INTERPOLATION WITH PREDICTION-ERROR FILTERS AND TRAINING DATA

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF GEOPHYSICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

William Curry

June 2008

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Jon F. Claerbout) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Biondo L. Biondi)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Norman Sleep)

Approved for the University Committee on Graduate Studies.

Abstract

With finite capital and logistical means, interpolation is a necessary part of seismic processing, especially for such data-dependent methods as surface-related multiple elimination. One such method is to first estimate a prediction-error filter (PEF) on a training data set, and then use that PEF to estimate missing data, where each step is a least-squares problem. This approach is useful because it can interpolate multiple simultaneous, aliased slopes, but requires regularly-sampled data. I adapt this approach to interpolate irregularly-sampled data, marine data with a large near-offset gap, and 3D prestack marine data with many dimensions.

I estimate a PEF from irregularly-sampled data in order to interpolate these data onto a regular grid. I do this by regridding the data onto multiple different grids and estimate a PEF simultaneously on all of the regridded data. I use this approach to interpolate both irregularly-sampled 3D synthetic data and 2D prestack land data using nonstationary PEFs, both with reasonable results.

Marine data typically contains a near-offset gap of several traces, which can be larger when surface obstacles are present, such as offshore platforms. Most methods that depend on lower-frequency information from the data fail for these large gaps. I estimate nonstationary PEFs from pseudoprimary data, which is generated by cross-correlating data within each shot, so that the correlation of multiples with primaries creates data at the near offsets that were not originally recorded. I use this approach in t-x-y and f-x-y, on both the Sigsbee2B 2D prestack synthetic dataset, and a 2D prestack field data set. While this method is currently unfeasible in the crossline direction in 3D marine data, the larger crossline aperture of a wide-azimuth dataset

would improve results considerably

Finally, I estimate nonstationary PEFs in many dimensions using the approximation that slope is constant as a function of frequency, and interpolate data in two, three, four, and five dimensions simultaneously by using nonstationary PEFs on frequency slices. This method correctly interpolates multiple simultaneous aliased slopes in the data. I interpolate both prestack 3D synthetic as well as field data to the densities required for future processing.

Preface

The markings [ER], [CR], and [NR] included in the figure captions in this thesis denote the extent to which each figure is reproducible by anyone desiring to do so. The electronic version of this thesis provides the original figures and Fortran90 programs written to produce all of the results. Most programs are included locally in the chapter directories. I assume you have a UNIX-based workstation with Fortran90 and C compilers, an X-Windows environment, and the Stanford Exploration Project (SEP) software available from our website (make rules, programming libraries, and LaTeX packages).

Reproducibility is a way of organizing computational research that allows both the author and the reader to verify and regenerate results. Reproducibility also facilitates the transfer of knowledge within SEP and between SEP, its sponsors, and the geophysical community at large.

ER denotes Easily Reproducible results of processing described in the paper. I claim that you can reproduce such a figure from the programs, parameters, and makefiles included in the electronic document. The data must either be included in the electronic distribution, be easily available to all researchers (e.g., SEG-EAGE data sets), or be available in the SEP data library. Before the publication of the electronic document, someone other than me tested my claim by destroying and rebuilding all ER figures.

CR denotes Conditional Reproducibility. I certify that the commands are in place to reproduce the figure if certain resources are available. CR results have been

tested to make sure that the makefile rules exist and the source codes referenced are provided. Conditional reproducibility applies to figures produced with excessively long or parallel computing processes or when using proprietary data.

NR denotes Non-Reproducible figures. SEP discourages authors from flagging their figures as NR except for figures that are used solely for motivation, comparison, or illustration of the theory, such as: cartoons, drawings, scans, or figures taken from other publications.

Our testing is currently limited to LINUX 2.6 (using the Intel Fortran90 compiler), but the code should be portable to other architectures. Reader's suggestions are welcome. For more information on reproducing SEP's electronic documents, please visit
`http://sepwww.stanford.edu/research/redoc/`.

Acknowledgments

A Ph.D. is a much longer endeavor than I had originally thought, and I have many people to thank for getting me through it. This list is far from exhaustive, but I would like to single out people who played particularly strong roles during my extended stay at SEP.

I would like to start by thanking my senior students and mentors Morgan Brown, Bob Clapp, and Antoine Guitton. They led me toward a research project and provided many hours of conversation about it and many other topics of mutual interest in geophysics and computing. I have since had the pleasure of having Bob on my defense committee and Antoine as a supervisor during an internship, and learned even more from them as a result. I thank my cohort, Gabriel Alvarez, Brad Artman, Andrey Karpushin, and Nick Vlad, who I shared many experiences with, and who gave me much needed support. I also thank the many students who arrived before and after me, in particular Guojian Shan, Jesse Lomask, Daniel Rosales, Doug Gratwick, Alejandro Valenciano, Jeff Shragge, Yaxun Tang, Madhav Vyas, Pierre Jousselin, Ben Witten, Claudio Cardoso, Gboyega Ayeni, and Roland Gunther for many useful discussions about our research and the world at large.

I owe a large debt to Jon Claerbout and Biondo Biondi for establishing and maintaining the intellectual framework of SEP and for keeping us focused on interesting topics that have applicability in the real world. They have given me the viewpoint that research is the most fun when working on new ways to approach a difficult field data set. Diane Lau deserves much credit for keeping a group as large as SEP running smoothly, and I owe the sponsors of SEP much for their financial support.

Anyone reading this thesis should thank Ken Larner for taking what was a semi-intelligible mess and helping me rewrite and refine it into what you are reading today. He was extremely supportive and helpful during his brief stay at SEP. Norm Sleep also deserves much credit for his thorough review of this thesis. It is much stronger because of his help.

Doug Schmitt and Mauricio Sacchi at the University of Alberta sparked my interest in geophysics, and I enjoyed their courses greatly. Bill MacDonald was my supervisor during an extended internship in Calgary. He sent me out on a seismic survey for a week, which got me thinking about interpolation as a research topic.

I would like to thank my many friends at Stanford and elsewhere who helped me in my attempt to remain as well-rounded as possible. They are too numerous to mention, and are in geophysics, earth sciences, and further abroad.

Finally, I thank my family: my father and my brother for their focus on science and education, and my mother for her continual support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Acquisition of reflection seismic data is a resource-intensive process: both source and receiver positions vary along the surface of the earth, sampling data along five different axes, including the finite recording time. Fully sampling all of these axes is logistically unrealistic for many reasons: a finite number of active recording channels, surface obstacles, experimental design, and limited capital. Meanwhile, many algorithms used in the processing of reflection seismic data are designed for regularly-sampled, unaliased data with a wide aperture. As such, interpolating reflection seismic data is a common practice. The interpolation method used will often vary depending on the type of acquisition. Let us now discuss common types of acquisitions.

ACQUISITION DESIGNS

Methods of acquiring surface reflection seismic data vary considerably. Marine acquisition has typically been constrained by the use of ships such that data are typically collected along parallel lines, as both the sources and receivers are towed along the same axis, usually by the same ship. In contrast, land acquisition is typically done with intersecting source and receiver lines, with considerable deviation from this regular grid caused by surface obstacles and logistical issues. Other methods, such as ocean-bottom cable or ocean-bottom sensor surveys have still different issues. I now

discuss the two most common types of geometry that are in part addressed in this thesis. For a more thorough description of acquisition geometries, multiple textbooks are available (Vermeer, 2002; Biondi, 2006).

Land acquisition geometry

Multiple geometries of land data acquisition have been proposed, with the cross-swath method as one of the most popular. In this geometry, both the sources and receivers are placed along intersecting lines. The receiver locations are typically less influenced by surface conditions, since it is possible to place receivers where access for sources is restricted, it is easier to plant a geophone than it is to position either a vibroseis truck or a drill for shot holes. An example of a 3D survey in South America with this geometry is shown in Figure 1.1, here source locations, denoted by an 'x', are more sporadically placed than are the receiver locations, denoted by points. A large gap exists in the bottom-left of the survey, most likely caused by a surface obstacle that prevents sampling with both source and receiver. This geometry has great variability in both source-receiver space as well as midpoint-offset space. The near offsets, for example, typically contain a wider azimuth range than do the far ones.

While the sampling issues in 3D are more clearly evident, the differences between receiver and source sampling in Figure 1.1 are also applicable for a 2D survey. An example of a 2D field acquisition geometry is shown in Figure 1.2. The aspect ratio in Figure 1.2 is severely distorted, as the axes are stretched by a factor of 500 on the 'y' axis relative to the 'x' axis. This example again illustrates that in land geometries sources are more unevenly distributed than are receivers, and that surface obstacles also cause problems (albeit less dramatic) in 2D acquisition. While land acquisition contains great variability, the marine case is less irregular because the ship tows the array in a continuous fashion, as discussed next.

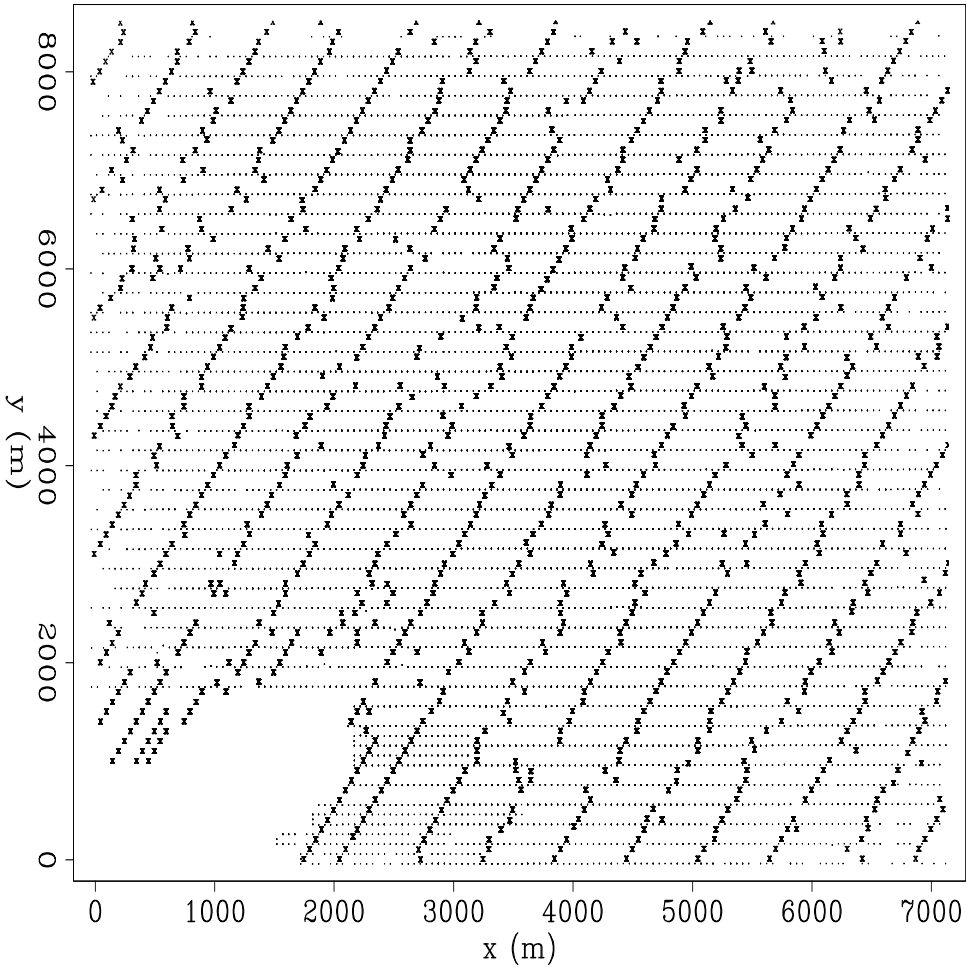

Figure 1.1: An example of the cross-swath geometry from a 3D land survey. The points are receiver locations while each 'x' denotes a source position. The receiver sampling is more regular than the source sampling. **ER** Intro/. landgeom3

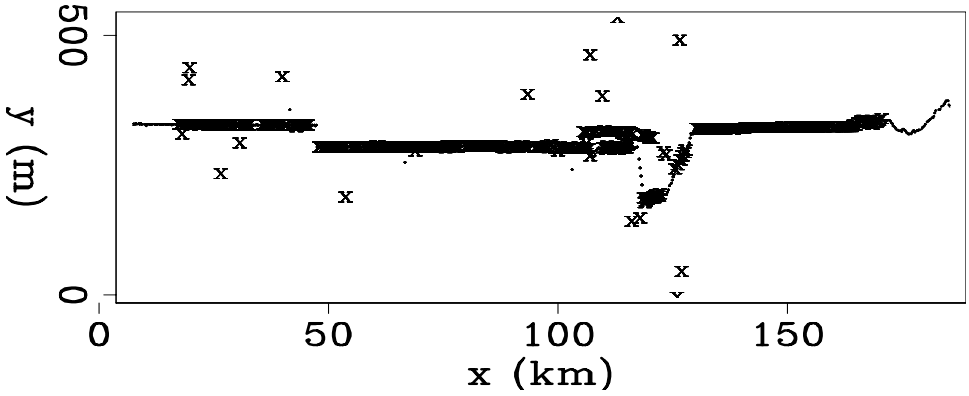

Figure 1.2: An example of receiver and source sampling for a 2D land survey. There are slight deviations along the 'y' direction for the receivers and much greater variability in source sampling. **ER** Intro/. landgeom2

Marine data geometry

Sampling in marine geometry is more regular than that in land geometry. This is largely because both the receivers and sources are towed by the same ship, so the offsets between the source and the receivers are roughly equal for each shot. One systemic gap in the data is created by the fixed distance between the source and the first receiver on the cable, leaving a near-offset gap that exists in every shot in both 2D and 3D marine surveys. Also in both 2D and 3D surveys the source interval is typically a multiple of the receiver interval since the air-gun source takes a while to recharge and the ship has traveled more than one receiver interval in the interim. To increase source sampling (and reduce the cost of acquisition) a second source is often added near the first one, creating what is referred to as *flip-flop shooting*, wherein the two sources are fired alternately. A map containing source locations for a 3D marine survey is shown in Figure 1.3. In this map, the two closely-spaced parallel lines are the alternating flip-flop sources for a single sail line. We can also see the comparatively sparse source distribution in the crossline (y) direction compared to the inline (x) direction. This usually results in severe aliasing in the crossline source direction.

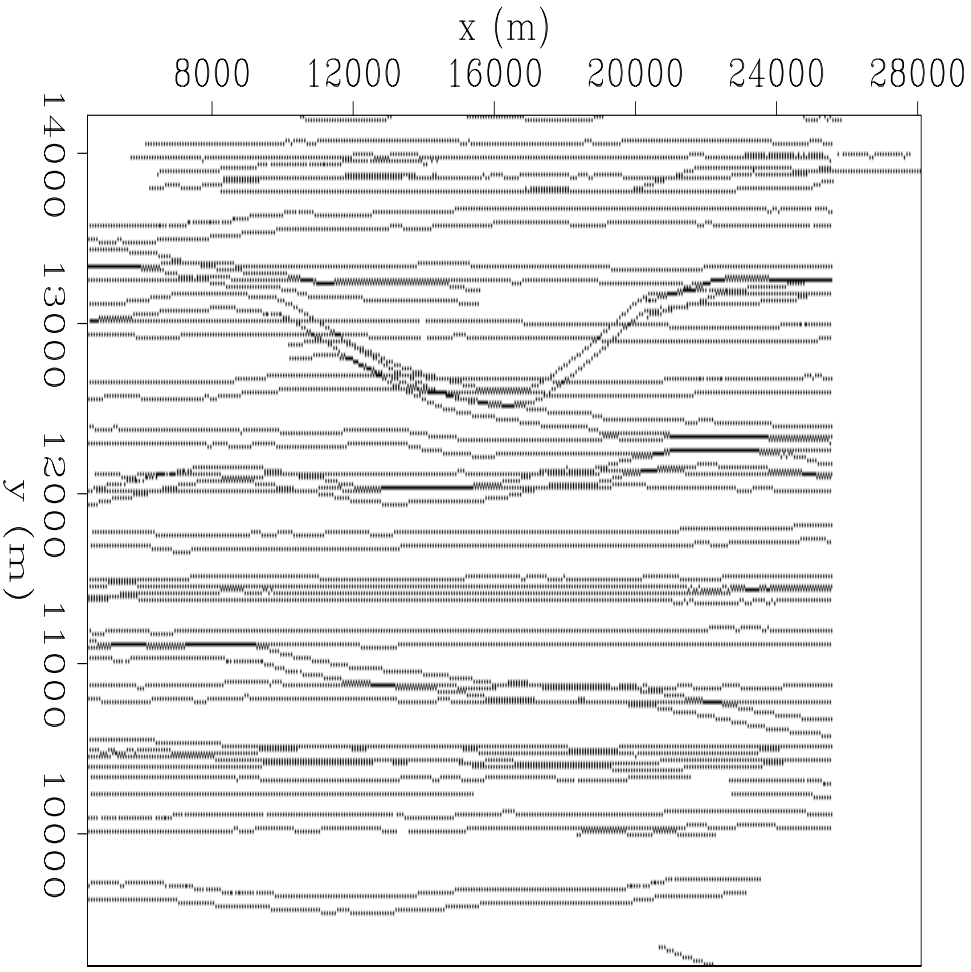

Figure 1.3: Source locations for a 3D marine survey. The ship sailed in approximately straight lines. The parallel lines are due to flip-flop shooting. Sources are much better sampled in the inline (horizontal) than the crossline (vertical) direction. **NR** Intro/. marinegeom1

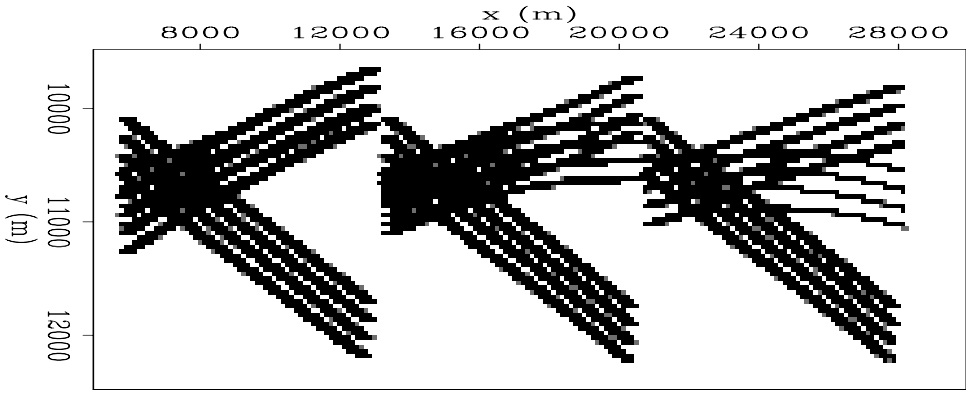

Figure 1.4: Different sets of receiver locations put on the same map for various source positions. The crossline offsets vary considerably from sail line to sail line, even if the sources are nearby. **NR** Intro/. marinegeom2

In addition to the sparse crossline source sampling, the crossline offset axis is also undersampled. Figure 1.4 shows a map view containing the positions of receiver cables for three points along three different sail lines. The three sets of cables on the bottom of the figure are relatively consistent, but the feathering is on the order of several hundred meters. The three overlapping sets of receiver cables at the top of the figure show that even when the source locations are near one another, in this case all near $y = 11500$ m, the cable feathering can be considerably different, as the ocean currents and previous ship movement that determine the cable feathering are functions of space *and* time.

Since the number of channels (both governed by length of cable and number of cables) is limited, a trade-off exists between the acquisition aperture, the total area covered by the recording array, and the sampling within that recording array. Interpolating data allows wider apertures to be used, provided that the interpolation method is able to capture information from the potentially aliased data.

THE NEED FOR INTERPOLATION

Many data-processing algorithms, including wave-equation migration and many multiple-removal methods, require regular, complete, and densely-sampled data as input, but as the previous section illustrates, actual sampling in the field is far from ideal. Surface obstacles and logistical requirements cause irregular sampling of sources and receivers, in both land and marine data. Acquisition design also results in systematic gaps in the data, such as the near-offset gap in marine data. Finite capital results in a trade-off of sampling density and aperture size, for example when adequately dense sampling along all axes is traded for larger sampling areas, resulting in spatially aliased data, particularly in the crossline direction in marine data.

Traditionally, many 2D algorithms had been applied on 3D data, one example of which is surface-related multiple elimination (SRME) (Verschuur et al., 1992). 2D SRME is effective on data from areas where the subsurface has mostly cylindrical symmetry with little variation in the crossline direction. When applied to data with multiples generated from out of the inline plane, the multiple prediction degrades. Two examples of this are shown in Figures 1.5 and 1.7. Figure 1.5a is a close-up of water-bottom multiples in field marine data (courtesy of CGGVeritas) generated from a sea floor with a varying dip in the cross-line direction, while Figure 1.5b is an example of the multiple model generated from a single receiver cable using 2D SRME. The right side of the prediction matches the multiples present in the data reasonably well, excluding the expected amplitude and phase differences. The left side of the prediction, however, is kinematically quite different from the observed multiples in the data. The predicted multiples arrive later than the actual multiples, since the actual multiple arrivals reflect from out-of-plane in the shallower water bottom.

Another example, this time synthetic, of shortcomings in 2D SRME is in Figure 1.7. Figure 1.7a shows synthetic marine data containing a horizontal water bottom and many diffractions and diffracted multiples from out-of-plane. Figure 1.7b shows the 2D multiple prediction from auto-convolution of these data. While the cloud of diffracted multiples is in roughly the correct place, the predicted diffracted multiples

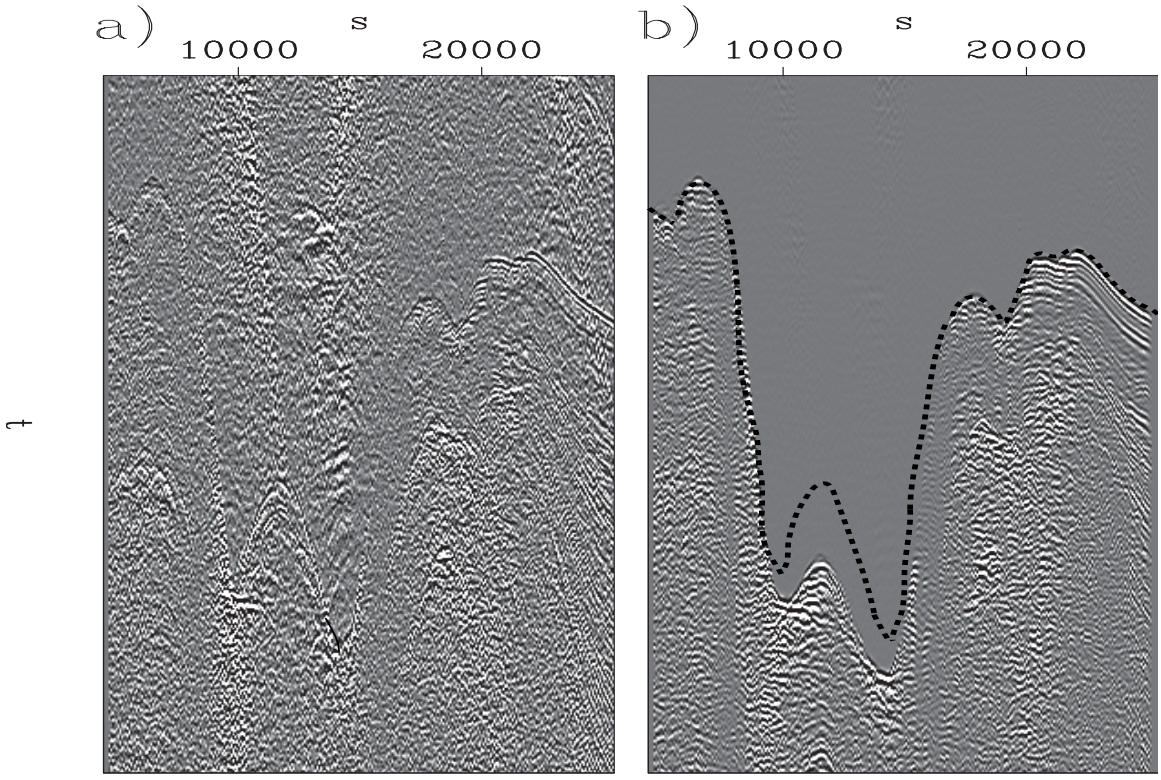

Figure 1.5: The failure of 2D multiple prediction in a 3D world: a) inline constant-offset section of original data; b) 2D multiple prediction for the same region, with the arrival times of the multiples from the actual data superimposed as a dashed line. **NR** Intro/. srmprann

are aliased and the positions of the many diffractions that make up the cloud of multiples are incorrect. The aliasing results from insufficient inline source sampling that is one-third that of the receiver sampling. The mispositioning of the events within the cloud of diffracted multiples is attributable to the 2D nature of the algorithm, in which all events are assumed to originate from within the vertical 2D plane at zero crossline-offset. Implementing 3D SRME on these data requires that a source is present at every receiver location, as described for the 2D case in Figure 1.6would require considerable crossline receiver and source interpolation as well as extrapolation of crossline offsets.

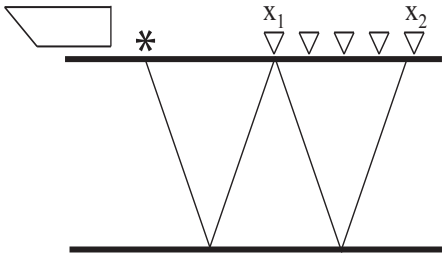

Figure 1.6: Schematic for surface-related multiple prediction. A free-surface multiple recorded at x_2 with a bounce point at x_1. This raypath can be expressed as two primaries, one with the original source and a receiver at x_1, the second with a source at x_1 and a receiver at x_2. The source and receiver sampling should be equal. **NR** Intro/. srmecartoon

EXISTING INTERPOLATION METHODOLOGIES

Interpolation has a long history, with many classic methods such as Gaussian polynomial and sinc-based (Shannon, 1949) interpolation. I place methods developed for reflection seismic data into two main categories: those that use mainly a-priori information about the physics of the recorded data, and those that use mainly statistical information gathered from the data.

The first category contains many methods, some based upon normal move-out, others on more sophisticated imaging operators such as prestack partial migration (Chemingui, 1999; Fomel, 2001; Biondi and Vlad, 2002; Clapp, 2003; Baumstein,

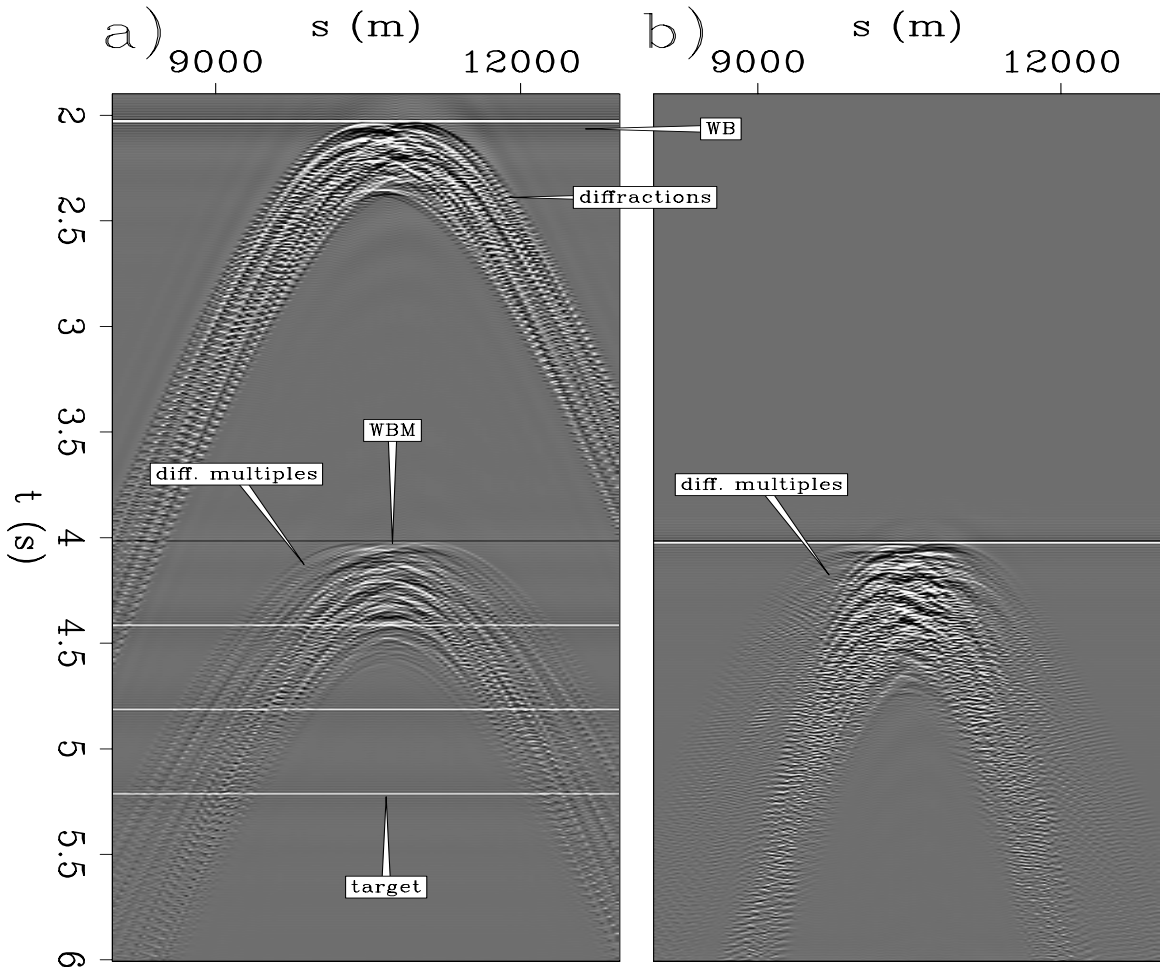

Figure 1.7: Synthetic marine data: a) original data; b) multiple prediction. The diffracted multiples created from out-of-plane in (a) are not accurately predicted by a 2D SRME prediction shown in (b), although the water-bottom multiple is accurately predicted. **CR** Intro/. srmpsann

2004), and others of full migration and demigration of data (Trad, 2002) or use of a surrogate for migration operators (Verschuur and Berkhout, 2005). These methods have several benefits, the first of which is the ability to extrapolate as well as interpolate data. Another benefit is that many of these methods can deal with irregular sampling of data, but the main drawback with these methods is that the events in the recorded data do not necessarily conform to the physics of the imaging operator. This can be because of an incorrect assumed velocity, the presence of energy (such as ground roll) that is not predictable by the imaging method, multiple reflections, or coherent noise.

Statistical methods use only information contained in the data to interpolate missing data. Statistical methods include a myriad of transform-based methods, such as Fourier, Radon, wavelet (Wang and Li, 1994), or curvelet (Thomson and Herrmann, 2006) transforms. Fourier transform-based methods exist in many forms (Gulunay and Chambers, 1997). Examples include non-uniform (Schonewille and Duijndam, 1998), minimum reweighted norm (Liu and Sacchi, 2004), Projection onto Convex Set (POCS) (Abma and Kabir, 2005), sparse (or high-resolution) methods (Zwartjes and Gisolf, 2007), and antileakage methods (Xu et al., 2005). Radon-based methods are also commonly used, often in the form of a high-resolution radon-based approach (Sacchi and Ulrych, 1995), and more recently a shifted-apex radon transform (Hargreaves and Trad, 2005). Most of these methods can interpolate irregularly-sampled data, but are limited in that they typically do not faithfully interpolate aliased data.

Another group of non-physically-based statistical methods are filter-based methods. These filters include dip filters and prediction or prediction-error filters. With dip filters, a single dip is estimated within a window of data, from which a filter is created that is then used to interpolate the unknown data in that dip direction (Fomel, 2002). While this method can cope with irregularly-sampled and spatially-aliased data, extension to treating multiple interfering dips is not straightforward and is considerably more complicated. Unlike a single local-dip filter, prediction (Spitz, 1991) or prediction-error (Claerbout, 1992) filters can estimate data in the presence

of multiple simultaneous slopes. Since these filters are larger (i.e., they have many more coefficients) than do two-column dip filters, their estimation requires regularly-sampled training data, typically estimated from a lower-frequency sampling of existing data. More recently, nonstationary prediction-error filters have been used to interpolate data in the time-space (t-x) domain (Crawley, 2000), instead of the previous patch-based approach (Claerbout, 1992).

MOTIVATION AND CONTRIBUTIONS

In this thesis, I create new nonstationary prediction-error filter (PEF)-based interpolation methods applicable to multiple applications, including problems of irregular data, large systemic gaps in acquisition, and higher-dimensional interpolation to the extent necessary for 3D SRME. I do this by adapting the choice of data used to estimate the PEF to each particular problem.

Methods to interpolate irregular data are typically limited to unaliased data or to a single dip, while prediction-error filters can capture multiple aliased slopes, but require regularly-sampled data. My contribution is a method to estimate a nonstationary PEF from irregularly-sampled data, using multiple regridded copies of the data as training data (Curry, 2003). While the more coarsely-gridded copies of the data are not exact, they produce a superior result than when using no information from the data or from using a single regridded copy of the data. This method succeeds in more accurately interpolating multiple simultaneous changing slopes that are irregularly sampled than do previous approaches.

Marine data contain a near-offset gap between the air gun source and the nearest towed receiver, but this missing near-offset information is essential for use in SRME methods. Most methods to recover these near offsets rely on the curvature of the primaries in nearby data, either in a Radon-transform-based approach or in a more straightforward NMO-based prediction wherein the stacking velocity is derived from the nearby recorded primaries. Similarly, a conventional approach using prediction-error filters would rely on using the nearby primaries as training data. In my approach,

I generate training data for a nonstationary PEF by cross-correlating traces within each shot to generate a *pseudoprimary* dataset (Curry and Shan, 2006). Instead of simply substituting these data into the near-offsets or manually matching them to the recorded data using deconvolution or matching filters, I use these as training data for a nonstationary PEF. I estimate this PEF both in time, offset, and source position, as well as in frequency, offset, and source position, and compare the results. Estimating nonstationary PEFs in the frequency domain provides a much faster, more parallelizable result.

3D SRME requires dense sampling of both receivers and sources in the inline and crossline directions. Currently, most algorithms for generating these data are based upon normal moveout (Levin, 2002), dip moveout (Baumstein and Hadidi, 2006), azimuth moveout (Matson and Abma, 2005), or a migration/demigration approach (Weisser and Taylor, 2006). While SRME is a purely data-driven approach with no required velocity model, these methods still require some sort of velocity estimate. I generate data densities suitable for 3D SRME by interpolation using nonstationary PEFs. Instead of estimating the filter in time and space, I estimate it in frequency and space by applying the Spitz (1991) approximation that links data at coarse sampling and low frequencies to data at finer sampling and higher frequencies, but instead of the patch-based approach of Spitz I estimate a single nonstationary PEF for each frequency. This new approach of nonstationary interpolation of frequency slices is much faster than both the nonstationary t-x based approach and the previous patch-based approach of Spitz in f-x. I apply this approach to interpolate both inline sources and crossline receivers in two, three, and four dimensions, and compare the results for the different choices of dimensionality and domain of the PEFs. I also iteratively interpolate the data to generate data with increases in density by factors of four and six in multiple dimensions, and show how the data degrades as the iterative interpolation proceeds.

In this thesis, I focus mainly on examining and comparing the interpolated data. In practice, these data are used in subsequent processing stages to produce a final migrated image and ultimately, a geologic interpretation. For the land data example, I

show a subsequent processing step used in velocity estimation, while in the subsequent marine data examples, the subsequent step is multiple prediction.

THESIS OVERVIEW

Chapter 2: PEFs and Interpolation

This chapter is a review of the background theory for this thesis. I review estimation of a prediction-error filter on fully-sampled training data, and interpolation of data with an estimated prediction-error filter. I show the extent to which use of inexact training data can degrade the interpolation result. I then review the estimation of nonstationary prediction-error filters, how to interpolate data using them, and demonstrate the interpolation with perfectly-known training data on a 3D synthetic model, comparing it to a patch-based approach.

Chapter 3: Interpolation of irregularly-sampled data

The previous chapter assumes that the training data for the PEF are completely known. I review how to exclude unknown data from the PEF estimation by weighting equations containing unknown data to zero, and then propose a new method of estimating a PEF when the data are irregularly-sampled and PEF estimation was previously impossible, with all equations weighted to zero. I use multiple copies of the data placed on grids that vary in both size and origin as training data, estimate both stationary and non-stationary PEFs on synthetic examples, and then interpolate to create a regular source-offset cube from an irregularly-sampled prestack 2D land data set from Colombia.

Chapter 4: Interpolation of near offsets with multiples

Here, I fill in the systemic near-offset gap in marine data by using information contained in free-surface multiple reflections. I cross-correlate traces within each shot

using the known principle that primaries correlated with free-surface multiples create pseudo-primaries, with one receiver acting as a virtual source. With the near-offsets of these pseudo-primaries as training data for non-stationary PEFs, I use the estimated PEFs to interpolate traces in the large near-offset gap. I test this method on the Sigsbee2B dataset and a large near-offset gap, using nonstationary filters in both time and, for the first time, in frequency and compare the results. I then apply this same approach to field data with a smaller gap. Finally, using a 3D synthetic model, I investigate the issues with using this approach in 3D, both in terms of crossline variability of 3D field data and the severely limited source distribution in the crossline direction.

Chapter 5: Nonstationary f-x interpolation of prestack 3D data

One common application of interpolation and regularization methods is for 3D surface-related multiple elimination, in which the data need to be regularly and fully sampled so that there is a source at every receiver location in both the inline and crossline directions. Data need to be created in the inline source direction as well as both the crossline receiver and crossline source directions.

After first reviewing Spitz's observation that the wavenumber spectra of a plane wave in the f-x domain at a single frequency and spatial sampling with that of a higher frequency and finer spatial sampling are linked, I then use coarser-sampled lower frequencies as training data for a complex-valued nonstationary PEF that I use to interpolate a higher frequency to a finer sampling. I demonstrate this method on multidimensional plane waves as well as a 3D synthetic model.

I apply the method to interpolate 3D prestack synthetic data containing many diffracted multiples. I investigate the different domains on which these filters can be estimated, as well as the dimensionality of the filter for interpolating both inline sources and crossline receivers. For iteratively interpolating data to the extent required for 3D SRME, I continue this investigation for a 3D field data example with

only four cables, pushing the nonlinear approach of interpolation previously interpolated data to its breaking point.

Chapter 6: Conclusions

I summarize the different approaches to generating training data for a PEF used to interpolate data, and highlight some future avenues of research. This non-stationary PEF-based approach is very generalizable, and can be tailored to other applications by varying the training data and domain used, and the tools present in inverse theory.

Chapter 2

Prediction-error filters and interpolation

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

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

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

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

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

PREDICTION-ERROR FILTER ESTIMATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We then move the second term to one side to obtain

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

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

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

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

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

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

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

MULTI-DIMENSIONAL FILTERING IN THE HELICAL COORDINATE

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

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

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

and perform a 2D autocorrelation to produce

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

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

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

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

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

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

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

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

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

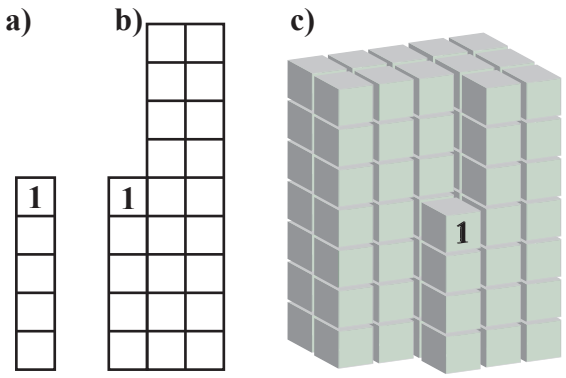

Figure 2.1: Examples of the structure of 1D, 2D, and 3D PEFs: a) five term 1D PEF; b) 23-term 2D PEF; c) 88-term 3D PEF. **NR** PEF/. helix

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

PEF ESTIMATION ON SYNTHETIC DATA

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

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

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

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

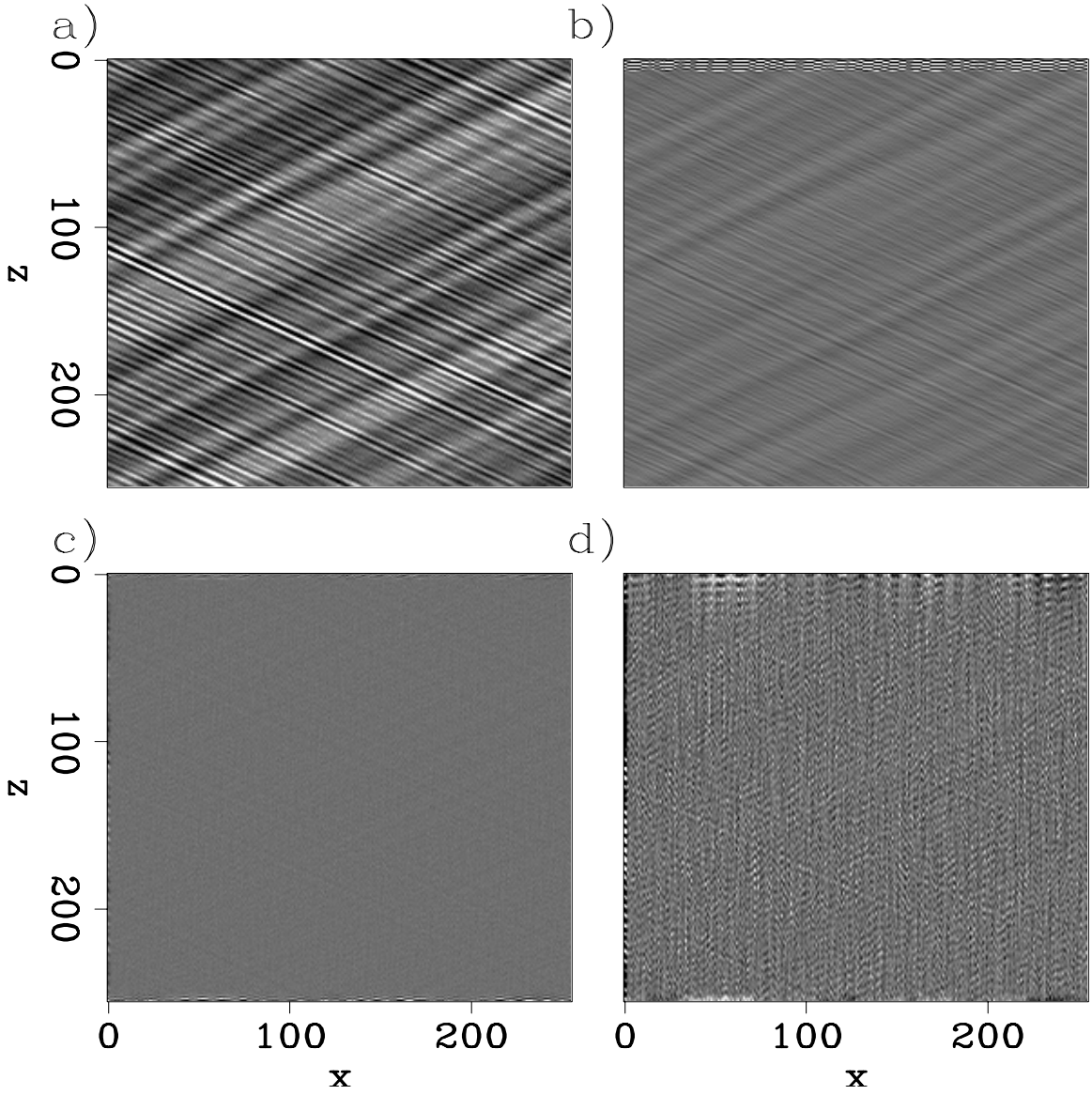

Figure 2.2: Estimation of a 2D PEF on 2D synthetic data with two dips. (a): original input training data; (b): residual ($\mathbf{r}_\mathrm{d}$) after PEF estimation using a single 10×1 vertical filter; (c): residual ($\mathbf{r}_\mathrm{d}$) for a 10×3 filter; (d): residual in (c), divided by the 1D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

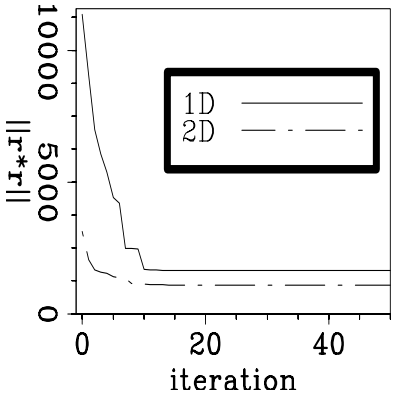

Figure 2.3: Residual power as a function of conjugate direction iteration for a 10-element 1D PEF and a 24-element 2D PEF. The 2D PEF converges more quickly and has a lower residual. **ER** PEF/. planeestcv

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

INTERPOLATION WITH A PREDICTION-ERROR FILTER

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

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

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

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

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

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

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

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

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

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

equation 2.16, giving

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

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

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

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

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

Interpolation example on synthetic data

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

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

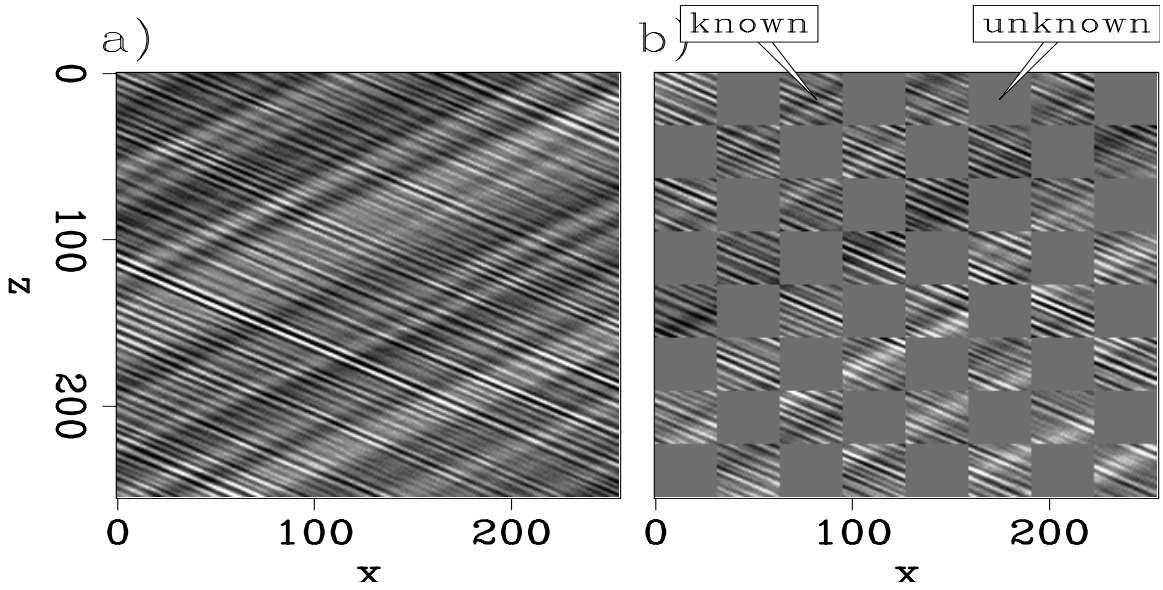

Figure 2.4: Synthetic 2D data. (a): original data, identical to that in Figure 2.2a. (b): the original data with 50 percent of the values missing in a checkerboard pattern of 32×32 cells. **ER** PEF/. dataholeann

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

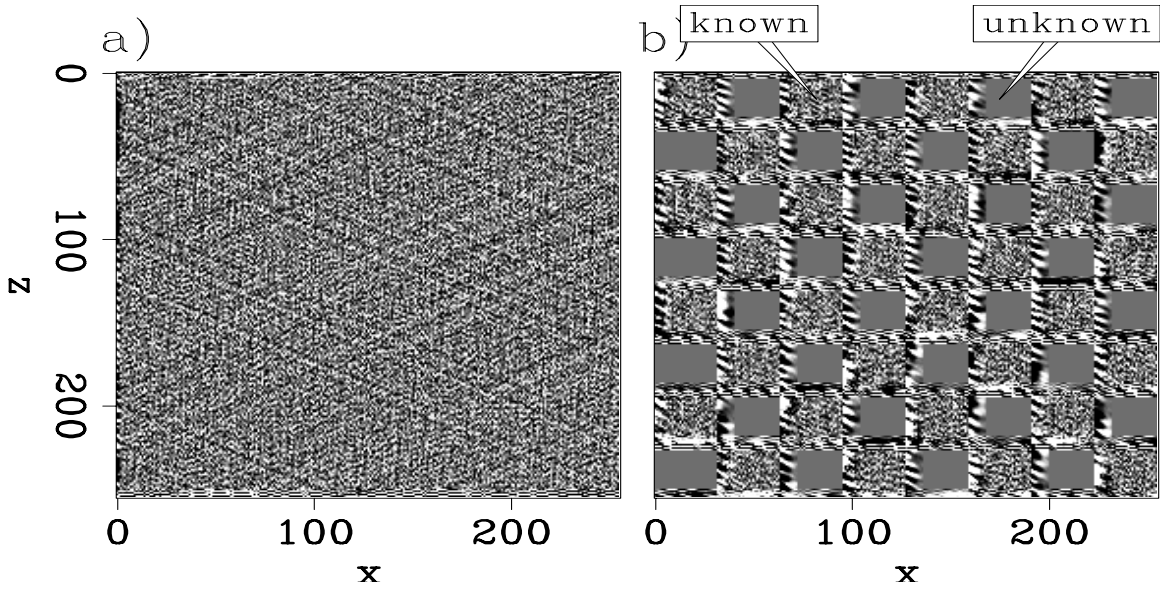

Figure 2.5: Synthetic 2D data shown in Figure 2.4 after filtering with a 10×10 PEF. (a): filtered original data. (b): filtered sampled data, $\mathbf{r}_0$ in equation 2.18. The amplitudes are magnified by a factor of 20 compared to those in Figure 2.4. The boundaries between sampled and unknown data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

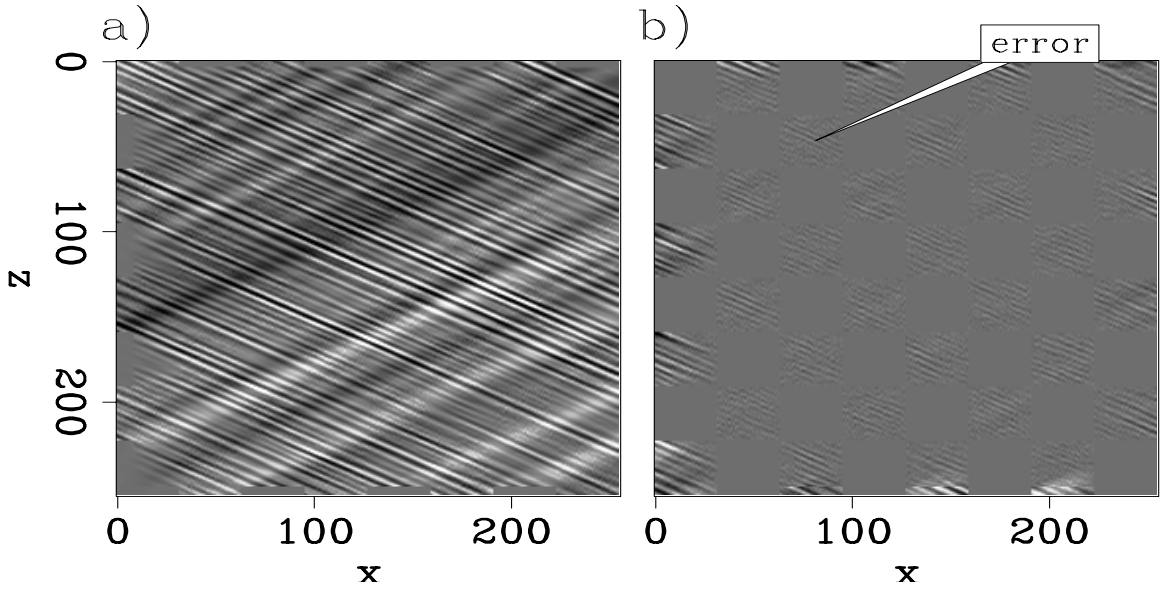

Figure 2.6: 2D Synthetic data in Figure 2.4 interpolated using a 2D PEF. (a): interpolated result. (b): difference between interpolated and original fully-sampled data. The interpolation captures the slopes of the training data, with the amplitudes under-predicted at the edges of the data. **ER** PEF/.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

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

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

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

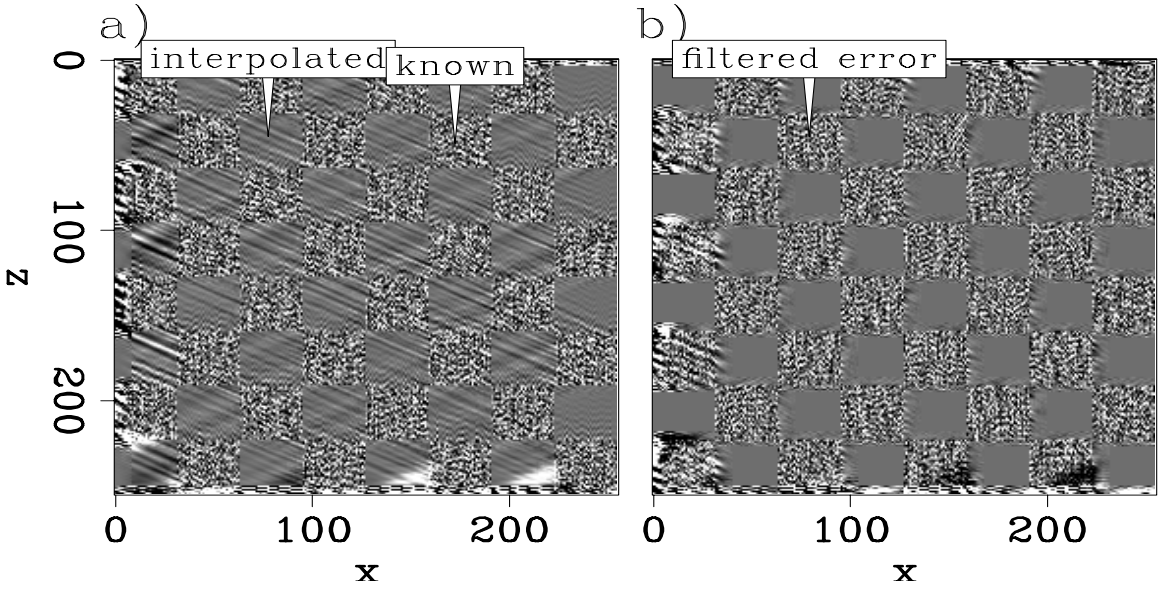

Figure 2.7: Interpolation and interpolation error, the difference between the interpolated data and the originally fully-sampled data, filtered with the PEF. (a): filtered interpolation result. (b): filtered difference between interpolation and original fully-sampled data. The random noise present in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

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

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

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

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

Interpolation with imperfect training data

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

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

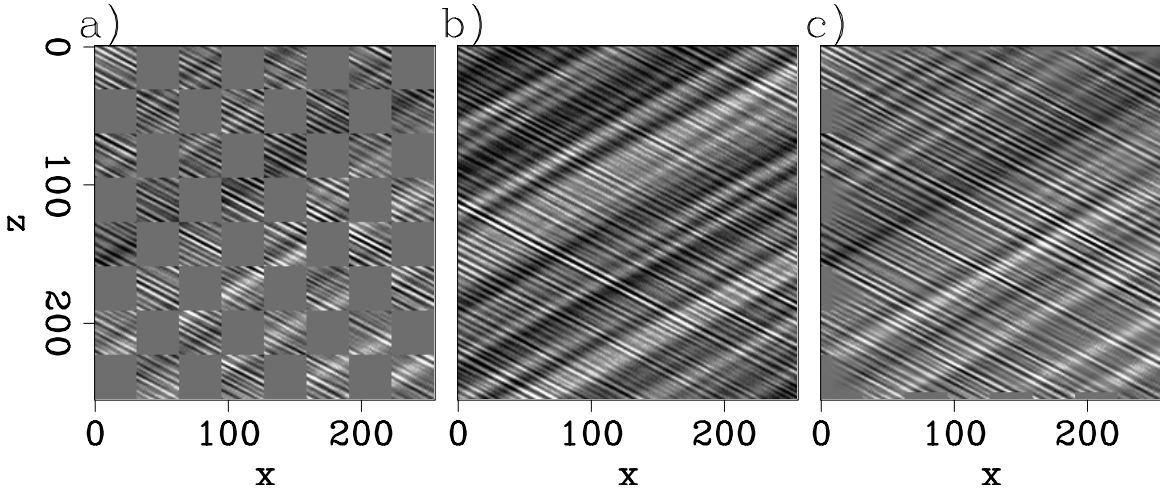

Figure 2.8: Interpolation with training data with different amplitude scale and phase. (a): sampled data. (b): training data with different phase and 100 times the amplitude of the ideal training data set in Figure 2.4a. (c): interpolated result. The interpolation is almost identical to that in Figure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

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

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

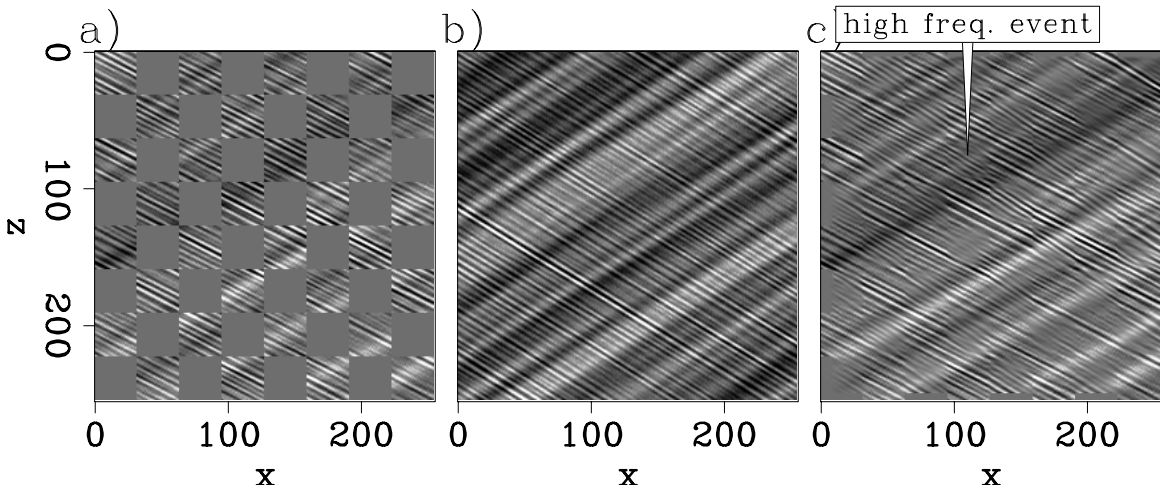

Figure 2.9: Interpolation with training data with different amplitude scale, phase, and slopes. (a): sampled data. (b): training data with slopes 15 degrees less than and greater than the original slopes. (c): interpolated result. The interpolation of the low-frequency planar event is reasonably good,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

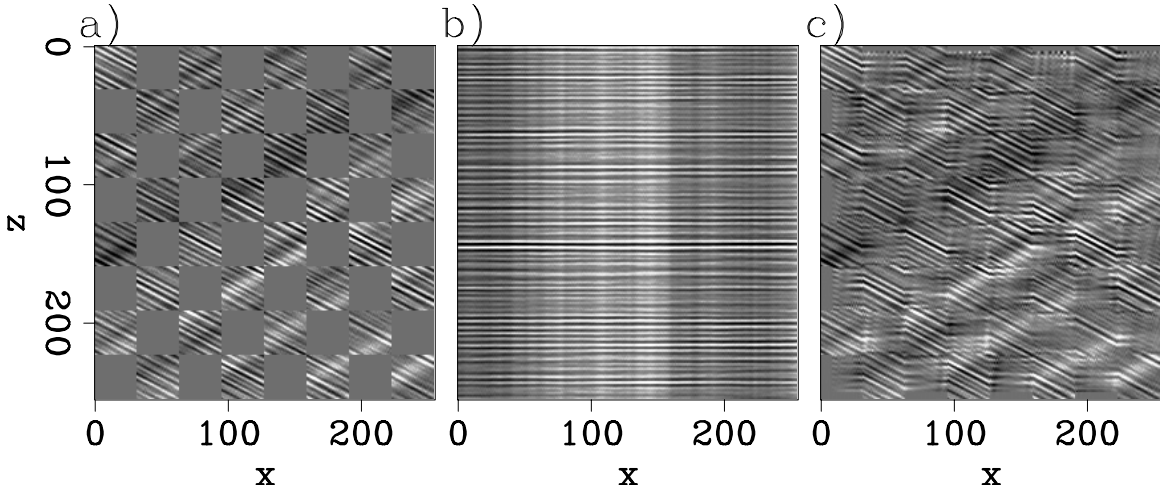

Figure 2.10: Interpolation with training data with slopes that are wildly different from those in the ideal training data. (a): sampled data. (b): training data with vertical and horizontal slopes. (c): interpolated result. The interpolation is hopelessly incorrect. **ER** PEF/. statbadpeffill

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

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

NONSTATIONARY PREDICTION-ERROR FILTERS

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

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

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

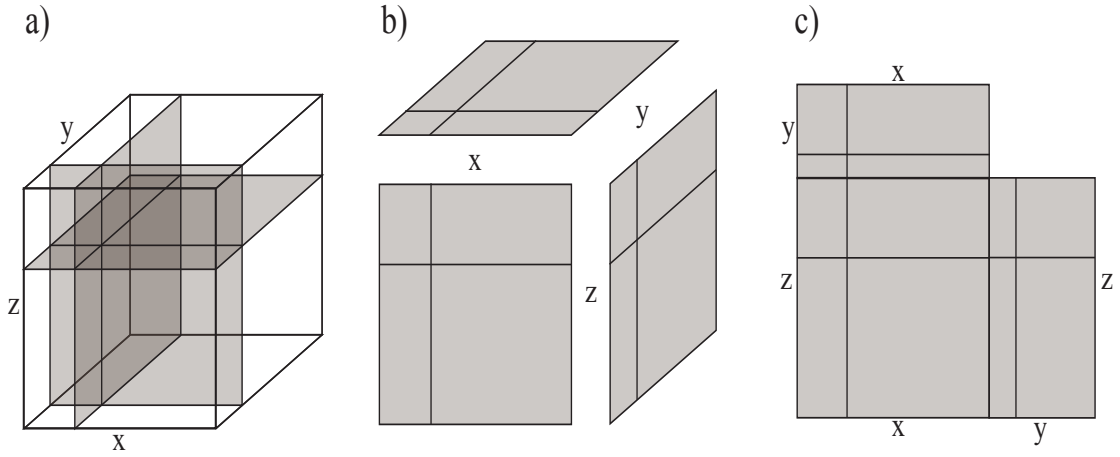

Figure 2.11: A schematic of the 'cubeplot' figures used throughout this thesis. Three input planes from within a cube (a) are separated (b) and placed side-by-side, where the lines on each image denote the intersections of the other slices through that image.
NR PEF/. cubeplot

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

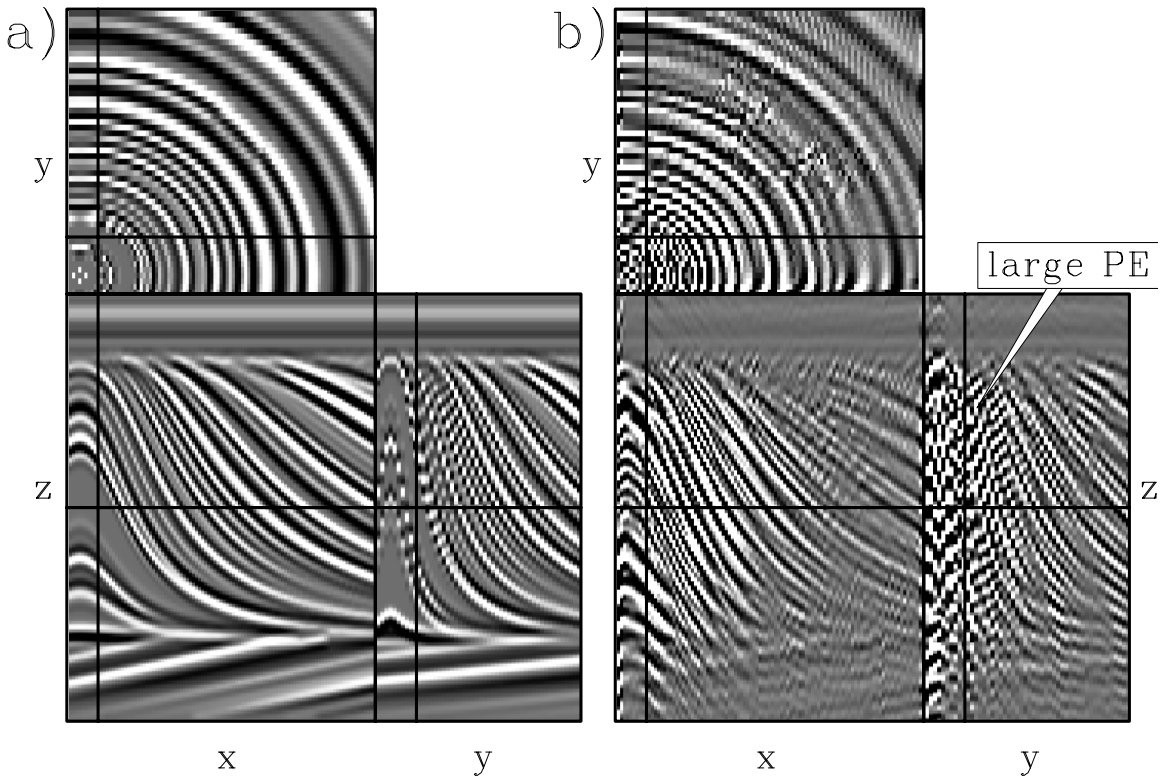

Figure 2.12: Estimating a PEF on the three-dimensional quarter-dome data. (a): original fully-sampled data. (b): filtered residual (i.e. prediction-error) from convolving a PEF with the original data, then dividing by a 1D PEF estimated on (a). Only the stationary upper and lower energy is predicted.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

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

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

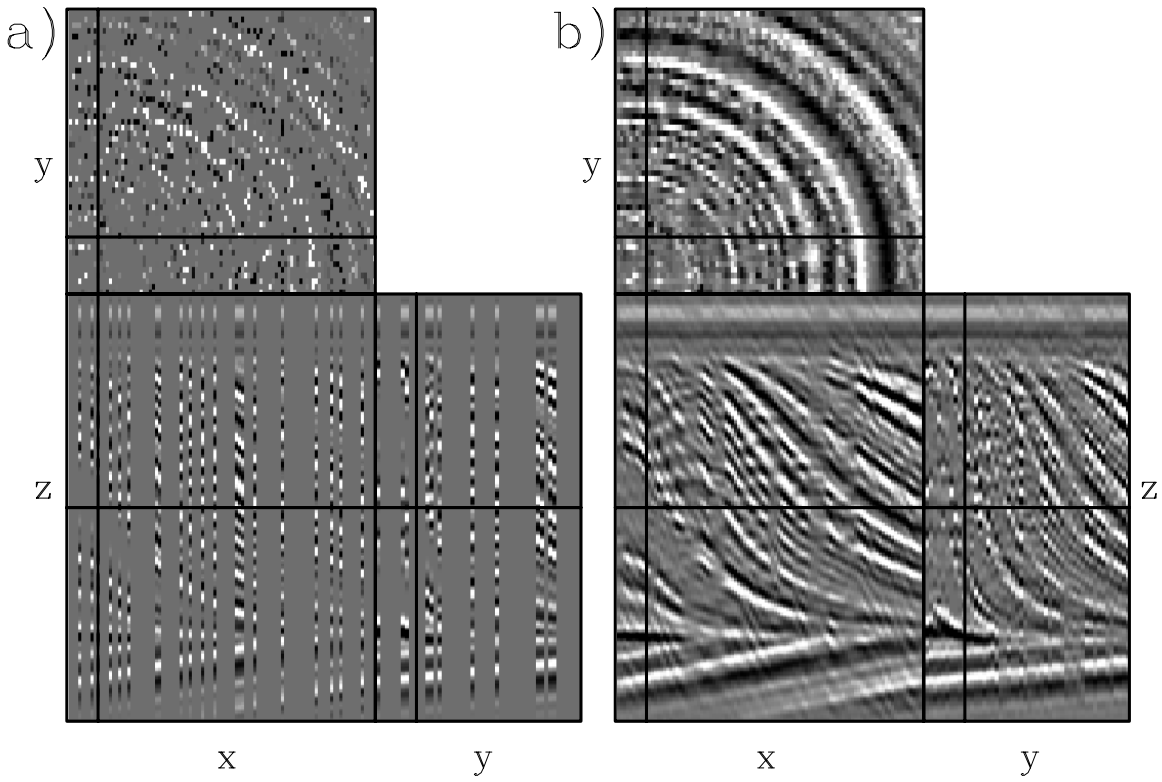

Figure 2.13: Interpolating the quarter-dome data from Figure 2.12 with a 3D PEF. (a): randomly-sampled traces with only 20 percent of data present. (b): data from (a) interpolated with a 3D PEF estimated on the fully-sampled data. **ER** PEF/. qdomestatfillann

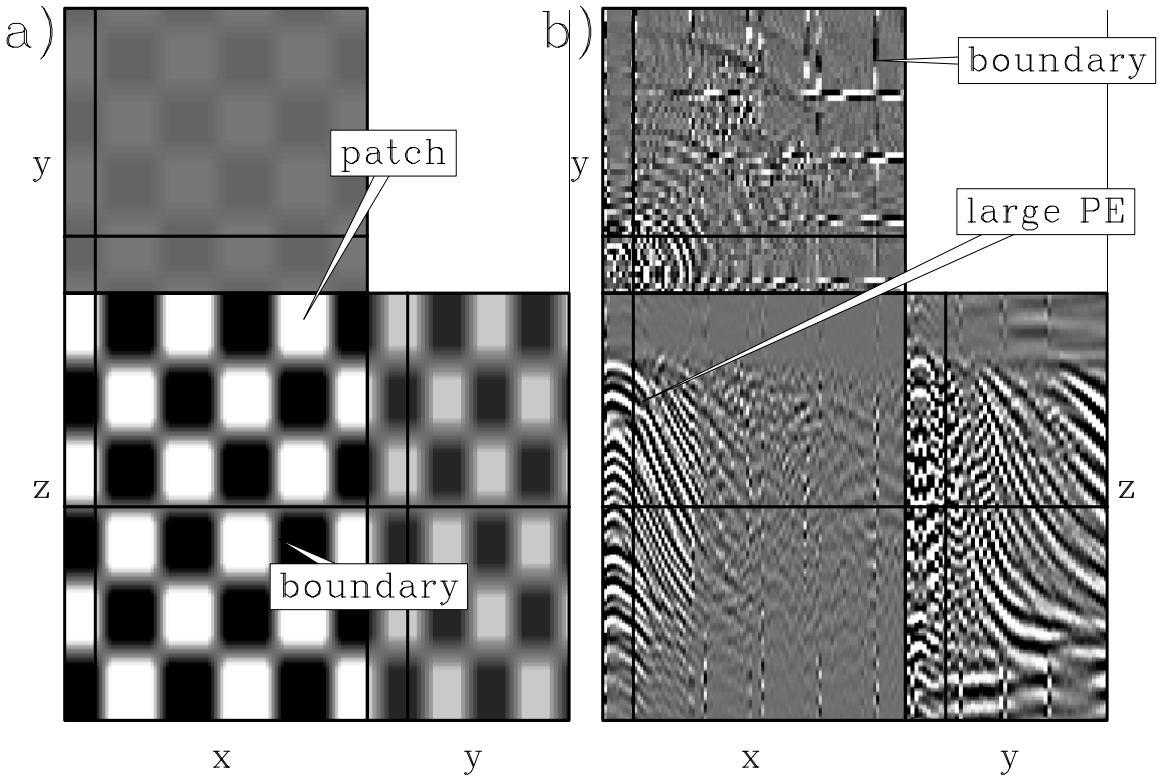

Figure 2.14: Multiple PEFs estimated on the quarter-dome synthetic in patches. (a): a representation of the size of the patches used, depicted by alternating black and white patches and gray representing overlap between patches. (b): patched filtered residual of PEF estimation for the 216 different patches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

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

Nonstationary PEF estimation

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

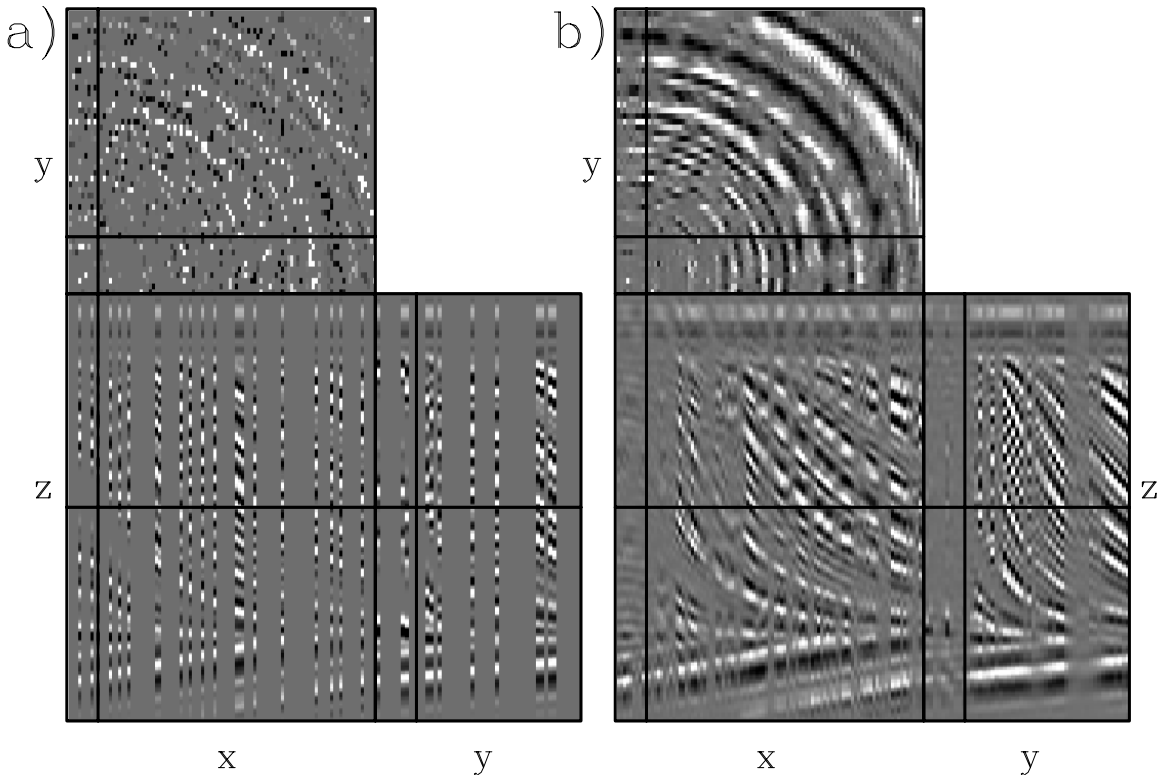

Figure 2.15: The quarter-dome synthetic interpolated in patches. The three faces correspond to three slices through the 3D cube. (a): input sampled data. (b): the interpolated result. The result is still poor due to the rapidly changing dips combined with the sparse sampling of the data, which increased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

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

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

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

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

so that

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

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

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

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

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

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

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

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

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

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

Nonstationary prediction-error filter interpolation: 3D example

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

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

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

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

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

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

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

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

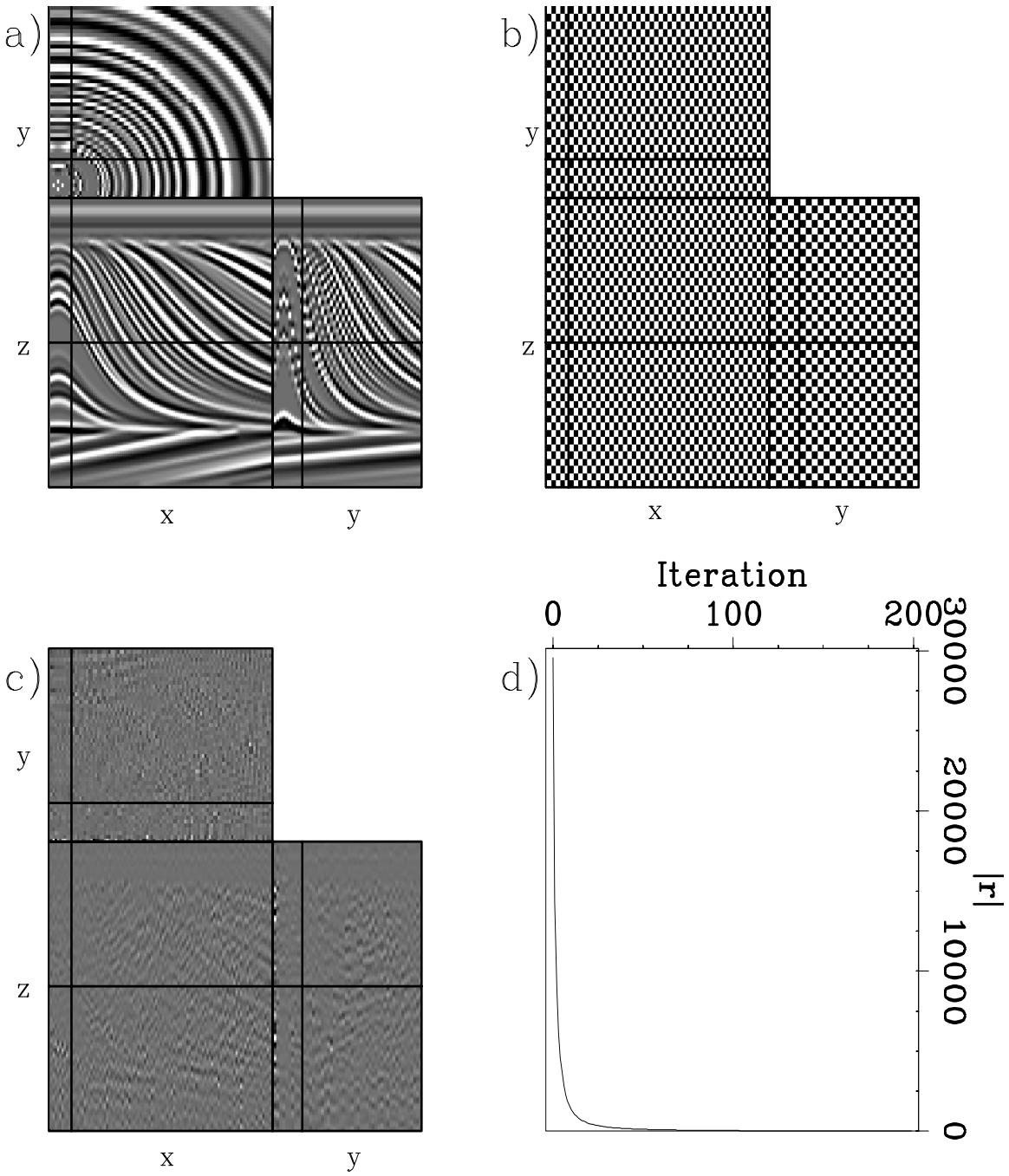

Figure 2.16: Estimation of a nonstationary PEF on the quarter-dome synthetic. (a): original data. (b): diagram of non-stationarity of filter coefficients. (c): data residual of PEF estimation after division by a 1D PEF estimated on (a) To emphasize differences. (d): convergence of PEF estimation. **ER**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

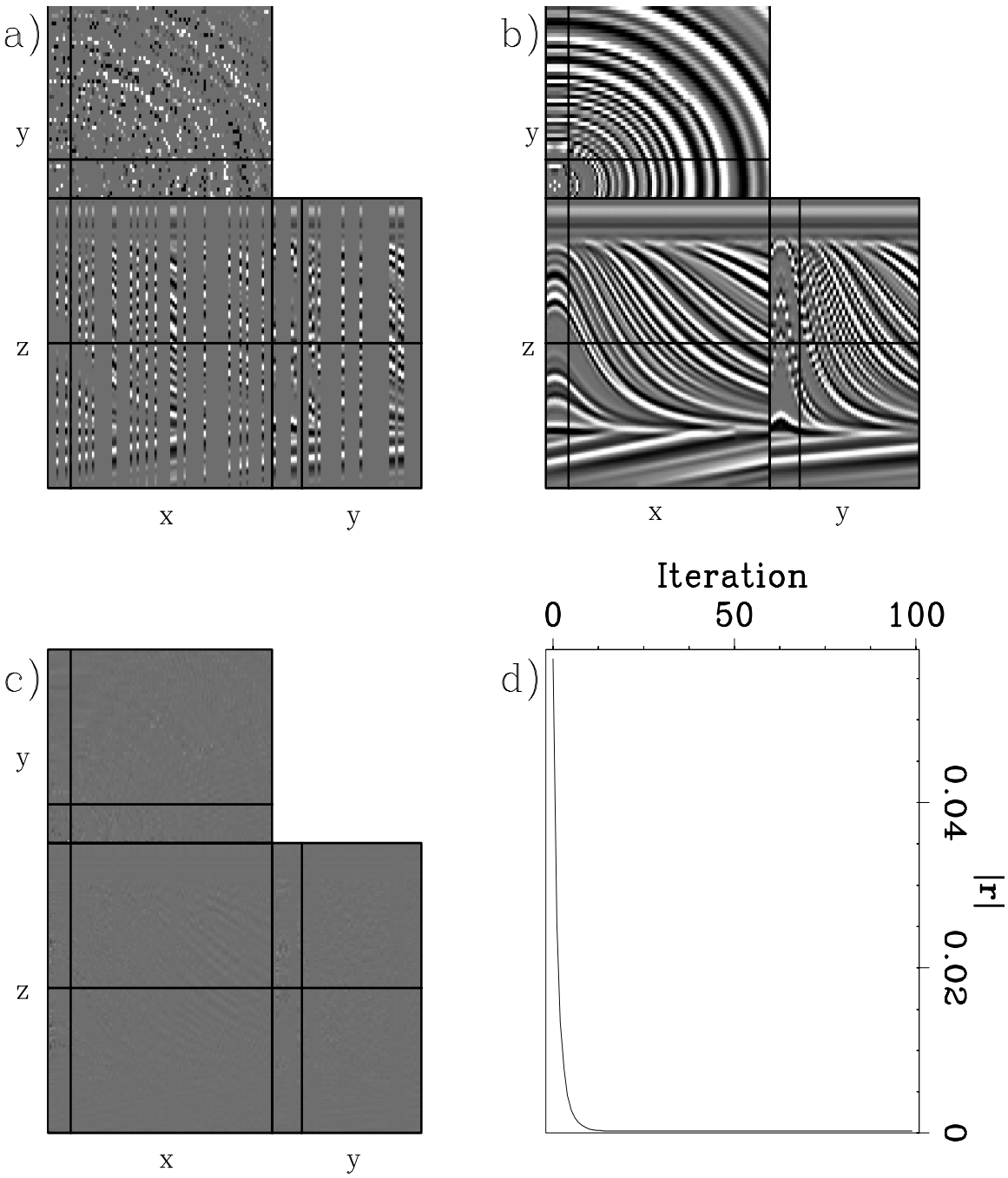

Figure 2.17: Interpolation of the quarter-dome with a nonstationary PEF. (a): Sampled quarter-dome data. (b): Interpolated result. (c): difference between interpolated result and original data. (d): convergence of interpolation of missing data. **ER** PEF/. nspeffillann

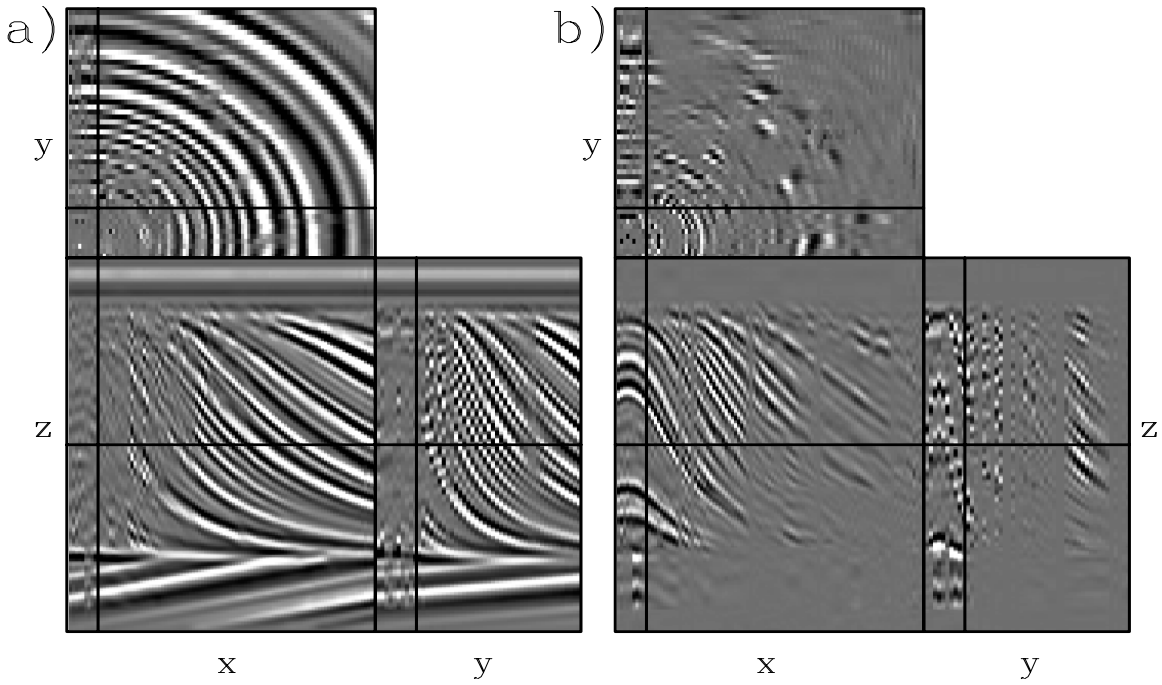

Figure 2.18: Interpolation of the quarter-dome with a nonstationary PEF with the same number of coefficients as the patched case. (a): interpolated data. (b): difference between interpolated result and original data. The interpolation is much better than the patched case but is worse than the smoothly-nonstationary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

Chapter 3

Interpolation of irregularly sampled data

Most modern seismic acquisition methods aim to sample data regularly along all axes. Deviations from this sampling happen for various reasons. On land and ocean-bottom cable data, obstacles and terrain cause both sources and receivers to be moved. An example of the distribution of sources in an ocean-bottom cable (OBC) survey is shown on Figure 3.1a. The source positions are not evenly distributed because of the source ships both firing irregularly and not sailing in straight lines. There is also a large gap in the center of the survey because of an ocean platform. The ocean-bottom receiver cable positions in Figure 3.1b show that the receiver cables are not straight and are unevenly spaced in the crossline. Some acquisition designs intentionally acquire data with irregular sampling. One example of this is an ocean-bottom node array, where the nodes may be randomly placed on the sea floor. Land data can be randomly sampled to reduce acquisition footprint (Zhou and Schuster, 1995).

While data are often irregularly sampled, many of the algorithms that we wish to apply to seismic data, such as a spatial Fourier transform or a finite-difference migration, are best-suited for data on a regular grid. Interpolation methods for irregular data typically are based upon either move-out or partial prestack migration

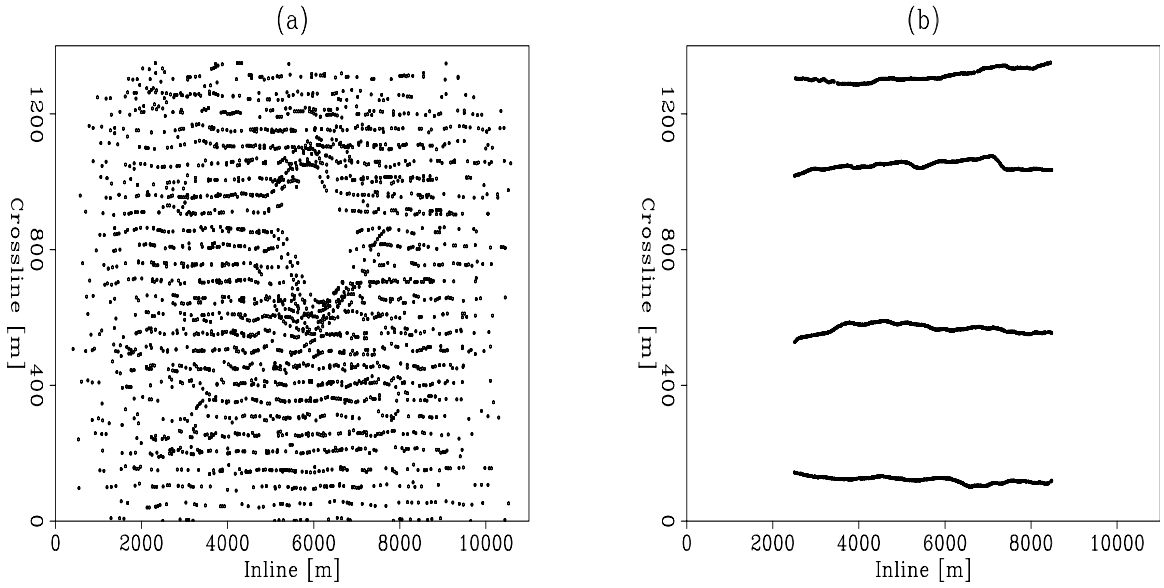

Figure 3.1: An example of source positions (left) and receiver positions (right) from an OBC survey. Figure courtesy of Daniel Rosales. **NR** MSPEF/. acquisition

based methods (Chemingui, 1999; Fomel, 2001; Clapp, 2003) that require a root-mean-square velocity model, or Fourier or Radon transform-based methods (Gulunay and Chambers, 1996; Schonewille and Duijndam, 1998; Liu and Sacchi, 2004; Xu et al., 2005), that typically require unaliased data. Aliased data may be interpolated in f-x (Spitz, 1991), but this method requires regularly-sampled data. As described in Chapter 2, a prediction-error filter can be used to interpolate data on a regular grid with missing samples. This PEF can interpolate many simultaneous slopes that can be aliased, which is often the case with irregularly sampled data. However, a prediction-error filter is also dependent upon regularly-sampled training data. In practice, we use nearest-neighbor interpolation to place the irregular sampled data on to a regular grid with both known and unknown values. When estimating a PEF, the rows of the autocorrelation matrix that depend on missing samples in the training data are weighted to zero. With many samples missing from the data, we encounter a problem of too many rows being weighted to zero, leaving insufficient information to estimate a PEF.

Previously, in order to add more information to the PEF estimation, the prediction-error filter was spaced out over the data, so that the values of the filter sample every second point of the data along each axis (Crawley, 2000; Claerbout, 2004). This spacing of the filter can be changed to other integer values, and a single multi-scale PEF can be estimated from one copy of the input data. This method requires somewhat regular sampling so that the PEF falls upon entirely known data at some filter spacing. Also, care must be taken to ensure that the filter spacing in time is not too coarse to alias the data. Alternatively, instead of changing the spacing of the filter on the original data, I generate training data with fewer holes by regridding the sparse data onto a coarser grid so that more values are nonzero (known), so fewer rows of the autocorrelation matrix will be zeroed (Curry and Brown, 2001). Since the sampling varies with position, different areas are best captured by different grids, both in cell size and position. In order to use as much of the data as possible, I estimate a single PEF from multiple regridded copies of the data, and then use this PEF to interpolate the missing data. I test this on a random sampling of traces from the two-slope example introduced in Chapter 2.

This multi-grid approach to training data can also be used for nonstationary PEF estimation (Curry, 2002). Multiple grids are even more useful here as many more unknown coefficients must be estimated than in the stationary case. Also, because different areas have different sampling and contain unique information because of this nonstationarity, multiple grids are needed to extract this information. To test this approach, I randomly sample the quarter-dome synthetic data from the previous chapter, and estimate a nonstationary PEF from multi-gridded data, with more regridded copies of data producing better results. I then perform this multi-grid estimation on 2D land data in the source-offset domain to produce a PEF that fills most gaps and improves velocity supergathers.

PREDICTION-ERROR FILTER ESTIMATION WITH MISSING DATA

In Chapter 2, I estimate a multi-dimensional prediction-error filter, $\mathbf{f}$, of length n_{f} (including the leading 1) using regularly-sampled training data, $\mathbf{d}$, of length n_{d} by solving a least-squares problem,

$$\mathbf{Kf} = -(\mathbf{D}^{\dagger}\mathbf{D})^{-\mathbf{1}}\mathbf{D}^{\dagger}\mathbf{d}, \tag{3.1}$$

where the unknown filter coefficients, $\mathbf{Kf}$, are estimated from the training data, a convolutional matrix, $\mathbf{D}$, and its adjoint , $\mathbf{D}^{\dagger}$, constructed from elements of $\mathbf{d}$. When data values are unknown, we premultiply $\mathbf{D}$ with a $n_{\mathrm{d}} \times n_{\mathrm{d}}$ diagonal matrix, $\mathbf{W}$, that multiplies rows with unknown data by zero and all others by one. Replacing $\mathbf{D}$ in equation 3.1 with $\mathbf{WD}$ and recognizing that $\mathbf{W}$ is a symmetric, idempotent matrix ($\mathbf{W}^2 = \mathbf{W}^{\dagger} = \mathbf{W}$), equation 3.1 becomes

$$\mathbf{Kf} = -(\mathbf{D}^{\dagger}\mathbf{WD})^{-\mathbf{1}}\mathbf{D}^{\dagger}\mathbf{Wd}. \tag{3.2}$$

For example, if a nine-point data vector ($n_{\mathrm{d}} = 9$) with a missing third data point, ($d_3 =?$) were used to estimate a four-component PEF ($n_{\mathrm{f}} = 4$), matrices $\mathbf{WD}$ would be written as

$$\mathbf{WD} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 \\ d_2 & d_1 & 0 & 0 \\ ? & d_2 & d_1 & 0 \\ d_4 & ? & d_2 & d_1 \\ d_5 & d_4 & ? & d_2 \\ d_6 & d_5 & d_4 & ? \\ d_7 & d_6 & d_5 & d_4 \\ d_8 & d_7 & d_6 & d_5 \\ d_9 & d_8 & d_7 & d_6 \end{bmatrix}. \tag{3.3}$$

In addition to weighting equations with missing data to zero, we can also weight equations where the filter rolls off the known data to zero, so that for the same example

$$\mathbf{WD} = \left[\begin{array}{ccccccccc} 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ \hline 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ \hline 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \end{array}\right] \left[\begin{array}{cccc} d_1 & ? & ? & ? \\ d_2 & d_1 & ? & ? \\ ? & d_2 & d_1 & ? \\ \hline d_4 & ? & d_2 & d_1 \\ d_5 & d_4 & ? & d_2 \\ d_6 & d_5 & d_4 & ? \\ \hline d_7 & d_6 & d_5 & d_4 \\ d_8 & d_7 & d_6 & d_5 \\ d_9 & d_8 & d_7 & d_6 \end{array}\right]. \tag{3.4}$$

Introducing a single unknown point in the training data in equation 3.3 causes n_{p} rows of the **D** matrix to be ignored, as the unknown point is multiplied by each filter coefficient once. The **W** matrix in equation 3.4 also includes a mute for areas where the filter rolls off of the data boundary. An example of this in two dimensions with one unknown data value and a 7×3 2D PEF is shown in Figure 3.2.

Because each unknown training data value greatly decreases the number of usable rows of the **D** matrix, sparsely sampled training data, where the gridded data have many unknown cells, can have an all-zero **W** matrix, making PEF estimation impossible. For such sparsely sampled data, we can regrid the data with coarser bins so that the number of missing data points (and the number of zero-values in **W**) is reduced, as in Figures 3.3a and 3.3c. We can also shift the origin point of this grid, as in Figures 3.3d-f. Depending on the origin point and the cell size of the grid, the gaps present in the data will differ. If many different grids are used, more of these gaps will be filled so that more rows of the **W** matrix are non-zero. The regridding procedure used to generate these multiple grids is described next.

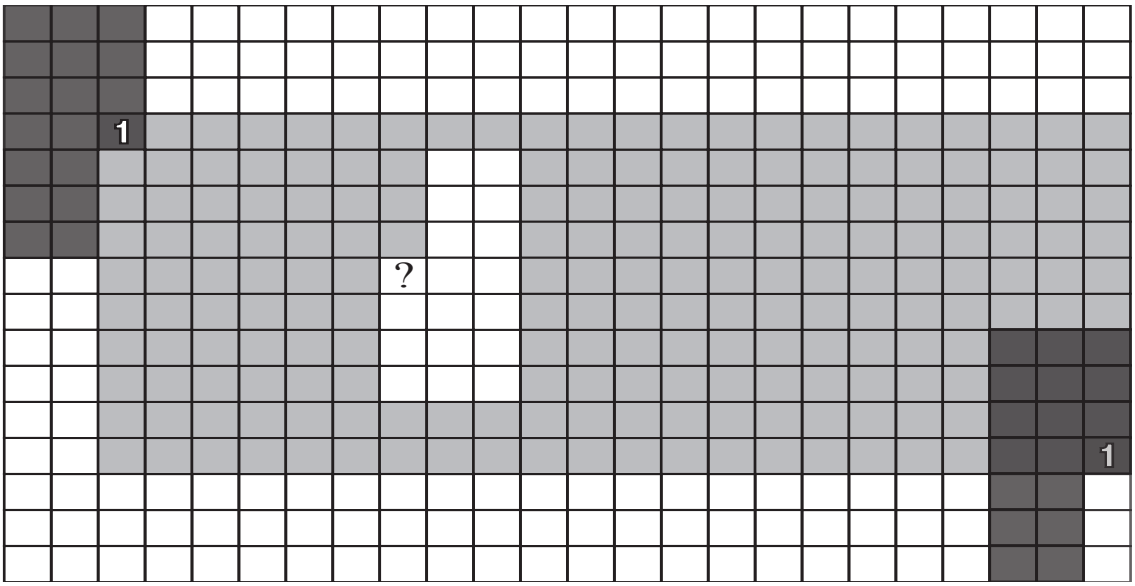

Figure 3.2: An example of missing data and boundary roll-off when estimating a 7×3 (18 coefficient) PEF, shown in black. The 24×16 grid has three rows at both the top and bottom as well as two columns on the left where the PEF falls of the boundary of known data, so those output locations are ignored, denoted with white cells. The one unknown data value, denoted with a ?, causes $n_{\mathrm{p}} = 18$ output points, also shown in white, to be ignored. The remaining usable output locations, where the PEF falls entirely upon known data, are shaded gray. **NR** MSPEF/. PEF-boundaries

ESTIMATION OF A PREDICTION-ERROR FILTER WITH MULTI-GRID DATA

Sparsely sampled data can be represented as a combination of known and unknown values on a desired regularly sampled grid as shown in Figure 3.3a. Mapping from one grid to another is a two-stage process. The first stage is mapping the known values of the original grid to a vector of known points, as in Figure 3.3b. We do this with a sampling matrix $\mathbf{B}$ that is n_{d} elements horizontally and n_{k} elements vertically, so it maps from the n_{d}-length vector containing both known and unknown points to a n_{k}-length vector. These data points can be placed on a different grid with a normalized linear interpolation matrix that maps the known data points onto the $n_{\mathrm{d}_{\mathrm{r}_i}}$-length output grid vector with a matrix, $\mathbf{L}_i$. This new grid is generally coarser and also can have different origin points, as the grids in Figures 3.3c-f do. Cascading these two operators maps data from a fine grid to a coarser grid, so that

$$\mathbf{d}_{\mathrm{r}_i} = \mathbf{L}_i \mathbf{B} \mathbf{d}. \tag{3.5}$$

Applying equation 3.5 to the illustrated example in Figure 3.3a with varying grid origins and sizes produces Figures 3.3c-i, with Figure 3.3b as the intermediate $\mathbf{Bd}$ result. These different grids each provide additional information to the PEF estimation. This regridding is more useful in higher dimensions, where the $\mathbf{L}$ matrix is a bilinear (2D) or trilinear (3D) interpolation matrix. A coarsened version of the data is useful as training data for a PEF because plane waves are somewhat scale-invariant (Claerbout, 2004), so that a PEF estimated on coarser scale data typically contains the same slopes as do the original scale of data. This assumption is not completely satisfied for the frequency content of the data of curved events, where slope is scale dependent.

Regularly-sampled data have a single grid that optimally captures the data: one datum within each bin. Because this chapter addresses irregularly sampled data, no single grid is ideal because different regions of the training data are captured best with different grids. To use irregularly-sampled data effectively, we can estimate a

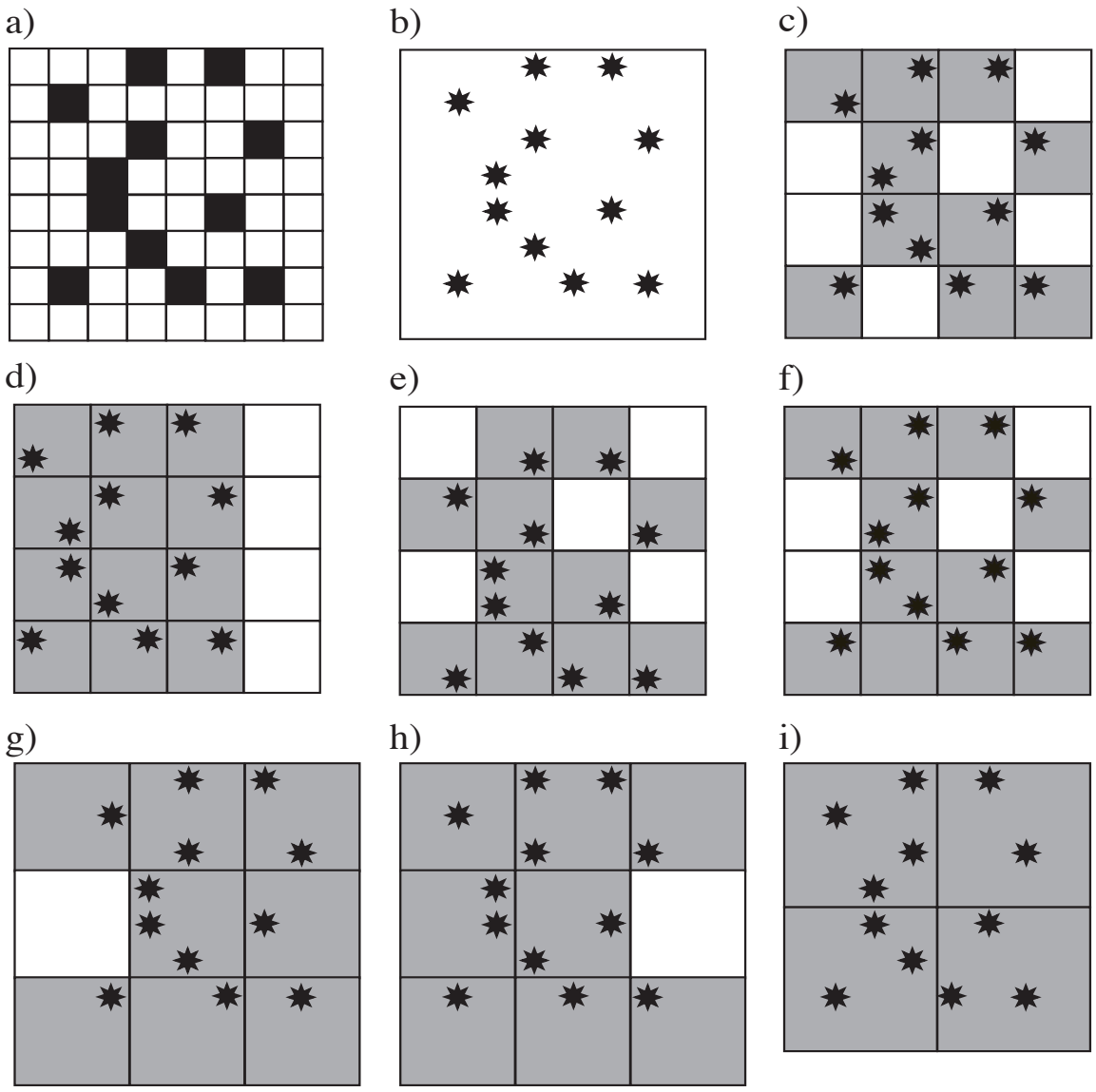

Figure 3.3: Regridding of data with $n_{\mathrm{k}} = 12$ known points on a $n_{\mathrm{d}} = 64$-cell grid. (a): original data on a finely-sampled grid; known data are shaded. (b): data from known cells are sampled with matrix $\mathbf{B}$ and their locations stored. (c)-(i): multiple different grids with varying origin points for multiple cell sizes generated by applying different $\mathbf{L}_i$ matrices. The different grids produce different distributions of known grid values. **NR** MSPEF/. Regridding

single set of PEF coefficients on multiple regridded copies of the data, each grid with a different cell size or origin. Taking equation 3.2 and replacing the $\mathbf{W}$, $\mathbf{D}$, and $\mathbf{d}$ matrices and vectors with their multi-grid equivalents (denoted by a subscripted m), we have

$$\mathbf{Kf} = -(\mathbf{D}_{\mathrm{m}}^{\dagger}\mathbf{W}_{\mathrm{m}}\mathbf{D}_{\mathrm{m}})^{-1}\mathbf{D}_{\mathrm{m}}^{\dagger}\mathbf{W}_{\mathrm{m}}\mathbf{d}_{\mathrm{m}}. \tag{3.6}$$

The data vector $\mathbf{d}_{\mathrm{m}}$ is a concatenation of n regridded versions of the data, $\mathbf{d}_{\mathrm{m}} = [\mathbf{d}_{\mathrm{r}_1}|\mathbf{d}_{\mathrm{r}_2}|\cdots|\mathbf{d}_{\mathrm{r}_n}]^T$, for a total length of $n_{\mathrm{d}_m} = \sum_i n_{\mathrm{d}_{\mathrm{r}_i}}$. Similarly, $\mathbf{D}_{\mathrm{m}}$ is a concatenation of n convolutional matrices, totaling $n_{\mathrm{d_m}}$ rows and n_{p} columns, so that

$$\mathbf{D}_{\mathrm{m}} = \begin{bmatrix} \mathbf{D}_{\mathrm{r}_1} \\ \mathbf{D}_{\mathrm{r}_2} \\ \vdots \\ \mathbf{D}_{\mathrm{r}_n} \end{bmatrix}. \tag{3.7}$$

$\mathbf{W}_{\mathrm{m}}$ is a $n_{\mathrm{d_m}} \times n_{\mathrm{d_m}}$ matrix constructed from multiple versions of the $\mathbf{W}$ matrix in equation 3.2 for each of the rescaled copies of the data, so that

$$\mathbf{W}_{\mathrm{m}} = \begin{bmatrix} \mathbf{W}_{\mathrm{r}_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{\mathrm{r}_2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{W}_{\mathrm{r}_n} \end{bmatrix}. \tag{3.8}$$

This approach uses multiple regridded copies of the sampled data as training data, as data on a coarser grid should have fewer holes and better constrain a PEF.

STATIONARY PEF ESTIMATION ON SYNTHETIC DATA WITH MULTIPLE GRIDS

Figure 3.4a is the same two-plane data used as the stationary test case in Chapter 2, and Figure 3.4b is a random sampling of 30 percent of the data traces. Interpolating these data using the same (10×3) 2D PEF estimated in Chapter 2 from fully-sampled

data in Figure 3.4a results in the successful interpolation in Figure 3.4c, where the interpolated data contain the same spectrum as the recorded data, with slightly-lower amplitude where the large gaps were. We now attempt to estimate a PEF using only the sampled data by using the multiple-grid approach to generating training data.

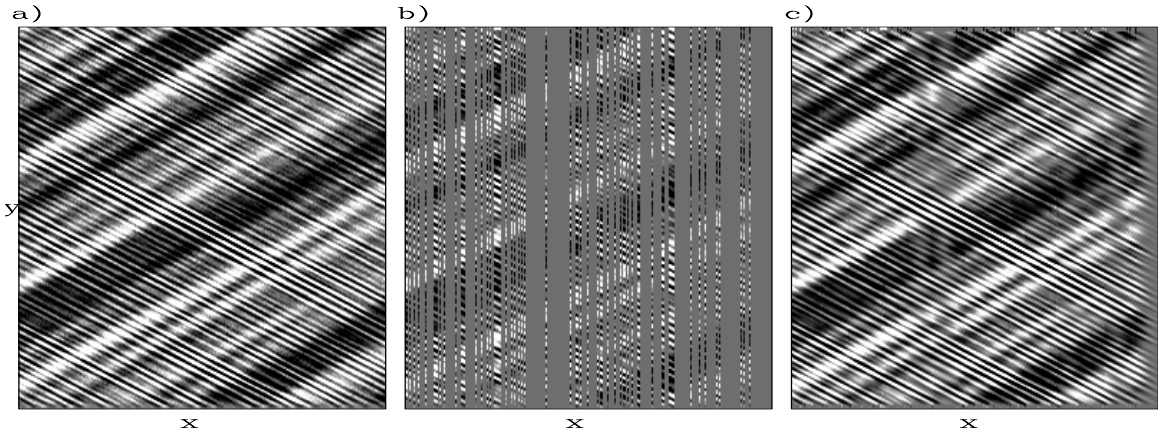

Figure 3.4: The two-plane data from Figure 2.2: (a) fully sampled; (b) with 30% of traces randomly sampled; (c) interpolation of (b) using a PEF estimated on (a). **ER** MSPEF/. idealtraceann

Multiple regridded versions of the sampled data are shown in Figure 3.5, where I vary the coarseness of the grid from the original 256×256 down to 64×64. The higher-frequency event is not present in the coarser grids, as the regridding has effectively acted as a high-cut filter in both dimensions.

I generate seven regridded copies (one 192×192, two 128×128 and four 64×64) of the sampled data using equation 3.5, and place those regridded data along with the original 256×256 data into a 150 thousand-element vector $\mathbf{d}_{\text{m}}$ and the matrix $\mathbf{D}_{\text{m}}$ in equation 3.6. I then use all of these scales of training data to generate a single PEF. I use 100 iterations of a conjugate-direction solver on equation 3.6, and with the PEF thus obtained, I use it to interpolate the missing data using equation 2.18. Comparing this result in Figure 3.6c with both an interpolation using an isotropic Laplacian filter in Figure 3.6a and a PEF estimated on a single 64×64 regridded data in

Figure 3.6b, the multiple-grid result interpolates both the low-frequency and the high-frequency events, whereas the Laplacian filter (minimum curvature interpolation) only interpolated the low-frequency event and the single-grid PEF incorrectly interpolated the high-frequency event with the slope of the low-frequency event.

The multiple-grid PEF, however, produced a poorer result than the interpolation with the ideal PEF (Figure 3.4c), which more completely interpolated the high frequency event. This is not surprising: Figure 3.4c was generated using a PEF estimated on the fully-sampled data, in effect providing the answer to the problem. The differences between Figures 3.4c and 3.6c lie in the approximations used to obtain the training data; that a regridded version of the data could act as a surrogate for the fully-sampled data. The finer of the grids were sufficient for the PEF to capture the high-frequency event, but not with the fidelity of the fully-sampled data, meanwhile the low-frequency event was sufficiently captured in both the multi-grid and the single 64×64 grid, as it was present in all of the grids in Figure 3.5. This example was stationary, with two constant dips. I next adapt nonstationary PEF estimation to use multiple grids of data.

ESTIMATION OF A NONSTATIONARY PREDICTION-ERROR FILTER WITH MULTI-GRID DATA

Multi-grid training data can also be used to estimate nonstationary prediction-error filters, in order to extract as much local information as possible, that is crucial for generating a nonstationary PEF. The multiple grid estimation equations are similar to those for nonstationary PEF estimation in equation 2.21, so

$$\min_{\mathbf{f}_{\rm ns}} \|\mathbf{r}_{\rm d}\|^2 + \epsilon^2 \|\mathbf{r}_{\rm f}\|^2 \tag{3.9}$$

$$\mathbf{r}_{\rm d} = \mathbf{W}_{\rm m}\mathbf{D}_{{\rm ns}_{\rm m}}\mathbf{K}_{\rm ns}\mathbf{f}_{\rm ns} + \mathbf{d}_{\rm m} \tag{3.10}$$

$$\mathbf{r}_{\rm f} = \mathbf{R}\mathbf{f}_{\rm ns}. \tag{3.11}$$

Here, I do not change the nonstationary filter, rather the data being used in the estimation, so the estimation is still the minimization of a nonstationary convolution. In equation 3.11, the data vector, $\mathbf{d}_{\mathrm{m}}$, is now the same $n_{\mathrm{d_m}}$-element vector of concatenated regridded versions of the data from equation 3.6, and the nonstationary data convolutional matrix $\mathbf{D}_{\mathrm{ns_m}}$ is again a concatenation of multiple data-convolutional matrices (this time the nonstationary convolutional matrices from equation 2.21).

One complication to this approach is that each of the component matrices in the $\mathbf{D}_{\mathrm{ns_m}}$ matrix are mapping between the non-stationary filter of length $n_{\mathrm{d}} \times n_{\mathrm{f}}$ to a component of the rescaled data vector $\mathbf{d}_{\mathrm{m}}$, which is of a differing length, $n_{\mathrm{d_{r_i}}}$. I use a subsampling matrix, $\mathbf{S}$, that I apply to the nonstationary filter, $\mathbf{f}_{\mathrm{ns}}$, and subsample it from a $n_{\mathrm{d}} \times n_{\mathrm{f}}$-length vector to a $n_{\mathrm{d_{r_i}}} \times n_{\mathrm{f}}$ length vector, so that $\mathbf{D}_{\mathrm{ns_m}}$ is

$$\mathbf{D}_{\mathrm{ns_m}} = \begin{bmatrix} \mathbf{D}_{\mathrm{ns_{r_1}}} \mathbf{S}_{\mathrm{r_1}} \\ \mathbf{D}_{\mathrm{ns_{r_2}}} \mathbf{S}_{\mathrm{r_2}} \\ \cdots \\ \mathbf{D}_{\mathrm{ns_{r_n}}} \mathbf{S}_{\mathrm{r}_n} \end{bmatrix}. \tag{3.12}$$

Now we have replaced all of the data from equation 3.11 with multiple regridded versions of that data. We also have to include the matrix $\mathbf{W}_{\mathrm{m}}$ from equation 3.6 that zero-weights rows of $\mathbf{D}_{\mathrm{ns_m}}$ containing unknown data.

MULTIGRID ESTIMATION OF A NONSTATIONARY PEF ON SYNTHETIC DATA

Now that nonstationary PEF estimation from multiple grids of data has been covered, I test this approach on the 3D synthetic quarter-dome data from Chapter 2. Recall that I estimated a nonstationary 3D PEF on a fully-sampled version of the quarter-dome synthetic dataset and used it to interpolate a near-perfect result from a severe random sampling of 30 percent of the traces from the data; that was the result, however, of using ideal training data. Now, instead of using ideal fully-sampled training data we start with only the randomly-sampled traces from the data, and we

generate multiple regridded training data from these sparse data.

The quarter-dome data are again shown fully-sampled in Figure 3.7a and with only 30 percent of traces randomly selected in Figure 3.7b. A PEF cannot be estimated on the data in Figure 3.7b because of the lack of contiguous data, making $\mathbf{W}$ zero *everywhere* in equation 3.11. To overcome this problem, the randomly-sampled data are regridded to four different bin sizes in Figures 3.8a-d, where the bins are anywhere from 25 to 400 percent larger along each axis. As the size of the bins increases, the holes in the data become smaller, and more contiguous regions appear from which we can estimate prediction-error filter coefficients. Eventually, the quality of the regridded data degrades, such as the $40 \times 20 \times 10$ data in Figure 3.8d where the sampled data have destructively interfered within the large grid cells.

Figure 3.9a presents a baseline comparison generated by interpolating the randomly sampled data in Figure 3.7b by solving equation 2.18, but using a three-dimensional Laplacian filter rather than a prediction-error filter. The stationary regions at shallow and great depths are properly interpolated as the slopes in both of these regions are small. The areas with large slopes are poorly reconstructed. This is because the Laplacian filter is isotropic and does not interpolate well along the steep events. I then estimated a $10 \times 3 \times 3$ nonstationary PEF that varied every second point on each axis, for a total of over 9 million coefficients. I estimate this PEF on the four regridded copies of the data shown in Figure 3.8, and then use it to interpolate the data to produce the result in Figure 3.9b. This result is much better than the Laplacian result, but is still unacceptable over the portions of the data with high curvature. Given the sparsity of the data and the rapidly-changing slope of the underlying model, there is no hope of reconstructing this region without external information.

I generate more regridded copies of the data by not only varying the size of the cells in the grid, but also their origin point. Figure 3.9c was generated by estimating the same nonstationary PEF, but now on an additional 28 different grids where the origin of the grid has been variously shifted. As grid bin size increases, the number of possible shifts does likewise, so there is only one $160 \times 80 \times 40$ or $120 \times 60 \times 30$

grid, but there are four possible $80 \times 40 \times 20$ grids and 27 possible $40 \times 20 \times 10$ grids.

This result is noticeably improved over that with four scales of data in Figure 3.9b. The slowly-varying areas at the deep right of the depth slice are not noticeably improved, as they were reasonably interpolated with four scales. The center-left of the depth slice is noticeably improved, where the coherent energy introduced by the additional shifted grids of data appears in the difference panel between the four-grid and 34-grid PEF in Figure 3.9d. The multiple-grid approach still cannot reconstruct the rapidly-changing area of the model with the quality of the result of the ideal nonstationary PEF in Figure 2.17b, but this goal is unrealistic given the small amount of input data. The improvement of the result when more gridded datasets are added to the estimation shows how additional regridded data contributes to the result.

Both the stationary result and the nonstationary quarter-dome result are based upon random sampling of traces. Most data are not acquired in such a manner, such as the 2D land field data shown in the next section.

MULTIGRID ESTIMATION OF A NONSTATIONARY PEF ON 2D LAND FIELD DATA

While the previous examples are synthetic data with random sampling, field data are typically not randomly sampled, nor so sparsely sampled. The Hulia dataset is a 2D split-spread land survey acquired in Colombia. The sampling of the data, shown in source-offset coordinates in Figure 3.10 has occasional gaps on the 790-element source axis as well as both a near-offset gap and periodic gaps in the 420-element offset axis, both sampled in 20-m intervals. With the reciprocal traces included, the missing sources now contain some offsets from these reciprocal traces, with roughly half of the grid unsampled.

The recorded data in Figure 3.11 have a low signal-to-noise ratio, with few discernible reflectors in the constant-offset section (520-m) on the left panel. Both the shot gather at 6220 m in the right panel and the time slice at 1.68s in the top panel

show little usable signal. The gaps in the recorded data can be seen in all three of the panels, the missing sources in the constant-offset panel, the missing offsets from the reciprocal traces of these shots in the shot gather, and the near-offset gap in the time slice.

I break the data into 10 overlapping windows along the source axis that I treat as independent problems because of memory restrictions. I generate four regridded versions of the data, where the cells are 25, 66, and 150 percent larger along each axis than in the original 20-m $\times$20-m, 4-ms cell. Once these regridded copies of the data are generated, I estimate a $10 \times 3 \times 3$ PEF that varies every 40 time samples, every 10 source locations, and every 15 offsets, for a total of 375000 filter coefficients. I applied 900 iterations of a conjugate-direction solver on equation 3.6 to estimate these PEF coefficients. This lower number of coefficients, compared to the quarter-dome example is possible because of the more gradual changes in dip in the field data.After estimating the PEF from the multiple grids of data, I apply it in 400 iterations of a conjugate-direction solver on equation 2.29 to reconstruct the missing data (Figure 3.12). The total computational cost of this interpolation is on the order of one day for each window on a single core of a machine from 2006.

The interpolation in Figure 3.12 is a mixed result. The shot gather on the right panel of Figure 3.12 contains multiple dips, many aliased, which were correctly interpolated. The near-offset gap was only interpolated successfully at later times, which I attribute to three factors. First, there is a lack of out-of-plane information; the near-offset gap is present for all sources. Second, the recorded data at later times extends to further offsets, which implies that more data contribute to estimate a PEF at near offsets and later times. Finally, the frequency content of the data is much lower at later times, and would be less attenuated in the coarser grids of data.

The constant-offset section on the left panel of Figure 3.12 contains relatively few gaps compared with many other offsets, but the gaps in the region with the most coherent reflections, at roughly 13000 m, are correctly interpolated. Conversely, the interpolated gap at roughly 4000 m is weak below the first arrival. This is because the recorded data surrounding the gap is almost incoherent, as seen in Figure 3.11. The

time slice in the top panel of Figure 3.12 is more coherent, even at the near offsets, as this slice is at 1.68 s.

This interpolation can also be judged based on the results of further processing of the data. One such example is migration, which requires a velocity model that is not available for this dataset. One way to generate a root-mean-square velocity model is through semblance scans. Figure 3.13a is a semblance scan for an area of the survey with higher signal-to-noise, where I calculated semblances from ten adjoining common midpoint gathers and summed the results. Figure 3.13b shows the same semblance scan using the interpolated data, which, containing less noise, is easier to interpret, particularly at early times. The differences are slight but still noticeable, and given that this image is the result of combining ten CMP gathers the benefits of the interpolation are not totally negligible.

CONCLUSIONS AND FURTHER WORK

Multi-grid training data can be used to estimate a prediction-error filter on data that are insufficiently sampled. It was useful for both stationary and nonstationary cases, in that a PEF could be estimated from data that were insufficiently sampled to estimate a PEF in the conventional manner. While the PEF produced is superior to a PEF produced from a single grid of data, this PEF is inferior to one estimated from ideal training data. The basic assumption used is that a regridded version of the data contains the same dip spectra as the data on the original grid, essentially the familiar assumption that the slopes at lower frequencies are the same as those at higher frequencies. This assumption was reasonably valid for the stationary and quarter-dome cases, and somewhat less accurate for the field data.

Results were improved by increasing the number of regridded data sets used. The size and location of the grids were manually chosen and the cell sizes were evenly distributed, which is reasonable for randomly-sampled data. An improvement to this method is to choose scales more intelligently so that they best fit the data; the portions of the data that are better-captured by one grid could also be weighted higher

relative to the other scales by using a more sophisticated $\mathbf{W}_{\mathrm{m}}$ matrix. However, field data are rarely randomly-sampled, as ships travel in straight lines and receivers are typically attached to cables. Increasing the size of grid cells in this scenario is a poor choice, since the cell size from one grid to another requires the same aspect ratio for scale-invariance so the recorded data will be placed on too coarse of a grid. For example, receivers in a 3D land survey are well-sampled along the cable, but are poorly-sampled between cables. Since the desired output grid cell is (roughly) square, the only grid that would provide contiguous data from cable to cable would drastically undersample the axis along the cable, so that higher frequency and higher-wavenumber events would be penalized.

This method is not ideal for other systematic gaps in data acquisition, such as the near-offset gap present in the field data. These gaps require external information instead of only information from the nearest traces. One example of such information is pseudo-primary data, discussed in the next chapter.

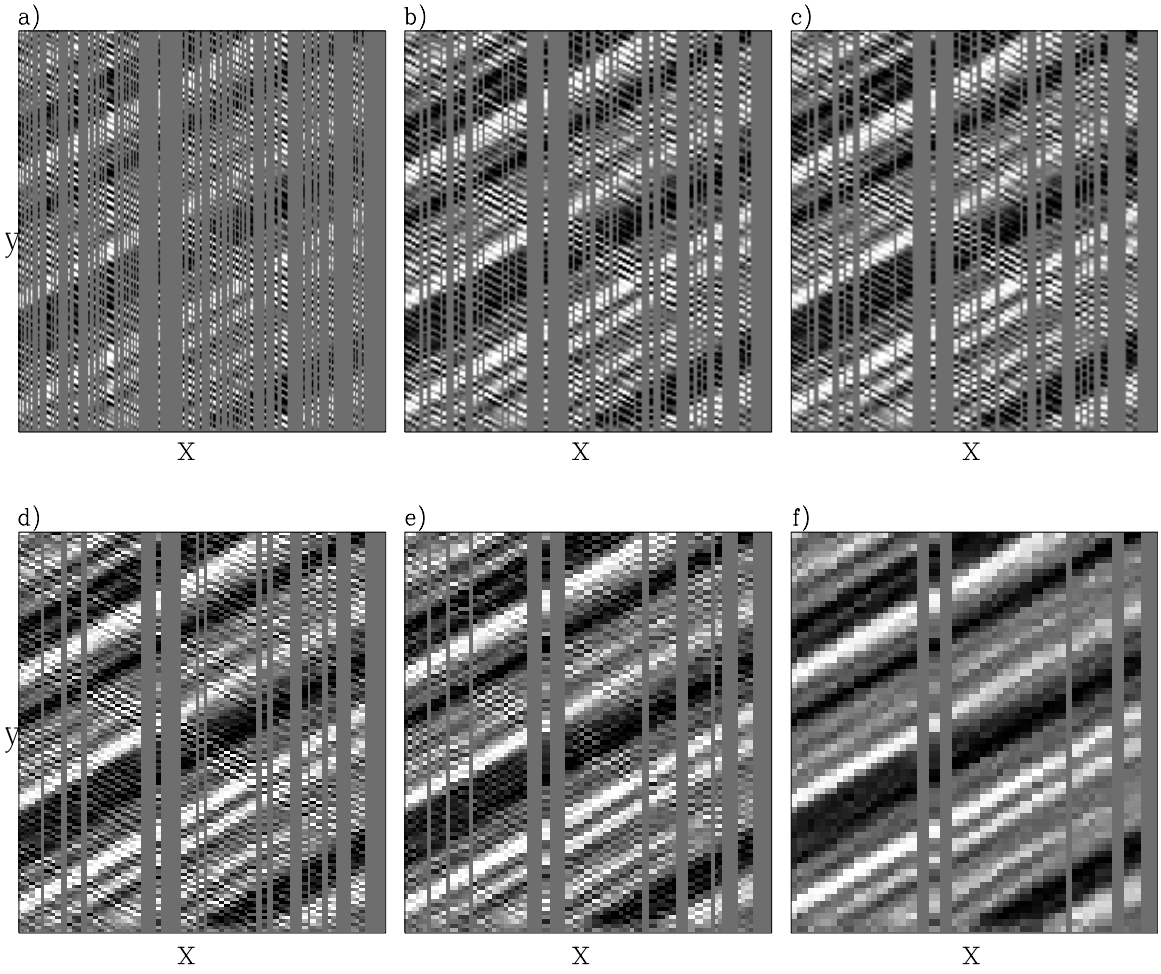

Figure 3.5: Six coarsening regridded versions of the (256×256) sampled data: a) 256×256; b) 192×192; c) 160×160; d) 128×128; e) 96×96; f) 64×64. At the coarser scales the high-frequency event disappears. **ER** MSPEF/. tracescalesann

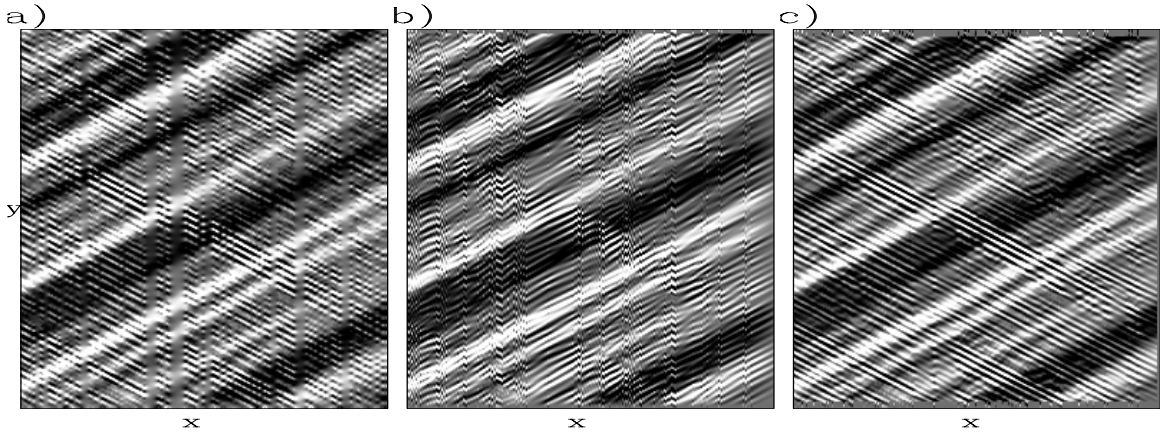

Figure 3.6: Sampled data from Figure 3.4b interpolated three ways using equation 2.18 with the filter $\mathbf{f}$ as a: (a) Laplacian filter; (b) PEF estimated from a single 64×64 regridded version of 3.4b; (c) PEF estimated from nine different regridded copies of the data. The high-frequency dip is correctly interpolated only with the PEF estimated on multiple-grid data. **ER** MSPEF/. multitracefillann

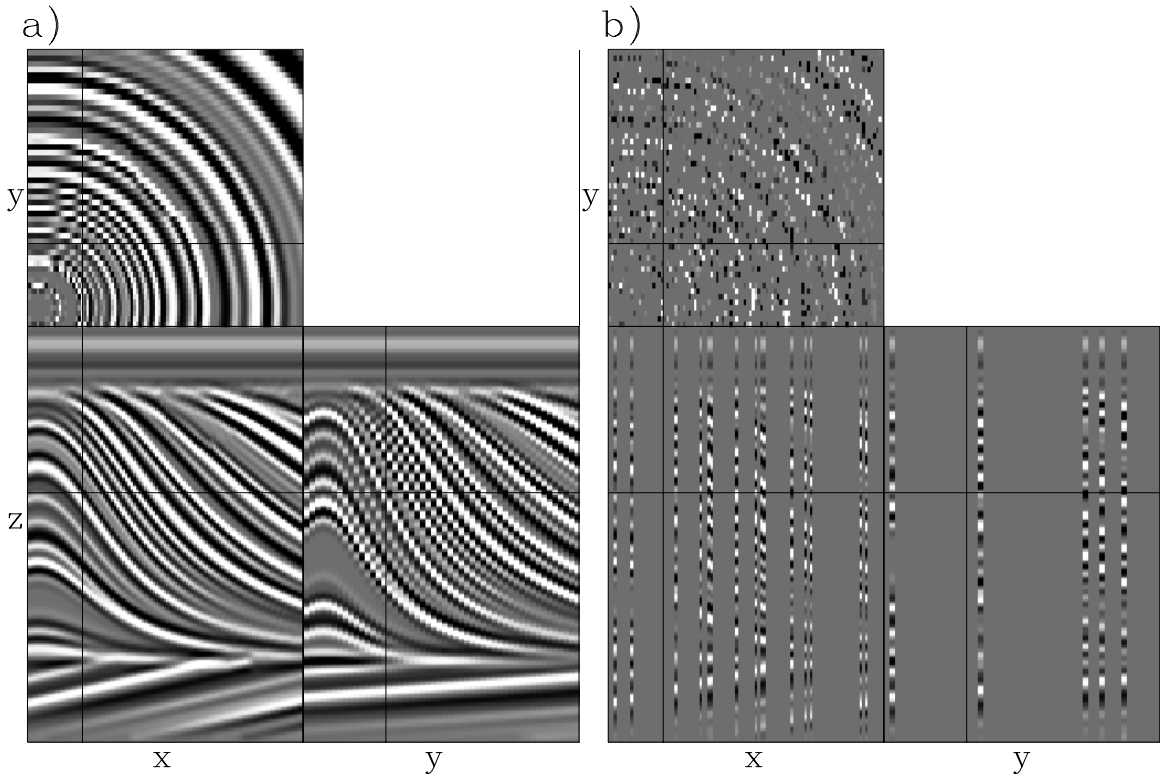

Figure 3.7: The quarter-dome synthetic: fully-sampled (a) and with 30 percent of traces randomly-sampled (b). **ER** MSPEF/. qdomeann

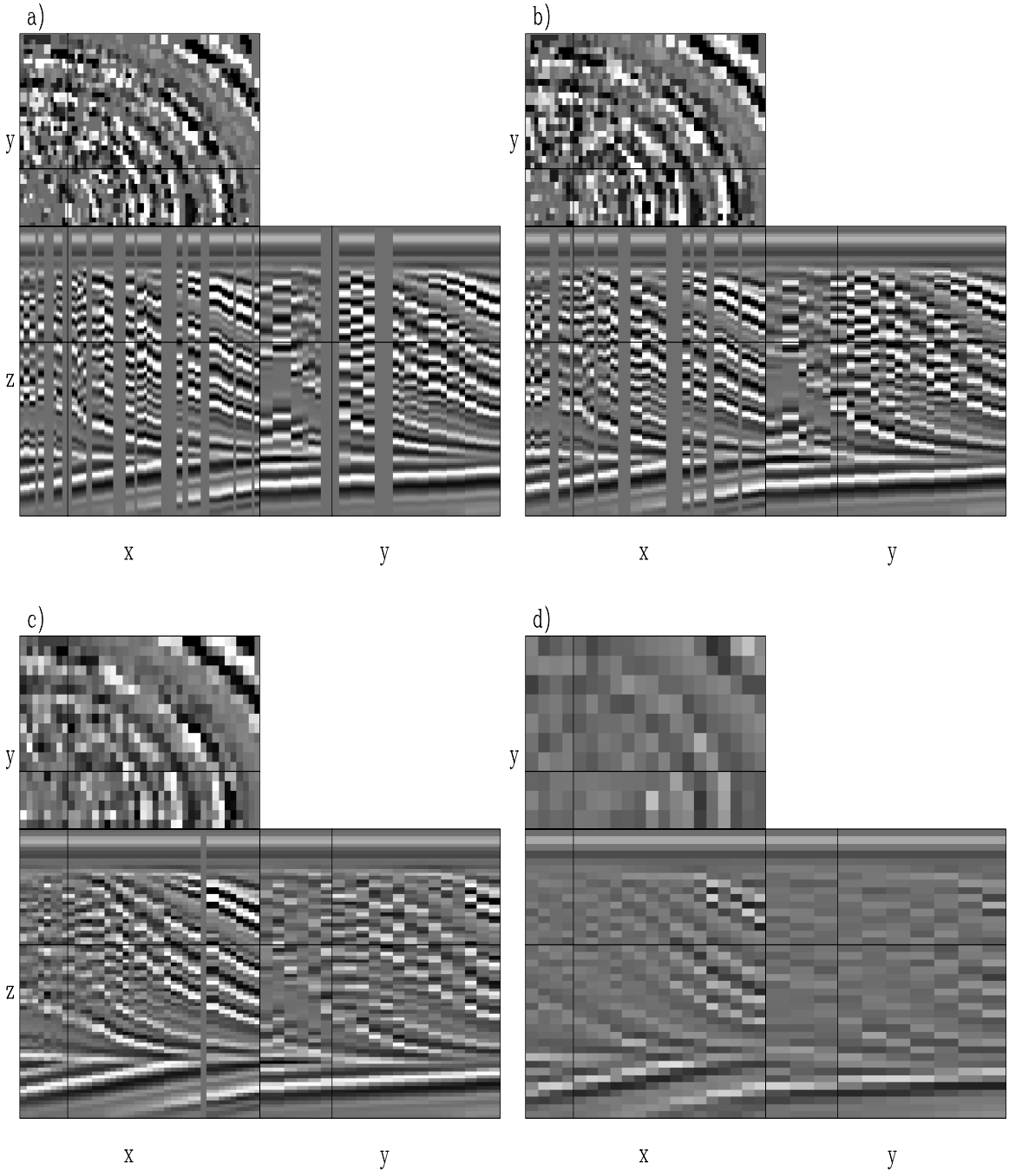

Figure 3.8: The randomly-sampled quarter-dome data (200×100×50) on four different grids: (a) $160 \times 80 \times 40$, (b) $120 \times 60 \times 30$, (c) $80 \times 40 \times 20$, (d): $40 \times 20 \times 10$. As the number of cells in the grid decreases, the number of unknown cells also decreases. **ER** MSPEF/. qdomescaledann

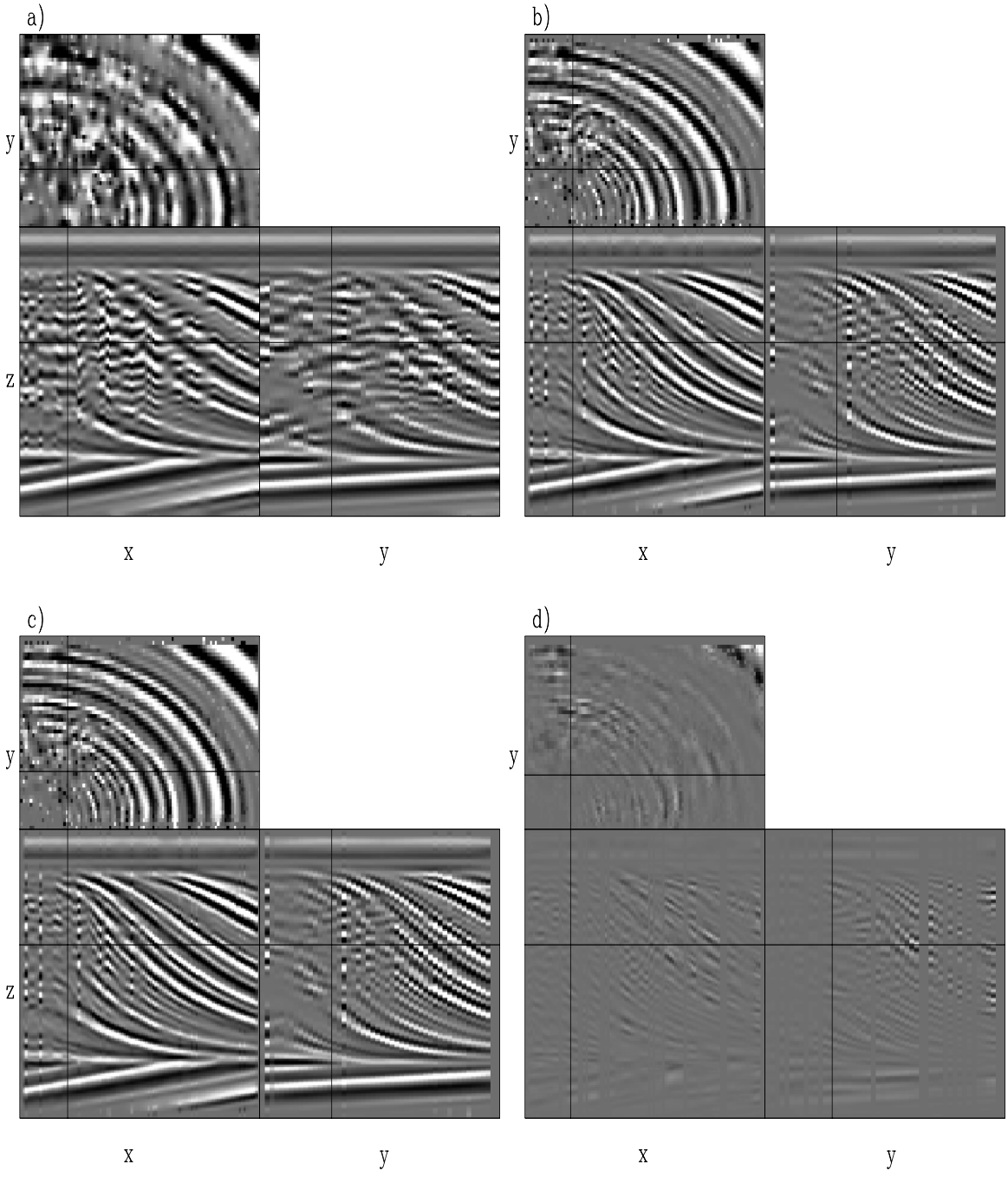

Figure 3.9: Interpolations of the randomly sampled quarter-dome data from Figure 3.7b: a): Laplacian interpolation; b): Interpolation with a PEF generated from four grids of data; c): Interpolation generated from 32 grids of data; d) Difference between b and c. The added grids of data improve the range over where the interpolation succeeds, but all methods fail at the highly nonstationary region. **CR** MSPEF/. qdomeinterpann

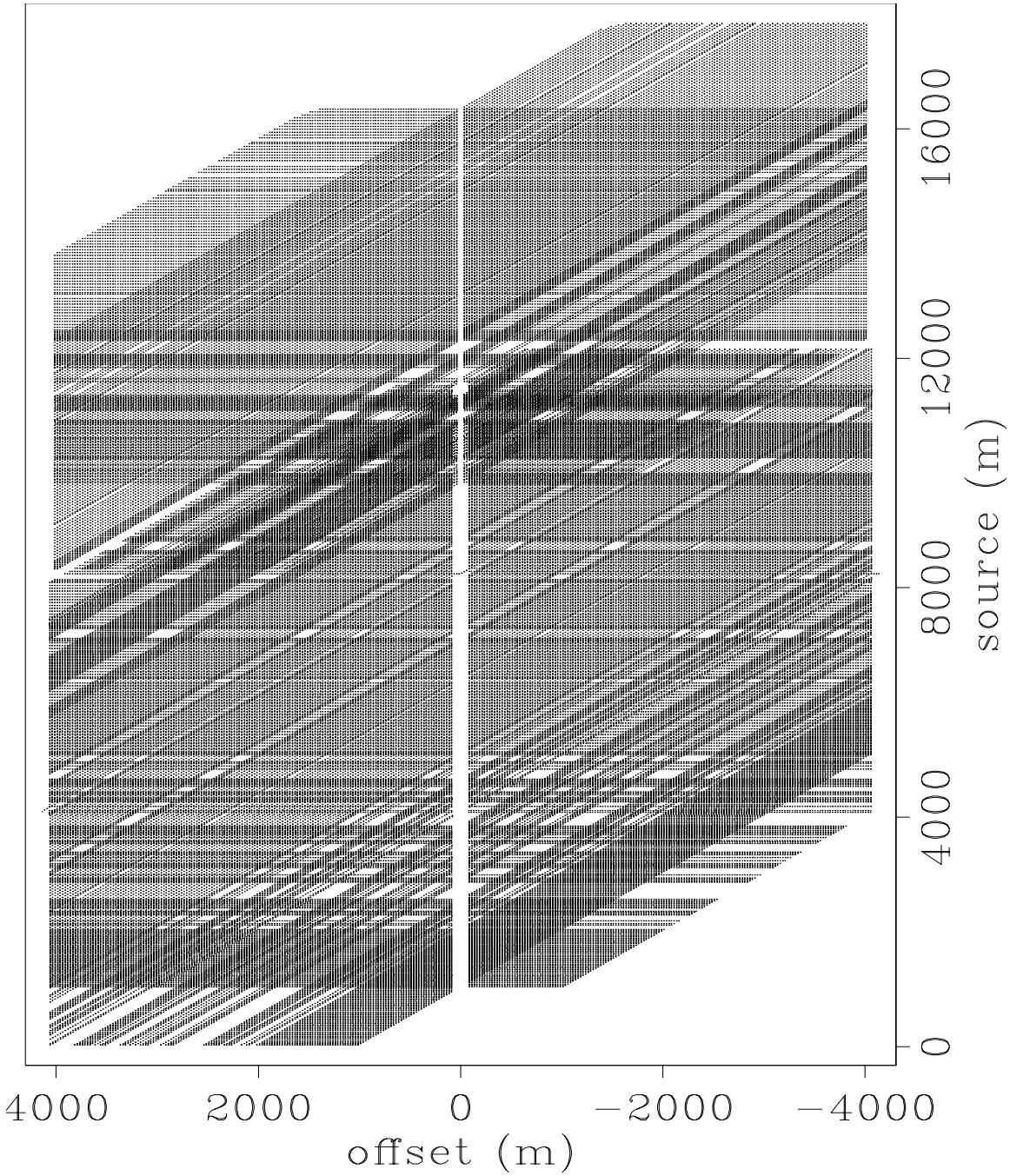

Figure 3.10: The source-offset locations of the Hulia dataset, with points representing the positions of traces in source-offset space. Both original recorded data coordinates as well as traces predicted by source-receiver reciprocity, which appear as diagonal lines, are included. **ER** MSPEF/. huliamap

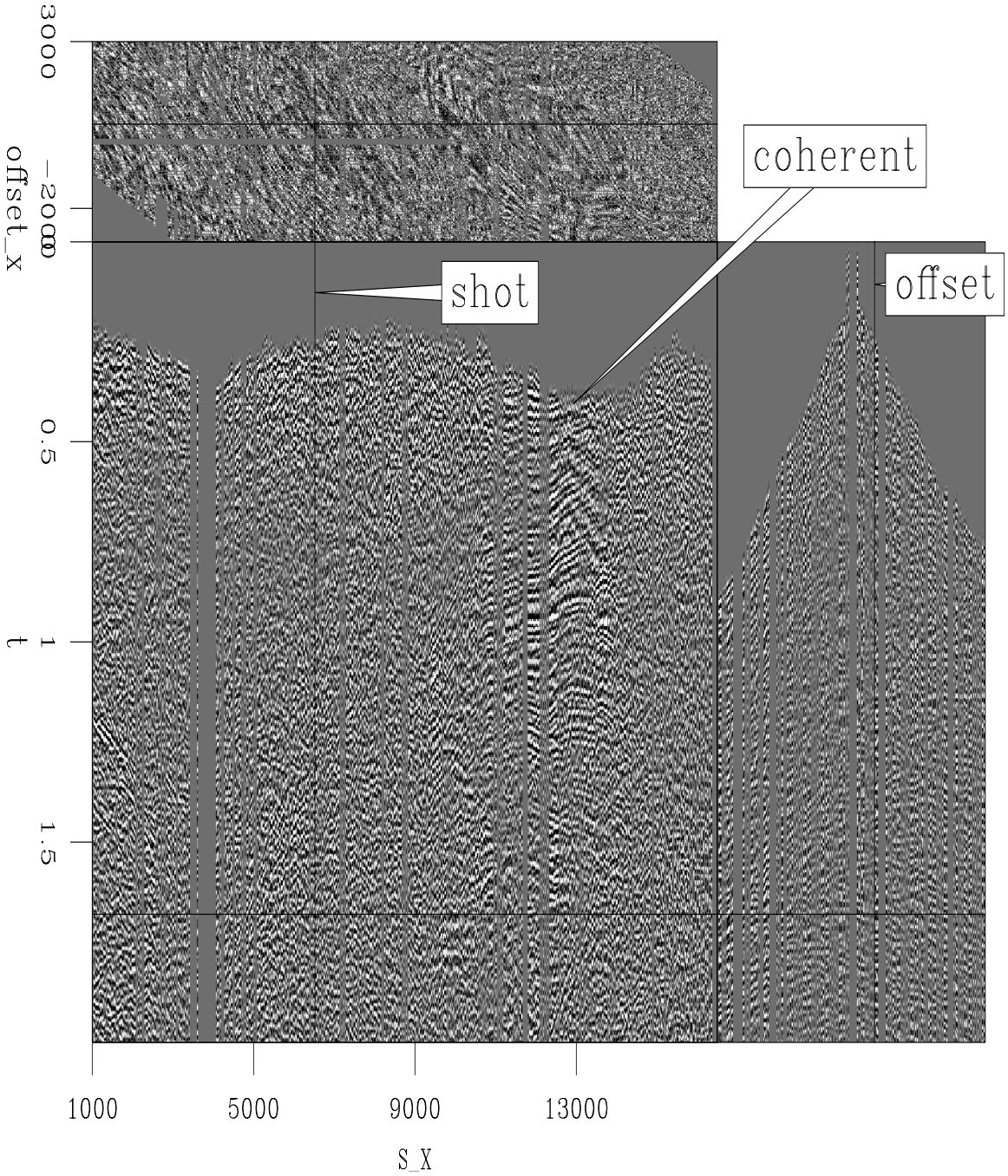

Figure 3.11: The Hulia dataset in source and offset. As seen, the data contains gaps visible in the common-offset gather at 520 m on the left, the shot gather at 6220 m on the right, and the time slice at 1.68 s on the top. **ER** MSPEF/. huliaann

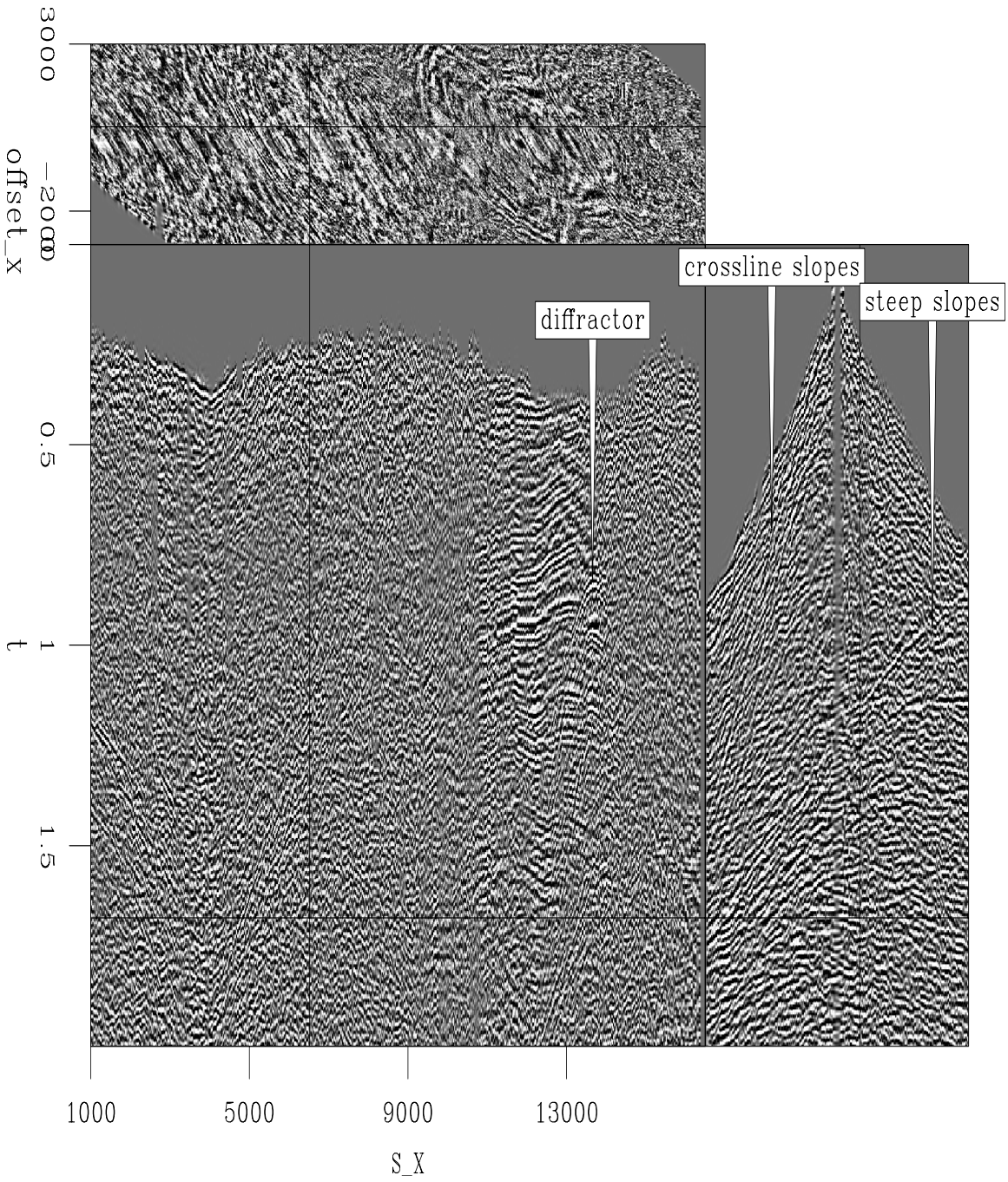

Figure 3.12: The Hulia dataset interpolated with nonstationary PEFs estimated from 4 regridded versions of the data. The shot gather contains multiple conflicting dips, as well as spatially-variable slopes that are successfully interpolated. **CR** MSPEF/. huliainterpann

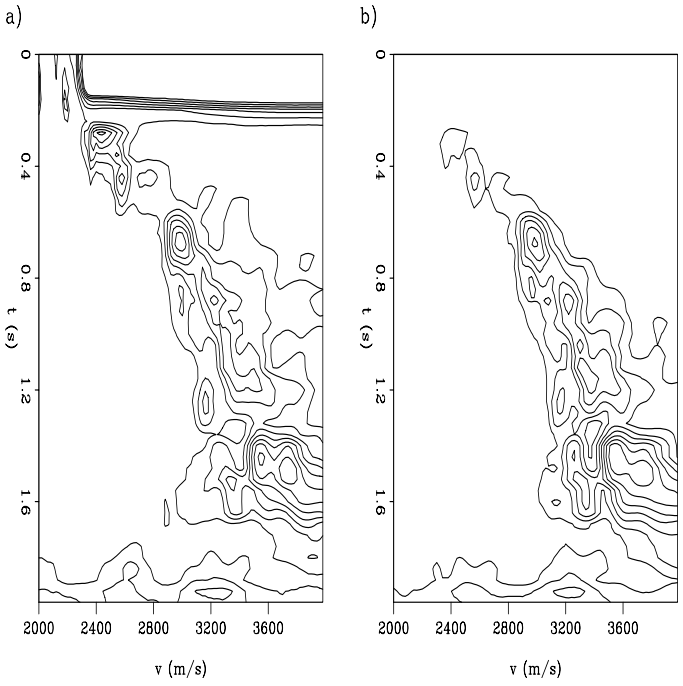

Figure 3.13: Two semblance scans generated from super-gathers. (a) semblance generated from 10 adjacent CMP gathers from original data. (b) semblance generated from same data after interpolation. The supergather in (b) is slightly more focused and less noisy than the original super-gather in (a). **CR** MSPEF/. huliascans

Chapter 4

Interpolation of near offsets using multiples

As discussed in Chapter 1, 3D marine reflection seismic data are acquired along as many as five axes, most of which are inadequately sampled. Chapter 2 describes how to address this problem by interpolating data with a nonstationary prediction-error filter that is first estimated from fully-sampled training data and then used to interpolate missing data to produce an interpolated output. These training data need not be perfect, and may differ in amplitude and phase but should contain the local multi-dimensional amplitude spectra of the data we wish to recreate. In this chapter, I generate pseudo-primary data by crosscorrelating multiples and primaries in the recorded data (Berkhout and Verschuur, 2003). These pseudo-primary data can be generated at the missing near offsets, but contain many artifacts, so it is undesirable simply to replace the missing data with the pseudo-primaries. Fortunately, many of the problems with the pseudo-primaries do not influence PEF estimation, so a desirable PEF can be obtained from these data, and then used to interpolate the missing near inline offsets to produce a result that is superior to direct substitution of the pseudo-primaries into the missing offsets.

In most data sets, after the over-sampled time axis, the inline receiver axis is

the most densely-sampled axis; the receivers are attached to a single cable so the sampling is dense and regular along this cable. Since both the air-gun source and receiver cable are usually towed by the same boat, the maximum inline offset, the largest inline distance between the source and a receiver, is limited by the length of the cable. The near end of the receiver cable is not at the source for obvious reasons; instead the receiver cable is towed a fixed distance behind the source. This distance, the near-offset, is consistent throughout the survey and is typically several times the inline receiver sampling interval. An example of a single shot from a 2D marine survey showing this near-offset gap is in Figure 4.1.

Near-offset traces are particularly valuable. Many methods attempt to recreate zero-offset data from larger offsets; standard multiple-removal techniques, such as surface-related multiple elimination (SRME) (Verschuur et al., 1992), require zero-offset data. Moveout differences between primaries and free-surface multiples are slight at these near-offsets thus compromising the performance of radon-based multiple removal algorithms that discriminate based on differential moveout.

There are many methods that could be used to reconstruct this near-offset gap. A simple way of recreating the missing near offsets is to replace the missing offsets with an NMO-corrected trace from the nearest offset. More sophisticated radon-based methods (Sacchi and Ulrych, 1995; Trad et al., 2002) are commonly used, as are Fourier-based methods (Liu and Sacchi, 2001; Xu et al., 2005). These methods all use existing data recorded adjacent to the missing near offsets to create the missing data, so the performance degrades as the gap increases in size.

Another approach to generating data at the missing near offsets starts by first creating pseudo-primary data. Pseudo-primary data are created by crosscorrelating every trace with every other trace within a shot (Berkhout and Verschuur, 2005). The free-surface multiples correlate with the primaries at lags comparable to times when a primary reflection would arrive if one of the receiver locations was the source. Since the receivers now act as virtual sources, near-offset traces can be generated by crosscorrelating traces from nearby receivers, and zero-offset traces by autocorrelating a single trace. These crosscorrelated traces contain many correlations besides those

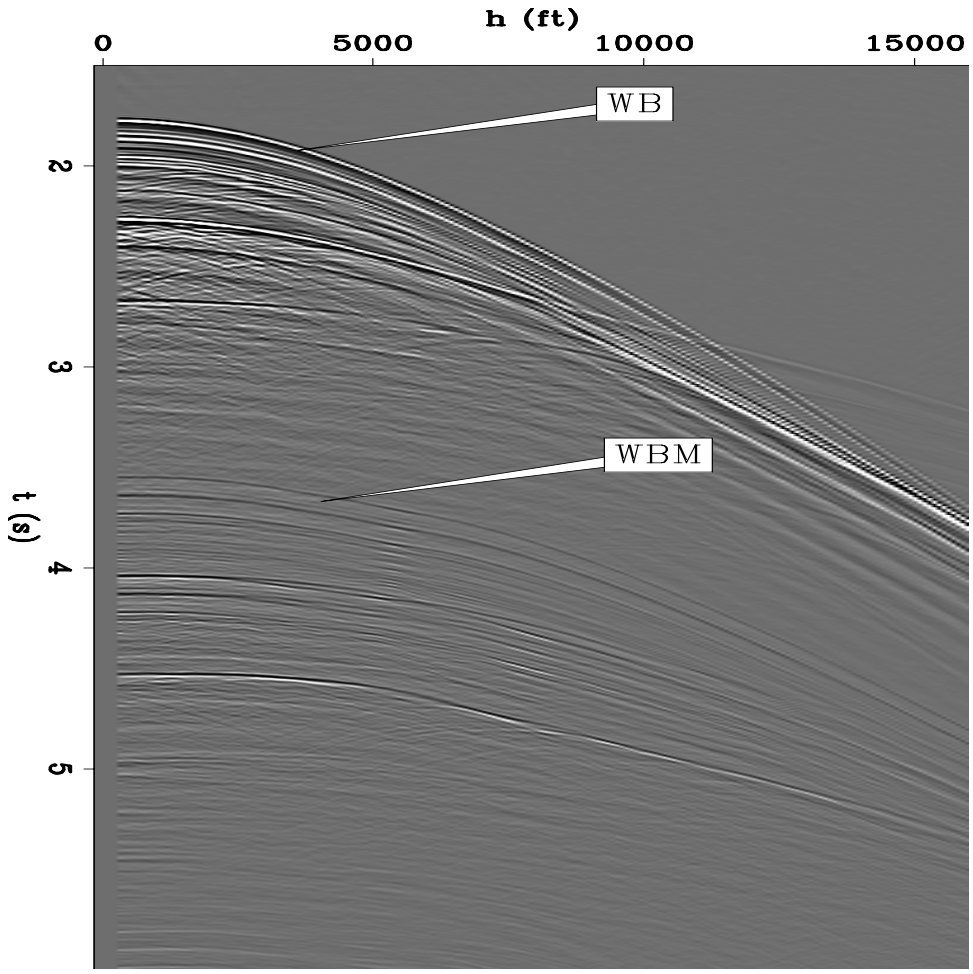

Figure 4.1: A single shot profile from a 2D Gulf of Mexico seismic survey. The nearest offset is at 330 ft. for a near-offset gap of four traces. **ER** Pseudo/. realshotann

between primaries and free-surface multiples or between free-surface multiples and higher-order free-surface multiples, so that the pseudo-primary signal-to-noise ratio for a single crosscorrelation is poor. This can be improved by summing crosscorrelations of the same receiver location pair for many source positions, so the desired pseudo-primary correlations sum while the other correlations interfere destructively. The resulting pseudo-primaries are data that honor the kinematics of the recorded data, but also contain noise, have a different amplitude scale, and have a squared wavelet compared to the recorded data.

Simply substituting these pseudo-primaries for the missing data does not produce an adequate result, as the data contain a squared wavelet, a high level of noise, and spurious events that do not correspond to primary reflections. Instead, here I use the pseudo-primary data as training data for a nonstationary prediction-error filter; whereas the data are inadequate as an interpolation result, they are quite acceptable as training data. The PEF estimation process is relatively insensitive to the phase, amplitude, and squared-wavelet issues that make direct substitution undesirable. This PEF is then used for the interpolation step to fill in the inline near-offset gap.

This approach of using the pseudo-primaries as training data for a PEF can be performed in the time and offset (t-h) domain, the time, offset and source (t-h-s) domain or the frequency, offset and source (f-h-s) domain. In the t-h domain, I interpolate each shot record independently, using a separate non-stationary 2D t-h PEF generated from the corresponding pseudo-primary shot. In the f-h-s domain, the process is performed on overlapping time windows of shot records NMO-corrected using water velocity. A non-stationary 2D PEF is estimated on each source-offset-frequency slice of each time window of the pseudo-primaries, which is then used to interpolate the near-offset gap of the data for the same frequency slice and time window of the original data. The interpolated result is then inverse Fourier transformed back to the time domain, the time windows reassembled, and the NMO correction removed.

GENERATING PSEUDO-PRIMARIES

Multiple reflections are typically viewed as undesired noise to be removed from reflection seismic data. One way to do this is first to predict the multiples and then subtract them from the data. Free-surface multiples can be predicted by autoconvolving data so that the convolution of primaries with themselves creates events at arrival times that are the same as those of multiple reflections with a single bounce point at the free surface. This approach creates free-surface multiple reflections with correct kinematics without the need for any additional subsurface information except the recorded data (Riley and Claerbout, 1976; Reiter et al., 1991; Verschuur et al., 1992).

Autocorrelation can be used to extract synthetic active source data from incoming waves that reflect at the free surface and return within the recording array (Claerbout, 1968; Cole, 1995; Schuster, 2001; Artman, 2007). This has traditionally been though of in a passive context, where random noise is assumed to be arriving from all locations. The reflecting waves from active-source experiments can be treated in the same manner, where the primary reflections correlate with the free-surface multiples (Reiter et al., 1991; Berkhout and Verschuur, 1994, 2003; Shan, 2003). Autocorrelating data, d, (implemented as multiplication of complex-conjugates in the ω domain) for two receiver points, r_1 and r_2, both for a single shot, s, gives the pseudo-primaries, p,

$$p(s, r_1, r_2, \omega) = d(s, r_1, \omega)\bar{d}(s, r_2, \omega). \tag{4.1}$$

One of the receiver coordinates, r_1, becomes the virtual source location, while the other receiver coordinate, r_2, remains the receiver location, or vice-versa, and the $\bar{d}$ denotes the complex-conjugate of d. This is similar to 2D surface-related multiple prediction, where instead of cross-correlation the data are convolved with itself, so that the convolution of primaries with primaries produces surface-related multiples (Verschuur et al., 1992).

Figure 4.2a shows an example of a fully-sampled split-spread shot from the Sigsbee2B data set, while Figure 4.2b shows a slice of the pseudo-primary output of

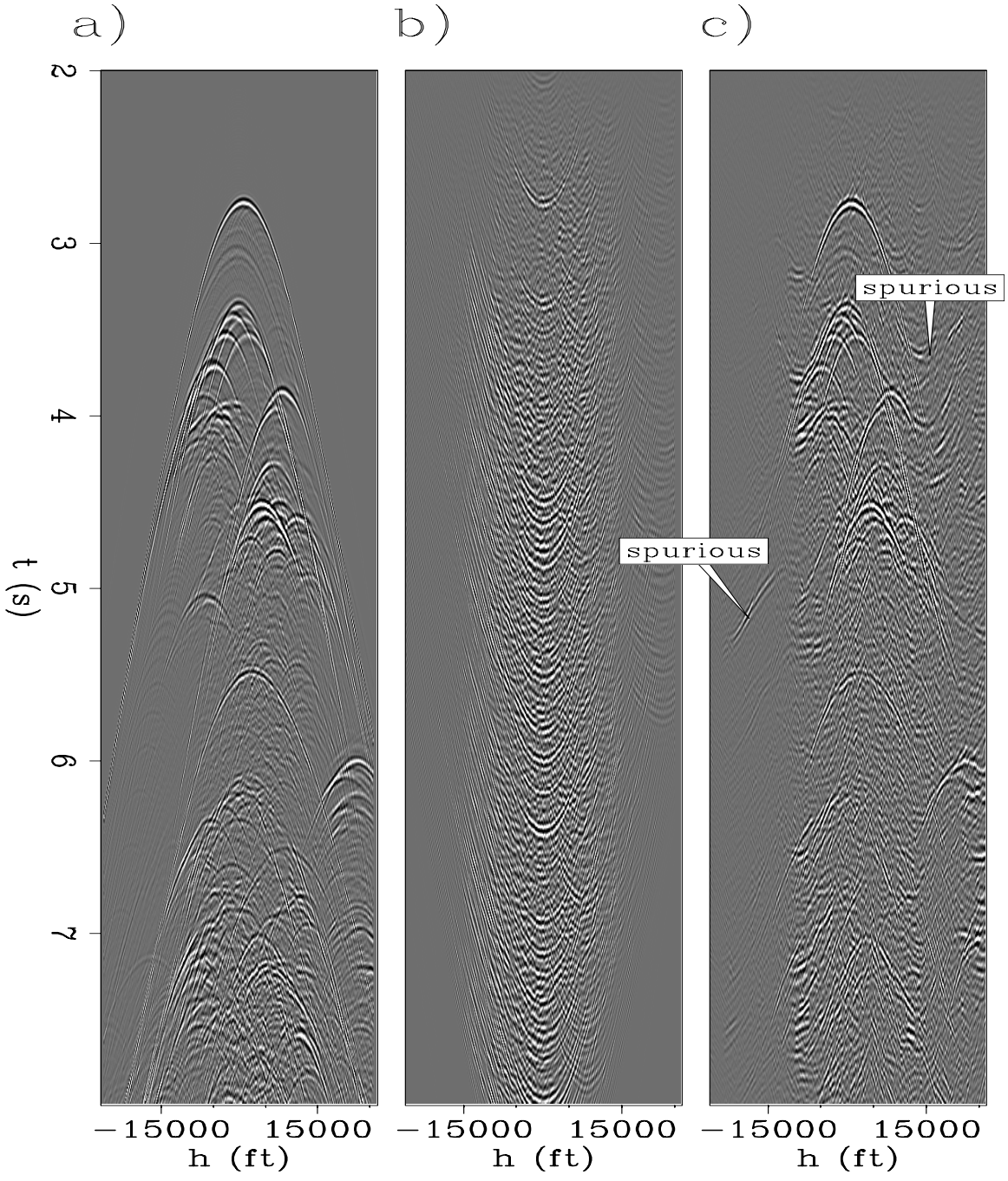

Figure 4.2: Crosscorrelation of a single shot. (a): Original fully-sampled split-spread shot; (b): The same shot recreated from crosscorrelating the traces in (a) using equation 4.1. (c): using equation 4.2 correlations are made for 496 shots and are then summed. The quality of the pseudo-primaries is poor for a single shot, but improves after the summation of many shots. All data are scaled by t. **ER** Pseudo/. sigsbee1shotann

autocorrelating the zero-offset trace with all of the other traces within that single shot. Clearly, the original shot and the pseudo-primaries generated from the autocorrelation are different. The water-bottom reflection at zero offset is present as are the first group of diffractors, but they are not present at the more distant offsets. The many other correlations between events produce undesired noise in the output. Primaries can correlate with other primaries, multiples with multiples, and noise with other noise or signal. Much of these undesired correlations, however, vary as a function of source position, s, so when the correlations for the same receiver pair r_1 and r_2 are performed for many different source locations, s, and are then summed, the unwanted events destructively interfere, while the correct pseudo-primaries constructively interfere.

$$p(r_1, r_2, \omega) = \sum_s d(s, r_1, \omega)\bar{d}(s, r_2, \omega). \tag{4.2}$$

As shown in Figure 4.2c, where 496 sources were used, this summing over multiple source positions thus can greatly improve the pseudo-primary signal. The range of usable offsets is greatly improved, as is the signal-to-noise ratio. The pseudo-primaries now look more similar in character to the original data.

Pseudo-primaries generated from recorded multiples are interesting in part because they have different illumination than do the recorded primaries. Figure 4.3a illustrates a desired near-offset primary ray-path that is not recorded because of the gap between the source and the nearest receiver. In Figure 4.3b, the source is positioned such that the raypath first travels from the source to the water-bottom and back up to the water surface within the recording array. This ray then reflects back into the subsurface and eventually returns into the recording array at another receiver. This four-segment raypath is a free-surface multiple, which can be reconsidered as two distinct events: the first a recorded primary event from the source s to the first receiver r_1 and the second leg as another primary from the virtual source r_1 to another receiver r_2. Crosscorrelating the multiple recorded at r_2 with the primary recorded at r_1 produces a pseudo-primary trace with a virtual source at r_1 and a receiver at r_2. Comparing Figures 4.3a and 4.3b, we see that we can transform multiples that we

record into pseudo-primaries with virtual source locations where we did not originally record data. This is most useful at unrecorded near offsets.

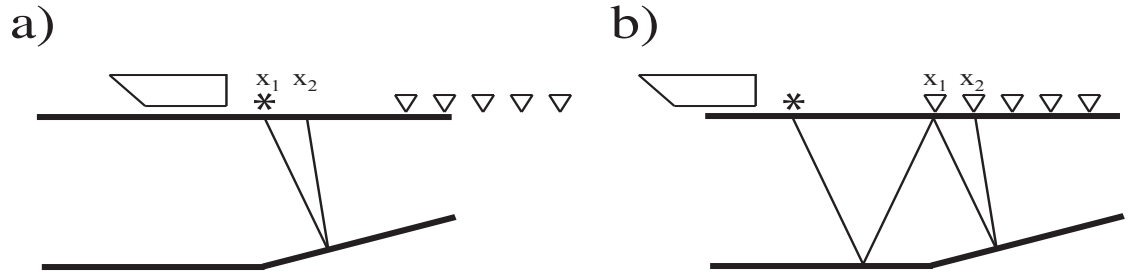

Figure 4.3: Raypaths of a primary and multiple reflection. (a): an unrecorded primary reflection that hits the surface at near offset. (b): a recorded multiple that first reflects at the surface within the recording array, then returns within the recording array. **NR** Pseudo/. nearoffsetcartoon

The generated pseudo-primaries differ from the recorded data in several ways. First, the crosscorrelation of the data squares the amplitude spectrum of the pseudo-primaries, meaning that the wavelet of the pseudo-primaries will become zero-phase, different than that of the input data. Second, the amplitudes of the pseudo-primaries will differ from the amplitudes of the actual primaries, as the pseudo-primaries are a correlation and summation of different events. This amplitude difference will be both on a global scale difference between the pseudo-primaries and the primaries, as well as different relative amplitudes within each data set. Third, spurious events from other correlations may still exist in the data as the number of sources is limited.

Because of these differences between the pseudo-primaries and primaries, a direct substitution of the created near-offset pseudo-primaries is not adequate. The pseudo-primaries, however, can be used as training data for a nonstationary prediction-error filter, as from Chapter 2 we know that the PEF is insensitive to the phase of the training data. This PEF is then used to interpolate the missing near offsets, so that the negative aspects of the pseudoprimaries, such as the incorrect wavelet and extra noise, are ignored, while the positive aspects of the pseudo-primaries, contained in the autocorrelation of these data, are used.

INTERPOLATING WITH NONSTATIONARY FILTERS IN T-X OR IN F-X

Now that an adequate training data set has been found, we can use the interpolation method as described in Chapter 2. Recall that a nonstationary prediction-error filter is first estimated from fully-sampled *training* data and is then used to interpolate *missing* data. The training data required by this method should have the same local autocorrelation as that of the desired output interpolated data. Pseudo-primaries, having roughly the same dip as the missing data, serve as training data for a PEF. Some of the problems with the pseudo-primary data, such as the different phase and amplitude, do not influence the PEF.

Reviewing equation 2.21,

$$\begin{aligned} & \min_{\mathbf{f}_{\text{ns}}} \|\mathbf{r}_{\text{d}}\|^2 + \epsilon^2 \|\mathbf{r}_{\text{f}}\|^2, \text{where} \\ \mathbf{r}_{\text{d}} &= \mathbf{D}_{\text{ns}}\mathbf{K}_{\text{ns}}\mathbf{f}_{\text{ns}} + \mathbf{d} \\ \mathbf{r}_{\text{f}} &= \mathbf{R}\mathbf{f}_{\text{ns}} \end{aligned} \quad . \tag{4.3}$$

Here, the unknown nonstationary PEF coefficients, $\mathbf{f}_{\text{ns}}$, are estimated from the pseudo-primary data, $\mathbf{d}$, a convolutional matrix, $\mathbf{D}_{\text{ns}}$, that is a function of $\mathbf{d}$, and a regularization operator, $\mathbf{R}$, that applies a Laplacian filter across the spatial axes of the PEF coefficients. This system of equations is solved to estimate a multi-dimensional, nonstationary prediction-error filter from a set of fully-sampled pseudo-primaries, such as was generated in the previous section.

Once this filter has been estimated from the pseudo-primary data, the filter is used in a second least-squares problem. In this problem, we estimate an interpolated output, $\mathbf{m}$, composed of missing data, $\mathbf{m}_{\text{unknown}}$, and known data, $\mathbf{m}_{\text{unknown}}$, that when convolved with the nonstationary PEF produces a minimized output, as in equation 2.29,

$$\mathbf{r}_{\text{m}} = \mathbf{F}_{\text{ns}}\mathbf{m}_{\text{unknown}} + \mathbf{F}_{\text{ns}}\mathbf{m}_{\text{known}}. \tag{4.4}$$

Here the known nonstationary PEF convolution matrix, $\mathbf{F}_{\text{ns}}$, derived from $\mathbf{f}_{\text{ns}}$ obtained

in the previous step, is multiplied with both the known and unknown data values, $\mathbf{m}_{\text{known}}$ and $\mathbf{m}_{\text{unknown}}$, to create a known quantity and a term with the unknown values to be interpolated. These terms are summed to form the residual $\mathbf{r}_{\text{m}}$. We minimize the L_2 norm of this residual to estimate the interpolated data values.

Interpolation in time and space

The output pseudo-primaries from equation 4.2 are in frequency, source position, and receiver position. We can reorganize this to time, offset, and source position, and then estimate a nonstationary PEF, either shot-by-shot by estimating 2D PEFs in time and offset solving equation 4.3 for each shot, or on the entire data set with a single 3D PEF in time, offset, and source solving a much larger version of equation 4.3. This PEF, or series of PEFs, is then used to fill in the missing near-offset gap using equation 4.4 either once or for each shot.

Interpolation in frequency and space

Interpolation in the frequency domain requires a different approach than in the time domain. Spitz (1991) shows that 2D plane waves can be predicted for a single frequency by using a 1D spatial prediction filter, because a single plane wave at each frequency appears as a complex sinusoid in space. The wavenumber of this sinusoid increases as a function of frequency, so each frequency requires a unique 1D PEF. Data containing a combination of plane waves appear as a summation of complex sinusoids at each frequency, which can still be predicted by a reasonably short 1D PEF. In three dimensions, this filter would be a 2D filter, in four dimensions a 3D filter, and so on, making the frequency-based approach a series of smaller problems whereas the same region in time would be solved as a single larger problem.

Nonstationarity, discussed in Chapter 2, is addressed in a slightly different manner in f-x. Along the spatial axes, a nonstationary PEF can be used to capture slopes that change as a function of position. Since a Fourier transform is performed on the

time axis to convert both the training and interpolated data to the frequency domain, we implicitly assume that the slopes of the plane waves do not vary as a function of time.

I address this problem of time nonstationarity by breaking up the problem into overlapping time windows that we assume to be stationary. First, perform a water-velocity normal move-out correction on both the pseudo-primaries and the original recorded data to (roughly) flatten them, and then break up the data into overlapping windows along the time axis. I perform the NMO to reduce the amount of energy crossing the boundaries between patches. From there, each time window of both the pseudo-primaries and the recorded data is Fourier transformed along the time axis, so that the data are sorted into source, offset, frequency, and time window. A unique 2D nonstationary complex-valued PEF is estimated in source and offset on each frequency of each time window of the pseudo-primary data by solving equation 4.3. This nonstationary source-offset (h-s) PEF is then applied to fill the missing data in offset on the corresponding time window and frequency of the recorded data by solving equation 4.4, and this series of problems is repeated for each frequency of each time window.

After the data are interpolated, they are first inverse-Fourier transformed back to time, then the time-windows are reassembled with appropriate weighting in overlapping regions, and finally are inverse NMO-corrected to return the data to their original form of time, source, and offset.

SIGSBEE DATA EXAMPLE

In order to test the effectiveness of this pseudo-primary-based method, I use a synthetic example where we already know the answer. To create a challenging test case, I take a split-spread version of the Sigsbee2B synthetic data set, with a zero-offset section and a shot gather shown in Figures 4.4a and 4.4b, respectively, and remove the nearest 2100 ft of offset on either side, for a total of 4200 ft of missing offset, or a gap of 29 traces at the 150 ft sampling in offset, shown in Figure 4.4c. While this gap

is overly large for a single boat, a two-boat "undershooting" of an offshore platform could have gaps this large.

Let us first examine the cross-correlated pseudo-primaries prior to the summation in equation 4.2 for a single output trace from one receiver pair for all shots, both using the fully sampled input data, shown in Figure 4.5b, and using the data missing the near offsets, shown in Figure 4.5c. This would correspond to $p(s, r_1 = 41000, r_2 = 41000, t)$ in equation 4.1. I refer to these images as pseudo-primary-contribution gathers, as they fulfill a role similar to multiple-contribution gathers in the SRME algorithm (Dragoset and Jericevic, 1998). The summations of these gathers, the predicted pseudo-primary traces, are shown in Figure 4.5a, plotted alongside the original trace. We see that the traces in Figure 4.5a look quite similar, especially the two pseudo-primary traces, so the missing sources at near-offsets did not significantly detract from the result.

Moving up from the single output trace in Figure 4.5, we now look at an entire zero-offset section in Figure 4.6. Figure 4.6a shows the original zero-offset section while 4.6b shows pseudo-primaries generated using fully-sampled data, and 4.6c shows pseudo-primaries generated from data with missing near offsets.

Note four important differences between Figures 4.6a and 4.6c. First, the pseudo-primaries have different illumination than do the true primaries, so the relative amplitudes within each image differ. The amplitude of the water-bottom reflection in the pseudo-primary image with limited offsets in Figure 4.6c is more variable than is that in the original image in Figure 4.6a, or even that in the pseudo-primaries generated with all of the offsets in Figure 4.6b. Also, the subtle reflections below the water bottom on the left side of the image at 4-5 s are much less pronounced when the input offsets to the pseudo-primary generation are limited. Second, the pseudo-primaries compared to the original data exhibit cross-talk. The cross-talk is composed of both coherent events such as those above the water bottom and water-bottom multiple in Figure 4.6c, and more random noise. This cross-talk increases only slightly when the offsets are limited in the input to the pseudo-primary generation. Third, the wavelet of the pseudo-primary data differs from that in the original data as a result of the

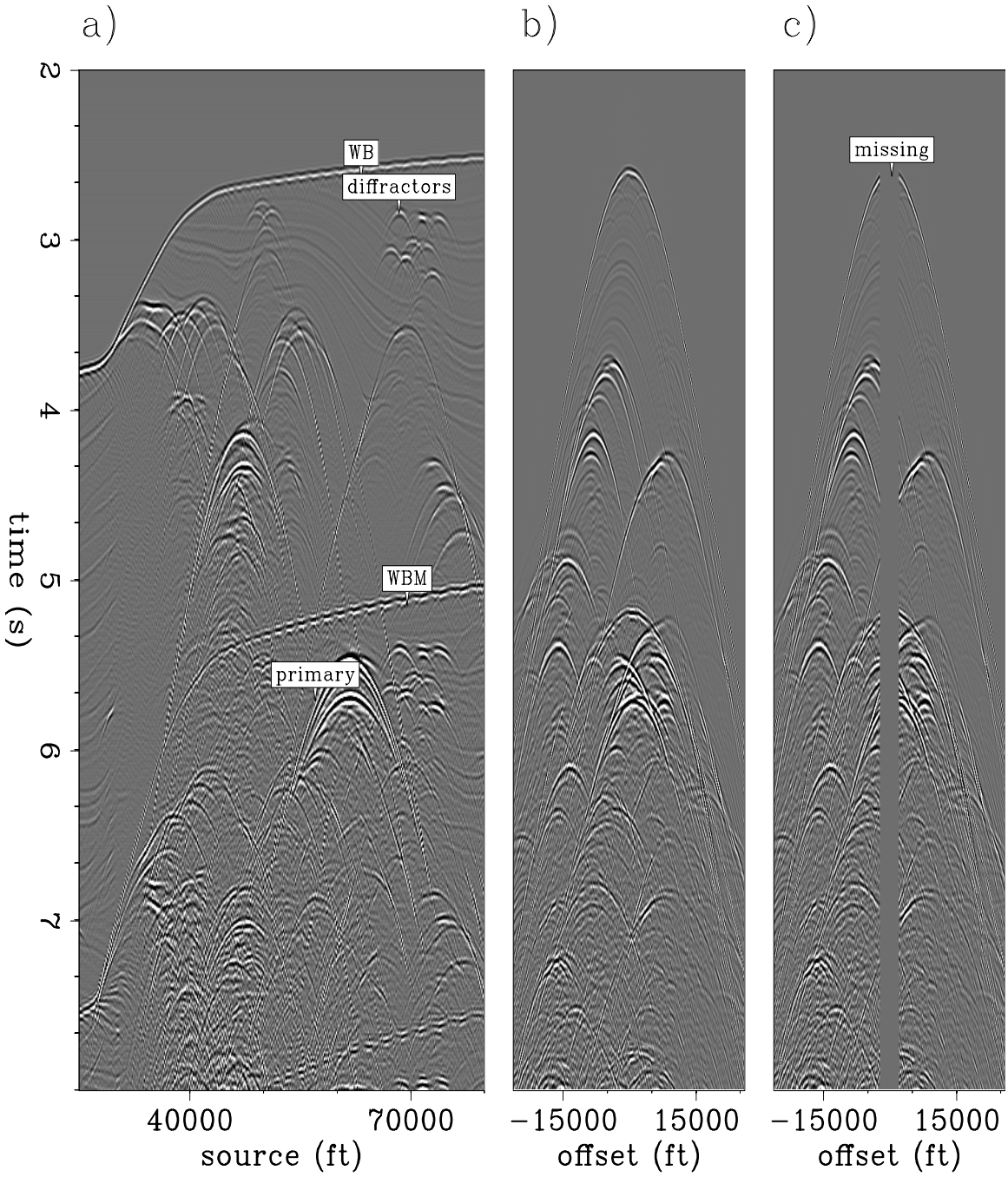

Figure 4.4: Original Sigsbee data. (a): zero-offset section. (b): one shot. (c): resampled shot. The near 4200ft or 29 traces of offset were excluded. All figures have amplitude scaled by $t^{0.8}$. **ER** Pseudo/. splitspreadann

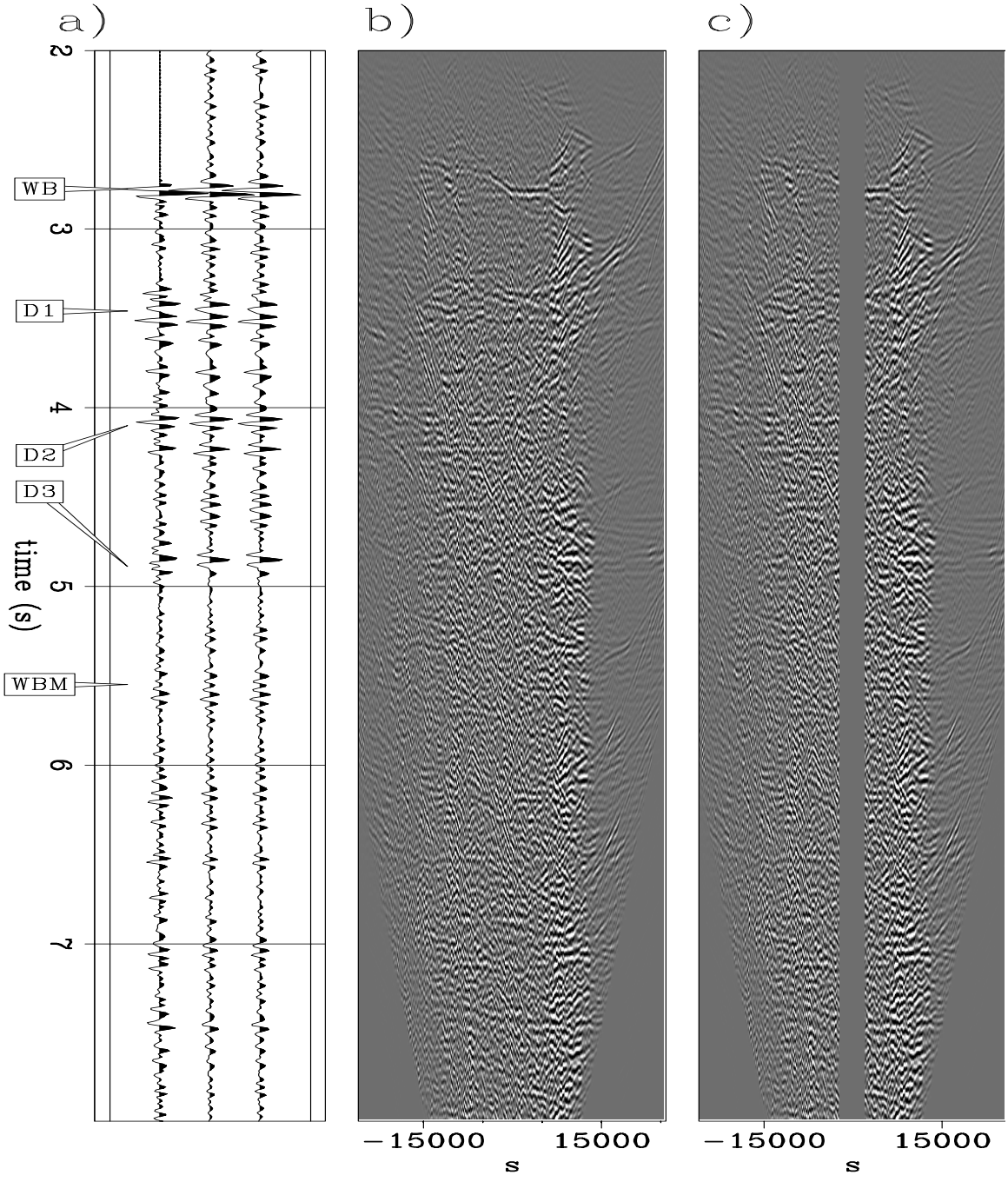

Figure 4.5: Generation of a pseudo-primary trace for $p(s, r_1 = 41000, r_2 = 41000, t)$. (a): comparison of traces of (left to right) original data, pseudo-primary data generated from fully-sampled input data, and pseudo-primary data generated from data with missing near offsets. (b): A pseudo-primary contribution gather, where the horizontal axis is shot location s. (c): The same pseudo-primary contribution gather as (b), but with the missing near offsets. The images are scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. pseudocontribann

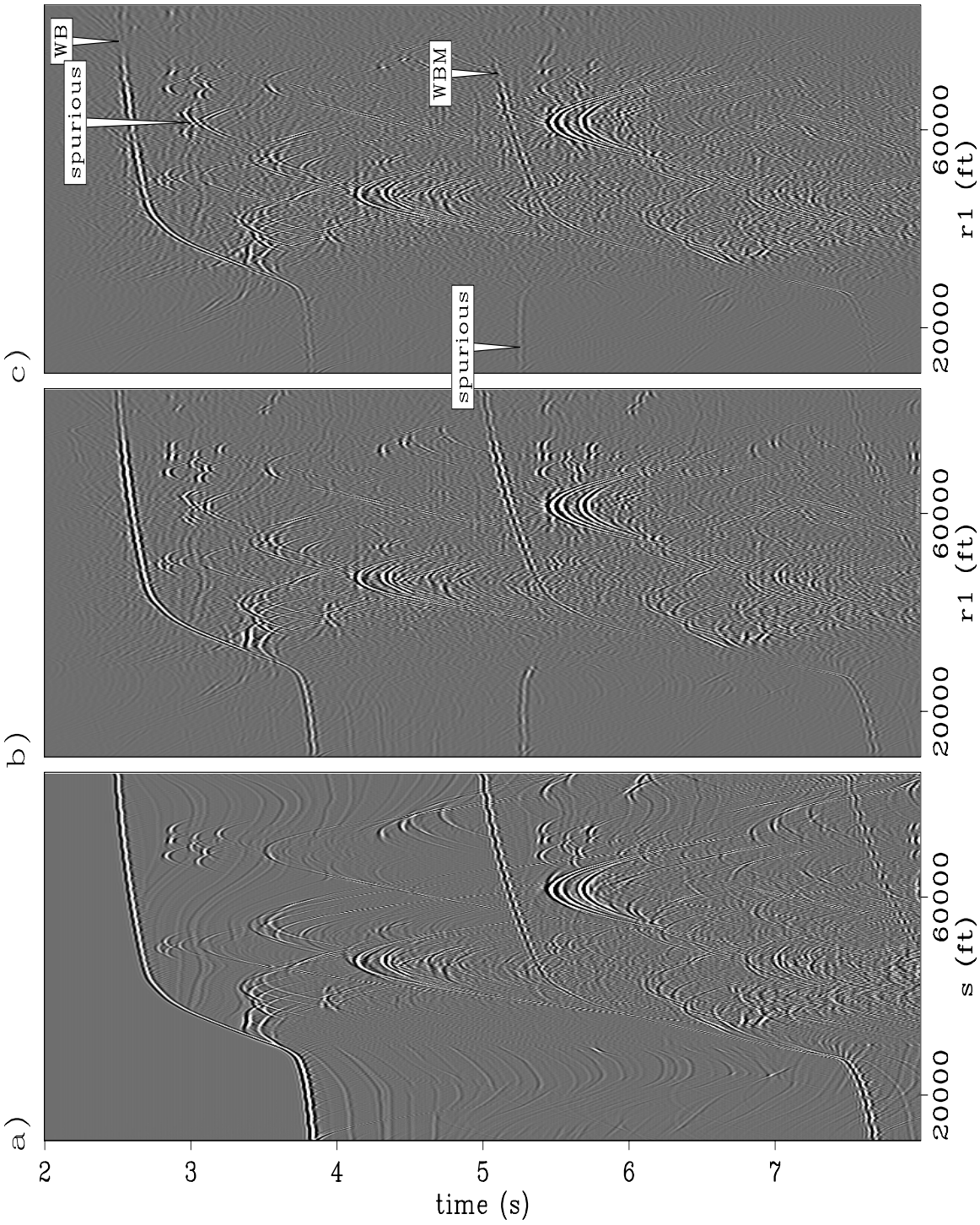

Figure 4.6: A comparison of zero-offset sections: (a) original data, (b) pseudo-primaries generated using all offsets, (c) pseudo-primaries generated using all offsets other than the missing near offsets. The quality of the pseudo-primaries degrades slightly with missing near offsets. The images are scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. pseudocontribsliceann

cross-correlation of the primary wavelet and the multiple wavelet. Finally, the amplitude scale of the pseudo-primary images (4.6b-c) is roughly a factor of 20 higher than in the original data.

Figure 4.7 shows the nearest 10000 ft of offset of the pseudo-primaries generated from all of the split-spread shots from Figure 4.4c that are missing the near offsets. The common-offset section on the left panel is at zero offset. Looking at the shot gather on the right panel, we see that the water-bottom reflection is the strongest event at early times, but a considerable amount of cross-talk is present before the water-bottom reflection. The time slice shown on the top panel of Figure 4.7 shows that in addition to the desired reflections and diffractions extracted from the multiple reflections, there are also strong spurious events that are not easily identifiable as cross-talk; in particular, note the event at roughly 60000 ft and near offsets.

The most straightforward way to use these pseudo-primaries to interpolate the missing near offsets would be to replace the missing traces with the pseudo-primaries, shown in Figure 4.8. The pseudo-primaries were scaled by a factor of 0.05 to match the mean amplitude of the sampled data. The cross-talk and squaring of the wavelet are both obvious in this Figure 4.8, making the region of interpolated data is easily distinguishable, as do the spurious events, and the extra slope in the more complex areas. As we see next, we can improve greatly on this result by using the pseudo-primaries as training data for a prediction-error filter.

Interpolation of Sigsbee in time and offset

A single output shot record of the pseudo-primaries ($p(r_1, r_2, t)$) generated in Figure 4.7 is used as the training data ($\mathbf{d}$) in equation 4.3 to generate a nonstationary PEF, $\mathbf{f}_{\text{ns}}$. The regularization operator in equation 4.3, $\mathbf{R}_{\text{ns}}$, is a two-dimensional Laplacian that operates over time and offset for each local filter lag. Once estimated, the PEF is then used to interpolate the missing near offsets of the sampled data, $\mathbf{m}_{\text{unknown}}$, using equation 4.4. This process is repeated for all of the shots of both the pseudo-primary data and the original sampled data.

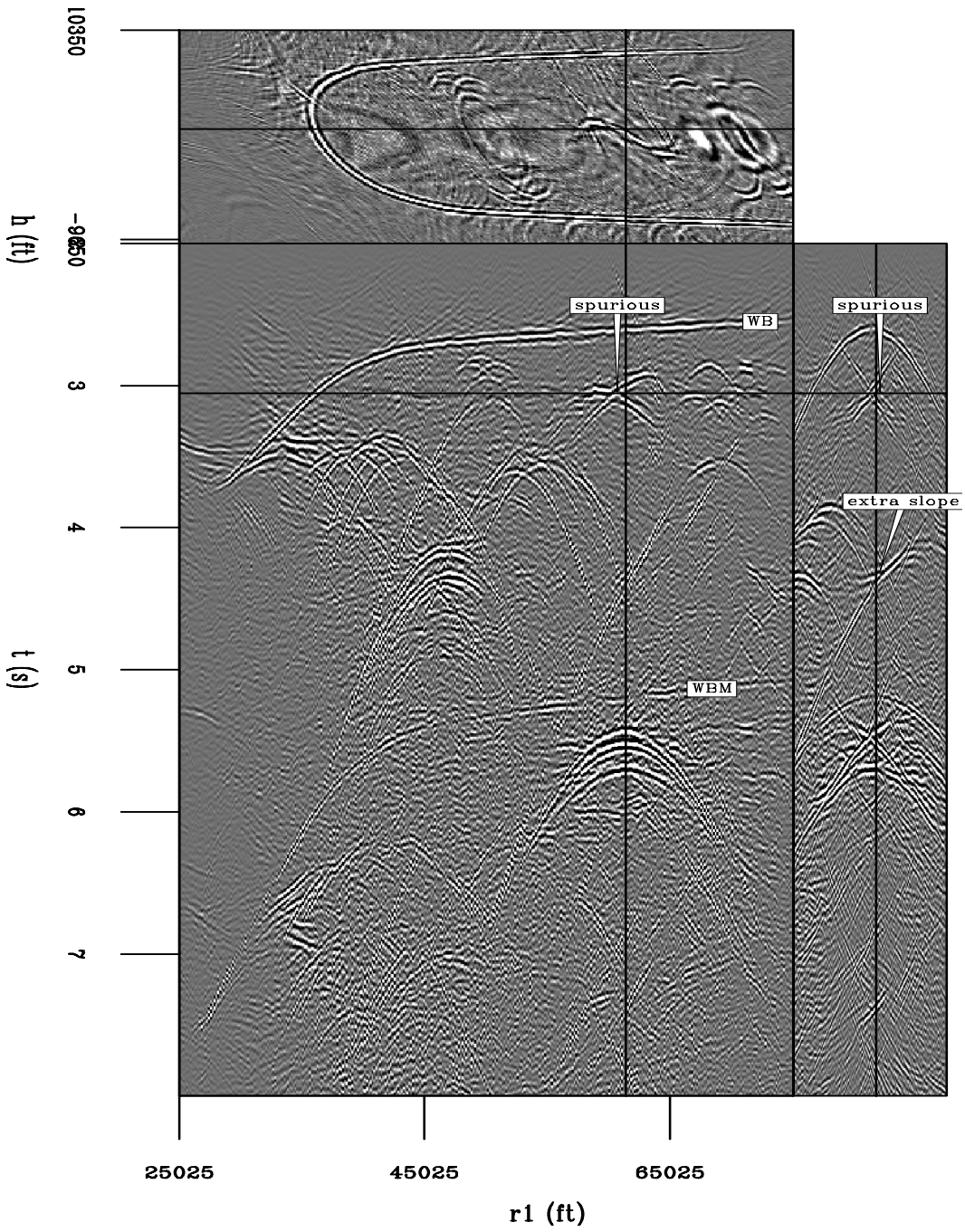

Figure 4.7: The near 10000 ft of offsets of pseudo-primaries generated from input data missing the 2000 ft of near offsets. The front panel is a constant-offset section, the right panel is a single shot, and the top panel a time slice. The image is scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. pseudoann

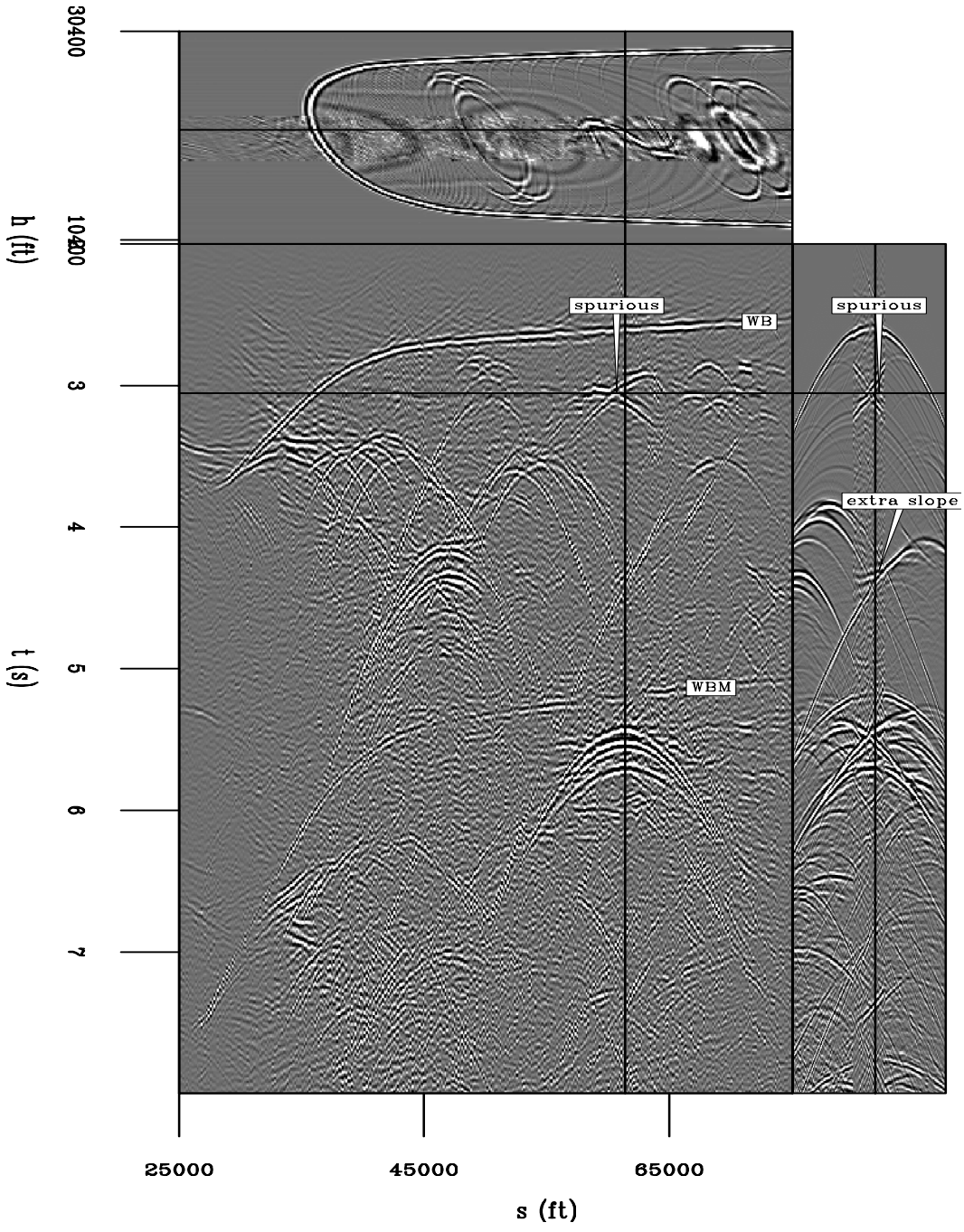

Figure 4.8: Original data missing the near offsets, with pseudo-primaries spliced into the locations where traces were missing. Note three points of interest, a spurious event caused by the correlation of primaries with other primaries, seen in both the constant-offset section and the shot, and an area with crossing events and noise. The image is scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. pseudocutann

Figure 4.9 was generated by interpolating the near-offset gap using a t-h nonstationary PEF estimated on each shot record of the pseudo-primary data. Each 2D PEF has ten points in time and six points in offset, and varies every four points on each axis for a total of 76000 free coefficients for each shot. I applied 400 iterations of the conjugate-direction solver to solve both the filter-estimation and interpolation problems.

There are several interesting things to see in the t-h interpolated result in Figure 4.9. First, no cross-talk is present in the interpolation, such as a spurious event. This is one of the benefits of using the t-h PEF-based approach, since low amplitude at the edges of the known data results in no large residuals in equation 4.4, regardless of the crosstalk present in the training data for the PEF. Second, the events present in the original data in a single shot shown in the right panel have been interpolated, in most cases with good results, although the quality of the interpolation degrades deeper in the section. This can largely be traced to the quality of the input pseudo-primaries. It also appears that problems with the relative amplitudes in the pseudo-primaries are amplified in this result. Third, the common-offset section shows significant differences from shot to shot. This is because a 2D PEF was used and each shot was interpolated independently. This is different when a 3D PEF in time, offset *and shot* is used, where correlations between shots are be used and the inconsistencies between shots penalized, discussed next. Finally, the wavelet issues in the pseudo-primaries appear to be mostly removed in the t-h interpolated result.

While the t-h approach gives a reasonable result when viewing a single shot, the extremely choppy constant-offset section shows one limitation of the t-h approach. I next use a 3D nonstationary t-h-s PEF that is $10 \times 5 \times 5$ elements and varies every 10 elements on the time axis, 3 elements on the offset axis, and 4 elements on the source axis. I solve for this PEF using 100 iterations of a conjugate-direction solver. Once estimated, I use this PEF, with 121 million coefficients, to interpolate the missing data, using 200 iterations of a conjugate-direction solver on equation 4.4, with a starting guess for the unknown data created from the nearest recorded offset from the same midpoint NMO-corrected to the missing offset.

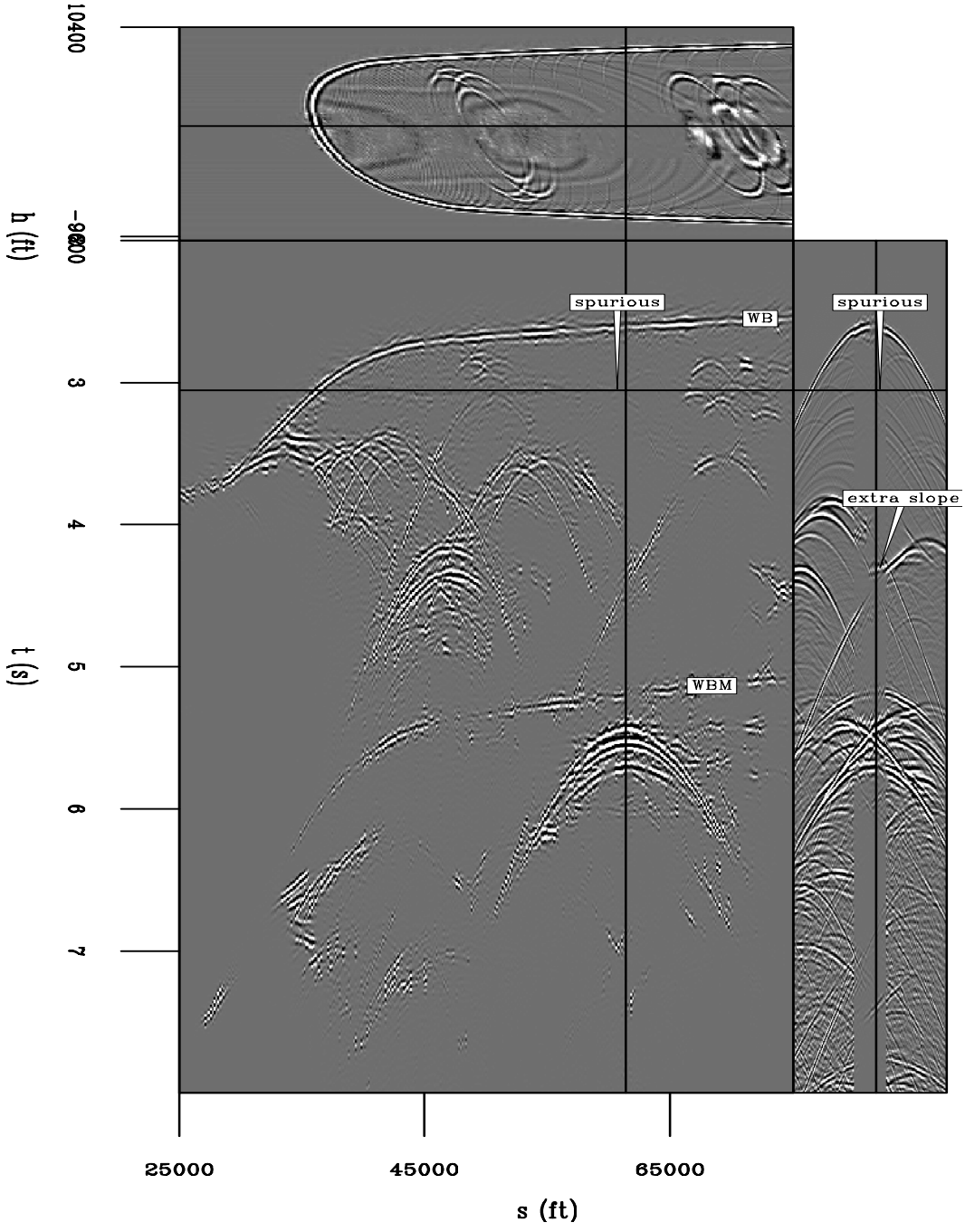

Figure 4.9: Interpolation with pseudo-primaries and a t-h PEF. The front panel is an interpolated constant-offset section, the right panel is a single shot record, and the top panel a time slice. The spurious event is not present, but the region with conflicting slopes is only partially interpolated, with an incorrect slope present. The image is scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. txinterpann

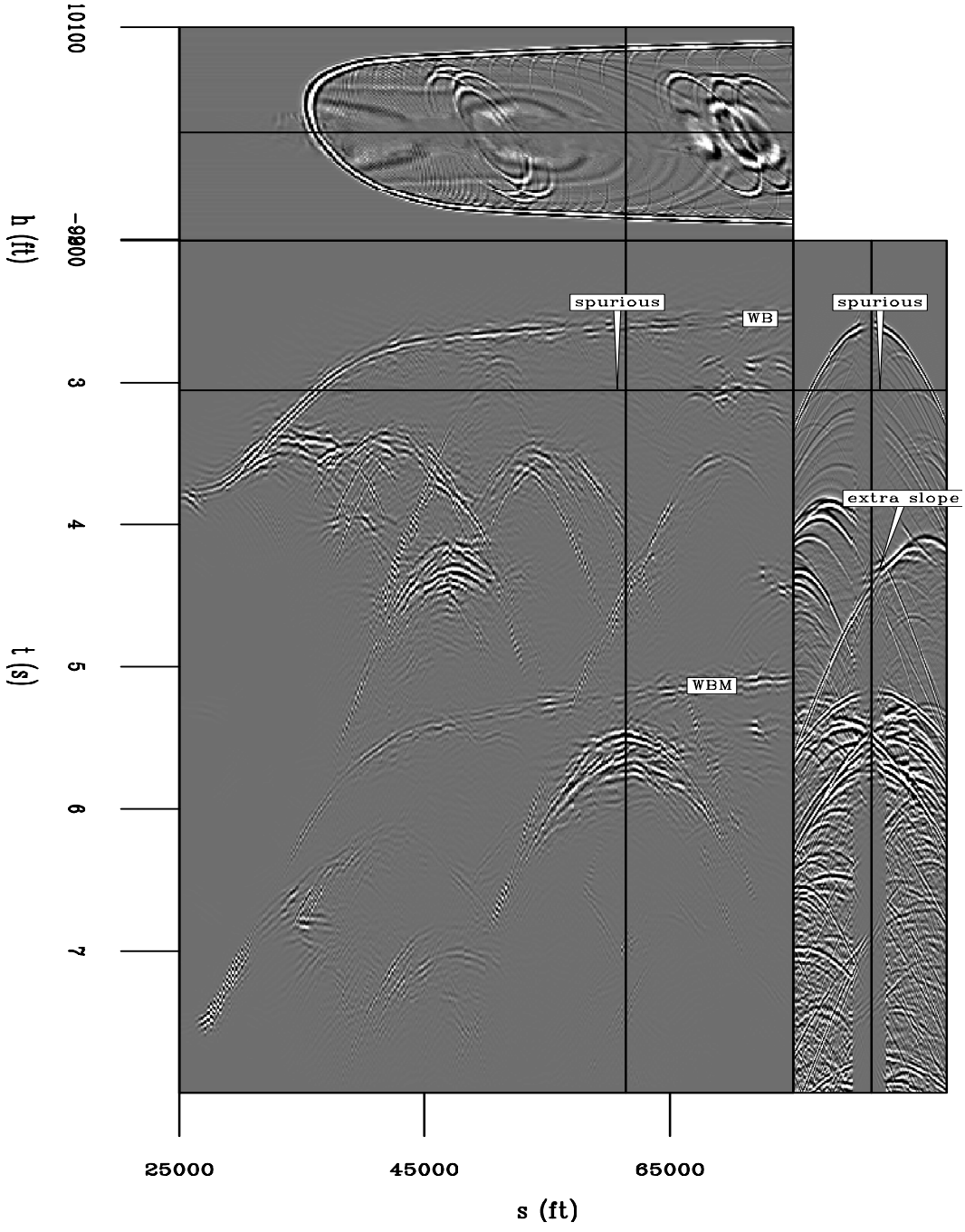

Figure 4.10: Interpolation with pseudo-primaries and a 3D t-h-s PEF. The front panel is an interpolated constant-offset section, the right panel is a single shot record, and the top panel a time slice. The spurious event is not present and the region with crossing slopes is better interpolated than in the 2D example. The water-bottom is more variable in amplitude, but the constant-offset section is more consistent from shot to shot. The image is scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. txyinterpann

This 3D result, shown in Figure 4.10, is in part an improvement over the 2D approach. The roughness of the 2D approach along the source axis is gone, with a smoother result that contains slightly more noise. The spurious event is still gone, and the the diffractions in the constant-offset section are more continuous, although the water-bottom amplitude is more variable. Next I show that using a 2D PEF on frequency slices provides most of the benefits of using a 3D *t*-*h*-*s* PEF, but with a much lower computational cost and memory requirement.

Interpolation of Sigsbee in frequency, offset, and shot

Figure 4.11 shows the result of near-offset interpolation using 2D PEFs in the *f*-*h*-*s* domain. Here the training and sampled data were first NMO-corrected at water velocity, then were broken into overlapping windows along the time axis. Both sets of these time windows were then Fourier transformed to frequency. Each frequency slice of each window is treated independently, where we estimate a nonstationary PEF on a frequency slice of the pseudo-primaries, solving equation 4.3 with the frequency slice of the pseudo-primaries serving as training data ($\mathbf{d}$) to generate a nonstationary complex-valued PEF $\mathbf{f}_{\text{ns}}$. This PEF is then used in equation 4.4 as $\mathbf{F}_{\text{ns}}$, where $\mathbf{m}_{\text{known}}$ is the corresponding frequency slice of the recorded data, and $\mathbf{m}_{\text{unknown}}$ are the missing near offsets of this frequency slice.

In Figure 4.11 the time axis (1500 samples long) of both the input sampled data and the pseudo-primaries were broken into 40 overlapping windows of 64 points each. Both sets of windows were then Fourier transformed to produce 2560 2D frequency slices. Each frequency slice is treated as a separate problem, with a 2D nonstationary PEF estimated on the pseudo-primary frequency slice; the 2D PEF was four samples long on both the offset and shot axes, and the filter varied every four points on each axis, for a total of 67456 filter coefficients for each 496-shot by 136-offset frequency slice, estimated with 100 iterations of a conjugate-direction solver. The data were then interpolated using 200 iterations of a conjugate-direction solver on equation 4.4, again with an initial model of the missing data an NMO-corrected copy of the trace

from the same midpoint and the nearest offset.

The frequency-domain interpolation in Figure 4.11 differs in several respects from the t-h domain interpolation in Figure 4.9. First, the f-h-s result still has some crosstalk present, while the t-h and t-h-s results have almost no crosstalk from spurious events. This is most visible prior to the water-bottom reflection. I believe this is in part due to the stationarity assumption within each time window. However, the large spurious event at a (1) and (2) is not present in the result in either case. Second, the f-h-s result shows less shot-to-shot variation than does the t-h result, and less amplitude variability than the t-h-s result. This is because in the t-h result each shot is interpolated separately as an independent problem, while the f-h-s interpolation estimates a PEF that spans both the offset and source axes, minimizing this jitter. Third, there is some ringing present in the t-h result that is not present Finally, the f-h-s result appears to contain more detail in the result than does the t-h result. In particular, events below the water-bottom in the shot that were not interpolated in the t-h case, in the right panel of Figure 4.9, were interpolated in the f-h-s case on the right panel of Figure 4.11.

These results can also be viewed after further processing. Figure 4.12a shows the zero-offset of a shot-profile migration of the fully-sampled data with the correct velocity. Figure 4.12b shows a multiple prediction based upon autoconvolution of the fully-sampled data, followed by the same migration as in 4.12a. The multiples kinematically match those in 4.12a, but the amplitudes are boosted in order to observe differences. The deformation of the multiples in the center of the image is due to the salt body present in the velocity model used in the migration. The same migrated multiple model is shown in 4.12c, but the multiples were generated with data missing the near 4000 ft of offsets, while 4.12d shows the migrated multiple model with these near offsets replaced by the f-h-s interpolation result in Figure 4.11. The missing near offsets in 4.12c significantly degrades the multiple model, as the water bottom multiple is uneven and the multiples near the edges of the salt body are different. The multiples generated from the f-h-s interpolation are much more similar to the fully-sampled multiples than to the multiples generated without the near offsets.

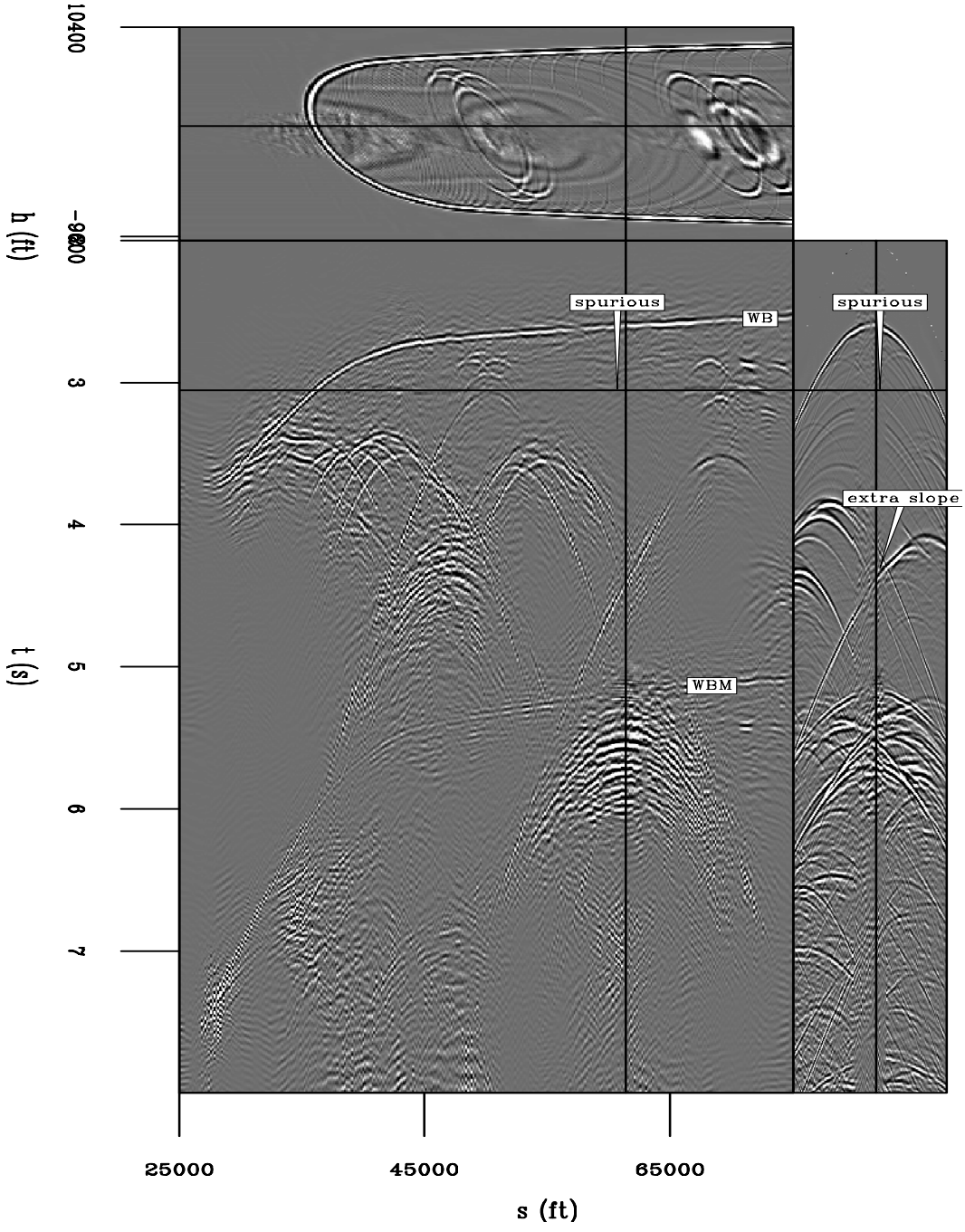

Figure 4.11: Interpolation with pseudo-primaries with 2D f-h-s PEFs. The result is more consistent from shot-to-shot, but still contains some cross-talk (around the water bottom) from the pseudo-primary data. The spurious event has been removed and the crossing slopes are believably interpolated. The image is scaled by $t^{0.8}$ for display purposes. **CR** Pseudo/. fxinterpann

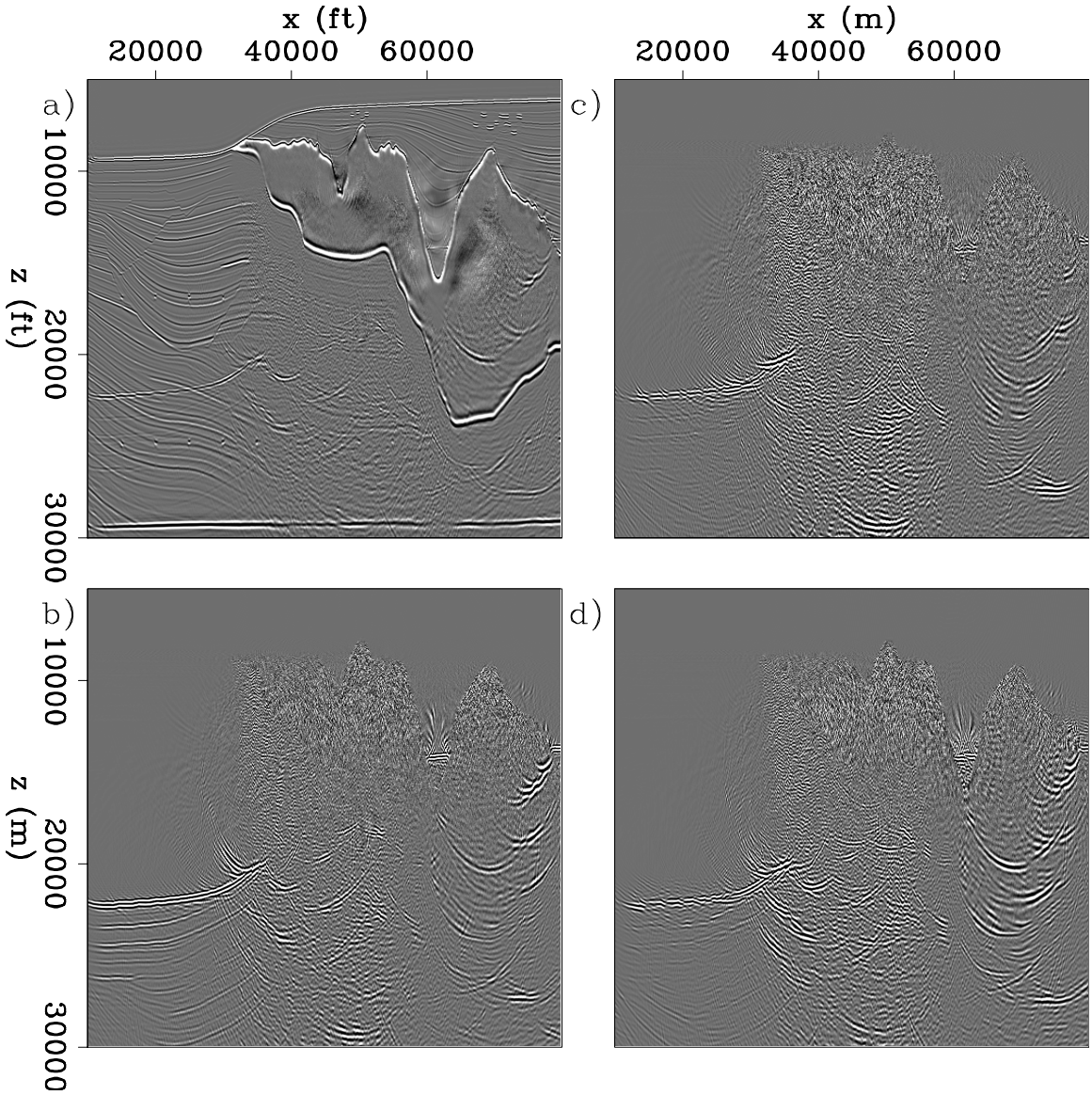

Figure 4.12: Migrated multiple prediction results for the f-h-s interpolation. (a): migrated fully-sampled data with multiples; (b): migrated multiple model generated with fully-sampled data; (c): migrated multiple model generated with no near offsets; (d) migrated multiple model generated with f-h-s interpolated near offsets. The interpolation restores much of the detail in the multiple model, especially under the edges of the salt body. **CR** Pseudo/. migmult

In addition to examining a migrated multiple model, we can also examine angle-domain common-image-gathers from both a migration of both the original data without the near offsets and a migration of the f-h-s interpolated data, as shown in Figure 4.13. Figure 4.13a is an angle gather for the water-bottom and a diffraction below the water-bottom, where the missing near offsets cause a dimming of the low angles, while Figure 4.13b is the angle gather from the interpolated data, where the low angles have comparable amplitude to the steeper angles.

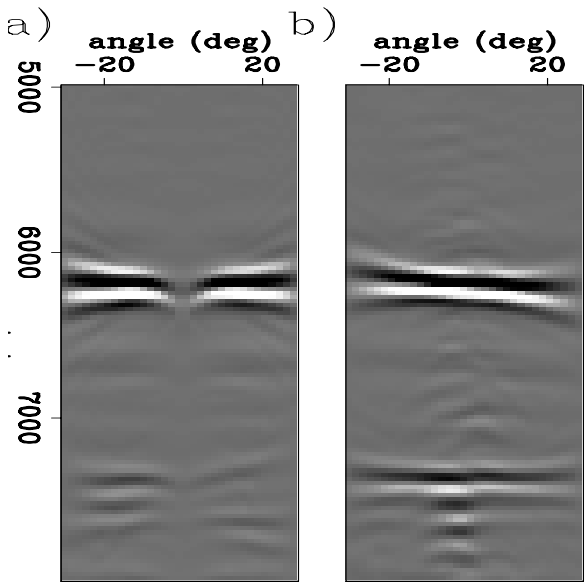

Figure 4.13: Angle-domain-common-image-gathers from migrations with and without near-offset f-h-s interpolation: (a) the original data migrated with missing near offsets; (b) the interpolated data after migration. The interpolation boosts the signal at flat angles for the water bottom and diffraction shown here. **CR** Pseudo/. migangle

This synthetic data example produced promising results for interpolation of a large gap using information only from the recorded data. The f-h-s approach slightly improved results and was still multiple times faster than the t-h or t-h-s interpolation methods. The synthetic data were noise-free, and only contained the desired primaries and multiples, with an idealized 2D geometry. Next, we examine how this method works on field data.

FIELD DATA EXAMPLE

Many have used the Mississippi Canyon multiples dataset from WesternGeco as a benchmark for testing multiple removal methods (Verschuur and Prein, 1999; Guitton and Cambois, 1999; Hadidi et al., 1999; Dragoset, 1999; Lamont et al., 1999; Lokshtanov, 1999; Matson et al., 1999; Berkhout, 1999; Weglein, 1999). We now use the strong free-surface multiples present in these data to generate pseudo-primaries. Figure 5.22 shows a source-offset cube of the recorded data; I created split-spread data from the off-end marine acquisition using source-receiver reciprocity. The near-offset gap in this case is six traces, much smaller than that in the Sigsbee2B example, but these field data contains much more than just primary and multiple reflections.

Pseudo-primary generation from field data

The pseudo-primary-contribution gather, shown in Figure 4.15, does contain the water-bottom, top-of-salt, bottom-of-salt reflections, and water-bottom multiples, caused by the correlation between the water-bottom reflection and the first-order multiple of the water-bottom, top-of-salt, bottom-of-salt, and second-order water-bottom multiple, respectively. This cube is a series of cross-correlations of single traces, with no amplitude scaling or deconvolution performed before cross-correlation. The signal-to-noise ratio in the data is lower than that in the synthetic example. The side panel shows the same receiver-pair correlation for multiple shots that are later summed to produce a single output trace.

Figure 4.16 shows the volume of pseudo-primaries generated by summing the cross-correlations from all of the sources in the recorded data. The pseudo-primary data are somewhat poorer than the recorded field data in Figure 5.22, with the long wavelet present in the original data strongly featured in the pseudo-primaries. Additionally, when compared to the pseudo-primaries in the synthetic case, the quality of the pseudo-primaries for this example is significantly worse. Reviewing three of the points I made about the pseudo-primaries in Sigsbee, we see that these problems are even more pronounced in this field data example. For example, while there were variations

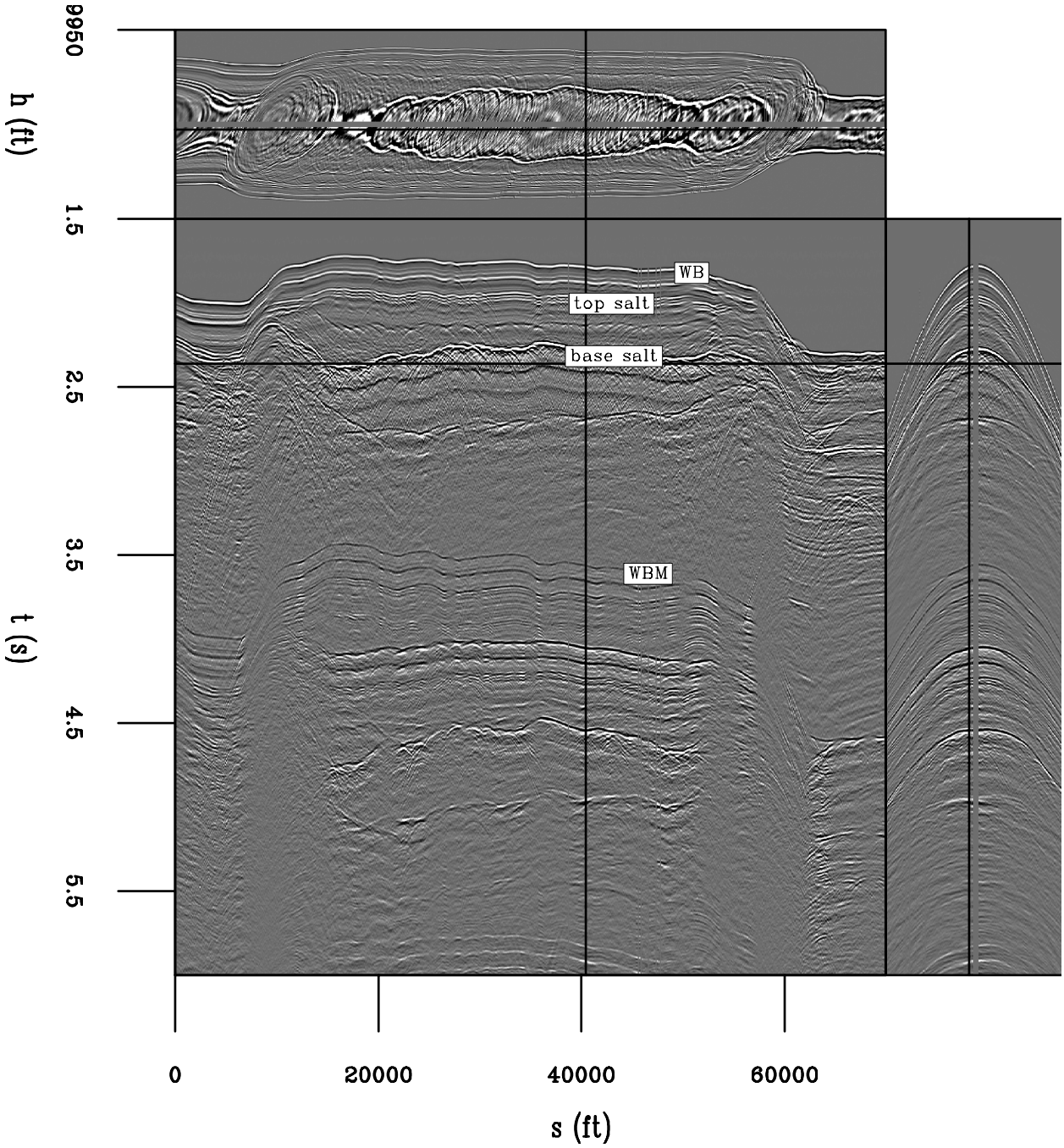

Figure 4.14: A Gulf of Mexico dataset. The front panel is a constant-offset section, and the side panel is a single shout, with the negative offsets predicted by reciprocity.
ER Pseudo/. fieldinann

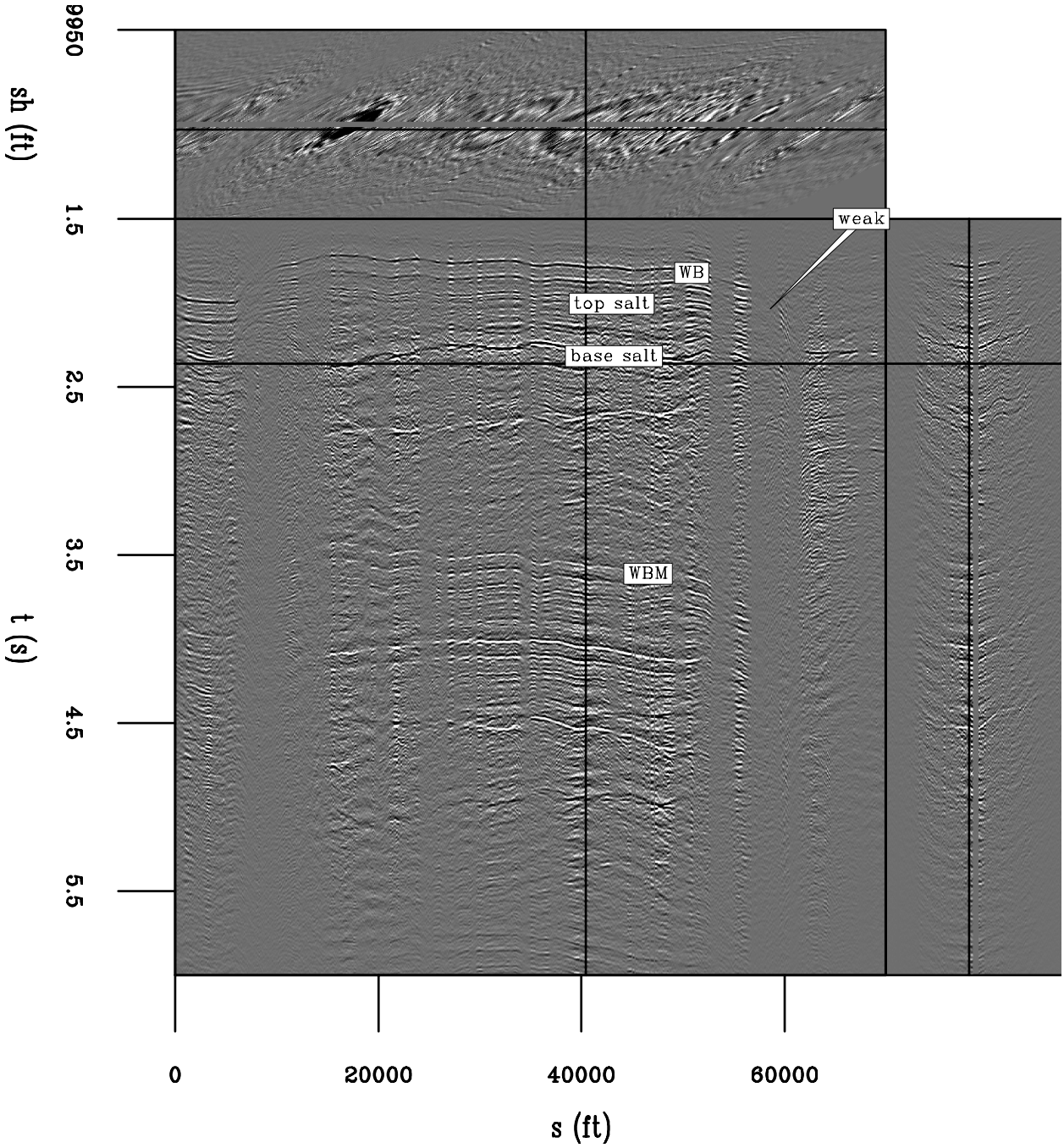

Figure 4.15: Pseudo-primary contribution gather. The front face is a constant-offset section, while the side face contains the shots that will be summed to produce the pseudo-primaries. A $t^{0.8}$ gain has been applied. **CR** Pseudo/. fieldcontribann

in the relative amplitude of the Sigsbee pseudo-primaries, the variations in the pseudo-primaries from the field data are much more pronounced; where the water-bottom reflection is dipping, the pseudo-primary reflection is nearly absent, and the pseudo-primaries of the salt body are much more pronounced than anywhere else. Second, the amplitude spectrum in the Sigsbee example was squared as a result of the cross-correlation involved in pseudo-primary generation. The much longer wavelet in the field data makes this more obvious, with the side-lobes of the water-bottom reflection appearing before the water-bottom reflection in the recorded data. This ringing is probably associated with a water bubble from the source. Third, the cross-talk in this field data is also present in this pseudo-primary result. These slight smile-shaped events have a slope the direction opposite that of the desired pseudo-primaries, and are more obvious at farther offsets, although the base of these smiles are at roughly zero-offset in Figure 4.16.

Interpolation of field data in f-h-s

As in the Sigsbee example, we first take the pseudo-primaries in Figure 4.16, perform a water-velocity normal moveout, and then break the result into 150 overlapping time windows of 32 samples each. These time windows are then Fourier transformed into 150×16 source-offset frequency slices. We then estimate a 3×4 nonstationary PEF that varies every sample along the source and offset axes for each of these slices, and use it to interpolate the corresponding slice of the recorded data with the near-offset gap. For each frequency of each time patch I use 80 iterations of a conjugate-direction solver both to estimate the PEF and to interpolate the missing data. The data are then transformed back to time, the windows are reassembled, and the NMO correction is reversed. This result is shown in Figure 4.17.

The interpolation result in Figure 4.17 is greatly improved over that in the pseudo-primary data. The most obvious differences between the pseudo-primaries and the interpolated result are that the ringing wavelet has been removed and the polarity of the images differ. The dipping portions of the water-bottom that were not present

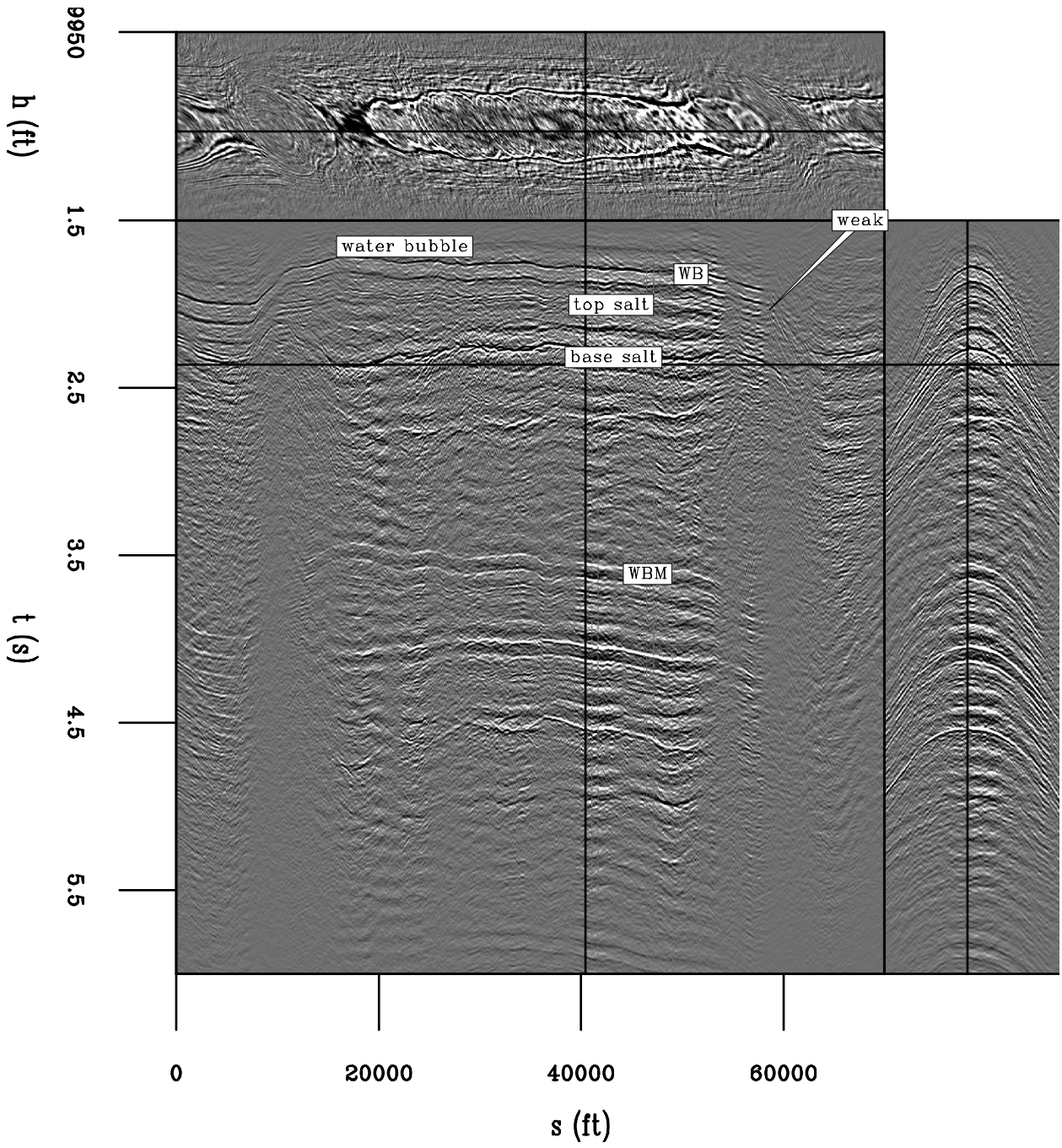

Figure 4.16: Pseudo-primaries of data. The front face is a constant-offset section, and the side face is a shot gather. A $t^{0.8}$ gain has been applied. The gross structure of the original data is present, but the squared wavelet strongly dominates the data. **CR** Pseudo/. fieldpseudoann

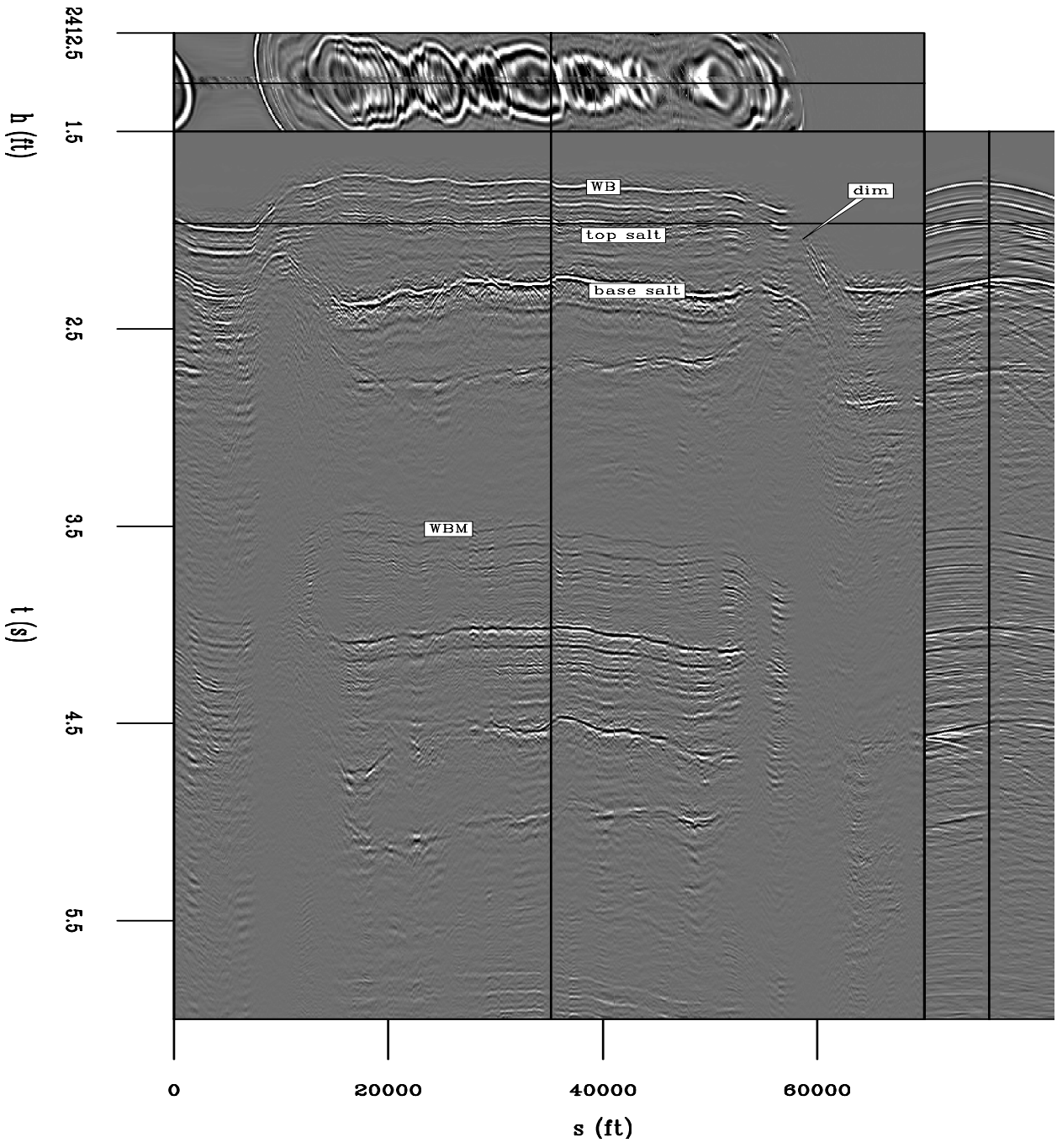

Figure 4.17: Interpolation of field data with a 2D f-h-s PEF using pseudo-primaries. The front face is a constant-offset section, and the side face is a shot. The near offsets appear much more reasonable than the pseudo-primaries in Figure 4.16. A $t^{0.8}$ gain has been applied, and the top and side panels are zoomed in relative to earlier figures in order to emphasize the interpolated values. **CR** Pseudo/. fieldfxann

in the pseudo-primaries were not interpolated. While this gap is much smaller than that in the Sigsbee example, the result is not as impressive. The water bottom, top of salt, bottom of salt, and the associated multiples are reasonably well interpolated, but the subtle stratigraphic reflectors outside of the salt body and the steeply dipping portions of the water bottom are not present in the interpolated result. For example, the diffractions below the bottom of salt reflection or at the edges of salt are not present in the interpolated result. The subtler stratigraphic reflectors outside of the salt body are well interpolated, even at later times.

PSEUDOPRIMARIES IN 3D

The pseudoprimaries generated so far have been either for a two-dimensional synthetic model or for two-dimensional field data example with strong multiple reflections. I now attempt this same 2D pseudoprimary generation on 3D prestack field data containing a region with significant crossline dip. The original recorded data in Figure 4.18a shows a single constant-offset section from 250m inline offset and the second receiver cable. This is the same data used in Chapter 1 in Figure 1.5 where a 2D SRME algorithm fails to produce accurate multiples in the submarine canyon. I generate pseudo-primaries using the same 2D subset of these data, where I take the second receiver cable and solve equation 4.2 to produce the pseudoprimaries for the same source locations and offset in Figure 4.18b.

The produced pseudoprimaries have a virtual source sampling that is equal to the receiver spacing, which is three times the original source spacing. The pseudoprimaries have a lower signal-to-noise ratio than the previous two examples, because the multiples are not as strong in these data as in the previous examples. The areas with a largely two-dimensional water bottom ($s_x : 17000 - 25000$m) have a pseudoprimary reflection that is kinematically in the correct place, and the top of the salt body reflection at $s_x = 19000$ is also visible. The other side of the canyon ($s_x : 5000 - 7000$) is also at roughly the correct arrival time. The water-bottom canyon, with a significant crossline slope, is not visible.

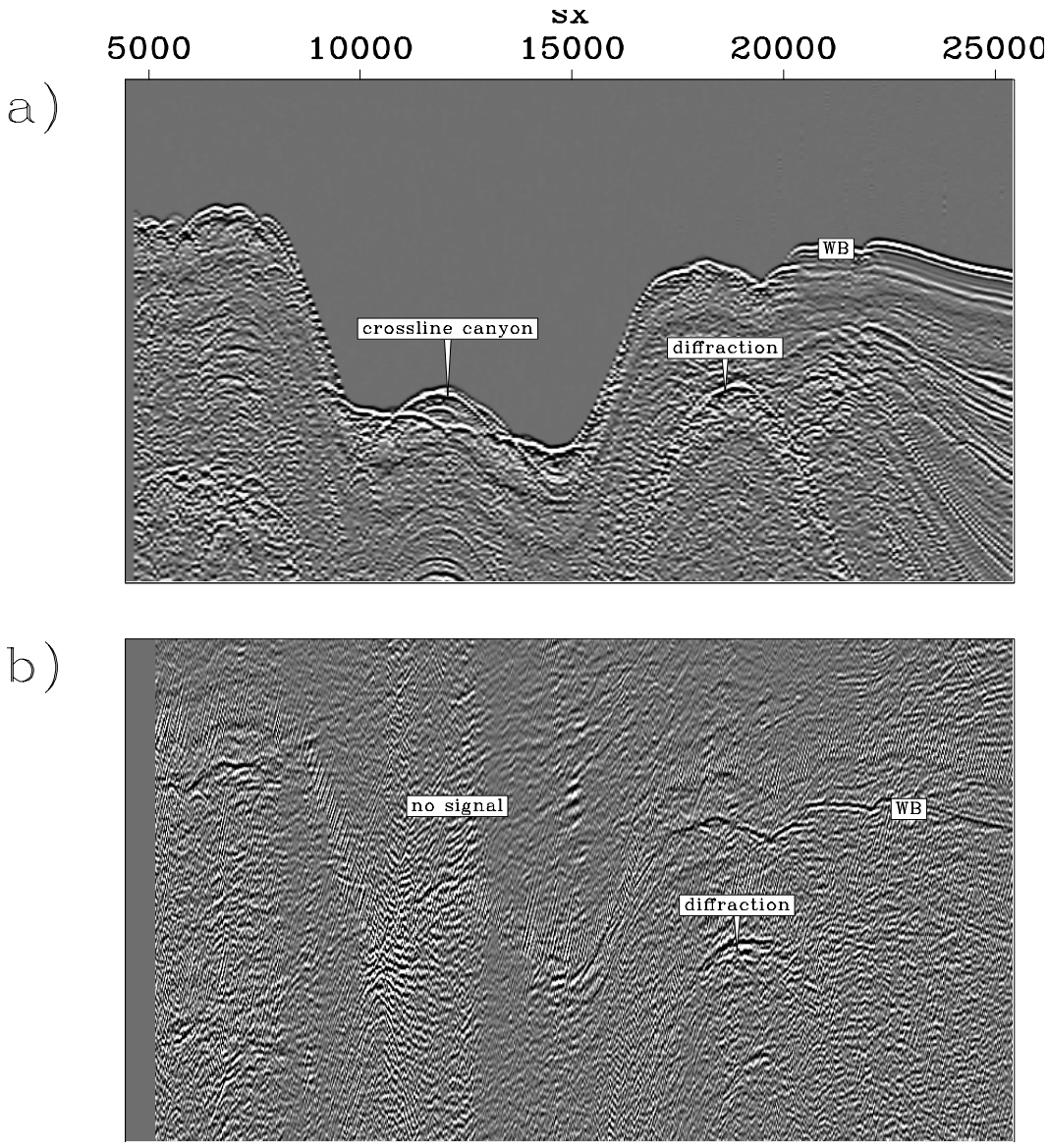

Figure 4.18: 2D pseudoprimary generation for 3D field marine data. a) original data; b) pseudoprimaries. The canyon present in the water-bottom has a large crossline component and the pseudoprimaries are incorrect. **CR** Pseudo/. 3dfieldpseudoann

To address this problem, producing a full 3D volume of pseudoprimaries from the recorded field data is not an appealing prospect. The source density is very low in the crossline direction, and the variable cable feathering along each sail line as well as from sail line to sail line would reduce the number of sources for each output virtual source location. Instead of dealing with these added complexities, I use a synthetic example where the geology is fully known and the acquisition is along ideal overlapping straight lines.

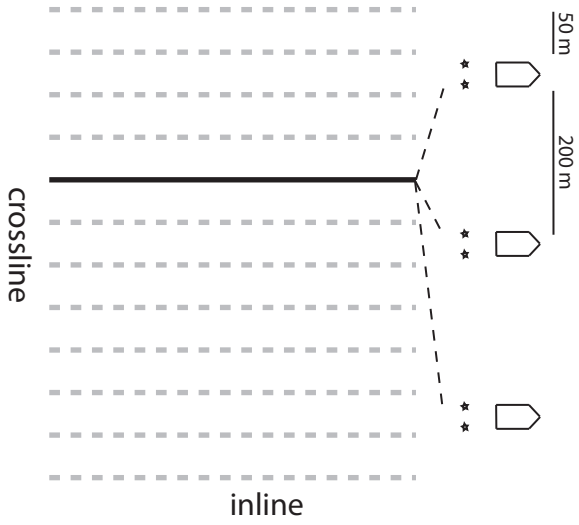

Figure 4.19: Crossline source distributions for a pseudoprimares generated from a single receiver cable at zero offset. **NR** Pseudo/. 3dpseudoann

Figure 4.19 is an illustration of the source distribution for a single receiver line from the 3D synthetic dataset used in both Chapters 1 and 5, courtesy of ExxonMobil. As shown in Figure 5.7a, the data are composed of a horizontal water-bottom reflection under which there is a prism filled with point diffractors. At slightly below the arrival of the diffracted multiples, three reflectors are present. The acquisition geometry, shown in Figure 4.19, is ideal, so that at each subsequent sail line the cables shift by a distance exactly equal to that between four receiver cables. As such, any one receiver cable location occurs for three sail lines given the recording aperture of 550m, which when multiplied by the two crossline source positions per sail line, gives six crossline source contributions for any one crossline receiver location under these ideal circumstances.

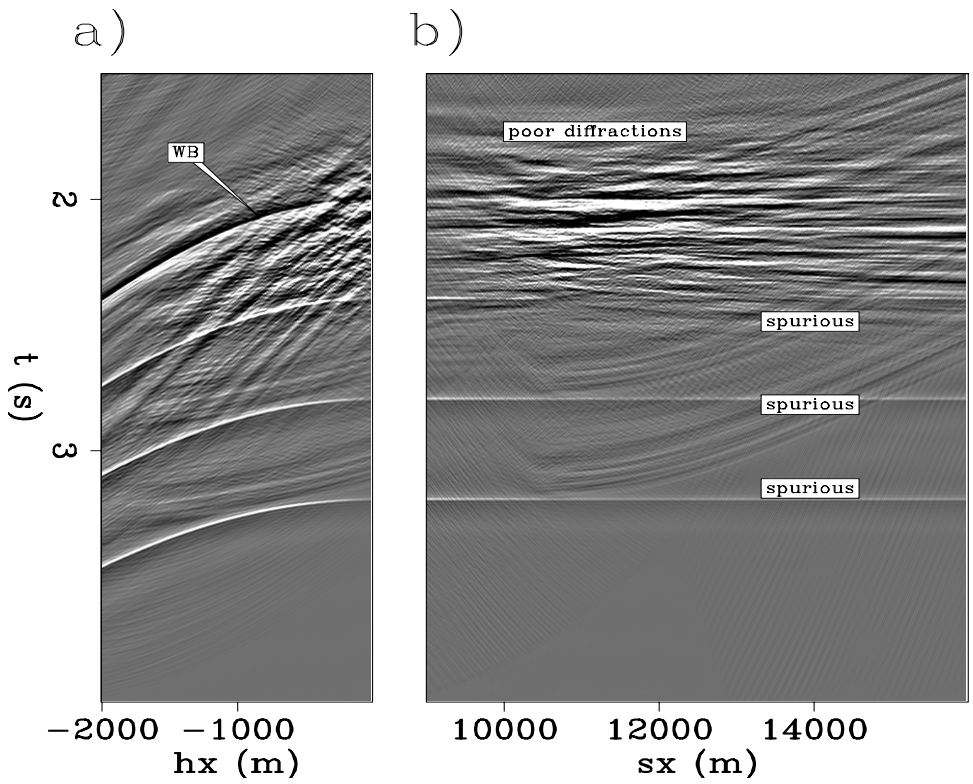

Figure 4.20: pseudoprimaries generated for a single receiver cable from six different crossline source positions. **CR** Pseudo/. 3dsynpseudoann

Figure 4.20 is the result of creating pseudoprimaries for zero crossline offset for a single receiver cable, where each active receiver location is cross-correlated with every other receiver location along the same cable for all sources in the three sail lines. These 640 sources for each of the three sail lines are added together to produce the pseudoprimary image in Figure 4.20, generated from roughly 4TB of individual crosscorrelations. While the water-bottom reflection initially appears to be in the correct place, the diffracted multiples are a blurred mess at near offsets and are absent at the far offsets where the limbs of the recorded diffractions are in the recorded data. There are also apparent reflectors below the water bottom that are created from the correlation of the water-bottom primary reflection with the much deeper series of three primary reflections at below the water-bottom multiple arrival time, illustrating the potential pitfalls of unwanted correlations.

In order to determine the variability of the predicted pseudoprimares with crossline source position, I have produced a crossline gather similar to the pseudoprimary contribution gathers I produced earlier in this chapter. While the full 3D uncollapsed cube of crosscorrelations would have 7 dimensions: time, inline and crossline source, inline and crossline first receiver, and inline and crossline second receiver, I only show the predicted multiples for a single first receiver (virtual source), second receiver, and inline source position, showing how the predicted multiples vary as the crossline location of the source varies. Figure 4.21 contains two of these gathers, where 4.21a is a zoom in on an area of the data with little diffracted multiples at the water bottom. The arrival time of the water bottom changes as a function of distance from the receiver cable to the source, with the stationary phase point at zero. Looking at a larger section of the time axis for a different location located in the cloud of diffractions below the water-bottom in Figure 4.21b, there is some similarity between the traces in adjacent flip-flop shots, but very little similarity between the three different sail lines, which in part explains the incoherence of the diffracted pseudoprimaries in Figure 4.20.

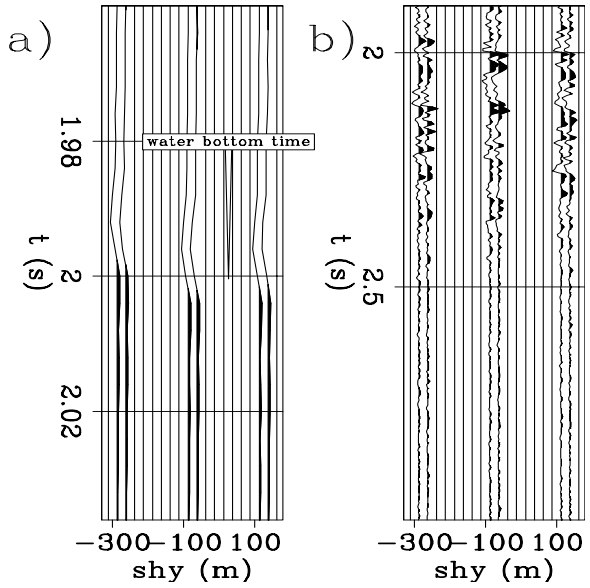

Figure 4.21: A crossline pseudoprimary contribution gather. The location of the water-bottom reflection changes as a function of the crossline distance from the inline location of the pseudoprimaries to the six sources. **CR** Pseudo/. 3dsynpseudogatherann

Pseudoprimaries and source sampling: Sigsbee analogue

I now use the Sigsbee2B dataset to further explore the limitations of source distribution, and attempt to separate the issues of source density and source aperture, using a subsampling of a 2D dataset as a proxy for 3D crossline sampling. Figure **??** is a comparison of different source densities on the zero-offset section of the pseudoprimaries, with the source sampling interval increased to twice (a), 10 times (b), and 50 times (c) that of the receiver sampling. The signal quality of the pseudoprimaries is still reasonable with 1/10 of the sources, but reducing it to 1/50 degrades the signal beyone recognition. This source density can also be viewed in shot gathers, as in Figure 4.23. Here the shot gathers look reasonable even at 1/20 the original source sampling, and the far offsets degrade only somewhat more than the near offsets.

Now instead of comparing the density of sources I compare their spatial extent in Figure 4.24. Figure 4.24a is a shot gather generated from all of the sources present in the full recording aperture of 53000 ft, while in subsequent figures I limit it to 15000 ft (4.24b), 10000 ft (4.24c), 5000 ft (4.24d), and 2000 ft (4.24e). The spatial extent of useful pseudoprimaires reduces with the source aperture, when the source aperture is larger than the range of offsets useful pseudoprimaries.

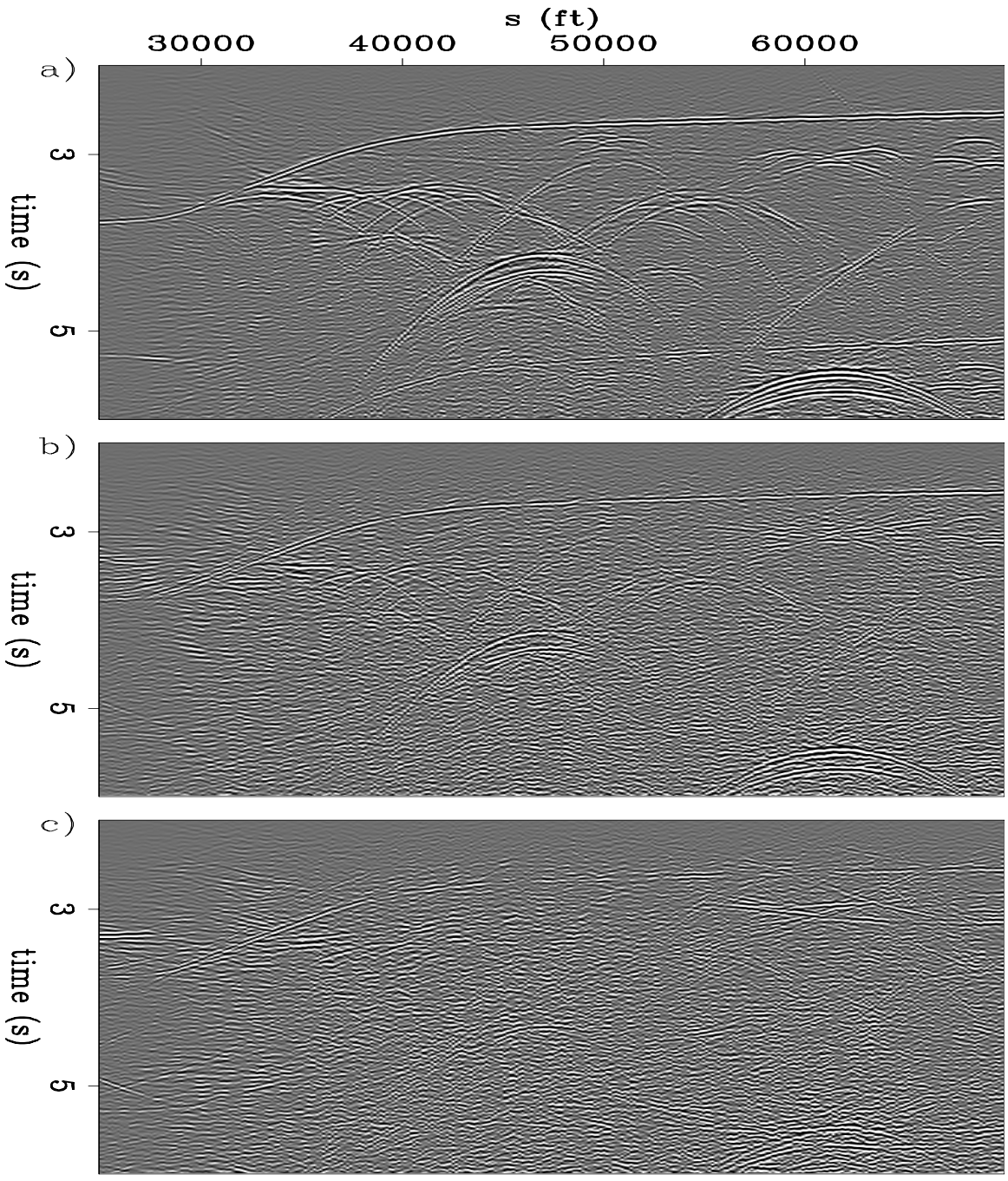

Figure 4.22: A comparison of source intervals on pseudoprimary quality, viewed as a zero-offset section: (a) 300 ft, (b) 750 ft, (c) 1500 ft, (d) 3000 ft, (e) 7500 ft. The signal quality degrades as the density of sources is reduced, but is still coherent with one source for every 20 offsets. **CR** Pseudo/. dsdeps

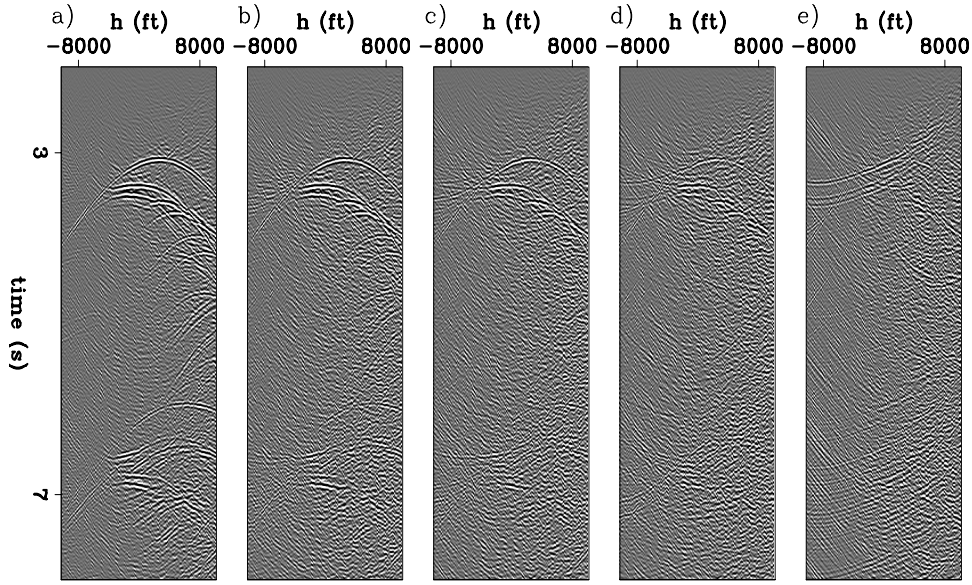

Figure 4.23: The same comparison as in Figure 4.22, but viewed as a virtual source gather at s=35000 ft and source sampling of: (a) 300 ft, (b) 750 ft, (c) 1500 ft, (d) 3000 ft, (e) 7500 ft. **CR** Pseudo/. dsdeph

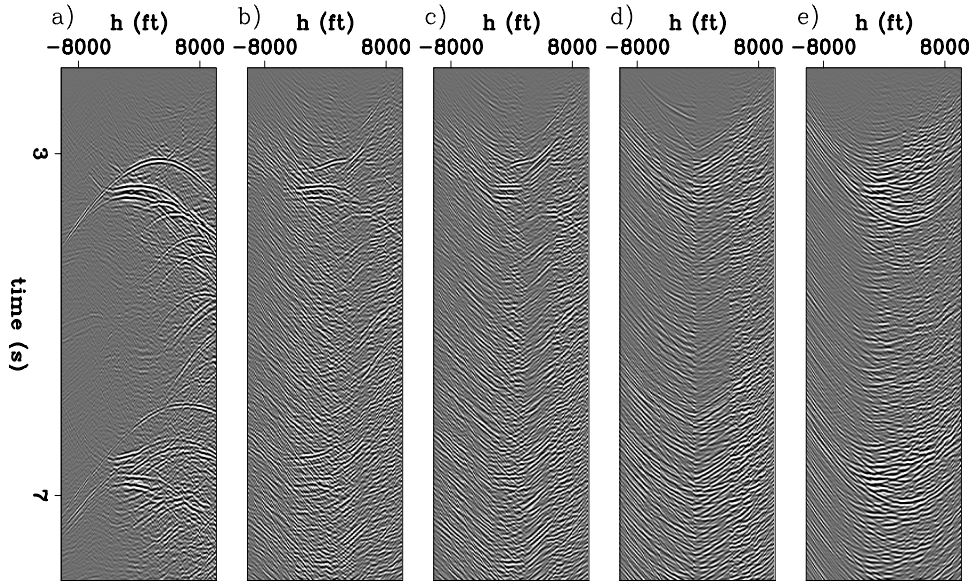

Figure 4.24: A comparison of aperture for pseudoprimaries at s=35000 ft with sources along (a) 15000, (b) 10000, (c) 5000, (d) 2000 ft surrounding the virtual source point. The data rapidly degrades as the aperture is limited. Pseudo/. nsdeph

CONCLUSIONS AND FUTURE WORK

The results in both the synthetic and field data examples show that this method works at capturing large inline features in the data. The field data example shows in particular that the PEF is insensitive to issues with the squared spectrum, amplitude scale and phase, which avoids the need for preprocessing involving amplitude scaling and deconvolution prior to cross-correlation.

The utility of this method in 3D is limited by the poor spatial distribution of sources, which has also been the case with passive interferometric methods. However, future acquisition methods such as wide-aziumth seismic may increase source density and more importantly aperture, which could in turn produce more useful pseudoprimaries in the crossline direction. Alternatively, this approach could be combined with moveout operators to produce data to add to the correlation.

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

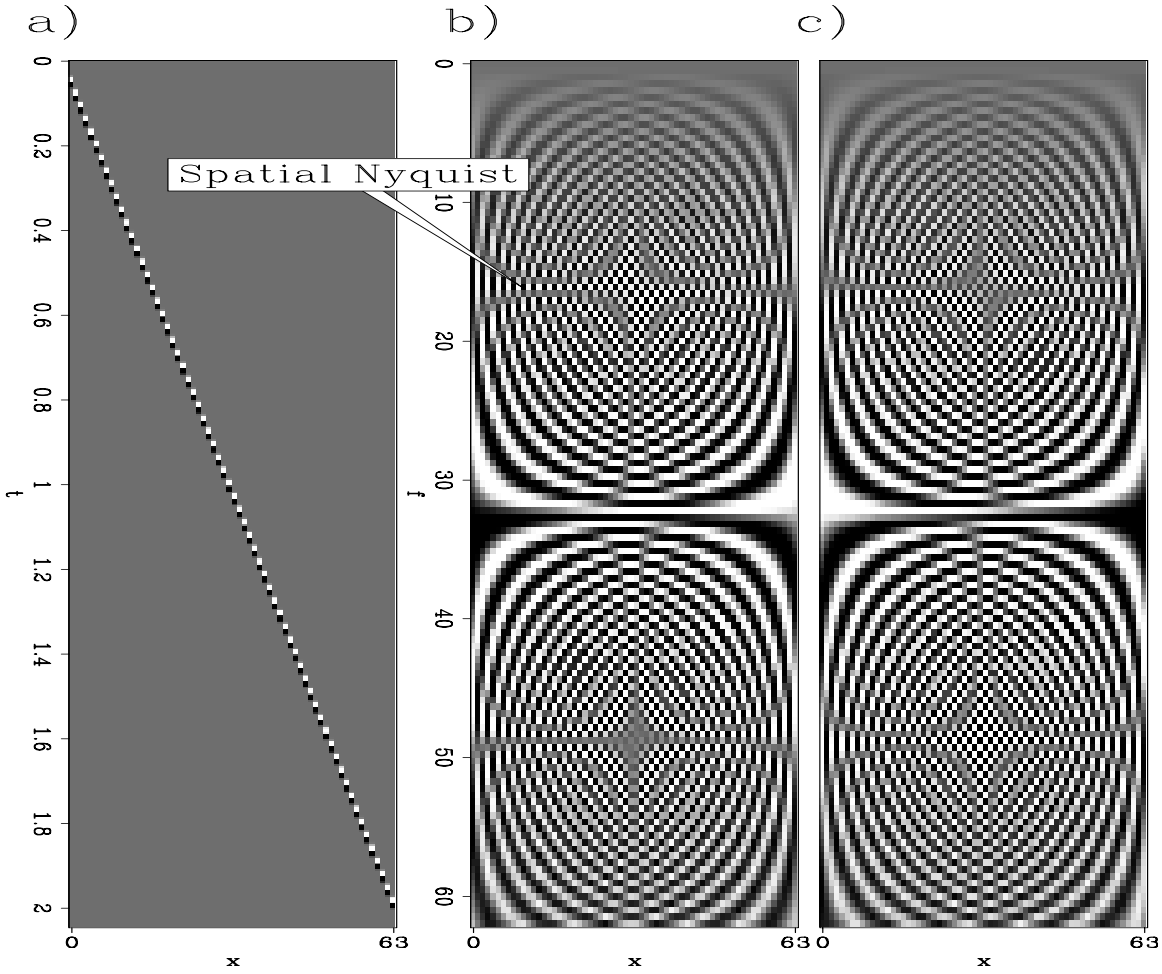

Figure 5.1: A spatially-aliased plane wave in t-x, generated with a Ricker wavelet with a central frequency of 55 Hz and a slope of 650 m/s sampled 64 times (every 20 m) shown in time and space in (a). The real (b) and imaginary (c) values of the positive frequencies show how the spatial wavenumber increases 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),

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

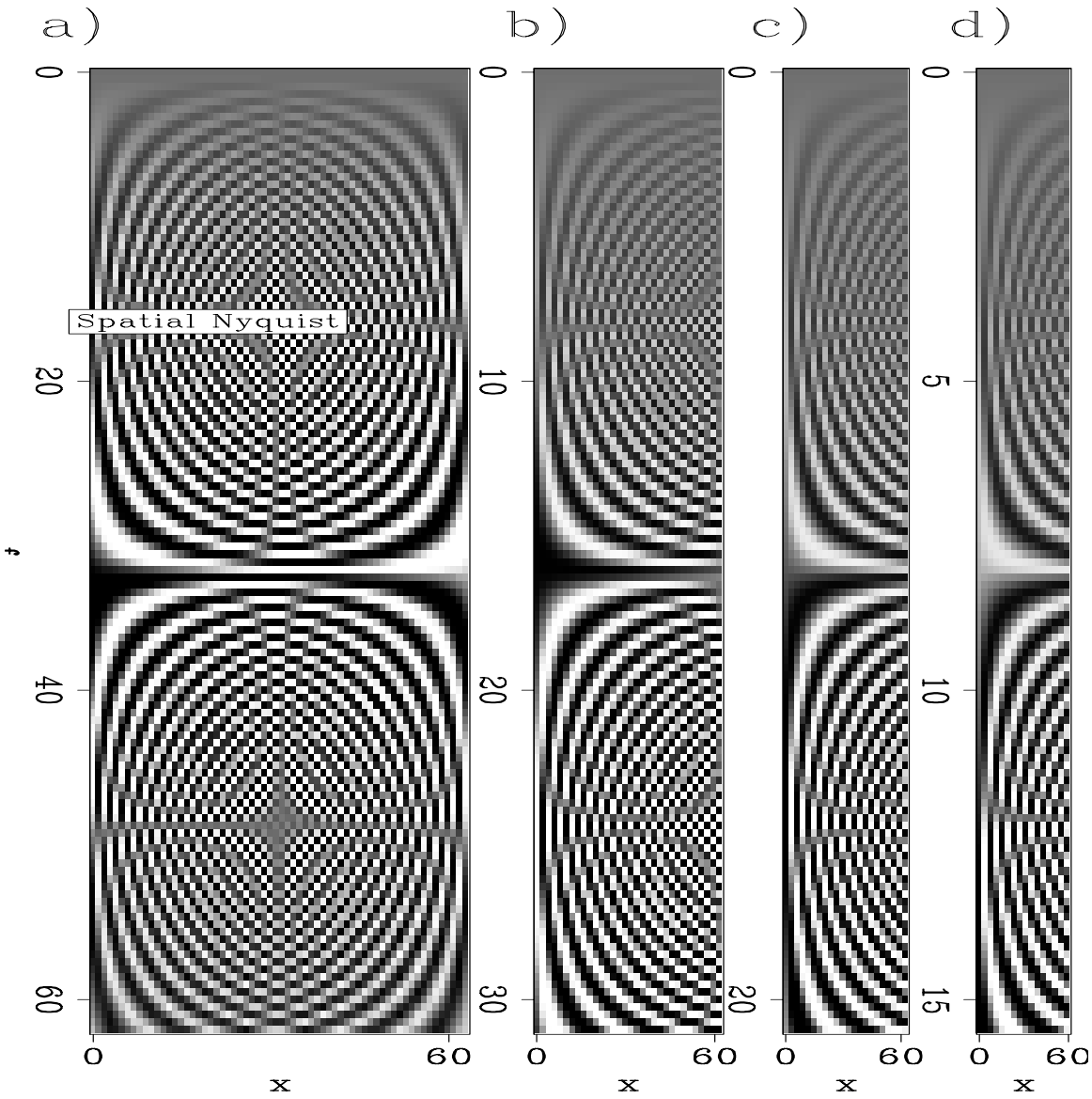

Figure 5.2: A plane wave in f-x with four different sampling rates. a) original f-x data, with Δx and Δf. b) resampled to $\frac{\Delta f}{2}$ (showing the lower half of frequencies) and $2\Delta x$. c) $\frac{\Delta f}{3}$ (showing the lower third of frequencies) and $3\Delta x$. d) $\frac{\Delta f}{4}$ (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

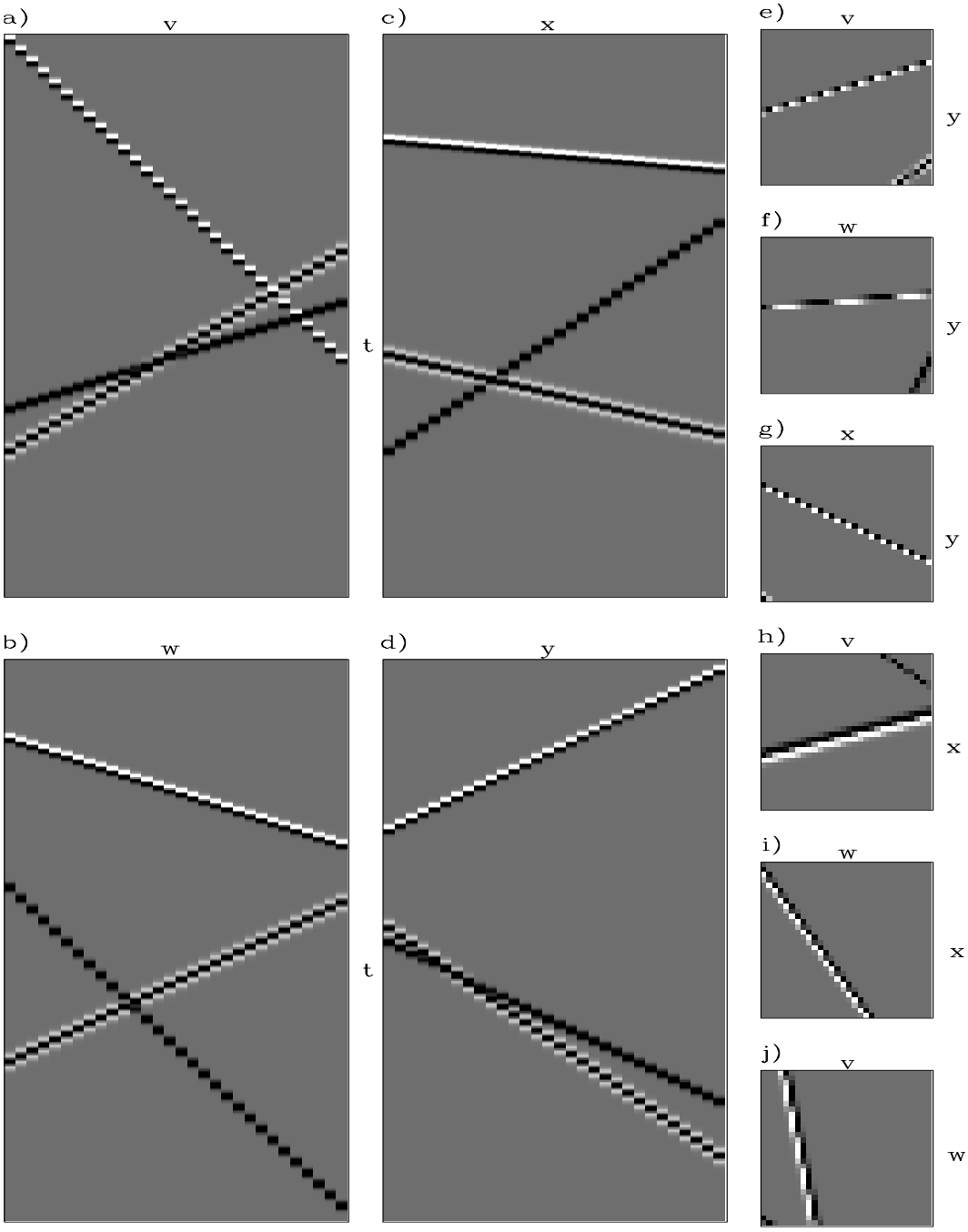

Figure 5.3: Three plane waves in five arbitrary dimensions, four spatial and time (v,w,x,y,t). The ten slices correspond to combinations of all of the axes: a) v,t; b) w,t; c) x,t; d) y,t; e) v,y; f) w,y; g) x,y; h) v,x; i) w,x; j) v,w. The plane waves are severely aliased along many of the axes. **ER**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

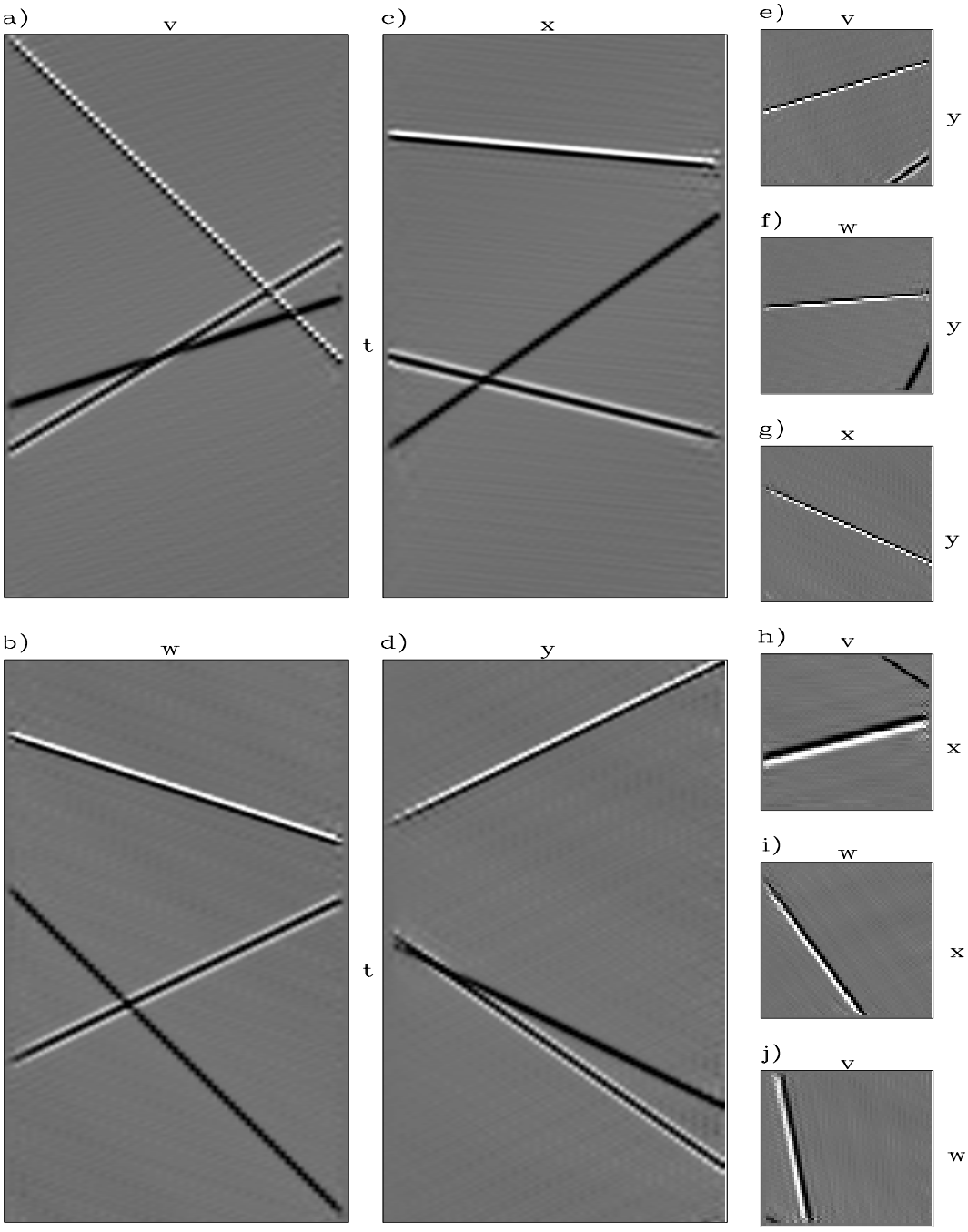

Figure 5.4: Interpolation of the data in Figure 5.3 using 128 four-dimensional PEFs. The aliased data are properly interpolated along all axes, and the amount of data has been increased by a factor of 16. **ER** fxNS/. planesinterpann

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,

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

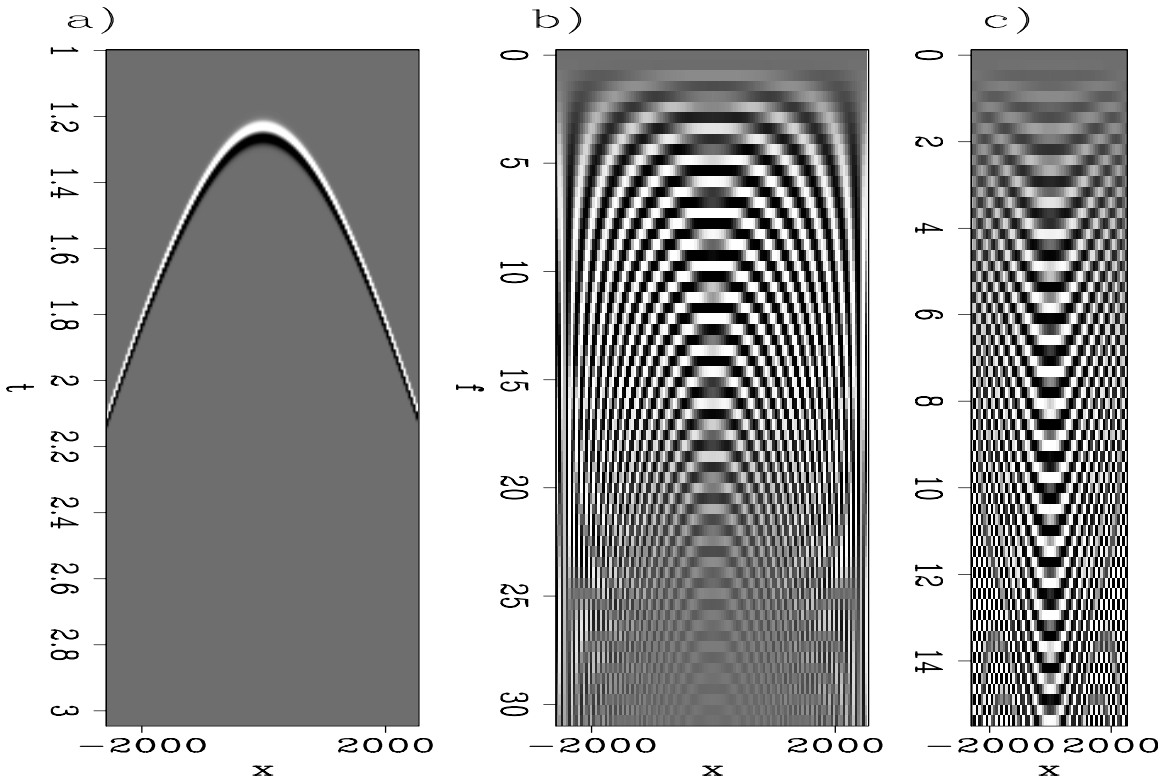

Figure 5.5: A single hyperbola in t-x and f-x: a) a hyperbola in t-x; b) the real part of the same hyperbola in f-x; c) the real part of the same data with half the spatial samples removed and the lower half of frequencies shown. For each frequency, the local wavenumber content is the same in both b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

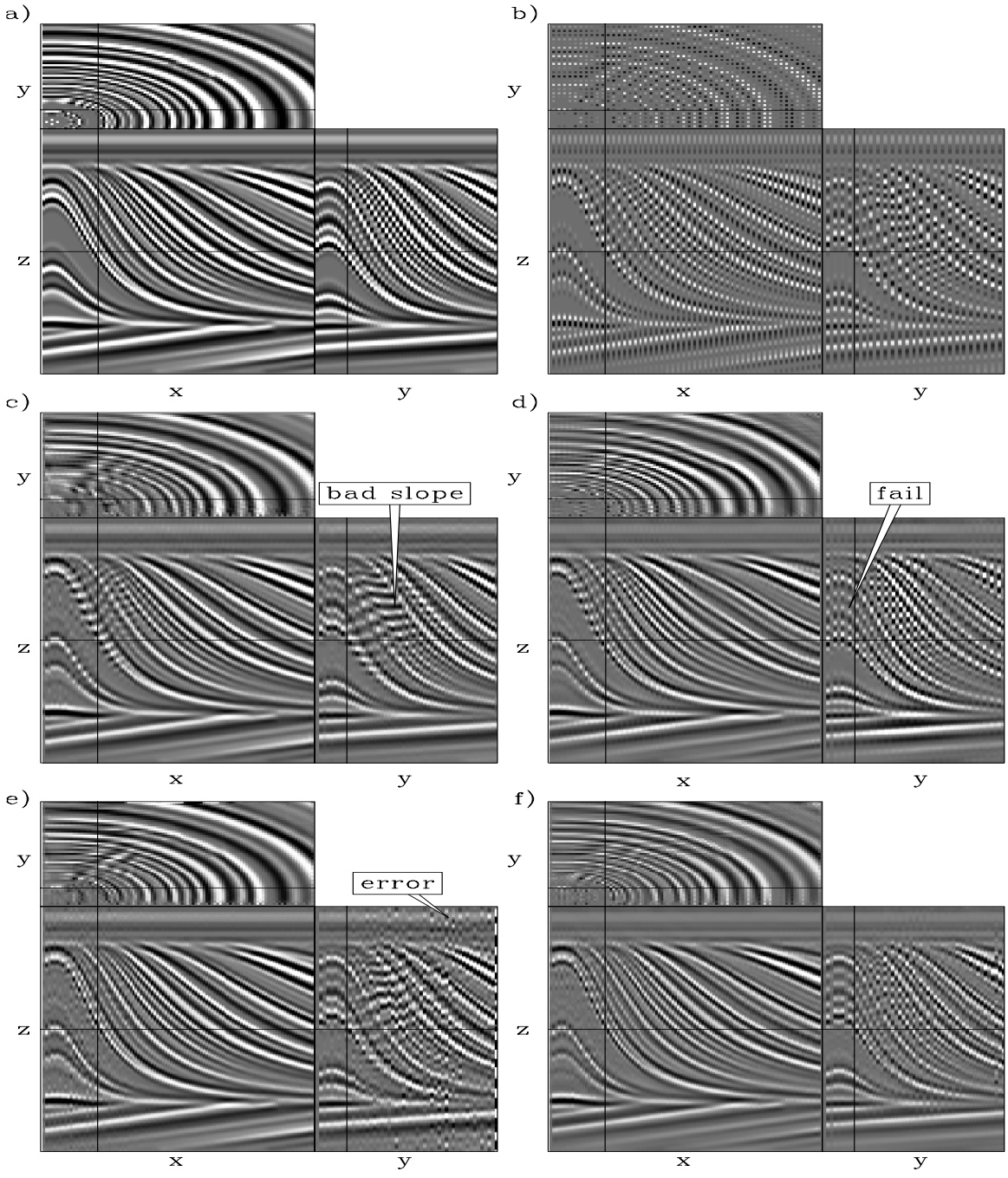

Figure 5.6: Interpolation of the quarter-dome synthetic. a) original data. b) data sub-sampled by a factor of two on each spatial axis. c) Stationary f-x interpolation. d) f-x interpolation in patches. e) Nonstationary f-x interpolation. f) Nonstationary f-x interpolation in time patches. The nonstationary 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

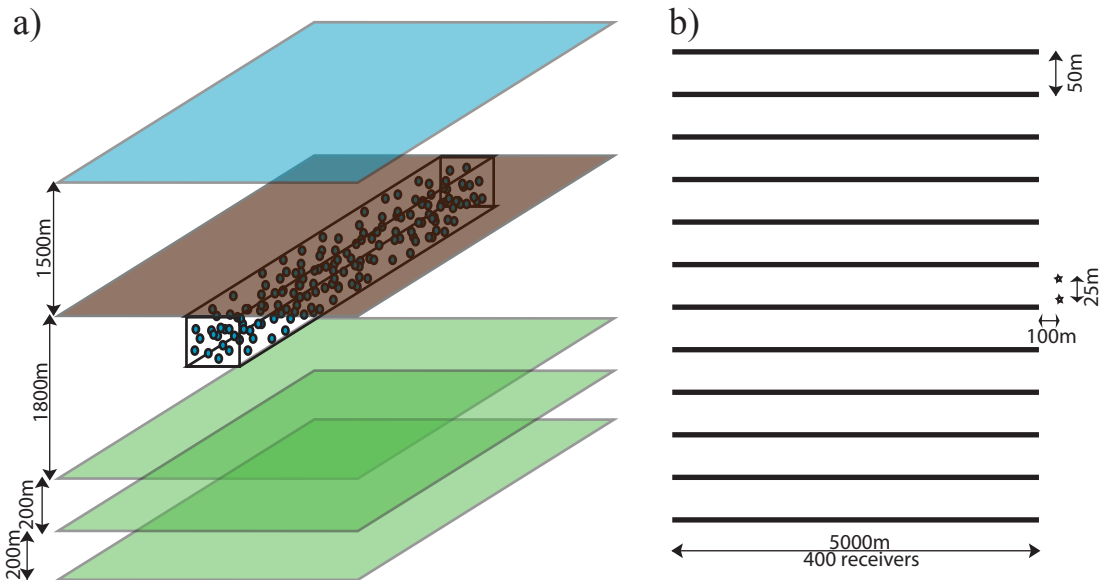

Figure 5.7: Schematic of the synthetic dataset: a) shows the model, a prism-shaped region below the water bottom filled with point diffractors, below which lie three reflectors whose primary reflection times are nearly the same times as those of water-bottom multiples; b) is a plan view showing the acquisition 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

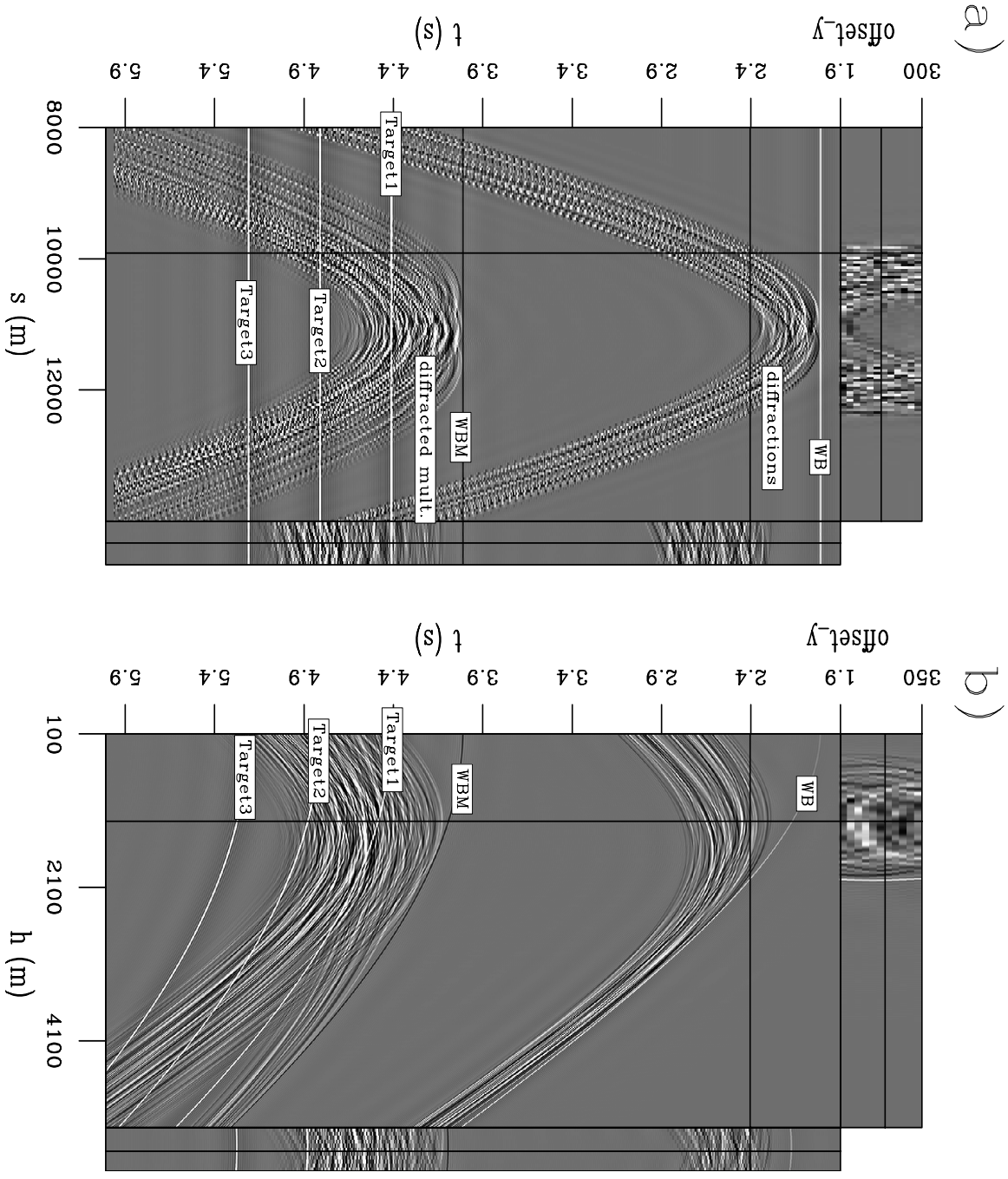

Figure 5.8: Synthetic dataset. a) a cube with a constant-offset section on the front face, crossline offsets on the side face, and a time slice through this cube on the top; b) another cube showing a single shot with inline offsets on the front face, crossline offsets on the side face, and a time slice 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

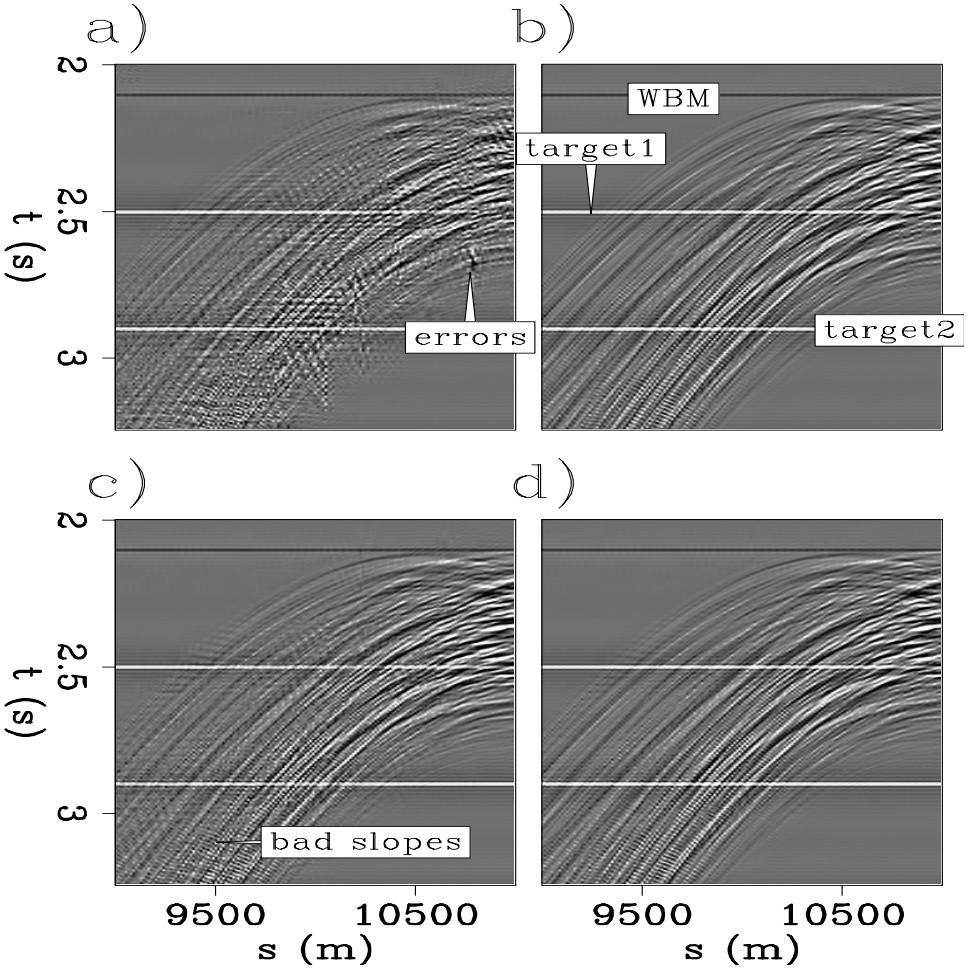

Figure 5.9: A zoomed-in comparison of inline source interpolation along constant-offset sections at 1137.5 m using different PEF dimensions. a) 2D interpolation along constant inline offset sections. b) 3D interpolation along inline source, inline offset cubes. c) 3D interpolation along inline source, 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

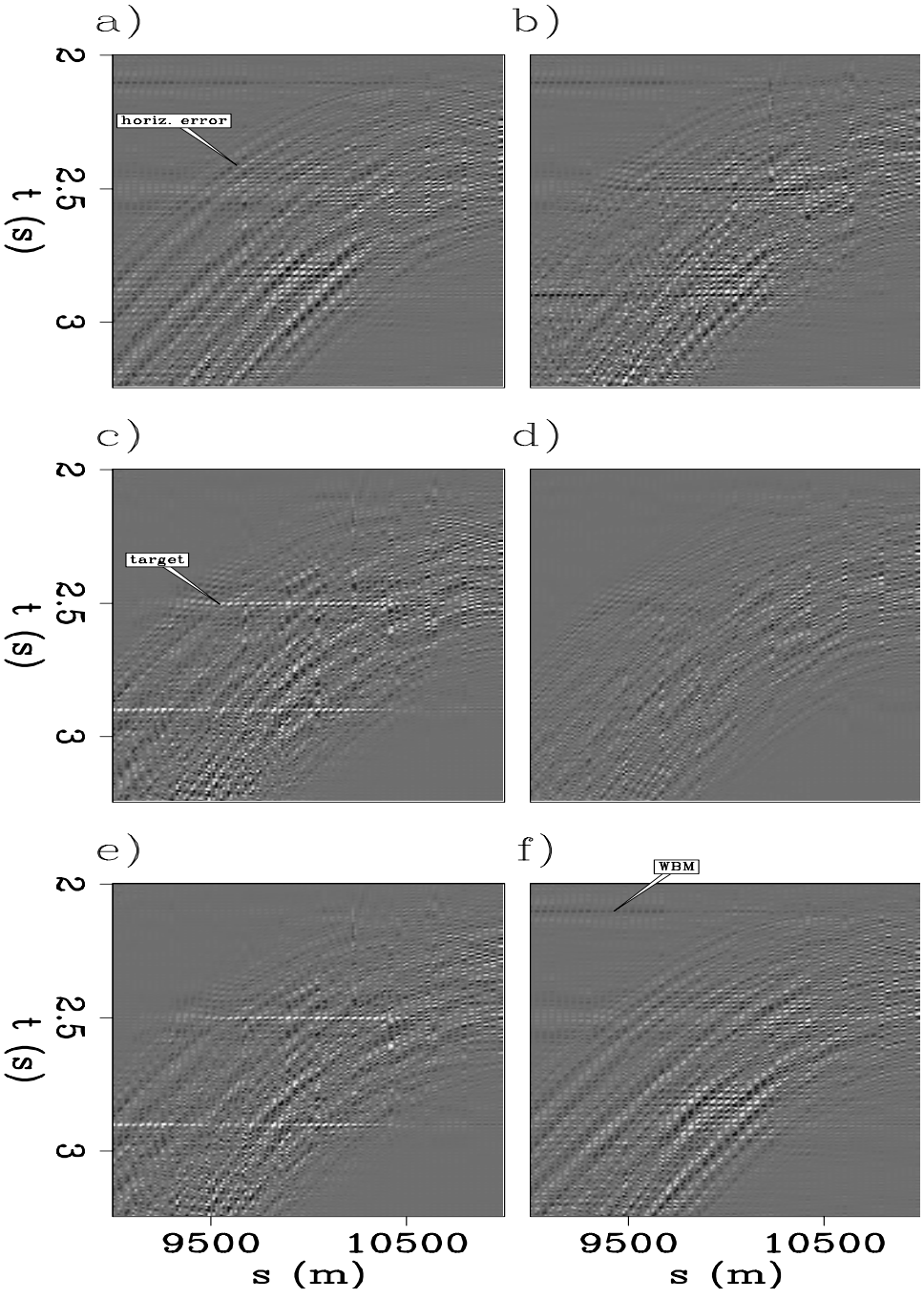

Figure 5.10: Differences between the plots in Figure 5.9. a) 2D vs. 3D inline interpolation; b) 2D vs. 3D crossline interpolation; c) 3D inline vs. 3D crossline interpolation; d) 3D inline vs. 4D interpolation; e) 3D crossline vs. 4D interpolation; f) 2D vs. 4D interpolation. The 2D results have horizontal 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

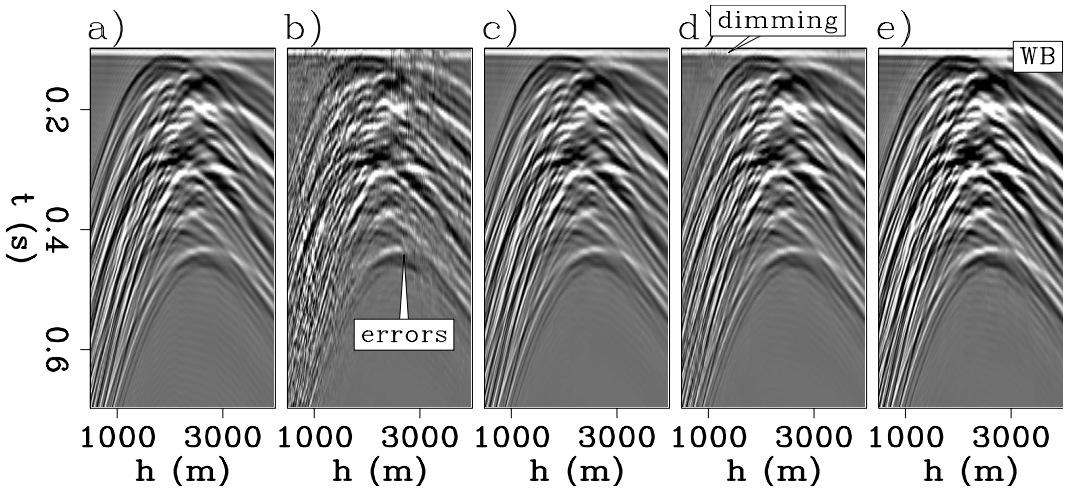

Figure 5.11: One cable from an interpolated inline source at 11706.5 m. a) original recorded shot located adjacent to the interpolated shots in b-e. b) 2D interpolation along constant inline offset sections. c) 3D interpolation along inline source, inline offset cubes. d) 3D interpolation along inline 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

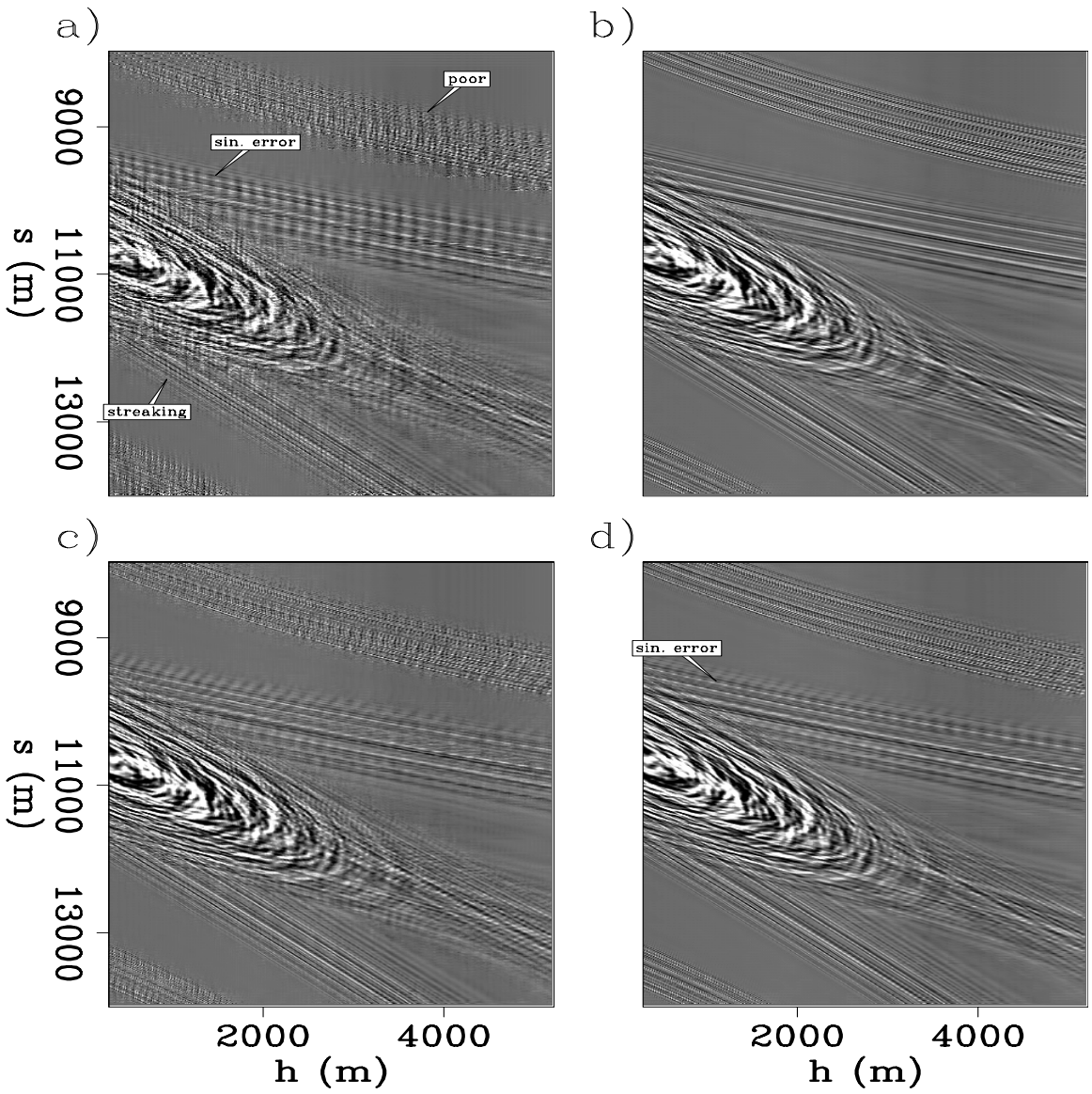

Figure 5.12: Time slices from 4.336 s in inline source and inline offset, where the inline source axis has been interpolated by a factor of three. a) 2D interpolation along constant inline-offset sections. b) 3D interpolation along inline-source, inline-offset cubes. c) 3D interpolation along inline-source, 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

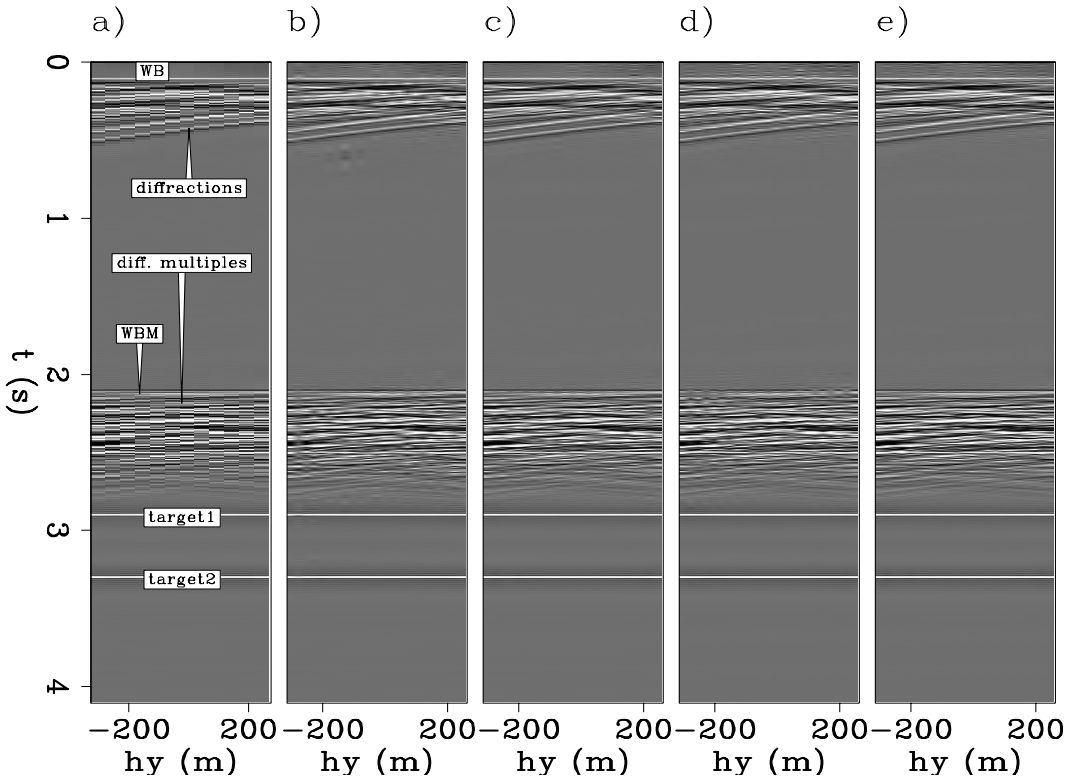

Figure 5.13: Receiver cable interpolation viewed for a single source at 10931.25 m and a single inline offset at 900 m. a) original data. b) 2D interpolation of crossline offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, inline-source cube interpolation.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

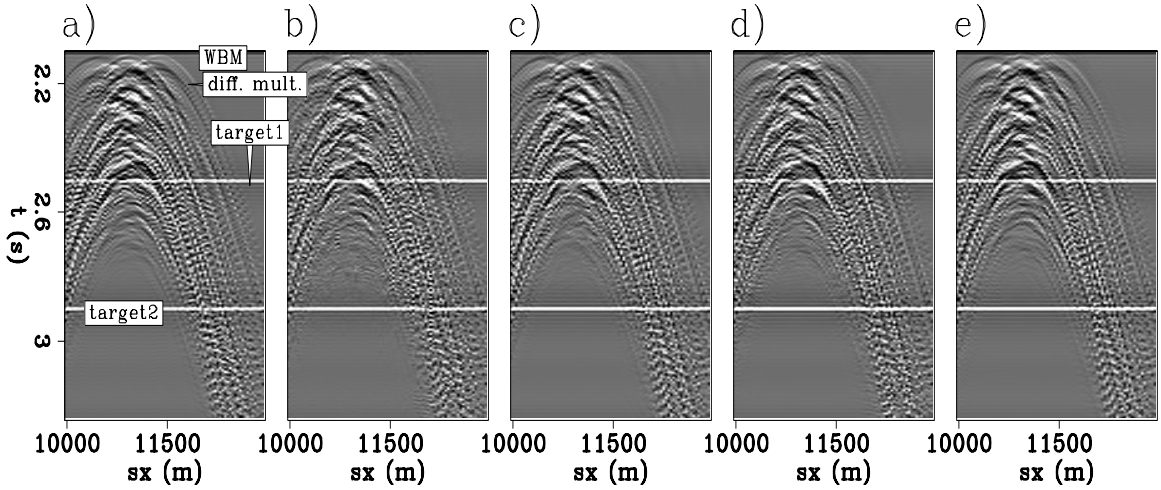

Figure 5.14: Receiver-cable interpolation viewed for constant-offset (900 m) sections. a) original data from a nearby receiver cable. b) 2D interpolation of crossline-offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, inline-source cube interpolation. e) 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

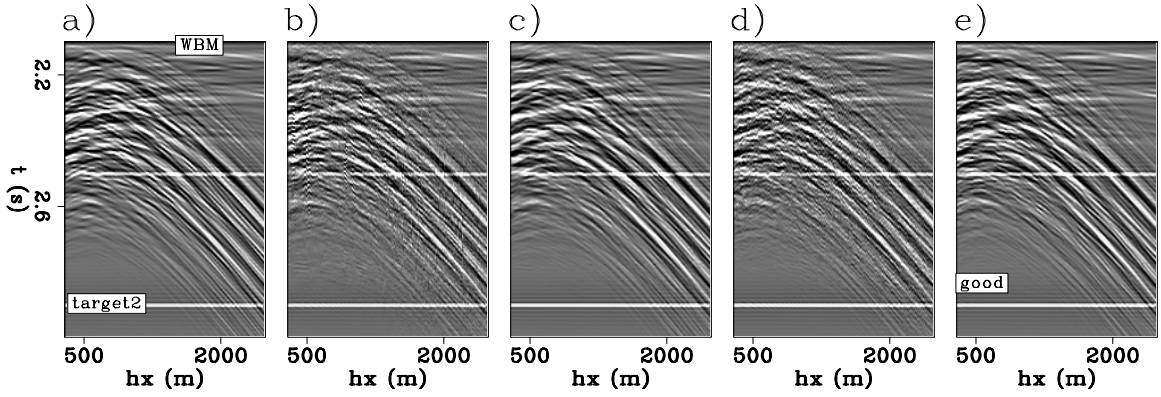

Figure 5.15: Receiver cable interpolation viewed for a single shot at 10931.25 m and a single interpolated cable at zero crossline offset. a) recorded data from a nearby receiver cable. b) 2D interpolation of crossline-offset gathers. c) 3D crossline-offset, inline-offset cube interpolation. d) 3D crossline-offset, 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

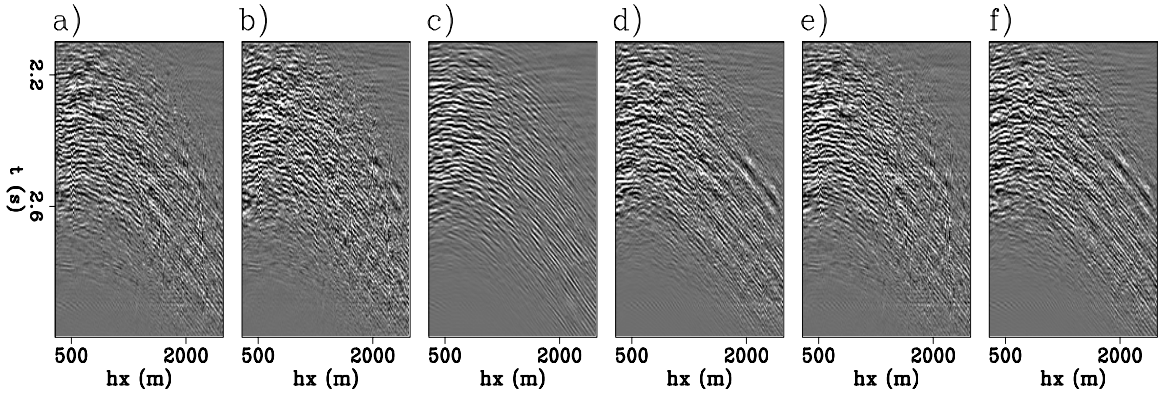

Figure 5.16: Differences between images in Figure 5.15. a) 2D vs. 3D receiver interpolation; b) 2D vs. 3D source interpolation; c) 3D receiver vs. 4D interpolation; d) 3D source vs. 4D interpolation; e) 2D vs. 4D interpolation; f) 3D source vs. 3D receiver interpolation. The 3D crossline receiver, inline 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

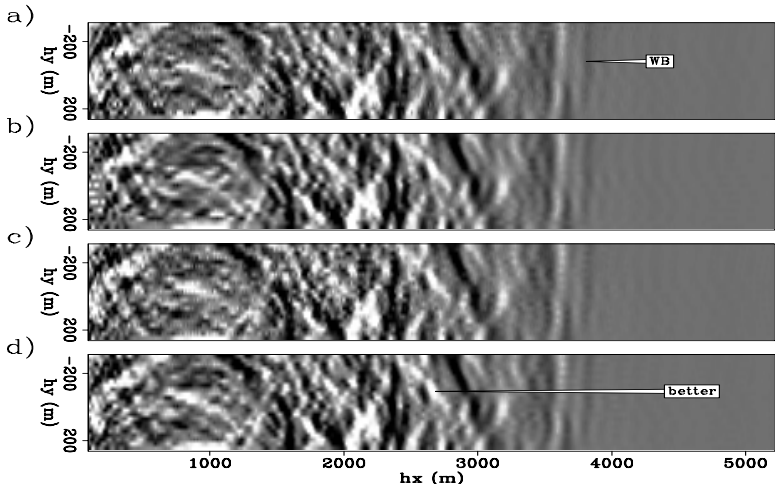

Figure 5.17: Receiver cable interpolation viewed in a time slice at 2.276 s through a crossline-offset, inline-offset cube at $s_x = 10931.25$ m. a) 2D interpolation of crossline-offset gathers. b) 3D crossline-offset, inline-offset cube interpolation. c) 3D crossline-offset, inline-source cube interpolation. 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

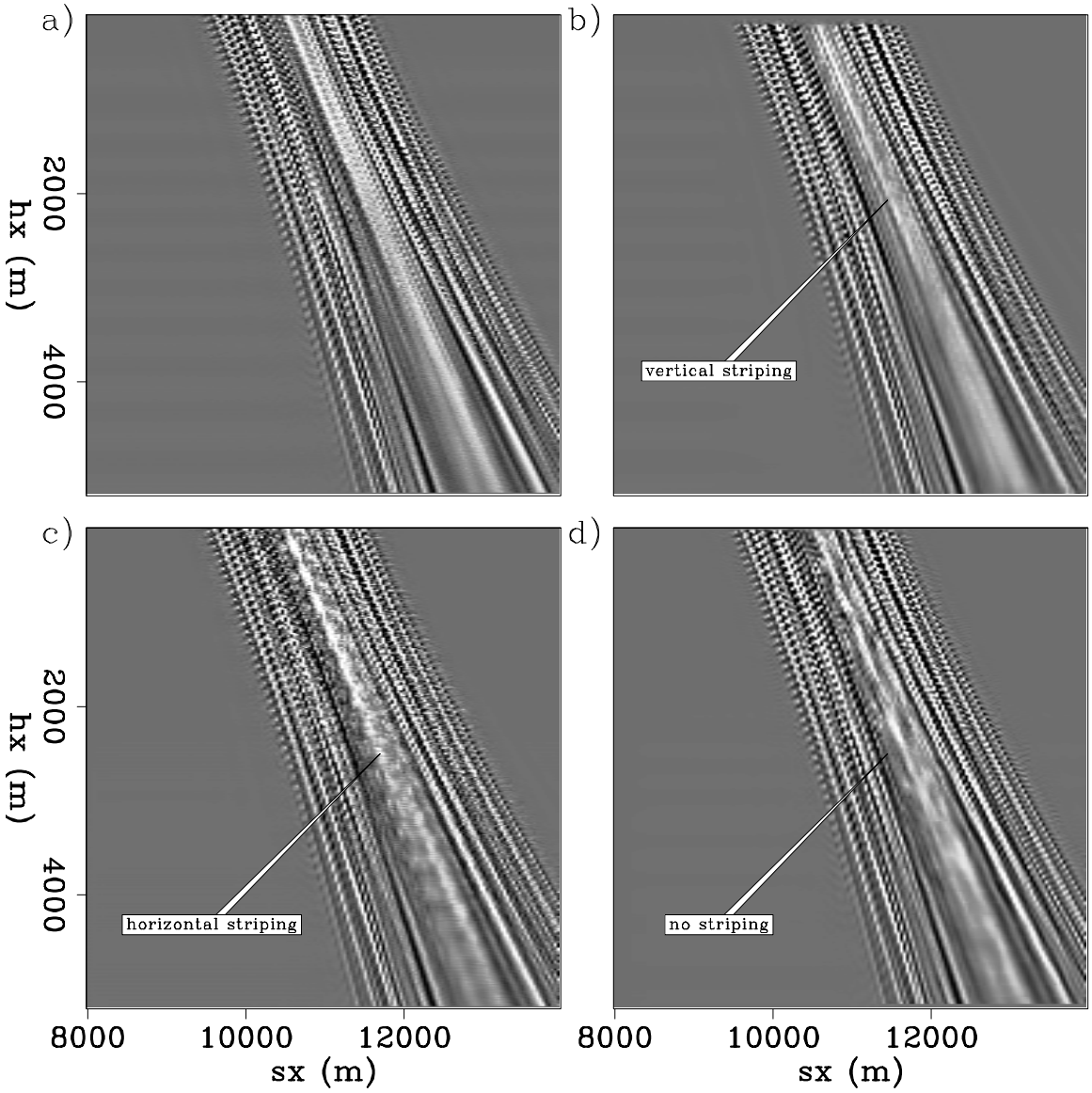

Figure 5.18: Receiver cable interpolation ($h_y = -25$ m) viewed in a time slice at 2.276 s through an inline-offset, inline-source cube below the water-bottom reflection for an interpolated receiver cable. a) 2D interpolation of crossline-offset gathers. b) 3D crossline-offset, inline-offset cube interpolation. 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

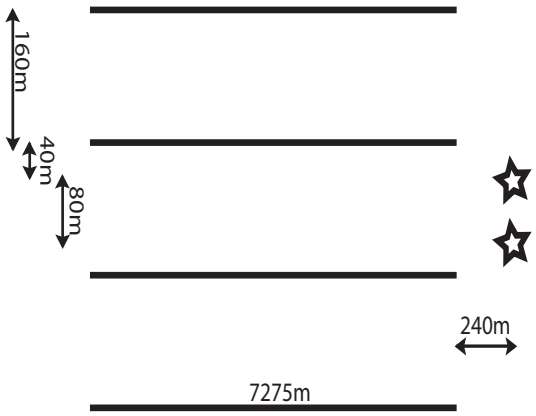

Figure 5.19: Schematic plan-view diagram of the acquisition geometry. **NR** fxNS/. realgeom

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

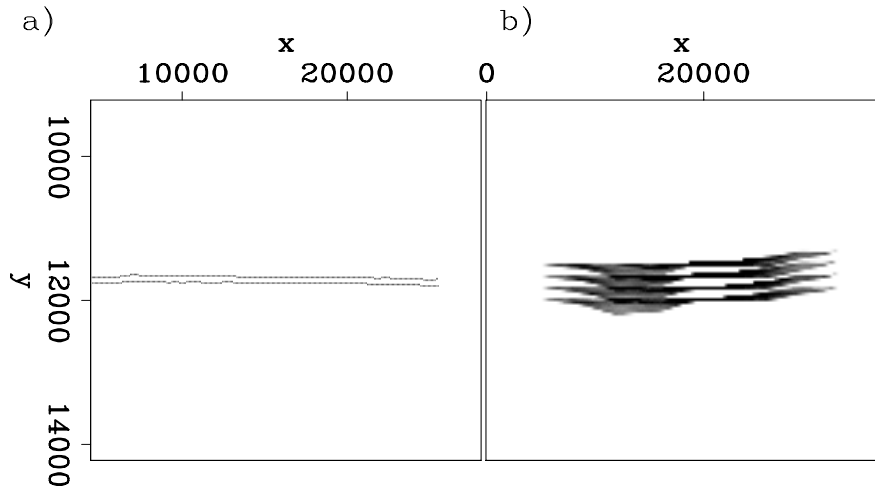

Figure 5.20: Source and receiver distributions of the input sail line: a) source positions; b) receiver position map. The sail line is relatively straight with minimal cable feathering. **NR** fxNS/. fieldsourec

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

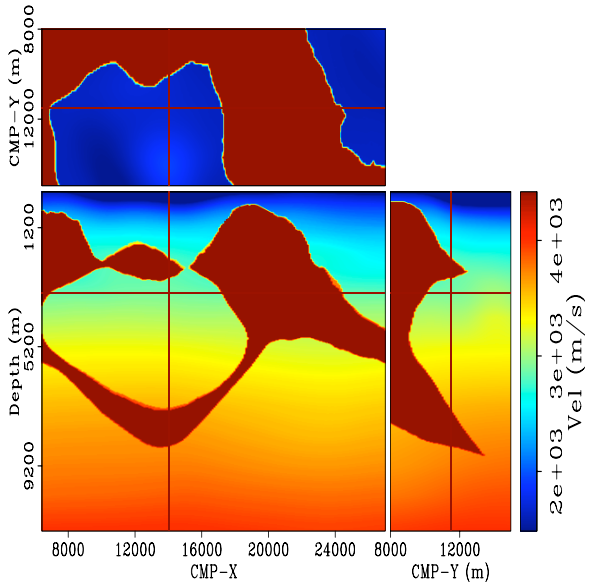

Figure 5.21: Migration velocity model of the dataset. Note the submarine canyon with a significant crossline dip. **NR** fxNS/. vmodel

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

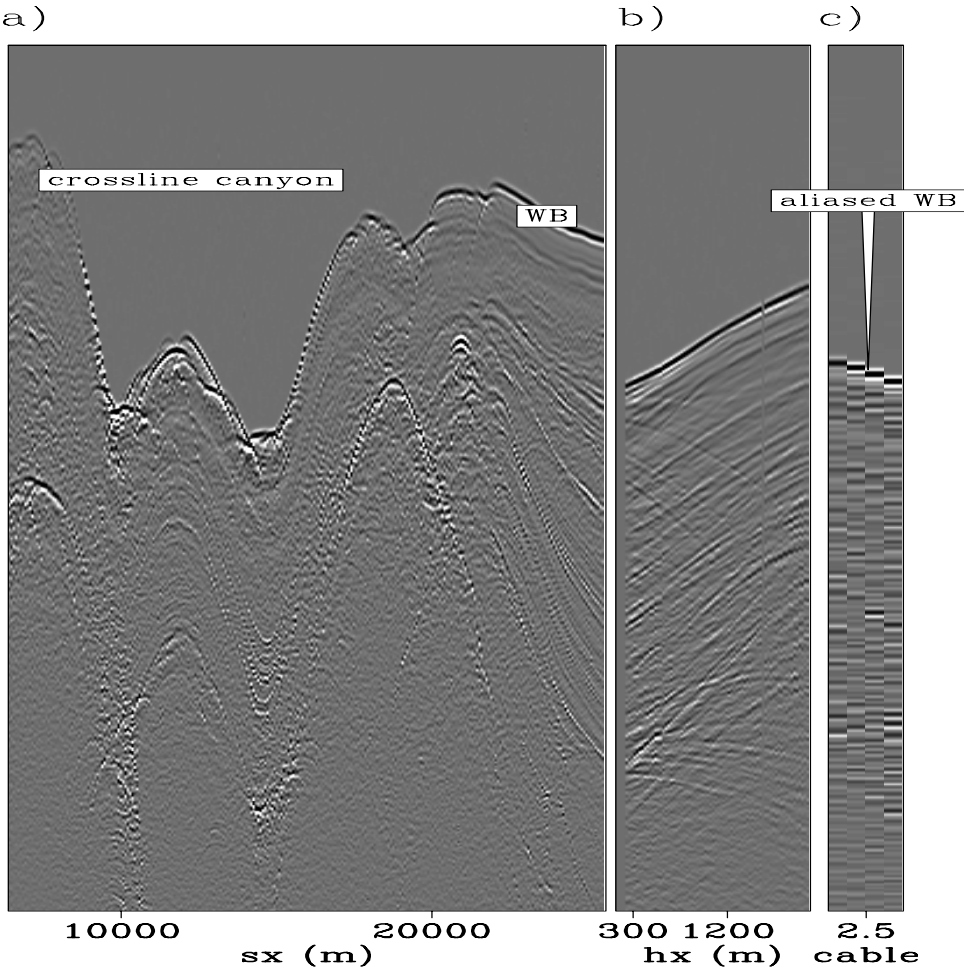

Figure 5.22: Input sail line: a) constant-offset section at 600 m inline offset and the second receiver cable; b) second cable of a shot at 15,800 m; c) crossline-offset gather at 600 m inline offset and a source position of 15,800 m. The data have been NMO corrected, and a gain has been applied. **NR**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

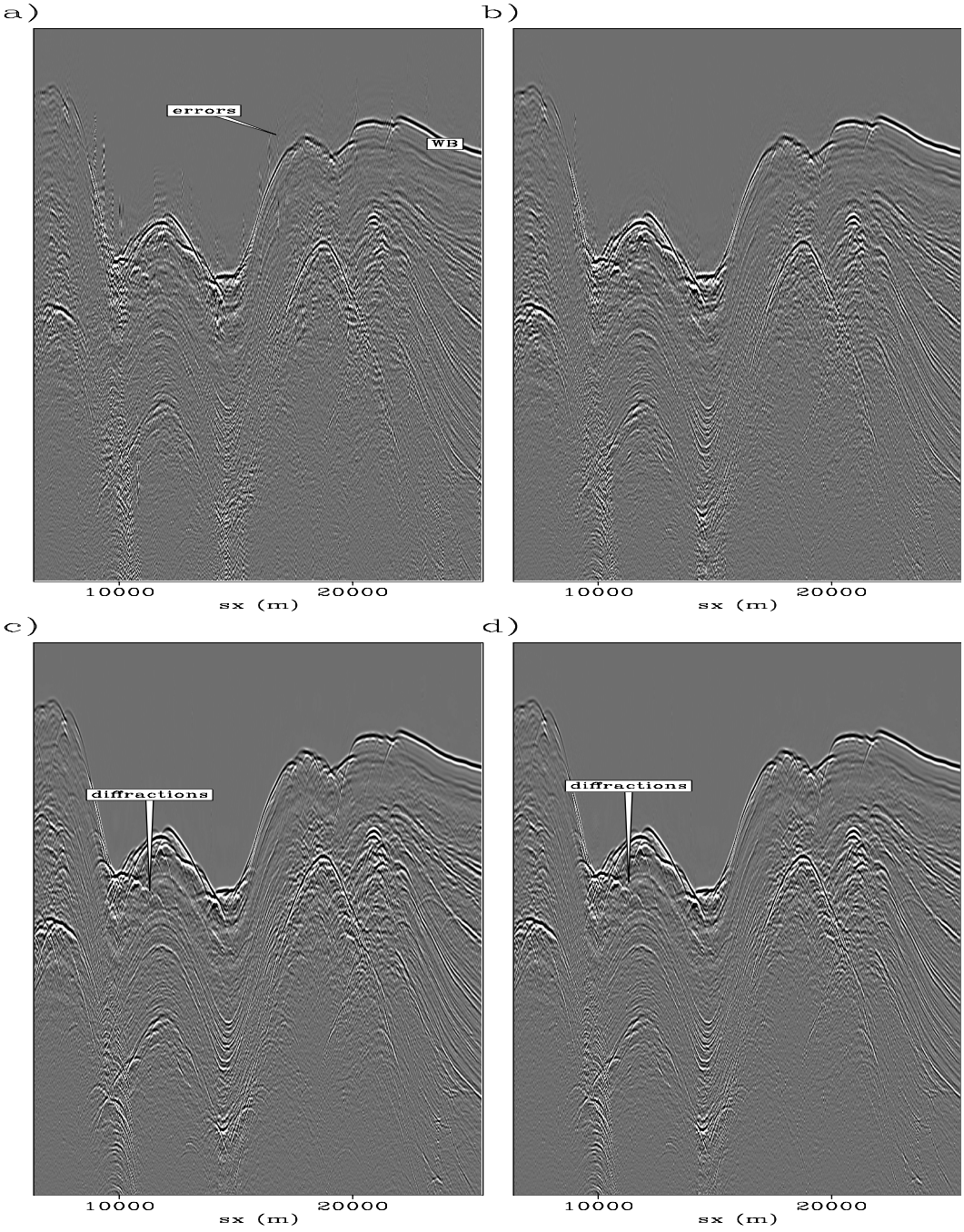

Figure 5.23: Interpolation of sources in a constant-offset section by a factor of three: a) 2D b) 3D (inline source, inline offset); c) 3D (inline source, crossline offset), d) 4D. The differences in this view are minor, with slight amounts of energy appearing before the water-bottom in the 2D interpolation.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

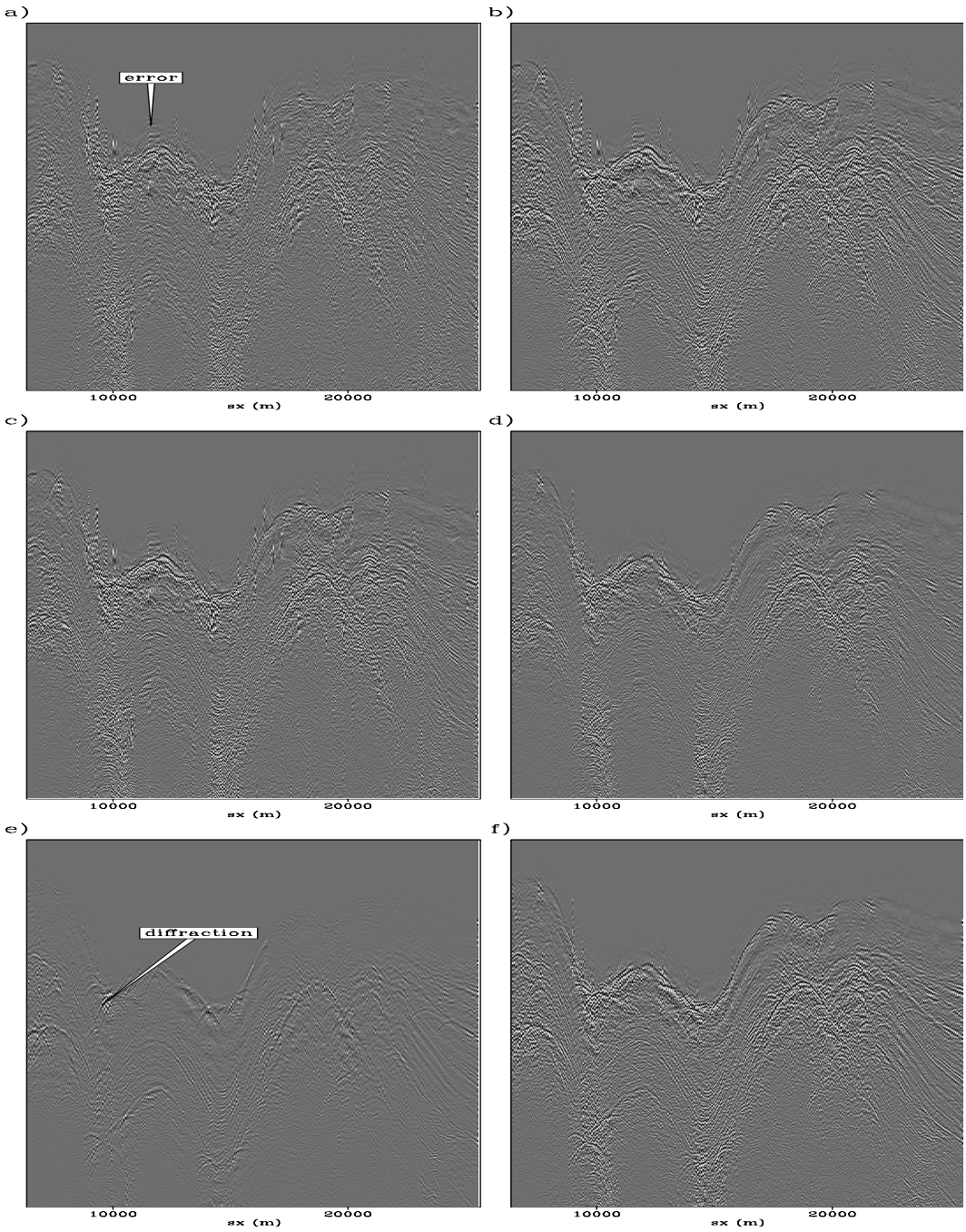

Figure 5.24: Differences between images in Figure 5.23. a) 2D vs. 3D crossline offset interpolation b) 2D vs. 3D (inline source, inline offset); c) 2D vs. 4D; d) 3D crossline offset vs. 4D; e) 3D inline offset vs. 4D; f) 3D inline vs. 3D crossline. **NR** fxNS/. fieldinlinecoffdiffann

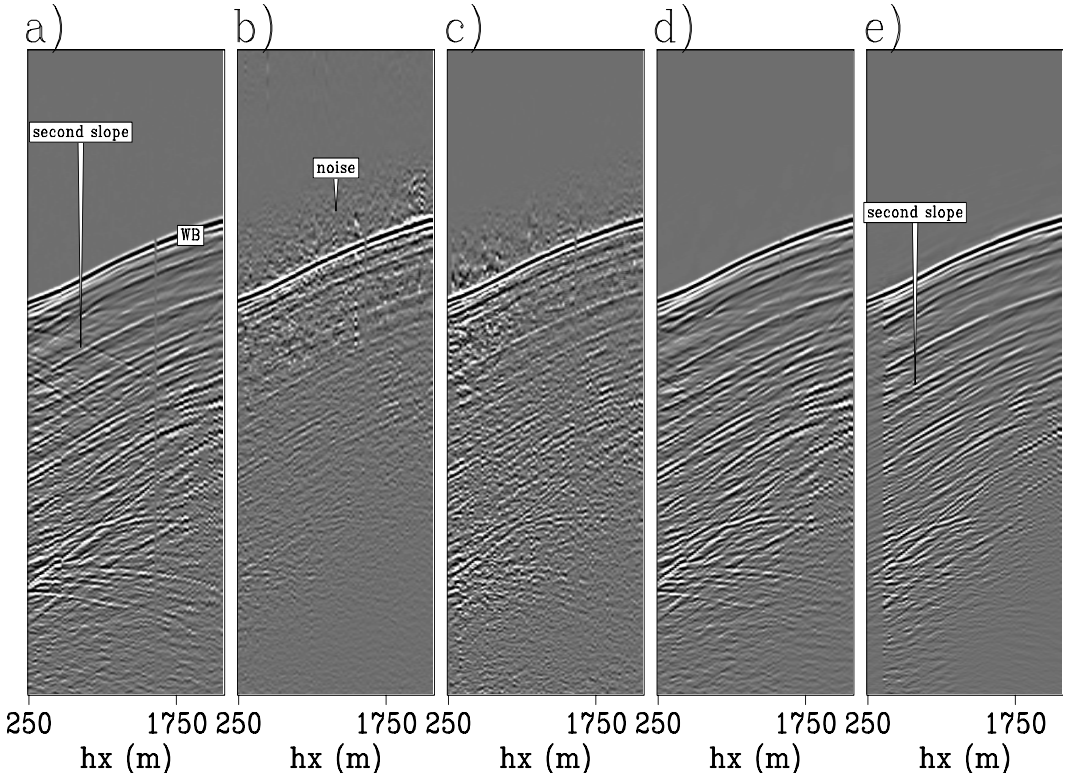

Figure 5.25: Near offsets of interpolated shot gathers: a) nearest recorded shot gather from 25 m away; b) 2D constant-offset-section-based interpolation; c) 3D (inline-source, crossline-offset) interpolation; d) 3D (inline-source, inline-offset) interpolation; e) 4D. The differences among the results 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

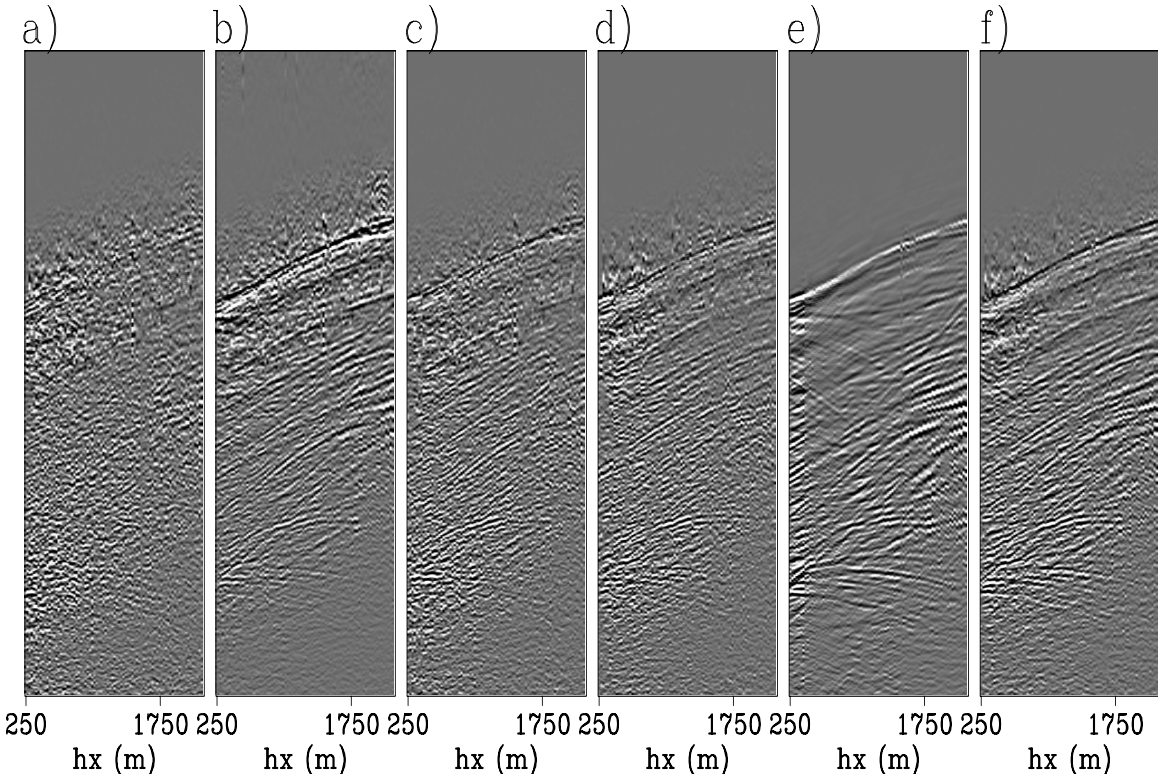

Figure 5.26: Differences between images in Figure 5.25. a) 2D vs. 3D crossline offset interpolation b) 2D vs. 3D (inline source, inline offset); c) 2D vs. 4D; d) 3D crossline offset vs. 4D; e) 3D inline offset vs. 4D; f) 3D inline vs. 3D crossline. The 3D inline and 4D results show the greatest coherent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

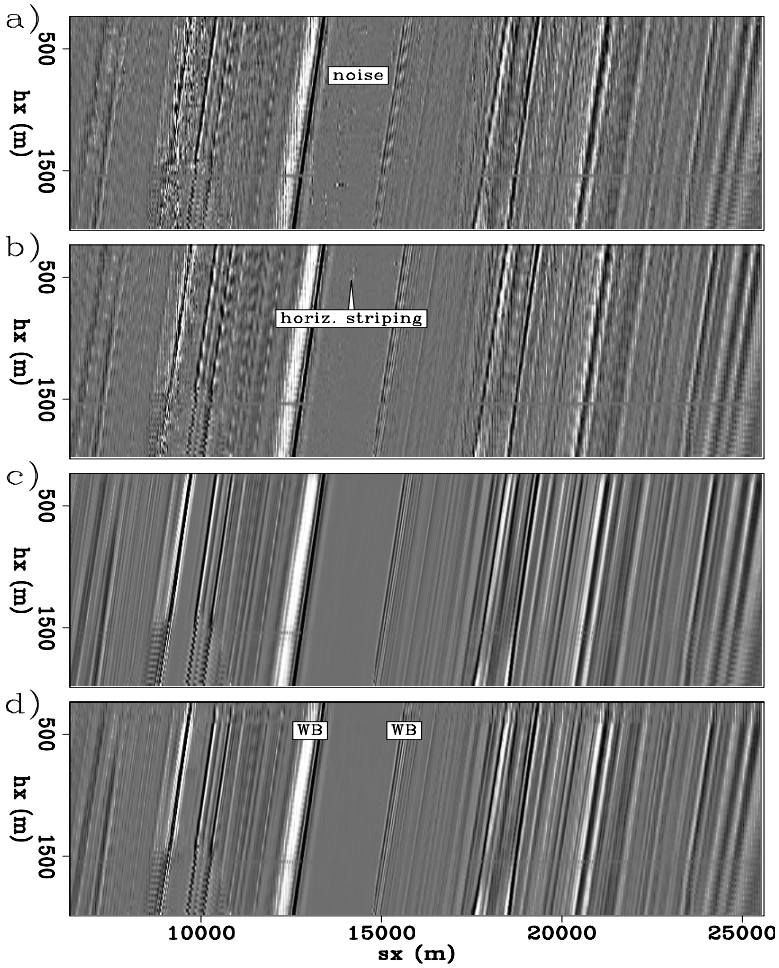

Figure 5.27: Time slices through inline-source, inline-offset cubes interpolated along the inline-source direction by a factor of three. a) 2D constant-offset-section-based interpolation; b) 3D (inline-source, crossline-offset) interpolation; c) 3D (inline-source, inline-offset) interpolation; d) 4D.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

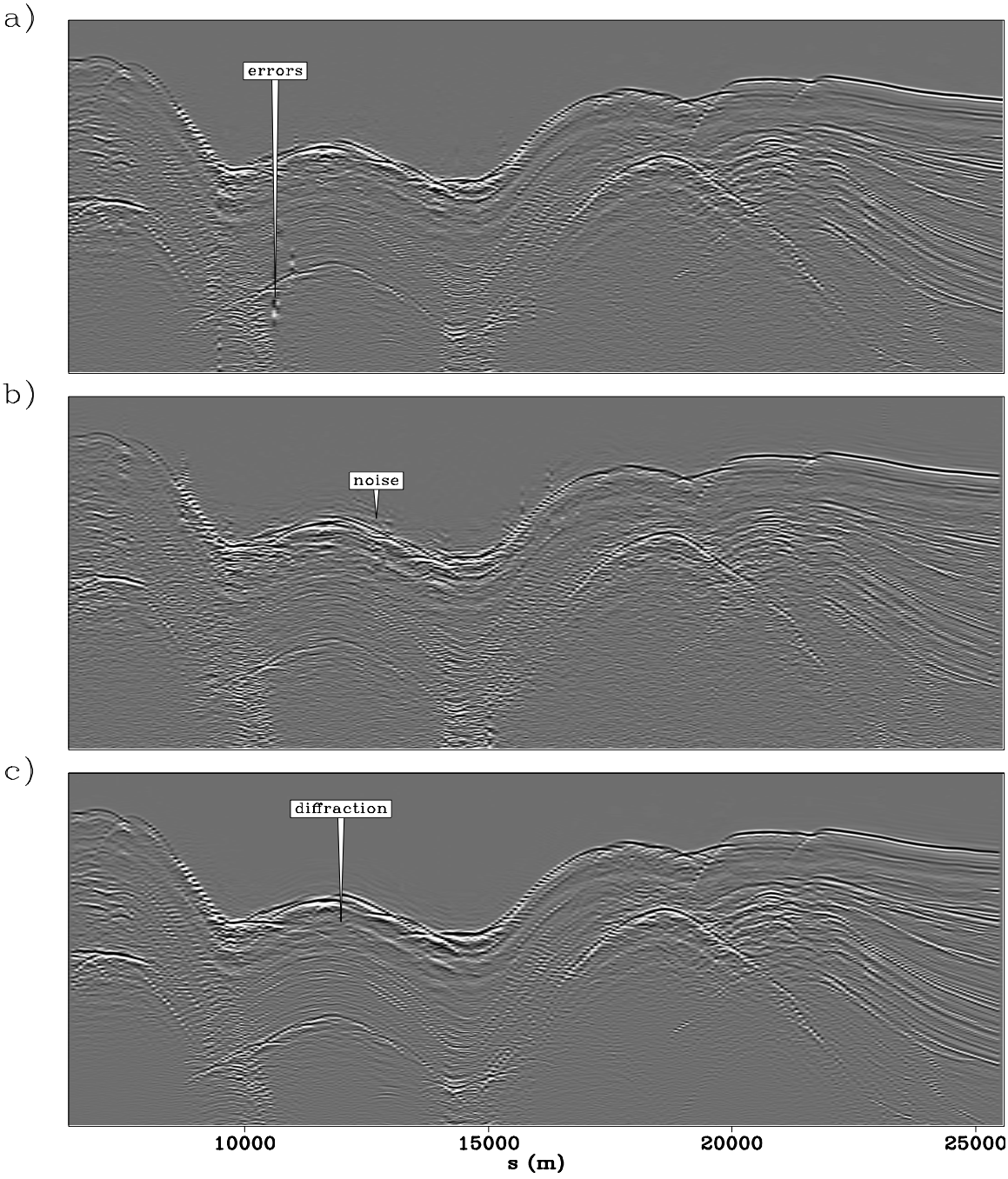

Figure 5.28: Receiver cable interpolation of field data. a) 3D (inline-offset, crossline-offset); b) 3D (inline-source, crossline-offset); c) 4D interpolation. The 4D result shows the most continuity, with no noise on the traces ad little anticausal noise. **NR** fxNS/. fieldxlinecoffann

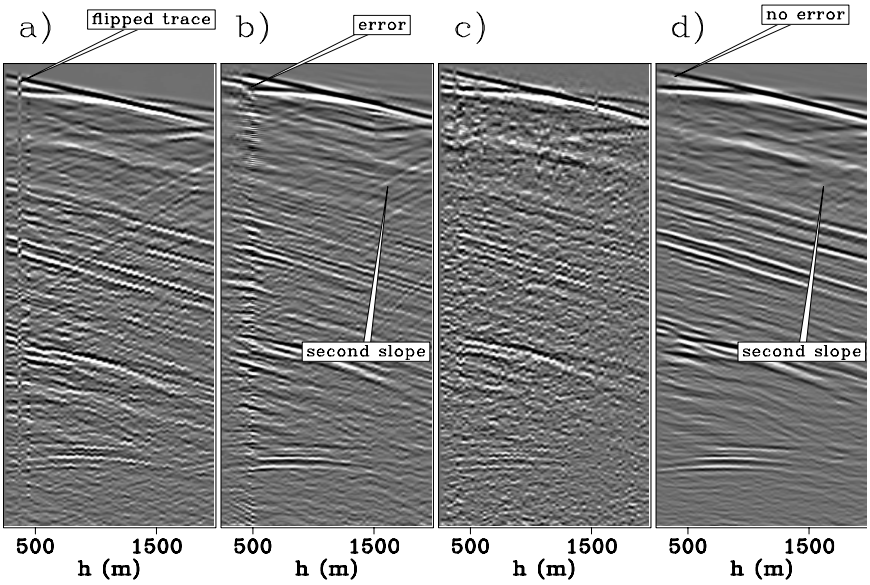

Figure 5.29: An interpolated receiver cable between the second and third cable, for a single shot. a) recorded data from the third cable; b) 3D (crossline-offset, inline-offset) shot-by-shot interpolation; c) 3D (crossline-offset, inline-shot) inline offset-by-offset interpolation; d) 4D interpolation. 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

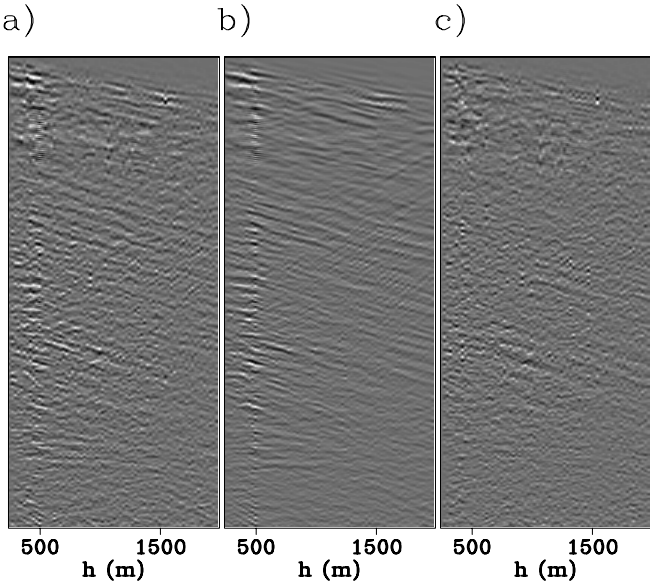

Figure 5.30: Differences between images in Figure 5.30. a) 3D inline offset, crossline offset vs. 3D inline source, crossline offset; b) 3D inline offset, crossline offset vs. 4D; c) 3D crossline offset, inline source vs. 4D interpolation. The 4D result is missing of the the second slope but also fortunately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

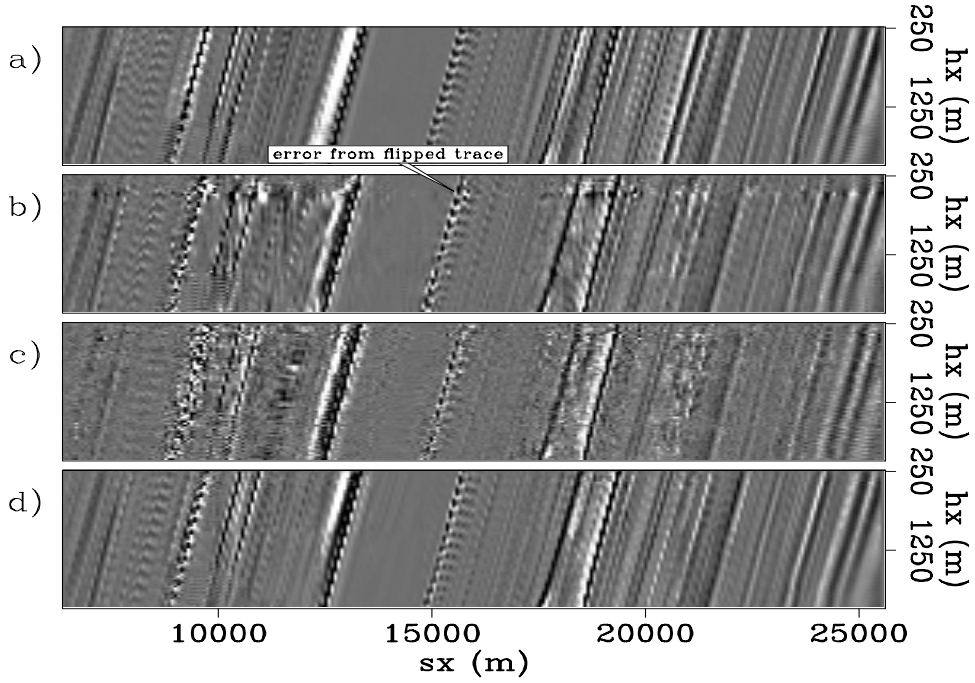

Figure 5.31: Time slice from an inline source-receiver cube of an interpolated receiver cable. a) recorded second receiver cable; b) 3D inline-source, inline-offset interpolation; c) 3D inline-source, crossline-offset interpolation; d) 4D interpolation. The 4D result is definitely the best for this view.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

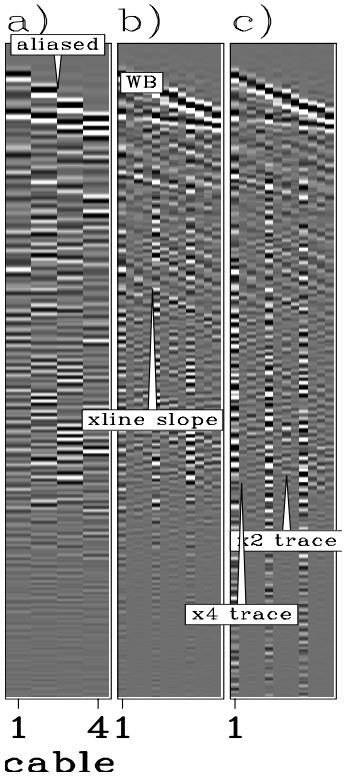

Figure 5.32: Crossline-offset sections of the interpolated data. a) recorded data; b) recorded data with interpolated cables; c) interpolated shot with interpolated cables. The amplitudes between the recorded traces and the interpolated traces becomes apparent at later times. **NR** fxNS/. fielditercxoffann

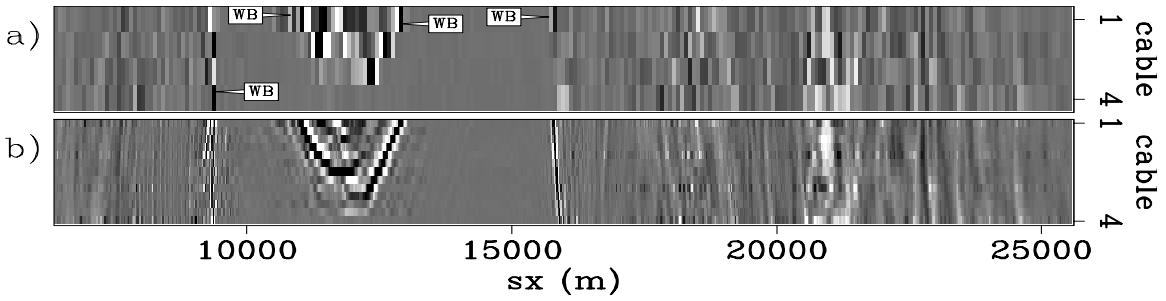

Figure 5.33: Time slice through an inline-source, crossline-offset cube. a) original recorded data; b) data interpolated by a factor of three along the inline-source and a factor of four in the crossline-offset direction. The aliased water-bottom reflection is smoothly interpolated. **NR** fxNS/. fielditerxssliceann

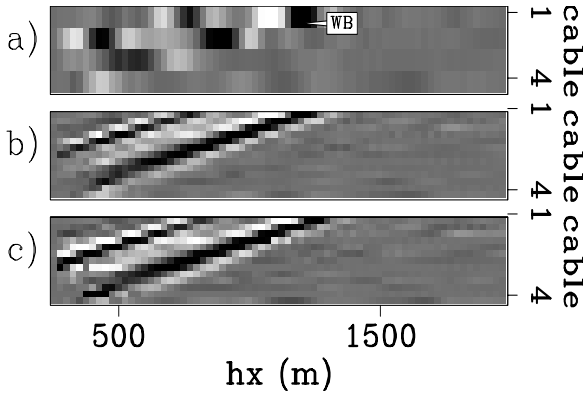

Figure 5.34: Time slice through a single shot (inline-offset, crossline-offset) cube. a) original recorded shot; b) original shot with interpolated receiver cables; c) interpolated shot with interpolated receiver cables. The source interpolation does not appear to produce a degraded result. **NR** fxNS/.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

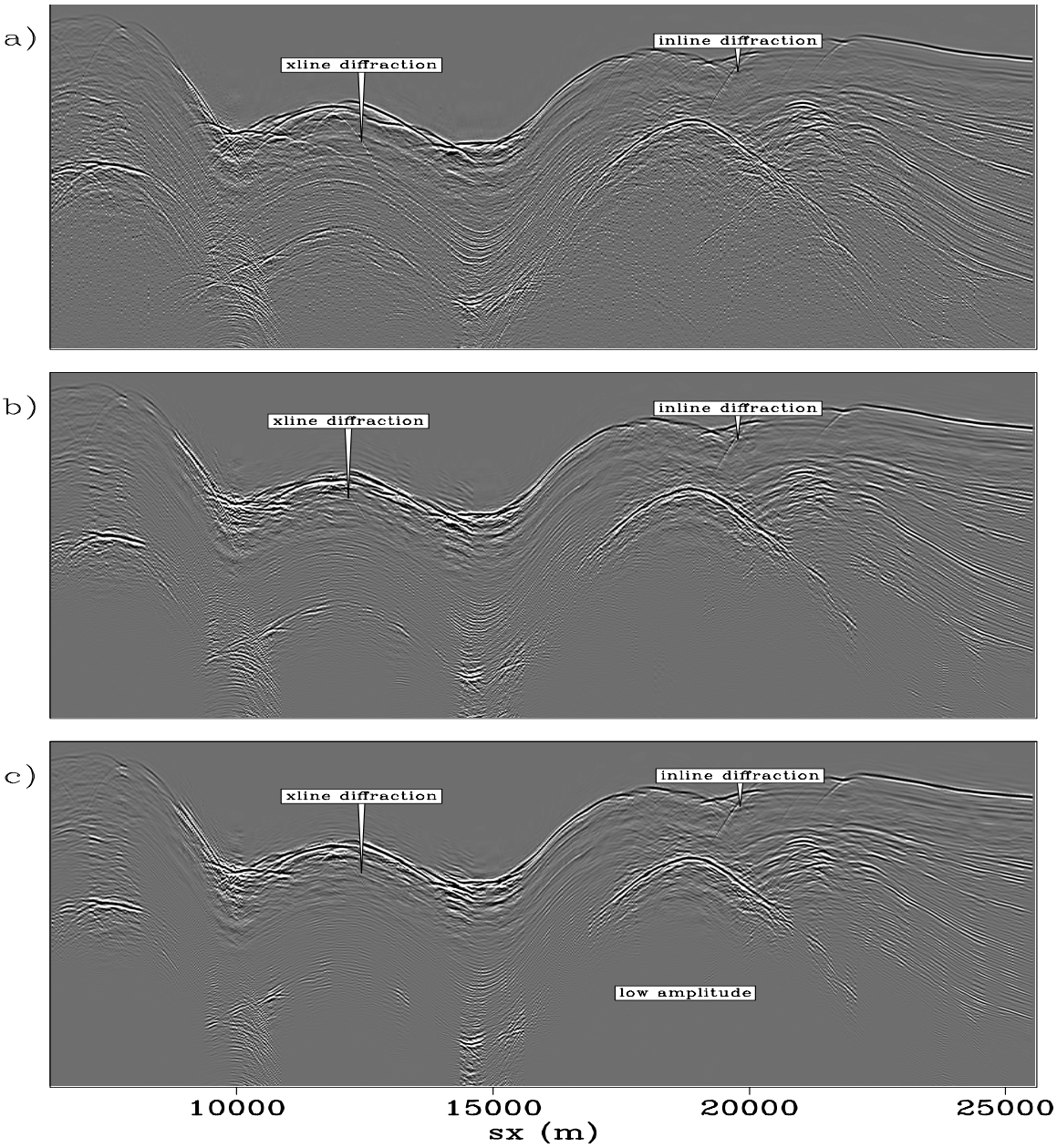

Figure 5.35: Constant-inline-offset sections from the interpolated data: a) along the second recorded receiver cable with interpolated sources; b) along a receiver cable interpolated by a factor of two (between the second and third cables); c) along a receiver cable interpolated by a factor of four (between (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

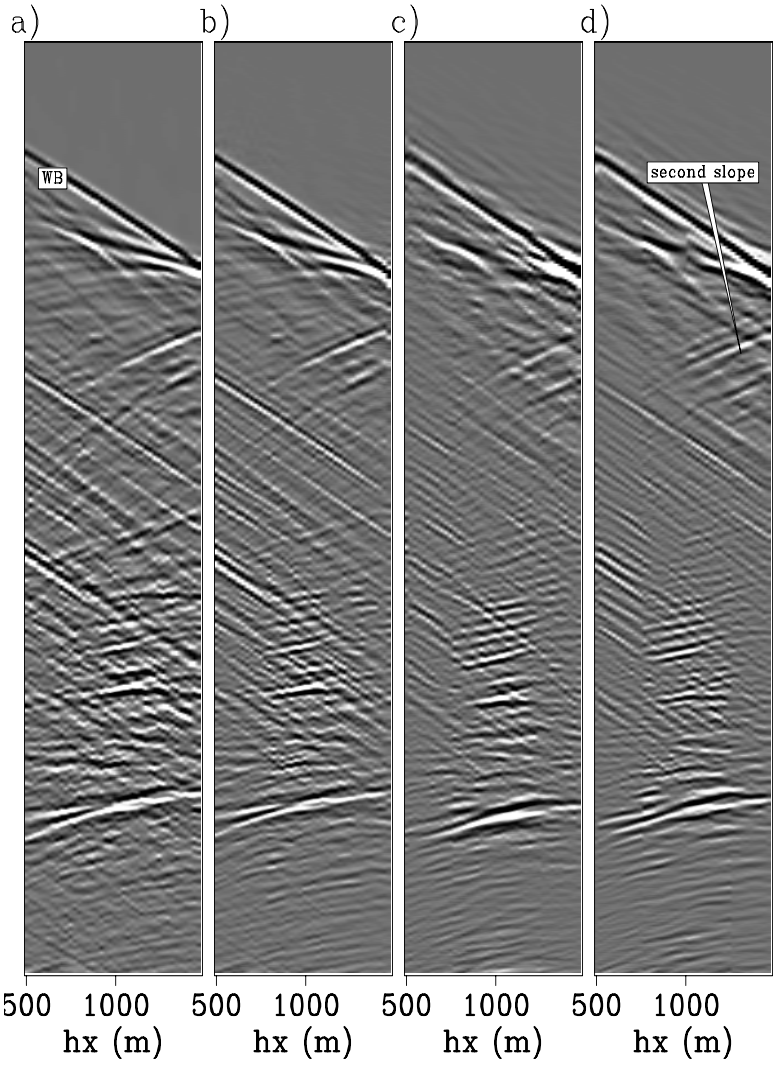

Figure 5.36: Interpolated receiver cables. a) originally recorded receiver cable; b) receiver cable from an interpolated shot; c) receiver cable interpolated by a factor of two from interpolated shot; d) receiver cable interpolated by a factor of four from the interpolated shot. This shot is from the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

Chapter 6

Conclusions

This thesis has focused on using nonstationary prediction-error filters in multidimensional interpolation. Where it innovates is in how the prediction-error filters are generated, the domain in which they operate, and the sampling of the data being interpolated. I developed three distinct approaches that are catered to irregularly-sampled data, data with large near-offset gaps, and the large amount of multidimensional interpolation required for 3D prestack marine data.

Most approaches to interpolating irregularly-sampled seismic data are transform, moveout, or migration-based approaches. I addressed the problem of irregularly-sampled data with prediction-error filters by using multiple regridded copies of the data to estimate a PEF. This method performed reasonably well on stationary synthetic data, and also on the spatially-variable quarter-dome data. Both of these datasets are relatively low-frequency and were randomly sampled. When this method was applied to field 2D land data, the performance was acceptable, but the acquisition of these data was far from random. One obvious improvement to this method would be to automatically choose optimal scales on which to estimate the PEF and to weigh data in the estimation based on how well they are characterized at each scale.

I conclude from these experiments that while theoretically pleasing, randomly-sampled synthetic data do not serve as a particularly useful analogue for most irregularly-sampled field data. Field data are typically sampled along tracks with deviations therefrom, be it the source lines and receiver cables for a 3D land survey or the smoothly varying sail lines and receiver cables of a marine survey. Both land and marine data are plagued by large gaps and undersampled axes, not random sampling along all spatial axes.

One such large gap in marine seismic data is a near-offset gap between the towed source and the nearest receiver, a problem usually addressed by normal-moveout or a Radon-transform-based interpolation. Instead of using information from the nearby primaries as in a Radon or an RMS-velocity-based method, I generated pseudoprimaries by using active-source interferometry, in which all receivers are crosscorrelated for each shot so that the correlation of primaries with free-surface multiples creates pseudoprimaries with a virtual source location at one of the receivers. These data are generated at a wider range of offsets than the original recorded data, including the near-offset information absent from the original recording. I used these pseudoprimaries as training data for nonstationary PEFs, both in time and space and in frequency and space.

This method of interpolating the large near-offset gap using a PEF estimated on pseudoprimaries works well on synthetic data, where I introduce a large gap, for the most part correctly reconstruct the missing data. The benefit of using PEFs in this process is that spurious events that do not correspond to a correlation between a multiple and a primary are often removed from the result, the squared wavelet of the pseudoprimaries is ignored, and the signal-to-noise ratio of these near offsets considerably improves. While the method works well on synthetic data, it had somewhat less success with the field data. Illumination issues with the multiple reflections also appear in the pseudoprimaries, causing steeply-dipping reflections to be comparatively weak in amplitude. Many of the methods used in signal/noise separation could be helpful here, as the training data used contain both coherent signal and coherent and incoherent noise.

While this method works well in 2D scenarios, the prospects for 3D are limited by the limited source distribution in the crossline. Much as with both 3D SRME and passive interferometric methods, pseudoprimary generation is highly dependent upon adequate source coverage. With the advent of wider-azimuth marine data, the prospects for keeping this method strictly data-driven improve, although the need for it may decrease.

The last chapter of this thesis is the most practical. The jump from 2D to 3D SRME methods requires a large amount of data to be created, both extrapolated an interpolated. Most 3D SRME implementations rely upon moveout or partial prestack migration operators to generate these data along the inline source, crossline source, and crossline receiver axes. I introduce nonstationary frequency-space interpolation, where the assumption in Spitz interpolation is used with nonstationary multidimensional prediction-error filters to interpolate data in anywhere from two to five dimensions. I first compare a nonstationary PEF-based methods applied frequency-by-frequency in different domains with different dimensionalities for a synthetic dataset, and conclude that for inline source interpolation, a strictly inline approach produces the best result. Meanwhile, a source-by-source 3D interpolation of receiver cables gives slightly better results than does a full 4D approach.

With this nonstationary frequency domain approach to field data with only four cables, the inline source interpolation is again best solved as a strictly inline problem, while the crossline receiver interpolation produces mixed results. A 4D interpolation produces a more robust result while a 3D shot-by-shot interpolation produces a more detailed result that is more easily compromised by noise.

Finally, iterating inline source interpolation with crossline receiver interpolation is able to produce data at densities required for 3D SRME. The drop in quality of the data is not so much caused by the assumption of previously interpolated data as known data, but more the jump in difficulty from inline-source to crossline-receiver interpolation. I conclude that interpolating many of the large factors of data required for full prestack interpolation is possible using prediction-error filters, in part because the interpolation takes place along multiple axes, so the assumptions behind the

interpolation are less stretched.

This thesis has been based upon the idea of using a nonstationary PEF as a container for useful information from training data that are not good enough to directly substitute for missing data. I have made three choices of training data, all that approximate the data differently. The choice of training data should be considered specific to the task at hand, both in terms of the character of the data in question and the sampling of that data, and is an open framework for use in many other situations.

Bibliography

Abma, R. and N. Kabir, 2005, 3d interpolation of irregular data with a pocs algorithm, 2150–2153. Soc. of Expl. Geophys.

Artman, B., 2007, Passive seismic imaging: PhD thesis, Stanford University.

Baumstein, A., 2004, 3D inverse shot record DmO: A tool for shot-domain data regularization: 74th Ann. Internat. Mtg., 2001–2004, Soc. of Expl. Geophys.

Baumstein, A. and M. Hadidi, 2006, 3d surface-related multiple elimination: Data reconstruction and application to field data: Geophysics, E25–E33, Soc. of Expl. Geophys.

Baumstein, A., D. L. H. K. D. A. and T. G. Farrington, 2006, Attenuating diffracted multiples with 3d srme - a feasibility study, F021. Eur. Assn. Geosci. Eng.

Berkhout, A. and D. Verschuur, 2003, Transformation of multiples into primary reflections: 73rd Ann. Internat. Mtg., 1925–1928, Soc. of Expl. Geophys.

Berkhout, A. J., 1999, Multiple removal based on the feedback model: The Leading Edge, **18**, 127–131.

Berkhout, A. J. and D. Verschuur, 2005, Transformation of multiples into primary reflections, 1039–1042. Soc. of Expl. Geophys.

Berkhout, A. J. and D. J. Verschuur, 1994, Multiple technology: Part 2, migration of multiple reflections: 64th Ann. Internat. Mtg, 1497–1500, Soc. of Expl. Geophys.

Biondi, B., 2006, 3d seismic imaging: Covington Group.

Biondi, B. and I. Vlad, 2002, Amplitude preserving prestack imaging of irregularly sampled 3D data: 72nd Ann. Internat. Mtg, 2170–2173, Soc. of Expl. Geophys.

Burg, J. P., 1975, Maximum entropy spectral analysis: SEP-Report, **6**, 0–0.

Chemingui, N., 1999, Imaging irregularly sampled 3D prestacked data: PhD thesis,

Stanford University.

Claerbout, J., 1998, Multidimensional recursive filters via a helix: Geophysics, **63**, 1532–1541.

Claerbout, J. F., 1968, Synthesis of a layered medium from its acoustic transmission response: Geophysics, **33**, 264–269.

———, 1976, Fundamentals of geophysical data processing: Blackwell.

———, 1992, Earth soundings analysis: Processing versus inversion: Blackwell Scientic Publications.

———, 2004, Image Estimation by Example: `http://sepwww.stanford.edu/sep/prof/index.html`.

Clapp, R. G., 2001, Multiple realizations: Model variance and data uncertainty: SEP-Report, **108**, 147–158.

———, 2003, AMO regularization: Effective approximate inverses for amplitude preservation: SEP-Report, **114**, 159–170.

Clapp, R. G., S. Fomel, S. Crawley, and J. F. Claerbout, 1999, Directional smoothing of non-stationary filters: SEP-Report, **100**, 197–209.

Cole, S. P., 1995, Passive seismic and drill-bit experiments using 2-D arrays: PhD thesis, Stanford University.

Crawley, S., 2000, Seismic trace interpolation with nonstationary prediction-error filters: PhD thesis, Stanford University.

Curry, W., 2002, Non-stationary, multi-scale prediction-error filters and irregularly sampled data: SEP-Report, **111**, 327–337.

———, 2003, Interpolation of irregularly sampled data with nonstationary, multiscale prediction-error filters, 1913–1916. Soc. of Expl. Geophys.

Curry, W. and M. Brown, 2001, A new multiscale prediction-error filter for sparse data interpolation: SEP-Report, **110**, 113–122.

Curry, W. J. and G. Shan, 2006, Interpolation of near offsets with multiples and prediction-error filters, G026], keywords=. Eur. Assn. Geosci. Eng.

Dragoset, B., 1999, A practical approach to surface multiple attenuation: The Leading Edge, **18**, 104–108.

Dragoset, W. H. and Z. Jericevic, 1998, Some remarks on surface multiple attenuation:

Geophysics, **63**, 772–789.

Fomel, S., 2001, Three-dimensional seismic data regularization: PhD thesis, Stanford University.

———, 2002, Applications of plane-wave destruction filters: Geophysics, **67**, 1946–1960.

Guitton, A., 2003, Multiple attenuation with multidimensional prediction-error filters: SEP-Report, **113**, 57–74.

Guitton, A. and G. Cambois, 1999, Multiple elimination using a pattern-recognition technique: The Leading Edge, **18**, 92–98.

Gulunay, N. and R. Chambers, 1997, Generalized f-k domain trace interpolation: 67th Ann. Internat. Mtg, 1100–1103, Soc. of Expl. Geophys.

Gulunay, N. and R. E. Chambers, 1996, Unaliased f-k domain trace interpolation (UfKi): 66th Ann. Internat. Mtg, 1461–1464, Soc. of Expl. Geophys.

Hadidi, M. T., M. Sabih, D. E. Johnston, and C. Calderon-Macias, 1999, Mobil's results for the 1997 workshop on multiple attenuation: The Leading Edge, **18**, 100–103.

Hargreaves, N., B. V. W. R. W. and D. Trad, 2005, Multiple attenuation using an apex-shifted radon transform, 385–388. Soc. of Expl. Geophys.

Hestenes, M. R. and E. L. Stiefel, 1952, Methods of conjugate gradients for solving linear systems.: J. Res. Nat. Bur. Standards, 409–436.

Lamont, M. G., B. M. Hartley, and N. F. Uren, 1999, Multiple attenuation using the mmo and isr preconditioning transforms: The Leading Edge, **18**, 110–114.

Levin, S., 2002, Prestack poststack 3D multiple prediction: 72nd Ann. Internat. Mtg, 2110–2113, Soc. of Expl. Geophys.

Liu, B. and M. Sacchi, 2001, Minimum weighted norm interpolation of seismic data with adaptive weights: 71st Ann. Internat. Mtg, 1921–1924, Soc. of Expl. Geophys.

Liu, B. and M. D. Sacchi, 2004, Minimum weighted norm interpolation of seismic records: Geophysics, **69**, 1560–1568.

Lokshtanov, D., 1999, Multiple suppression by data-consistent deconvolution: The Leading Edge, **18**, 115–119.

Margrave, G. F., 1998, Theory of nonstationary linear filtering in the Fourier domain

with application to time-variant filtering: Geophysics, **63**, 244–259.

Matson, K. H. and R. Abma, 2005, Fast 3d surface-related multiple elimination using azimuth moveout for multiples, 2064–2067. Soc. of Expl. Geophys.

Matson, K. H., D. Paschal, and A. B. Weglein, 1999, A comparison of three multiple-attenuation methods applied to a hard water-bottom data set: The Leading Edge, **18**, 120–126.

Reiter, E. C., M. N. Toksoz, T. H. Keho, and G. M. Purdy, 1991, Imaging with deep-water multiples: Geophysics, **56**, 1081–1086.

Riley, D. C. and J. F. Claerbout, 1976, 2-D multiple reflections: Geophysics, **41**, 592–620.

Robinson, E. A. and S. Treitel, 1967, Principles of digital Wiener filtering: Geophys. Prosp., **15**, 311–333.

Sacchi, M. D. and T. J. Ulrych, 1995, Model re-weighted least-squares Radon operators: 65th Ann. Internat. Mtg, 616–618, Soc. of Expl. Geophys.

Schonewille, M. A. and A. J. W. Duijndam, 1998, Efficient nonuniform Fourier and Radon filtering and reconstruction: 68th Ann. Internat. Mtg, 1692–1695, Soc. of Expl. Geophys.

Schuster, G., 2001, Theory of Daylight/Interferometric Imaging - Tutorial: 63rd Mtg., Session: A–32, Eur. Assn. Geosci. Eng.

Shan, G., 2003, Source-receiver migration of multiple reflections: 73rd Ann. Internat. Mtg., 1008–1011, Soc. of Expl. Geophys.

Shannon, C. E., 1949, Communication in the presense of noise: Proc. I.R.E., **37**, 10–21.

Shewchuk, J. R., 1994, An introduction to the conjugate gradient method without the agonizing pain: ftp://reports.adm.cs.cmu.edu/usr0/anon/1994/CMU-CS-94-125.ps.

Spitz, S., 1991, Seismic trace interpolation in the F-X domain: Geophysics, **56**, 785–794.

Thomson, D., G. H. H. M. and F. Herrmann, 2006, A parallel-windowed fast-discrete curvelet transform applied to seismic processing, 2767–2771. Soc. of Expl. Geophys.

Trad, D., 2002, Interpolation with migration operators: 72nd Ann. Internat. Mtg,

2174–2177, Soc. of Expl. Geophys.

Trad, D., T. J. Ulrych, and M. D. Sacchi, 2002, Accurate interpolation with high-resolution time-variant Radon transforms: Geophysics, **67**, 644–656.

Vermeer, G. J., 2002, 3-d seismic survey design: Soc. of Expl. Geophys.

Verschuur, D. J. and A. J. Berkhout, 2005, Reconstruction of sparsely sampled data using a high-resolution version of the focal transform, 2154–2157. Soc. of Expl. Geophys.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: Geophysics, **57**, 1166–1177.

Verschuur, D. J. and R. J. Prein, 1999, Multiple removal results from delft university: The Leading Edge, **18**, 86–91.

Wang, Y., 2002, Seismic trace interpolation in the f-x-y domain: Geophysics, **67**, 1232–1239.

Wang, Z. and Y. Li, 1994, Trace interpolation using wavelet transform: 64th Ann. Internat. Mtg, 729, Soc. of Expl. Geophys.

Weglein, A. B., 1999, How can the inverse-scattering method really predict and subtract all multiples from a multidimensional earth with absolutely no subsurface information?: The Leading Edge, **18**, 132–136.

Weisser, T., A. P. P. H. and R. Taylor, 2006, Wave equation multiple modelling: acquisition independent 3d srme: First Break, **24**, 75–80.

Wiener, N., 1964, Extrapolation, interpolation, and smoothing of stationary time series: The MIT Press.

Xu, S., Y. Zhang, D. Pham, and G. Lambare, 2005, Antileakage Fourier transform for seismic data regularization: Geophysics, **70**, V87–V95.

Yilmaz, O., 1987, Seismic Data Processing: Soc. of Expl. Geophys. (Video course available from SEG Publications Department).

Zhou, C. and G. T. Schuster, 1995, Quasi-random migration of 3-D field data: 65th Ann. Internat. Mtg, 1145–1148, Soc. of Expl. Geophys.

Zwartjes, P. and A. Gisolf, 2007, Fourier reconstruction with sparse inversion: Geophysical Prospecting, 199–221, Eur. Assn. Geosci. Eng.