloud of diffractors, as the speckling is no longer present, but some of the problems at the top and the middle of the image still remain. The 3D inline offset, inline source interpolation in Figure 5.12b gives an excellent result, with the visible problems in the other results gone. The sinusoidal errors in the diffraction flanks are gone, and the recorded data cannot be discerned from the interpolated data. Moving to a full four-dimensional interpolation, the result shown in Figure 5.11e was difficult to differentiate from that of the 3D inline source, inline offset interpolation. In this view, the differences are more pronounced, with the 4D interpolation containing some of the sinusoidal noise present in the 2D and 3D crossline offset results at the center of the slice.

Overall, inline source interpolation of these data appears to be successful. While

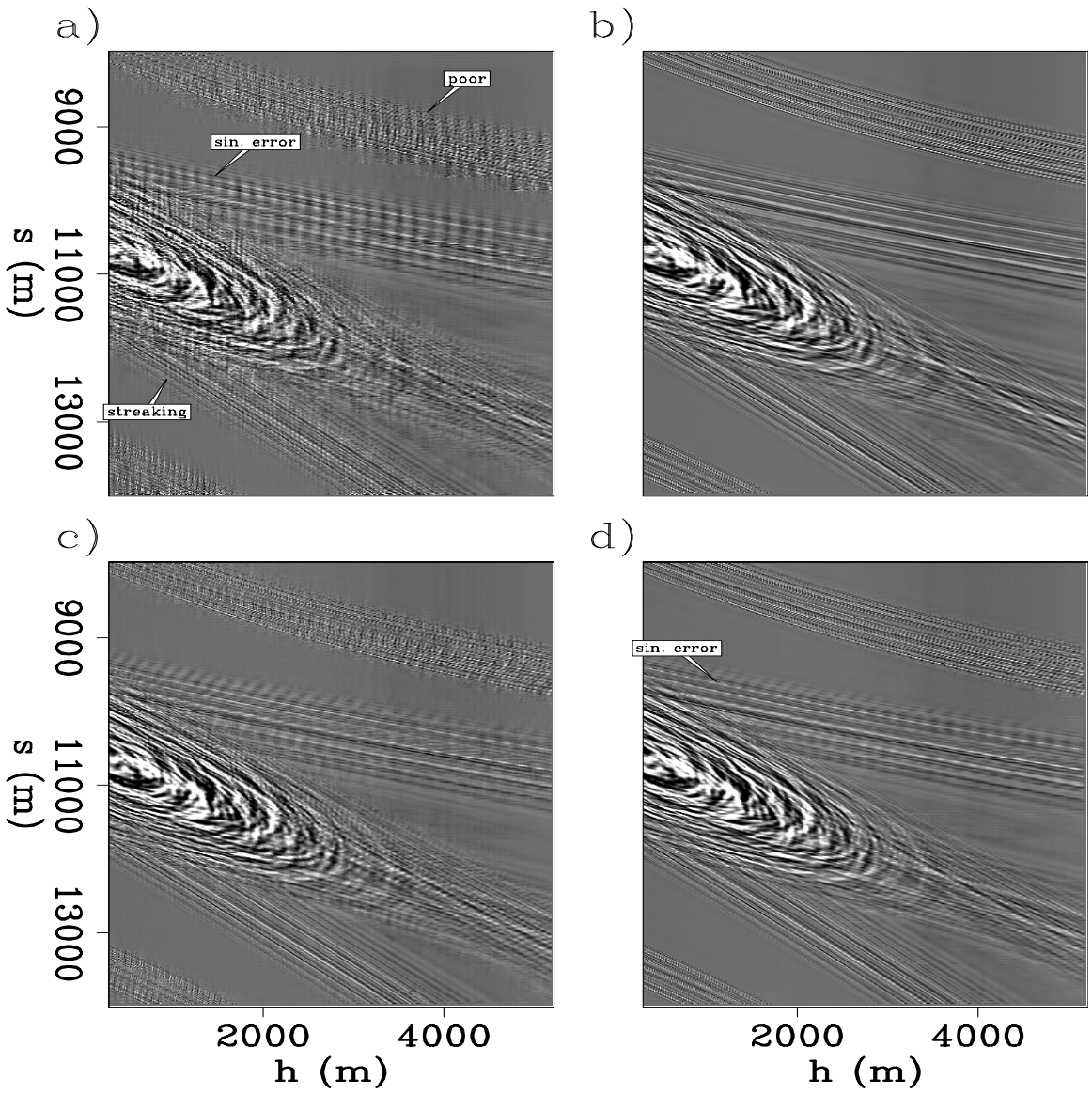

Figure 5.12: Time slices from 4.336 s in inline source and inline offset, where the inline source axis has been interpolated by a factor of three. a) 2D interpolation along constant inline-offset sections. b) 3D interpolation along inline-source, inline-offset cubes. c) 3D interpolation along inline-source, crossline-offset cubes. d) 4D interpolation along inline-source, inline-offset, crossline-offset hypercubes. The 3D inline-source, inline-offset interpolation in Figure 5.12b produces a superior result to all others, particularly on the linear events at the center of the slice, with no vertical streaking or sinusoidal error. **ER** fxNS/. exxoncoffinterpslicecompann

the 2D interpolation along only the source axis was insufficient, other methods produce good results, with the 3D inline offset approach yielding the best result. This is surprising given that the 4D interpolation uses more data at the same time. I attribute the poorer performance of the 4D interpolation to the short length of the added crossline offset axis. Since the same set of filter coefficients are used for multiple receiver cables, and the nonstationary PEF regularization also operates over receiver cables means that information across this axis is mixed, while the few points to not contribute enough to the PEF to counter this effect. Now that the problem of inline source interpolation on these data has been addressed, and a solution similar to that for 2D prestack data has produced the best result, I next consider crossline receiver interpolation, for which a prestack 2D algorithm is useless.

Synthetic Data: Crossline receiver interpolation

Crossline receiver interpolation is a much more difficult problem than inline source interpolation because of the small number of samples along the interpolated axis, the sparser sampling of crossline receivers, and the larger factor of interpolation required. Applying a simple 2D interpolation along crossline offsets will not produce an acceptable result, as only 12 points are being used at any one time. Increasing the dimensionality of the problem is an obvious approach, but the question remains as to which dimensions are the most useful and if increasing the dimensionality of the problem is worthwhile. For these data, I interpolate crossline receiver cables using 2D, 3D, and 4D approaches. In addition to judging the results along the interpolated axis, I again examine the results along all other axes, as examining interpolated receiver cables, constant-inline offset sections, and various time slices produces further insight into the results.

I test four interpolation methods here. All four methods have the same preprocessing performed on the data as that in the inline source interpolation test. The data are again NMO-corrected at water velocity, followed by dividing the time axis into 40 overlapping patches of 64 samples each, followed by a Fourier Transform over

the time axis. The training data for the PEFs are the same data, but extended in length by a factor of two by zero-padding along the time axis (for each patch) before Fourier transformation to produce data, with half the sampling interval in frequency. The PEFs were estimated using 60 iterations of a conjugate-direction solver, while the interpolation was also performed with 60 iterations of the same solver.

The first approach is two-dimensional, in which a 1D PEF of three complex elements that vary every second sample is estimated across the crossline offset axis and used to interpolate that one-dimensional vector for each of the $410 \times 160 \times 40 \times 32$ problems, for a total number of coefficients that equals the number of data values. I then test two 3D interpolation methods, one in cubes of crossline offset and inline offset for a shot-by-shot interpolation, with a 2D 3×6 PEF that varies every fourth point on both axes, now applied independently on all 160 shots, 40 patches, and 32 frequencies of the data. The second 3D interpolation method over crossline offset and inline source positions again uses a 3×6 PEF that varies every fourth point along both axes and independently for all 410 inline offsets, 40 patches and 32 frequencies, again for a number of coefficients that roughly equals the size of the training data. Since I do not wish to interpolate the inline receiver axis and assume it to be well-sampled, I subsample the training data along that axis when estimating the PEF, so that the result is not interpolated over the inline receiver axis. However, along the aliased inline source axis I do not subsample the training data and instead subsample the interpolated result over the inline source axis. This is because of the small number of receiver cables that subsampling along the crossline offset axis would produce. Finally, I test the full 4D interpolation method, where the 3D $3 \times 6 \times 6$ PEF varies every fourth point along the crossline receiver axis and every fifth point along the inline source and inline receiver axes. This 4D approach is applied independently for each of the 40 patches and 32 frequencies.

Looking at the differences among these interpolations along the axis of interpolation, the crossline offset axis, provides little insight, as seen in Figure 5.13. The original data in Figure 5.13a do not have many observable trends except for the flat water-bottom and lower reflectors, and some diffractions that are sloping to the left

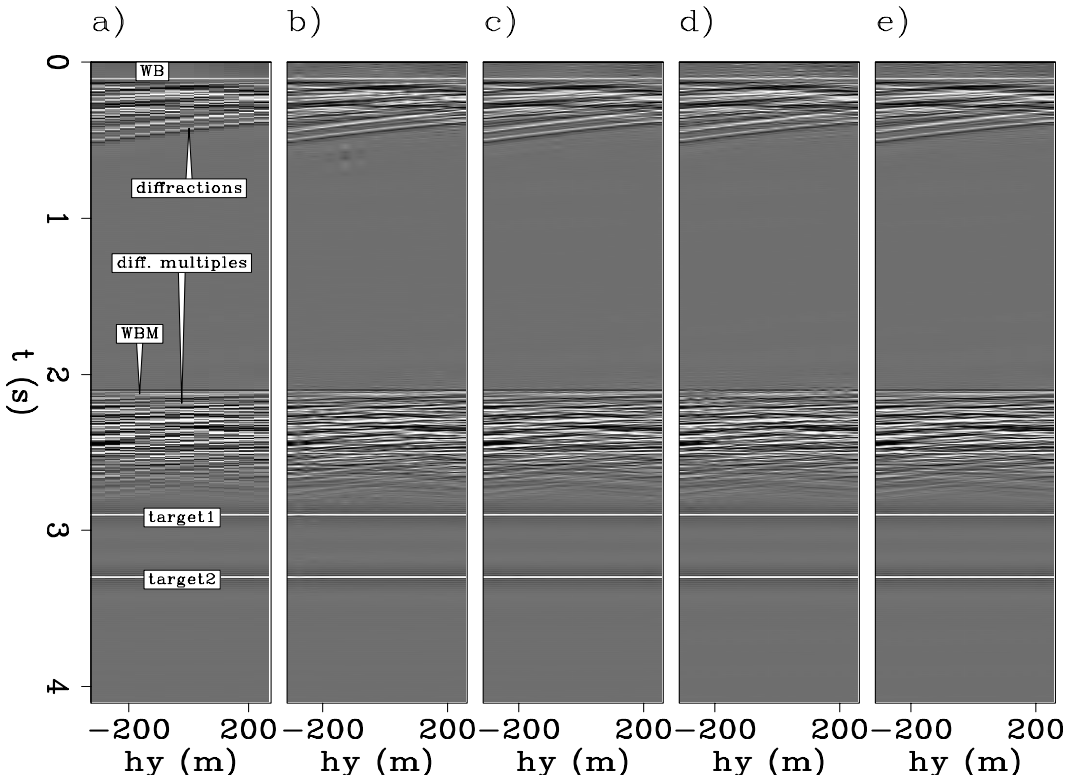

Figure 5.13: Receiver cable interpolation viewed for a single source at 10931.25 m and a single inline offset at 900 m. a) original data. b) 2D interpolation of crossline offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, inline-source cube interpolation. e) 4D interpolation. The 4D interpolation shows the most continuity. **ER** fxNS/. exxoncableinterpcxoffann

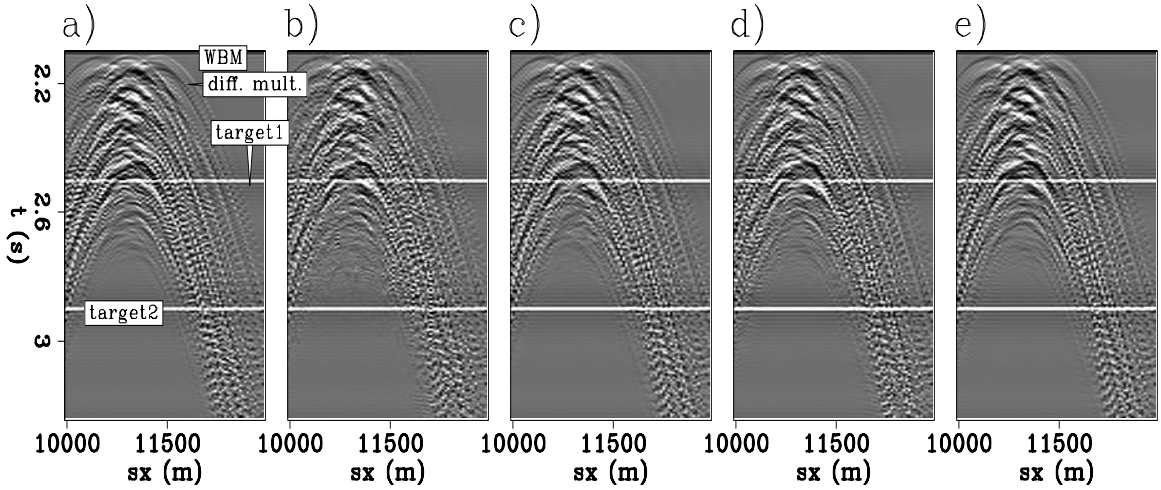

Figure 5.14: Receiver-cable interpolation viewed for constant-offset (900 m) sections. a) original data from a nearby receiver cable. b) 2D interpolation of crossline-offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, inline-source cube interpolation. e) 4D interpolation. Looking at the bottom of the cloud of diffracted multiples, the 3D and 4D results appear to be comparably better than the 2D result. **ER** fxNS/. exxoncableinterpcoffann

after the water bottom. The 2D result is the least believable, with obvious distinctions between the recorded and interpolated data. The 3D crossline offset, inline source interpolation (5.13c) is the next worst result, with the recorded data more distinguishable from the interpolated data in both the diffracted multiples as well as the primary diffractions at earlier times. The 3D inline result (5.13b) is able to produce a believable result in the diffractions below the water bottom, but the multiples are still not properly interpolated. Finally, the 4D result (5.13e) produces what I believe is the best result in this comparison, with the interpolated data indistinguishable from the recorded data in both the primary diffractions and the diffracted multiples.

Shown in Figure 5.14 are the results along the source axis. A constant-offset section taken from a single cable is shown in Figure 5.14a, while the nearest interpolated cable produced from the four methods comprises Figures 5.14b-e. This view is not at all enlightening, even when zoomed into the diffracted multiples below the water-bottom multiple. The only method that produces a visibly different result is

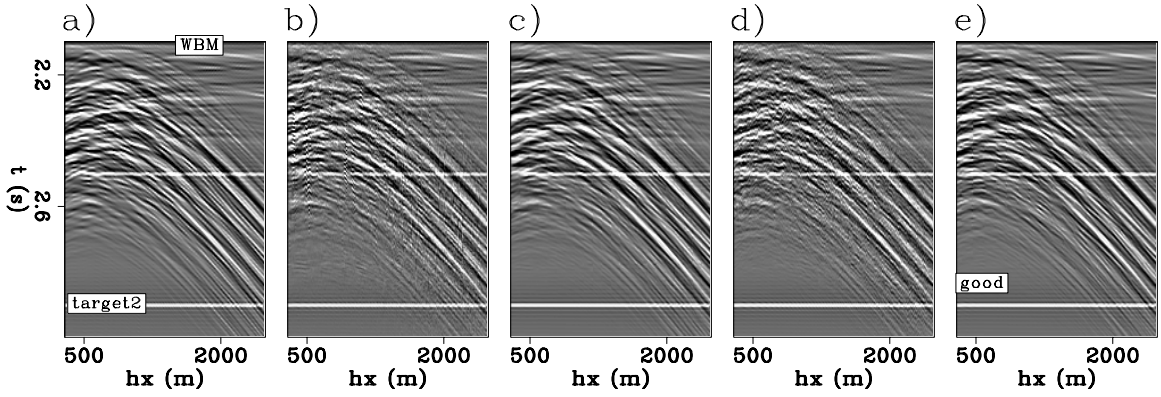

Figure 5.15: Receiver cable interpolation viewed for a single shot at 10931.25 m and a single interpolated cable at zero crossline offset. a) recorded data from a nearby receiver cable. b) 2D interpolation of crossline-offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, inline-source cube interpolation. e) 4D interpolation. The 3D inline-offset, crossline-offset and 4D interpolations produce the best results. **ER** fxNS/. exxoncableinterpshotann

the 2D approach, which shows considerable noise below the apexes of the diffracted multiples. The base of the cloud of diffracted multiples appears the clearest in the 4D result.

Another better-sampled axis along which to view the data is the inline-offset axis, shown in Figure 5.15. Again zoomed on the diffracted multiples, the nearby recorded cable (Figure 5.14a) looks continuous and noise free, while the 2D interpolation (Figure 5.14b) and the 3D inline-source, crossline-receiver interpolation (Figure 5.14d) both contain what appears as noise. This noise arises because each of the inline offsets for these cases were treated as independent problems, whereas the 3D inline-offset, crossline-offset interpolation (Figure 5.14c) and the 4D interpolation (Figure 5.14) operate over the entire axis shown in this figure.

Having examined various vertical sections involving the time axis, let us now look at various time slices through the four-dimensional result. Figure 5.17 shows a time slice through a single shot record, where the 12 cables have been interpolated by a factor of two to yield a total of 23 cables. The flank of the diffractions can be seen at

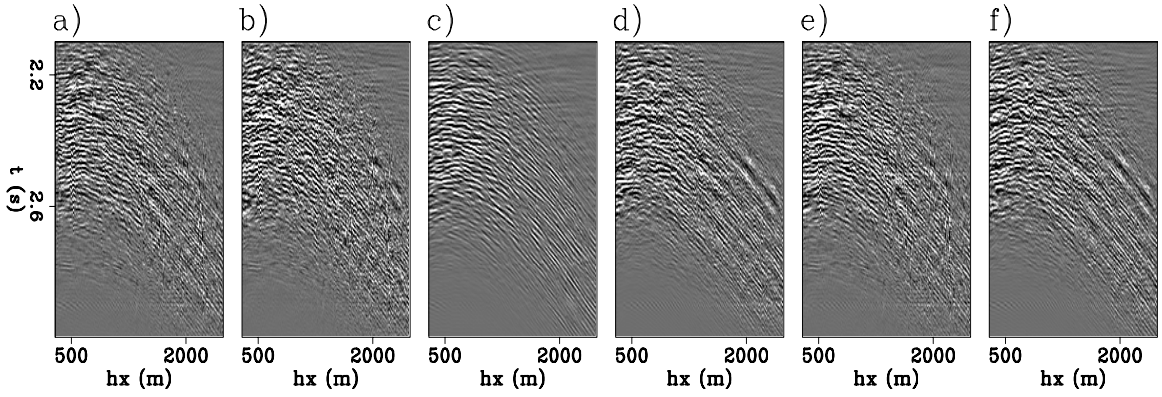

Figure 5.16: Differences between images in Figure 5.15. a) 2D vs. 3D receiver interpolation; b) 2D vs. 3D source interpolation; c) 3D receiver vs. 4D interpolation; d) 3D source vs. 4D interpolation; e) 2D vs. 4D interpolation; f) 3D source vs. 3D receiver interpolation. The 3D crossline receiver, inline receiver and 4D interpolations are most similar, with the 3D source interpolation producing errors at far offsets. **ER** fxNS/. exxoncableshotdiff

roughly 3500 m inline offset, and the apexes of the diffractions are at roughly 1000 m offset. Because of the small number of cables present, the figure is quite short in h_y, making it difficult to discern differences in the images. The 2D interpolation in Figure 5.17a shows some noise at the far crossline offsets and far inline offsets where the acquired data are visible. At near offsets, noise is also visible. The 3D inline offset, crossline offset interpolation in Figure 5.17b is more continuous than the 2D result at far inline offsets, while the 3D crossline offset, inline source interpolation in Figure 5.17c is considerably more discontinuous than either the 2D or the 3D inline result throughout the slice. Finally, the quality of the full 4D interpolation in Figure 5.17d is somewhere between the two 3D results, with a slight loss in continuity of the interpolated data at far inline offsets.

Finally, I show an inline-offset, inline-source time slice for each of the four methods from below the water-bottom reflection, where the flanks of the diffractions appear as strong linear events. In this comparison (Figure 5.18), all of the data are created by interpolation. The 2D interpolation in Figure 5.18a gives what appears to be a

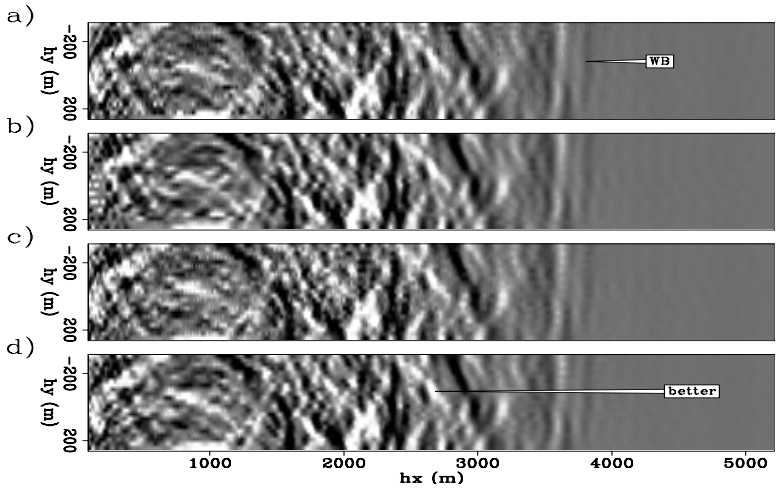

Figure 5.17: Receiver cable interpolation viewed in a time slice at 2.276 s through a crossline-offset, inline-offset cube at $s_x = 10931.25$ m. a) 2D interpolation of crossline-offset gathers. b) 3D crossline-offset, inline-offset cube interpolation. c) 3D crossline-offset, inline-source cube interpolation. d) 4D interpolation. The 3D crossline-offset, inline-offset and 4D interpolations produce the most continuous result. **ER** fxNS/. exxoncableinterpxosliceann

visually pleasing result, as do the results in Figures 5.18b and 5.18d. The 3D crossline-offset, inline-source result in 5.18c appears to contain more noise than do the other results, with more heterogeneity across the inline-offset axis.

The crossline interpolation of this prestack synthetic example shows how some domains are better for seeing differences in interpolation results than are others. In particular, the axes that contain the recorded data are not particularly useful since the number of samples along the crossline offset axis is so small. Time slices through an inline-source, crossline-offset cube were not shown as they were even less useful for drawing distinctions among the data. Instead, comparing a combination of interpolated time slices through an inline-source, inline-offset cube and interpolated shot gathers appeared to show the most dramatic differences. These synthetic data show, somewhat surprisingly, that simply adding axes to an interpolation does not necessarily improve the result. In particular, for inline-source interpolation a 3D inline-only interpolation seems to provide the best result, although including the

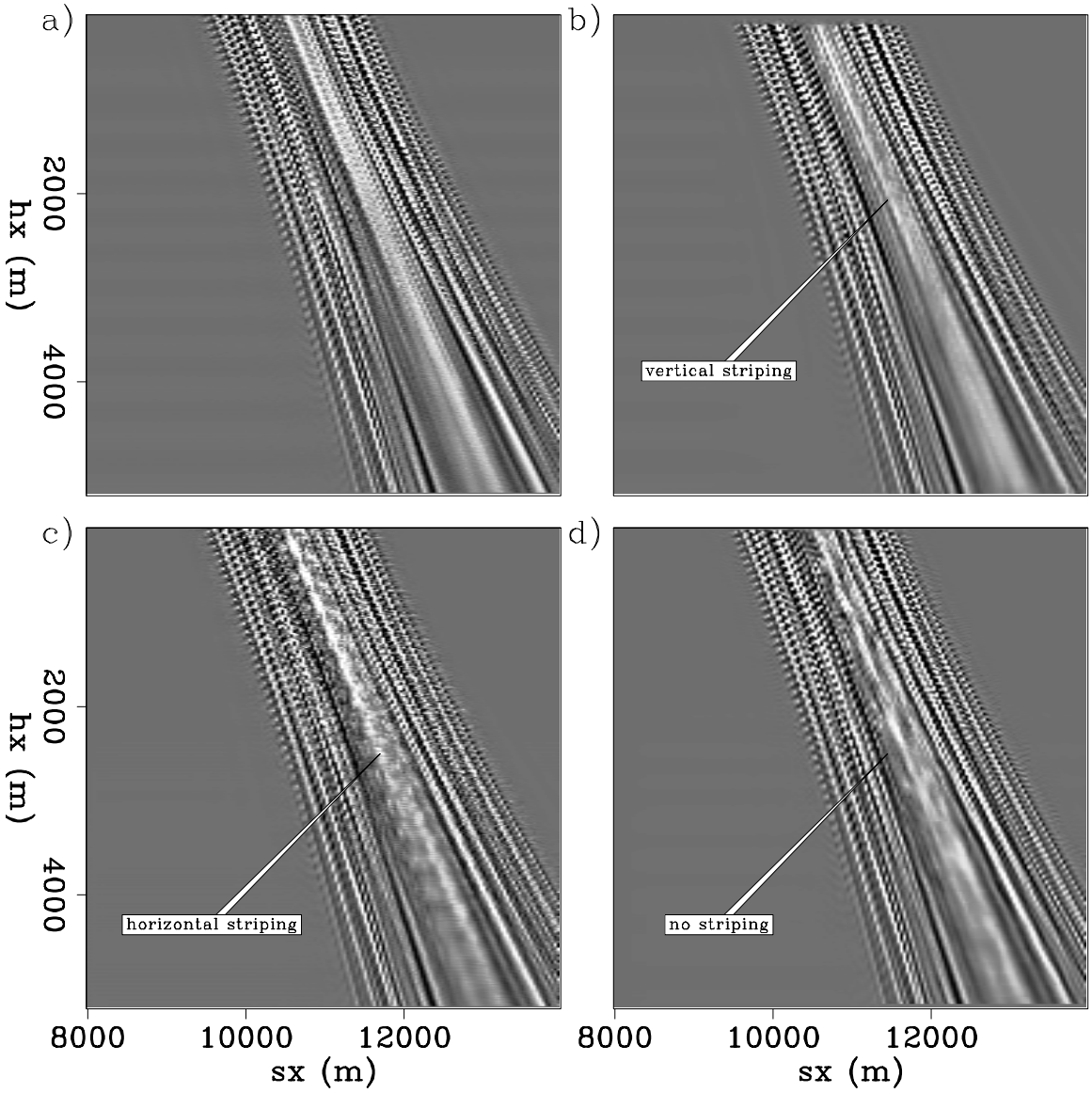

Figure 5.18: Receiver cable interpolation ($h_y = -25$ m) viewed in a time slice at 2.276 s through an inline-offset, inline-source cube below the water-bottom reflection for an interpolated receiver cable. a) 2D interpolation of crossline-offset gathers. b) 3D crossline-offset, inline-offset cube interpolation. c) 3D crossline-offset, inline-source cube interpolation. d) 4D interpolation. The 2D result is overly smooth, the 3D (crossline-offset, inline-offset) result shows vertical striping, the 3D (crossline-offset, inline-source) interpolation shows horizontal striping, and the 4D result contains more detail and has no striping. **ER** fxNS/. exxoncableinterpsosliceann

crossline-offset axis is preferable to performing a strictly 2D interpolation. Crossline interpolation shows that using the aliased inline-source axis is not as useful as the well-sampled inline-offset axis in interpolation of receiver cables, but the 4D interpolation is a marginal improvement over the other results. Next, we shall see complications that field data add to this problem, and how an even more poorly-sampled crossline offset axis changes things.

FIELD 3D MARINE DATA EXAMPLES

For the 3D prestack field data shown in Chapters 1 and 4, courtesy of CGGVeritas, I next interpolate sources by a factor of three in the inline direction, and receiver cables by a factor of four, from four to 13, the density (but not aperture) required for 3D SRME. I compare multiple approaches to this problem and show how quickly the results degrade when iteratively interpolated.

Figure 5.19 shows a schematic of the acquisition geometry of two alternating sources and four receiver cables, with the (idealized) recorded and desired sampling parameters listed in Table 5.2. The inline source and receiver spacings are ideally 25 m, while the crossline spacing is ideally 40 m, equal to the crossline distance between the two towed sources and the nearest receiver cables.

Table 5.2: Sampling of the field marine data.

Axis	Δ_{recorded}	Δ_{desired}
h_x	25 m	25 m
h_y	160 m	40 m
s_x	75 m	25 m
s_y	320 m	40 m

The actual source locations for the single sail line used in these tests (Figure 5.20a) are reasonably straight. While the receiver locations (Figure 5.20b) deviate more from the idealized straight line geometry than do the sources, the deviations are relatively minor compared with those of other sail lines in this data set. I interpolate sources along the sail line, and cables between other cables, gridding the data as if the sail

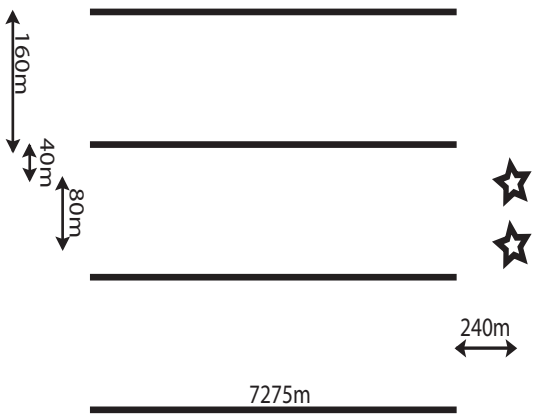

Figure 5.19: Schematic plan-view diagram of the acquisition geometry. **NR** fxNS/. realgeom

line and cables are straight. Because this interpolation is a statistical method and these deviations are smooth, the name of the spatial axis is inconsequential, but the displays of inline offset might not exactly honor the inline offset when cable feathering is included.

These data, shown in Figure 5.22, contain a water-bottom canyon with significant crossline dip, as shown in the failure of both 2D SRMP in Chapter 1 and 2D pseudo-primary generation in Chapter 4 in this canyon. This canyon, shown in the left half of the constant-offset section (at 600 m inline offset) in Figure 5.22a has a maximum crossline dip of approximately 14 degrees, measured from the provided migration velocity model in Figure 5.21, such that the apex of the reflection from this water bottom falls outside of the recording array. This reflection is spatially aliased on the edges of this canyon, both in inline source and crossline offset. In Figure 5.22b, I show a recording from the second receiver cable for a single shot at 15800 m, where this canyon in steeply dipping in the inline. This shot has been NMO-corrected using an RMS velocity generated from the migration velocity model, and because of the strong lateral heterogeneity of the velocity in this region the data as seen along the receiver cable appear overcorrected. Looking at the crossline offsets in Figure 5.22c, where both inline offset and source location are fixed at 400 and 15800 m, respectively, the slope in this coarsely-sampled axis is aliased, even after the NMO correction. These

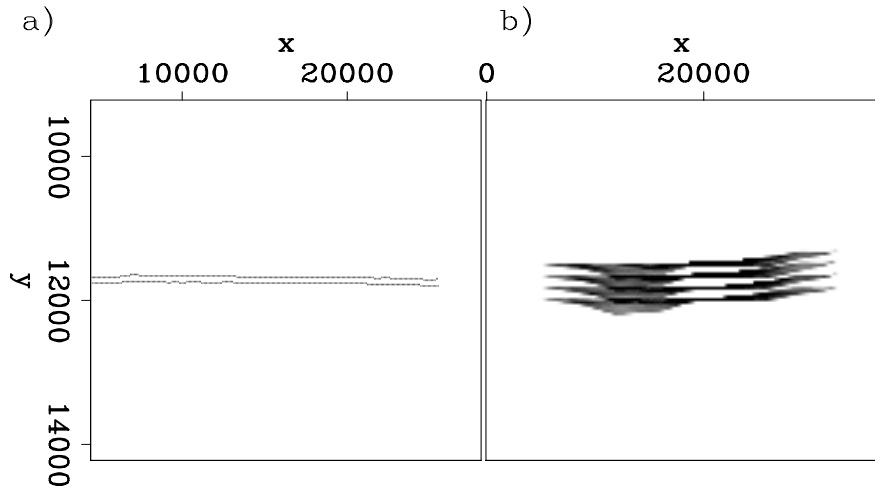

Figure 5.20: Source and receiver distributions of the input sail line: a) source positions; b) receiver position map. The sail line is relatively straight with minimal cable feathering. **NR** fxNS/. fieldsourec

data are severely undersampled, with a factor of three required in the inline-source direction, a factor of four required in the crossline-receiver direction. In the next two examples, I demonstrate nonstationary interpolation of these frequency slices in various different dimensions for both the inline-source axis and the inline-receiver axis, and then focus on interpolating both axes in an iterative fashion.

Field data: Inline-source interpolation

To interpolate the inline sources by a factor of three, I first perform an NMO correction on the data to roughly flatten the data along the offset axes. I do this prior to applying the nonstationary f-x interpolation in 30 overlapping windows of 128 samples along the time axis, in order to reduce the number of events crossing the boundaries of these time windows. Once these data have been NMO corrected and are divided into overlapping windows, I zero-pad the time windows extending them by a factor of three of the original length and then perform a temporal Fourier transform to create training data with one-third the frequencies of my desired output data. I use the

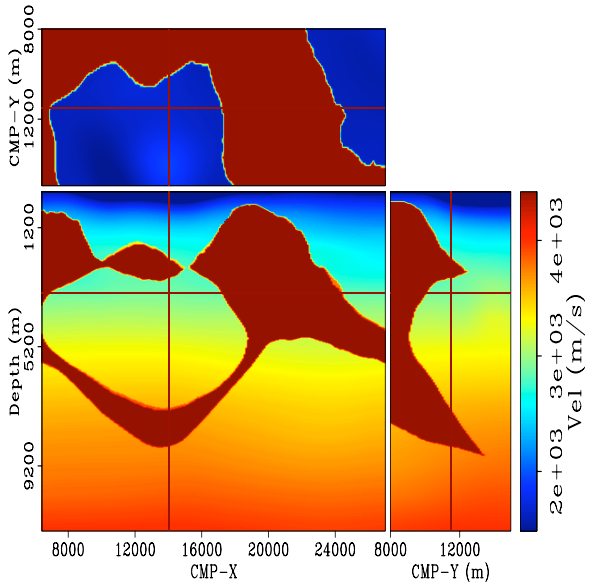

Figure 5.21: Migration velocity model of the dataset. Note the submarine canyon with a significant crossline dip. **NR** fxNS/. vmodel

unpadded data and perform the same Fourier transform to create the input data to be interpolated.

To interpolate inline sources, I first perform a two-dimensional interpolation using 5-element PEFs that vary every eight samples estimated along the inline source axis, and repeat the process for all offsets, both inline and crossline. I also perform two 3D interpolations, the first with 6×2 PEFs that operate over the inline source and crossline-offset axes, varying every five and four points, respectively, and repeat these interpolations over the inline-offset axis. The second 3D interpolation is over inline source and inline offset, using 6×2 PEFs that vary every sixteen and twenty points along the inline source and inline offset axes, respectively. Finally, I perform a full four-dimensional interpolation with all of the data in each frequency slice used simultaneously, using $5 \times 5 \times 2$ 3D PEFs operating across inline source, inline offset, and crossline offset that vary every five, eight, and four points along each axis, respectively. In all cases, the inline-offset axis is subsampled before the PEF estimation as that axis is not interpolated, and the crossline-offset axis is also interpolated when

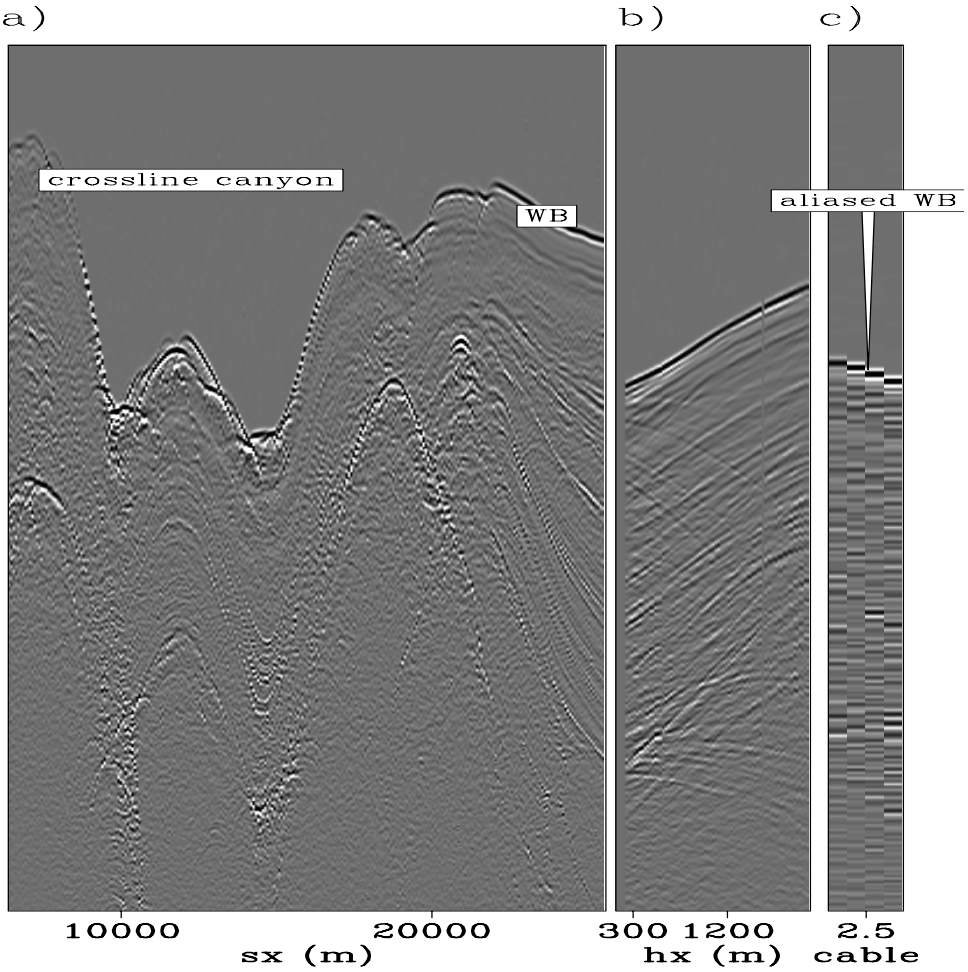

Figure 5.22: Input sail line: a) constant-offset section at 600 m inline offset and the second receiver cable; b) second cable of a shot at 15,800 m; c) crossline-offset gather at 600 m inline offset and a source position of 15,800 m. The data have been NMO corrected, and a gain has been applied. **NR** fxNS/. fieldinann

used to estimate a PEF.

I first compare these different interpolations by examining a constant-offset section, where every third trace is known, shown in Figure 5.23. The 2D interpolation result in Figure 5.23a contains more noise than do the other results, most noticeably before the first arrival. Unfortunately, it is difficult to detect differences among the other results when looking at this constant-offset section. Looking at the differences between these figures in Figure 5.24 shows that the 3D inline and 4D interpolations are the most similar, while the 3D crossline interpolation adds little to the 2D approach. In all cases the flat regions of the water bottom are well-interpolated, with the largest differences in the submarine canyon and in the salt body reflection below the water bottom.

Instead of looking at the axis along which the interpolation is done, the multidimensional nature of this problem gives us several other potential ways to display the interpolated data. Figure 5.25a shows a receiver cable identical to that in Figure 5.22b, while Figures 5.25b-e are the 2D, 3D inline, 3D crossline, and full 4D interpolated cables from the nearest shot to that for Figure 5.25a. Here the differences in the interpolation algorithms become much more obvious. The 2D result in Figure 5.25b is clearly much poorer than any of the other results. The 3D inline-source, crossline-offset interpolation in Figure 5.25c is a marginal improvement over the 2D interpolation, but still has interpolated energy appearing before the water bottom, and also lacks the second dip that is present in the recorded shot in that for Figure 5.25a. Switching the domain of 3D interpolation to inline-source and inline-offset, where the interpolation also spans the horizontal axis of the figure yields the result shown in Figure 5.25d. This result is close to that for the nearby recorded shot, excluding a dimming of the secondary slopes present in the original shot. Extraneous energy before the water bottom is completely absent. The result of full 4D interpolation, in Figure 5.25e, is not as appealing as that from Figure 5.25d. I attribute this to the nonstationary PEFs using the same coefficients for all four of the receiver cables, which spreads information local to each cable. The interpolated receiver cable in this case, however, also does not contain any energy before the water bottom. The

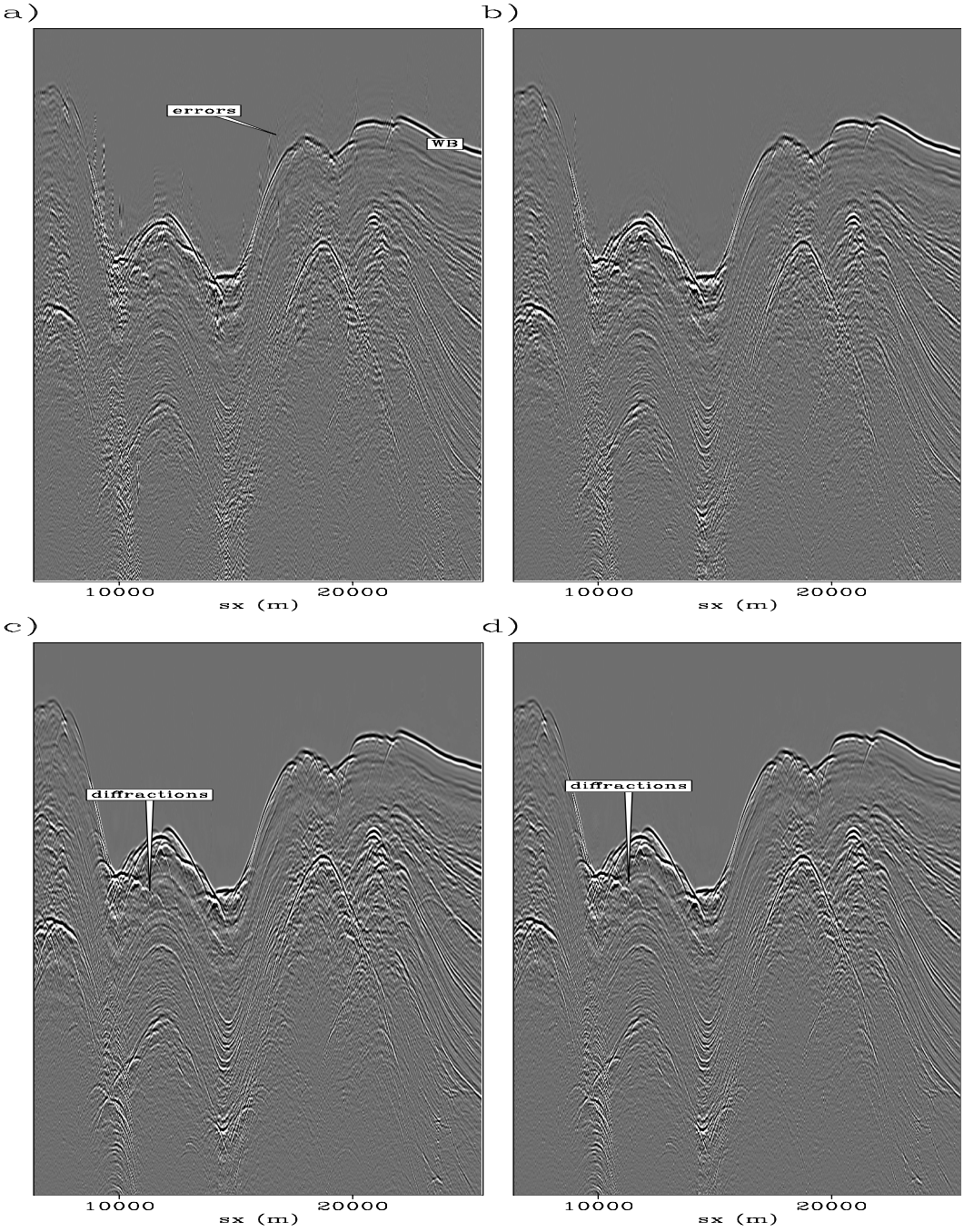

Figure 5.23: Interpolation of sources in a constant-offset section by a factor of three: a) 2D b) 3D (inline source, inline offset); c) 3D (inline source, crossline offset), d) 4D. The differences in this view are minor, with slight amounts of energy appearing before the water-bottom in the 2D interpolation. **NR** fxNS/. fieldinlinesourceann

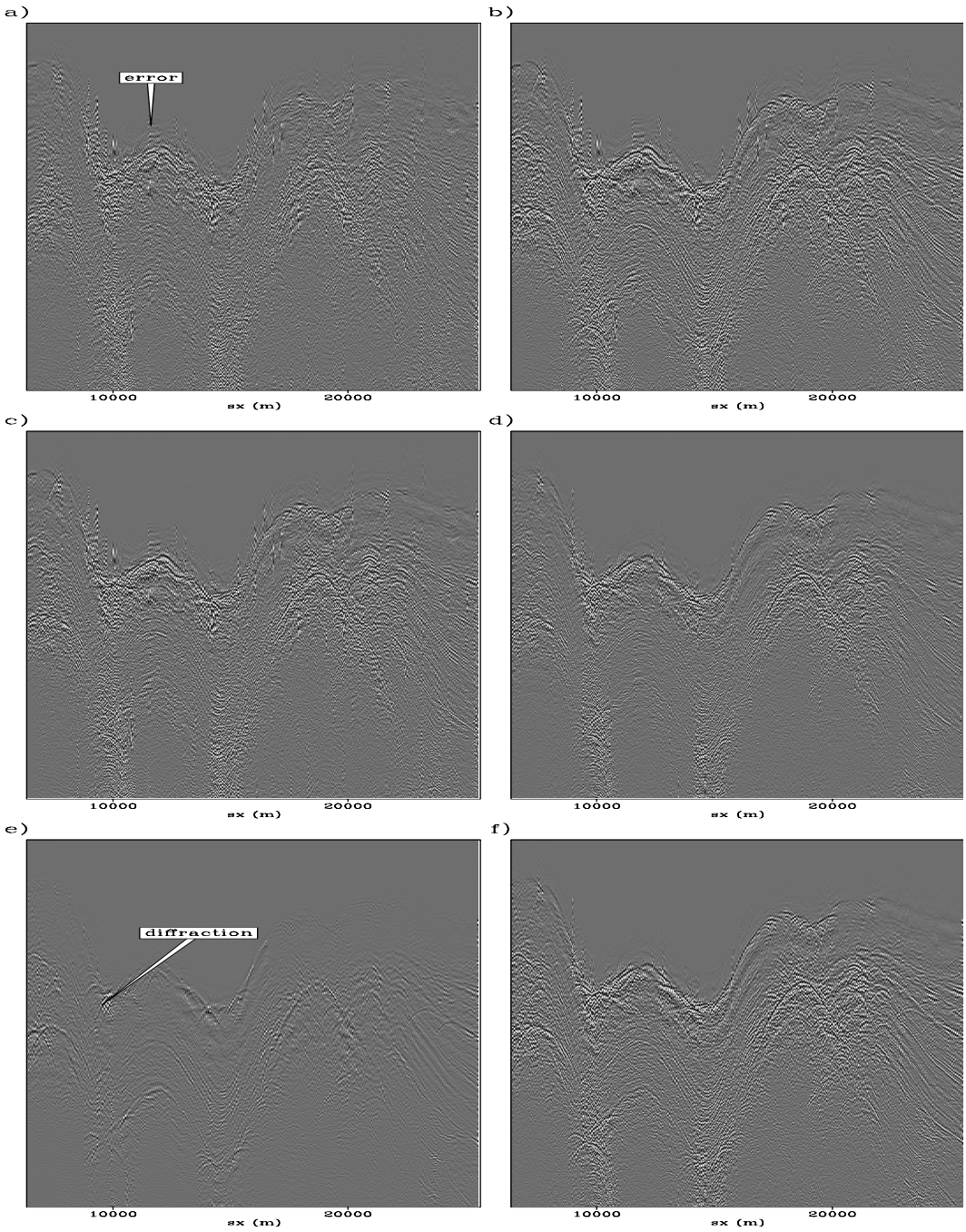

Figure 5.24: Differences between images in Figure 5.23. a) 2D vs. 3D crossline offset interpolation b) 2D vs. 3D (inline source, inline offset); c) 2D vs. 4D; d) 3D crossline offset vs. 4D; e) 3D inline offset vs. 4D; f) 3D inline vs. 3D crossline. **NR** fxNS/. fieldinlinecoffdiffann

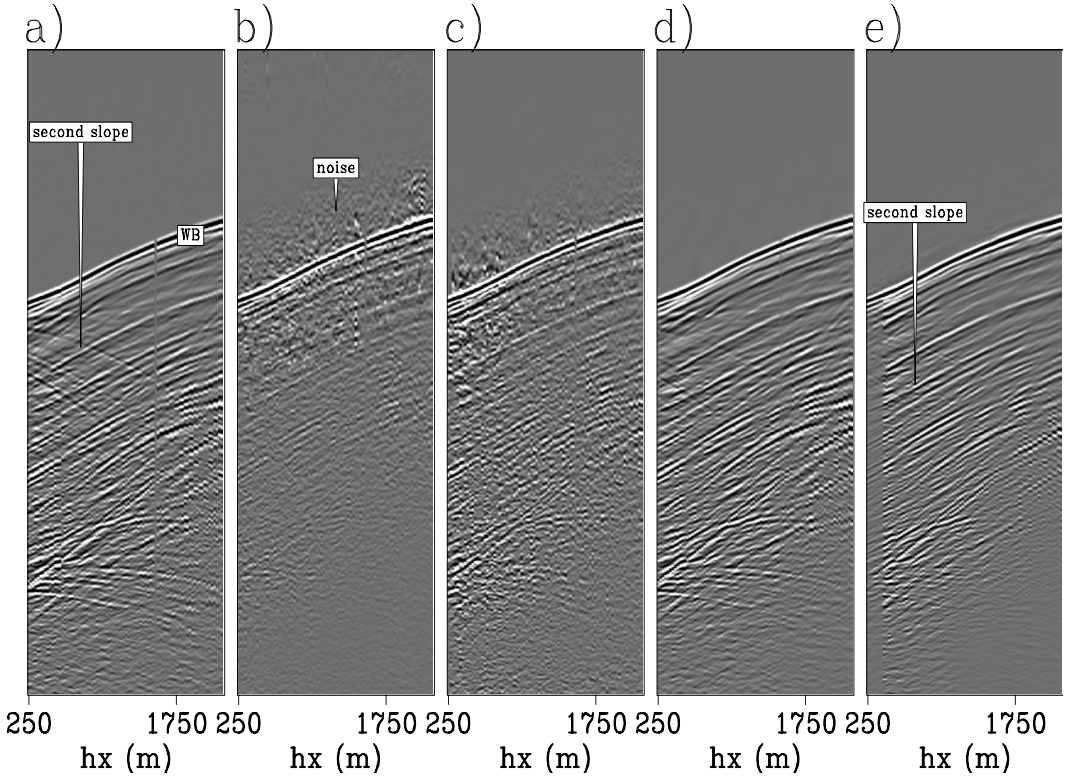

Figure 5.25: Near offsets of interpolated shot gathers: a) nearest recorded shot gather from 25 m away; b) 2D constant-offset-section-based interpolation; c) 3D (inline-source, crossline-offset) interpolation; d) 3D (inline-source, inline-offset) interpolation; e) 4D. The differences among the results are much more dramatic than in the inline source view, with the 3D inline result appearing most like the nearby data. **NR** fxNS/. fieldinlineshotann

difference panels in Figure 5.26 show a surprising difference between the 3D inline and 4D results. with the amplitude of the 4D result much lower than the 3D inline result and much of the second slope missing. I attribute this to the smearing over the crossline offset axis introduced by reusing the same filter coefficients over all of four crossline offsets. Since this source location is in an area with great crossline heterogeneity, this produced a less accurate result.

Finally, an inline-source, inline-offset time slice, where the slice intersects the water-bottom reflection, is shown in Figure 5.27. Here the 2D interpolation in Figure

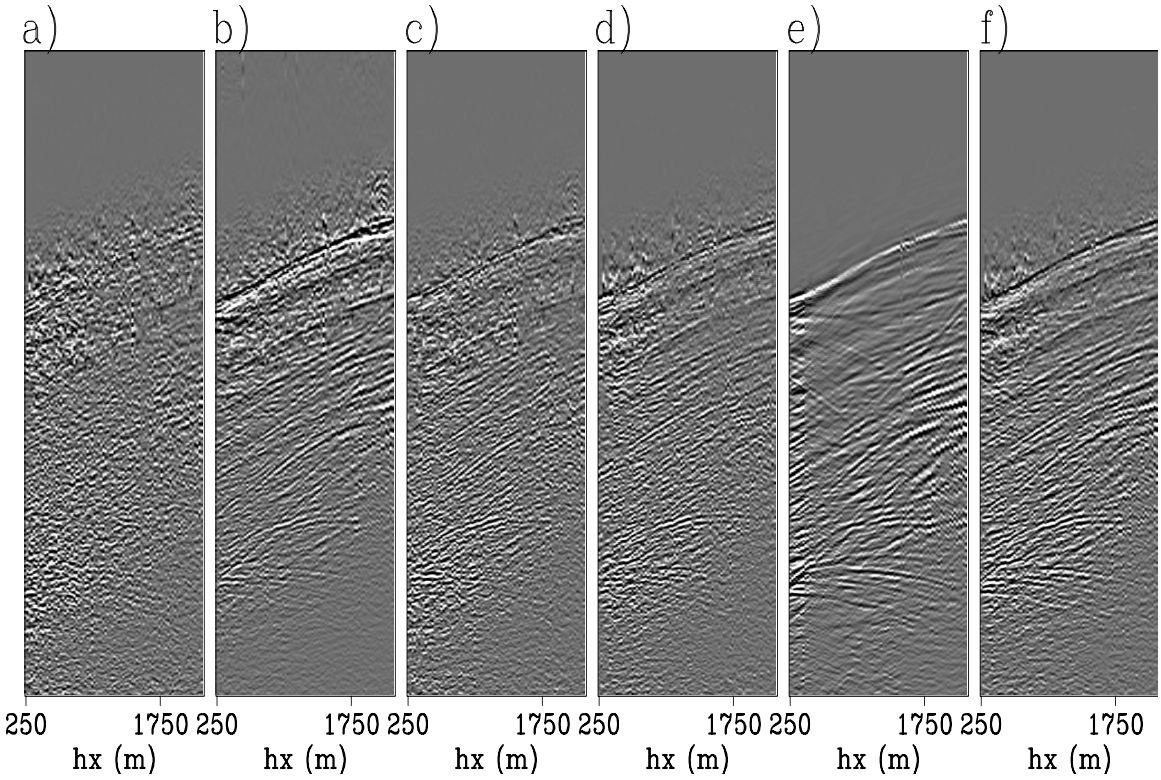

Figure 5.26: Differences between images in Figure 5.25. a) 2D vs. 3D crossline offset interpolation b) 2D vs. 3D (inline source, inline offset); c) 2D vs. 4D; d) 3D crossline offset vs. 4D; e) 3D inline offset vs. 4D; f) 3D inline vs. 3D crossline. The 3D inline and 4D results show the greatest coherent differences. **NR** fxNS/. fieldinlineshotdiff

5.27a contains energy before the water bottom at roughly 14000-15000 m source position, while all offsets appear rough and poorly interpolated. The 3D inline-source, crossline-offset in Figure 5.27b contains some variability from one inline offset to another, as each inline offset is treated as an independent problem. Meanwhile, the 3D inline-source, inline-offset result in Figure 5.27c is continuous along both axes, as is the 4D interpolation in Figure 5.27d.

This inline-interpolation test illustrates how examining axes other than those in which the data are interpolated gives greater insight than does simply comparing the data along the interpolated axis. Here, as in both of the synthetic examples, the 3D inline interpolation produces a preferable result to that of the more elaborate 4D

interpolation. Next, we examine crossline-receiver interpolation for these same data.

3D field data: Crossline offset interpolation

These field data have been acquired with four cables spaced by 160 m; meanwhile a recording density of 40 m in the crossline offset axis is desired, such that there is a source at every receiver location. I perform three different approaches to interpolating these data, all using nonstationary frequency-space PEFs. I do not attempt a 2D approach, as using only four data values for each interpolation problem is unlikely to be successful. Instead, I use a 3D inline-source, crossline-offset domain approach, a 3D inline-offset, crossline-offset, or shot-by-shot approach, and finally a full 4D approach, with the entire sail line interpolated at once. Again the data are NMO-corrected, broken into patches, and then Fourier transformed to produce 64 frequency slices for each of the 30 patches, for a total of 1920 inline-source, inline-offset, crossline-offset cubes (each $256 \times 300 \times 4$ samples) for this sail line.

The 3D inline-source, crossline-offset interpolation uses a 2D 5×2 PEF that varies every tenth inline source and does not vary over the four crossline offsets. I solve a separate problem for each of the 300 inline offsets for a total of roughly 103 million filter coefficients. The other 3D interpolation is over the inline offset and crossline offset axes, with a 2D 5×2 PEF that varies every ten points on the inline-offset axis and does not vary on the crossline-offset axis, where each of the 256 shots is interpolated independently. The 4D interpolation uses 3D $5 \times 5 \times 2$ PEFs that vary every 20 points on the inline-receiver axis, every 16 points on the inline-source axis, and do not vary over the four points on the crossline-offset axis, for a total of 17 million filter coefficients. All of these problems are again solved with a conjugate-direction solver with 60 iterations for each PEF-estimation problem and 60 iterations for each interpolation problem once the PEF has been estimated. I now compare multiple views of results for these different approaches.

Figure 5.28 shows a constant-offset section generated for an interpolated cable created between the two central cables of the sail line. Figure 5.28a is the result

of the 3D inline-offset, crossline-offset interpolation, where each trace is interpolated separately from the other offsets at that source location. The interpolation looks good, although a few traces, mostly in the submarine canyon, contain obvious errors. The other 3D interpolation result in Figure 5.28b does not appear as clean as the source-by-source result in Figure 5.28a. In particular, the diffractions below the water bottom at 19000 m and 21000 m source location appear weaker, while energy appears before the water bottom in the submarine canyon. Finally, the full 4D result in Figure 5.28c is much clearer than either of the 3D results, with little random noise present. Unlike the 3D inline-source, crossline-offset result there is almost no errant energy arriving before the water bottom. Next, we examine these same results by looking at an interpolated receiver cable for a single shot.

Let us now compare these same interpolation results by examining a receiver cable for a single shot at 12950 m, in the submarine canyon (Figure 5.29). Note two important details in the recorded third receiver cable in Figure 5.29a: first, a single receiver at roughly 450 m offset has flipped polarity, and, second, backscattered energy (a second weaker slope in the recorded data) appears in the middle of the section. The first 3D result (Figure 5.29), the source-by-source interpolation, captures both the main energy and the secondary slope, and is difficult to distinguish from the nearby recorded data except at the near offsets. The flipped polarity of a trace has degraded the result: several interpolated traces surrounding the trace with flipped polarity are influenced. The 3D inline-source, crossline-offset result in Figure 5.29c is not so obviously degraded by the single flipped trace as in the other 3D example, but is much noisier, in part because the horizontal axis of the figure is not the axis used in interpolation. Also, this result does not contain the second slope present in the recorded data. Finally, the full 4D result in Figure 5.29d shows no errors caused by the flipped polarity of the nearby trace. Because of the added dimensionality of the interpolation, the single bad trace is averaged out by the larger amount of data and additional axes of regularization used in the filter estimation. While this result does not contain noise from the bad trace, it also does not contain much of the backscattered energy (possibly signal), with the second slope much dimmer in the 4D interpolated data than in either the recorded data or the source-by-source

interpolation in Figure 5.29b. This can again be linked to the added information from the inline source direction; this energy is not consistent over the 16 shots over which the filter does not vary.

Finally, in Figure 5.31 is a time slice through an interpolated inline-source, inline-offset cube, from slightly below the water-bottom reflection, giving yet another view for comparing the interpolation results. Figure 5.31a is an inline-source, inline-offset section from the recorded second receiver cable. This cable does not contain the near-offset trace with the flipped polarity. The 3D source-by-source interpolation produces the result in Figure 5.31b, where we again see the distortion of the flipped trace in the nearby cable at the near offsets. The 3D inline-source, crossline-offset result in Figure 5.31c does not contain the near-offset problem in Figure 5.31b, but varies strongly from one inline offset to another. The 4D result is not influenced by the polarity-reversed trace and also is smooth along inline offsets. The 4D result looks by far the most like the slice from the nearby cable, although this slice is from a region that does not contain the backscattered energy.

This crossline-offset interpolation, more than any other example so far, shows the need to view the result along as many axes as possible to diagnose problems. The 4D interpolation gives the most consistent and robust results, but not necessarily the best interpolation in all areas at all times and locations. This is because, in total, fewer filter coefficients are being estimated because of the lesser variability of the filter coefficients in the 4D interpolation, whereas in the 3D case a separate set of filter coefficients is estimated for each inline source. While these differences for a single interpolation can be subtle, they are amplified as data are iteratively interpolated, which is often necessary in order to generate enough data for adequate sampling, discussed next.

Field data: Iterative interpolation

In order to interpolate data by large factors, it is known it is preferable to interpolate in several steps, by first interpolating data by a small factor to partially increase data

density, and then interpolating the previously interpolated data to create greater data density along that same axis. The explanation for this can be thought of in the context of a larger problem. A PEF might be estimated on fully-sampled data that includes both known and unknown data, which can be thought of as a nonlinear problem in which unknown filter coefficients are also convolved with unknown data. I treat this problem as two linear steps in which the nonlinear part of the data is ignored, and these unknown data are removed from the PEF estimation. Instead of making larger assumptions in order to treat larger interpolations in two linear steps, I instead interpolate the data by a smaller amount, assume that data are correctly interpolated, and estimate a PEF on the interpolated as well as the original data to create the remaining unknown data. For example, this approach would interpolate by a factor of four in two steps of a factor of two instead of a single factor-of-four interpolation in which only the lower quarter of frequencies is used.

Another example is the interpolation of these field data with the data interpolated by a factor of three in the inline-source direction and a factor of four along the crossline offset axis. Interpolating in multiple dimensions is straightforward as long as the interpolation factor is the same. Since it is not the case in this situation, the data would have to be interpolated by a factor of twelve on all axes simultaneously in order to generate data at both the factor of three and factor of four desired. Instead, I interpolate first by a factor of three on the inline source axis, then by a factor of two on the crossline offset axis, and then by a factor of two again on the crossline axis to generate the total factor of twelve needed. I show how the quality of the interpolation degrades as this iterative interpolation proceeds, and how this degradation appears along the many axes in these data.

Previously, only having four samples, the crossline offset axis was too poorly-sampled to view a useful section containing that axis. Even after the factor of two interpolation in the previous section, the seven traces were not enough to construct a reasonably wide slice. Now I interpolate the entire sail line from the previous section first by a factor of three along the inline-source axis, then twice by two factors of two in the crossline offset axis for a total of 13 traces in the crossline-offset direction. I

show a crossline-offset section for a single source and a single inline-offset in Figure 5.32, where Figure 5.32a is the input data with four receiver cables. Little of the character of the data is obvious from this image, other than that the water-bottom reflection is spatially-aliased along this axis. This source and offset location is the worst-case scenario from this sail line, the area with the greatest amount of crossline heterogeneity, steepest slope, and strongest aliasing. Figure 5.32b shows the receiver cables interpolated by a factor of four, for the four original crossline offsets from a recorded shot. While the water-bottom reflection is properly interpolated, producing an unaliased output, the amplitude of the interpolated traces decreases at later times in the section. Figure 5.32c shows the same interpolated crossline-offset section for the next (interpolated) source; all data in this image are interpolated. The difference between the data interpolated from originally recorded cables and the interpolated cables is larger, with the original traces having a higher amplitude. Now that we have viewed the axis of interpolation, let us look at other views containing this previously unseen axis.

Viewing the time slice through an inline-source, crossline-offset cube as in Figure 5.33 shows both interpolated axes at once. Figure 5.33a is a slice through the input cube, where the slice cuts through the water-bottom reflection in the submarine canyon from roughly 9000 to 16000 m source position. Iterative interpolation produces the result in Figure 5.33b. The aliased water-bottom reflection at 12000 m is interpolated with a minimum of acausal ringing. Upon closer inspection, the four original cables are visible with slightly higher amplitude than along the interpolated cables.

Figure 5.34 contains a time slice through a inline-offset, crossline-offset cube for a single recorded shot (Figure 5.34a) as well as for a recorded source with interpolated cables (Figure 5.34b), and the same cable interpolation for an interpolated shot (Figure 5.34c). In this view, the difference between results for a recorded shot and for an interpolated shot is small. Next, rather than showing images containing the interpolated crossline-offset axis, let us look at interpolations from nearby recorded receiver cables.

Figure 5.35 shows three constant-inline-offset sections from 425 m inline offset. Figure 5.35a is a constant-offset section for a recorded receiver cable, with the source axis interpolated by a factor of three. Once this factor-of-three interpolation has been performed, the receiver cables are interpolated by a factor of two to produce the result in Figure 5.35b. In the largely two-dimensional areas of the data, the data are nicely interpolated, with diffractions from the water-bottom at 20000 m source position well interpolated, although the speckled areas of the source interpolation in Figure 5.35a are not interpolated in subsequent steps. The areas in the submarine canyon with large crossline variability are not as well interpolated, both because of that and the small number of samples in the crossline offset direction. The weak diffractions emanating from the center of the water-bottom canyon that are correctly interpolated in the inline source interpolation are poorly treated by the crossline offset interpolation. The second iteration of the crossline source interpolation produces a cable between the previously interpolated cable in Figure 5.35b and the recorded cable in Figure 5.35a. The deterioration of the result is less noticeable from that of the inline-source interpolation followed by the first crossline offset interpolation. The water-bottom-event, even in the submarine canyon, is nicely interpolated, and most of the diffractions in the largely two-dimensional areas are still present. The interpolation degrades in the submarine canyon, both in the diffractions immediately below the water-bottom and in the reflectors beneath the water-bottom reflections, which become less continuous.

Finally, in Figure 5.36, consider a single receiver cable from a single shot, and compare the original recorded data with the different stages of this iterative interpolation. Figure 5.36a shows the near offsets of the third receiver cable for source location at 9200 m. This location is in the most challenging area of the survey, with lots of crossline heterogeneity in the data. The residual moveout in Figure 5.36a is quite steep after NMO-correction, and the data are aliased in the crossline offset direction. Moving from the recorded data in Figure 5.36a to the data created by inline-source interpolation in Figure 5.36b, where the source is 25 m away, the interpolation creates data that are reasonably difficult to tell apart from the recorded adjacent data. The backscattered energy is almost entirely present, with some of the energy halfway down

the section absent. The water-bottom event is almost identical to that in the recorded data. In the first factor-of-two crossline-offset interpolation in Figure 5.36c, a cable between the third and fourth cable is created using the 4D interpolation described in the previous section. A significant amount of backscattered energy is still present in these created data. The water-bottom reflection is still reasonably well-interpolated, although slight anticausal ringing and a drop in frequency of that reflector are present. The backscattered energy appears to stop at the nearest offsets, perhaps because of a switch from one set of filter coefficients to another. Also, the proximity of this filter to the edge causes some edge effects to smooth out the information present in the filters. Proceeding from the first factor of two interpolation to the second in Figure 5.36d, where another receiver cable is created between the third receiver cable and the interpolated cable in Figure 5.36c, little has changed. There is a slight increase in the amount of ringing before the water bottom, but the backscattered energy is still present as are all of the major features present in the original recorded data.

The iterative interpolations produce good results for the first factor of three, and then degrade noticeably for the crossline interpolations.

CONCLUSIONS AND FUTURE WORK

Nonstationary interpolation in frequency and space is an efficient and reasonably-accurate way of interpolating large quantities of data in many dimensions. The smaller size of frequency-by-frequency filters allows for higher-dimensional interpolation to be performed than in a t-x formulation, while the nonstationary approach removes the need for the many overlapping spatial patches of a traditional f-x approach, an approach in which higher-dimensional interpolation becomes less feasible because of the increased amount of overlap required with each additional dimension.

Applying this approach to prestack synthetic 3D data, I examined the results along many different axes, many showing different aspects of the result. Inline-source interpolation of these synthetic data are best served by an inline approach interpolating in 3D with inline sources and inline receivers in a cable-by-cable approach.

For conventional marine-streamer data, the added crossline-offset axis contains too little additional information to be of use. The crossline-offset interpolation of these synthetic data tell a similar story, wherein a shot-by-shot 3D interpolation performs well.

This story changes with field data. The 3D inline-source, inline-offset interpolation still is the most reliable for inline-source interpolation, while the crossline-offset interpolation results differ, with the full 4D interpolation producing a less detailed but more robust result than the shot-by-shot interpolation that succeeded with the synthetic example. I then iteratively interpolated these data by first interpolating inline sources, then crossline receiver cables twice. The largest drop in accuracy is from the inline-source interpolation to the crossline-receiver interpolation, not from the factor of two to the factor of four in the crossline interpolation as I originally expected.

The data in this chapter all had a relatively small number of samples along the crossline offset axis. With a wide azimuth acquisition using towed streamers, the number of samples along this axis increases considerably, making a filter that includes the crossline offset axis more useful. The 4D interpolation that includes this axis used in this chapter should be more useful for these types of wide-azimuth data.

In this chapter, I have focused on inline source and crossline receiver interpolation, but not crossline source interpolation. The nonuniform spacing of sail lines as well as the changing cable feathering from one sail line to another introduce further complications, as well as the need to extrapolate as well as interpolate data. This nonstationary-PEF based approach, when used in addition to one of the many pre-existing move-out or partial migration-based approaches can produce enough data for a 3D SRME result.

ACKNOWLEDGEMENTS

I would like to thank ExxonMobil Upstream Research Company for the 3D synthetic dataset, and CGGVeritas for the 3D field data set.

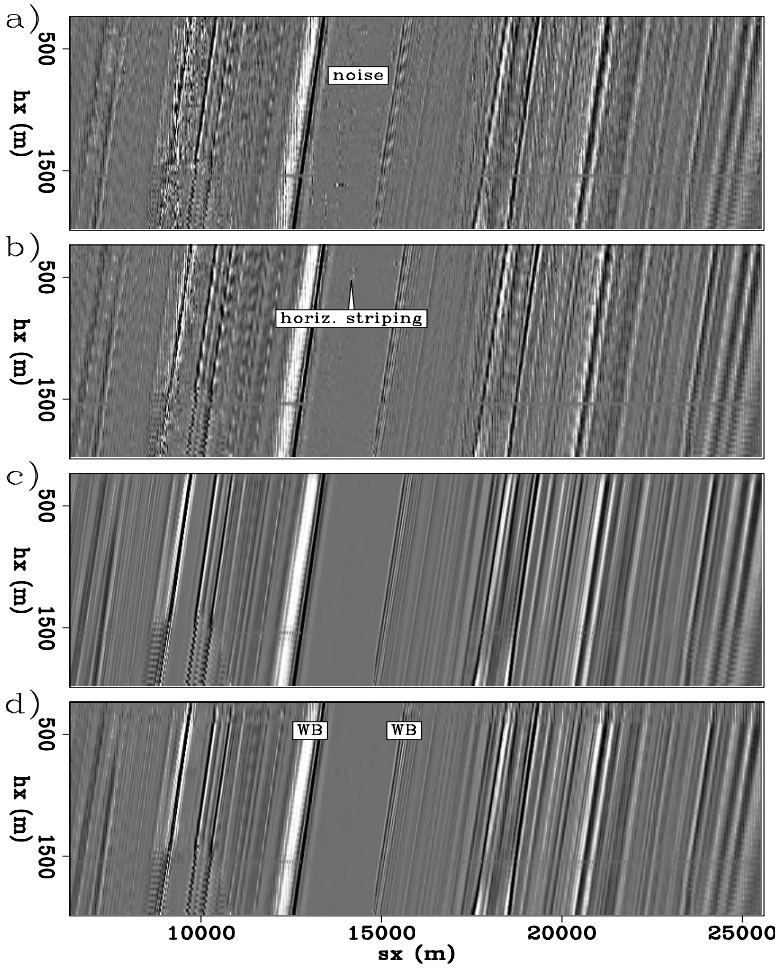

Figure 5.27: Time slices through inline-source, inline-offset cubes interpolated along the inline-source direction by a factor of three. a) 2D constant-offset-section-based interpolation; b) 3D (inline-source, crossline-offset) interpolation; c) 3D (inline-source, inline-offset) interpolation; d) 4D. **NR** fxNS/. fieldinlinesliceann

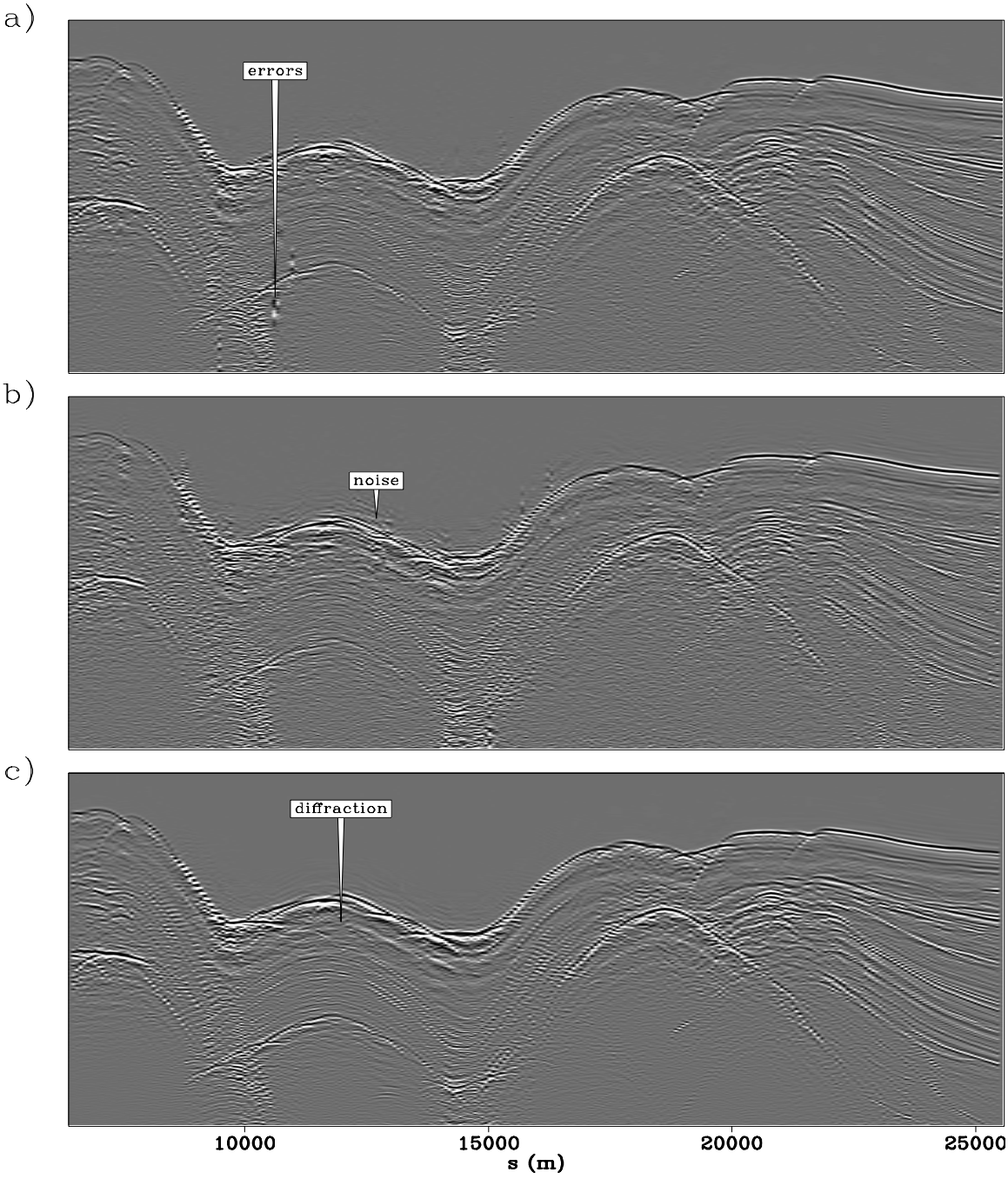

Figure 5.28: Receiver cable interpolation of field data. a) 3D (inline-offset, crossline-offset); b) 3D (inline-source, crossline-offset); c) 4D interpolation. The 4D result shows the most continuity, with no noise on the traces ad little anticausal noise. **NR** fxNS/. fieldxlinecoffann

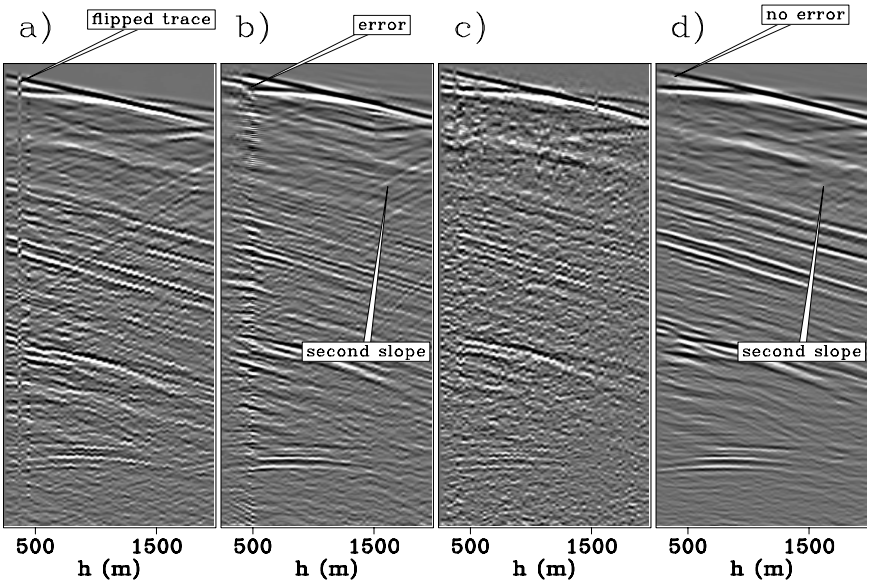

Figure 5.29: An interpolated receiver cable between the second and third cable, for a single shot. a) recorded data from the third cable; b) 3D (crossline-offset, inline-offset) shot-by-shot interpolation; c) 3D (crossline-offset, inline-shot) inline offset-by-offset interpolation; d) 4D interpolation. The flipped polarity of the near-offset trace in (a) causes large errors in (b). The 4D result contains less of the second dip present in (a) but is not degraded by the flipped trace in (a). **NR** fxNS/. fieldxlinesourceann

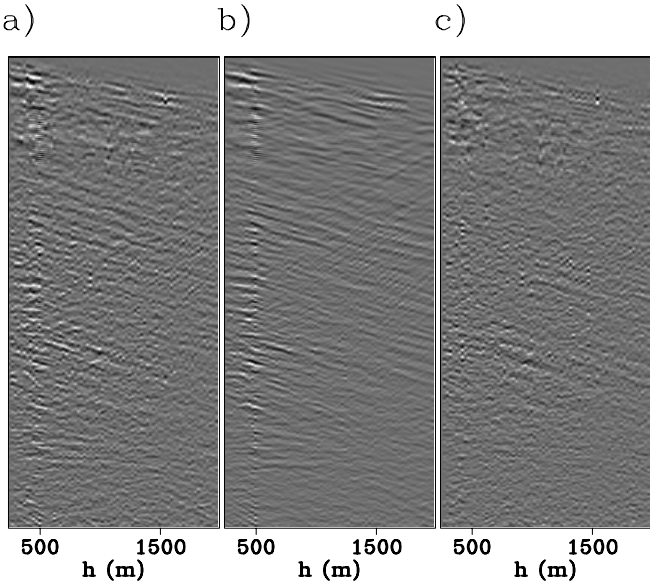

Figure 5.30: Differences between images in Figure 5.30. a) 3D inline offset, crossline offset vs. 3D inline source, crossline offset; b) 3D inline offset, crossline offset vs. 4D; c) 3D crossline offset, inline source vs. 4D interpolation. The 4D result is missing of the the second slope but also fortunately lacks the noise generated from the flipped trace. **NR** fxNS/. fieldxlinesourcediff

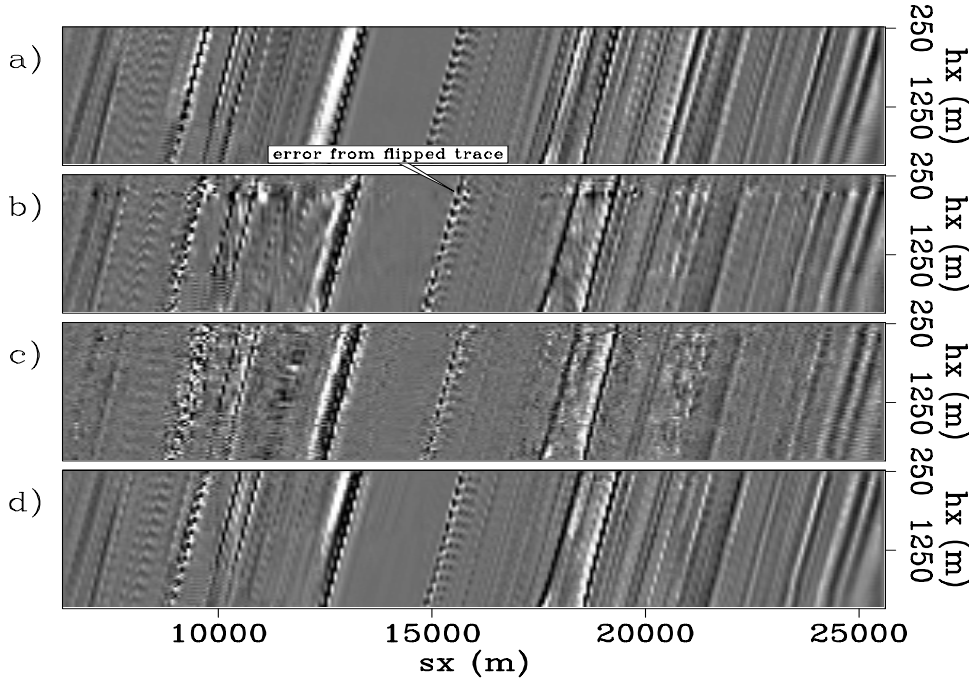

Figure 5.31: Time slice from an inline source-receiver cube of an interpolated receiver cable. a) recorded second receiver cable; b) 3D inline-source, inline-offset interpolation; c) 3D inline-source, crossline-offset interpolation; d) 4D interpolation. The 4D result is definitely the best for this view. **NR** fxNS/. fieldxlinesliceann

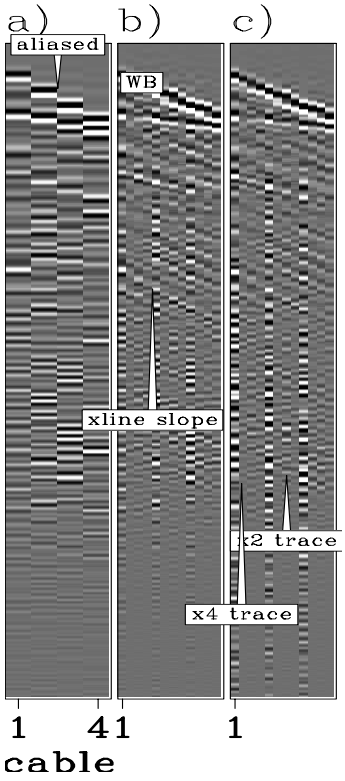

Figure 5.32: Crossline-offset sections of the interpolated data. a) recorded data; b) recorded data with interpolated cables; c) interpolated shot with interpolated cables. The amplitudes between the recorded traces and the interpolated traces becomes apparent at later times. **NR** fxNS/. fielditercxoffann

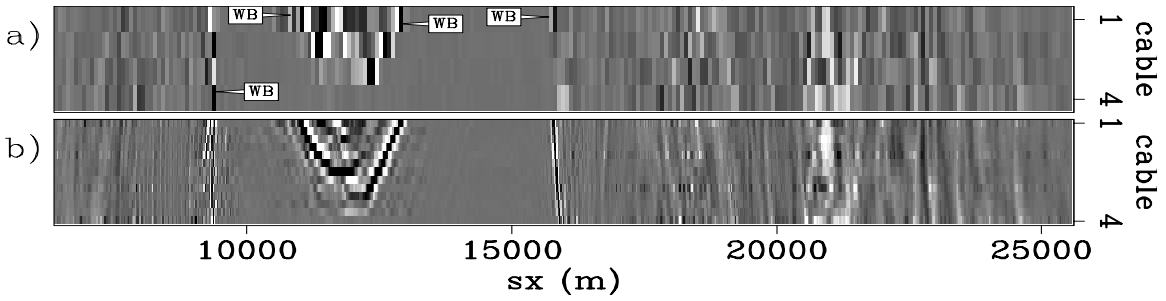

Figure 5.33: Time slice through an inline-source, crossline-offset cube. a) original recorded data; b) data interpolated by a factor of three along the inline-source and a factor of four in the crossline-offset direction. The aliased water-bottom reflection is smoothly interpolated. **NR** fxNS/. fielditerxssliceann

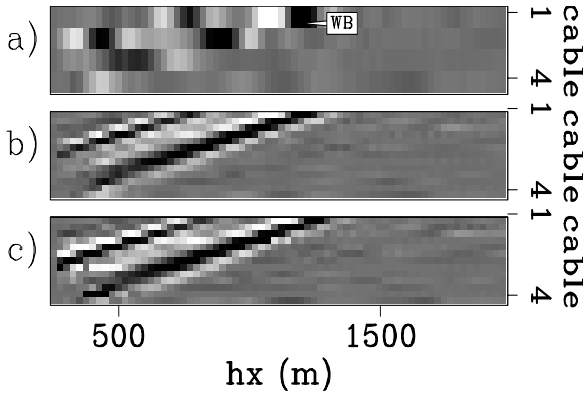

Figure 5.34: Time slice through a single shot (inline-offset, crossline-offset) cube. a) original recorded shot; b) original shot with interpolated receiver cables; c) interpolated shot with interpolated receiver cables. The source interpolation does not appear to produce a degraded result. **NR** fxNS/. fielditerxosliceann

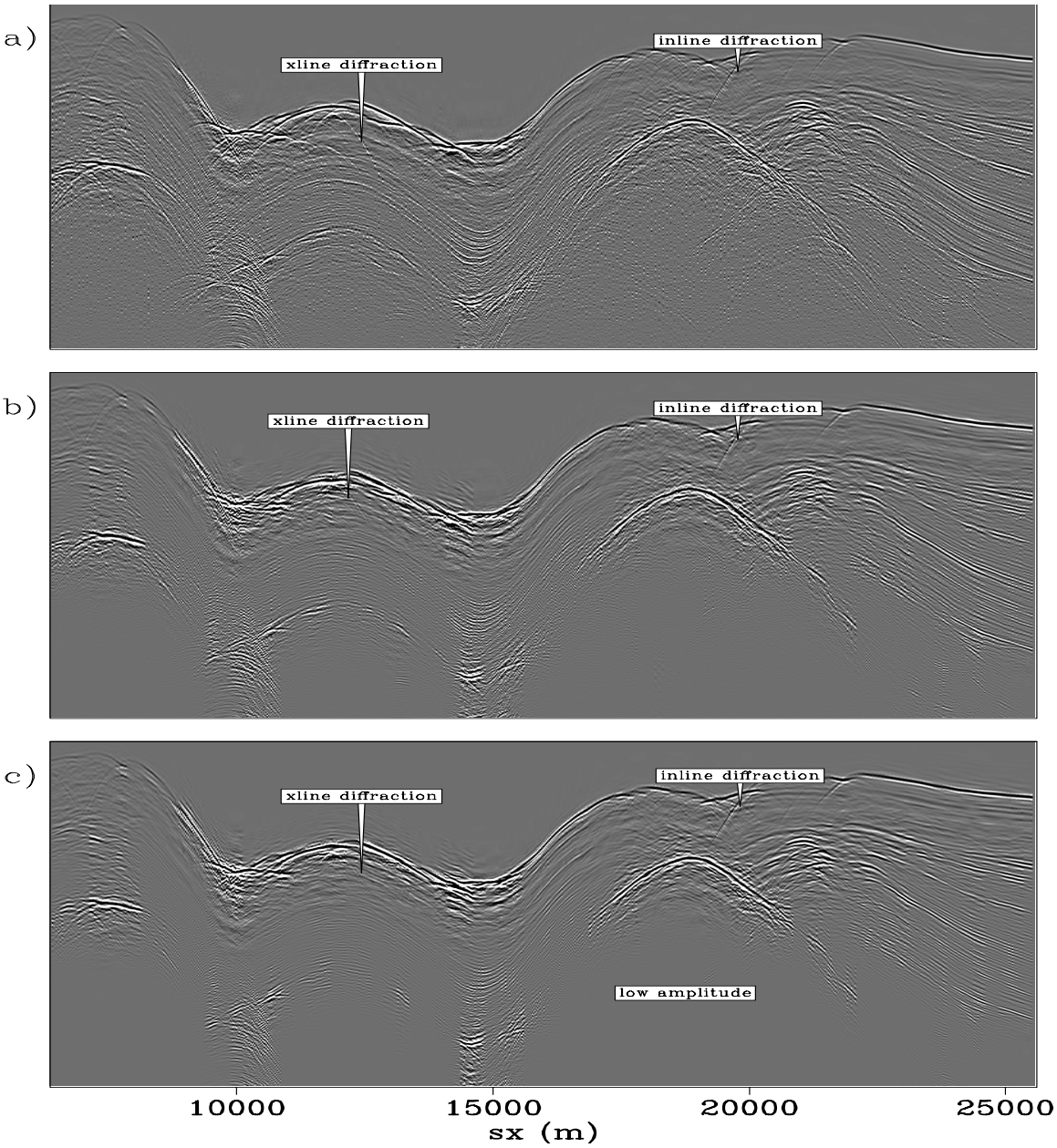

Figure 5.35: Constant-inline-offset sections from the interpolated data: a) along the second recorded receiver cable with interpolated sources; b) along a receiver cable interpolated by a factor of two (between the second and third cables); c) along a receiver cable interpolated by a factor of four (between (a) and (b)). The quality of the interpolation degrades with the number of passes applied. especially in the area with large crossline variability. All figures have been gained. **NR** fxNS/. fieldcoffdegradeann

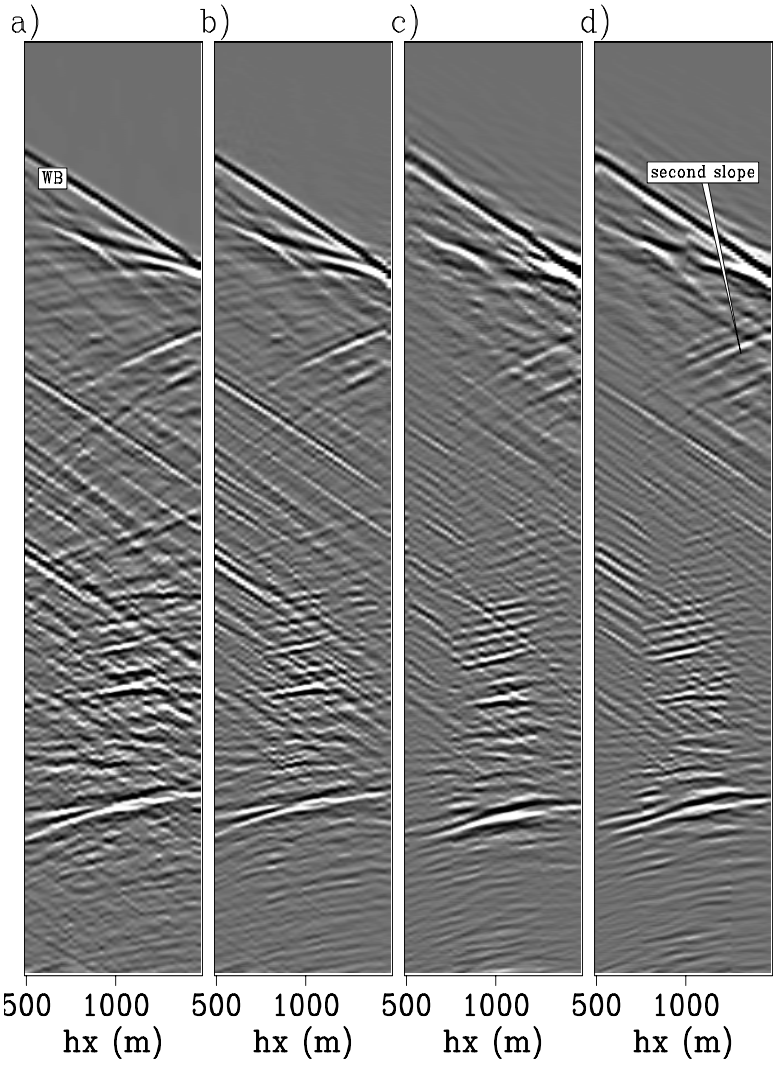

Figure 5.36: Interpolated receiver cables. a) originally recorded receiver cable; b) receiver cable from an interpolated shot; c) receiver cable interpolated by a factor of two from interpolated shot; d) receiver cable interpolated by a factor of four from the interpolated shot. This shot is from the region of the data with the greatest crossline heterogeneity, making this a worst-case scenario for these data. **NR** fxNS/. fieldshotdegradeann