The Graduate University for Advanced Studies
School of Multi Disciplinary Science
Department of Statistical Science
4-6-7 Minami-Azabu, Minato-ku, Tokyo, 106-8569 Japan

Principal Component Analysis and Local Regression Analysis on Acoustic Logging Data

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICAL SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF STATISTICAL SCIENCE

Shin'ichi Watanabe

June 2005

Preface

This thesis is an outcome after spending almost five years at the Institute of Statistical Mathematics as a student of the Graduate University for Advanced Studies. Since the author's main duties are to work for a company named Schlumberger K.K., Japan, a subsidiary of Schlumberger Limited that is the oilfield services company supplying technology and information solutions for customers in the international oil and gas industry domains, the total amount of time as being a student has been limited. However, in this inextensible time frame, the author learned various things and encountered many respectable persons.

In the author's working domains we deal with data obtained mainly from geophysical measurements that are utilized to find evidence of hydrocarbon in the underground. For these data various analysis methods are applied and new processing techniques are also attempted. This thesis focuses on statistical approaches to mainly single-well acoustic imaging data for 1) analyzing characteristics of these data and 2) denoising. Our ultimate goal for single-well acoustic imaging data is to delineate subsurface geological structure near a borehole. Although various processing techniques have been proposed and developed by multiple people, these imaging operations have not yet fully acknowledged as the standard way to visualize geological layers in the vicinity of wellbores.

This thesis is composed of the following contents: Chapter 1 is intended to introduce background of single-well acoustic imaging and notations. Chapter 2 attempts to study PCA (principal component analysis) and its related techniques such as PCA on-line and local PCA. We analyzed single-well acoustic data by these techniques. The core methodology of this thesis, local likelihood regression, appears in Chapter 3.

This chapter deals with a polynomial model whose predictor variable is a scalar and response variable has a vector shape. This model is also applied on real data from single-well acoustic logging. In Chapter 4, local approaches are attempted in another geophysical analysis domain. It is spatial interpolation called kriging. Both local constant and local linear regressions are considered between a scalar predictor variable and a scalar response variable. Lastly conclusive remarks are stated in Chapter 5.

Most chapters include MATLAB[1] codes that are used primary on Matlab6.5.1. Lately Matlab7 is also used to produce computational results.

[1]MATLAB is a registered trademark of The MathWorks, Inc.

Acknowledgments

I am grateful to Schlumberger K.K. (SKK) for granting this study. Especially thanks should go to Alain Brie, Tomoyoshi Ono, Dr. Takeshi Endo, and Dr. Hugues Djikpesse for their support and encouragement. Dr. Jakob B. U. Haldorsen is thanked for his MATLAB migration codes. I owe it to the oilfield glossary web pages developed by Schlumberger Limited to introduce technical terminologies.

I would like to express my deep gratitude to Prof. Shinto Eguchi and Dr. Mihoko Minami, the Institute of Statistical Mathematics and the Graduate University for Advanced Studies, for their continuous and patient advising activities. My thanks should go to many colleagues at the Graduate University for Advanced Studies.

In 1998, Dr. Yasuyuki Toyooka suddenly passed away. He was an author's adviser for M. Eng. at the faculty of engineering, Osaka University. His attitude toward statistics gave big influences to young people who were studying statistics there. His encouragements made for the author at those days were unforgettable in the author's mind. At his memorial services the author could meet Prof. Shinto Eguchi again after 14 years and this could be an initial starting point for the author to apply for the Graduate University for Advanced Studies.

Lastly I wish to thank my family members, Kazuko, Haruka, Maki, and Satoshi, for having cheered me up for five years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we introduce our motivation, some background concepts, and notations, since our data are coming form geophysical domains. Specifically single-well acoustic imaging techniques are described conceptually so that our data domain can be intuitively understood.

1.1 Motivation

Imaging near-borehole structure is a processing operation to delineate geological structural features near a well bore. It uses acoustic full-waveform data acquired at an array of receivers in an acoustic well logging tool and utilize techniques used in surface seismic processing techniques (Hornby [7]). Resultant images have finer resolution because higher frequencies are used in the acoustic well logging when comparing with acoustic frequencies used in acquisition for the surface seismic. This single-well acoustic imaging has been applied for mapping formation boundaries and fracture/fault shape. In general reflected/refracted signals coming from geological key features of interest are subtle. Therefore, they are buried in various component waves together with measurement noises in the acquired full-waveform data.

Our initial motivation is to denoise the acquired data efficiently by utilizing statistical features that the original data retain. First we studied PCA and local PCA, then we extended our study to localized likelihood approach with adaptive band-width

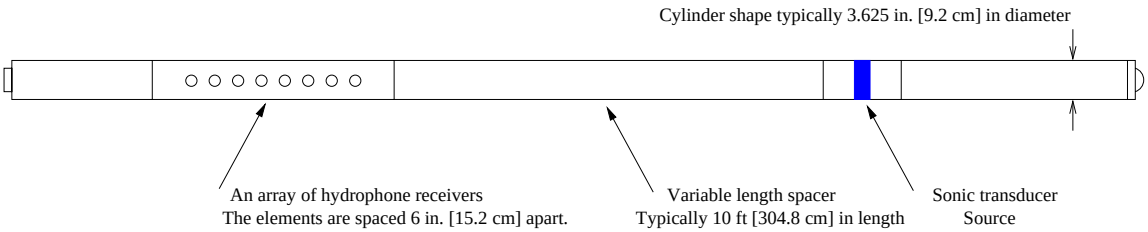

Figure 1.1: Schematic sonic logging tool: There exist one or more transmitters and an array of hydrophone receivers.

selection.

Another interest is to seek for some local approaches in the geophysical data processing domain. This aims at proposing equivalent or better processing approaches by making use of local information that may be available in the original data. For this sake we studied spatial interpolation techniques named kriging and find an area where local regression may be applicable for the computational simplicity.

1.2 Single-well Acoustic Imaging

In order to grasp subsurface geological features near a borehole, both acquisition and processing techniques have been developed as single-well acoustic imaging (Esmersoy *et al.* [4]) using an acoustic source and an array of hydrophone receivers deployed in a single well.

1.2.1 Sonic Logging Tool

A typical acquisition hardware is depicted in Figure 1.1 and we call this hardware sonic logging tool. The sonic logging tool has 1) one or more transmitters whose driving frequency range is approximately 5 to 15 kHz; and 2) an array of hydrophone receivers.

Our primary concern is to delineate major bed boundaries in the vicinity of well-bores with using reflected signals as shown in Figure 1.2. The single-well acoustic imaging may be used to fracture/fault identifications. In principle data acquisition is

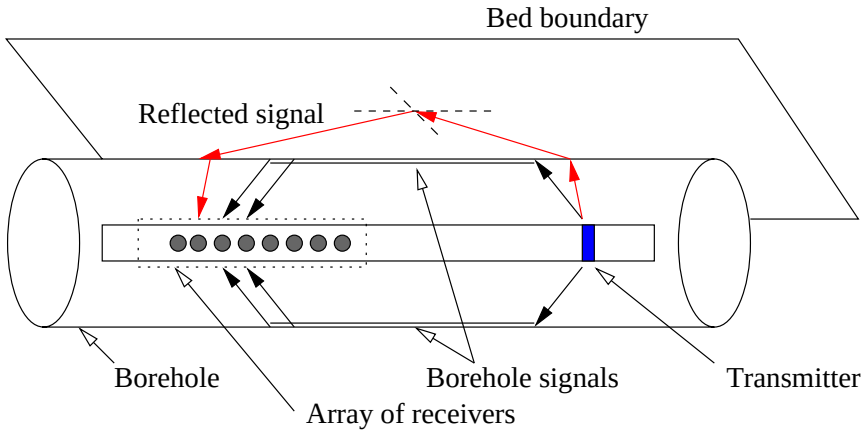

Figure 1.2: Sonic imaging principle: Acoustic impedance contrasts, such as bed boundaries, reflect sonic waves back to the borehole where they are detected by the receiver array.

composed of the following steps:

1. At a given tool position the transmitter fires and then the receivers acquire data. Both borehole and reflected signals are recorded in the data.

2. The tool moves to a next tool position. Offset length from the previous tool position is matched with the inter-receiver spacing.

3. Go to Step 1 to acquire another set of data.

This single-well acoustic imaging fills a resolution gap that lies in between seismic methods and borehole logs. This means that the frequency range used in single-well acoustic imaging is higher than that from surface seismic, thereby image resolution gets improved. However the depth of investigation gets shorter but longer than that from borehole logs. These features are illustrated in Figure 1.3. By combining and integrating these seismic, single-well acoustic imaging, and borehole logs, it is possible to capture a shape of geological features of interest accurately.

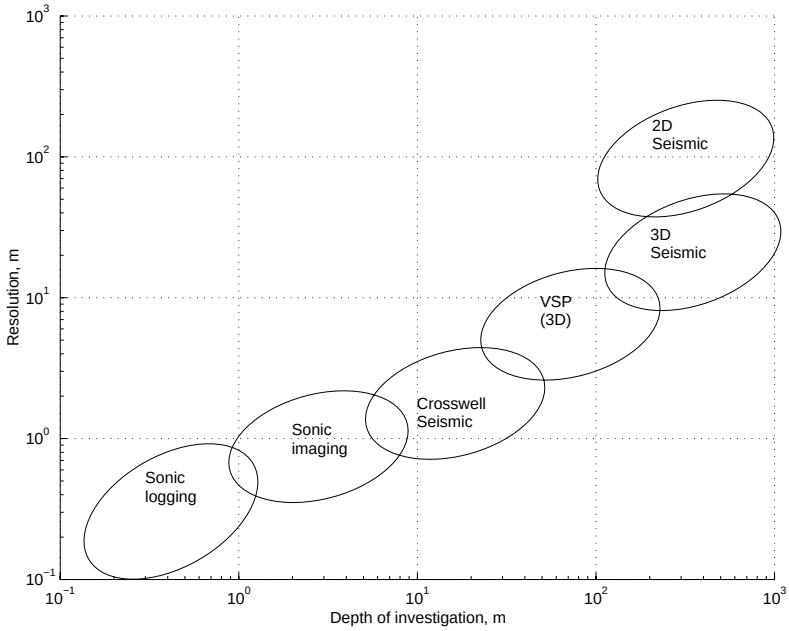

Figure 1.3: Depth of investigation versus resolution for various geophysical data. Modified from Chang *et al.* [2]. Sonic imaging fits in between sonic logging and seismic methods.

1.2.2 Software Processing

Processing techniques applied to single-well acoustic imaging are analogous to a 2D surface seismic survey with frequencies scaled up and acquisition geometry scaled by two to three orders of magnitude. Basically they consist of the following processing components:

1. Preprocessing or filtering to eliminate unwanted signals such as borehole signals.
2. Migration to place reflector signal events at right positions around a well while taking formation velocity (or its reciprocal called slowness) into account.
3. Presentation to map and display migrated data along the well-trajectory.

In this thesis we focus the preprocessing stage and apply various statistical approaches on common-offset data that have a two-dimensional, (space, time), shape.

Tang and Cheng [14] gives a broadband perspective for acoustic well-logging and its applications including single-well acoustic imaging.

1.2.3 Key Technical Terms

Key terms are briefly described in this section.

- Borehole is a wellbore, including the openhole or uncased portion of a well.
- Openhole is the uncased section of a well. In a cased section, steel pipes are lowered and cemented in place to protect the well.
- Log is the measurement versus depth or time, or both, of one or more physical quantities in or around a well. Borehole is the wellbore or uncased portion of the well.
- Logging tool is the downhole hardware used to make a log. Tools are typically cylinders from 1.5 to 5 in. [3.8 to 12.7 cm] in diameter.
- Trace is a single waveform recorded by a single receiver in the sonic logging tool.

- Common-offset: Pertaining to traces that have the same offset in between transmitter and receiver. Figure 1.4 illustrates a common-offset gather made from the fixed offset length of Tx-R1, where Tx and R1 denote the transmitter and the 1^{st} receiver in the receiver array.

- Variable-density log (VDL) is a presentation of the acoustic waveform at a receiver of sonic measurements, in which the amplitude is presented in color or the shades of a gray scale. Figure 1.5 illustrates two types of acoustic waveform for the same data set. Figure 1.5(a) shows wiggle format that has a collection of traces. In this example a single trace is repeated to form the collection. In Figure 1.5(b) corresponding VDL is presented with a gray scale. In this VDL, the higher the amplitude value is, the darker assigned color becomes.

- Seismic: Pertaining to waves of elastic energy, such as that transmitted by P- and S-waves, in the frequency range of approximately 1 to 100 Hz.

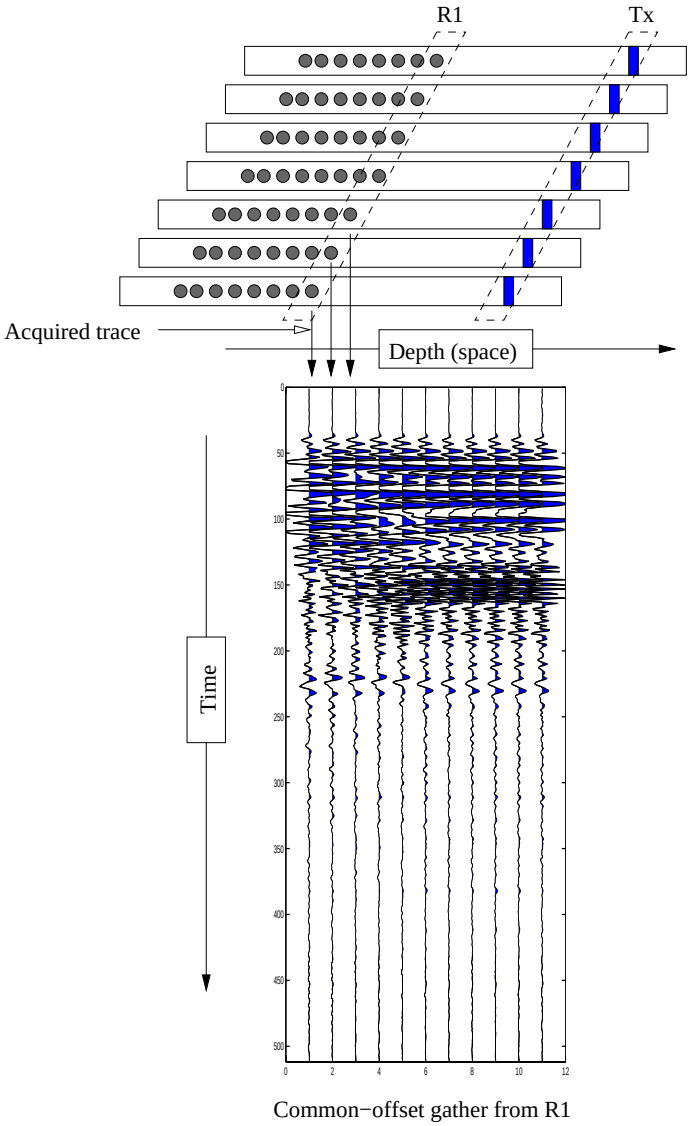

Figure 1.4: Common-offset gather: A collection of traces acquired by a fixed receiver over the depth axis.

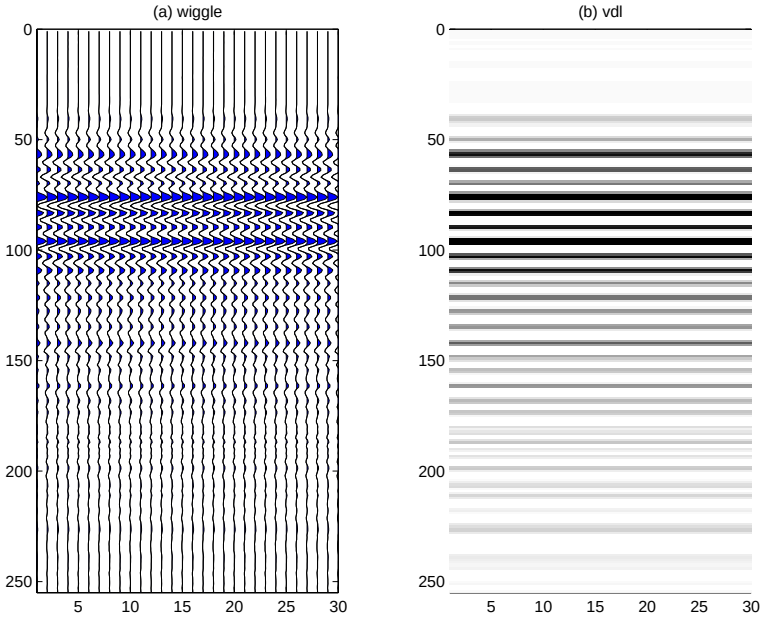

Figure 1.5: Waveform presentation: (a) Wiggle and (b) VDL.

1.3 Notation and Abbreviations

1.3.1 List of Symbols

Symbol	:	Description
$\|\cdot\|$	:	norm in a general vector space
$\mathbf{A}^T$	:	transposition of matrix $\mathbf{A}$
$C(h)$	:	covariance at distance h
D	:	number of depth samples
E	:	statistical expectation operator
d	:	discrete space index
$\gamma(h)$	:	semi-variogram at distance h
$\mathbf{n}_d$	:	noise vector at depth d
$\mathbf{R}$	:	sample covariance matrix
$\mathbf{R}_d$	:	sample covariance matrix ($\mathbf{y}_d\mathbf{y}_d^T$) at depth d
T	:	number of time samples
t	:	discrete time index
$\mathbf{W}_d$	:	matrix whose columns are the p eigenvectors of $\mathbf{R}_d$ (if its dimension is *T-by-p*)
w_α	:	linearly associated weights
$\mathbf{x}_d$	:	waveform component vector at depth d
$\mathbf{Y}$	:	data table or common-offset gather
$\mathbf{y}_d$	:	observation vector at depth d
λ_i	:	the i^{th} ordered eigenvalue
ϕ_i	:	the i^{th} ordered eigenvector
$(\mathbf{u}\cdot\mathbf{v})$	:	a scalar product of the vectors $\mathbf{u}$ and $\mathbf{v}$
$\bar{\mathbf{x}}$	:	an average (of $\mathbf{x}$ in this case)
$\tilde{\mathbf{x}}$	:	an estimate (of $\mathbf{x}$ in this case)
diag($\mathbf{V}$)	:	when $\mathbf{V}$ is a vector with N components, diag($\mathbf{V}$) is a square matrix of order N with the element $\mathbf{V}$ on the main diagonal. if a matrix, diagonal elements of matrix $\mathbf{V}$.
$Z(\mathbf{x})$	:	intrinsic random function at position $\mathbf{x}$

1.3.2 List of Abbreviations

APEX	:	adaptive principal components extraction
CV	:	cross validation
EDF	:	effective degree of freedom
LSQ	:	least-squares
MATLAB	:	The Mathworks product for numerical computation and visualization
MSE	:	mean square error
PCA	:	principal component analysis
UTM	:	universal transverse mercator
VDL	:	variable-density log

Chapter 2

Principal Component Analysis

This chapter investigates PCA as an inference tool for a 2-d data model. For various PCA techniques, Matlab codes have been implemented. These codes are studied on both synthetic and real sonic logging data. This chapter is organized as follows. In the first section, we introduce normal PCA process (so called PCA bulk). Next, we attempt to apply PCA on-line process that has been studied in *Neural Network* domains. Our investigation reaches to local PCA, in which a weight function is introduced to narrow down an examination area, in the following section. The background reason for this localization is coming from the fact that: Hsu [8] applied normal PCA process to separate wave components and to extract features in acoustic well-logging data; however (1) this approach had to endure massive computation depending on a processing region and (2) inference results were averaged over the processing region.

2.1 Formulation

Let $\mathbf{x_d}$ be a *T-by-1* random vector representing a wave component of acoustic well-logging data at depth d, where T is the number of time samples in the wave train. Here we assume that wave components such as compressional, shear, and Stoneley are embedded in $\mathbf{x}_d$ as a realization of a random vector. Then observed data $\mathbf{y}_d$ can be formed as

$$\mathbf{y}_d = \mathbf{x}_d + \mathbf{n}_d, \ (d = 1, \cdots, D) \tag{2.1}$$

where $\mathbf{n}_d$ represents measurement noises whose dimension is same as one in $\mathbf{y}_d$. Our data table or common-offset gather is defined as

$$\mathbf{Y} = \left[\mathbf{y}_1 \, \mathbf{y}_2 \cdots \mathbf{y}_D\right], \ \{\mathbf{Y} \,|\, 1 \leq t \leq T, 1 \leq d \leq D\} \tag{2.2}$$

where T and D are the number of time and space samples, respectively.

General objectives in inference are to estimate characteristics of $\mathbf{x}$ from observed data $\mathbf{y}$ with use of a covariance matrix generated from the observation data. The sample covariance matrix $\mathbf{R}$ is

$$\mathbf{R} = \frac{1}{D} \sum_{d=1}^{D} (\mathbf{y}_d - \bar{\mathbf{y}})(\mathbf{y}_d - \bar{\mathbf{y}})^T \tag{2.3}$$

where

$$\bar{\mathbf{y}} = \frac{1}{D} \sum_{d=1}^{D} \mathbf{y}_d. \tag{2.4}$$

In the following section, we focus on PCA as an inference tool for the data model defined in (2.2).

2.2 PCA Process

2.2.1 Algorithm

Using the sample covariance matrix $\mathbf{R}$ whose size is *T-by-T* and eigenvector analyzing algorithm, $\mathbf{x}$ can be reconstructed from eigenvectors. To perform the reconstruction first solve

$$\mathbf{R}\phi_i = \lambda_i \phi_i, \ (i = 1, \cdots, T) \tag{2.5}$$

where ϕ_i is an eigenvector and λ_i is an eigenvalue. If $\|\phi_i\| = 1$, λ_i is calculated as

$$\lambda_i = {\phi_i}^T \mathbf{R} \phi_i, \ (i = 1, \cdots, T). \tag{2.6}$$

Then reconstruction $\tilde{\mathbf{x}}$ is defined as

$$\tilde{\mathbf{x}}_d[p] = \sum_{i=1}^{p} ((\mathbf{y}_d - \bar{\mathbf{y}})^T \phi_i)\phi_i + \bar{\mathbf{y}}, \ (p \leq T) \tag{2.7}$$

where ϕ_i is ordered so that associated eigenvalues are aligned to have largest first, smallest last.

2.2.2 Matlab Codes

In this section implemented Matlab codes are listed. They are *pcaBulk*, *pcaSort*, and *pcaBulkRestore*; and are summarized in Table 2.1.

Name	Descriptions
pcaBulk	computes PCA for bulk data.
pcaSort	sorts eigenvectors and associated eigenvalues in a decreasing order.
pcaBulkRestore	reconstructs original images.

Table 2.1: Matlab functions to perform PCA and reconstruction.

■ *pcaBulk*

```
function [v, c] = pcaBulk( x )
%
% function [v, c] = pcaBulk( x )
% computes PCA for bulk data
%
% Input:
%        x - sample waveform matrix whose
%            dimension is (t,d) where
%            t is a number of time steps and
%            d is a number of space (depth) steps.
%
% Output:
%        v - a matrix whose columns, [v1 v2 ... vt], are
%            eigenvectors.
```

```
%        c - a diagonal matrix of eigenvalues.
%
% Remark:
%
%        Use pcaSort() to sort out v and c in a manner that
%        the eigenvalues in c are ordered largest first,
%        smallest last.

[t, d] = size(x);
u = (sum(x,2)/d) * ones(1,d); % mean vector;
s = (x-u) * (x-u)' / d;

[v,c] = eig(s);
```

■ *pcaSort*

```
function [vs, cs] = pcaSort (v, c)
%
% function [vs, cs] = pcaSort( v, c )
% sorts v and c in a decreasing order of diag(c).
%
% Input:
%        v - a matrix whose columns, [v1 v2 ... vn], are
%            eigenvectors.
%        c - a diagonal matrix of eigenvalues.
%
% Output:
%        vs - a sorted matrix whose columns, [v1 v2 ... vn], are
%             reordered so that associated eigenvalues in c are
%             ordered largest first, smallest last.
%        dc - a sorted matrix whose diagonal components are
%             aligned in a decreasing order.
%

[m t] = size(v);
lamda = diag(c);
[y I] = sort(lamda);
cs    = diag(flipud(y));  % sorted diagonal matrix
                          % in a decreasing order
I     = flipud(I);
for j = 1:t,
   vs(:,j) = v(:,I(j,1));
end
```

■ *pcaBulkRestore*

```
function yr = pcaBulkRestore (vs, y, order)
%
```

```
% function yr =  pcaBulkRestore (vs, y, order)
% reconstructs the original image from given eigenvectors
% 'vs' and a given 'order' number.
%
% Input:
%        vs - a matrix whose columns, [v1 v2 ... vt], are
%             eigenvectors. These eigenvectors must be aligned
%             such that corresponding eigenvalues are sorted
%             in a decreasing order.
%        y -  an observation matrix whose dimension must be
%             (t,d) where t is a number of time steps and
%             d is a number of depth steps.
%        order - a positive integer number to indicate that
%             to which order's reconstruction is made.
%             order = 1 ... uses the first eigenvector,
%             order = i ... uses the i-th eigenvector.
%             (order <= t)
%
% Output:
%        yr - a reconstructed matrix whose dimension must be
%             (t,d);
%

[t, d] = size(y);
u = (sum(y,2)/d) * ones(1,d); % mean vector
yr = zeros(t,d);
for s=1:d,
   yr(:,s) = ( (y(:,s) - u(:,s))' * ...
              vs(:,order) .* vs(:,order)) + u(:,s);
end
```

2.2.3 Synthetic Data Example

To ensure the work of generated Matlab codes a 2-d section whose dimension is *3-by-250* is synthetically generated with the following conditions

$$\mathbf{y}_d \sim \mathbf{N}\left(\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T, diag\left(\begin{bmatrix}5 & 3 & 1\end{bmatrix}\right)\right), (d = 1, \cdots, 250) \tag{2.8}$$

where $\mathbf{N}(a, b)$ represents the normal distribution with the mean and variance values are a and b, respectively. Figure 2.1(a) presents the original 2-d section that is synthetically generated from (2.8). Reconstructed images from an individual eigenvector, the 1^{st}, 2^{nd}, and 3^{rd}, are shown in Figures 2.1(b), (c), and (d), respectively. In

Figure 2.1(e) all reconstructed images are overlaid to restore the original shape with three eigenvectors and their associated eigenvalues. Eigenvectors and eigenvalues are computed as shown in Table 2.2.

Symbol	Eigenvector			Eigenvalue
ϕ_1	$[-0.9904$	0.1313	$-0.0441]^T$	5.1610
ϕ_2	$[-0.1326$	-0.9908	$0.0282]^T$	3.0548
ϕ_3	$[-0.0399$	0.0337	$0.9986]^T$	1.0816

Table 2.2: Eigenvectors and associated eigenvalues computed from 2-d synthetic data shown in Figure 2.1(a).

2.2.4 Openhole Sonic Data Example

In this section a common-offset gather obtained from single-well acoustic logging data is dealt. As an input to the PCA bulk analysis a 2-d section whose dimension is *200-by-263* is prepared. This input data set is taken from Yamamoto *et al.* [18]. Specifically, a part of the 2-d section is extracted as (30:80,1:250) (i.e.,*51-by-250*) and it is used for reconstruction of compressional components.

Reconstruction of Compressional Components

Figure 2.2 presents both input data, which mainly contain compressional waves (*51-by-250*), and reconstructed images. In Figure 2.3 detailed looking on the 100^{th} trace is made. Primary eigenvectors are selectively presented in Figure 2.4. The orthogonality among primary eigenvectors is examined by taking the scalar product of $(\phi_1 \cdot \phi_i)$ where i varies from 2 to 5. The order of any product value is less than 10^{-16} in the absolute sense.

Reconstruction of Reflection Events

Similarly the original section whose dimension is *200-by-263* is examined. Figures 2.5, 2.6, and 2.7 present processed results. In Figure 2.8 a detailed looking is made for the

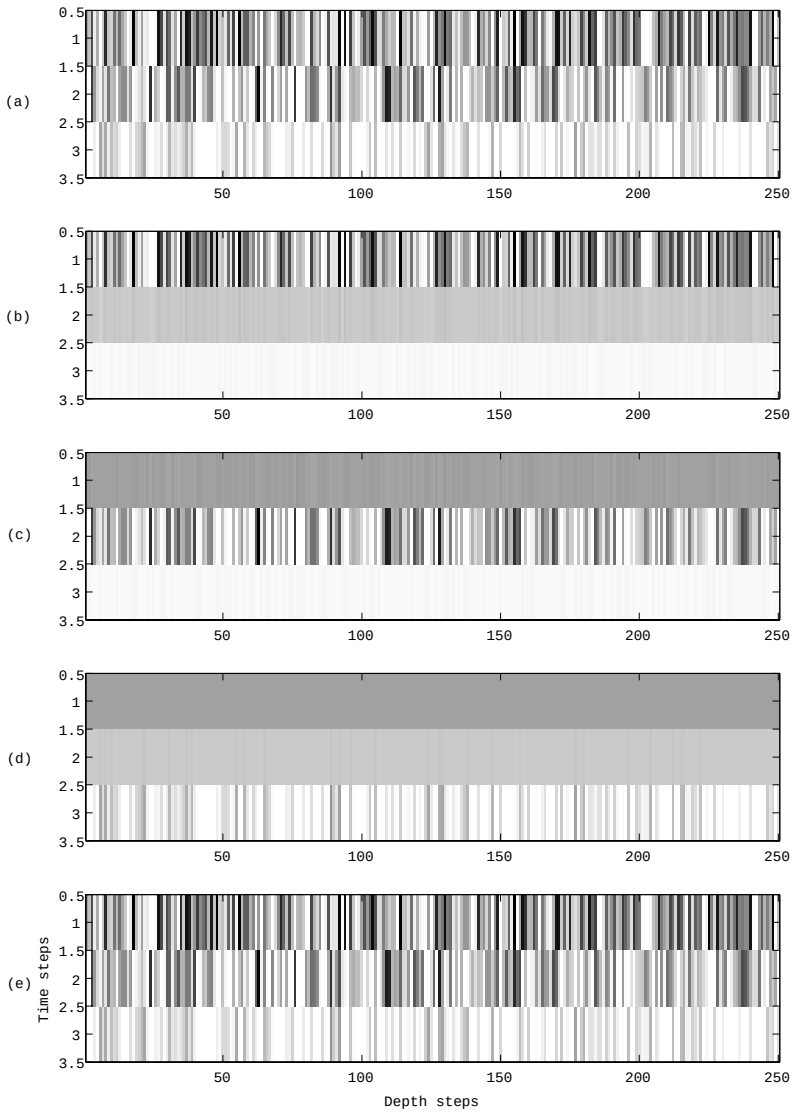

Figure 2.1: Reconstruction of synthetic data by PCA. (a) The original 2-d section is synthetically generated from (2.8). Reconstructed images from an individual eigenvector, the 1^{st}, 2^{nd}, and 3^{rd}, are shown in (b), (c), and (d), respectively. In (e) all reconstructed images are overlaid to recover the original shape.

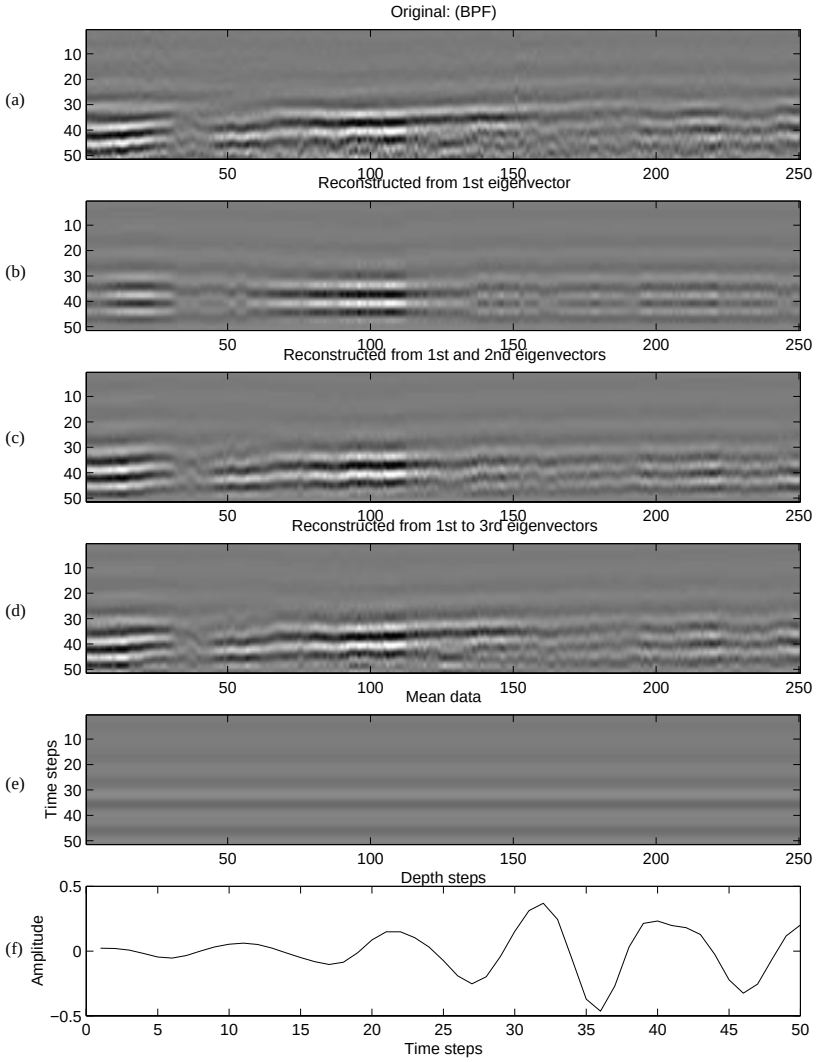

Figure 2.2: Reconstruction of common-offset gather data by PCA bulk. Input data have only 51 time samples per trace. The original image (band-passed) is shown in (a). This image is reconstructed from the 1^{st} eigenvector (b), both the 1^{st} and 2^{nd} eigenvectors (c), and the initial three eigenvectors (d). Sample mean data are imaged and plotted in (e) and (f), respectively.

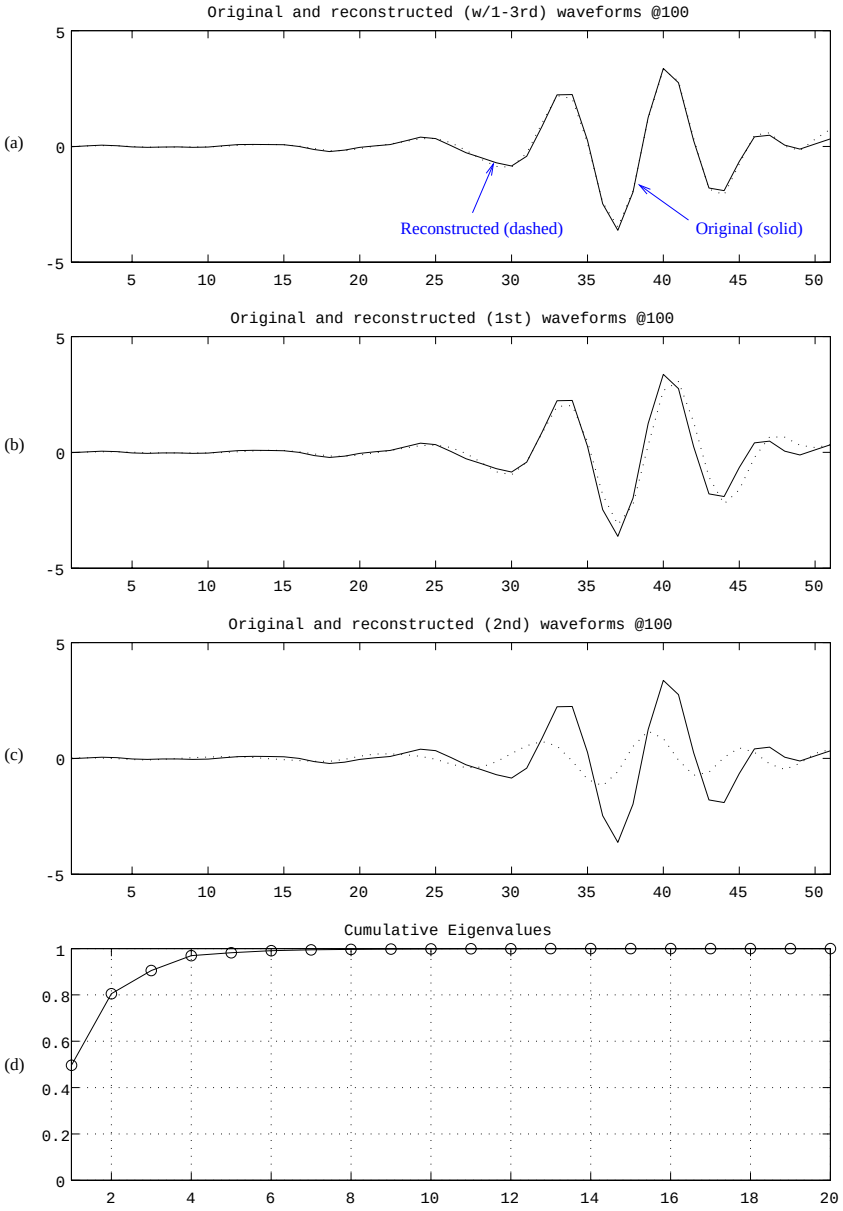

Figure 2.3: (a) A reconstructed compressional wave from the three primary eigenvectors. (b) Individual reconstruction only from the 1^{st} eigenvector. (c) Same as (b) but for the 2^{nd} eigenvector. In (d) eigenvalues are plotted in a cumulative manner. The original waveform is extracted from the trace number 100 at Figure 2.2(a).

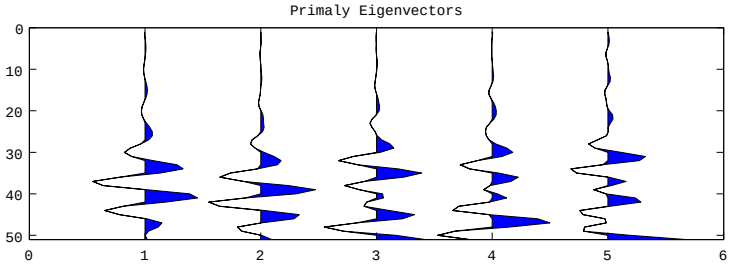

Figure 2.4: Primary five eigenvectors from ϕ_1 to ϕ_5 extracted from Figure 2.2(a).

same 100^{th} trace and comparison is made in between the original and reconstructed wavetrains. The orthogonality among primary eigenvectors is examined by taking the scalar product of $(\phi_1 \cdot \phi_i)$ where i varies from 2 to 5. The order of any product value is less than 10^{-16} in the absolute sense.

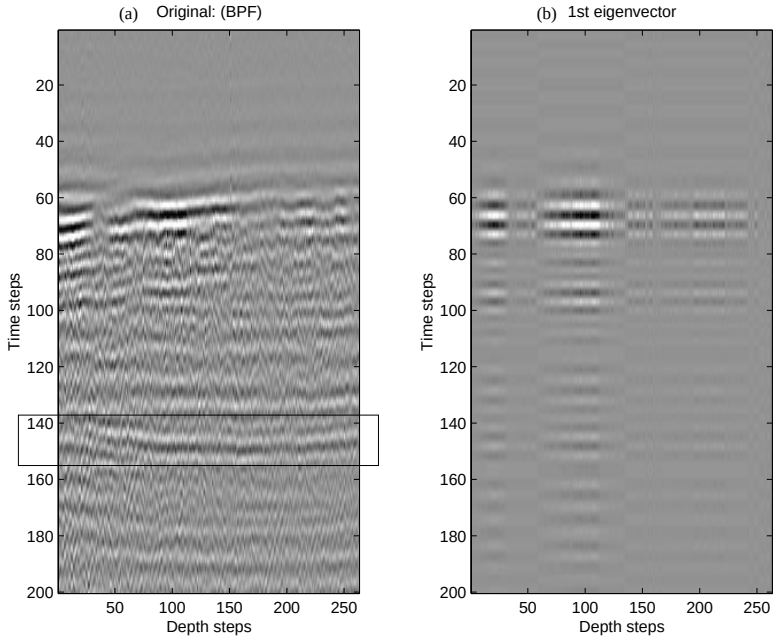

Figure 2.5: (a) Original common-offset gather consisting of *200-by-263* samples. On this data PCA bulk is performed. A rectangular encloses reflection signals coming from a reflector near the borehole. (b) Reconstructed data from the 1^{st} eigenvector.

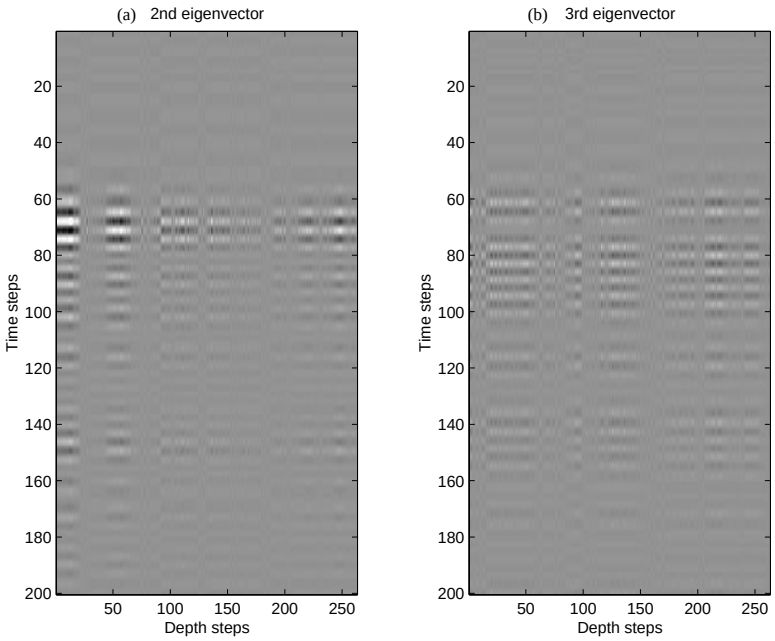

Figure 2.6: (a) Reconstructed data from the 2^{nd} eigenvector for the common-offset gather shown in Figures 2.5(a). (b) Same as (a) but for the 3^{rd} eigenvector.

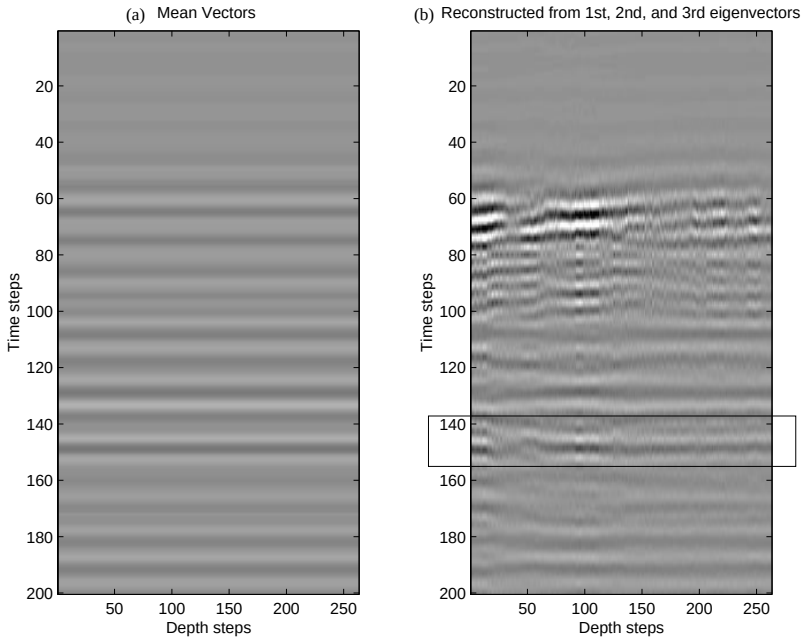

Figure 2.7: (a) Mean vector from the common-offset gather shown in Figures 2.5(a). (b) Reconstructed data with use of three primary eigenvectors and the mean vector. A rectangular encloses reconstructed reflection signals.

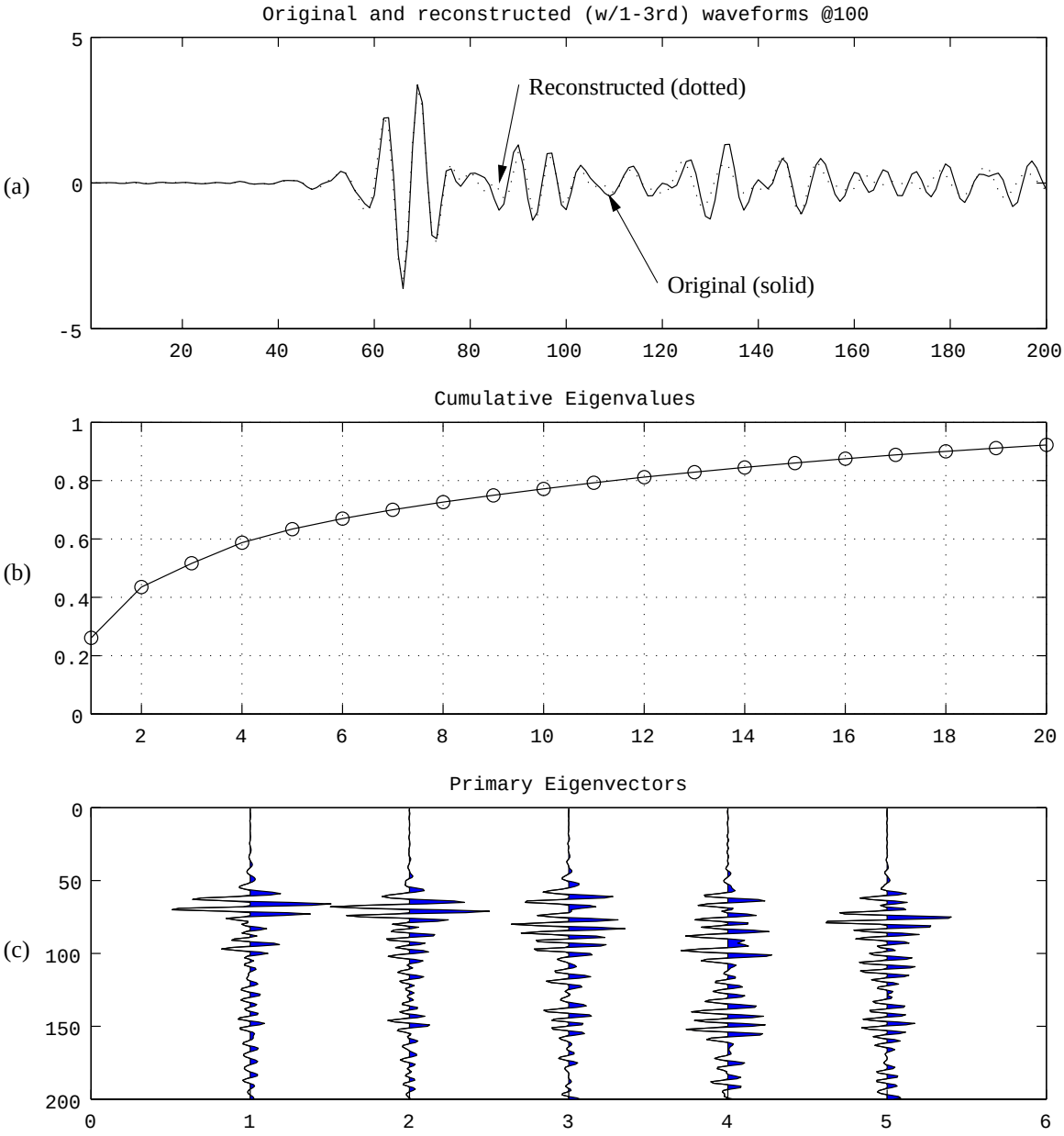

Figure 2.8: (a) Reconstructed multiple components (mainly including compressional and reflected waves) from the three primary eigenvectors for the 100^{th} trace in Figures 2.5(a). In (b) eigenvalues are plotted in a cumulative manner. (c) Five primary eigenvectors.

2.3 PCA On-line Process

PCA on-line process is to perform PCA by self-organizing rules. This is an unsupervised mechanism. We study three algorithms – two of them use feedforward connection and the rest utilizes both feedforward and feedback connections. These algorithms are implemented and applied on both synthetic data and single-well acoustic logging data.

2.3.1 Algorithm with Feedforward

For each sample covariance matrix $\mathbf{R}_d$ we compute a matrix $\mathbf{W}_d$ whose dimension is T-*by*-p ($p \leq T$) and columns are the p ($p \leq T$) eigenvectors of $\mathbf{R}_d$. Let the i^{th} column of $\mathbf{W}_d$ be $\mathbf{W}_d^i$, then the PCA on-line process ALG-1 which is due to Oja *et al.* found in Chatterjee *et al.* [3] is defined as

$$\mathbf{W}_{d+1}{}^i = \mathbf{W}_d{}^i + \eta_d \Bigg(\mathbf{R}_d \mathbf{W}_d{}^i - \mathbf{W}_d{}^i \mathbf{W}_d{}^{iT} \mathbf{R}_d \mathbf{W}_d{}^i - \gamma \sum_{j=1}^{i-1} \mathbf{W}_d{}^j \mathbf{W}_d{}^{jT} \mathbf{R}_d \mathbf{W}_d{}^i \Bigg), \; (i = 1, \cdots, p) \quad (2.9)$$

where $\{\eta_d\}$ is a sequence of scalar gains and γ is a parameter which is greater than or equal to one. When $p = 1$, the first eigenvector is recursively obtained by

$$\mathbf{W}_{d+1} = \mathbf{W}_d + \eta_d \mathbf{W}_d{}^T \mathbf{y}_d \left(\mathbf{y}_d - \mathbf{W}_d{}^T \mathbf{y}_d \mathbf{W}_d \right) \quad (2.10)$$

where $\mathbf{W}_d{}^T \mathbf{y}_d$ is a scalar.

Similarly ALG-2 which is based on *Xu and et al.* (see Chatterjee *et al.* [3]) is formed as

$$\mathbf{W}_{d+1}{}^i = \mathbf{W}_d{}^i + \eta_d \Bigg(2\mathbf{R}_d \mathbf{W}_d{}^i - \mathbf{W}_d{}^i \mathbf{W}_d{}^{iT} \mathbf{R}_d \mathbf{W}_d{}^i - \gamma \sum_{j=1}^{i-1} \mathbf{W}_d{}^j \mathbf{W}_d{}^{jT} \mathbf{R}_d \mathbf{W}_d{}^i - \mathbf{R}_d \mathbf{W}_d{}^i \mathbf{W}_d{}^{iT} \mathbf{W}_d{}^i - \gamma \sum_{j=1}^{i-1} \mathbf{R}_d \mathbf{W}_d{}^j \mathbf{W}_d{}^{jT} \mathbf{W}_d{}^i \Bigg), \; (i = 1, \cdots, p). \quad (2.11)$$

In case of $p = 1$, the first eigenvector is recursively obtained by

$$\mathbf{W}_{d+1} = \mathbf{W}_d + \eta_d \mathbf{W}_d{}^T \mathbf{y}_d \left(2\mathbf{y}_d - \mathbf{W}_d{}^T \mathbf{y}_d \mathbf{W}_d - \mathbf{y}_d \mathbf{W}_d{}^T \mathbf{W}_d\right) \tag{2.12}$$

where $\mathbf{W}_d{}^T \mathbf{y}_d$ and $\mathbf{W}_d{}^T \mathbf{W}_d$ are scalar values.

2.3.2 Algorithm with Feedforward and Feedback

Another attempts are made to make use of the adaptive principal components extraction (APEX), which uses both feedforward and feedback connections (Kung and Diamantaras [9]; Section 8.7 in Haykin [6]). In the APEX algorithm the first eigenvectors are computed with (2.10) and then, by re-using these vectors, the second eigenvectors are readily computed. Here consideration is only made up to the second eigenvectors.

The APEX algorithm can be summarized as follows:

1. Initialize the feedforward weight vector $\mathbf{W}_d$ and the feedback weight vector $\mathbf{A}_d$ to small random values at depth $d = 1$.

2. For $i = 1$ compute (2.10) to obtain the first eigenvector.

3. For $i = 2$, and $d = 1, 2, \cdots$, compute:

$$\mathbf{Z}_d{}^{i-1} = [\mathbf{Z}_d{}^1 \ \mathbf{Z}_d{}^2 \cdots \mathbf{Z}_d{}^{i-1}]^T \tag{2.13}$$

$$\mathbf{Z}_d{}^i = \mathbf{W}_d{}^{iT} \mathbf{Y}_d + \mathbf{A}_d{}^{iT} \mathbf{Z}_d{}^{i-1} \tag{2.14}$$

$$\mathbf{W}_{d+1}{}^i = \mathbf{W}_d{}^i + \eta_d [\mathbf{Z}_d{}^i \mathbf{Y}_d - \mathbf{Z}_d{}^{i^2} \mathbf{W}_d{}^i] \tag{2.15}$$

$$\mathbf{A}_{d+1}{}^i = \mathbf{A}_d{}^i - \eta_d [\mathbf{Z}_d{}^i \mathbf{Z}_d{}^{i-1} + \mathbf{Z}_d{}^{i^2} \mathbf{A}_d{}^i] \tag{2.16}$$

2.3.3 Matlab Codes

In this section implemented Matlab codes to perform ALG-1, ALG-2, and APEX are presented. To switch the algorithm, ALG-1 or ALG-2, an argument named *algorithm* is used in *pcaSelfOrganize*.

Name	Descriptions
pcaSelfOrganize	computes PCA with self-organizing rules (ALG-1 or ALG-2).
pcaAPEX	computes PCA with the APEX algorithm.

Table 2.3: Matlab functions to perform PCA on-line.

■ *pcaSelfOrganize*

```
function [wk1, ex1] = pcaSelfOrganize ( wk0, x, d, p, ex0, ...
alpha, beta, sgp, gamma, algorithm )
%
% function [wk1, ex1] =
%          pcaSelfOrganize( wk0, x, d, p, ex,
%                           alpha, beta, sgp,
%                           gamma, algorithm )
% computes PCA with the self-orgamizing rule.
%
% Input:
%        wk0 - a matrix whose columns,
%            [v1 v2 ... vn], are eigenvectors.
%        x - a sample vector (n-by-1) observed at depth d.
%        d - a natural number to indicate d-th step.
%            this number is used to produce a sequence of
%            scalar gains. See Remark for the sequence applied.
%        p - a number of eigenvectors (p =< n) used. The first
%            p columns are only used as the principal eigenvector
%            matrix of wk0.
%        ex0 - a mean vector of previous samples of x. ex0 should
%            have the same dimension of x.
%            In case of d=1, ex0 can be an arbitrary
%            (or zero) vector.
%        alpha - a parameter to form a scalar gain. See Remark.
%        beta - a parameter to form a scalar gain. See Remark.
%        sgp - a number of power value for the scalar gain.
%            (1/(alpha*d + beta))^(sgp) is set in the equation.
%        gamma - a scalar (g >= 1)
%        algorithm - 0.......... Oja et al. Refer to (1) in
%                                Chatterjee et al. [3].
%                    non zero... Xu et al.  Refer to (2) in
%                                Chatterjee et al. [3].
%                    Notice that if no input is given, then
%                    algorithm is regarded as 0 (i.e., Oja et al.).
%
% Output:
%        wk1 - a matrix whose columns, [v1 v2 ... vn], are
%            eigenvectors.
%        ex1 - an updated mean vector. This vector will be
```

```
%            an input for the next calling time as ex0.
%
% Remark:
%        As a sequence of scalar gain:
%        (1/(alpha*d + beta))^(sgp) is used.
%

if nargin < 10, algorithm = 0; end

[m,n] = size(wk0);
wk1 = zeros(size(wk0));          % an output
S0  = zeros(m,1);                % a work matrix

if (d == 1)
   ex1 = x;
else
   ex1 = ((d-1)*ex0 + x) ./ d;  % update the mean vector
   x   = x - ex1;
end

ak  = x * x';

if ( algorithm == 0 )            % Algorithm - Oja et al.

   for i=1:p,                    % over only principal columns
      S = S0;                    % clear S matrix


      for j=1:(i-1),
         S = S + wk0(:,j)*wk0(:,j)'*ak*wk0(:,i);
      end
      A = ak*wk0(:,i)-wk0(:,i)*wk0(:,i)'*ak*wk0(:,i)-gamma*S;
      wk1(:,i) =  wk0(:,i) + (1/(alpha*d + beta))^(sgp) * A;

   end

else                             % Algorithm - Xu et al.

   R0  = zeros(m,1);             % a work matrix

   for i=1:p,                    % over only principal columns
      S = S0;                    % clear S matrix
      R = R0;                    % clear R matrix

      for j=1:(i-1),
         S = S + wk0(:,j)*wk0(:,j)'*ak*wk0(:,i);
         R = R + ak*wk0(:,j)*wk0(:,j)'*wk0(:,i);
      end
      A = 2*ak*wk0(:,i)-wk0(:,i)*wk0(:,i)'*ak*wk0(:,i)-gamma*S ...
          -ak*wk0(:,i)*wk0(:,i)'*wk0(:,i)-gamma*R;
      wk1(:,i) =  wk0(:,i) + (1/(alpha*d + beta))^(sgp) * A;

   end

end
```

■ *pcaAPEX*

```
function [wk1, ak1, ex1] =
         pcaAPEX ( wk0, ak0, x, d, p, ex0, alpha, beta, sgp)
%
% function [wk1, ak1, ex1] =
%          pcaAPEX(  wk0, ak0, x, d, p, ex, alpha, beta, sgp )
% computes the addaptive principal components extraction (APEX).
%
% Input:
%         wk0 - a matrix whose columns,
%             [v1 v2 ... vn], are eigenvectors.
%         ak0 - a scalar parameter for feedback weight
%         x - a sample vector (n-by-1) observed at depth d.
%         d - a natural number to indicate d-th step.
%             this number is used to produce a sequence of
%             scalar gains. See Remark for the sequence applied.
%         p - a number of eigenvectors (p =< n) used. The first
%             p columns are only used as the principal eigenvector
%             matrix of wk0.
%         ex0 - a mean vector of previous samples of x. ex0 should
%             have the same dimension of x.
%             In case of d=1, ex0 can be an arbitrary
%             (or zero) vector.
%         alpha - a parameter to form a scalar gain. See Remark.
%         beta - a parameter to form a scalar gain. See Remark.
%         sgp - a number of power value for the scalar gain.
%             (1/(alpha*d + beta))^(sgp) is set in the equation.
%
% Output:
%         wk1 - a matrix whose columns, [v1 v2 ... vn], are
%             eigenvectors.
%         ak1 - an updated feed-back weight (scalar)
%         ex1 - an updated mean vector. This vector will be
%             an input for the next calling time as ex0.
%
% Remark:
%         As a sequence of scalar gain:
%         (1/(alpha*d + beta))^(sgp) is used.
%
% References:
%
%         Haykin [6]
%

[m,n] = size(wk0);
wk1 = zeros(size(wk0)); % an output

if (d == 1)
   ex1 = x;
else
```

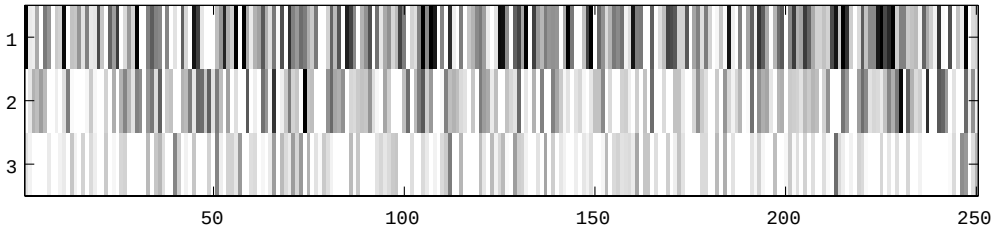

Figure 2.9: 2-d synthetic data applied for PCA on-line. The data are derived from (2.8) and consist of *3-by-250* points.

```
   ex1 = ((d-1)*ex0 + x) ./ d;  % update the mean vector
   x   = x - ex1;
end

ak  = x * x';

S = wk0(:,p)'*x + ak0*wk0(:,p-1)'*x;
A = S*x - S*S'*wk0(:,p);
B = (S*(wk0(:,p-1)'*x))' + S*S*ak0;

wk1(:,p) = wk0(:,p) + (1/(alpha*d + beta))^(sgp) * A;
ak1      = ak0      - (1/(alpha*d + beta))^(sgp) * B;
```

2.3.4 Synthetic Data Example I

To validate the algorithm implemented in Section 2.3.3 the synthetic 2-d section derived from (2.8) is reused. Figure 2.9 shows the generated images whose dimension is *3-by-250*. For this data three eigenvectors are recursively calculated over the entire depth samples (i.e., 250 steps).

As for initial eigenvectors an arbitrary orthogonal matrix whose components are

$$\mathbf{W}_1 = \begin{bmatrix} 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 0 & 2/\sqrt{6} \end{bmatrix} \tag{2.17}$$

is chosen and totally 249 iterations are performed.

Figure 2.10 shows the ALG-1's iteration of the three principal eigenvectors for

choices of $\gamma = 1.2$ and $\eta_d = 1/(13.5 + d)$. At the end of iterations the process yields the following eigenvectors:

$$\mathbf{W}_{250} = \begin{bmatrix} 1.0031 & -0.0949 & 0.0372 \\ 0.0664 & 0.9918 & -0.0013 \\ -0.0345 & -0.0442 & 0.9910 \end{bmatrix} \tag{2.18}$$

which consist of almost three orthogonal vectors. From Figure 2.10 it is observed that the estimates of three eigenvectors are converging to the true vectors as proceeding the iterations.

Figure 2.11 shows the ALG-2's iteration which has also the same choices of $\gamma = 1.2$ and $\eta_d = 1/(13.5 + d)$. Resultant eigenvectors from ALG-2 are

$$\mathbf{W}_{250} = \begin{bmatrix} 1.0002 & -0.1309 & 0.0608 \\ 0.0656 & 0.9856 & -0.0049 \\ -0.0345 & -0.0409 & 0.9937 \end{bmatrix}. \tag{2.19}$$

Notice that in Figures 2.10 and 2.11 the y-axis is taken to indicate direction cosine for the true vector and a computed eigenvector.

2.3.5 Synthetic Data Example II

In this section a more practical dimension whose size is *51-by-1000* is considered. A data set generated is made by the following equation:

$$y_d \sim \mathbf{N}\left(\mu^T, diag\left([5\ 3\ 1\ 0.5\ 0.1\ 0.01 \cdots 0.01]\right)\right),\ (d = 1, \cdots, 1000) \tag{2.20}$$

where μ is a mean vector whose dimension is *1-by-51* and element values are depicted in Figure 2.12. First three eigenvectors are recursively computed over 1000 depth samples. Results from ALG-1 and ALG-2 cases are presented in Figures 2.13 and 2.14, respectively. Note that initial eigenvectors used for both ALG-1 and 2 are computed from PCA described in Section 2.2.

In both algorithm cases it is observed that each of three primary eigenvectors is getting closer to the true vector because direction cosine value for the true vector and

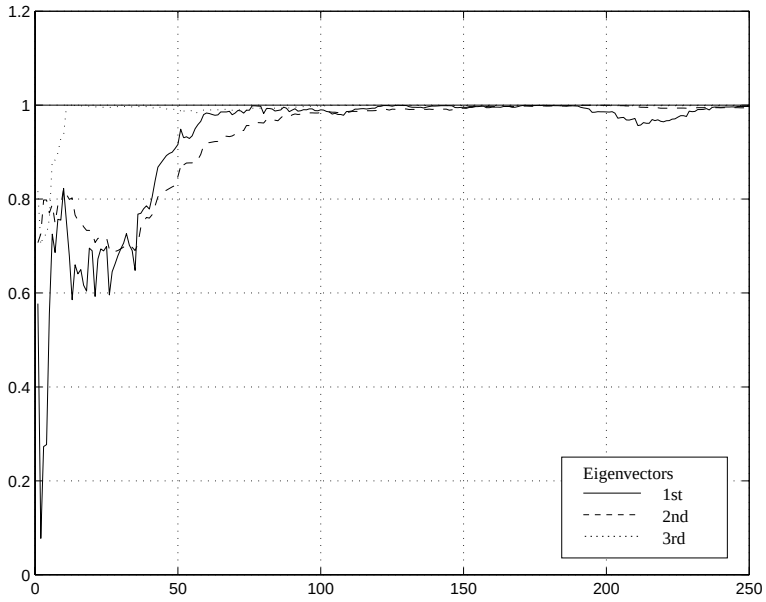

Figure 2.10: A PCA on-line process by ALG-1 over 250 depth samples for the synthetic data generated from (2.8). Processing parameters are taken as $(\gamma, \eta_d) = (1.2, 1/(13.5 + d))$.

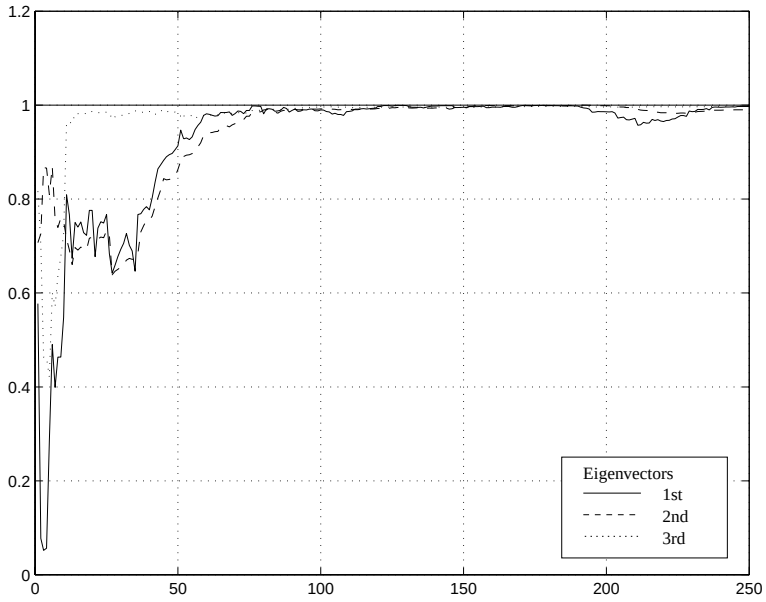

Figure 2.11: A PCA on-line process by ALG-2 over 250 depth samples for the synthetic data generated from (2.8). Processing parameters are taken as $(\gamma, \eta_d) = (1.2, 1/(13.5 + d))$.

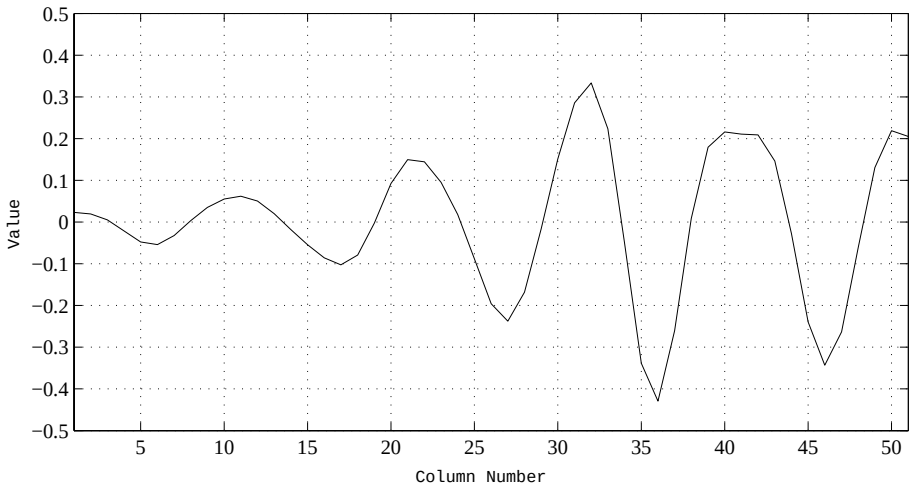

Figure 2.12: Element values for the mean vector. This vector consists of *1-by-51* elements.

the computed eigenvector is approaching to one.

2.3.6 Openhole Sonic Data Examples

A real data set to be applied is identical with the data used in Figure 2.2(a). In this case 250 traces are processed in a trace-by-trace manner. Basic parameters are (γ, η_d) where in the Matlab code γ is represented as a gamma parameter and $\{\eta_d\}$ is a sequence of scalar gain factors depending on the depth index.

In the PCA on-line process both the 1^{st} and 2^{nd} eigenvectors are recursively computed. For each eigenvector both ALG-1 and 2 are attempted. Initial eigenvectors used for both ALG-1 and 2 are computed from PCA described in Section 2.2. Figures 2.15 and 2.16 are processing results for the 1^{st} and 2^{nd} eigenvector, respectively. Each of these cases adopts $(\gamma, \eta_d) = (1.2, 1/(10 + 6d))$.

APEX processing results for the 2^{nd} eigenvector with $(\eta_d) = (1/(100 + 2d)$

Figure 2.17 presents computation results from the APEX algorithm with the 2^{nd} eigenvector $(i = 2)$. In this try the computation is repeated three times over the

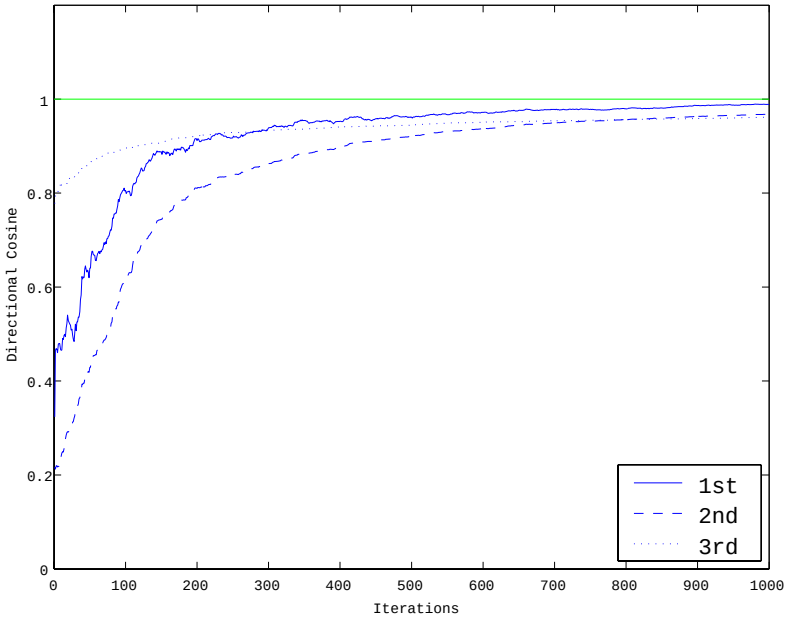

Figure 2.13: A PCA on-line process by ALG-1 over 1000 depth samples for the synthetic data generated from (2.20). Processing parameters are taken as $(\gamma, \eta_d) = (1.2, 1/(10 + 0.2d))$.

entire depth region of the common-offset gather data shown in Figure 2.2(a). It is observed that directional cosine values get higher at the early stage (at around the 60^{th} iteration) when comparing with ALG-1 and 2 results as shown in Figure 2.16. However fluctuation in directional cosine values appears.

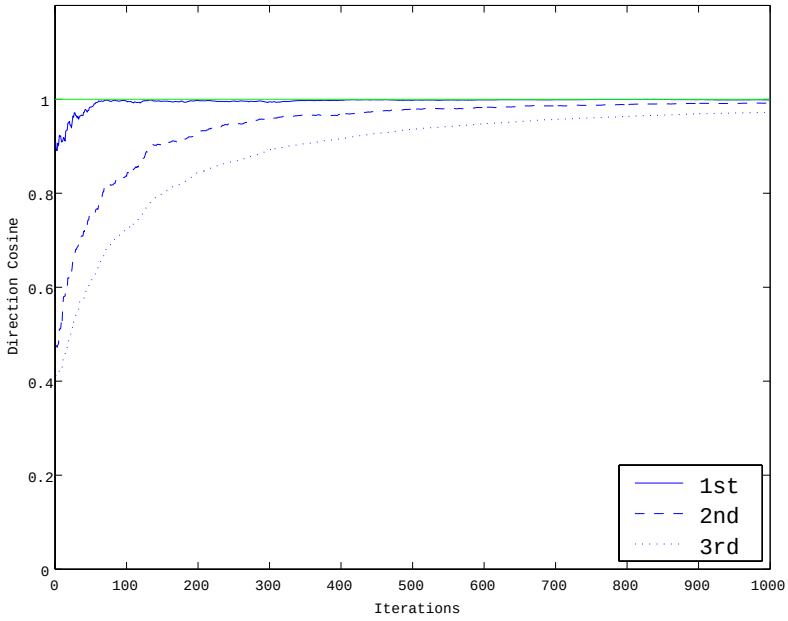

Figure 2.14: A PCA on-line process by ALG-2 over 1000 depth samples for the synthetic data generated from (2.20). Processing parameters are taken as $(\gamma, \eta_d) = (1.2, 1/(10 + 0.2d))$.

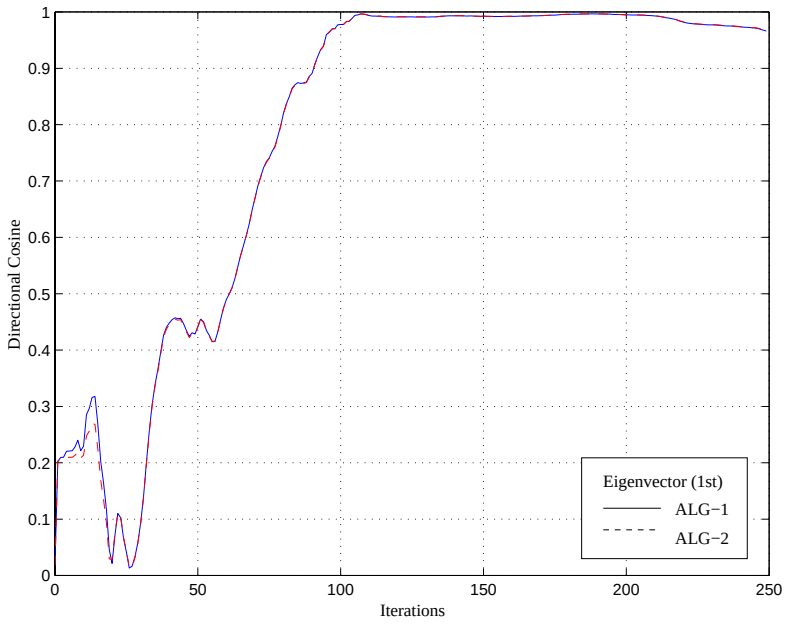

Figure 2.15: PCA on-line processing results from ALG-1 and ALG-2 for the 1^{st} eigenvector with $(\gamma, \eta_d) = (1.2, 1/(10 + 6d))$. The input data is taken from Figure 2.2(a). The 1^{st} principal component is computed over successive 250 trace data.

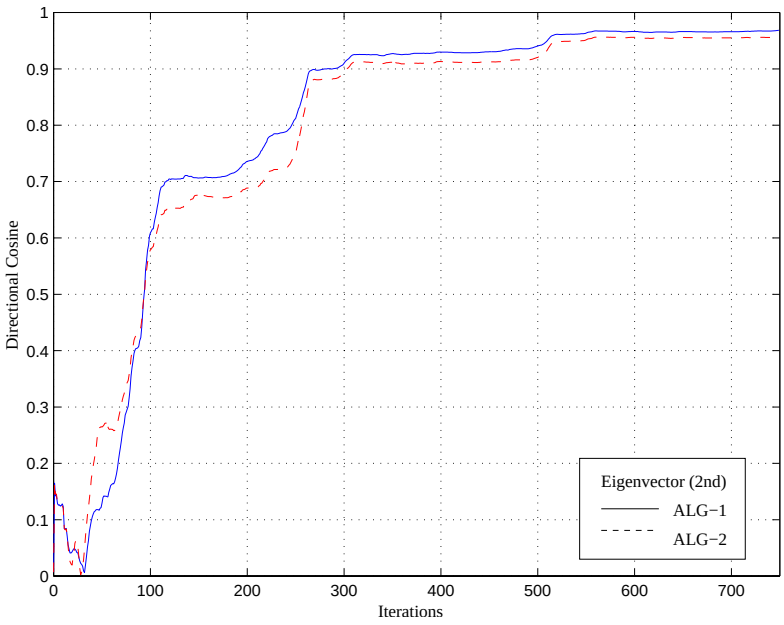

Figure 2.16: PCA on-line processing results from ALG-1 and ALG-2 for the 2^{nd} eigenvector with $(\gamma, \eta_d) = (1.2, 1/(10 + 6d))$. The input data is taken from Figure 2.2(a). The 2^{nd} principal component is computed over *three* iterations of successive 250 trace data. The computed eigenvector in a previous iteration is fed into a new iteration as the initial value.

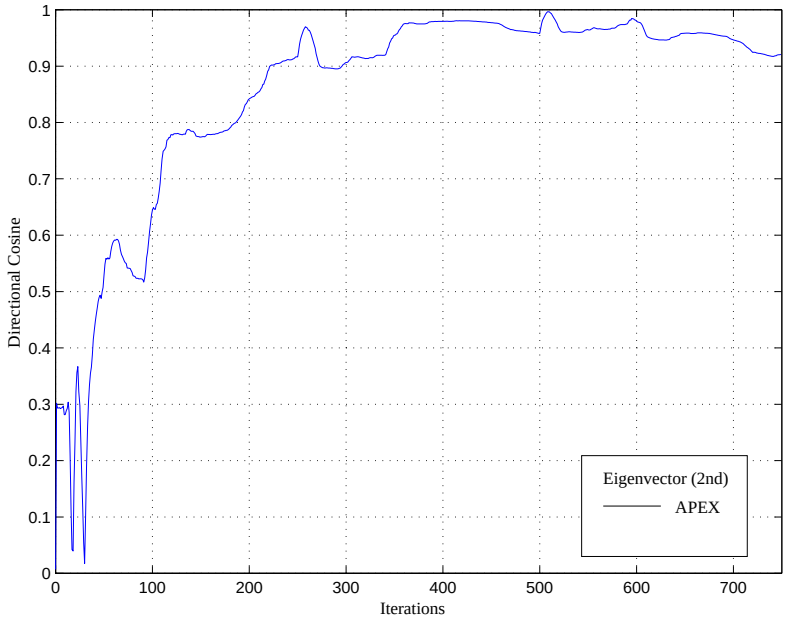

Figure 2.17: PCA on-line processing results from APEX for the 2^{nd} eigenvector ($\mathbf{W}_d{}^2$) with $\eta_d = 1/(100+2d)$. The input data is taken from Figure 2.2(a). The 2^{nd} principal component is computed over *three* times of successive 250 trace data (i.e., repeating on-line process *three* times).

2.4 Localized Analysis of PCA

PCA is a standard method for extracting informative data of lower dimension from an original data set in the paradigm of unsupervised learning. In acoustic well-logging data, various wave components are presented together with noises over an acquired depth region. The noises are spatially distributed beyond the Gaussian distribution with constant variance. Therefore it is essential to examine the data locally instead of handling globally so that local features can be extracted effectively. The key is to assess a localized sample variance matrix using a kernel function. This section briefly reviews a localized version of PCA.

2.4.1 Algorithm

The localized version of (2.3) can be formed as

$$\mathbf{R}^*(d) = \frac{1}{D}\sum_{d'=1}^{D} K_h(d-d')(\mathbf{y}_{d'} - \mathbf{y}^*(d))(\mathbf{y}_{d'} - \mathbf{y}^*(d))^T \tag{2.21}$$

where K_h is a localized weight function

$$K_h(x-d) = \exp\{-\frac{1}{2h^2}(x-d)^2\} \tag{2.22}$$

and

$$\mathbf{y}^*(d) = \frac{\sum_{d'=1}^{D} \mathbf{y}_{d'} K_h(d-d')}{\sum_{d'=1}^{D} K_h(d-d')}. \tag{2.23}$$

We may adopt the use of (2.5) to our present situation by writing

$$\mathbf{R}^*(d)\phi_i{}^*(d) = \lambda_i{}^*(d)\phi_i{}^*(d),\ (i = 1, \cdots, T) \tag{2.24}$$

where $\phi_i{}^*(d)$ and $\lambda_i{}^*(d)$ are an eigenvector and eigenvalue at depth d, respectively. If $\|\phi_i{}^*(d)\| = 1$, $\lambda_i{}^*(d)$ is calculated as

$$\lambda_i{}^*(d) = \phi_i{}^{*T}\mathbf{R}^*(d)\phi_i{}^*(d),\ (i = 1, \cdots, T). \tag{2.25}$$

Then reconstruction $\tilde{\mathbf{x}}_d$ is defined as

$$\tilde{\mathbf{x}}_d[p] = \sum_{i=1}^{p} \tilde{\mathbf{x}}_i + \mathbf{y}^*(d),\ (p \leq T) \tag{2.26}$$

where $\tilde{\mathbf{x}}_i$ is

$$\tilde{\mathbf{x}}_i = (\mathbf{y}_d - \mathbf{y}^*(d))^T \phi_i{}^*(d)\phi_i{}^*(d) \tag{2.27}$$

and $\phi_i{}^*(d)$ is ordered so that associated eigenvalues are aligned to have largest first, smallest last.

2.4.2 Matlab Codes

In this section implemented Matlab codes are listed. They are *pcaLocal*, *kh*, and *pcaLocalRestore*; and are summarized in Table 2.4.

Name	Descriptions
pcaLocal	computes local PCA.
kh	computes weight values to be used in *pcaLocal*.
pcaLocalRestore	reconstructs original images.

Table 2.4: Matlab functions to perform local PCA and reconstruction.

■ *pcaLocal*

```
function [vd, cd, yud] = pcaLocal( x, d, ld, h )
%
% function [vd, cd, yud] = pcaLocal( x, d, ld, h )
% computes local PCA at depth of 'd'
%
% Input:
%        x - sample waveform matrix whose
%            dimension is (nt,nd) where
%            nt is a number of time steps and
%            nd is a number of space (depth) steps.
%        d - a depth index at which local PCA is
%            performed.
%        ld - a processing window width.
%            0 < ld <= 2*nd
%        h - a scalar to specify the shape of
```

```
%            the localized weight.
%
% Output:
%        vd - a matrix whose columns, [v1 v2 ... vnt], are
%             eigenvectors at depth 'd'.
%        cd - a diagonal matrix of eigenvalues at depth 'd'.
%        yud - a localized mean vector
%
%
% Remark:
%
%        pcaSort() is performed in this function
%

[nt, nd] = size(x);
hd = round((ld-1)/2);

  prs = max([d-hd  1]);  % processing start position
  pre = min([d+hd nd]);  % processing end   position
  pld = pre - prs + 1;

  xo  = x(:,prs:pre);
  khv = kh(prs-d:pre-d, h); % Weighting matrix
  sul = sum(khv);
  khw = diag(khv);

  yud = (sum(xo*khw,2)/sul);
  u_l = yud*ones(1,pld);

  s_l = zeros(nt, nt);
  for k = 1:pld
     s_l = s_l + khw(k,k)*(xo(:,k) - ...
           u_l(:,k))*(xo(:,k) - u_l(:,k))';
  end
     s_l = s_l / pld;

  [vo,co] = eig(s_l);

  [vd,cd] = pcaSort( vo, co );
```

■ *kh*

```
function w = kh(delta, h)
%
% function w = kh(delta, h)
%           computes weight values to be used in pcaLocal.
% Input
%    delta: a scalar or a vector containing
%           delta length from the center
%           reference point.
%           if zero, then w must be 1.
%    h:     A factor to define kurtosis of the weighting function.
```

```
%
% Output
%    w: a scalar or a vector of weighting
w = exp( -delta.^2 / (2 * h^2 ) );
```

Figure 2.18 presents computational results for a few h cases (h=15, 30, and 60). As a processing window (*delta* in the code) [-128:128] is set to compute 257 weight values for each h.

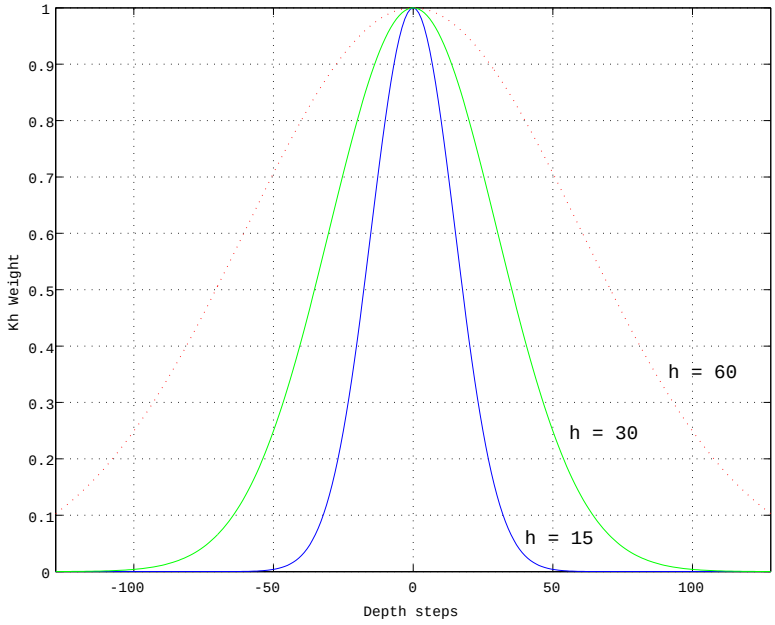

Figure 2.18: *Kh* curves for h=15, 30, and 60. The processing window is set to have [-128:128].

pcaLocalRestore

```
function ykr = pcaLocalRestore (vs, y, k, yuk,order)
%
% function ykr =  pcaLocalRestore (vs, y, k, yuk, order)
% reconstructs the original image from given eigenvectors
% 'vs' and a given 'order' number at depth k.
%
```

```
% Input:
%        vs - a matrix whose columns, [v1 v2 ... vt], are
%             eigenvectors. These eigenvectors must be aligned
%             such that corresponding eigenvalues are sorted
%             in a decreasing order.
%        y -  an observation matrix whose dimension must be
%             (t,d) where t is a number of time steps and
%             d is a number of depth steps.
%        k - depth index
%        yuk - a localized mean vector
%        order - a positive integer number or a vector to
%             indicate that to which order's reconstruction
%             is made.
%             order = 1 ... uses the first eigenvector,
%             order = i ... uses the i-th eigenvector.
%             order = [1 2 4] ... uses 1st, 2nd, and 4th
%                             eigenvectors with the mean vector.
%             (order <= t)
%
% Output:
%        ykr - a reconstructed vector at depth k
%              whose dimension must be (t,1);

[t, d] = size(y);
no     =  length(order);
ykr    = zeros([t,1]);

if no == 1
   if ((0 < order) & (order <= t))
      ykr = ( (y(:,k) - yuk)' * vs(:,order) .* vs(:,order));
   end
else
   for i=1:no
      if ((0 < order(i)) & (order(i) <= t))
         ykr = ykr + ( (y(:,k) - yuk)' * ...
               vs(:,order(i)) .* vs(:,order(i)));
      end
   end
      ykr = ykr + yuk;
end
```

2.4.3 Synthetic Data Example I

Two contiguous sections are synthetically generated from (2.28) and (2.29) and they are spliced as presented in Figure 2.19(a). In this plot contribution from various variance parts are not well visualized.

$$y_d \sim \mathbf{N}\left(5 \times ones(51,1), diag\left([0.01 \cdots 0.01\ 0.05\ 0.1\ 0.2\ 0.4]\right)\right),\ (d = 1, \cdots, 250) \tag{2.28}$$

$$y_d \sim \mathbf{N}\left(1 \times ones(51,1), diag\left([0.4\ 0.2\ 0.1\ 0.05\ 0.01 \cdots 0.01]\right)\right),\ (d = 251, \cdots, 500) \tag{2.29}$$

On this image local PCA is performed for the 1^{st} eigenvector with different weighting shapes as shown in Figure 2.19.

2.4.4 Synthetic Data Example II

Another synthetic data are made from (2.20) but for d = 1,..., 500. The three primary eigenvectors are considered on this data. In Figure 2.20 computational results made from local PCA analysis for $h = 30$ are presented. In this try it is observed that all the three eigenvectors basically attain high direction cosine over the entire processing region.

2.4.5 Openhole Sonic Data Examples

In this section we reconsider the same data processed in Section 2.2.4 with use of localized PCA techniques.

Reconstruction of Reflection Events

Local PCA results corresponding to Figures 2.5, 2.6, and 2.7 are presented in Figures 2.21, 2.22, and 2.23, respectively. Eigenvector images from local PCA consist of depth-dependent components due to locality. It should be remarked that the localized mean vectors shown in Figure 2.23(a) preserve depth-dependent waveform features clearly when comparing to Figure 2.7(a).

The three major eigenvalues are examined over the depth axis as depicted in Figure 2.24(a). These eigenvalues are accumulated and their contribution from the three major components is plotted in Figure 2.24(b). It is found that the reconstructed

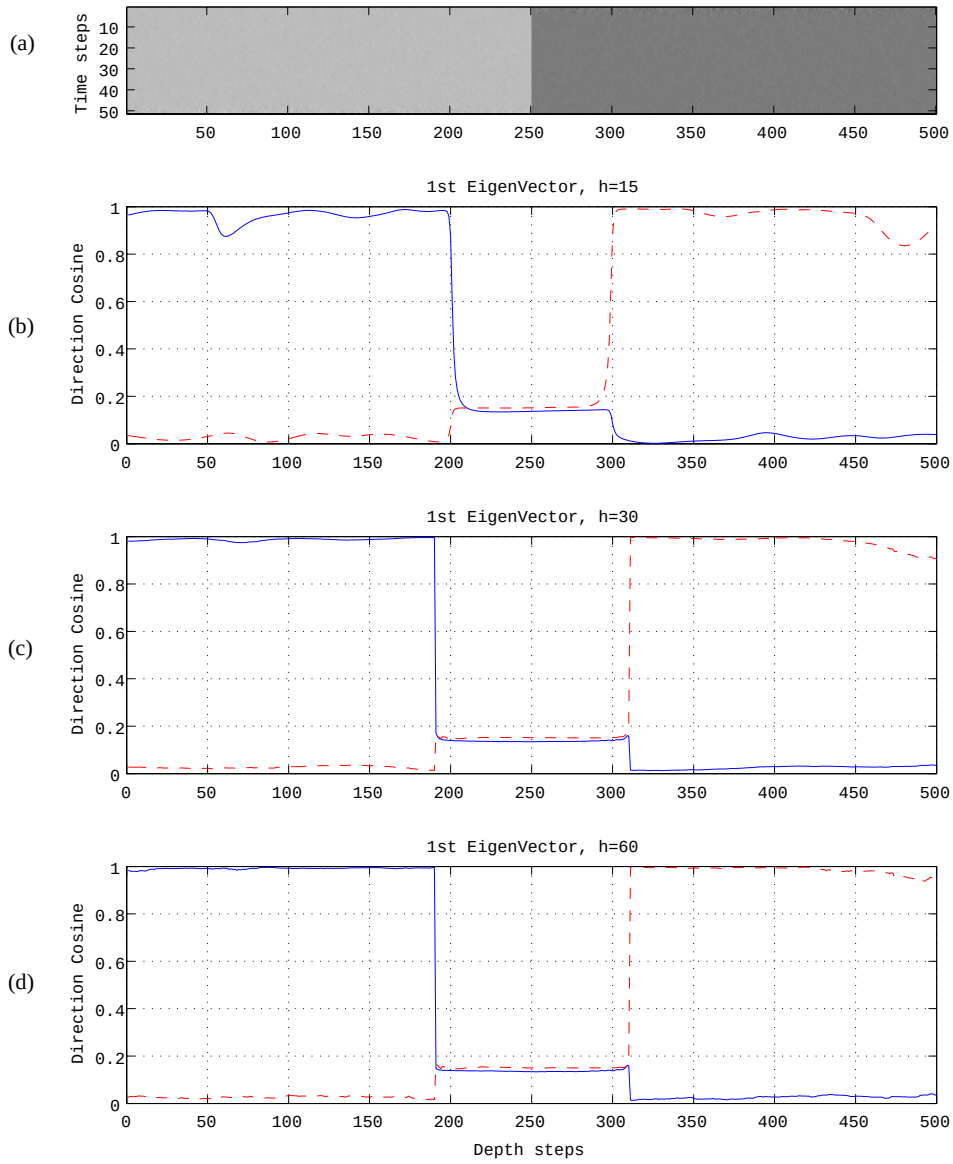

Figure 2.19: Synthetic images (a) and its local PCA analysis with different h values; $h = 15$ (b), $h = 30$ (c), and $h = 60$ (d). A processing window is set to have 121 weighting factors.

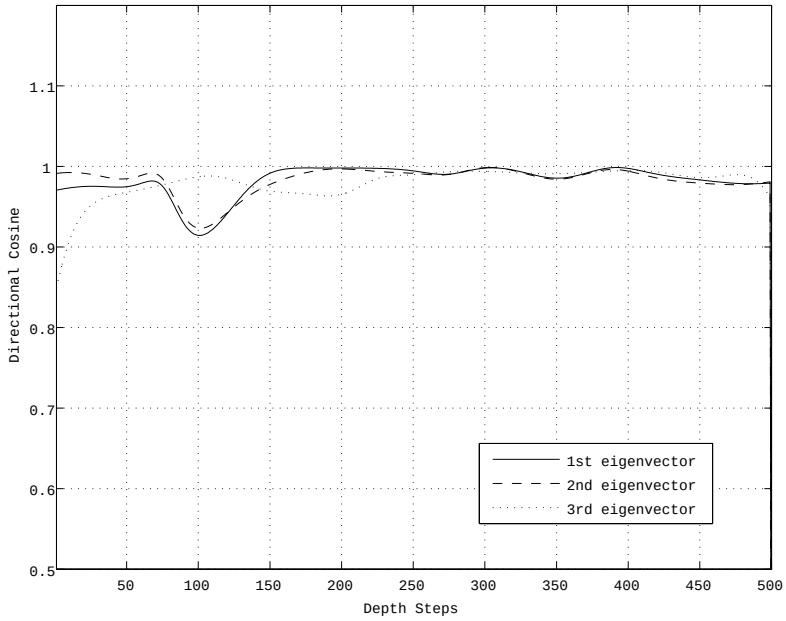

Figure 2.20: Local PCA analysis for synthetic data from (2.20) but for d=1,...,500. The three primary eigenvectors are computed and compared with true eigenvectors. A parameter h is set to have 30 in the localized weighting function in (2.22).

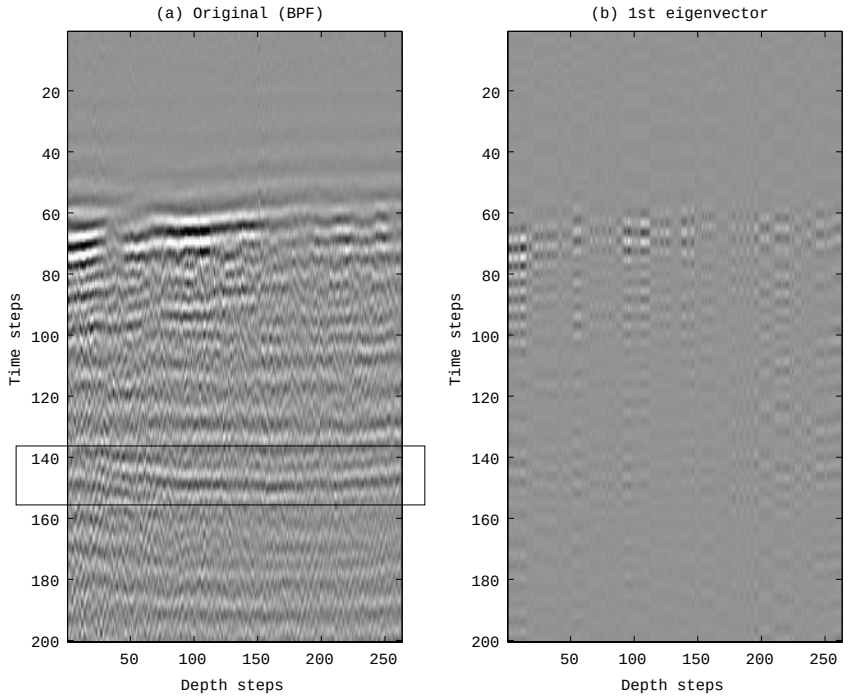

Figure 2.21: (a) Original common-offset gather. For this data local PCA is performed with the selection of $h = 15$, which is used in the localized weighting function of (2.22). A rectangular encloses reflection signals coming from a reflector near the borehole. (b) Reconstructed data traces from the 1^{st} eigenvector.

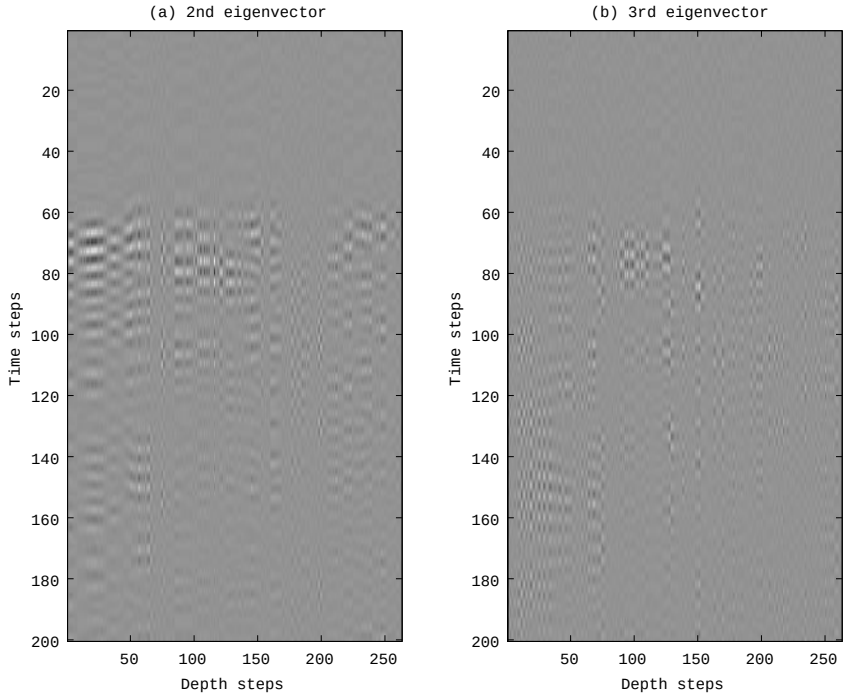

Figure 2.22: (a) Reconstructed data traces from the 2^{nd} eigenvector. (b) Reconstructed data traces from the 3^{rd} eigenvector.

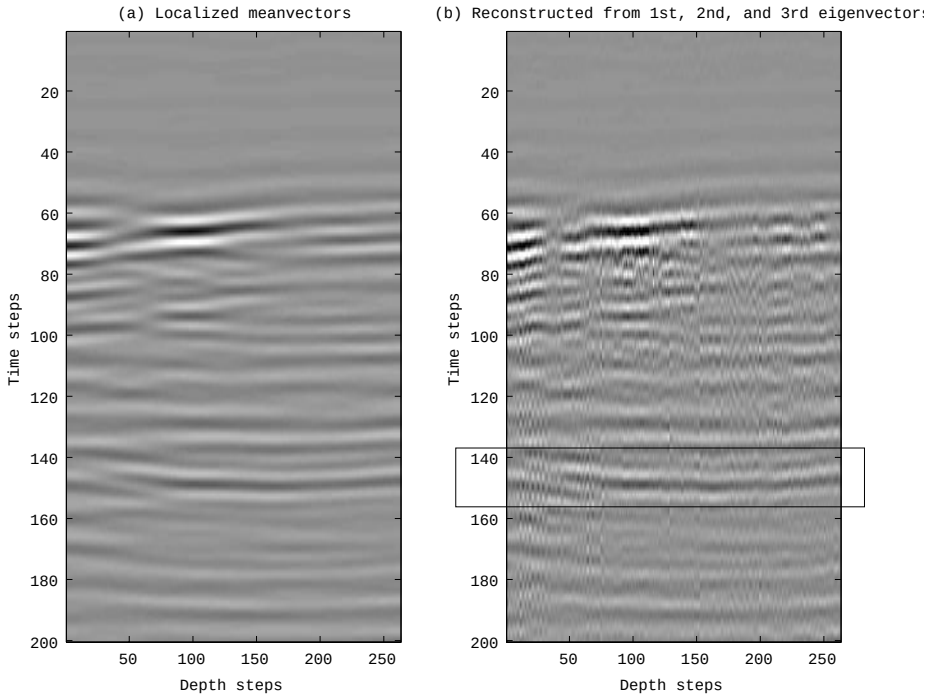

Figure 2.23: (a) Mean data traces computed from local PCA with $h = 15$. (b) Reconstructed data traces made from three primary eigenvectors and the localized mean traces. A rectangular encloses reconstructed reflection signals.

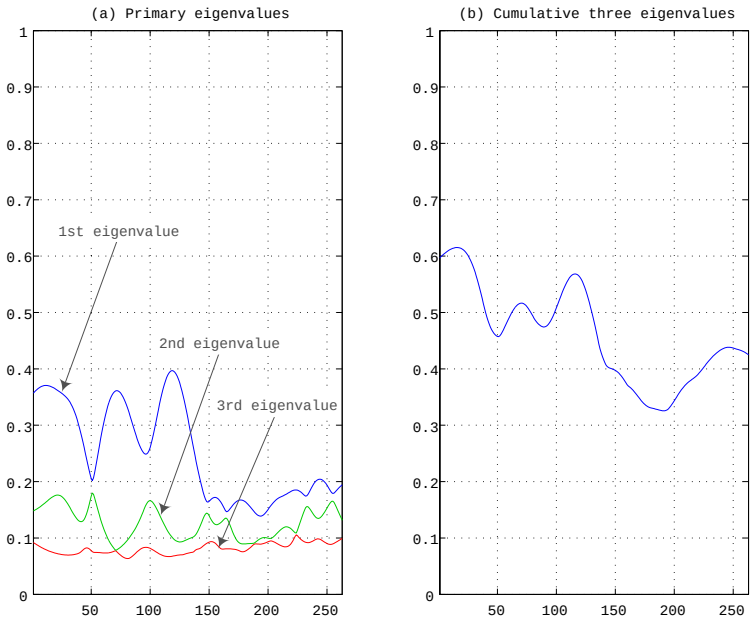

Figure 2.24: (a) The three primary eigenvalues over the depth steps. (b) Cumulative eigenvalues from the three primary eigenvalues.

images from the 1^{st}, 2^{nd}, and 3^{rd} eigenvectors hold about a half of information from the original images in Figure 2.21(a).

In order to verify directionality of three major eigenvectors over the depth axis, directional cosine values are continuously computed with referencing to a previous eigenvector. Figure 2.25 presents the computational results. It is found that basically each eigenvector preserves the same direction except for selective regions. This reinforces the necessity to introduce localization operations to capture local features.

It is observed that, when comparing to Figure 2.7(b), Figure 2.23(b) shows better images in a sense that the reflector shape is much closer to the original. Table 2.5 shows MSE values for three vector cases – the sample mean vector (2.4), the localized mean vector (2.23), and the reconstructed vector (2.26) with $p = 3$. It can be concluded that MSE from the reconstructed vector with $p = 3$ in local PCA case attains the minimum among three estimators on this data set.

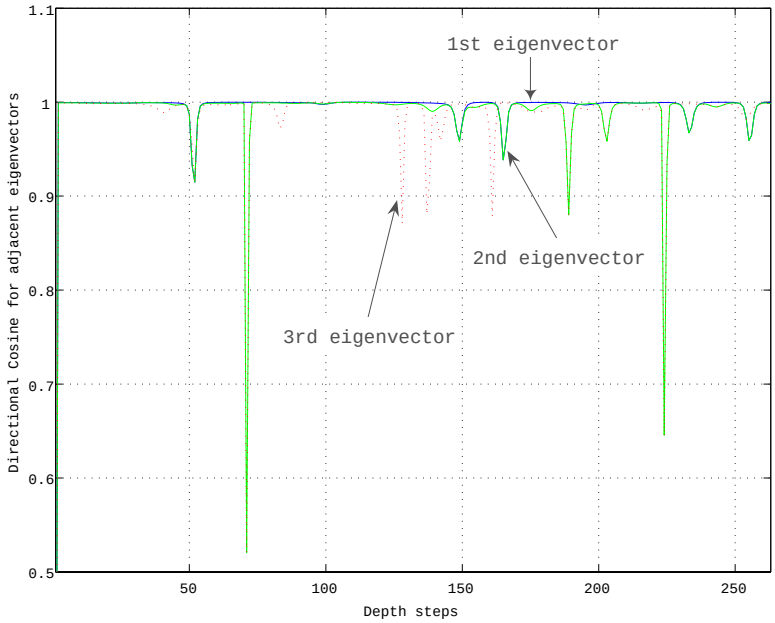

Figure 2.25: Directional cosine values with referencing to a previous eigenvector. The three primary eigenvectors are considered.

Local PCA Effects on Migrated Image

In order to see effects of local PCA on migrated images we have applied localized PCA on a migrated section which is conventionally processed as pre-stack migration. Figure 2.26 shows the original migrated image in which a solid line represents a well trajectory. The migrated image is presented down to 35 ft below the trajectory. This migrated image is generated from eight-receiver data (Yamamoto *et al.* [18]).

In Figure 2.27 a corresponding image that has local PCA operations with a weighting parameter of $h = 10$ is presented. While reconstructing the image three primary eigenvectors are taken into account. The imaging quality of the major reflector in Figure 2.27 is similar to one in Figure 2.26. However it is observed that subtle criss-cross like marks caused from migration operations are suppressed and relatively less noisy image is generated.

As a future experiment it is worth trying that local PCA is to be performed on raw data (i.e., original eight-receiver data) and then resultant data are migrated.

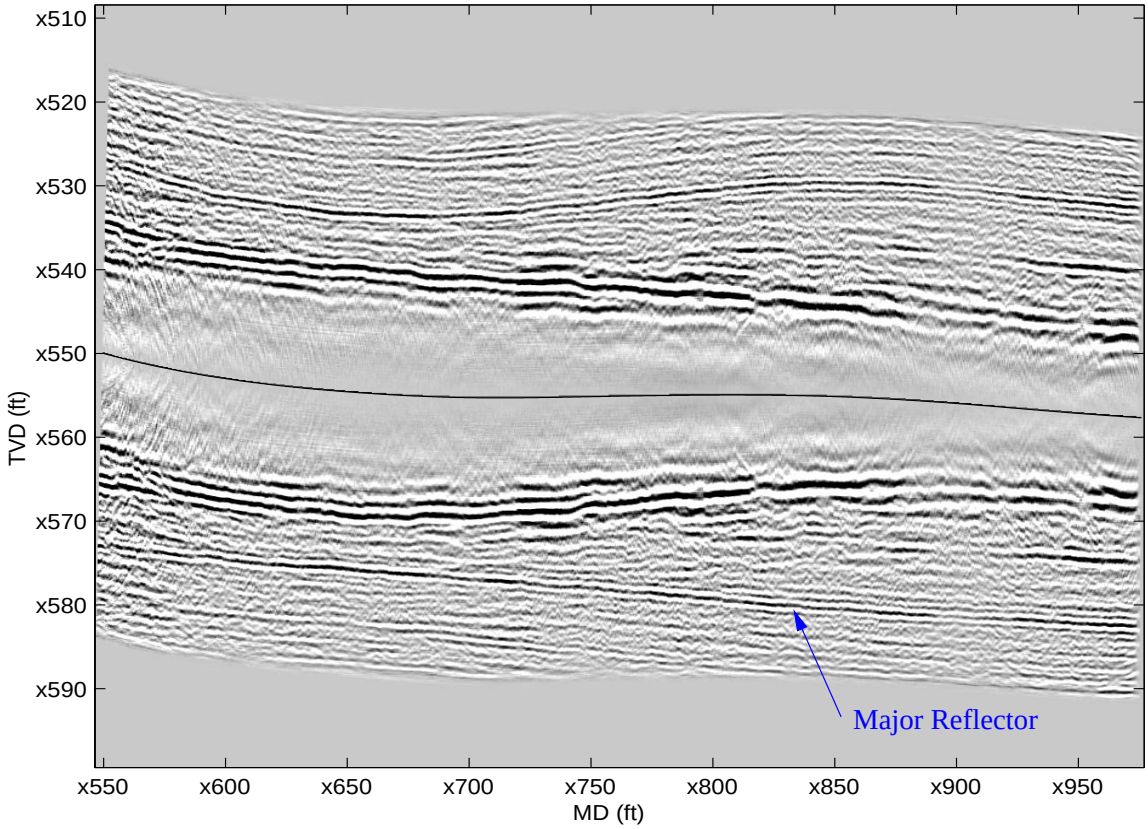

Figure 2.26: A conventional migrated image. A solid line represents the well trajectory, and migrated images are mirrored on either side of this line.

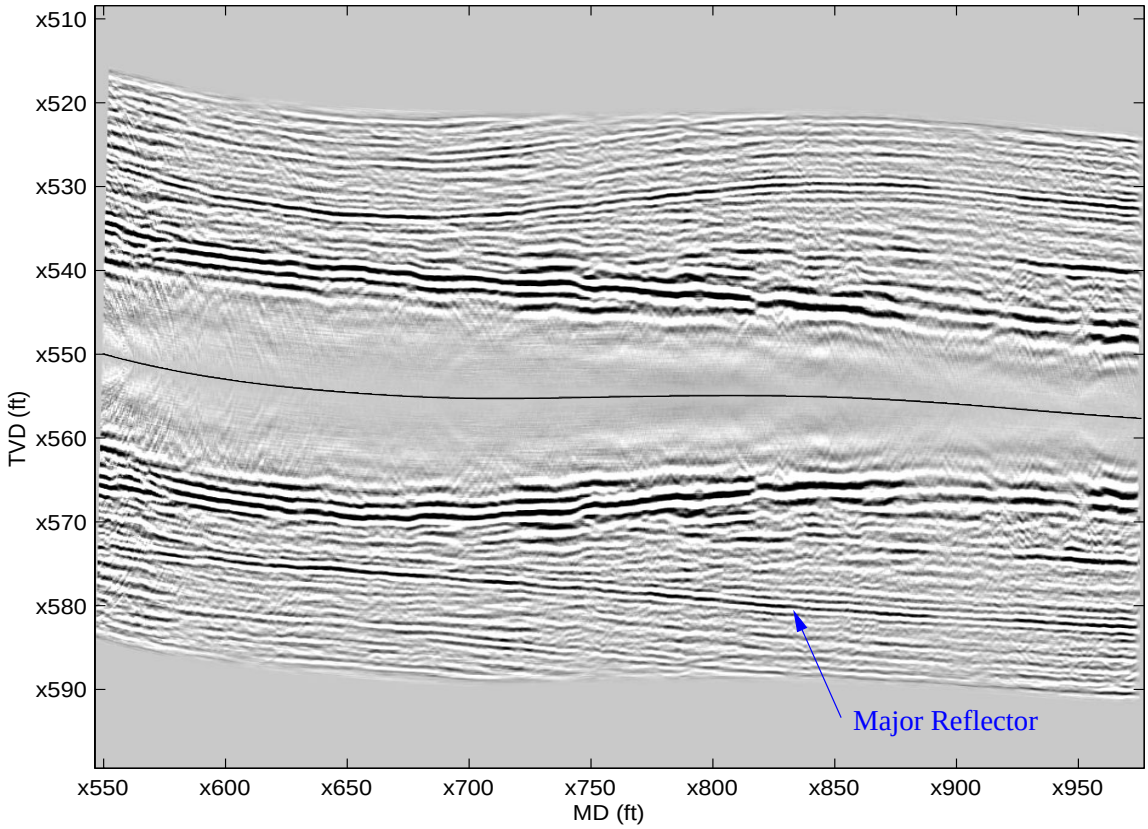

Figure 2.27: Migrated images with local PCA operation. Three primary eigenimages are used to reconstruct the image. A solid line represents the well trajectory, and migrated images are mirrored on either side of this line.

Error from	Equation	MSE
Sample mean vector (2.4)	$\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}_d-\bar{\mathbf{y}})^T(\mathbf{y}_d-\bar{\mathbf{y}})$	40.79
Localized mean vector (2.23)	$\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}_d-\mathbf{y}^*(d))^T(\mathbf{y}_d-\mathbf{y}^*(d))$	20.75
Reconstructed vector (2.26) with $p=3$	$\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}_d-\tilde{\mathbf{x}}_d[3])^T(\mathbf{y}_d-\tilde{\mathbf{x}}_d[3])$	11.88

Table 2.5: MSE: errors from different reconstructed methods.

Chapter 3

Local Likelihood Regression

This chapter deals with a regression model in a two-dimensional (2-d) domain and a local likelihood approach for estimating parameters in the model. The localization is realized by introducing a kernel function. Cross validation (CV) estimates give an insight into obtaining optimal bandwidth for the localization. We apply the method to acoustic well-logging data and demonstrate noise reduction with the optimal bandwidth selection on a 2-d (space, time) domain. The method is generic enough to apply to various other types of data without restriction to geophysical data.

3.1 Introduction

As an application of acoustic well-logging, a sonic imaging technique was proposed and applied (Hornby [7]; Esmersoy *et al.* [4]) to delineate subsurface bedding boundaries. Then experiments were conducted to delineate acoustic reflectors near the borehole with a special imaging tool and with processing software similar to that used in surface seismic processing (Watanabe *et al.* [17]; Yamamoto *et al.* [18]). To get clear subsurface migrated images, it is essential to perform (1) removal of head waves and multiples; and (2) noise reduction on acoustic well-logging image data before migration operations. For the former, techniques such as wave separation, predictive deconvolution, frequency-wavenumber (f-k) filters, and mean/median filters were applied; however, for the latter, no practical attempt except for high-cut

filters was made to suppress noise that blurs reflected signals.

There are methods that have been applied for the feature extraction or the suppression of unwanted signals in geophysical data processing domains. Wavelet denoising methods have been widely applied. Lately a 2-d seismic wavelet from the physical wavelet frame (Zhang and Ulrych [20]) has been applied to seismic data. PCA and the closely related Karhunen-Loève (K-L) transform have also been extensively used for various purposes including noise suppression in the acoustic measurement domain. Yilmaz [19] demonstrated the K-L use in attenuating multiples and also rejecting random noise for seismic data by simply eliminating the corresponding eigenimages. Hsu [8] applied the K-L for separating compressional, shear, and Stoneley waves from noise and reflected converted waves in acoustic well-logging waveforms. PCA is powerful to extract features over a processing region depending on selected eigenvectors and corresponding eigenvalues, however in case for processing a large data or focusing a specific region one will introduce a processing window. Then it comes down to a question: What's an optimal window size? Localization of analysis is one of the essential keys to both making the computational burden light and leading to more accurate features in the focused window. It is also desired to have some rationale when selecting the window size.

We propose a new approach to reconstructing 2-d images by introducing a regression model in a localized sense. In this chapter, we intend to (1) formulate our model and estimation methods and to (2) present their applicability to real acoustic well-logging image data. Therefore, we first formulate the local regression model in the 2-d (space, time) domain and deal with an optimal parameter selection. In order to measure the performance of model fitting we introduce the leave-one-out cross validation (CV) criteria (Section 2.4 in Loader [10]). Then we apply the local regression model to real acoustic well-logging data and present estimated results. The use of CV is attempted and an optimal parameter, that defines the degree of locality in the selected model, is obtained from a CV plot. This single bandwidth selection is then extended to multiple bandwidth selection that takes into account the variant locality over the time domain. Our results demonstrate that the applied method successfully achieved: (1) identification of continuous reflectors, and (2) suppression of random

noise in the acquired acoustic data.

3.2 Formulation

3.2.1 Single Bandwidth Selection

Suppose that we observe sound pressure y_t from a wave component of acoustic well-logging data at time t with a depth at d. Let $\mathbf{y} = (y_1, \cdots, y_T)^T$ with components y_t over time. We would like to extract an intrinsic relationship of d and $\mathbf{y}$, deleting apparent noise. In general, the objective of regression analysis is to estimate the conditional expectation of the response variable given the covariate data. In our context, response variables form vector $\mathbf{y}$ and the covariate is a function of the observed depth d, so that the regression function to be estimated is $\mathbf{m}(d) = E(\mathbf{Y}|D = d)$. The standard regression is formulated to have a univariate response variable, while our analysis has to employ the vector case $\mathbf{y}$. However, we will see that an almost standard formulation is easily applicable to our vector case.

We assume a k^{th} order polynomial model $\mathbf{p}(d, \mathbf{B})$ for the regression function $\mathbf{m}(d)$ by

$$\mathbf{p}(d, \mathbf{B}) = \mathbf{B}\mathbf{x}(d) = \mathbf{b}_0 + d\mathbf{b}_1 + \cdots + d^k\mathbf{b}_k, \tag{3.1}$$

where $\mathbf{B} = (\mathbf{b}_0, \mathbf{b}_1, \cdots, \mathbf{b}_k)$ is a $T \times (k+1)$ matrix of regression coefficients and $\mathbf{x}(d) = (1, d, \cdots, d^k)^T$ is a $(k+1)$ vector of k-polynomial covariates. Therefore we get a regression model

$$\mathbf{y}_i = \mathbf{p}(d_i, \mathbf{B}) + \mathbf{n}_i, (i = 1, \cdots, I) \tag{3.2}$$

where $\mathbf{n}_i$ is an error vector. The usual least-squares estimator is obtained by minimizing the sum of squares. However, good fitting of the model to a real dataset can not be expected unless the model sufficiently reflects the random variability of the dataset. In practice our dataset has a complex structure consisting of various wave components such as compressional, shear, and Stoneley waves. Therefore, it would be very difficult to suggest a parametric approach appropriate for our dataset.

Recently nonparametric methodology in the statistical community has been extensively developed. A basic assumption is that the regression function $\mathbf{m}(d)$ is a smooth vector function of d without imposing any parametric form such as (3.1). Let us take an approach by local likelihood as one of the most promising methods. The key idea is to connect the parametric and nonparametric methods by using a kernel function $K(\cdot, d; h)$, which is designed to be decreased smoothly while moving away from the depth d. In fact, we select the kernel function by

$$K(\tilde{d}, d; h) = \exp\left\{-\frac{(d-\tilde{d})^2}{2h^2}\right\}. \tag{3.3}$$

We note that for a large bandwidth h the kernel function $K(\tilde{d}, d; h)$ almost gives a constant weight 1 for all $\tilde{d}$ while for a small h it gives near-zero weight for $\tilde{d}$ away from d.

The local likelihood method leads to the kernel-weighted sum of squares

$$L_d(\mathbf{B}; h) = \frac{1}{2}\sum_{i=1}^{I} K(d, d_i; h)\|\mathbf{y}_i - \mathbf{p}(d_i, \mathbf{B})\|^2. \tag{3.4}$$

For the target depth d, the i^{th} observation $(d_i, \mathbf{y}_i)$ is weighted by the kernel function. Hence the more remote the depth d_i is from d, the less weight $\mathbf{y}_i$ is assigned. In fact the original proposal doesn't use the Gaussian kernel defined in (3.3) (Tibshirani and Hastie [15]) but a uniform kernel that is one if $d-h \le \tilde{d} \le d+h$ and zero otherwise. Then $L_d(\mathbf{B}; h)$ is the sum of squares based only on the subdataset with the depth d_i in the interval $(d-h, d+h)$. For $k=0$ it is called the scatterplot smoother. If the bandwidth h is properly selected, we can expect that $L_d(\mathbf{B}; h)$ with the Gaussian kernel includes the local information around the target depth d, smoothly balanced with the global information.

The local likelihood estimator for $\mathbf{B}$ is defined by the minimizer of $L_d(\mathbf{B}; h)$ over

$\mathbf{B}$, say $\hat{\mathbf{B}}(d,h)$. In practice it is explicitly given by

$$\hat{\mathbf{B}}(d,h) = \left(\sum_{i=1}^{I} K(d,d_i;h)\mathbf{y}_i\mathbf{x}(d_i)^T\right)\left(\sum_{j=1}^{I} K(d,d_j;h)\mathbf{x}(d_j)\mathbf{x}(d_j)^T\right)^{-1}. \tag{3.5}$$

We therefore obtain the estimate of $\mathbf{m}(d)$ by $\hat{\mathbf{m}}_h(d) = \mathbf{p}(d, \hat{\mathbf{B}}(d,h))$. This estimator is a linear estimator of $\mathbf{y}_i$ with coefficients depending on d and h. In fact we can express $\hat{\mathbf{m}}_h(d)$ by

$$\hat{\mathbf{m}}_h(d) = \sum_{i=1}^{I} \ell_i(d,h)\mathbf{y}_i, \tag{3.6}$$

where

$$\ell_i(d,h) = K(d,d_i;h)\mathbf{x}(d_i)^T\left(\sum_{j=1}^{I} K(d,d_j;h)\mathbf{x}(d_j)\mathbf{x}(d_j)^T\right)^{-1}\mathbf{x}(d_i). \tag{3.7}$$

The simplest case of $k = 0$ gives a scatterplot smoother

$$\ell_i(d,h) = \frac{K(d,d_i;h)}{\sum_j K(d,d_j;h)}.$$

We have formulated our method using the idea of local likelihood. The role of bandwidth h is the central idea of localizing the likelihood function by the kernel weighting. Thus the bandwidth, h, controls the degree of localization. For a smaller h, the resulting regression vector gives larger variance; for larger h, it gives larger bias.

To find a reasonable locality of the model adaptability, we consider the cross validation defined by

$$\mathrm{CV}(h) = \frac{1}{I}\sum_{i=1}^{I} \|\mathbf{y}_i - \hat{\mathbf{m}}_h^{(-i)}(d_i)\|^2, \tag{3.8}$$

where $\hat{\mathbf{m}}_h^{(-i)}(d_i)$ denotes a local likelihood estimate of $\mathbf{m}(d)$ based on the leave-d_i-out dataset. Noting that

$$\hat{\mathbf{m}}_h^{(-i)}(d) = \mathbf{p}(d, \hat{\mathbf{B}}_{d,h}^{(-i)})$$

with the leave-d_i-out estimate $\hat{\mathbf{B}}_{d,h}^{(-i)}$ for $\mathbf{B}$, we can, by straightforward calculus on matrix algebra, express

$$\mathrm{CV}(h) = \frac{1}{I}\sum_{i=1}^{I} \frac{\|\mathbf{y}_i - \hat{\mathbf{m}}_h(d_i)\|^2}{\{1-\ell_i(d_i,h)\}^2}. \tag{3.9}$$

This formula is easily obtained by the result of Henderson's theorem (Section 2.5 in Loader, 1999). See Appendix for a self-contained proof. From this we observe that the empirical error

$$\mathrm{Error}(h) = \frac{1}{I}\sum_{i=1}^{I} \|\mathbf{y}_i - \hat{\mathbf{m}}_h(d_i)\|^2 \tag{3.10}$$

is simply adjusted by a scalar function $\{1-\ell_i(d_i,h)\}^2$. We select an optimal h defined by

$$\hat{h}_{\mathrm{opt}} = \operatorname*{argmin}_{h>0} \mathrm{CV}(h).$$

As a post analysis, we investigate the fitness of our modeling to the dataset. For this the effective degree of freedom (EDF),

$$\mathrm{EDF}(k) = \sum_{i=1}^{I} \ell_i(d_i, h_{\mathrm{opt}}), \tag{3.11}$$

is useful for the assessment. Figure 3.1 summarizes processing steps and key functions described in this section. It should be noted that influence functions and EDF are used to evaluate the validity for applied models.

3.2.2 Multiple Bandwidth Selection

Our main proposal is to extend the single bandwidth h method to a multiple case. This is implemented by separating the vector regression model, (3.2), into independent component regression model

$$y_{ti} = p_t(d_i, \mathbf{b}_t^*) + n_{ti}$$

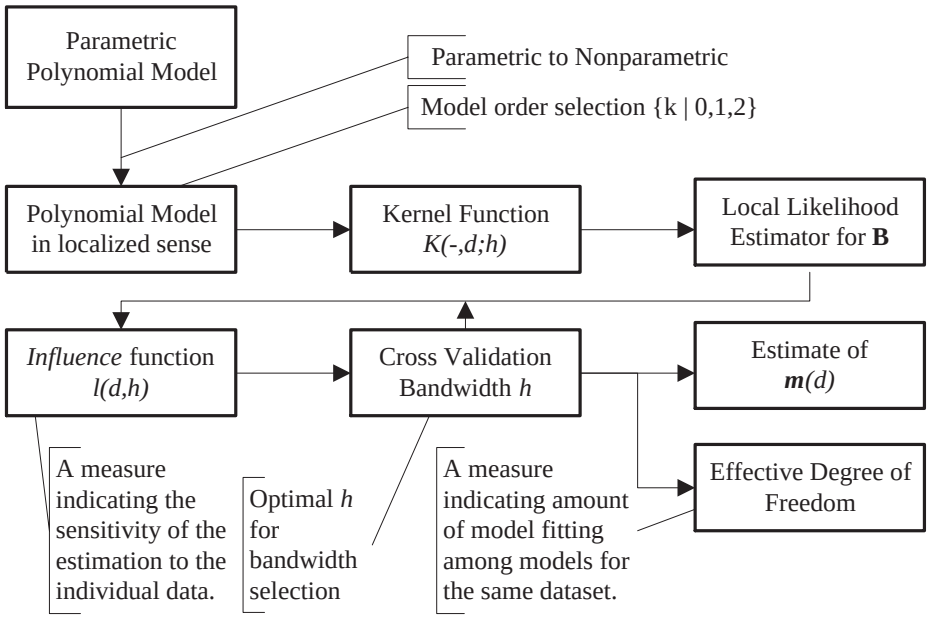

Figure 3.1: Processing steps and key functions for localized regression methods.

for $t = 1, \cdots, T$ and $i = 1, \cdots, I$, where $p_t(d_i, \mathbf{b}_t^*) = \mathbf{x}(d_i)^T \mathbf{b}_t^*$ and $\mathbf{b}_t^*$ is the t^{th} row vector of $\mathbf{B}$ with the matrix $\mathbf{B}$ of regression coefficients defined in (3.2). Let $\mathbf{h} = (h_1, \cdots, h_T)^T$ be a vector of bandwidths and let

$$L_d^*(\mathbf{B}; \mathbf{h}) = \frac{1}{2} \sum_{i=1}^{I} \sum_{t=1}^{T} K(d, d_i; h_t) \{y_{ti} - p_t(d_i, \mathbf{b}_t^*)\}^2. \tag{3.12}$$

If the vector $\mathbf{h}$ equals $(h, \cdots, h)^T$ then (3.12) reduces to (3.4). The minimization of (3.12) over $\mathbf{B}$ is easily obtained by separating into those of

$$\tilde{L}_d(\tilde{\mathbf{b}}; h_t) = \frac{1}{2} \sum_{i=1}^{I} K(d, d_i; h_t) \{y_{ti} - p_t(d_i, \mathbf{b}_t^*)\}^2. \tag{3.13}$$

Hence the estimator is $\hat{\mathbf{B}}(d, \mathbf{h}) = (\hat{\mathbf{b}}_1^*, \cdots, \hat{\mathbf{b}}_T^*)^T$, where

$$\hat{\mathbf{b}}_t^*(d, h_t) = \left(\sum_{i=1}^{I} K(d, d_i; h_t) y_{ti} \mathbf{x}(d_i)^T \right) \left(\sum_{j=1}^{I} K(d, d_j; h_t) \mathbf{x}(d_j) \mathbf{x}(d_j)^T \right)^{-1}. \tag{3.14}$$

By an argument similar to the derivation of (3.9) we get an interpretable expression of the cross-validated sum of squares by

$$\mathrm{CV}(\mathbf{h}) = \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\{y_{ti} - \hat{m}(d_i, h_t)\}^2}{\{1 - \ell_i(d_i, h_t)\}^2} \tag{3.15}$$

where $\hat{m}_t(d, h_t) = p_t(d, \hat{\mathbf{b}}_t^*(d, h_t))$. The effective degree of freedom is given by

$$\mathrm{EDF}(k) = \frac{1}{T}\sum_{i=1}^{I}\sum_{t=1}^{T}\ell_i(d_i, h_{t_{\mathrm{opt}}}). \tag{3.16}$$

We must pay attention to some effects of overfitting in the vector bandwidth case, as described above. To relax the overfitness, let us divide the time range into N-sub intervals, $\mathcal{T}_1, \cdots, \mathcal{T}_N$, with length τ. Thus we proposed the revised version for (3.12) as

$$L_d^*(\mathbf{B}; \mathbf{h}^*) = \frac{1}{2}\sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{j=1}^{N}K(d, d_i; h_j)1(t, \mathcal{T}_j)\{y_{ti} - p_t(d_i, \mathbf{b}_t^*)\}^2, \tag{3.17}$$

where $\mathbf{h}^* = (h_1^*, \cdots, h_N^*)^T$ and $1(t, \mathcal{T})$ is an indicator function of $\mathcal{T}$. Figure 3.2 illustrates single and multiple bandwidth selections.

This revised version reflects that image data at the present time t would be influenced by those at both past and future points near t. In Section 3.3 we will apply our method to real data.

3.2.3 Matlab Codes

In this section implemented Matlab codes are listed. They are *regLocal*, *regInfl*, and *lrCV*; and are summarized in Table 3.1.

■ *regLocal*

```
function [yud] = regLocal( x, d, ld, h, order )
%
% function [yud] = regLocal( x, d, ld, h, order )
```

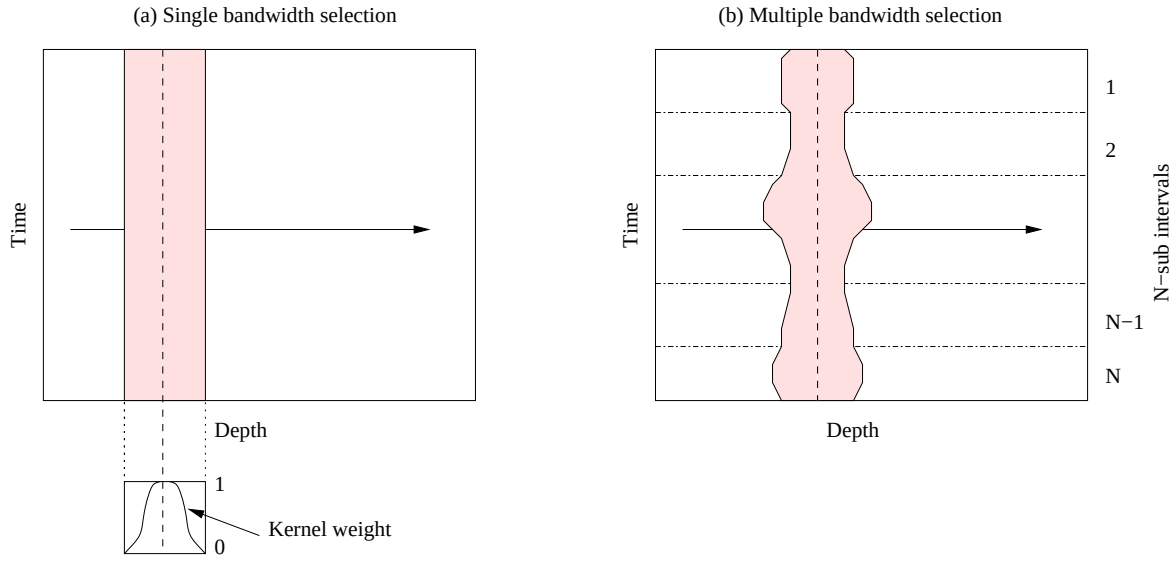

Figure 3.2: Single and multiple bandwidth selections. In (a) single selection, constant depth range is used with the kernel weight. In case for (b) multiple bandwidth selection, time-variant depth range is applied.

Name	Descriptions
regLocal	computes localized regression vector.
regInfl	computes influence values.
lrCV	computes leave-one-out estimation for cross-validation.

Table 3.1: Matlab functions to perform localized regression methods.

```
% computes localized regression vector at 'd'
%
% Input:
%       x - sample waveform matrix whose
%           dimension is (nt,nd) where
%           nt is a number of time steps and
%           nd is a number of space (depth) steps.
%       d - a depth index at which localized regression is
%           performed.
%       ld - a processing window width.
%           0 < ld <= 2*nd
%       h - a scalar to specify the shape of
%           the localized weight.
%       order - regression order {order|0,1,2,3}
%
```

```
% Output:
%        yud - a localized regression vector
%
% Remark:
%
%    kh() is used for the kernel function.
%

[nt, nd] = size(x);
hd = round((ld-1)/2);

  prs = max([d-hd  1]);  % processing start position
  pre = min([d+hd nd]);  % processing end   position
  pld = pre - prs + 1;

  xo  = x(:,prs:pre);
  khv = kh(prs-d:pre-d, h); % Weighting matrix
  sul = sum(khv);
  khw = diag(khv);

  if order == 0     % regression order = 0

     yud = (sum(xo*khw,2)/sul);

  elseif order ==1  % regression order = 1

     dp = [prs:pre];
     A = sum(xo*khw,2);
     B = sul;
     C = dp * khv';
     D = sum(xo*diag(diag(dp'*khv)),2);
     E = (dp .^2) * khv';

     alpha = (A*C-B*D) / (C^2 - B*E);
     beta  = (A*E-C*D) / (B*E - C^2);

     yud = alpha*d + beta;

   elseif order == 2  % regression order = 2

    dp = [1:pld];
    A = sum(xo*khw,2);
    B = sul;
    C = dp * khv';
    D = sum(xo*diag(diag(dp'*khv)),2);
    E = (dp .^2) * khv';
    F = sum(xo*diag(diag((dp .^2)'*khv)),2);
    G = (dp .^3) * khv';
    H = (dp .^4) * khv';

    X = [B C E; C E G; E G H];
    IX = inv(X);

    alpha = IX(3,1) * A + IX(3,2) * D + IX(3,3) * F;
    beta  = IX(2,1) * A + IX(2,2) * D + IX(2,3) * F;
```

```
    gamma = IX(1,1) * A + IX(1,2) * D + IX(1,3) * F;

    yud = alpha*(d-prs+1)^2 + beta*(d-prs+1) + gamma;

  elseif order == 3  % regression order = 3

    dp = [1:pld];
    A = sum(xo*khw,2);
    B = sul;
    C = dp * khv';
    D = sum(xo*diag(diag(dp'*khv)),2);
    E = (dp .^2) * khv';
    F = sum(xo*diag(diag((dp .^2)'*khv)),2);
    G = (dp .^3) * khv';
    H = (dp .^4) * khv';
    I = (dp .^5) * khv';
    J = sum(xo*diag(diag((dp .^3)'*khv)),2);
    K = (dp .^6) * khv';

    X = [B C E G; C E G H; E G H I; G H I K];
    IX = inv(X);

    alpha = IX(4,1) * A + IX(4,2) * D + IX(4,3) * F + IX(4,4) * J;
    beta  = IX(3,1) * A + IX(3,2) * D + IX(3,3) * F + IX(3,4) * J;
    gamma = IX(2,1) * A + IX(2,2) * D + IX(2,3) * F + IX(2,4) * J;
    delta = IX(1,1) * A + IX(1,2) * D + IX(1,3) * F + IX(1,4) * J;

    yud = alpha*(d-prs+1)^3 + beta*(d-prs+1)^2 + ...
          gamma*(d-prs+1) + delta;

   end
```

■ *regInfl*

```
function [infl] = regInfl( x, d, ld, h, order )
%
% function [infl] = regInfl( x, d, ld, h, order )
% computes influence values.
%
% See regLocal for descriptions of input parameters.
%
% S. Watanabe
%
% May 16, 2001 Initial coding.

  [nt, nd] = size(x);
  hd = round((ld-1)/2);

  prs = max([d-hd  1]);  % processing start position
  pre = min([d+hd nd]);  % processing end   position
  pld = pre - prs + 1;
  ppd =   d - prs + 1;
```

```
khv = kh(prs-d:pre-d, h); % Weighting matrix
sul = sum(khv);
khw = diag(khv);

if order == 0     % regression order = 0

   infl =  khv(ppd)/sul;

elseif order == 1 % regression order = 1

   x = [1 d]';

   S = zeros(2,2);
   for k=1:pld
     dj = prs+k-1;
     S = S + khv(k)*[1 dj]'*[1 dj];
   end

   infl = khv(ppd)*x'*inv(S)*x;

elseif order == 2 % regression order = 2

   x = [1 d d^2]';

   Q = zeros(3,3);
   for k=1:pld
     dj = prs+k-1;
     Q = Q + khv(k)*[1 dj dj^2 ]'*[1 dj dj^2];
   end

   infl = khv(ppd)*x'*inv(Q)*x;

elseif order == 3 % regression order = 3

   x = [1 d d^2 d^3]';

   R = zeros(4,4);
   for k=1:pld
     dj = prs+k-1;
     R = R + khv(k)*[1 dj dj^2  dj^3]'*[1 dj dj^2 dj^3];
   end

   infl = khv(ppd)*x'*inv(R)*x;

end
```

■ *lrCV*

```
function [yud] = lrCV( x, d, ld, h ,order)
%
% function [yud] = lrCV( x, d, ld, h ,order)
% computes leave-one-out estimation at depth 'd'.
%
```

```
% Input:
%        x - sample waveform matrix whose
%            dimension is (nt,nd) where
%            nt is a number of time steps and
%            nd is a number of space (depth) steps.
%        d - a depth index at which locarized regression is
%            performed.
%        ld - a processing window width.
%            0 < ld <= 2*nd
%        h - a scalar to specify the shape of
%            the localized weight.
%        order - regression order
%          0: a
%          1: a*d  +b
%          2: a*d^2+b*d+c
%          3: a*d^3+b*d^2+c*d+e
%
% Output:
%        yud - estimated mean vector
%
% S. Watanabe
%

[nt, nd] = size(x);
hd = round((ld-1)/2);

  prs = max([d-hd  1]);  % processing start position
  pre = min([d+hd nd]);  % processing end   position
  pld = pre - prs + 1;

  xo  = x(:,prs:pre);
  khv = kh(prs-d:pre-d, h); % weighting matrix
  khv(find(khv == 1)) = 0;  % zero out for the weight
                            % at position 'd'
  sul = sum(khv);
  khw = diag(khv);

  if(order == 0)  % regression order = 0;

     yud = (sum(xo*khw,2)/sul);

  elseif (order == 1)  % regression order = 1;

   dp = [prs:pre];
   A = sum(xo*khw,2);
   B = sul;
   C = dp * khv';
   D = sum(xo*diag(diag(dp'*khv)),2);
   E = (dp .^2) * khv';

   alpha = (A*C-B*D) / (C^2 - B*E);
   beta  = (A*E-C*D) / (B*E - C^2);

   yud = alpha*d + beta;
```

```
elseif (order == 2)  % regression order = 2;

 dp = [1:pld];
 A = sum(xo*khw,2);
 B = sul;
 C = dp * khv';
 D = sum(xo*diag(diag(dp'*khv)),2);
 E = (dp .^2) * khv';
 F = sum(xo*diag(diag((dp .^2)'*khv)),2);
 G = (dp .^3) * khv';
 H = (dp .^4) * khv';

 X = [B C E; C E G; E G H];
 IX = inv(X);

 alpha = IX(3,1) * A + IX(3,2) * D + IX(3,3) * F;
 beta  = IX(2,1) * A + IX(2,2) * D + IX(2,3) * F;
 gamma = IX(1,1) * A + IX(1,2) * D + IX(1,3) * F;

 yud = alpha*(d-prs+1)^2 + beta*(d-prs+1) + gamma;

elseif (order == 3)  % regression_order = 3;

 dp = [1:pld];
 A = sum(xo*khw,2);
 B = sul;
 C = dp * khv';
 D = sum(xo*diag(diag(dp'*khv)),2);
 E = (dp .^2) * khv';
 F = sum(xo*diag(diag((dp .^2)'*khv)),2);
 G = (dp .^3) * khv';
 H = (dp .^4) * khv';
 I = (dp .^5) * khv';
 J = sum(xo*diag(diag((dp .^3)'*khv)),2);
 K = (dp .^6) * khv';

 X = [B C E G; C E G H; E G H I; G H I K];
 IX = inv(X);

 alpha = IX(4,1) * A + IX(4,2) * D + IX(4,3) * F + IX(4,4) * J;
 beta  = IX(3,1) * A + IX(3,2) * D + IX(3,3) * F + IX(3,4) * J;
 gamma = IX(2,1) * A + IX(2,2) * D + IX(2,3) * F + IX(2,4) * J;
 delta = IX(1,1) * A + IX(1,2) * D + IX(1,3) * F + IX(1,4) * J;

 yud = alpha*(d-prs+1)^3 + beta*(d-prs+1)^2 + ...
       gamma*(d-prs+1) + delta;

end
```

3.3 Real Data Examples

We consider real acoustic well-logging data (Yamamoto *et al.* [18]) in this section. Figure 3.3(a) shows single-receiver waveforms or a common-offset gather from a transmitter-receiver (T-R) spacing of 45 ft (13.72 m). There are 263 traces in a certain horizontal well, appearing at every 0.5 ft (0.15 m). In each waveform, the initial 200 time samples are extracted from the entire 512 time samples with a sampling rate of 20 μs. In this time region, compressional head waves arrive at about the 70^{th} time step. Between the 140^{th} and 145^{th} time step, reflected waves from a certain reflector appear. Shear head waves are trimmed away from this gather. We observe a lot of crisscross-like events that blur the image, and these events eventually degrade the imaging quality after migration. For this data, we apply the polynomial model and get optimal parameters by using the local likelihood method. Then we create reconstructed images using optimal parameters obtained so that only the major waveform components remain in the reconstructed images.

In the case of $h = 6$, a local regression fit to the original data is computed for orders 0, 1, and 2. They are respectively local constant, local linear, and local quadratic fits shown in Figures 3.3(b), 3.3(c), and 3.3(d). The higher the order, the closer the fitted images are to the original. Empirical error values are presented to indicate the degree of fitting. It is confirmed here that fitting with a high-order polynomial leads to an estimate with less bias.

In Figure 3.4, local quadratic fits for the same data shown in Figure 3.3(a) are presented with four different h parameters, which are 2, 10, 20, and 50. They are respectively plotted in Figures 3.4(a), 3.4(b), 3.4(c), and 3.4(d). Clearly, the fit produced by the smallest $h = 2$ produces a much noisier fit than the largest $h = 50$. It also appears that $h = 50$ has over smoothed, since the events start to appear as straight horizontal lines.

In order to define an optimal h for the model selection, the CV estimates are computed for polynomial orders of 0, 1, and 2. Figures 3.5 and 3.6 present cross validation plots for $\{h|0 \leq h \leq 200\}$ and $\{h|0 \leq h \leq 20\}$ cases, respectively. In Figure 3.5 we can see that the higher the polinomial order becomes, the smaller CV

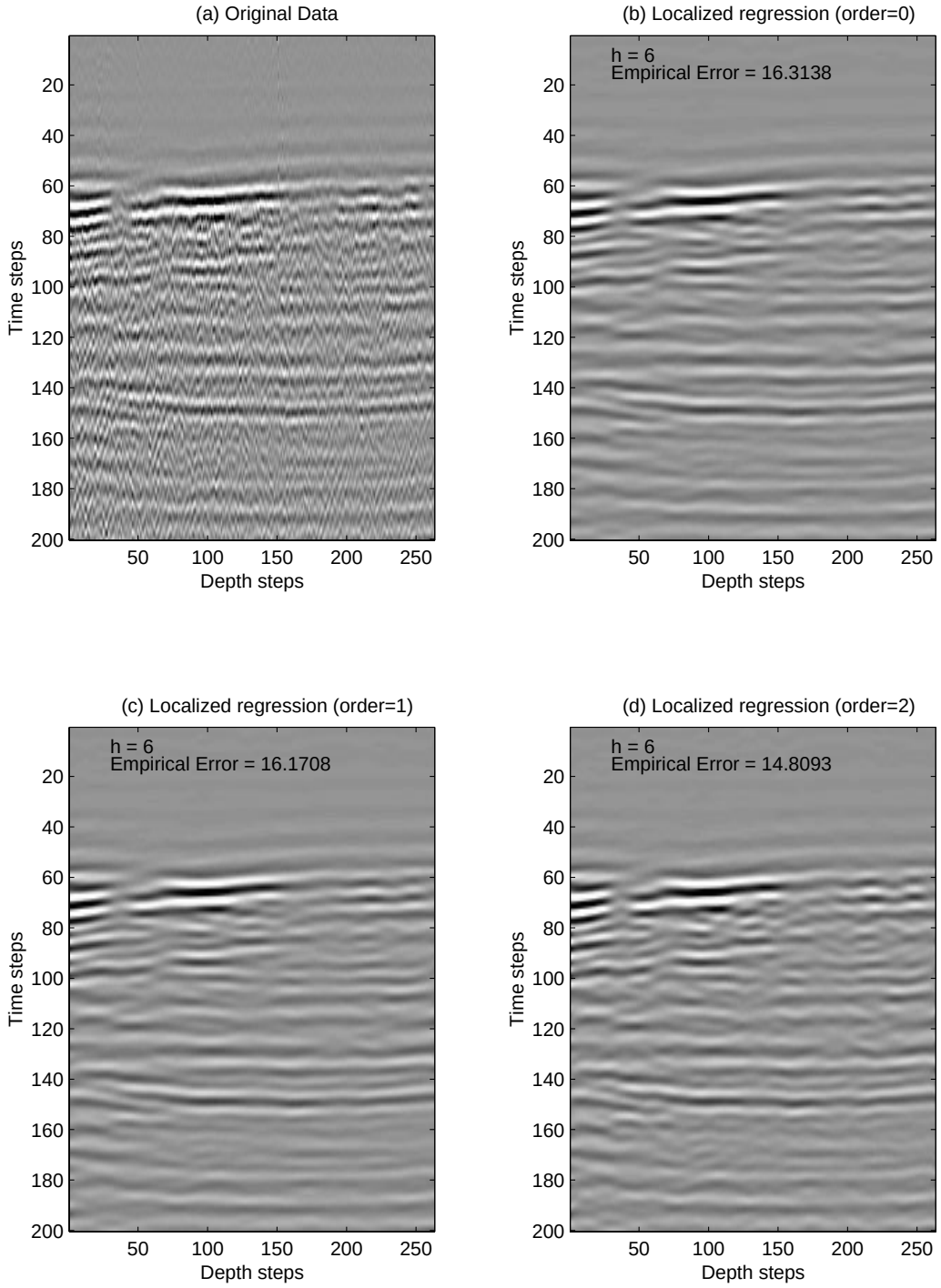

Figure 3.3: Acoustic well-logging data and reconstructed images from the localized regression model with the bandwidth $h = 6$. (a) Original single-receiver waveforms from a T-R spacing of 45 ft (13.72 m). (b) Image reconstruction from local constant fitting. (c) The same as (b) but for local linear fitting. (d) The same as (b) but for local quadratic fitting.

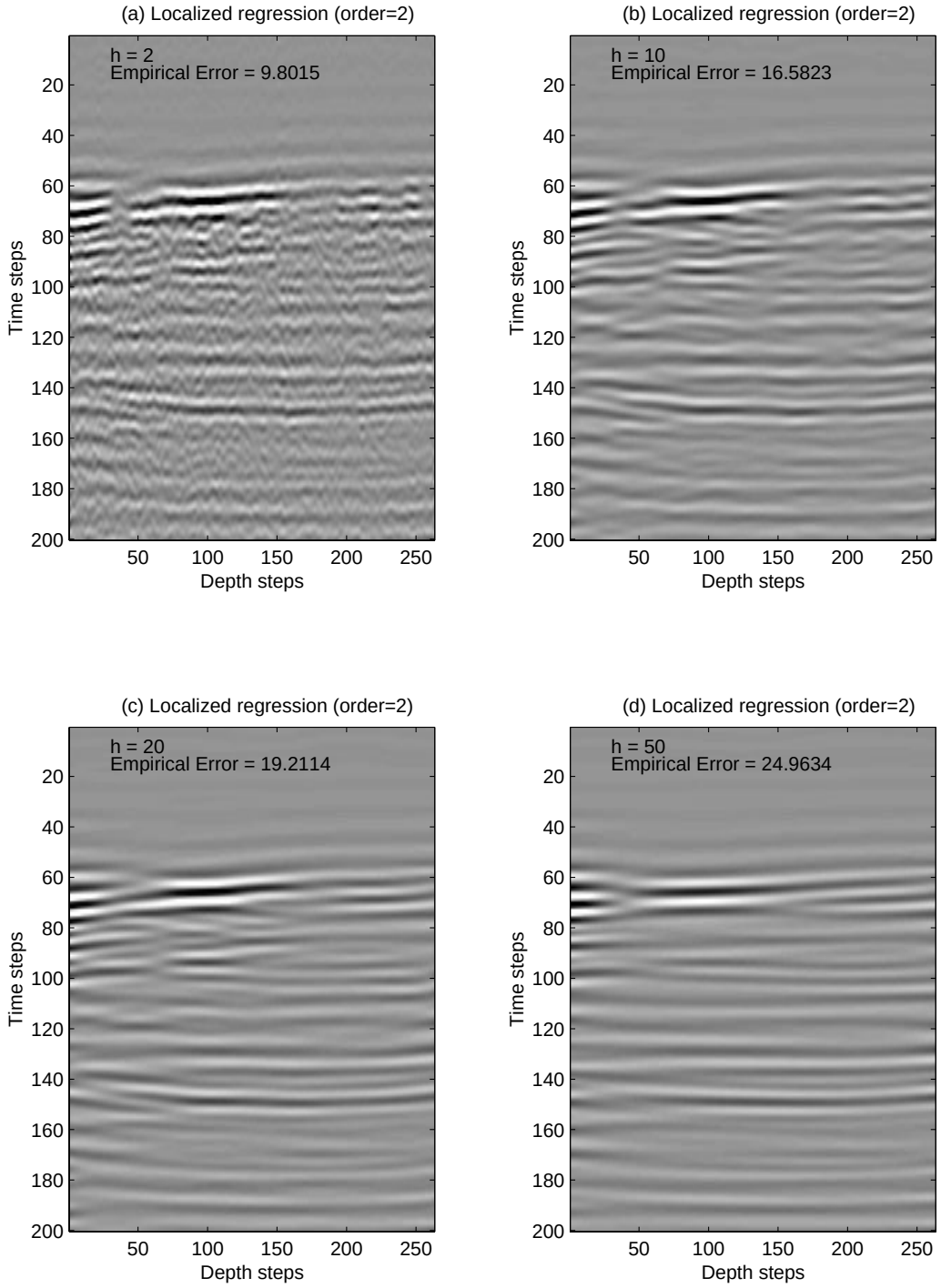

Figure 3.4: Reconstructed images from the localized regression on the acoustic well-logging data shown in Figure 3.3(a). Four different h parameters, (a) $h = 2$, (b) $h = 10$, (c) $h = 20$, and (d) $h = 50$, are used while preserving the fixed polynomial order of 2 (local quadratic fitting).

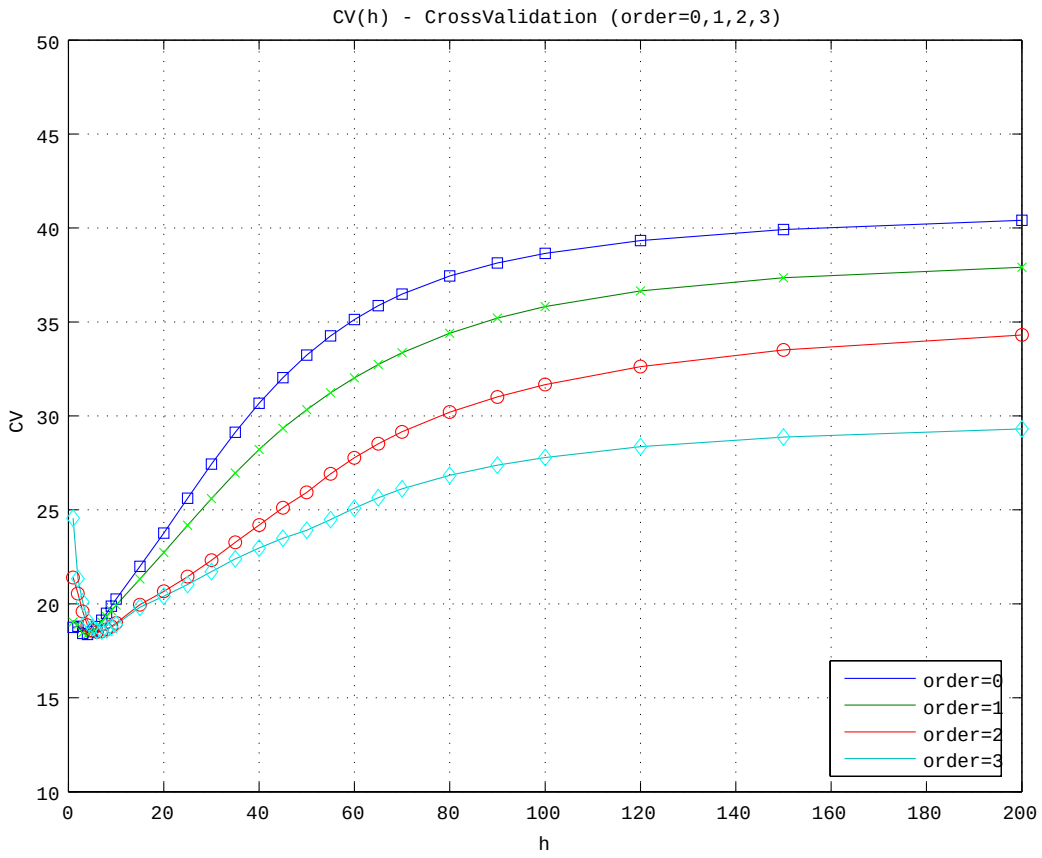

Figure 3.5: Cross validation plots with polynomial orders of 0 (squares), 1 (crosses), 2 (circles), and 3 (diamond) for the acoustic well-logging data shown in Figure 3.3(a).

values are for large h values. From Figure 3.6 we can identify optimal h values as $h = 4$ for orders 0 and 1, $h = 6$ for *order*=2, and $h = 7$ for *order*=3. This gives a practical insight into bandwidth selection when attempting local regression fits. Figure 3.7 presents reconstructed images from the localized regression model for the orders of 0,1,and 3. For the case of *order* $= 2$, see Figure 3.3(d).

In Figure 3.8 influence functions, $\ell_i(d_i, h)$, are plotted for the local constant, local linear, local quadratic, and local cubic fits. Optimal h values for these cases are 4 for both the local constant and linear fits, 6 for the local quadratic fit, and 7 for the local cubic fit. The influence values are generally about 0.1, indicating that $\mathbf{Y}_i$ constitutes about 10% of the fitted value. But at both edges an abrupt increase

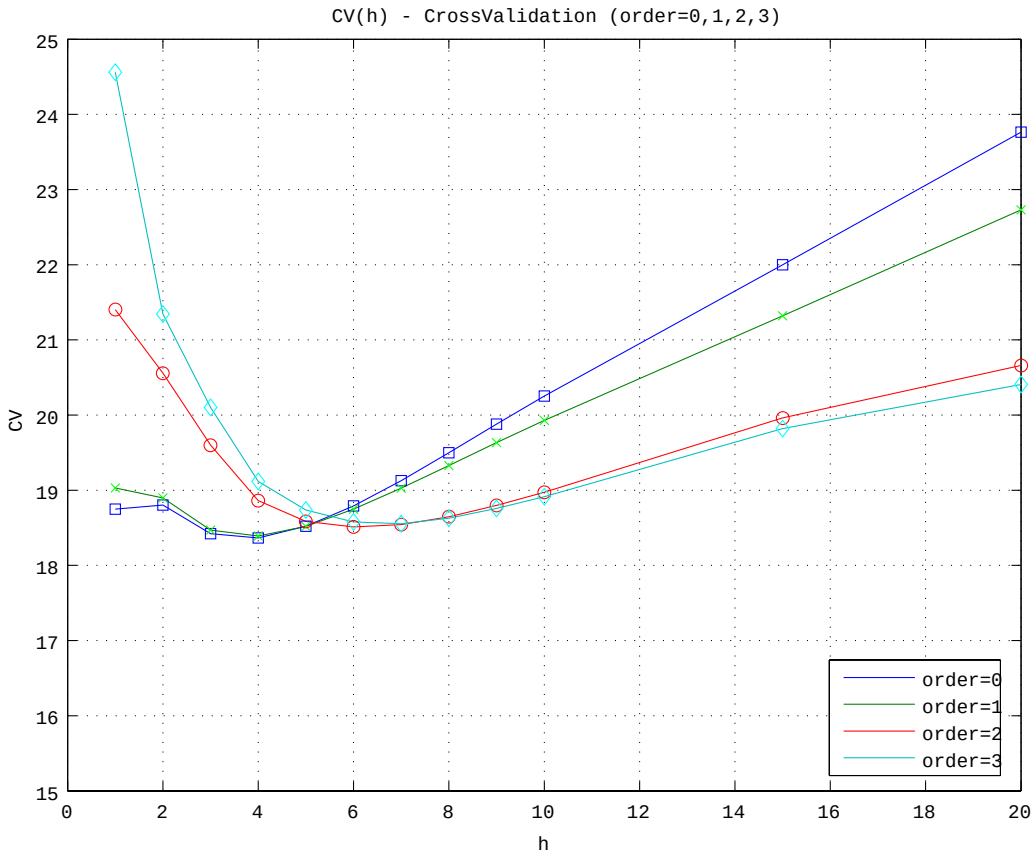

Figure 3.6: Close looks at around attaining local minima of cross validation plots (see Figure 3.5) with polynomial orders of 0 (squares), 1 (crosses), 2 (circles), and 3 (diamond) for the acoustic well-logging data shown in Figure 3.3(a).

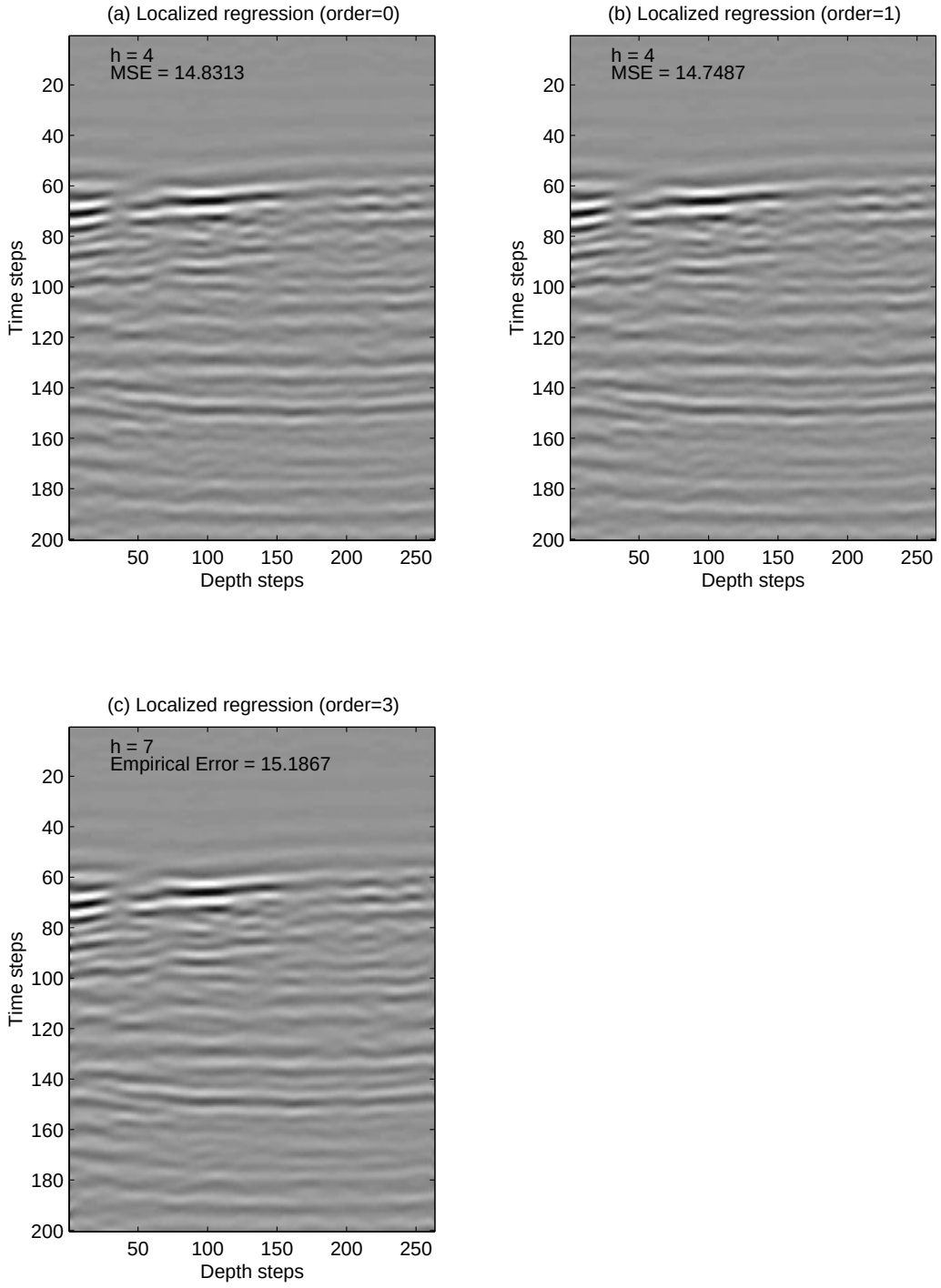

Figure 3.7: Reconstructed images from the localized regression on the acoustic well-logging data shown in Figure 3.3(a). Three different h parameter sets, (a) $order = 0$ and $h = 4$, (b) $order = 1$ and $h = 4$, and (c) $order = 3$ and $h = 7$, are used. These parameter sets are obtained as minima for orders 0, 1, and 3 in Figure 3.6.

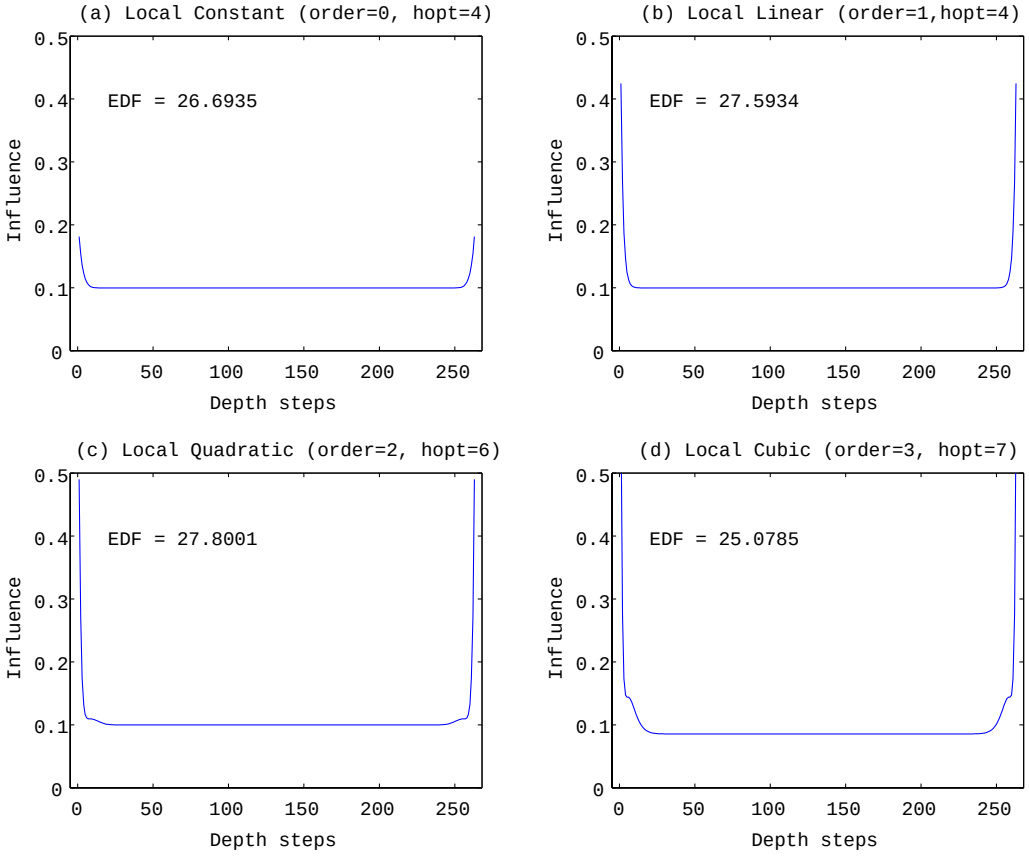

Figure 3.8: Influence functions for (a) local constant, (b) local linear, (c) local quadratic, and (d) local cubic fits to the acoustic well-logging data. Optimal h values are 4 for (a) and (b), 6 for (c), and 7 for (d).

is observed. This indicates the difficulty in polynomial fitting at these edges. The degrees of freedom defined using (3.11) are 26.7, 27.6, 27.8, and 25.1 for orders 0, 1, 2, and 3, respectively. These similar values suggest that any optimal bandwidth found by the cross validation technique is reasonable regardless of the polynomial order.

Multiple bandwidth selection is applied for the same data shown in Figure 3.3(a). We divide the entire time region into 10 segments and for each segment we compute an optimal h value. Then these optimal h values are smoothed by the local quadratic fit. In Figure 3.9(a), smoothed optimal h values are plotted in a solid line, and a dashed line indicates the constant optimal h value used in Figure 3.3(d). We observe two

sections centered at $t = 50$ and $t = 140$, where the corresponding optimal bandwidth is relatively smaller than neighbor time zones. The first part corresponds to low-frequency head waves, and the second to the reflected waves of interest. Thus we successfully give more effective degrees of freedom to these two important sections. In Figure 3.9(b) reconstructed images made from this multiple bandwidth selection are presented.

Based on the optimal order and bandwidth selection, a preliminary evaluation is performed to confirm the effects of local likelihood regression operations on migrated images. Figure 2.26 shows the migrated image from the same data shown in Figure 3.3, but in this case 906 depth traces are considered. In the migrated image, a solid line represents the well trajectory and migrated images are mirrored on either side of this line. It should be noted that both up and down coming signals are recorded with a set of hydrophones in the borehole, so that the mirroring operation can be performed. For this data, local likelihood regression is performed on eight different common-offset gathers. The regression order and the bandwidth used are 2 and 6, respectively. For the eight reconstructed data sections, the migration is conducted to generate the single image presented in Figure 3.10. It is observed that in Figure 2.26 subtle crisscross-like patterns exist due to background noise. These patterns are effectively smoothed on the migrated images in Figure 3.10.

3.4 Discussion and conclusions

In Section 3.2 we propose a new approach to reconstructing 2-d images by introducing a regression model in a localized sense. The model can be represented as a k^{th} order polynomial form onto a 2-d (space, time) domain in acoustic well-logging data. Knowing that no real generic model exists, we propose to introduce a kernel function for localization. This kernel function must be continuous, symmetric, peaked at the center, and supported on a certain depth region. The local likelihood method with use of the kernel leads us to an estimate that is a linear estimator of a waveform at a given depth point, and which takes into account the nearby waveform data. By iterating the local likelihood estimate over the entire depth region, we will get an

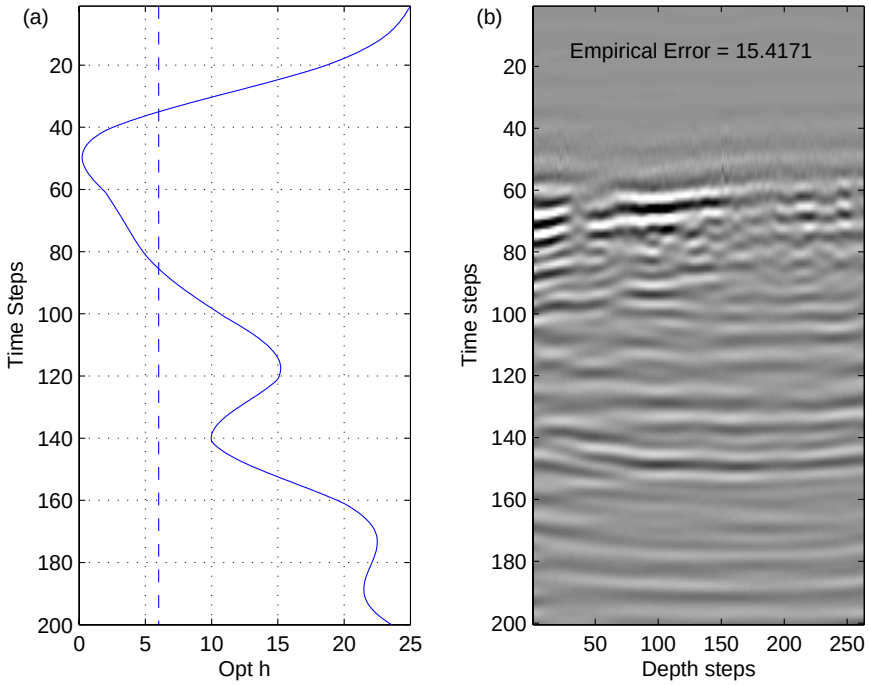

Figure 3.9: (a) Optimal h plot (solid-line) over the time axis for the acoustic well-logging data shown in Figure 3.3(a), and (b) reconstructed images from the localized regression with the independent component model. The local quadratic fit is used. Optimal h values are obtained over 10 contiguous segments along the time axis and then they are interpolated with a factor of 10. A dashed line in (a) indicates the constant optimal value corresponding to Figure 3.3(d).

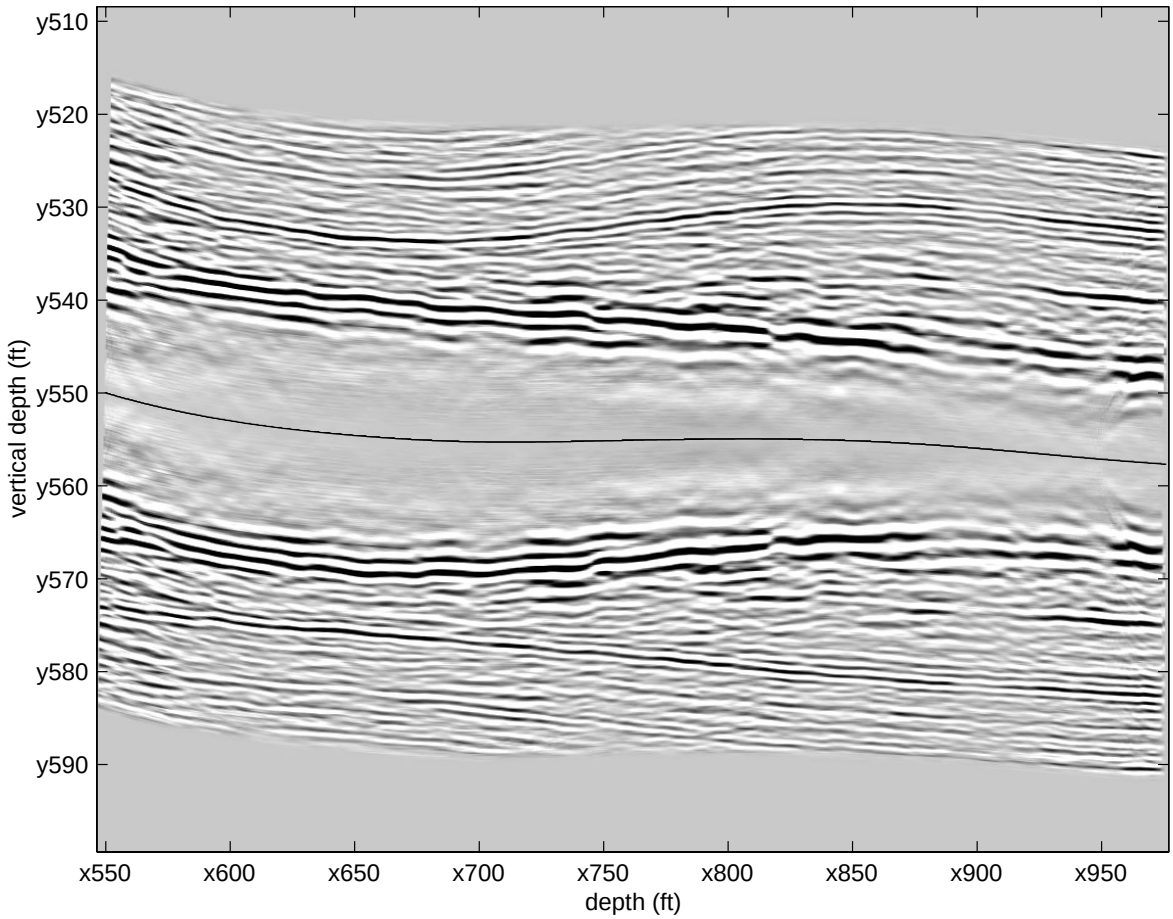

Figure 3.10: Migrated images with localized regression operations. The regression order and the bandwidth are 2 and 6, respectively. A solid line represents the well trajectory, and migrated images are mirrored on either side of this line. In Figure 2.26 migrated images without localized regression operations are presented.

estimated 2-d section on which miscellaneous random features are eliminated.

We focus on the selection of the polynomial order and kernel bandwidth. For the former it is practically sufficient to consider local linear and local quadratic functions (Loader [10]). For the latter, the cross validation method is introduced. This method can be a graphical aid in choosing a parameter that determines the optimal window size for local regression fits. Therefore CV derives a key for the question paused in Section 3.1: What's an optimal window size?

For acoustic well-logging data, static correction is practically unattainable because the acquisition is made in the borehole. Residual static correction may be performed statistically. Under this circumstance it is desired to have a robust method to suppress noise over the entire processing region with a light-computation cost. It is emphasized that the local likelihood regression takes into account the characteristics of sonic data in both time and space domains in conjunction with the appropriate bandwidth selection obtained by the CV criteria.

In summary, we have formulated localized regression fits by introducing the kernel function for a single common-offset gather from acoustic well-logging data. Using this method, we successfully obtain fits by considering two components: the polynomial order and the bandwidth selection. Practically speaking, it is efficient to use a low-order polynomial and to obtain the optimal h value by utilizing the cross validation (CV) plot. Our method demonstrated in this chapter can be applied in other domains where image reconstructions are necessary, mainly to preserve continuous events in a 2-d section and to suppress random noise appearing in that section.

Chapter 4

Local Regression for Kriging

4.1 Introduction

In geophysical domains it is common to make spatial interpolation from sparse datasets. Kriging is widely used for this purpose. Georges Matheron (1930–2000) named this method after D. G. Krige (Riplay [12]; Agterberg [1]). Among other kriging methods, we focus on ordinary kriging and review its methodology by following Wackernagel [16]. Then we apply this method for real porosity data obtained from different 16 wells. For this data set kriging using the Gaussian variogram model is made. Here computational difficulties lie in 1) selecting an appropriate semi-variogram model and 2) obtaining optimal parameters from nonlinear model fitting for the selected model. Our proposal is to make use of a local regression model on a variogram cloud and to apply results from the local regression analysis to a process for identifying semi-variogram features, although examination for this proposal is immature.

4.2 Methodology

Suppose that we estimate a value at $\mathbf{x}_0$ using data values from n neighboring sample points $\mathbf{x}_\alpha$ and linearly associated weights w_α

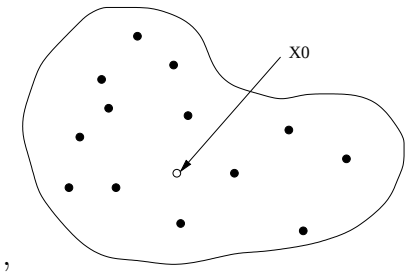

Figure 4.1: A domain containing sparsely sampled data and a point of interest.

$$\hat{Z}(\mathbf{x}_0) = \sum_{\alpha=1}^{n} w_\alpha Z(\mathbf{x}_\alpha). \tag{4.1}$$

Figure 4.1 schematically shows a domain containing sparsely sampled data and a point of interest. Here we assume that sparsely sampled data are characterized by intrinsic random function $Z(\mathbf{x})$ with a semi-variogram $\gamma(h)$ where h is the distance between two samples. The relationship between the semi-variogram and the covariance function is

$$\gamma(h) = C(0) - C(h). \tag{4.2}$$

The weighting coefficients w_α can be determined based on the minimum estimation variance criterion:

$$\begin{aligned} \sigma_E^2 &= E[(Z(\mathbf{x}_0) - \hat{Z}(\mathbf{x}_0))^2] \\ &= -\gamma(\mathbf{x}_0 - \mathbf{x}_0) - \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n} w_\alpha w_\beta \gamma(\mathbf{x}_\alpha - \mathbf{x}_\beta) \\ &\quad + 2\sum_{\alpha=1}^{n} w_\alpha \gamma(\mathbf{x}_\alpha - \mathbf{x}_0) \end{aligned} \tag{4.3}$$

subject to the unbiasedness condition

$$\sum_{\alpha=1}^{n} w_\alpha = 1. \tag{4.4}$$

Weights w_α can be obtained by minimizing (4.3) with constraint of (4.4). Having performed the Lagrange method the estimation variance of ordinary kriging is

$$\sigma_0^2 = \mu - \gamma(\mathbf{x}_0 - \mathbf{x}_0) + \sum_{\alpha=1}^{n} w_\alpha^* \gamma(\mathbf{x}_\alpha - \mathbf{x}_0) \tag{4.5}$$

where μ is the Lagrange parameter and w_α^* are obtained weighting coefficients.

For each node in a regular grid, an estimated value is computed from neighbor sample data together with their associated weights w_α^*.

4.2.1 Matlab Codes

In this section implemented Matlab codes are presented. They are summarized in Table 4.1. For *regKrig* we studied *davis* function found in Gratton and Lafleur [5].

Name	Descriptions
regLocal1	computes a local regression estimate.
regKrig	performs kriging based on empirical semi-variogram.

Table 4.1: Matlab functions to perform local regression and kriging for spatial data.

■ *regLocal1*

```
function yu = regLocal1( sv, x, h, order, sill_flag )

% function yu = regLocal1( sv, x, h, order, sill_flag )
% computes a local regression estimate for
% a predictor variable x.
%
% Input:
%        sv - m pairs of observations in
%             a matrix whose dimension is m-by-2.
%             The 1st/2nd column is for 'x'/'y' component.
%        x  - a point of interest where a local regression
%             estimate is computed. For multiple x points
%             a vecotr form can be specified.
%        h  - a positive bandwidth to make
%             a smoothing window (x-h,x+h)
%        order -  a regression polinomial order
%             0: Local Constant
```

```
%             1: Local Linear
%        sill_flag - 0: use local regression estimate
%                       at everywhere (default)
%                    1: set the maximum semi-variogram value
%                       for sill part.
% Output:
%        yu - a local regresiion estimate at a given
%             predictor point x. In case that x is a vector,
%             corresponding estimate values are returned
%             as a vector.
%
% Remark:
%    As a weight function the tricube weight function
%    that is defined as w(u)= (1-|u|^3)^3.
%
% S. watanabe

if nargin == 4
   sill_flag = 0;
end

[m,k] = size(sv);
mx    = max(size(x));
x     = reshape(x,mx,1);
w = [];
for i=1:mx
        lm = abs(sv(:,1) - x(i));
        for j=1:m
            if lm(j) > abs(h)
                lm(j) = 1.0;
            else
                lm(j) = lm(j)/abs(h);
            end
        end
        w = [w (ones(m,1) - lm.^3).^3];
end

if order == 0
  yu = w'*sv(:,2)./ sum(w)';
elseif order == 1
  xw = w'*sv(:,1)./ sum(w)';
  xz1 = zeros(mx,1);
  xz2 = zeros(mx,1);
  for i=1:mx
    xz1(i) = w(:,i)'* diag((sv(:,1) - xw(i)) * sv(:,2)');
    xz2(i) = w(:,i)'* (sv(:,1) - xw(i)).^2;
  end
  yu = w'*sv(:,2)./ sum(w)' + (x - xw) .* (xz1 ./ xz2);
end

if sill_flag == 1
   sweep_x = linspace(sv(1,1),sv(m,1));
   yyu = regLocal1( sv, sweep_x, h, order, 0);
   sill_value = max(yyu);
```

```
   sill_start = min(find(yu > 0.99*sill_value));
   yu(sill_start+1:max(size(yu))) = sill_value;
end
```

■ *regKrig*

```
function [Zp,Sp] = regKrig(sv, data, x0, h, order)

%  [Zp,Sp] = regKrig(sv, data, x0, h, order)
%
% performs kriging based on empirical semi-variogram
% generated from local regression fit.
%
% Input:
%        sv   - empirical semi-variogram values  m-by-2
%        data - orignal samples matrix [x y z]   n-by-3
%        x0   - a lis of points at which estimates are computed
%        h    - a positive bandwidth to make
%               a smoothing window (x-h,x+h)
%        order -  a regression polinomial order
%               0: Local Constant
%               1: Local Linear
%
% Output:
%        Zp   - kriged estimate matrix
%        Sp   - variance matrix
%
% Remark:
%         reglocal1() is called inside as sill_flag = 1
%
% Reference:
%
%         This program was modified from davis.m in:
%         MATLAB Kriging Toolbox, Version 4.0, July 2001
%         Yves Gratton and Caroline Lafleur
%

n = length(data);
x = data(:,1);
y = data(:,2);
z = data(:,3);
sill_flag = 1;

% Evaluation of matrix A

for i = 1:n
   for j = 1:n
        dx = x(i) - x(j);
        dy = y(i) - y(j);
```

```
        r  = sqrt (dx*dx + dy*dy);

        A(j,i) = regLocal1( sv, r, h, order, sill_flag );
   end
end

A(n+1,:) = ones(1,n);
A(:,n+1) = [ones(n,1); 0];


% Evaluation of vectors Y and then X

np = length(x0)

for k = 1:np
  disp(['Doing ' int2str(k) ' out of ' int2str(np)])

  % interpolation position
  xp = x0(k,1);
  yp = x0(k,2);

  for i = 1:n
     % vector Y
     dxp = x(i) - xp;
     dyp = y(i) - yp;
     rp  = sqrt ( dxp*dxp + dyp*dyp);

     Y(i,1) = regLocal1( sv, rp, h, order, sill_flag );
  end

  Y(n+1,1) = 1;

  % vector X
  X = inv(A) * Y;

  % weight vector: vector X with the last line missing
  W = X(1:length(X)-1,1);

  % interpolation at p
  Zp(k,1) = W' * z;

  % variance estimation
  Sp(k,1) = X' * Y;

end
```

4.3 Real Data Examples

We consider a scalar measurement made at spatially different locations. Table 4.2 lists porosity measurements over different 16 wells that have UTM (universal transverse

mercator) coordinates in meters (Russel [13]). Porosity indicates the percentage of pore volume or void space, or that volume within rock which can contain fluids (see Figure 4.2).

Well ID	X (m)	Y (m)	Porosity (%)
1	2280	890	4.0
2	1240	1210	1.5
3	1651	1290	5.7
4	2169	1230	2.9
5	2059	1690	10.4
6	1722	1630	16.1
7	891	1820	1.9
8	1385	2060	7.7
9	1682	2020	15.2
10	1885	2050	7.6
11	1991	2310	11.9
12	1694	2420	15.8
13	1023	2310	6.1
14	1305	2750	4.5
15	1705	2620	12.7
16	2301	2000	12.4

Table 4.2: Porosity samples from 16 wells.

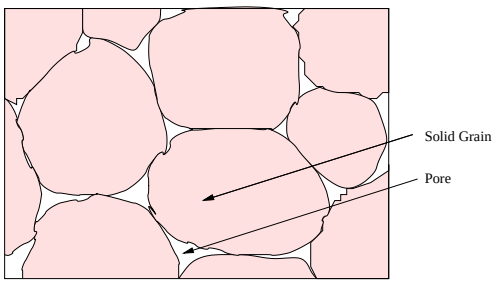

Figure 4.2: A schematic diagram to represent pore that is a discrete void within a rock. Pore space can contain air, water, hydrocarbons or other fluids.

In Figure 4.3 a map of the porosity samples taken from 16 wells is plotted in a 1600 m $\times$ 2000 m region. From these data a semi-variogram cloud is generated from every combination of pairs and is presented in Figure 4.4. We see that the semi-variogram cloud is sparsely distributed in the 2-d space and it is hard to identify relations between two variables, the spatial distance h and the semi-variogram γ.

In order to find relations between them, bins are introduced over the distance axis. Each bin has a fixed width over the distance axis and semi-variogram values fall into this bin are averaged to yield a representative value in this bin. For this porosity

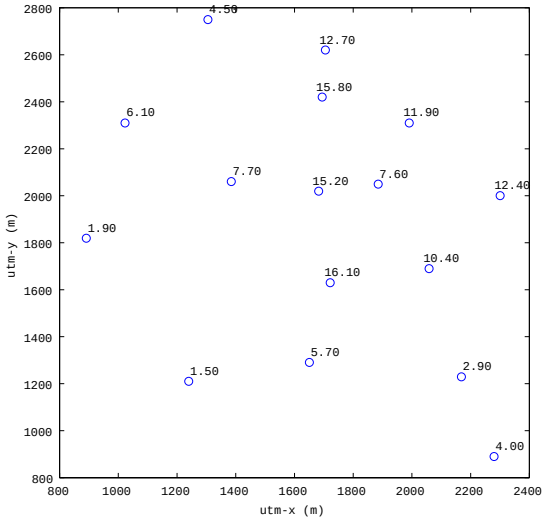

Figure 4.3: Spatially distributed porosity samples from 16 wells.

data we introduce 13 bins and in each bin an average value is computed. Then the Gaussian variogram model is selected and a least-squares fitting is made as depicted in Figure 4.5. These averaged values are plotted with circles and the dotted line represents LSQ fit from these points. When we eliminate one circle located at around the relative distance of 0.96, the solid line is obtained from twelve pluses. This Gaussian variogrammodel can be formulated as:

$$\gamma(h) = C_0(1 - \exp\{-(h/L)^2\}) + Y_0 \tag{4.6}$$

where h is the distance, C_0 is an offset between the sill and the nugget-effect, L is the length scale, and Y_0 is the nugget-effect. In Figure 4.6 this model is schematically presented.

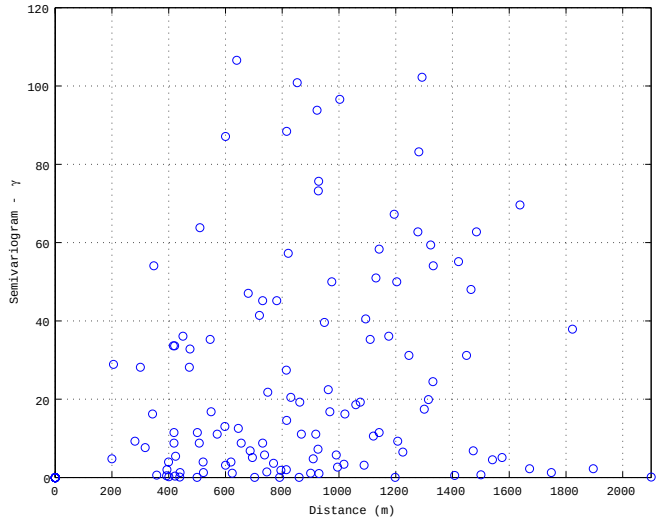

Figure 4.4: Empirical semi-variogram plot or a semi-variogram cloud from a combination of 136 different pairs.

4.4 Observations

As we see from Figures 4.7 and 4.8 local regression operations work well in each case. In order to form a stable sill shape, local regression results are compensated to have flat features starting at a distance position where local regression curves attain the maximum value. From our experiments following points are found:

- As the bandwidth gets wider estimated values have smoothed shape.
- Local linear yields less nugget-effect.

Figures 4.9 and 4.10 present kriged images for two bin cases (13 and 23) with wider bandwidth setting (h=0.3 in relative distance or h=630.02 m). These kriging results suggest that we must study further on this subject to propose efficient means of defining semi-variogram curves with using the local approaches.

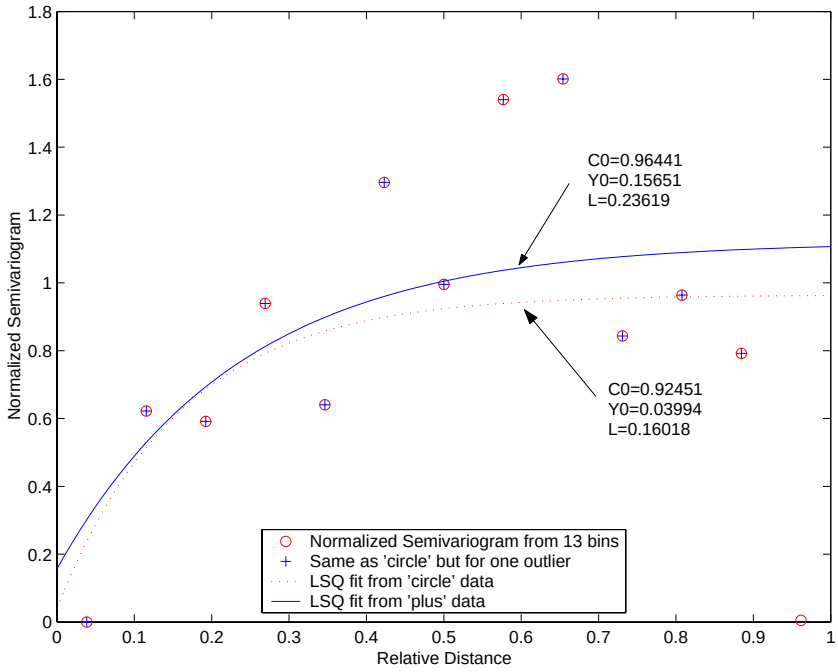

Figure 4.5: Semi-variogram plot and model fitting with the Gaussian model for 13 bins. The solid line represents a least-squares fit from pluses. The dotted curve is made from circles.

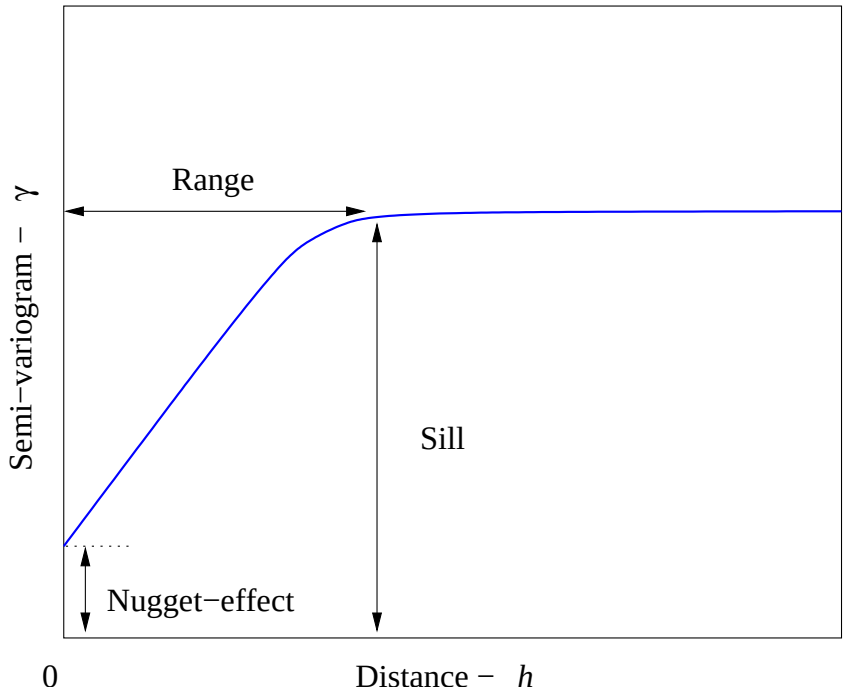

Figure 4.6: Typical semi-variogram plot. Sill is a constant semi-variogram value for a flat shape at or beyond a certain depth point. The nugget-effect appears, if semi-variogram has a positive value at h=0.

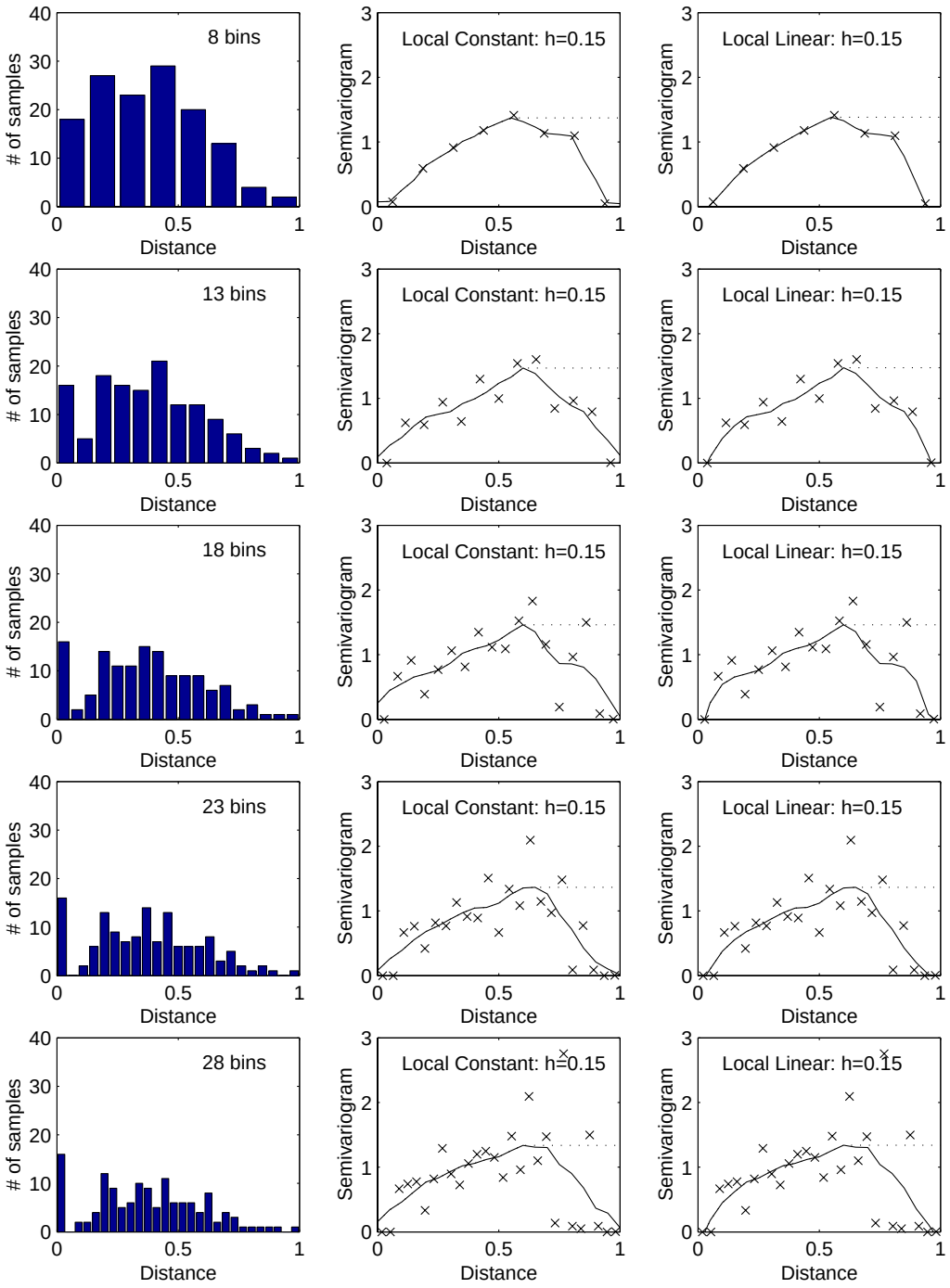

Figure 4.7: Histograms of bin samples (left) and local regression fit of the porosity data. Five different bin sets (n = 8, 13, 18, 23, and 28) are considered with both local constant (middle) and linear (right) cases. Crosses are empirical semi-variogram points computed from each bin. Solid lines show computational results and dashed lines are proposed sill shape. The bandwidth is selected as $h = 0.15$.

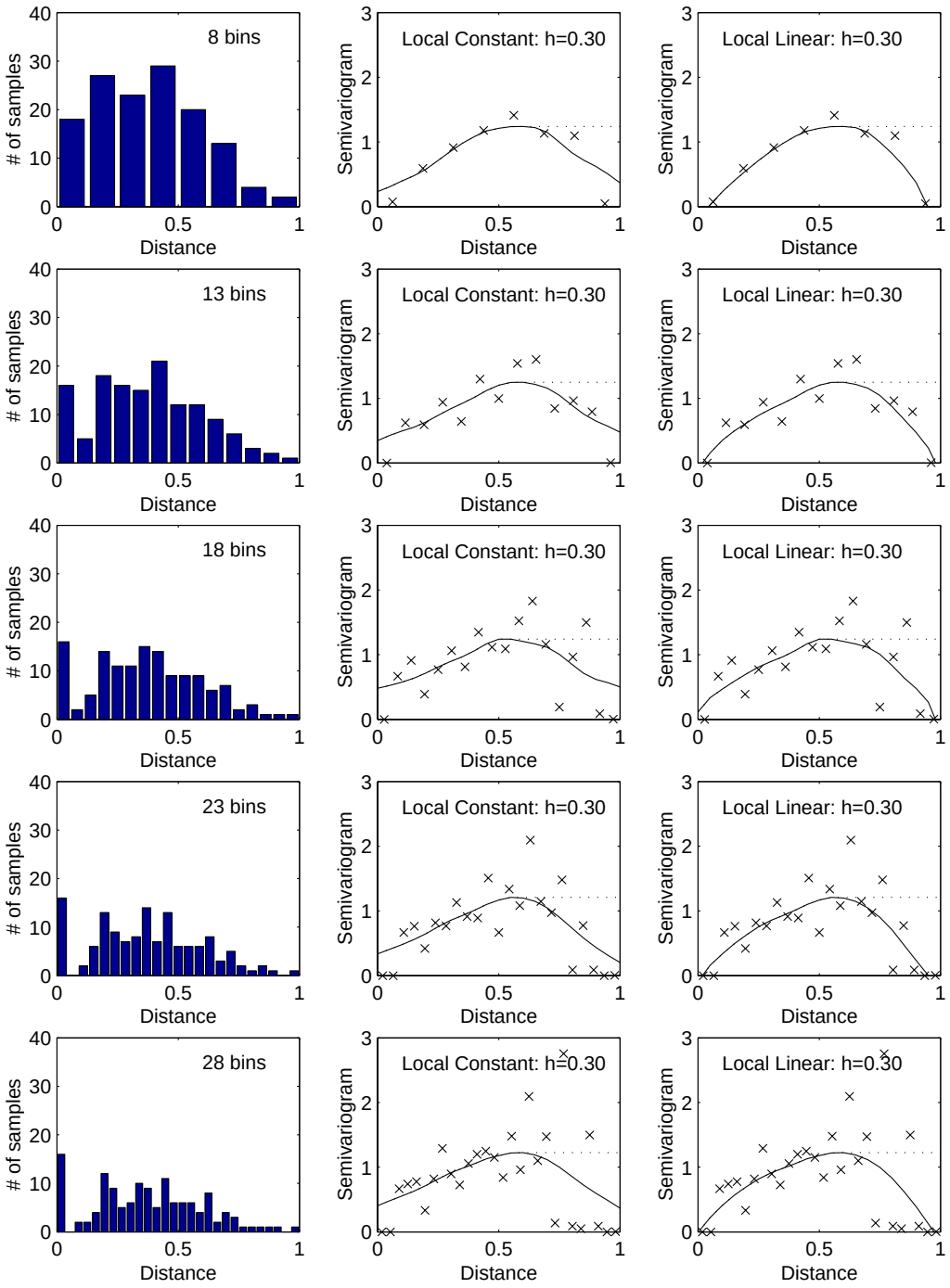

Figure 4.8: Histograms of bin samples (left) and local regression fit of the porosity data. Five different bin sets (n = 8, 13, 18, 23, and 28) are considered with both local constant (middle) and linear (right) cases. Crosses are empirical semi-variogram points computed from each bin. Solid lines show computational results and dashed lines are proposed sill shape. The bandwidth is selected as $h = 0.30$.

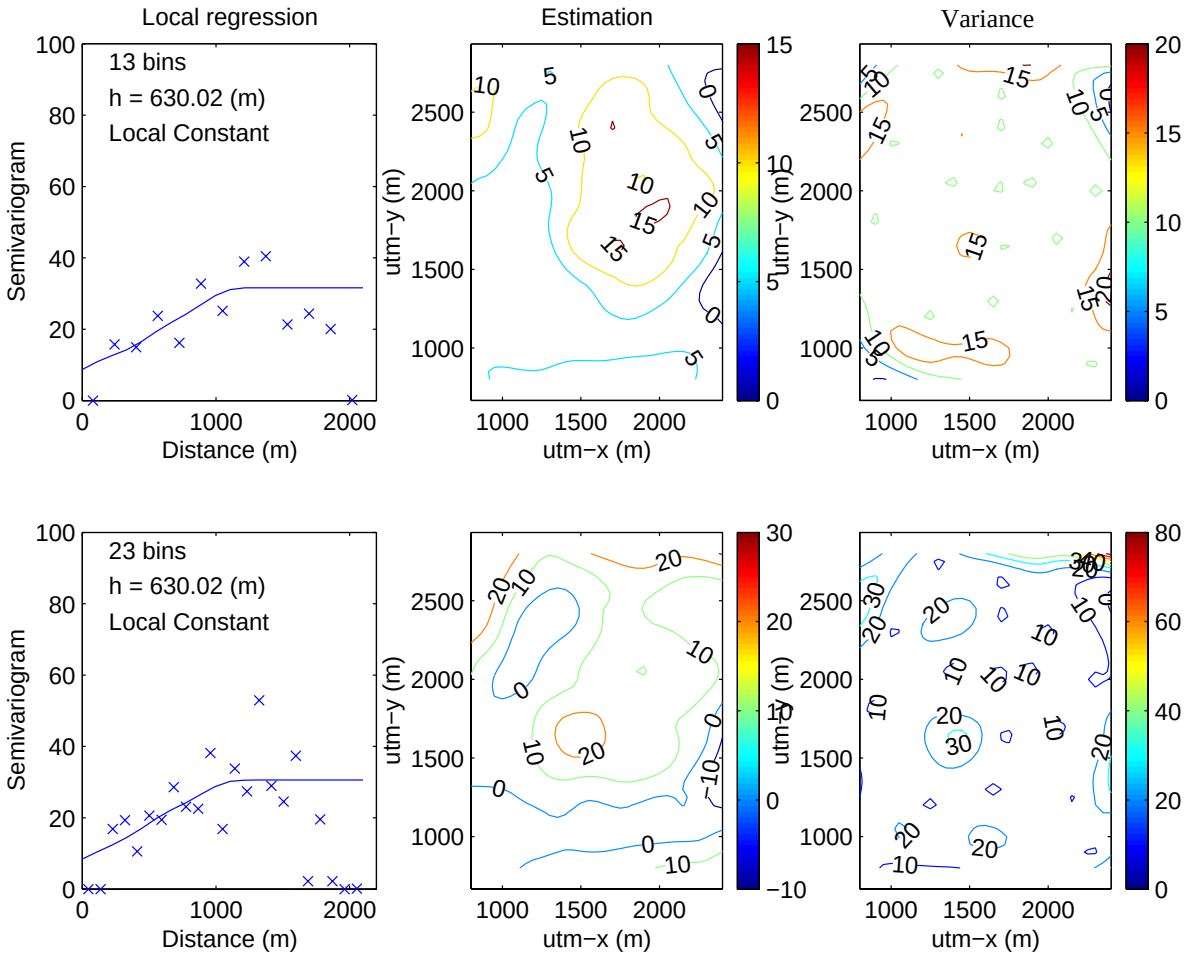

Figure 4.9: Kriging 16 porosity data with local constant fit from two bin cases: (a) 13 and (b) 23 bins. The bandwidth selected is corresponding to 0.3 in Figure 4.8. Middle plots show estimations and right plots indicate variances.

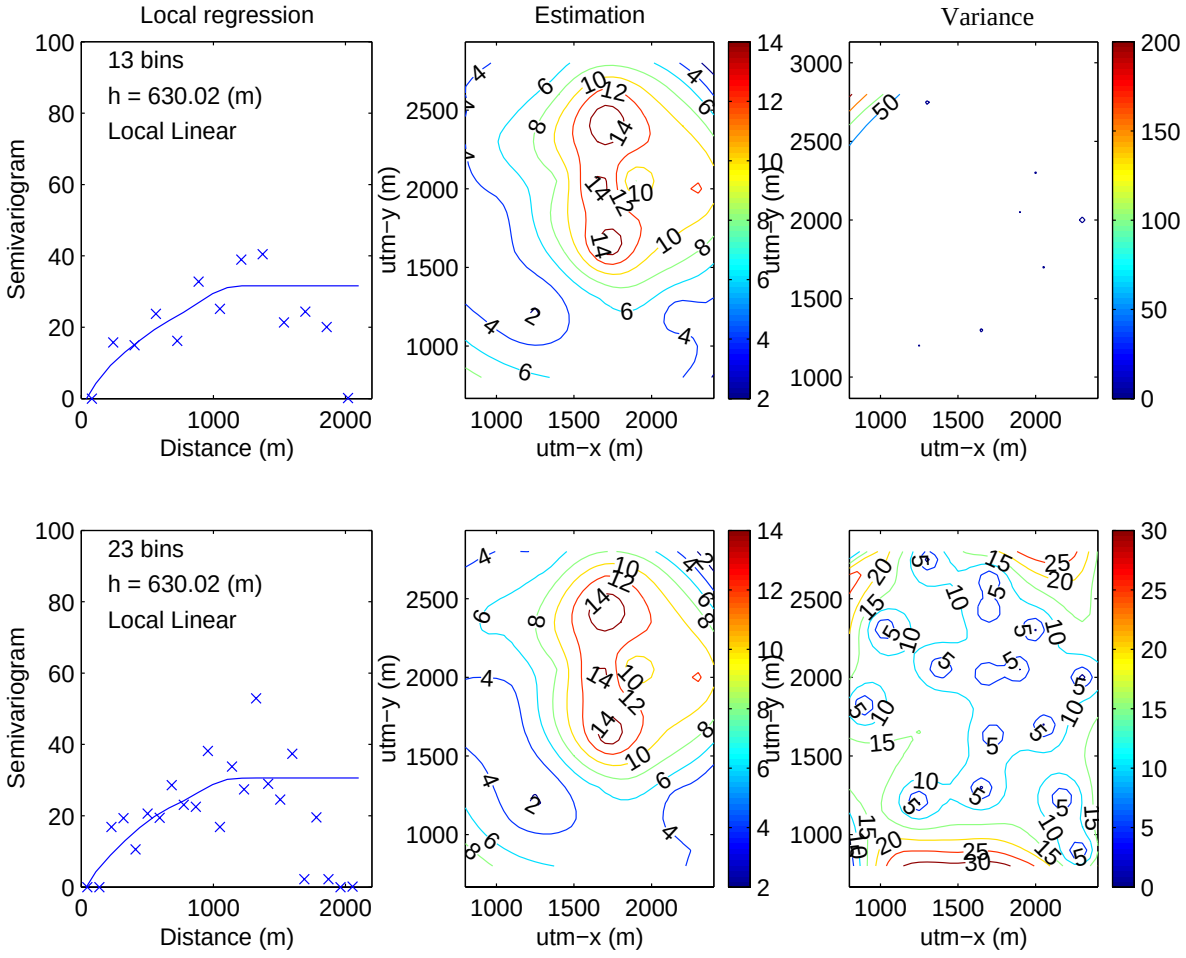

Figure 4.10: Kriging 16 porosity data with local linear fit from two bin cases: (a) 13 and (b) 23 bins. The bandwidth selected is corresponding to 0.3 in Figure 4.8. Middle plots show estimations and right plots indicate variances.

Chapter 5

Concluding Remarks

5.1 Contributions

This thesis investigates a few geophysical data domains on which we can make use of local approaches for selective statistical techniques.

In Chapter 2 we studied PCA and local PCA on the single-well acoustic data. With local PCA operations on the migrated images, crisscross like patterns or noises are effectively removed.

In Chapter 3 we use local regression approaches for the 2-d data domain and formulate localized regression fits by introducing the kernel function. we successfully obtain fits by considering two key parameters that are the polynomial order and bandwidth selection. For obtaining the optimal h value, cross validation (CV) plot is utilized. Lastly we proposed this CV use for the time axis so that waveform locality over the time axis can be taken into account adaptively.

Single-well acoustic imaging is still emerging technology. On this technical field what we have contributed are re-phrased as follows:

- Local analysis of PCA is found to be effective to remove crisscross like patterns on migrated images from single-well acoustic imaging.
- Local regression approach is introduced on the single-well acoustic data mainly for denoising.

- The vector-valued response variable is considered for the scalar predictor in the local regression approach.
- Visualization of local regression results is made.
- The bandwidth selection is made along the depth axis.
- The cross validation is used for selecting an optimal bandwidth.
- Multiple bandwidth selection is proposed by additionally taking account for the time axis.

Chapter 4 concentrates on the semi-variogram cloud from the porosity data with local regression approaches. This cloud indicates the dissimilarities against the spatial separation of sample pairs and is important when determining a sequence of average dissimilarities over the lateral direction. Normally this sequence is fitted with a theoretical curve, however this process may be practically difficult because a wrong model selection falls away from key features embraced by the original data. This local regression approach facilitates easy computation for forming an appropriate sequence of average dissimilarities with no worry about selecting an optimal model from theoretical models such as exponential, Gaussian, spherical and so on. In this local regression approach we consider simple cases that are local constant and local regression fits. For each fit our local regression model has a single predictor variable and a single response variable. This local regression use may be practically useful but this is not proved yet.

5.2 Suggestions for further research

Further studies are desired to examine capabilities for extracting reflector signals from single-well acoustic imaging data by means of localized regression approaches.

In Chapter 4 local regression is utilized to form semi-variogram models. Then generated models are referenced when making kriging operations. Finally kriged values over lattice points are fed into *contour* function to make contour plots. The

mathematical spline function can also be used in spatial interpolation from sparse data. Wackernagel [16] briefly introduces the equivalence between splines and kriging in Chapter 39. Further study is required to identify the equivalence or to find dissimilarities when local regression is introduced in kriging.

Appendix A

Derivation of Equation 3.9

In this appendix we derive the cross validation formula of (3.9).

We prepare a simple formula that

$$(\mathbf{A} - \mathbf{a}\mathbf{a}^T)^{-1}\mathbf{a} = \frac{1}{1 - \mathbf{a}^T\mathbf{A}^{-1}\mathbf{a}}\mathbf{A}^{-1}\mathbf{a} \tag{A.1}$$

for a nonsingular matrix $\mathbf{A}$ and vector $\mathbf{a}$. See, for example, 2.3 in Rao [11]. For simplicity, let $\mathbf{x}(d_i)$ be $\mathbf{x}_i$. Then it follows from (A.1) that

$$\left(\sum_{j\neq i} K(d, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i = \frac{\left(\sum_j K(d, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i}{1 - K(d, d_i; h)\mathbf{x}_i{}^T\left(\sum_j K(d, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i}. \tag{A.2}$$

Hence we can write that

$$\begin{aligned}
\hat{\mathbf{m}}^{(-i)}(d_i) &= \left\{\sum_{k\neq i} K(d_i, d_k; h)\mathbf{y}_k\mathbf{x}_k^T\right\}\frac{\left(\sum_j K(d_i, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i}{1 - \mathbf{x}_i{}^T\left(\sum_j K(d_i, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i} \\
&= \left\{\sum_{k} K(d_i, d_k; h)\mathbf{y}_k\mathbf{x}_k^T - \mathbf{y}_i\mathbf{x}_i^T\right\}\frac{\left(\sum_j K(d_i, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i}{1 - \mathbf{x}_i{}^T\left(\sum_j K(d_i, d_j; h)\mathbf{x}_j\mathbf{x}_j^T\right)^{-1}\mathbf{x}_i} \\
&= \frac{\hat{\mathbf{m}}(d_i) - \ell_i(d_i, h)\mathbf{y}_i}{1 - \ell_i(d_i, h)},
\end{aligned} \tag{A.3}$$

where $\ell_i(d, h)$ is defined using (3.7). Therefore we get

$$\mathrm{CV}(h) = \frac{1}{I}\sum_{i=1}^{I} \|\mathbf{y}_i - \hat{\mathbf{m}}_h^{(-i)}(d_i)\|^2 = \frac{1}{I}\sum_{i=1}^{I} \frac{\|\mathbf{y}_i - \hat{\mathbf{m}}_h(d_i)\|^2}{\{1 - \ell_i(d_i, h)\}^2}, \tag{A.4}$$

which completes the exercise.

Bibliography

[1] F.P. Agterberg. Georges Matheron - Founder of Spatial Statistics. In *Proc., International Association for Mathematical Geology (IAMG)*, Portsmouth, UK, 2003.

[2] C. Chang, D. Hoyle, S. Watanabe, R. Coates, M. Kane, K. Dodds, C. Esmersoy, and J. Foreman. Localized maps of the subsurface. Oilfield review, Schlumberger, Spring 1998.

[3] C. Chatterjee, V. P. Roychowdhury, and E.K.P. Chong. On relative convergence properties of principal component analysis algorithms. *IEEE Transactions on Neural Networks*, **9**:319–329, 1998.

[4] C. Esmersoy, C. Chang, B. Tichelaar, and E. Quint. Acoustic imaging of reservoir structure from a horizontal well. *The Leading Edge*, pages 940–946, 1998.

[5] Y. Gratton and C. Lafleur. *MATLAB Kriging Toolbox, Version 4.0*, 2001.

[6] S. Haykin. *NEURAL NETWORKS, Second Edition.* Prentice Hall, 1999.

[7] B. E. Hornby. Imaging of near-borehole structure using full-waveform sonic data. *Geophysics*, **54**:747–757, 1989.

[8] K. Hsu. Wave separation and feature extraction of acoustic well-logging waveforms using karhunen-loève transformation. *Geophysics*, **55**:176–184, 1990.

[9] S.Y. Kung and K.I. Diamantaras. A neural network learning algorithm for adaptive principal component extraction (APEX). In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.2,pp.861-864,Albuquerque, 1990.

[10] C. Loader. *Local Regression and Likelihood*. Springer, New York, 1999.

[11] C. R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, New York, 1981.

[12] B.D. Ripley. *Spatial Statistics*. Wiley, New York, 1981.

[13] B. Russell. Seismic techniques for prediction of petrophysical parameters: Avo, inversion, and multiple attributes. Lecture Notes, JNOC-TRC, Tokyo, 2000.

[14] X. Tang and A. Cheng. *Quantitative Borehole Acoustic Methods*. Elsevier, Oxford, 2004.

[15] R. Tibshirani and T. Hastie. Local likelihood estimation. *J. Amer. Statist. Assoc.*, **82**:559– 567, 1987.

[16] H. Wackernagel. *Multivariate Geostatistics: An Introduction With Applications*. Springer Verlag, 2003.

[17] S. Watanabe, K. Fujii, and H. Mikada. Reflector imaging using borehole acoustic reflection survey. In *Proc., the 4th Well Logging Symposium of Japan, SPWLA-J*, Paper Q, 1998.

[18] H. Yamamoto, S. Watanabe, J. M. V. Koelman, J. Geel, A. Brie, K. Fujii, and R. Coates. Borehole acoustic reflection survey experiments in horizontal wells for accurate well positioning. In *Proc., the 6th Well Logging Symposium of Japan, SPWLA-J*, 2000.

[19] O. Yilmaz. *Seismic data analysis: processing, inversion, and interpretation of seismic data*. SEG, 2001.

[20] R. Zhang and T. J. Ulrych. Physical wavelet frame denoising. *Geophysics*, **68**:225–231, 2003.

Index