

Data Fitting with Nonstationary Statistics

Jon Claerbout, Antoine Guitton, Stewart A. Levin and Kaiwen Wang

Stanford University

© November 28, 2019

Contents

0.1	PREFACE	i
0.2	INTRODUCTION	ii
0.2.1	What can you do with these methods?	ii
0.2.2	How does it work?	iii
0.3	PREDICTION ERROR FILTER = PEF	iii
0.3.1	PEF history	iii
0.4	CREDITS AND THANKS	iv
1	Nonstationary scalar signals	1
1.0.1	Mathematical setting	1
1.0.2	Spectral shaping the residual	2
1.0.3	Prediction-error filtering (deconvolution)	2
1.0.4	Code for prediction error = deconvolution = autoregression	3
1.0.5	The heart of nonstationary PEF with no calculus	4
1.0.6	Whiteness	4
1.0.7	Scaling components of gradients	5
1.0.8	Fluctuations	5
1.1	PREDICTION ERROR FILTER = PEF	5
1.1.1	The outside world—real estate	5
1.2	FINDING TOGETHER MISSING DATA AND ITS PEF	6
1.2.1	Further progress will require some fun play	7
1.3	CHOOSING THE STEP SIZE	8
1.3.1	Epsilon	8
1.4	NON-GAUSSIAN STATISTICS	9
1.4.1	The hyperbolic penalty function	9

1.4.2	How can the nonstationary PEF operator be linear?	10
1.5	DIVERSE APPLICATIONS	10
1.5.1	Weighting	10
1.5.2	Change in variables	11
1.5.3	Wild and crazy squeezing functions	11
1.5.4	Deconvolution of sensible data mixed with giant spikes	11
1.5.5	My favorite wavelet for modelers	11
1.5.6	Bubble removal	12
2	PEFs in time and space	15
2.0.7	2-D PEFs as plane wave destructors and plane wave builders	16
2.0.8	Two-dimensional PEF coding	17
2.0.9	Accumulating statistics over both x and t	18
2.0.10	Why 2-D PEFs improve gradients	19
2.1	INTERPOLATION BEYOND ALIASING	20
2.1.1	Dilation invariance interpolation	20
2.1.2	Multiscale missing data estimation	21
2.1.3	You are ready for subsequent chapters.	22
2.2	STRETCH MATCHING	22
2.3	DISJOINT REGIONS OF SPACE	22
2.3.1	Geostatistics	22
2.3.2	Gap filling	24
2.3.3	Rapid recognition of a spectral change	24
2.3.4	Boundaries between regions of constant spectrum	25
3	Updating models using PEFs	27
3.0.5	A giant industrial process	27
3.1	Code for model updating with PEFs	28
3.1.1	Applying the adjoint of a streaming filter	29
3.1.2	Code for applying $\mathbf{A}^*\mathbf{A}$ while estimating \mathbf{A}	29
3.2	DATA MOVEMENT	30
3.2.1	Instability management by regularization	30
3.2.2	Technical issues for seismologists	30

CONTENTS

3.2.3	Where might we go from here?	31
3.2.4	Antoine Guitton's Marmousi illustrations	31
3.2.5	Conclusion	34
4	Missing data interpolation	35
4.0.6	Stationary PEF infill	35
4.0.7	Nonstationary PEF infill	37
5	Vector-valued signals	41
5.0.8	Multi channels = vector-valued signals	41
5.1	MULTI CHANNEL PEF	43
5.1.1	Vector signal scaling	43
5.1.2	Pseudocode for vector signals	44
5.1.3	How the conjugate gradient method came to be oversold	45
5.1.4	The PEF output is orthogonal to its inputs	45
5.1.5	Restoring source spectra	45
5.2	CHOLESKY DECORRELATING AND SCALING	46
5.3	ROTATING FOR SPARSITY	47
5.3.1	Finding the angle of maximum sparsity (minimum entropy)	47
5.3.2	3-component vector data	48
5.3.3	Channel order and polarity	48
5.4	RESULTS OF KAIWEN WANG	48
6	Inverse interpolation	51
6.0.1	Sprinkled signals go to a uniform grid via PEFed residuals	52
6.1	REPAIRING THE NAVIGATION	55
6.2	IMAGING	55
6.3	DAYDREAMS	55
7	Appendices	57
7.1	WHY PEFs HAVE WHITE OUTPUT	57
7.1.1	Why 1-D PEFs have white output	57
7.1.2	The PEF gives the inverse covariance matrix.	58
7.1.3	Why 2-D PEFs have white output	58

7.2 THE HEART OF NONSTATIONARY PEF USING CALCULUS 59

Front matter

It is not that I'm so smart. But I stay with the questions much longer. –A.E.

0.1 PREFACE

After what in 2014 was to be my final book, *Geophysical Image Estimation by Example*¹, (GIEE) I stumbled on an approach to a large amount of geophysical data model fitting that is much simpler than traditional approaches. Even better, it avoids the often unreasonable academic presumption of stationarity (i.e., time and space invariant statistics). I could not resist embarking on this tutorial.

My previous book *GIEE* is freely available at <http://sep.stanford.edu/sep/prof/> or in paper for a small price at many booksellers, or at the printer, Lulu.com. It is widely referenced herein.

For teachers: I recommend covering material in this order: (1) GIEE Chapter 1 on adjoints, (2) this tutorial on PEFs, (3) GIEE conjugate gradients with diverse applications.

The most recent version of this manuscript should be at the website *Jon Claerbout's classroom*. Check here: <http://sep.stanford.edu/sep/prof/>. The manuscript you are now reading was formed November 28, 2019.

I am now ready to share further development with any and all. I'd like someone to teach me to learn how to use `Git` to make the book publicly available. Any participant is welcome to contribute illustrations (and ideas)—perhaps becoming a coauthor, even taking over this manuscript. The first priority now is more examples. Ultimately, all the examples should be presented in reader rebuildable form. Being 81 years old I'd like to retire to the role of back-seat driver.

Early beta versions of this tutorial will fail to provide rebuildable illustrations. I am no longer coding myself, so if there are ever to be rebuildable illustrations, I need coauthors. I set for myself the goal to take this tutorial out from beta when 50% of the illustrations can be destroyed and rebuilt by readers.

¹ Claerbout, J., 2014, *Geophysical Image Estimation by Example*: Lulu.com.

0.2 INTRODUCTION

The word *nonstationary* is commonly defined in the world of time signals. Signals become nonstationary when their mean or their variance changes. More interestingly, and the focus herein, signals become nonstationary when their spectrum (frequency content) changes.

The word *nonstationary* is also taken to apply to images, such as earth images, and also to wavefields seen with clusters of instruments. Wavefields are nonstationary when their arrival direction changes with time or location. They are nonstationary when their 2-D (two-dimensional) spectrum changes.

Herein the word *nonstationary* also refers to sampling irregularity. All signal recording instruments cost money; and in the world we study, we never have enough. Further, we are often limited in the locations we can place data recorders. In Chapter 6, the word nonstationary refers to our inability on the earth surface to acquire adequate numbers of uniformly spaced signals.

We require uniformly spaced signals for four reasons: (1) to enable pleasing displays of them, (2) to allow Fourier transformation, (3) to accommodate the equations of physics with *finite differences*, and (4) spectral shaping the residual—the difference between real data and modeled data.

Since spatial sampling uniformity is rarely achievable with real data, this tutorial explains how observed data on a nonuniform grid can be used to make *pseudo data* that is on a uniform grid; and further, linear interpolation of the pseudo data yields the observed data.

0.2.1 What can you do with these methods?

1. Build models to fit data with nonstationary statistics.
2. Perform blind deconvolution (estimate and remove a source wavelet).
3. Fill data gaps. Interpolate beyond aliasing (sometimes).
4. Transform residuals to IID (Independent, Identically Distributed) while fitting.
5. Swap easily among ℓ_1 , ℓ_2 , hyperbolic, and inequality penalties.
6. Stretch a signal unevenly to match another. Images too.
7. Predict price based on diverse aspects.
8. Remove crosstalk in multichannel signals (vector data).
9. Model robustly (i.e., multivariate median versus the mean).
10. Shave models with Occam's razor outdoing the ℓ_1 norm.
11. Bring randomly positioned data to a uniform Cartesian grid.
12. Join the world of BIG DATA by grasping multiple aspects of back projection.

0.2.2 How does it work?

This tutorial is novel by attacking data what is nonstationary, meaning that its statistical characterization is not constant in time and space. The methodology herein works by including a new data value to a previously solved regression. The newly arrived data value requires us to make a small adjustment to the previous solution. Then we continue with all the other data values.

The traditional fitting path is: residual→penalty function→gradient→solver. Herein the simpler path is: modeling→residual into adjoint→epsilon jump.

The simpler path enables this tutorial to cover a wide variety of applications in a small number of pages while yet being more explicit about how coding proceeds.

Although we begin here narrowly with a single 1-D scalar signal y_t , we soon expand broadly with $y_t(x, y, z)$ representing multidimensional data (images and voxels) and then multicomponent (vector-valued) signals \vec{y}_t .

Many researchers dealing with physical continua use “inverse theory” (data model fitting) with little grasp of how to supply the “inverse covariance matrix.” The needed algorithms including pseudo code are here.

0.3 PREDICTION ERROR FILTER = PEF

Knowledge of an autocorrelation is equivalent to knowledge of a spectrum. Less well known is that knowledge of either is equivalent to knowledge of a Prediction Error Filter (PEF).

Partial Differential Equations (PDEs) model the world, while PEFs help us uncover it.

	PDE	PEF
differencing star	input	output
<i>white noise</i> (source)	input	output
<i>colored signal</i>	output	input

0.3.1 PEF history

The name “Prediction Error Filter” appears first in the petroleum exploration industry although the idea emerges initially in the British market forecasting industry in the 1920s as the Yule-Walker equations (*a.k.a.* autoregression). The same equations next appear in 1949 in a book by Norbert Wiener in an appendix by Norman Levinson. Soon after, Enders Robinson extended the PEF idea to multichannel (vector-valued) signals. Meanwhile, as the petroleum exploration industry became computerized it found a physical model for scalar-valued PEFs. They found a lot of oil with it; and they pursued PEFs vigorously until about 1970 when their main focus shifted (to where it remains today) to image estimation. My friends John Burg and John Sherwood understood a 2-D extension to the PEF idea but it went unused until I discovered the helix interpretation of it (in about 1998) and used it extensively in my 2014 book *Geophysical Image Estimation by Example* (GIEE). Beyond 2-D, the PEF idea naturally extends to any number of dimensions. (Exploration industry

data exists in a 5-D space, time plus two Earth surface geographical coordinates (x, y) for each energy source plus another two for each signal receiver.)

From an application perspective, the weakness of autocorrelation, spectrum, and classic PEF is the lack of a natural extension to nonstationarity. Like autocorrelation and spectrum, the PEF theory became clumsy when applied to real-world data in which the statistics varied with time and space. Luckily, the nonstationary method is easy to code, promises quick results, and looks like fun! Although I recently turned 81, I cannot stop thinking about it.

In addition to all the old-time activities that are beginning to get easier and better, progress will be rapid and fun for even more reasons. The emerging field of Machine Learning² shares strong similarities and differences with us. Both fields are based on many flavors of back projection. Herein find about twelve back-projection pseudo codes all based on the (x, y, z, t) metric. Machine learning back projections are not limited to that metric, however they can be slow, and they can be spectacularly fragile. Never-the-less, the Machine Learning community brings a young, rapidly-growing, energetic community to the table, and that is another reason we will make progress and have fun. When this young community gets themselves up to speed, they will be looking for real world problems. Many such problems lurk here.

0.4 CREDITS AND THANKS

I was thrilled to have Antoine Guitton join me in the final month before first printing. By his efforts we now see the theory demonstrated on a test case (Marmousi) that has been studied by dozens of previous researchers. Sergey Fomel triggered this direction of research when he solved the nonstationarity problem that I had posed but could not solve. Bob Clapp ran an inspiring summer research group. Stewart Levin generously welcomes my incomplete thoughts on many topics. He page edited and provided a vastly cleaner 1-D whiteness proof. He did the computations and wrote Chapter 4. John Burg set me on the track for understanding the 2-D PEF. Kaiwen Wang worked with me and made all the illustrations in the multichannel chapter. Joseph Jennings provided the field-data debubble example and commented on early versions of the multichannel chapter. Jason Chang assisted me with LaTeX. Anne Cain did page editing.

Finally, my unbounded gratitude goes to my beloved wife Diane, who accepted to live with a kind of an alien. Without her continuous love and support over half a century, none of my books could have existed.

² See <https://www.youtube.com/watch?v=oJNHXP0XDk> for an 8-minute introduction by Steve Brunton.

Chapter 1

Nonstationary scalar signals

1.0.1 Mathematical setting

Regression defined

Statisticians use the term “regression” for a collection of overdetermined simultaneous linear equations. Given a model \mathbf{m} , a data set \mathbf{d} , a matrix operator \mathbf{F} , the regression defines a residual $\mathbf{r}(\mathbf{m}) = \mathbf{d} - \mathbf{F}\mathbf{m}$. We set out to minimize it $\mathbf{0} \approx \mathbf{r}(\mathbf{m})$.

Regression updating

In the stationary world (the world that assumes statistics are time invariant) there are many solution methods for regressions, both analytic and iterative. In the nonstationary world we presume there is a natural ordering for the regression equations—for the ordering of the components of \mathbf{d} with their rows in \mathbf{M} . Basically, we begin from a satisfactory solution to a regression set. Then an additional regression equation arrives. Call it the new bottom row. We want an updated solution to the updated regression set. This is an old problem in algebra with a well-known solution that assumes the new regression equation should have the same weight as all the old ones. However, we wish to assert that the new row is more valuable than old rows. In this way our solutions have the possibility to evolve along with the evolution of the nature of the incoming data.

For model update we put a residual into an adjoint.

The traditional model fitting path is: residual→penalty function→gradient→solver.

Herein the simpler path is: modeling→residual into adjoint→epsilon jump.

Besides addressing the stationarity issue, this simpler path puts draft codes in your hands for the vast array of issues that commonly arise. Results are broadly equivalent¹.

¹ The quadratic form you are minimizing is $\mathbf{r} \cdot \mathbf{r} = (\mathbf{d} - \mathbf{m}^* \mathbf{F}^*)(\mathbf{d} - \mathbf{F}\mathbf{m})$ with the derivative by \mathbf{m}^* being $-\mathbf{F}^* \mathbf{r}$ for the step $\Delta \mathbf{m} = -\epsilon \mathbf{F}^* \mathbf{r}$.

The special case of filtering

Not for logical reasons, but for the tutorial reason of being specific, we now leave behind the general matrix \mathbf{F} until Chapter 3. Meanwhile, we mostly specialize \mathbf{F} to filtering. This because the Cartesian metric is so central to our geophysical work.

1.0.2 Spectral shaping the residual

We learn by subtracting modeled data from observed data. That difference we call *the residual*. The residual reveals the limitations of our modeling. Understanding those limitations leads towards discoveries. Before residuals are to be minimized to learn the best fitting model, a principle of statistics says residuals should be scaled to uniform strength. Formally, Statistics says the residuals should be Independent and Identically Distributed (IID). In practice this means the residuals should have been scaled up to come out easily visible everywhere in both physical space and Fourier space so that all aspects of the data have been probed.

Suppose after fitting your model parameters you find some region in physical space or in Fourier space where the residuals are tiny. This region is where your data is contributing nothing to your model. Unless you accept that your data is worthless there, you had better scale up those residuals and try fitting again.

There is one region of Fourier space where signals are usually worthless. That is near the Nyquist frequency on the time axis. Why worthless? Because we habitually sample the time axis overly densely to assure that difference equations provide a good mimic of differential equations.

Scaling in physical space is easy, so that's not the topic here. It is because data frequency and dip content varies in physical space that we need Prediction Error Filters (PEFs). They come next. (Stationary theory has a “chicken and egg” problem (commonly ignored) that weights and filters should be constant during iterative solving while they are supposed to end out IID.)

1.0.3 Prediction-error filtering (deconvolution)

Start with a channel of data (a signal of many thousands of values). We denote these data numbers by $\mathbf{y} = (y_0, y_1, y_2, \dots)$. A little patch of numbers that we call a “filter” is denoted by $\mathbf{a} = (a_0, a_1, a_2, \dots, a_{n_\tau})$. In pseudo code these filter numbers are denoted by $\mathbf{a}(0), \mathbf{a}(1), \dots, \mathbf{a}(n_\tau)$. Likewise code for the data.

The filter numbers slide across the data numbers with the leader being $\mathbf{a}(0)$. An equation for sliding the filter numbers across the data numbers obtaining the output r_t is $r_t = \sum_{\tau=0}^{n_\tau} a_\tau y_{t-\tau}$. In a stationary world, the filter values are constants. In our nonstationary world, the filter values change a tiny bit after the arrival of each new data value.

Several computer languages allow the calculation $x \leftarrow x + y$ to be represented by $\mathbf{x}+=\mathbf{y}$. We use this notation herein, likewise $\mathbf{x}-=\mathbf{y}$ for subtraction. Pseudo code for finding $\mathbf{r}(\mathbf{t})$ is:

```

#                               CODE = STATIONARY CONVOLUTION
r(...) = 0.
for all t {
    do tau = 0, ntau
        r(t) += a(tau) * y(t-tau)
    }

```

This code multiplies the vector $\mathbf{a}(\tau)$ into the matrix $\mathbf{y}(t-\tau)$.

With each step in time we prepare to change the filter $\mathbf{a}(\tau)$ a tiny bit. To specify the change, we define the filter outputs $\mathbf{r}(t)$ to be residuals and we set the goal for residuals to have minimum energy. To prevent the filter \mathbf{a} from becoming all zeros, we constrain the first filter coefficient to be unity.

$$\mathbf{a} = [1, a_1, a_2, a_3, \dots] \quad (1.1)$$

To contend with the initial unit “1.0” outputting an input data value, the remaining filter coefficients try to destroy that data value. They must attempt to predict the input value’s negative. The filter output r_t is the residual of the attempted prediction. The name of the filter itself is the Prediction-Error Filter (PEF). PEFs are slightly misnamed because their prediction portion predicts the *negative* of the data.

The PEF output tends to whiteness. Whiteness means flatness in Fourier space. If the prediction is doing a good job, in the residual there should remain nothing periodic to predict. This is rigorously explained in the appendix in Chapter 7.

1.0.4 Code for prediction error = deconvolution = autoregression

Below is the code that does “deconvolution,” also known as “autoregression.” In the `#forward` loop it defines the residual $\mathbf{r}(t)$. In the `#adjoint` loop it puts that residual $\mathbf{r}(t)$ into the same matrix $\mathbf{y}(t-\tau)$ to find the filter update $\mathbf{da}(\tau) = \Delta\mathbf{a}$. Both loops are matrix multiplies, but one takes τ space to t space, while the other takes t space to τ space. Thus one matrix multiply is actually the transpose of the other.

Not only does this code live in a nonstationary world, but it is much simpler than comparable codes that live in a stationary world. Hooray!

```

r(...) = 0.          # CODE = NONSTATIONARY PREDICTION ERROR
a(...) = 0.
a( 0 ) = 1.0
do over time t {    # r(t)      = nonstationary prediction error.
    do tau= 0, ntau
        da(tau) = 0
        r(t)    += a(tau) * y(t-tau)      # forward
    do tau= 0, ntau
        da(tau) += r(t)    * y(t-tau)     # adjoint
    da(0) = 0.      # constraint
    do tau= 0, ntau
        a(tau)  -= da(tau) * epsilon
    }

```

The line `da(0)=0` is a constraint to prevent changing the `a(0)=1` maintaining the definition of $\mathbf{r}(t)$ as a residual. The last `tau` loop updates the PEF.

What we have done in the code is to apply the classroom fundamental: Put the residual into the adjoint² (transpose) to get the gradient; then go down. What remains is to confirm that the code really does reduce the residual.

1.0.5 The heart of nonstationary PEF with no calculus

Magic is coming: At any moment in time, in other words, at the newly arrived bottom regression equation, the old PEF gives an error residual $r_t = \sum_{\tau} a_{\tau} y_{t-\tau}$. Call this bottom row $\mathbf{b} = y_{t-\tau}$. \mathbf{b} is a bit of backward data. The residual there is $r_t = \mathbf{a} \cdot \mathbf{b}$. The filter update in the preceding code amounts to:

$$\text{da}(\text{tau}) \quad -= \quad \text{epsilon} * \mathbf{r}(\text{t}) * \mathbf{y}(\text{t}-\text{tau}) \quad (1.2)$$

$$\Delta \mathbf{a} \quad = \quad - \epsilon r_t y_{t-\tau} \quad (1.3)$$

$$\Delta \mathbf{a} \quad = \quad - \epsilon r_t \mathbf{b} \quad (1.4)$$

The filter output is $r_t = \mathbf{a} \cdot \mathbf{b}$. The updated output is

$$r_t = (\mathbf{a} + \Delta \mathbf{a}) \cdot \mathbf{b} = \mathbf{a} \cdot \mathbf{b} - \epsilon r_t (\mathbf{b} \cdot \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b})(1 - \epsilon(\mathbf{b} \cdot \mathbf{b})) \quad (1.5)$$

This updated output diminishes the output residual provided that $0 < \epsilon < 1/(\mathbf{b} \cdot \mathbf{b})$. Hooray! In volatile circumstances we might choose $\epsilon = 1/(\mathbf{b} \cdot \mathbf{b})$. Because new data is more valuable than old we usually choose $1/N < \epsilon \ll 1/(\mathbf{b} \cdot \mathbf{b})$.

The magic paragraph above encapsulates hard-won knowledge. It exemplifies the basic idea that we may solve nonstationary regressions merely by putting a residual into an adjoint. This approach is used in this tutorial to solve a wide variety of such problems. I was really surprised to see Equation (1.3) fall out of a simple code after I (with much help from Sergey Fomel) had derived it using a good deal of calculus and algebra some of which is in Appendix 5.2. And, all that analysis did not even yield the upper limit on epsilon apparent from Equation (1.5).

1.0.6 Whiteness

Intuitively, PEF output has sucked all the predictability from its input. Appendix 5.1.1 *Why 1-D PEFs have white output* shows that the PEF output tends to be spectrally white—to be a uniform function of frequency. The longer the filter, the whiter the output. The name deconvolution came about from a hypothetical model that the original sources were random impulses, but the received signal became spectrally colored (convolved) by reasons such as wave propagation. Thus, a PEF should return the data to its original impulses. It should deconvolve.

PEFs try to deconvolve, but they cannot restore delays. (This attribute is often called “minimum delay” or “minimum phase.”) They cannot restore delays because the PEF is *causal*, meaning it has only knowledge of the past. This because $[\dots, a_{-2}, a_{-1}] = \mathbf{0}$. Prediction-error filtering is sometimes called *blind* deconvolution—stressing that \mathbf{a} is estimated as well as applied.

²If coding adjoints is new to you, I recommend Chapter 1 in *GIEE* (Claerbout, 2014). It is free on the internet.

1.0.7 Scaling components of gradients

The thing that really matters about a gradient is the polarity of each component. While preserving the polarity of any component, you may shrink or stretch that component arbitrarily. This amounts to a variable change in the penalty function. Later we investigate polarity preserving *nonlinear* axis stretching to achieve behavior like that of the ℓ_1 -norm.

1.0.8 Fluctuations

In a stationary world the gradient is $\Delta \mathbf{a} = \mathbf{Y}^* \mathbf{r}$. The rows of \mathbf{Y}^* contain the fitting functions where, for example, the 9-th row contains the fitting function $\mathbf{y}_9 = y_{t-9}$. In a steady-state (stationary world) the solution is found when $\Delta \mathbf{a} = \mathbf{0}$. But in a non-stationary world we will not find exact vanishing of $\mathbf{d}\mathbf{a}(\tau) = \mathbf{y}(\tau - \tau) * \mathbf{r}(\tau)$ for all $\tau > 0$. Instead, during iteration $\mathbf{d}\mathbf{a}(\tau)$ becomes small and then bounces around. The fluctuation in size of $|\Delta \mathbf{a}|$ is not simply *epsilon*, but the fluctuations diminish as the residual becomes more and more orthogonal to all the fitting functions. We are too new at this game to know precisely how to choose ϵ , how much bouncing around to expect, or really how to characterize nonstationarity; but, we will come up with a good starting guess for ϵ . While theorizing, there is much we can learn from experience.

1.1 PREDICTION ERROR FILTER = PEF

Knowledge of an autocorrelation is equivalent to knowledge of a spectrum. Less well known is that knowledge of either is equivalent to knowledge of a Prediction Error Filter (PEF). Additionally, by being causal the PEF includes phase information. Partial differential equations (PDEs) model the world, while PEFs help us uncover it.

	PDE	PEF
differencing star	input	output
<i>white noise</i> (source)	input	output
<i>colored signal</i>	output	input

Chapter 2 shows the *white noise*, the *colored signal*, and the PEF being multidimensional (being images), while Chapter 5 shows them being vector-valued (multichannel signals).

1.1.1 The outside world—real estate

The regression updating approach introduced here is not limited to convolution matrices. It applies to all regression equations. For each new regression row, subtract from the solution a tiny suitably scaled copy of the new row. Move along; keep doing it. When you run out of equations, you can recycle the old ones. By cycling around a vast number of times with an epsilon tending to zero, you converge to the stationary solution. This updating procedure should be some long-known principle in mathematics. I have stumbled upon something called the *Widrow-Hoff learning rule*, which feels just like this updating.

For example, imagine a stack of records of home sales. The i -th member of the stack is like the t -th time of a signal. The data column contains the recorded sales prices. The first matrix column might contain the square footages, the next column might contain the number of bathrooms, etc. Because many of these variables have all positive elements, we should allow for removing their collective means by including a column of all “ones.” In the signal application, the i -th column contains the signal at the i -th lag. Columns containing all positive numbers might be replaced by their logarithms. The previously shown code finds a_i coefficients to predict (negatively) the signal. Associating lags with real-estate aspects, the code would predict (the negative and possibly the logarithm of) the sales price. You have made the first step towards “machine learning”.

1.2 FINDING TOGETHER MISSING DATA AND ITS PEF

One of the smartest guys I have known came up with a new general-purpose nonlinear solver for our lab. He asked us all to contribute simple test cases. I suggested, “How about simultaneous estimation of PEF and missing data?”

“That is too tough,” he replied.

We do it easily now by appending three lines to the preceding code. The `#forward` line is the usual computation of the prediction error. At the code’s bottom are the three lines for missing-data updating.

```
#                CODE = ESTIMATING TOGETHER MISSING DATA WITH ITS PEF
# y( t) is data.
# miss(t) = "true" where y( miss(t)) is missing (but zero)
r(...) = 0;                # prediction error
a(...) = 0;  a(0) = 1.     # PEF
do t = ntau, infinity {
  do tau= 0,ntau
    r(t)      +=                y(t-tau) * a(tau)    # forward
  do tau= 0,ntau
    if( tau > 0)
      a(tau)  -=  epsilonA * r(t) * y(t-tau)      # adjointA
  do tau= 0,ntau
    if( miss(t-tau))
      y(t-tau)  -=  epsilonY * r(t)      *      a(tau)  # adjointY
}
```

The data update may not be easy to understand, but it is a logical update because a residual is passed into an adjoint. The `#forward` code line takes $(t-tau)$ space to (t) space, while the `#adjointY` line takes (t) space, to $(t-tau)$ space. (I hope I have the correct sign on `epsilonY`!

We are not computing missing data so much as we are updating missing data. It must begin off having some value (such as zero). The forward line uses it. The final code line updates it. All data needs to pass through the program many times. It may also need to pass through backwards too. (Practice will tell us whether going backwards is essential.)

PEF estimation proceeds quickly on early parts of a data volume. Filling missing data is not so easy. You may need to run the above code over all the data many times. To

maintain continuity on both sides of large gaps, you could run the time loop backward on alternate passes. (Simply time reverse both \mathbf{y} and \mathbf{r} after each pass.) To speed the code, one might capture the \mathbf{t} values that are affected by missing data, thereafter iterating only on those.

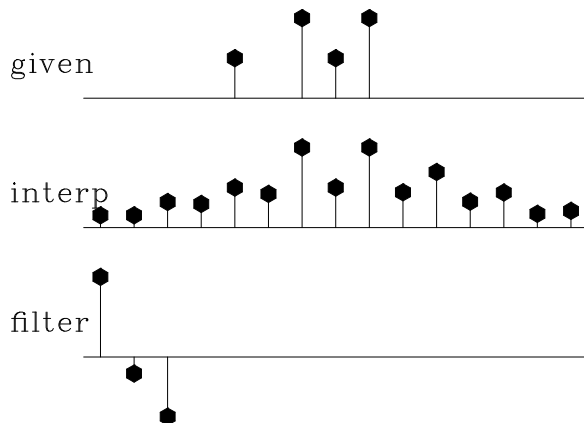
Perhaps because I am a doddering 81-year-old, I have not been able to convince students around here to play with it. Lucky for us, a former student Stewart A. Levin in helping a student prepared the wonderful examples you will see in Chapter 4.

1.2.1 Further progress will require some fun play

Finding missing data with its PEF is a non-linear problem. Code above should work easily so long as a small percentage of data values are missing. On the other hand, with enough missing data the non-linearity might produce good results with some initializations but bizarre results with others.

Few examples are available in the nonstationary world, but examples from the stationary world are strongly suggestive. A result of economic value is found in Chapter 2 (images) Figure 2.1. Another example from the stationary world is the intriguing result in Figure 1.1. It shows four known data values and eleven missing ones. The conclusion to draw is that

Figure 1.1: Top is given data, taken to be zeros off the ends of the axis. Middle is the given data with interpolated values. The restored data has the character of the given data. Bottom shows the best fitting filter. Its output (not shown) has minimum energy. (Claerbout, PVI) `signal/. missif`



PEF interpolation has picked up the character of the known data and used it to fill in the missing. Popular interpolations like linear, cubic, or sinc do nothing like this. The reason PEFs work so well is that they resemble differential equations (actually, finite difference forms of differential equations) which accounts for the more “physical” appearance.

Because data is expensive to collect, missing data examples abound. Consequently, the problems are worthwhile, so we are pushed into experimentation—which should be fun. It would be fun to view the data, the PEF, and the inverse PEF as the data streams through the code. It would be even more fun to have an interactive code with sliders to choose `epsilonA`, `epsilonY`, and our Δt viewing rate.

It would be still more fun to have this happening on images (Chapter 2). Playing with your constructions cultivates creative thinking, asserts the author of the MIT Scratch computer language in his book *Lifelong Kindergarten* (Resnick, 2017). Sharing your rebuildable projects with peers cultivates the same.

The above code is quite easily extended to 2-D and 3-D spaces. The only complication (explained in Chapter 2) is the shape of PEFs in higher dimensional spaces.

I wondered if our missing data code would work in the wider world of applications—the world beyond mere signals. Most likely not. A single missing data value affects τ_n regression equations while a missing home square footage affects only one regression equation.

1.3 CHOOSING THE STEP SIZE

In the method of *steepest* descent, one computes the distance to move along the gradient. Herein we guess it. We might call this the method of *cheapest* descent. (Haha)

1.3.1 Epsilon

An application parameter like `epsilon` requires some practitioner to choose its numerical value. This choice is best rationalized by making sure ϵ is free from physical units. Let us now attend to units. From the past of \mathbf{y} , the filter \mathbf{a} predicts the future of \mathbf{y} , so \mathbf{a} itself must be without physical units. The data y_t might have units of voltage. Its prediction error r_t has the same units. To repair the units in ϵ we need something with units of voltage squared for the denominator. Let us take it to be the variance σ_y^2 . You might compute it globally for your whole data set \mathbf{y} , or you could compute it by leaky integration (such as $\sigma_t^2 \leftarrow .99\sigma_{t-1}^2 + .01y_t^2$) to adjust itself with the nonstationary changes in data y_t . The filter update $\Delta\mathbf{a}$ with a unit-free ϵ is:

$$\Delta\mathbf{a} = -\frac{\epsilon r_t}{\sigma_y^2} \mathbf{d} \quad (1.6)$$

That is the story for `epsilonA` in the code above. For the missing data adaptation rate, `epsilonY`, no normalization is required because $\mathbf{r}(\mathbf{t})$ and $\mathbf{y}(\mathbf{t})$ have the same physical units; therefore the missing data $y_{t-\tau}$ updates are scaled from the residual r_t by the unit-free `epsilonY`.

`epsilonA` is the fractional change to the filter at each time step. In a process called “leaky integration,” any long-range average of the filter at time t is reduced by the $(1 - \epsilon)$ factor; then it is augmented by ϵ times a current estimate of it. After λ steps, the influence of any original time is reduced by the factor $(1 - \epsilon)^\lambda$. Setting that to $1/e = 1/2.718$ says $(1 - \epsilon)^\lambda = 1/e$. Taking the natural logarithm, $1 = -\lambda \ln(1 - \epsilon) \approx \lambda\epsilon$, so to good approximation

$$\epsilon = 1/\lambda \quad (1.7)$$

By the well known property of exponentials, half the area in the decaying signal appears before the distance λ —the other half after.

The memory function $(1 - \epsilon)^t$ is roughly like a rectangle function of length λ . Least squares analysis begins with the idea that there should be more regression equations than unknowns. Therefore, λ should significantly exceed the number of filter coefficients `ntau`.

With synthetic data, you may have runs of zero values. These do not count as data. Then, you need a bigger λ because the zeros do not provide the needed information.

Mathematicians are skilled at dealing with the stationary case. They are inclined to consider all residuals r_t to carry equal information. They may keep a running average m_t of a residual r_t by the identity (proof by induction):

$$m_t = \frac{t-1}{t} m_{t-1} + \frac{1}{t} r_t = \frac{1}{t} \sum_{k=1}^t r_k \quad (1.8)$$

This equation suggests that an ϵ decreasing proportional to $1/t$ (which is like λ proportional to t) may in some instances be a guide to practice, although it offers little guidance for nonstationarity other than that ϵ should be larger; it should drop off less rapidly than does $1/t$.

Given an immense amount of data, a “learning machine” should be able to come up with a way of choosing the adaptivity rate ϵ . But, besides needing an immense amount of data, learning machines are notoriously fragile. We should try conjuring up some physical/geometric concepts for dealing with the kind of nonstationarity that our data exhibits. With such concepts we should require far less data to achieve more robust results. We need examples to fire up our imaginations.

You are ready for Chapter 2.

1.4 NON-GAUSSIAN STATISTICS

The most common reason to depart from the Gaussian assumption in stationary *data fitting* is to tolerate massive bursts of noise. In *model regularization*, the reason is to encourage sparse models. In the stationary world these goals are commonly addressed with the ℓ_1 norm. In our nonstationary world we approach matters differently.

The traditional data fitting path is: residual \rightarrow penalty function \rightarrow gradient \rightarrow solver. Our nonstationary path is: modeling \rightarrow residual into adjoint \rightarrow epsilon jump for $\Delta \mathbf{a}$. Instead of cooking up other penalty functions, we might cook up guesses for nonlinear stretching components in \mathbf{r} or $\Delta \mathbf{a}$. We could measure and build upon the statistics of what we see coming out of r_t and components of $\Delta \mathbf{a}_t$. But, what would be the criteria? Do we need theoretical study, artificial intelligence, or simply examples and practice?

1.4.1 The hyperbolic penalty function

My book *GIEE* has many examples of use of the hyperbolic penalty function. Loosely, we call it ℓ_h . For small residuals it is like ℓ_2 , and for large ones it is like ℓ_1 . Results with ℓ_h are critically dependent on scaling the residual, such as $q = r/\bar{r}$. Our choice of \bar{r} specifies the location of the transition between ℓ_1 and ℓ_2 behavior. I have often taken \bar{r} to be at the 75th percentile of the residuals.

A marvelous feature of ℓ_1 and ℓ_h emerges on model space regularizations. They penalize large residuals only weakly, therefore encouraging models to contain many small values, thereby leaving the essence of the model in a small number of locations. Thus we build sparse models, the goal of Occam’s razor.

Happily, the nonstationary approach allows easy mixing and switching among norms. In summary:

Name	Scalar Residual	Scalar Penalty	Scalar Gradient	Vector Gradient
ℓ_2	$q = r$	$q^2/2$	q	\mathbf{q}
ℓ_1	$q = r$	$ q $	$q/ q $	$\text{sgn}(\mathbf{q})$
ℓ_h	$q = r/\bar{r}$	$(1 + q^2)^{1/2} - 1$	$q/(1 + q^2)^{1/2}$	$\text{softclip}(\mathbf{q})$

From the table, observe at q large, ℓ_h tends to ℓ_1 . At q small, ℓ_h tends to $q^2/2$ which matches ℓ_2 . To see a hyperbola $h(q)$, set $h - 1$ equal to the Scalar Penalty in the table, getting $h^2 = 1 + q^2$. The *softclip()* function of a signal applies the ℓ_h Scalar Gradient $q/(1 + q^2)^{1/2}$ to each value in the residual.

Coding requires a model gradient $\Delta\mathbf{m}$ or $\Delta\mathbf{a}$ that you form by putting the Vector Gradient into the adjoint of the modeling operator, then taking the negative. If you want ℓ_2 , ℓ_1 , or ℓ_h , then your gradient is either $\Delta\mathbf{a} = -\mathbf{Y}^*\mathbf{q}$, $-\mathbf{Y}^*\text{sgn}(\mathbf{q})$, or $-\mathbf{Y}^*\text{softclip}(\mathbf{q})$. You may also tilt the ℓ_h penalty making it into a “soft” inequality like “ReLU” in machine learning.

(Quick derivation: People choose ℓ_2 because its line search is analytic. We chose epsilon instead. For the search direction, let $P(\mathbf{q}(\mathbf{a}))$ be the Scalar Penalty function. The step direction is $-\Delta\mathbf{a} = \frac{\partial P}{\partial \mathbf{a}^*} = \frac{\partial P}{\partial \mathbf{q}^*} \frac{\partial \mathbf{q}^*}{\partial \mathbf{a}^*} = \frac{\partial \mathbf{q}^*}{\partial \mathbf{a}^*} \frac{\partial P}{\partial \mathbf{q}^*} = \mathbf{Y}^* \frac{\partial P}{\partial \mathbf{q}^*}$ where for $\frac{\partial P}{\partial \mathbf{q}^*}$ you get to choose a Vector Gradient from the table foregoing.)

An attribute of ℓ_1 and ℓ_2 fitting is that $\|\alpha\mathbf{r}\| = \alpha\|\mathbf{r}\|$. This attribute is not shared by ℓ_h . Technically ℓ_h is not a norm; it should be called a “measure.”

1.4.2 How can the nonstationary PEF operator be linear?

Formally, finding the PEF is $\mathbf{a} = \text{argmin}_{\mathbf{a}}(\mathbf{Y}\mathbf{a})$ subject to $a_0 = 1$, while using it is $\mathbf{r} = \mathbf{A}\mathbf{y}$. The combination is a nonlinear function of the data \mathbf{y} . But it is nearly linear. Notice that \mathbf{A} could have been built entirely from spatially nearby data, not at all from \mathbf{y} . Then \mathbf{A} would be nonstationary, yet a perfectly linear operator on \mathbf{y} .

I am no longer focused on conjugate-direction solutions to stationary linear problems, but if I were, I could at any stage make two copies of all data and models. The solution copy would evolve with iteration while the other copy would be fixed and would be used solely as the basis for PEFs. Thus, the PEFs would be changing with time while not changing with iteration, which makes the optimization problem a linear one, fully amenable to linear methods. In the spirit of conjugate gradients (as it is commonly practiced), on occasion we might restart with an updated copy. People with inaccurate adjoints often need to restart. (ha ha)

1.5 DIVERSE APPLICATIONS

1.5.1 Weighting

More PEF constraints are common. PEFs are often “gapped” meaning some a_τ coefficients following the “1” are constrained with $\Delta a_\tau = 0$. See the example in Chapter 2, Figure 1.3.

In reflection seismology, t^2 gain and debubble do not commute. Do the physics right by applying debubble first; then get a bad answer (because late data has been ignored). Do the statistics right; apply gain first; then violate the physics. How do we make a proper nonstationary inverse problem? I think the way is to merge the t^2 gain with the ϵ .

1.5.2 Change in variables

Because all we need to do is keep $\mathbf{d} \cdot \mathbf{d} = \mathbf{d}^* \mathbf{d}$ positive, we immediately envision more general linear changes of variables in which we keep $\mathbf{d}^* \mathbf{B}^* \mathbf{B} \mathbf{d}$ positive, implying the update $\Delta \mathbf{a} = -\epsilon r_t \mathbf{d}^* \mathbf{B}^* \mathbf{B}$. I conceive no example for that yet.

1.5.3 Wild and crazy squeezing functions

The logic leading up to Equation (1.3) requires only that we maintain polarity of the elements in that expression. Commonly, residuals like r are often squeezed down from the ℓ_2 -norm derivative r , to their ℓ_1 derivative, $\text{sgn}(r) = r/|r|$, or the derivative of the hyperbolic penalty function, $\text{softclip}(r)$. Imagine an arbitrary squeezing function $\text{RandSqueeze}()$ that squeezes its argument by an arbitrary polarity-preserving squeezing function. Each τ might have its own $\text{RandSqueeze}_\tau()$ mixing $\text{signum}()$ and $\text{softclip}()$ and the like. The possibilities are bewildering. We could update PEFs with the following:

$$\Delta a_\tau = -\epsilon \text{RandSqueeze}(r_t) \text{RandSqueeze}_\tau(y_{t-\tau}) \quad (1.9)$$

Recall the real estate application. It seems natural that each of the various columns with their diverse entries (bathrooms, square footages) would be entitled to its own $\text{RandSqueeze}_\tau()$. Given enough data, how would we identify the $\text{RandSqueeze}_\tau()$ in each column?

1.5.4 Deconvolution of sensible data mixed with giant spikes

The difference between $\text{sgn}(r_t)$ and $\text{sgn}(y_{t-\tau})$ is interesting. Deconvolution in the presence of large spike noise is improved using $\text{sgn}(r_t)$ to downplay predicting corrupted data. It is also improved by downplaying—with $\text{sgn}(y_{t-\tau})$ —regression equations that use corrupted data to try predicting good data. On the other hand, because a humongous data value is easy to recognize, we might more simply forget squeezing and mark such a location as missing data value.

1.5.5 My favorite wavelet for modelers

I digress to view current industrial marine wavelet deconvolution. Because acoustic pressure vanishes on the ocean surface, upcoming waves reflect back down with opposite polarity. This reflection happens twice, once at the air gun (about 10 meters deep), and once again at the hydrophones yielding roughly a second finite-difference response called a “ghost.” Where you wish to see an impulse on a seismogram, instead you see this ghost.

The Ricker wavelet, a second derivative of a Gaussian, is often chosen for modeling. Unfortunately, the Gaussian function is not causal (not vanishing before $t = 0$). A more

natural choice derives from the Futterman wavelet (*GIEE*) which is a causal representation of the spectrum $\exp(-|\omega|t/Q)$ where Q is the quality constant of rock. Figure 1.2 shows the Futterman wavelet and also its second finite difference. I advocate this latter wavelet for modelers because it is solidly backed by theory; and I often see it on data. The carry-away thought is that the second derivative of a Gaussian is a three-lobed wavelet, while that is hardly true of the second derivative of a Futterman wavelet.

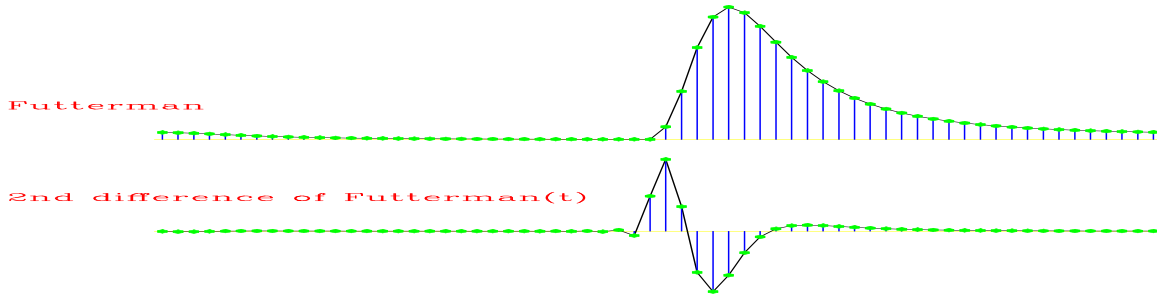


Figure 1.2: The causal constant Q response and its second finite difference. The first two lobes are approximately the same height, but the middle lobe has more area. That third lobe is really small. Its smallness explains why the water bottom could seem a Ricker wavelet (second derivative of a Gaussian) while the top of salt would seem a doublet. (Claerbout)

signal/. Futter

1.5.6 Bubble removal

The internet easily finds for you slow-motion video of gun shots under water. Perhaps unexpectedly, the rapidly expanding exhaust gas bubble soon slows; then, collapses to a point, where it behaves like a second shot—repeating again and again. This reverberation period (the interval between collapses) for exploration air guns (“guns” shooting bubbles of compressed air) is herein approximately 120 milliseconds. Imagers hate it. Interpreters hate it. Figure 1.3 shows marine data and a gapped PEF applied to it. It is a large gap, 80 milliseconds (ms), or $80/4=20$ samples on data sampled at 4 ms, actually, $\Delta \mathbf{a} = (1, 0, 0, \text{more zeros}, 0, a_{20}, a_{21}, \dots, a_{80})$.

REFERENCES

- Claerbout, J., 2014, Geophysical image estimation by example: Lulu.com.
 Resnick, M., 2017, Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play: The MIT Press, Cambridge, MA.

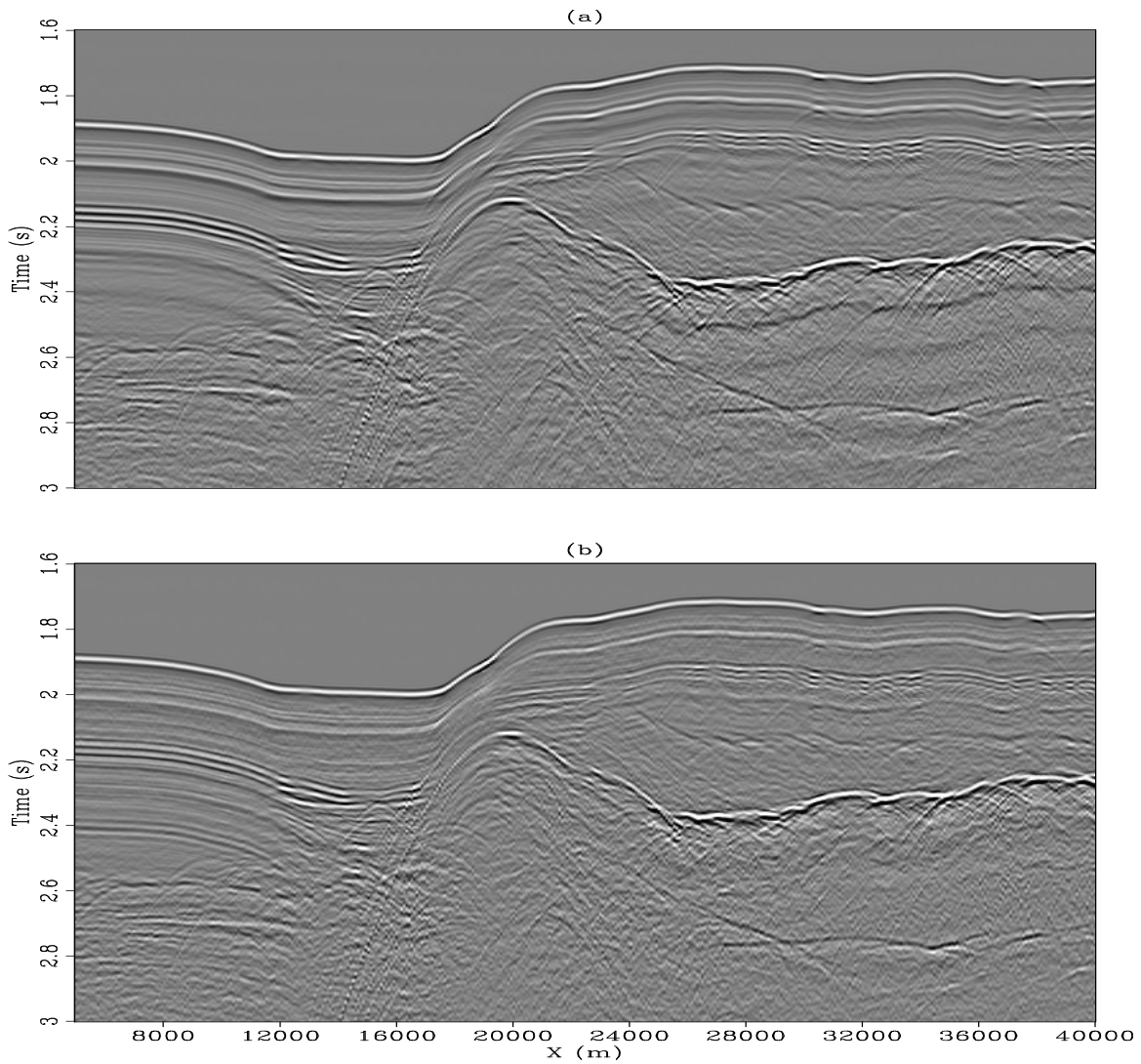


Figure 1.3: Debubble done by the nonstationary method. Original (top), debubbled (bottom). On the right third of the top plot, prominent bubbles appear as three quasihorizontal black bands between times 2.4s and 2.7s. Blink overlay display would make it more evident that there is bubble removal everywhere. (Joseph Jennings) `signal/. debubble-ovcomp`

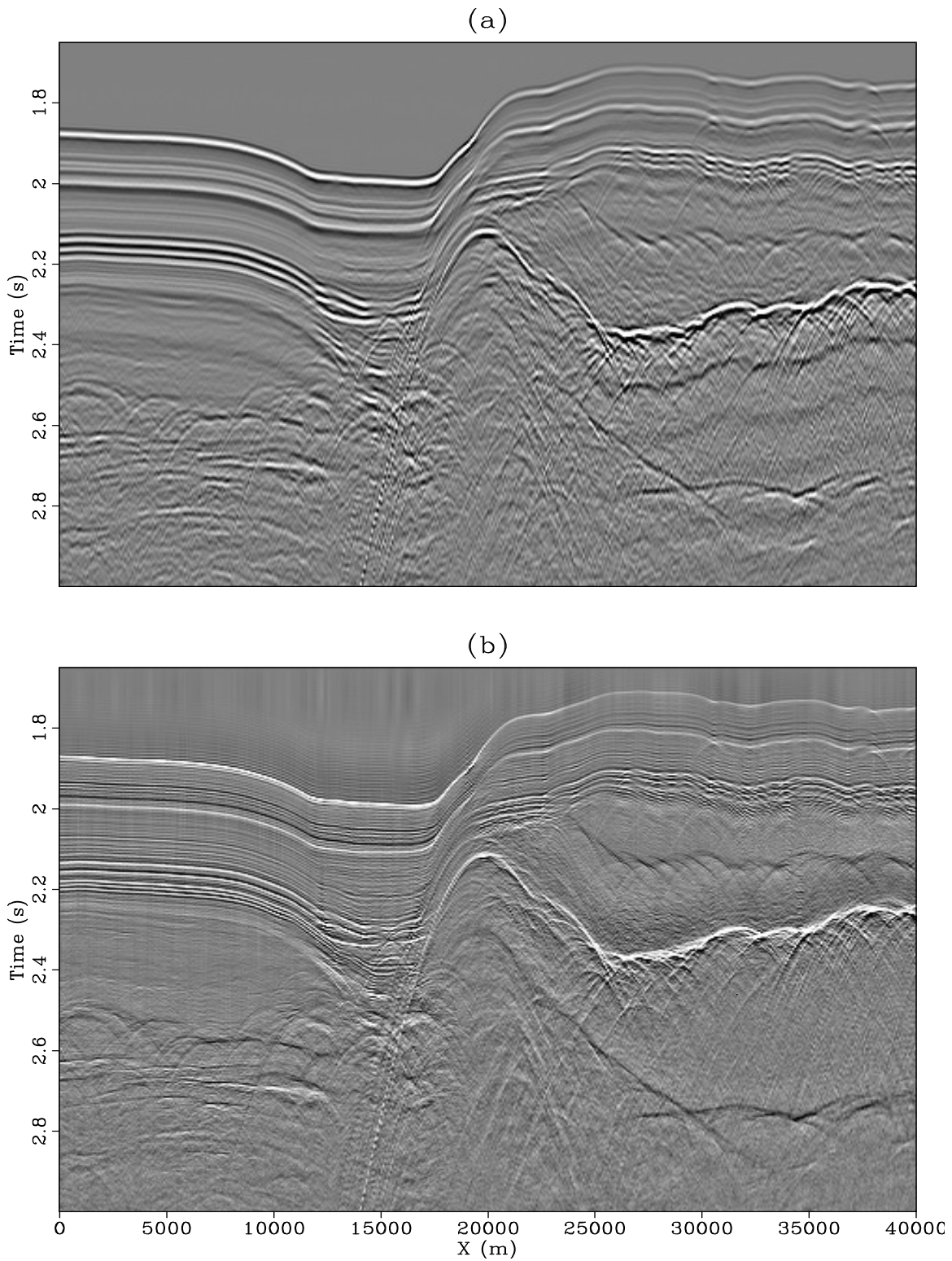


Figure 1.4: Gulf of Mexico. Top is before sparse decon, bottom after. Between 2.25s to 2.70s, the right side is salt (no reflectors). Notice salt top reflection is white, bottom black. Notice that sparse decon has eliminated bubble reverberation in the reflection-free salt zone (as well as elsewhere). (Antoine Guitton) [signal/. antoineGOM2](http://signal/.antoineGOM2)

Chapter 2

PEFs in time and space

In this chapter we deal with 2-D functions of space, say the (x, y) plane. About the same mathematics applies to a survey line of signals, say a (t, x) data plane. In one dimension PEFs do a spectacular job of destroying periodic functions. They do an admirable job of dealing with resonant signals. Further, we can use them to fill gaps in 1-D signals.

Now we move into two dimensions. 2-D PEFs do an excellent job of destroying (or building) straight lines. On a data space, they will destroy (or build) plane waves.

Point scatterers in the earth emit circular waves, say $d(t - \sqrt{(x - x_0)^2 + (z - z_0)^2})$. Locally these may look a little like plane waves, but they are not. In (x, t) space they are hyperbolic. We struggled for years chopping data into small patches where the plane wave approximation has some degree of validity. The problem with the patching approach is the many boundaries connecting the small patches. Non-stationary PEFs resolve big chunks of this difficulty.

Figure 2.1 shows an old stationary example from *GIEE*. In the stationary case, a global PEF is computed first; then, it is used to fill missing data.

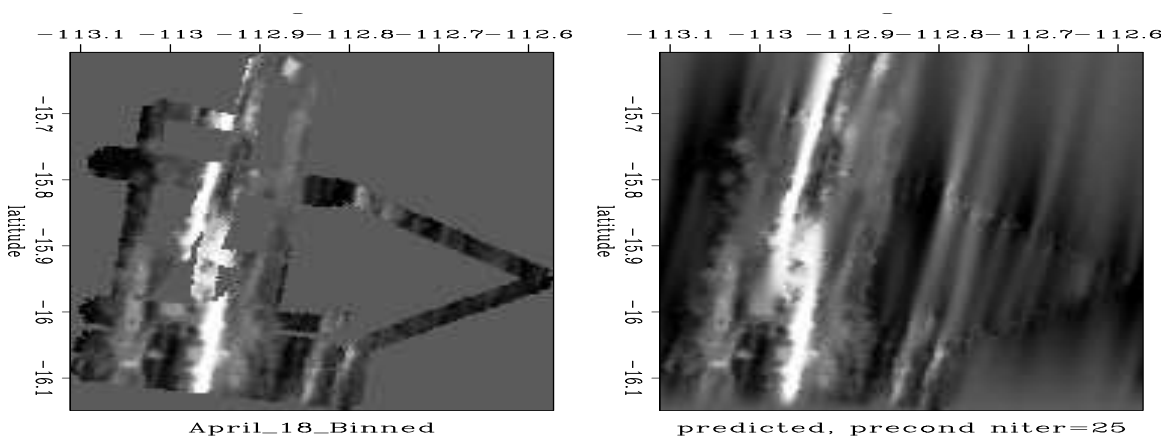


Figure 2.1: (left) Seabeam data of mid-Pacific transform fault. (right) After interpolation by stationary 2-D PEF. The purpose herein is to guess what the ship would have recorded if there were more hours in a day. (*GIEE*) [image/. seapef90](#)

In one dimension PEF output tends to whiteness. In two dimensions, the codes we assemble herein produce outputs that tend to 2-D whiteness, tending to flatten nonstationary spectra in the 2-D frequency (ω, k_x) -space. In other words, the local autocorrelation of the output tends to a delta function in 2-D lag space. In other words, we will be broadening the 2-D bandwidth of whatever signal we design the PEF upon.

We learn about the earth by fitting models to data. Chapter 3 shows how 2-D PEFs play a central role in this learning process. What is significantly new in this book is a pathway to dealing with curving events. This is the situation we always have in seismology where the angle of propagation varies from place to place.

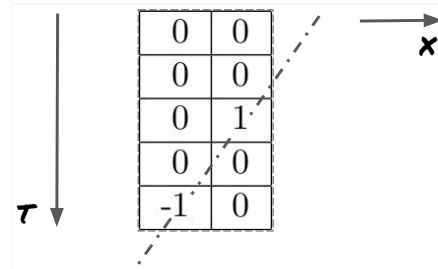
2.0.7 2-D PEFs as plane wave destructors and plane wave builders

We have seen 1-D PEFs applied to 2-D data. Now for 2-D PEFs. Two-dimensional PEFs are useful in seismology. Convolving an image with the PEF in Figure 2.2 would destroy aspects of the image with slope 2. Nearby slopes would merely be suppressed. Linear interpolation suggests that a PEF with a slightly lesser angle can be specified by spreading the -1 , by moving a fraction of it from the -1 to the pixel above it.

Newcomers often feel the $+1$ should be in a corner, not on a side, until they realize such a PEF could not suppress all angles. For example putting the $+1$ on the top right corner we would not be able to find coefficients inside the PEF that would destroy lines running southeast to northwest.

Figure 2.2: Plane wave destructor for events of slope 2. Applied to data it destroys that slope in the data. Used in a missing data program, that slope is produced where the data is missing. (Claerbout)

`image/. DippingPEF5`



A PDF can be specified, as I did in making Figure 2.2, or it can be learned from earlier codes. After a PEF is known, it may be used to fill in missing data as on page 6. Using the PEF in Figure 2.2 in a filtering program, that slope is destroyed. Using that PEF in a missing data program, that slope is built. (Outside our present topic of nonstationary data, stationary methods using polynomial division can fill large holes significantly more rapidly than the method herein.)

Convolving two PEFs each with a different slope builds a wider PEF able to destroy simultaneous presence of two differently sloped plane waves. In reflection seismology the vertical axis is time and the horizontal axis distance, so steep slopes are slow velocities.

2.0.8 Two-dimensional PEF coding

Signal analysis extends to images quite easily except for the 1.0 spike needing to be on the side of the PEF as in Figure 2.3. This old knowledge is summarized in Appendix 5.1.2 *Why 2-D PEFs have white output.*

Figure 2.3: A PEF is a function of lag $a(\tau_1, x_1)$. Here τ runs up, x runs left so the filter travels down and right. (Claerbout)

`image/. pef2-d`

$a(2,2)$	$a(2,1)$	$a(2,0)$
$a(1,2)$	$a(1,1)$	$a(1,0)$
$a(0,2)$	$a(0,1)$	1.0
$a(-1,2)$	$a(-1,1)$	0
$a(-2,2)$	$a(-2,1)$	0

Unlike our 1-D code, we now use negative subscripts on time.

As in 1-D, the PEF output is aligned with its input because $a(0,0)=1$. To avoid filters trying to use off-edge inputs, no output is computed (first two loops) at the beginning of the x axis nor at both ends of the time axis. At three locations in code below the lag loops (τ_1, x_1), cover the entire filter. First, the residual $r(\tau, x)$ calculation (`# Filter`) is simply the usual 1-D convolution seen again on the second axis. Next, the adjoint follows the usual rule of swapping input and output spaces. Then the constraint line preserves not only the 1.0, but also the zeros preceding it. Finally, the update line `a-=da` is almost trivial¹.

```
#                               CODE = 2-D PEF
read y(  0...nt , 0...nx)      # data
      r(  0...nt , 0...nx)=0.  # residual = PEF output
      a(-nta...nta, 0...nxa)=0. # filter      Illustrated size is a( -2...2, 0...2).
      a(  0      , 0      )=1.0 # spike
do for x = nxa to nx
do for t = nta to nt-nta

    do for x1=  0 to +nxa
    do for t1= -nta to +nta
        da(t1,x1) = 0.
        r ( t , x ) += a(t1,x1) * y(t-t1, x-x1)      # Filter

    do for x1=  0 to +nxa
    do for t1= -nta to +nta
        da(t1,x1) += r(t , x) * y(t-t1, x-x1)      # Adjoint

    do for t1= -nta to 0
        da(t1, 0) = 0.      # Constraints

    do for x1=  0 to +nxa
    do for t1= -nta to +nta
        a ( t1,x1) -= da(t1,x1) * epsilon/variance # Update
```

¹ Beware of instability if ϵ is taken too large. A stability limit for ϵ is defined after equation (1.5).

2.0.9 Accumulating statistics over both x and t

We will see in Chapter 3 that nearly everyone's code for fitting models to survey data needs a 2-D PEF. A serious limitation of the foregoing code (CODE = 2-D PEF) is that the data statistics are updated entirely from the time axis. You are surveying down a road. Every 100 meters you record a 10 second signal. Then you update a 2D-PEF handling these signals (traces) one after another. After the bottom of one trace you return to wholly different statistics (especially wave slopes) at the top of the next. You need to have saved all the PEFs of the previous trace and be relying initially on those at early times.

A straightforward extension to the 1-D code allows us to average the statistics in a 2-D region of (x, t) -space. Define this region as an area of roughly $\lambda_t \times \lambda_x$ pixels.

We update filters with $\mathbf{a} \leftarrow \bar{\mathbf{a}} + \epsilon \Delta \mathbf{a}$. Previously we updated the PEF from the location $\bar{\mathbf{a}}(t - \Delta t, x)$. Now we begin also updating from the neighboring trace $\bar{\mathbf{a}}(t, x - \Delta x)$. The proposal is to update from a weighted average $\bar{\mathbf{a}}$ where

$$\bar{\mathbf{a}} = \mathbf{a}(t - \Delta t, x) \frac{\lambda_t^2}{\lambda_t^2 + \lambda_x^2} + \mathbf{a}(t, x - \Delta x) \frac{\lambda_x^2}{\lambda_t^2 + \lambda_x^2} \quad (2.1)$$

The scale factors sum to unity so we may designate them by $\cos^2 \theta$ and $\sin^2 \theta$ named `cos2t` and `sin2t` in the code. We need to allocate memory for each filter at the previous space level $x - \Delta x$ so we allocate memory `ab(nxa, nta, nt)`. At the first value of x we cannot refer to the previous x so the allocated memory should be initialized to zeros with each filter having a properly placed 1.0. The most basic allocation and initialization follows:

```
allocate ab( -nta:nta, 0:nxa, nt)
ab(*,*,*) = 0
do for t = 0 to nt
  ab(0,0,t) = 1.0
```

Returning to (CODE = 2-D PEF), we now need to update the filter adding $\Delta \mathbf{a}$ to the $\bar{\mathbf{a}}$ of equation (2.1). The last three lines of (CODE = 2-D PEF) become these five lines

```
do for xl= 0 to +nxa
do for tl= -nta to +nta
  a (tl,xl) = a(tl,xl)*cos2t + ab(tl,xl,t)*sin2t      # Average
  a (tl,xl) -= da(tl,xl) * epsilon/variance           # Update
  ab(tl,xl,t) = a(tl,xl)                             # Remember for x+dx
```

The line `#Average` implements equation (2.1). The next line was already in (CODE = 2-D PEF). The last line puts the new \mathbf{a} in the buffer `ab(*,*,t)` for the next trace.

Notice that λ_t and λ_x need not be constants. The length and width of the region of statistical averaging may vary in x and t .

Before you jump to a new trace, you have the option of smoothing `ab(*,*,t)` by leaky integration *backwards* on time. Doing this, your region of smoothing has doubled from the quarter plane prior to t and x to the half plane prior to x . The adjoint of a 2-D filter is the same filter with both t and x axes reversed. Bob Clapp makes the sensible recommendation of alternating between the PEF and its adjoint.

2.0.10 Why 2-D PEFs improve gradients

This example shows why PEFs improve gradients. The initial residual (starting from $\mathbf{m} = \mathbf{0}$) is the data \mathbf{d} . Figure 2.4 shows a shot gather \mathbf{d} before and after stationary PEFing $\mathbf{A}\mathbf{d}$. Notice the back-scattered energy (travel time decreasing as distance increases). Near zero offset, it almost vanishes on the raw data whereas it is prominent after the PEF. This backscattered energy tells us a great deal about reflectors topping near 2.5-2.8s. Here is why PEFs improve gradients: Strong and obvious but redundant information is subdued, enabling subtle information to become visible, hence sooner to come into use, not waiting until quirks of the strong are exhaustively over interpreted.

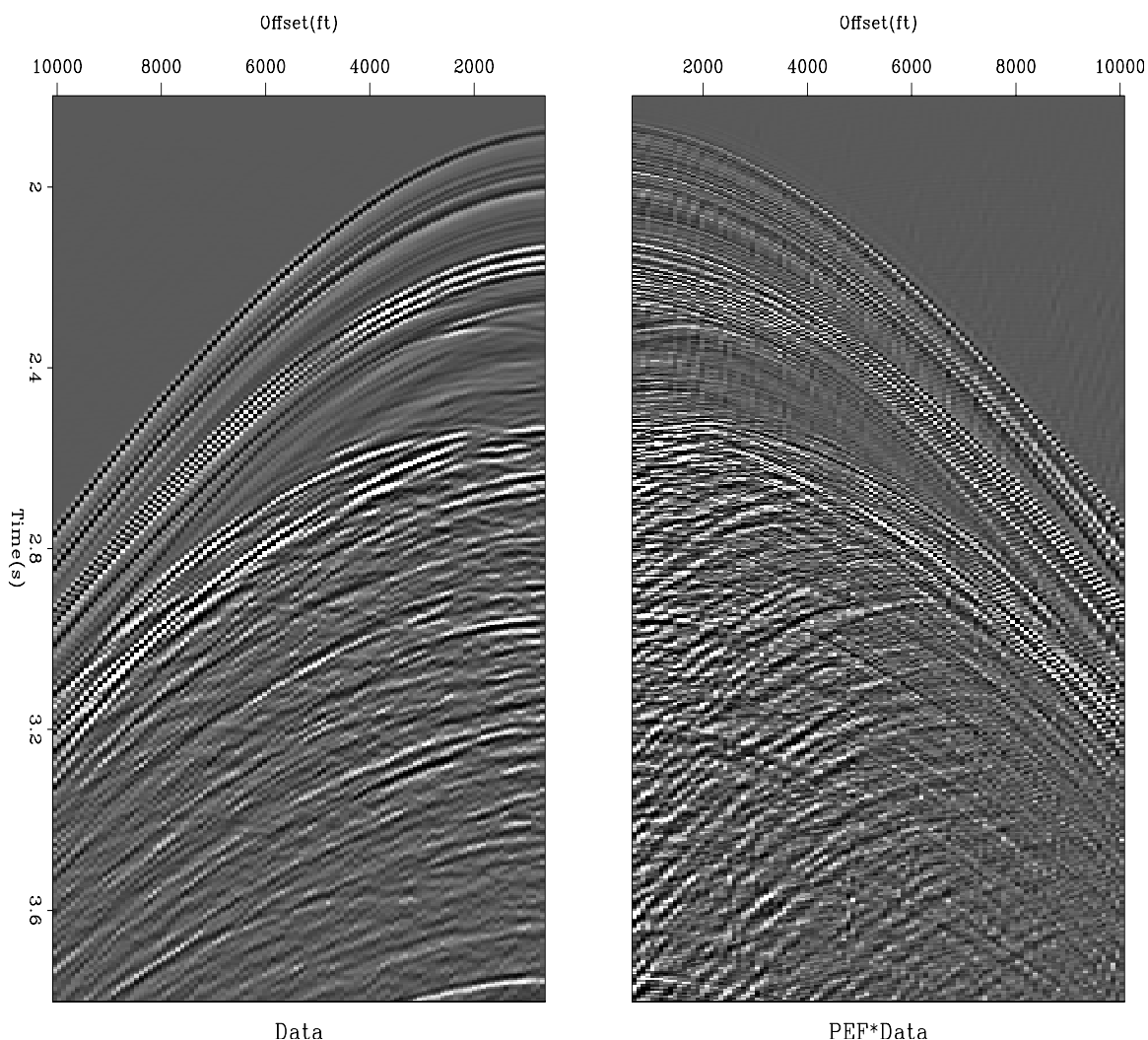


Figure 2.4: (left) Shot gather; (right) mirror imaged after global 2D PEF (20×5). (Antoine Guitton, GIEE) [image/. antoinedecon2](#)

Fortunately, Antoine Guitton at the Colorado School of Mines joined me in the last month of this book preparation to demonstrate these ideas in the next chapter, Chapter 3.

2.1 INTERPOLATION BEYOND ALIASING

Wavefields are parameterized by their temporal frequency, and by their velocity, namely, their slope in (x, t) -space, altogether, two 1-D functions. PEFs in (x, t) -space are a 2-D function. Consequently, with a PEF, we have more adjustable coefficients than needed to characterize waves. PEFs can characterize stuff we might well consider to be noise. Herein however, PEFs are calculated in such a manner that forces them to be more wave-like.

The scalar wave equation template has the property of “dilation invariance,” meaning that halving all of $(\Delta t, \Delta x, \Delta y, \Delta z)$ on a finite difference representation of the scalar wave equation leaves the finite differencing template effectively unchanged. Likewise we may impose the assumption of dilation invariance upon a PEF. We may apply it with all of $(\Delta t, \Delta x, \Delta y, \Delta z)$ doubled, halved, or otherwise scaled. In other words, we may interlace both x and t axes with zeros. A PEF that perfectly predicts plane waves of various slopes can be interlaced with zeros on both time and space axes still predicting the same slopes. Such a PEF scaling concept was used in my book (Claerbout, 1992) *Earth Soundings Analysis, Processing versus Inversion* (PVI) with the assumption of stationarity to produce Figure 2.5. It shows badly spatially aliased data processed to interpolate three intermedi-

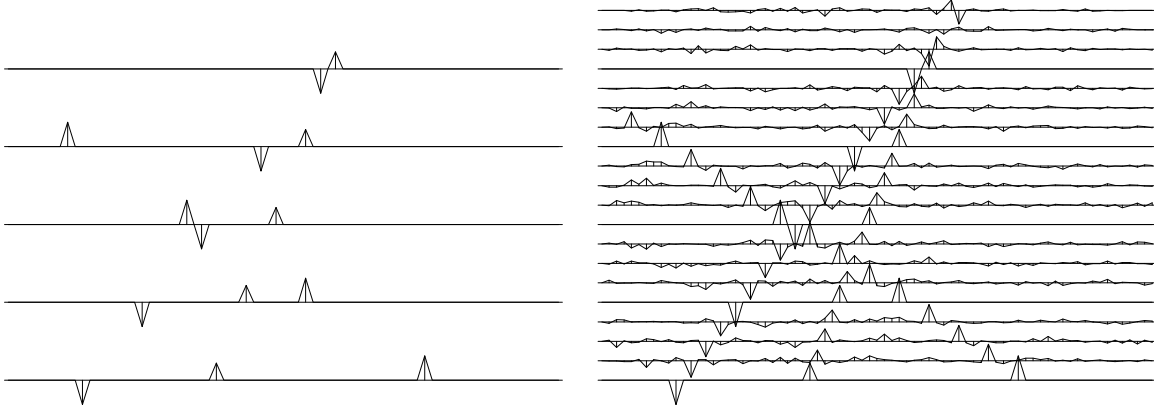


Figure 2.5: Left is five signals, each showing three arrivals. An expanded PEF from the left was compressed to create interpolated data on the right. There are three new traces between the given traces. The original traces are preserved. (Claerbout, PVI) [image/. lace3](#)

ate channels. Naturally, an imaging process (such as “migration”) would fare much better with the interpolated data. Sadly, the technique never came into use, both because of the complexity of the coding, and because of the required stationarity assumption. Herein both those problems are addressed and (I believe) solved. Starting from our earlier pseudo code for missing data on page 6, and the pseudo code 2-D PEF on page 17, let us combine these ideas into three additional lines of pseudo code to do the job in a nonstationary world, a world of curving event arrivals, data gaps, but not large gaps.

2.1.1 Dilation invariance interpolation

The 2-D PEF code on page 17 contains line (1) below. Line (2) is likewise, but it accesses prediction signals at double the distance away from the data being predicted. These two

lines produce two different residuals \mathbf{r}_1 and \mathbf{r}_2 , each of them densely sampled on time \mathbf{t} and \mathbf{x} . We should create and study three frame blink movies $[\mathbf{y}|\mathbf{r}_1|\mathbf{r}_2]$ of miscellaneous seismic data to gain some insights I cannot predict theoretically: Which of \mathbf{r}_1 and \mathbf{r}_2 is better? Is that true for all kinds of data? Is \mathbf{r}_2 a reasonable proxy for \mathbf{r}_1 ?

```

Loops over t and x:
  Loops over filter (t1,x1):
(1)      r1(t ,x ) += a(t1,x1) * y(t-t1 , x-x1 )
(2)      r2(t ,x ) += a(t1,x1) * y(t-t1*2, x-x1*2)           # Dilated PEF

  Loops over filter (t1,x1):
    Only where da() is unconstrained:
(3)      da(t1,x1) -= r1(t , x) * y(t-t1 , x-x1 ) * epsilon1
(4)      da(t1,x1) -= r2(t , x) * y(t-t1*2, x-x1*2) * epsilon2

```

Line (3) updates the PEF from \mathbf{r}_1 , while line (4) updates it from \mathbf{r}_2 . It does not hurt to use both the updates, although only one is needed. We could average them, or weight them inversely by a running norm of their residual, or find some reason to simply choose one of them.

2.1.2 Multiscale missing data estimation

Observe the form of missing data updates in one dimension from pseudocode on page 6. Express it in two dimensions, without and with trace skipping.

```

Loops over t and x:
  Loops over filter (t1,x1):
      r1(t) = same code as above           # usual PEF
      r2(t) = same code as above           # Dilated PEF
  Loops over filter (t1,x1):
    Only where data is missing:
(5)      y(t-t1, x-x1 ) -= r1(t,x) * a(t1, x1) * epsilon3
(6)      y(t-t1*2,x-x1*2) -= r2(t,x) * a(t1, x1) * epsilon4

```

We intend to use only lines (2), (4), and (5), with the usual looping statements and constraints that you find in earlier codes. Start from missing data presumed zero.

```

#          CODE = INTERPOLATION BEYOND ALIASING
(2)      r2( t , x ) += a( t1,x1) * y(t-t1*2, x-x1*2)
(4)      da( t1, x1 ) -= r2( t ,x ) * y(t-t1*2, x-x1*2) * epsilon2
(5)      y ( t-t1, x-x1 ) -= r1( t ,x ) * a( t1, x1 ) * epsilon3

```

Line (2) uses “long legs” to reach out to make a residual for a sparse filter. Line (4) updates that filter. Line (5) asks us for the dilation invariance assumption $\mathbf{r}_1 \approx \mathbf{r}_2$, then switches to the dense filter. Assuming $\mathbf{r}_1(\mathbf{t},\mathbf{x})=\mathbf{r}_2(\mathbf{t},\mathbf{x})$, line (5) updates $\mathbf{y}(\mathbf{t},\mathbf{x})$ where it is not known.

Viscosity breaks the dilation invariance of the scalar wave equation. I wonder what would break it on PEFs ($\mathbf{r}_1 \neq \mathbf{r}_2$). I await someone to perform tests. Should dilation invariance fail on field data, the excellent stationary result in Figure 2.5 suggests a pathway nearby remains to be found.

2.1.3 You are ready for subsequent chapters.

2.2 STRETCH MATCHING

Sometimes we have two signals that are nearly the same but for some reason, one is stretched a little from place to place. Tree rings seem an obvious example. I mostly encounter seismograms where a survey was done both before and after oil and gas production, so there are stretches along the seismogram that have shrunken or grown. A decade or two back, navigation was not what it is now, especially for seismograms recorded at sea. Navigation was one reason, tidal currents are another. Towed cables might not be where intended. So, signals might shift in both time and space. A first thought is to make a running crosscorrelation. The trouble is, crosscorrelation tends to square spectra which diminishes the high frequencies, those being just the ones most needed to resolve small shifts. Let us consider the time-variable filter that best converts one signal to the other.

Take the filter \mathbf{a} to predict signal \mathbf{x} from signal \mathbf{y} . Either signal might lag the other. Take the filter to be two-sided, $[\mathbf{a}(-9), \mathbf{a}(-8), \dots, \mathbf{a}(0), \mathbf{a}(1), \dots, \mathbf{a}(9)]$. Let us begin from $\mathbf{a}(0)=1$, but not hold that as a constraint because the signals may be out of scale.

```

r(...) = 0.          # CODE = NONSTATIONARY EXTRAPOLATION FILTER
a(...) = 0.
a( 0 ) = 1.
do over time t {    # r(t) = nonstationary extrapolation error
  do i= -ni, ni
    r(t) += a(i) * y(t-i) - x(t)          # forward
  do i= -ni, ni
    a(i) -= r(t) * y(t-i) * epsilon      # adjoint
  do i= -ni, ni
    shift(t) = i * a(i)
  }

```

The last loop is to extract a time `shift` from the filters. Here I have simply computed the moment. That would be correct if signals \mathbf{x} and \mathbf{y} had the same variance. If not, I leave it to you calculate their standard deviations σ_x and σ_y and scale the shift in the code above by σ_x/σ_y thus yielding the `shift` in pixels.

Do not forget, if you have only one signal, or if it is short, you likely should loop over the data multiple times while decreasing epsilon.

Besides time shifting, this filtering operator has the power of gaining and of changing color. Suppose, for example that brother \mathbf{y} and sister \mathbf{x} each recited a message. This filtering could not only bring them into synchronization, it would raise his pitch. Likewise in 2-D starting from their photos, he might come out resembling her too much!

2.3 DISJOINT REGIONS OF SPACE

2.3.1 Geostatistics

Figure 2.6 illustrates using PEF technology refilling an artificial hole in an image of the Gulf of Mexico. This illustration (taken from *GIEE*) uses mature stationary technology. The

center panel illustrates filling in missing data from knowledge of a PEF gained outside the hole. The statistics at the hole in the center panel are weaker and smoother than the statistics of the surrounding data. Long wavelengths have entered the hole but diminish slowly in strength as they propagate away from the edges of known data. Shorter wavelengths are less predictable and diminish rapidly to zero as we enter the unknown. Actually, it is not low frequency but narrow bandedness that enables projection far into the hole from its boundaries.

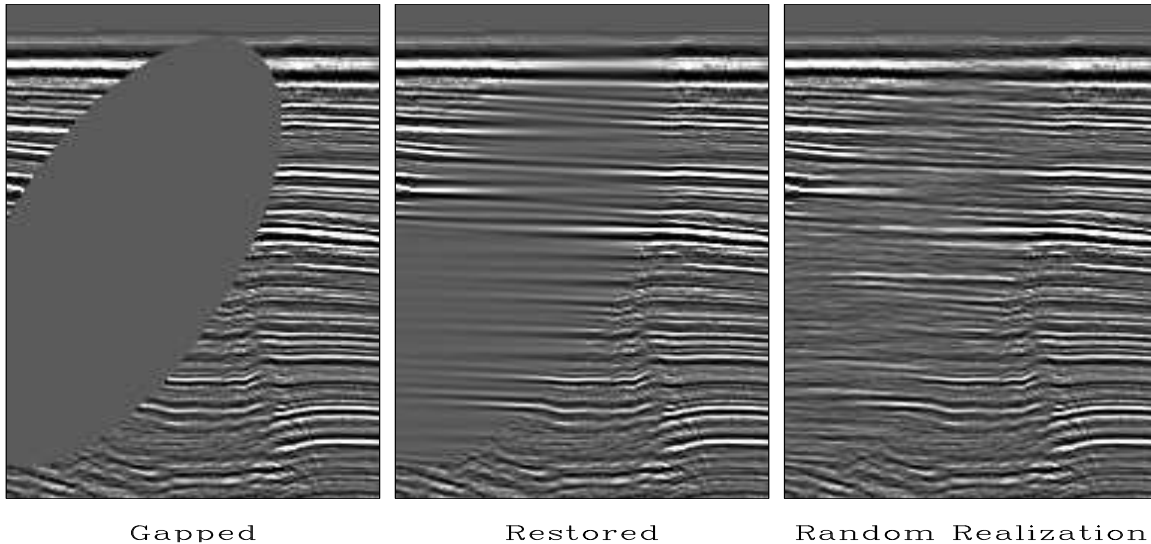


Figure 2.6: A 2-D stationary example from *GIEE*. A CDP stack with a hole punched in it. The center panel attempts to fill the hole by methodology similar to herein. The right panel uses random numbers inverse to the PEF to create panel fill with the global spectrum while assuring continuity at the hole boundary. (Morgan Brown) [image/. WGstack-hole-fillr](#)

The right panel illustrates a concept we have not covered. This panel has the same spectrum inside the hole as outside. *Nice*. And, it does not decay in strength going inward from the boundaries of the hole. *Nice*. Before I ask you which you prefer, the central panel or the right panel, I should tell you that the right panel is one of millions of panels that could have been shown. Each of the millions uses a different set of random numbers. A statistician (i.e., Albert Tarantola) would say the solution to a geophysical inverse problem is a random variable. The center panel is the mean of the random variable. The right panel is one realization of the many possible realizations. The average of all the realizations is the center panel.

Geophysicists tend to like the center panel; geostatisticians tend to prefer an ensemble of solutions, such as the right panel. In stationary theory, the center panel solves a regularization such as $\mathbf{0} \approx \mathbf{A}\mathbf{m}$. The solution to the right panel uses a different regularization, $\mathbf{0} \approx \mathbf{A}\mathbf{m} - \mathbf{r}$, where \mathbf{r} is random numbers inside the hole and zeros outside. The variance of the prediction error outside would match the variance of the random numbers inside. Got it? Good. Now it is your turn to write a nonstationary program. Let's call it "CODE = GEOSTATISTICS."

Start from my 1-D missing data program on page 6. Make the Geostatistics modifications. Test them on the example of Figure 1.1. If your results are fun, and I may use them,

your name will be associated with it.

2.3.2 Gap filling

When filling a 1-D gap, I wonder if we would get the same fill if we scanned time backward. Stationary theory finds a PEF from the autocorrelation function. In that world, the PEF of forward-going data must be identical with that of backward-going data. But, when it comes to filling a gap in data, should we not be using that PEF going in both directions? We should experiment with this idea by comparing one direction to two directions. Would convergence run faster if we ran alternating directions? After each time scan we would simply time reverse both the input and the output, y_t and r_t , for the next scan. In 2-D, reversal would run over both axes.

You might like to jump to Chapter 3

2.3.3 Rapid recognition of a spectral change

This booklet begins with the goal of escaping the strait jacket of stationarity, intending merely to allow for slowly variable spectral change. Real life, of course has many important examples in which a spectral change is so rapid that our methods cannot adapt to it—imagine you are tracking a sandstone. Suddenly, you encounter a fault with shale on the other side and permeability is blocked—this could be bad fortune or *very* good fortune!

Warming up to an unexpectedly precise measurement of location of spectral change consider this 1-D example: Let $T = 1$ and $o = -1$. The time function

$$(\dots, T, T, T, o, o, o, T, T, T, o, o, o, T, T, T, o, o, T, T, o, o, T, T, o, o, T, T, o, o, T, T, o, o, \dots)$$

begins with period 6 and abruptly switches to period 4. The magnitude of the prediction error running to the right is quite different from the one running to the left. Running right, the prediction error is approximately zero, but, it suddenly thunders at the moment of spectral change, thunder gradually dying away again as the PEF adapts. Running left, again there is another thunder of prediction error; but, this thunder is on the opposite side of the abrupt spectral change. Having both directions is the key to defining a sharp boundary between the two spectra. Let the prediction variance going right be σ_{right} and going left be σ_{left} . The local PEF is then defined by a weighted average of the two PEFs.

$$\mathbf{a} = \frac{\sigma_{\text{right}}}{\sigma_{\text{right}} + \sigma_{\text{left}}} \mathbf{a}_{\text{left}} + \frac{\sigma_{\text{left}}}{\sigma_{\text{right}} + \sigma_{\text{left}}} \mathbf{a}_{\text{right}} \quad (2.2)$$

A weight is big where the other side has big error variance. The width of the zone of transition is comparable to the duration of the PEFs, much shorter than the distance of adaptation. This is an amazing result. We have sharply defined the location for the spectral change even though the PEF estimation cannot be expected to adapt rapidly to spectral changes. Amazing! This completes your introduction for the image of Lenna, Figure 2.8.

2.3.4 Boundaries between regions of constant spectrum

There is no direct application to predicting financial markets. But, with recorded data, one can experiment with predictions in time forward, and backward. Including space with time makes it more intriguing. In space, there is not only forwards and backwards but sideways and at other angles. The PEF idea in 3-D (Figure 2.7) shows that sweeping a plane (the top surface) upward through a volume transforms an unfiltered upper half-space to a filtered lower one. Whatever trajectory the sweep takes, it may also be done backward, even at other angles.

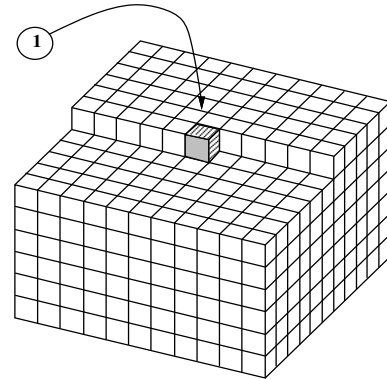


Figure 2.7: The coefficients in a 3-D PEF. (*GIEE*) `image/. 3dpef`

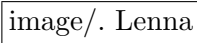
You are trying to remove noise from the test photo of Lenna (Figure 2.8). Your sweep abruptly transitions from her smooth cheek to her straight hair, to the curly fabric of her hat. To win this competition, you surely want sweeps in opposite directions or even more directions. Fear not that mathematics limits us to slow spectral transitions. The location of a sharp spectral transition can be defined by having colliding sweeps, each sweep abruptly losing its predictability along the same edge. But Lenna is not ours yet.

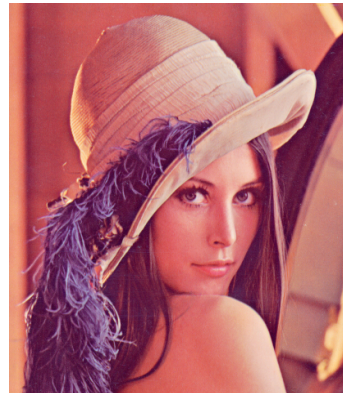
How should we composite the additional sweeps that are available in higher dimensional spaces? Obviously, we get two sweep directions for each spatial dimension; but, more might be possible at 45° angles or with hexagonal coordinates.

Unfortunately, Equation (2.2), is actually wrong (one of the PEFs needs to be reversed), and, obviously, PEFs of various rotations cannot be added. The various angles, however, do help define regions of near homogeneity, but putting it all together to best define Lenna, remains a challenge.

REFERENCES

Claerbout, J. F., 1992, *Earth Soundings Analysis: Processing versus Inversion*: Blackwell Scientific Publications.

Figure 2.8: Lenna, a widely known photo used for testing engineering objectives in photometry. (Wikipedia) 



Chapter 3

Updating models using PEFs

While fitting modeled data to observed data, the residuals should be scaled and filtered to be uniform in variance as a function of space and of frequency. This notion, called IID, was introduced on page 2. PEFs allow us to achieve data fitting with IID'd residuals. An appendix (on page 58) shows that PEFs build in the notion of inverse covariance matrix.

Chapter 1 shows how to compute a PEF in one dimension. Nonstationary methodology allows us frequency being a function of time. Chapter 2 shows how to compute PEFs in higher dimensional spaces. Nonstationary methodology allows us dip being a function of location.

To tackle a wide range of physical problems we now introduce an operator \mathbf{F} that may define a wide range of physical settings. Upon finding a physical residual $\mathbf{r} = \mathbf{d} - \mathbf{F}\mathbf{m}$ we may compute its PEF \mathbf{A} by means of Chapters 1 and 2. It is easy to apply the PEF to the *physical* residual getting the *statistical* residual $\mathbf{q} = \mathbf{A}\mathbf{r} = \mathbf{A}(\mathbf{d} - \mathbf{F}\mathbf{m})$. What remains is our project here, to upgrade the model $\mathbf{m} = \mathbf{m} + \epsilon \Delta\mathbf{m}$ while applying the PEF to the physical residual. How will we get $\Delta\mathbf{m}$?

3.0.5 A giant industrial process

Unless you live in Houston, you've likely never heard of reflection seismology. Mathematically it resembles medical imaging but on a much larger scale. Seismic survey contracting is a multi-billion dollar per year industry whose customers are the petroleum industry. Numerous other industries fit models in Cartesian continua. None appear to use multidimensional PEFs in image building. Without trying to drag you into any of these fields I'd like to show you some samples of data fitting with and without PEFs.

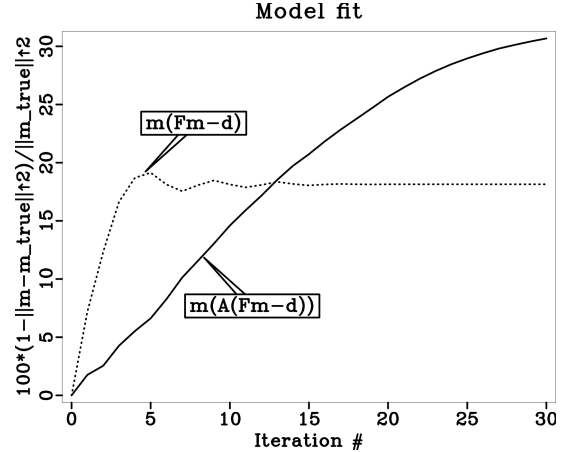
In the real earth we never know the true answer ($\mathbf{m} = \text{earth reflectivity}(x, z)$). But, we can model the data of any \mathbf{m} . The advantage of manufactured data (synthetic data) is we can measure overall quality by the nearness of the estimated model to the true model $\|\mathbf{m} - \mathbf{m}_{\text{true}}\|$ as a function of iteration, whereas for real data we are reduced to looking at the difference between real and modeled data $\|\mathbf{d} - \mathbf{F}\mathbf{m}\|$.

For a well-known and widely studied model (Marmousi), and a well-known operator (Born Modeling operator \mathbf{F}) Figure 3.1 shows two curves as a function of iteration. These

curves are the percentage of the model found, namely $100 \times (1 - \|\mathbf{m} - \mathbf{m}_{\text{true}}\| / \|\mathbf{m}_{\text{true}}\|)$ with and without use of a PEF. The curve raising the higher has found the greater percentage of the true model. The PEF wins, 31% versus 18%. Use of the PEF has enabled finding almost twice as much of the model.

Figure 3.1: Percentage of model found as a function of iteration count. The curve that climbs the higher is using the PEF. Hooray! The astounding aspect is that the PEFs have pulled almost twice as much model from the data. (Guitton)

ag/. compmodelfit2MARINE2



I had expected PEF use to give better answers, but I did not expect it to start off more slowly. The slower start might result from the PEF method trying all slopes whereas without the PEF slopes used first are those dominant in the data. Both desirable attributes, initial speed and ultimate minimum, could be achieved by gapping the coefficients of the PEF during early iterations.

An interesting fact is that with or without the PEF neither final model comes close to fitting the correct one. What is going on? The problem is linear and the size of model space is $121 \times 369 = 44,649$ components. Theoretically, we should get the exact answer in 44,649 iterations. Reality is that we always stop long before then. If we had a unitary operator we should get the correct answer in one iteration. I see the problem stemming from the fact that finite difference operators merely approximate differential operators. Our mathematics is nonphysical at higher frequencies.

3.1 Code for model updating with PEFs

For the special case $\mathbf{m} = \mathbf{0}$, the regression $\mathbf{0} \approx \mathbf{A}(\mathbf{d} - \mathbf{F}\mathbf{m}) = \mathbf{A}\mathbf{d}$ is simply the PEF problem that we solved in earlier chapters. As \mathbf{m} grows, the statistical energy E in the residual $\mathbf{q}(\mathbf{m})$ is expressed as:

$$E = \mathbf{q} \cdot \mathbf{q} = \mathbf{q}^* \mathbf{q} = (\mathbf{d}^* - \mathbf{m}^* \mathbf{F}^*) \mathbf{A}^* \mathbf{A} (\mathbf{d} - \mathbf{F}\mathbf{m}). \quad (3.1)$$

The mismatch energy gradient by the model is:

$$\Delta \mathbf{m} = - \frac{\partial E}{\partial \mathbf{m}^*} = \mathbf{F}^* \mathbf{A}^* \mathbf{A} (\mathbf{d} - \mathbf{F}\mathbf{m}) = \mathbf{F}^* \mathbf{A}^* \mathbf{r}. \quad (3.2)$$

So, the computational problem is to apply $\mathbf{A}^* \mathbf{A}$ to the residual \mathbf{r} simultaneously with finding \mathbf{A} . It is the PEF of \mathbf{r} . Following are the steps to update the model grid:

$$\mathbf{r} = (\mathbf{d} - \mathbf{F}\mathbf{m}) \quad (3.3)$$

The code organization assures us that \mathbf{A} and \mathbf{A}^* apply the same filter. Notice that the program also works when the time axis is run backward. In two dimensions, either or both the axes may be run backward. Flipping axes flips the region in which statistics are gathered.

3.2 DATA MOVEMENT

The approach herein has the potential for “streaming,” meaning that the entire data volume need not be kept in memory—it all flows through the box defined by the codes herein. Our PEFs changed significantly from shot to shot (since dip changes). Although the conjugate direction solver does not require linearity, quick tests showed that our PEFs would change only slightly from iteration to iteration so for this pioneering effort we chose the cautious route of freezing the PEFs after the first iteration. That makes our tested process strictly linear.

The earth velocity in the Marmousi model is given so to simplify matters in this first test the processing uses it without estimating it.

An oversimplified view of the Born Modeling operator \mathbf{F} is that each shot gather is flattened, then they are all added to get the earth image. More correctly, the operator downward propagates hypothetical shots, and downward propagates the observed data, and then crosscorrelates them. Thus earth dips are correctly dealt with. In Figure 3.2, what you see is based on $\mathbf{d} - \mathbf{F}\mathbf{m}$ for a single shot after the first iteration. After the above processing, often called “migration”, all shots are added to create subsequent illustrations of earth images.

3.2.1 Instability management by regularization

We have done no instability management, but theoretically it could be needed. By the “jaggies” it almost looks to be incipient in Figures 3.3-3.4. The general solution is to add to the overall quadratic something like $\epsilon_m^2 \|\mathbf{m}'\mathbf{B}'\mathbf{B}\mathbf{m}\|$ which means after each iteration of data fitting boosting \mathbf{m} , you follow by another shrinking it with $\mathbf{m} \leftarrow \mathbf{m} - \epsilon_m \mathbf{B}\mathbf{m}$. And where does \mathbf{B} come from? Like any other PEF, you build it from \mathbf{m} .

3.2.2 Technical issues for seismologists

1. Born Modeling operator \mathbf{F} with two-way wave equation.
2. Data are streamer data modeling \mathbf{F} (primaries only) with maximum offset of 4 km, with 88 shots spaced by 100 meters. Each shot has 160 traces 25 meters apart.
3. Inverse crime: data are modeled and inverted with the same operator.
4. Maximum frequency is 20Hz with central frequency of 10Hz.
5. Inversion with conjugate direction solver, 30 iterations max.
6. When streaming PEFs are used, $\epsilon = 0.5$ and size of PEF is 20×5 .

7. Streaming PEFs \mathbf{A} are based on shot gathers. They are estimated from the input data and kept constant during the inversion. Each gather has its own filter as a function of time and offset.

3.2.3 Where might we go from here?

If today you were to put PEFs on *receiver* gathers instead of *shot* gathers, you'd be off on a path of original research. Shots are four times as widely separated as receivers, but PEFs generally untroubled by spatially aliased data so it should work fine.

With a little more courage you might think of a 3-D PEF formulation for the shot-geophone-time space. There might seem to be too many filter coefficients, but nothing says those coefficients must be dense in 3-D space.

3.2.4 Antoine Guitton's Marmousi illustrations

Figure 3.2 shows a shot gather. There are 88 of these.

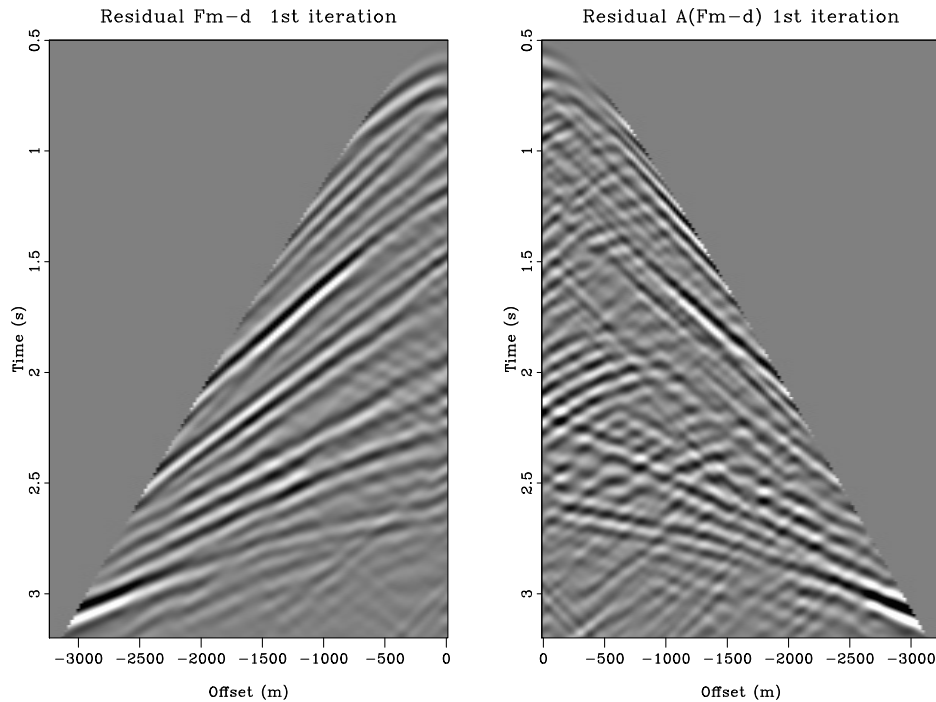


Figure 3.2: A single shot and its mirror reflection, but the reflection has had a 2D PEF applied, so, like the cover of this book, the backscattering is enhanced. Cutting off data above a diagonal line is called “muting”. It is done to eliminate near surface waves since they do not contribute to the final earth image. ag/. compshotwithwithoutnspef42

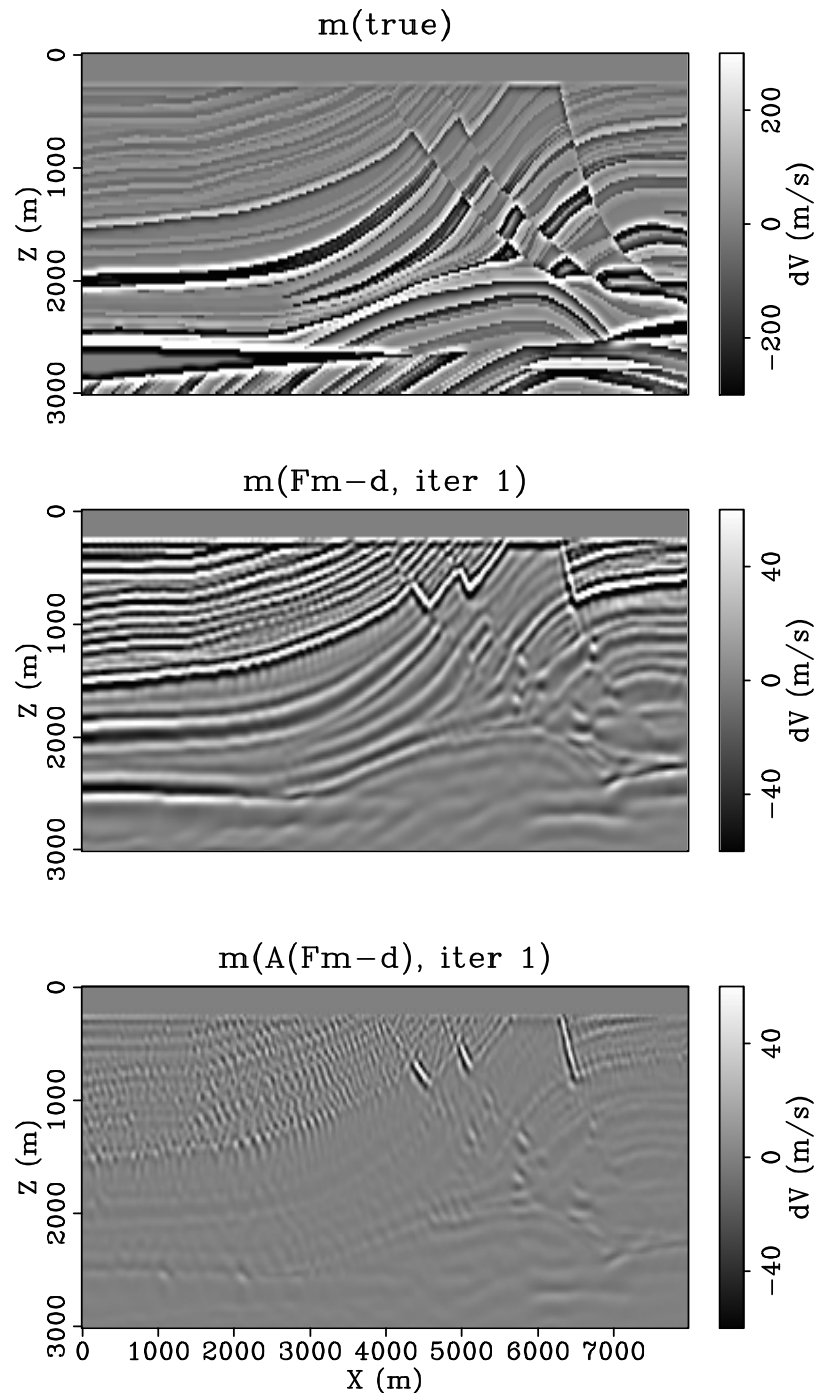


Figure 3.3: After a single iteration we see the traditional adjoint estimate. Original model (top). One iteration of $\mathbf{Fm} - \mathbf{d}$ (middle). One iteration of $\mathbf{A}(\mathbf{Fm} - \mathbf{d})$ (bottom). Because this is the first iteration, results are scaled by a factor five compared to the true model. Notice how without PEFs the flat layers in the shallow part are retrieved easily. Because the PEF whitens the spectrum of the data and highlights backscattering and weak events, the first iteration with PEF \mathbf{A} tends to image faults and diffracted events first explaining why the non-PEFed result comes quicker. `ag/. compmodelsiteration1`

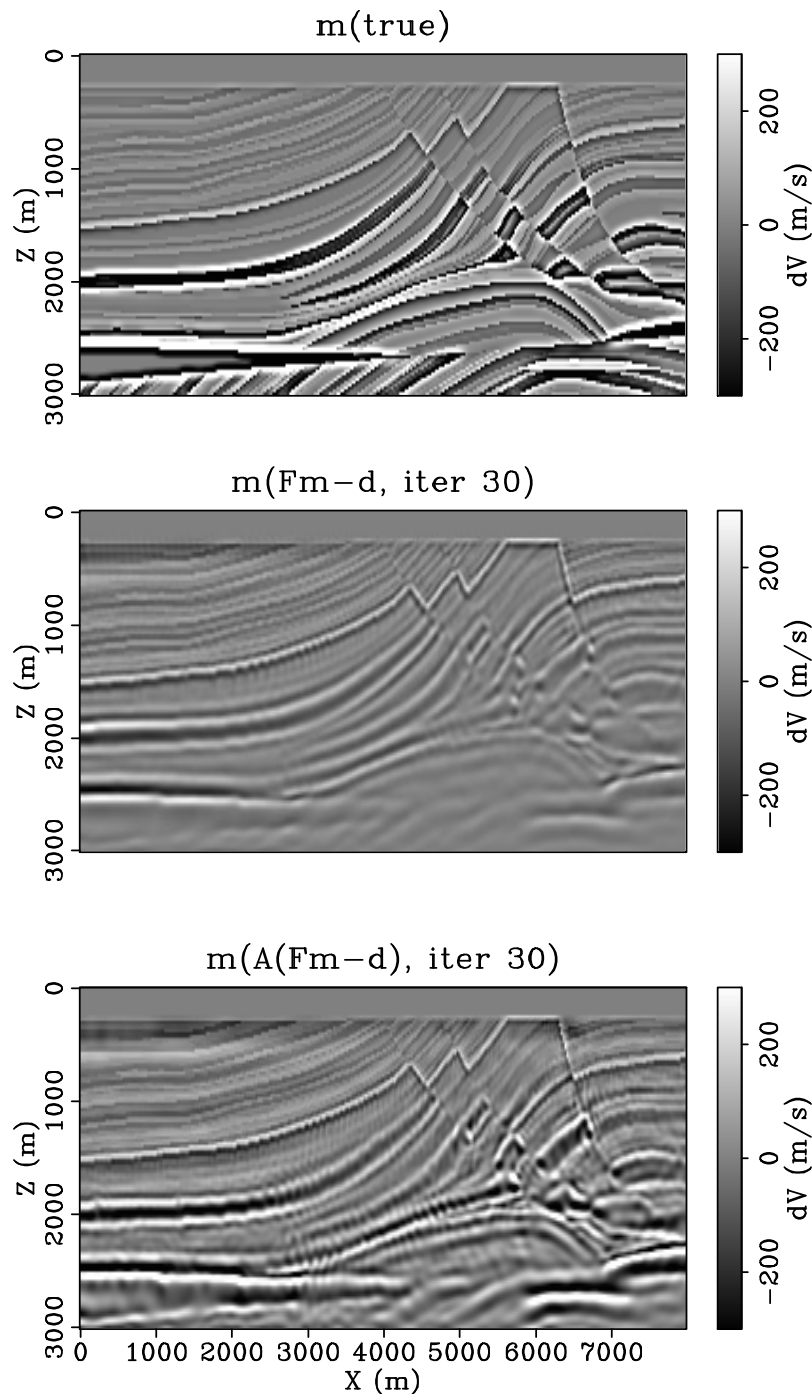


Figure 3.4: Models after 30 iterations. Top: Model \mathbf{m}_{true} . Middle: $\mathbf{m}(\mathbf{r} = (\mathbf{F}\mathbf{m} - \mathbf{d}))$. Bottom: $\mathbf{m}(\mathbf{r} = (\mathbf{A}(\mathbf{F}\mathbf{m} - \mathbf{d})))$. All three images are displayed with the same scaling. The three images are not easily compared on their paper representation herein. It is easier to compare on this internet blink view: <http://sep.stanford.edu/sep/prof/ag.gif>. Select interesting locations to position your pointer. Video of my interpretation is here: <http://sep.stanford.edu/sep/prof/Marmousi2.mp4>. What is obvious on this paper representation is without the PEF the model is weaker. With the PEF higher frequencies have emerged hence the stronger amplitude. [ag/. compmodelsiteration30](#)

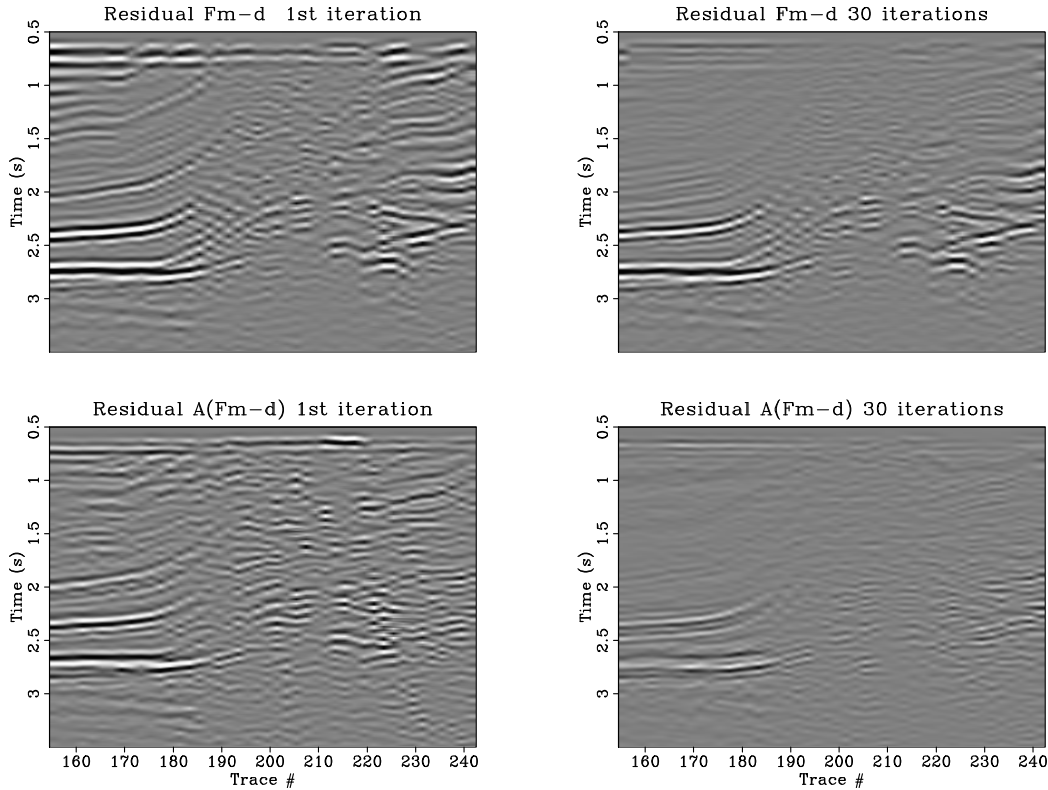


Figure 3.5: Residuals at constant offset ($h=25$ m) for the first iteration (left side) and the last (right side). Non-PEFed on top, PEFed on the bottom. Because these are incommensurate residuals, before and after absolute amplitudes are not comparable. However, notice the PEFed residuals have lost more lower frequency and steeper dipped events. Presumably, those have gone into the model. Also, at early times the PEFed residuals have weakened further suggesting the PEFed results are better solved there than the non-PEFed.

`ag/.compwithwithoutoffsetnspef155`

3.2.5 Conclusion

We are thrilled by these results. The notions of IID and PEF have long been ignored by the seismic imaging community. That community should appreciate the fine results shown here. These results were obtained in one month of Antoine Guitton’s spare time. (He has a day job too.)

Because time was limited, many simplifications were adopted offering later workers (you?) the opportunity to learn whether proceeding more correctly would bring better results or would simply bring trouble. PEFs were frozen at their initial values not changing with iteration. Equation 2.1 and its related code were ignored.

I’m almost ready to speculate on the relation of PEFs to Hessians and on how PEFs might be introduced to the broader problem of simultaneously estimating velocity with reflectivity. When estimating these two, is there a role for two-channel PEF (Chapter 5)?

Chapter 4

Missing data interpolation

¹Geophysical data often has “holes”, either local regions of absent data, or obviously wrong data. We may begin by ignoring them, but as we work there comes a time to deal with the blemishes. As the old Swedish proverb says: *Även solen har sina fläckar.*” (Even the sun has spots.)

Both the stationary and nonstationary approaches to PEF missing data infill are simple, elegant, and have strong theoretical underpinnings. As the proverb suggests, there were small irregularities that might often turn out salient.

One preprocessing step common to both our stationary and nonstationary PEF missing data infill was to generate a locally-smoothed copy of our data before interpolation, smoothly interpolated into the missing data regions, and subsequently adding it back after PEF infill. This step is beneficial because we want our PEF design to focus on local texture and not any broad, smooth background upon which the texture is present. After all, almost any standard interpolator can handle smooth interpolation.

4.0.6 Stationary PEF infill

The GIEE approach to PEF interpolation of missing data is elegant, powerful, and unconditionally stable. Its main limitation is the stationary assumption that the data texture and their statistics are invariant across the image section. In many cases, such as in Figure 4.1, breaking up the input into, possibly overlapping, patches and infilling each separately can be effective, however klunky.

One concern that has been raised repeatedly is that our PEFs are unidirectional, stepping along a helix unwrapping of the multidimensional grid. After all, there is nothing causal about an elevation map. Recall that stationary theory finds a PEF from the auto-correlation function. In that world, i.e. one where either there is no missing data or the PEF is one contiguous set of coefficients in the helix sense, the PEF of forward-going data must be identical with that of backward-going data. In practice, for more than one dimension a normal, noncontiguous PEF layout will not always land on the identical set of known values and there can be some differences.

¹Adapted from SEP report 173

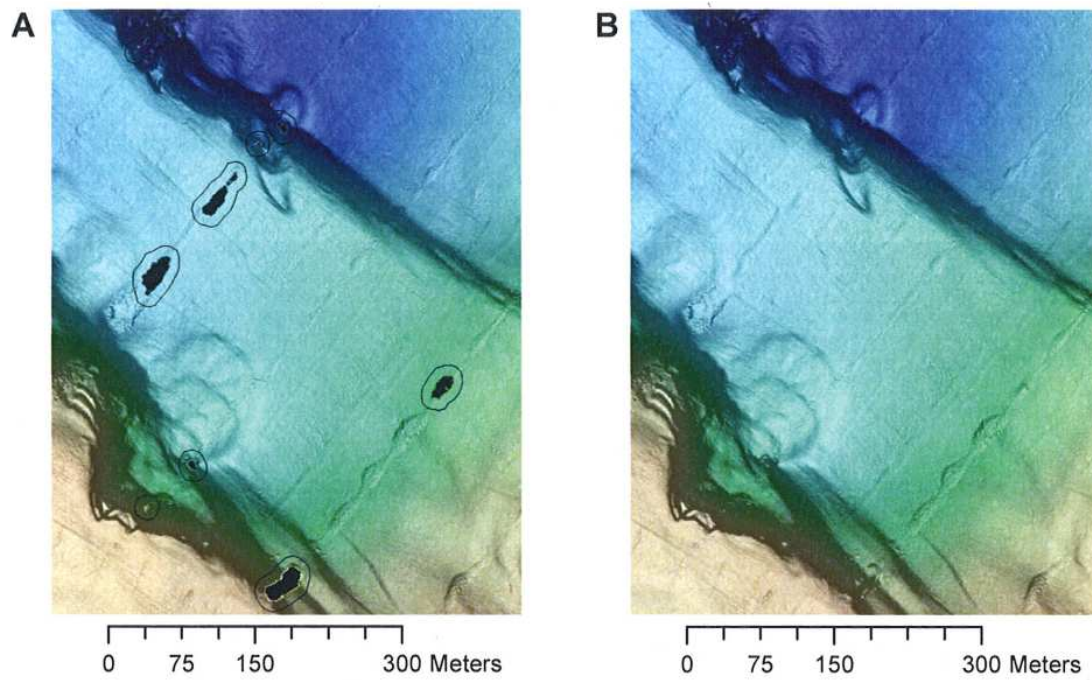


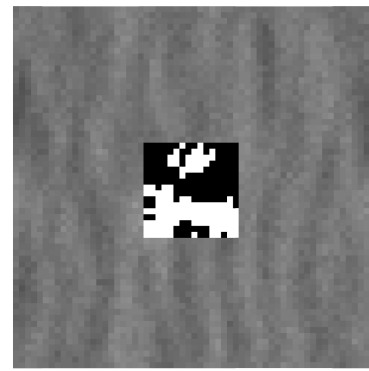
Figure 4.1: Gulf of California AUV Bathymetry collected by MBARI interpolated using local stationary PEF. The black patches are data gaps and the black lines outline the areas of interest on which PEFs were designed. Courtesy C.M. Castillo (2018). [sal/. GofCLocalPEF](https://github.com/mbari/gofclocalpef)

4.0.7 Nonstationary PEF infill

Our nonstationary PEF can also be applied to interpolate missing data. This follows directly from the previous work, adapting the filter as usual when passing over known data and incorporating some or all of the filter’s prediction in data gaps with some scale factor ϵ_Y . Here, unlike the stationary case, we can make multiple passes over the data and can choose different directions for different passes.

As the PEF adaptive update can and should be chosen to reduce its prediction error, we can expect a stable extrapolation into gaps with, most likely, a tail off in amplitude with distance from the edge of the hole. Figure 4.2 shows a simple example with a square hole in purely random data.

Figure 4.2: Infill of hole surrounded by random Gaussian numbers with $\epsilon_Y = \frac{1}{2}$ `sal/. PEFinterpBigepsd`



Oops!

No, this is *not* a bug. It is a consequence of the fact that the leading coefficient of any PEF has magnitude one, and, therefore, unless all other coefficients are zero, the PEF has a norm greater than one. Upon encountering a gap, the PEF can, indeed, produce geometrically increasing new values. Indeed, once such growth kicks in, the PEF will further adapt to better and better predict the geometric growth pattern.

To prevent this we rescaled our ϵ_Y by dividing it by the norm of the initial PEF filter we designed on the live data in the area of interest. This, too, managed to blow up! We finally realized that because the prediction error filter, A , was adapting as it entered the gaps, and adapting differently from iteration to iteration, we had to recalculate the filter norm at every step in order to ensure that our missing data updates were under control and stayed usefully incremental.

The situation for the filter adaptation parameter ϵ_A was less challenging because the guideline in Claerbout and Wang (2018, chap. 1.3) to use the reciprocal of the variance of the data, Y , to rescale it to a unitless value worked reasonably well. The value we settled on, $\epsilon_A = 0.05 / \sum d_i^2$, where d_i are the samples under the current filter, was a compromise between adapting painfully slowly and allowing the last orientation in an iteration to introduce a visible directional bias in the infill.

In addition to setting ϵ_Y and ϵ_A , we also could vary the number of interpolation passes over the grid. We settled on 64 eight-way passes, admittedly overkill, but sufficient to converge the infill all but the largest gaps.

Examples

To assist in Castillo’s thesis work (Castillo, 2018), our PEF interpolation was implemented under the ArcGIS framework using the ArcPy tool API. Details of this implementation may be found in SEP 173.

It is a fundamental property of prediction error filters that they are designed to track individual sinusoids and combinations of them, limited by just the number of PEF coefficients. Figures 4.3 and 4.4 are evidence that our stationary code is working. For both of these examples we specifically avoided cases where either the hole or the sinusoids serendipitously aligned with coordinate axes. In addition, we made sure that the criss-crossing sinusoids had distinctly different wavenumbers and slopes.

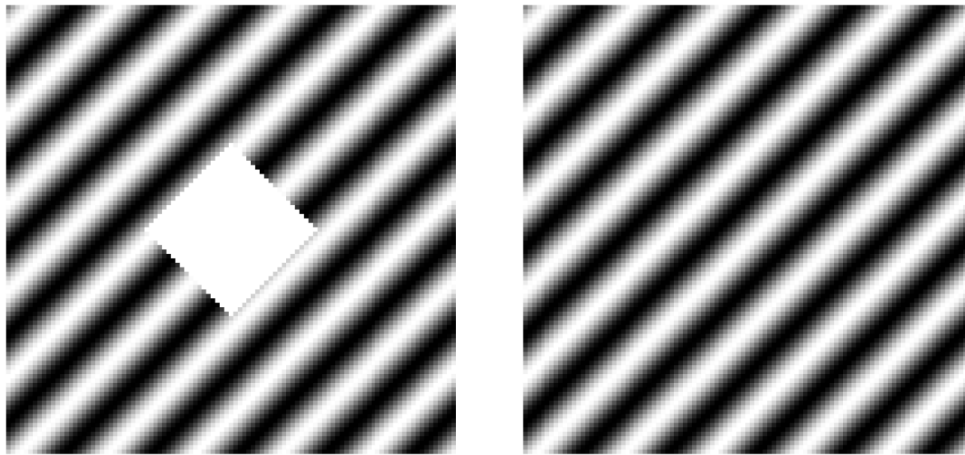


Figure 4.3: Test of stationary PEF missing data infill to ensure it reproduces input spatial sinusoid in a hole that is not aligned with x and y axes. `sal/. DiamondPair`

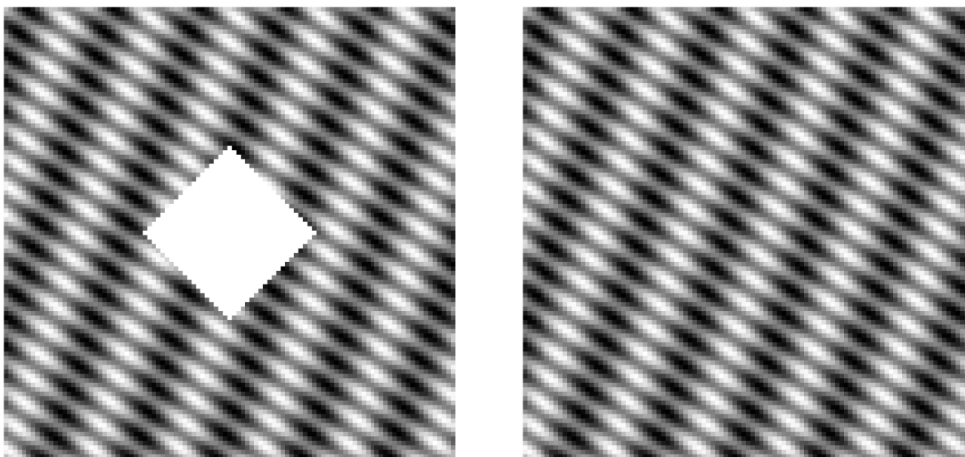


Figure 4.4: Test of stationary PEF missing data infill to ensure it reproduces two input spatial sinusoids with incommensurate orientations and wavenumbers. `sal/. CrissCross`

While Fourier theory says that even the most complex seafloor grid of bathymetry can be

perfectly reconstructed with a finite number of sinusoids, there is no reason to expect that a global Fourier reconstruction will infill data gaps in a useful way when, as is almost always the case, the statistics and pattern of the bathymetry change locally. This is illustrated in the challenging dataset of Figure 4.5. Our nonstationary PEF interpolation is highly realistic, exhibiting the characteristics of the data that our eye expects to see in the hole.

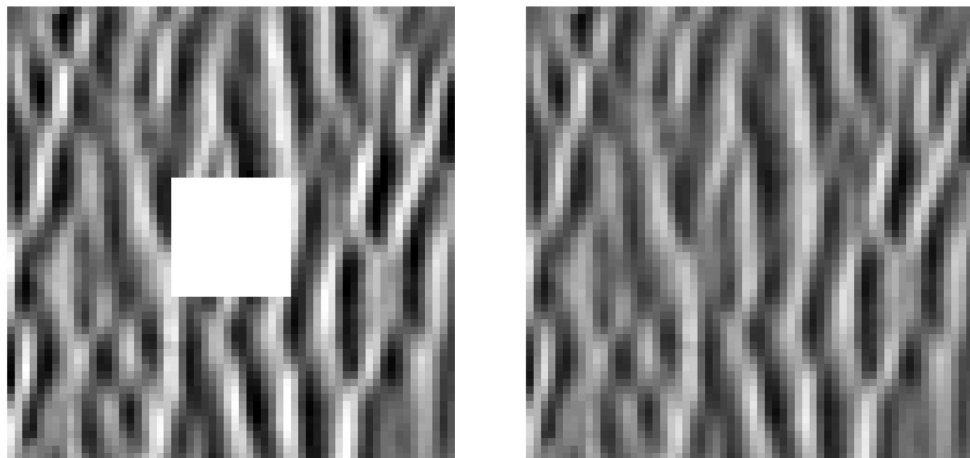


Figure 4.5: Portion of seafloor texture after detrending and removing a 16x16 square in the center alongside a nonstationary PEF missing data infill. `sal/. SeafloorTexture`

Finally, Figure 4.6 shows a large scale application of PEF interpolation to infill typical gaps in seafloor coverage that arise as the echo sounding vessel shifts and meanders as it moves along its nominal path.

Summary

We have implemented both stationary and nonstationary PEF missing data infill. We developed reasonable choices for the adjustable parameters in the nonstationary setting using a combination of theoretical reasoning and empirical trials. While further optimization can help reduce the time it takes to iterate to an answer, we would challenge anyone to find the location of the originally missing data in our results.

REFERENCES

- Castillo, C. M., 2018, Deep thoughts about the shallow subsurface: PhD thesis, Stanford University.
- Claerbout, J. and K. Wang, 2018, Data Fitting with Nonstationary Statistics: Lulu Press, www.lulu.com.

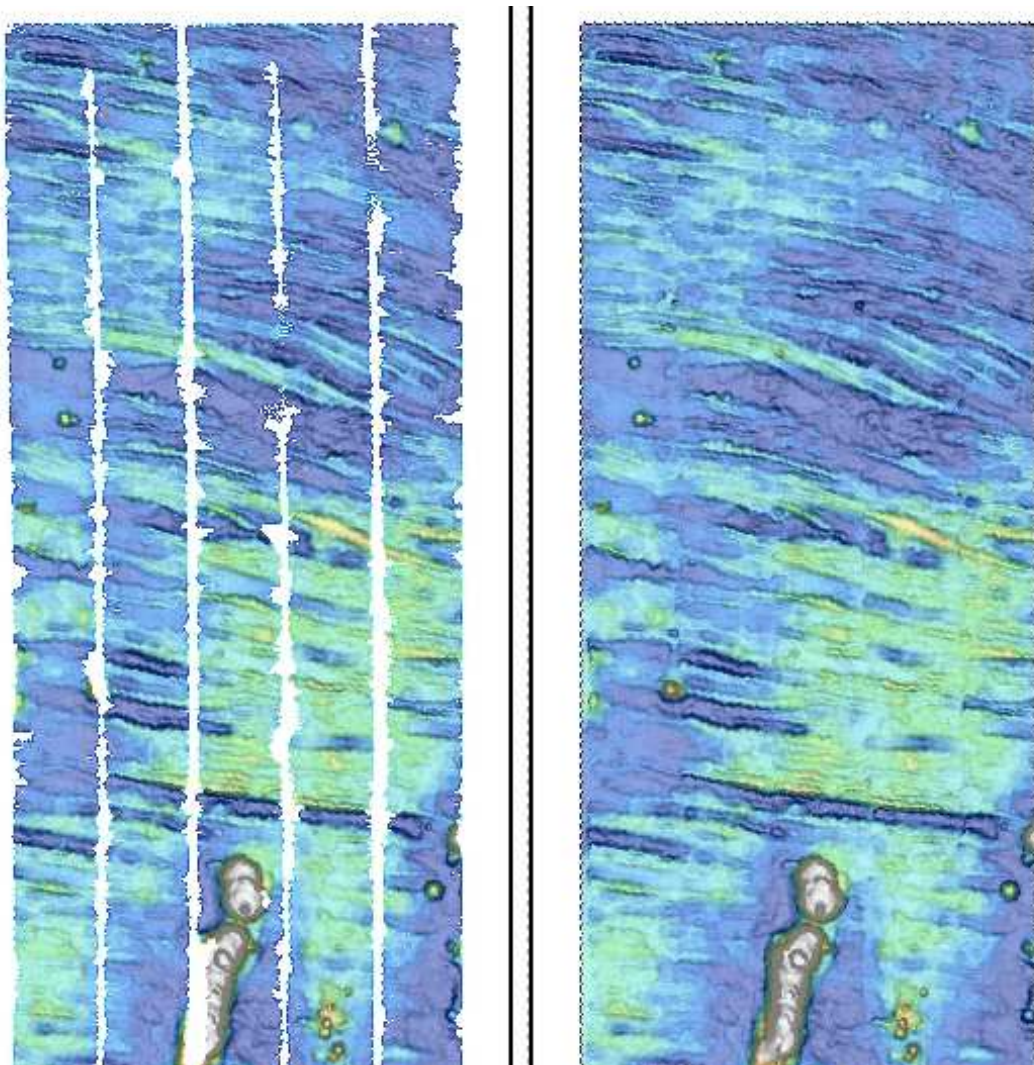


Figure 4.6: Track gap interpolation of a 500 m bathymetric grid collected in Western Pacific Basin interpolate using local PEFs. Courtesy C. M. Castillo (2018). [sal/. TrackInterp](#)

Chapter 5

Vector-valued signals

¹We have done much with PEFs on scalar-valued signals. Vector-valued signals are for 3-component seismographs and the like. The idea of deconvolution with a PEF extends to multicomponent signals. In ideal geometries, different wave types arrive on different channels; but in real life, wave types get mixed. Pressure waves tend to arrive on vertical seismographs, and shear waves arrive on horizontals; but, dipping waves corrupt each channel with the other. The main goal herein is to disentangle this channel crosstalk.

Scalar blind deconvolution is widely used in the seismic survey industry. The simple information flow in the upper quarter of Figure 5.1 is pretty much what we have done in Chapter 1 with the addition of the bandpass filter at the end. Oversimplifying, the idea is that Earth layers have random densities (impedances), therefore random echo polarities at a fine scale. This layering z_t gets smeared by the source wavelet, which is not an ideal impulse, instead being a mixture of air bubbles, ghosts, and weathered-layer reverberations leading to the observed output y_t . Those corrupting processes amount to causal filters, best undone with a PEF producing the output r_t . The bandpass filter at the end is there for subjective reasons, mainly we do not want to clutter our view with the highest possible frequency that a grid can hold because we know it is just noise. A popular alternative to the bandpass filter is gapping the PEF. Instead of limiting high frequencies, it does much the same by broadening the autocorrelation spike of the “white” output.

5.0.8 Multi channels = vector-valued signals

Widespread adaptation of multicomponent recorders leads to new opportunities indicated by the lower bulk of Figure 5.1. Hypothetical statistically independent channels z_1 and z_2 become colored making our ideal unpolluted channels x_1 and x_2 , which unfortunately “crosstalk” before giving us our observations y_1 and y_2 . Learning herein the theory of matrix valued PEFs, we design a matrix of filters, say $\mathbf{A} = a_{ij}$ attempting to achieve the original purity of \mathbf{z} . Normally, we do not wish to achieve the pure whiteness of \mathbf{z} . Rather than apply a bandpass filter herein, we use our estimates \hat{b}_{11} and \hat{b}_{22} to find $\hat{\mathbf{x}}$ as our attempt to restore the original colored signals \mathbf{x} .

Others may make other choices, but we are choosing to display $\hat{\mathbf{x}}$ for a reason. We want

¹This chapter draws from (Claerbout and Wang, 2017).

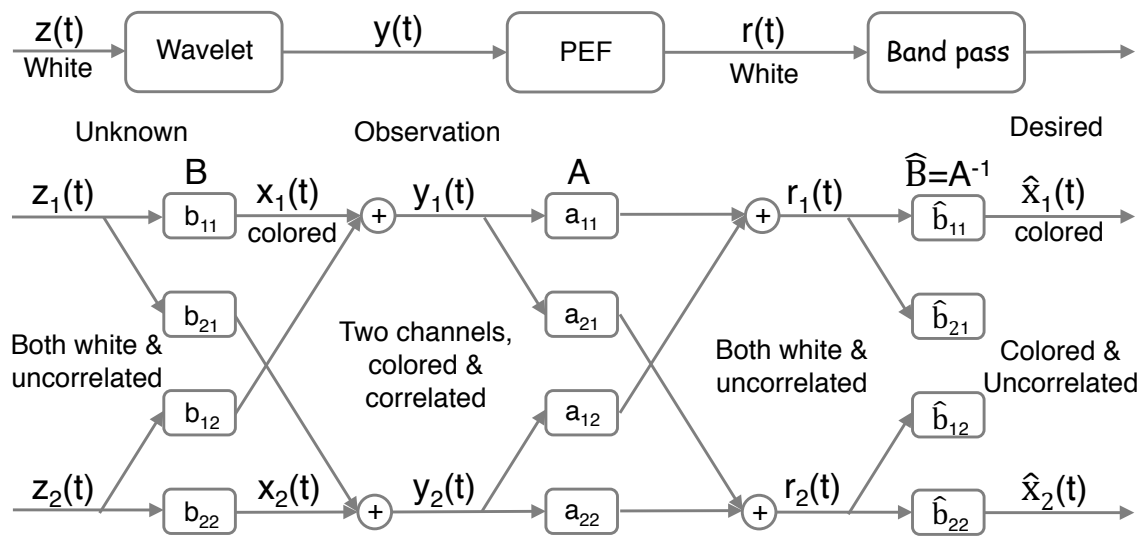


Figure 5.1: Top is scalar decon. Bottom is vector decon. In nature, two uncorrelated white random signals \mathbf{z} get colored thereby creating \mathbf{x} , which then gets mixed and creates our observations \mathbf{y} . Vector decon converts \mathbf{y} to uncorrelated white signals \mathbf{r} , which hopefully are a reasonable approximation to \mathbf{z} . If $\mathbf{r} \approx \mathbf{z}$, then $\mathbf{A}\mathbf{B} \approx \mathbf{I}$, therefore, recoloring \mathbf{r} without mixing gives us $\hat{\mathbf{x}}$, which should match the original colored signals \mathbf{x} . (Kaiwen Wang)

tests of whether or not our method works in practice. If it does, we can expect to see the S-wave channel coming out lower frequency than the P-wave channel, because the Earth acts as a wavelength filter. It is generally believed the Earth dissipates waves proportional to their spatial frequencies. Cutting both P and S at the same spatial frequency implies S cuts off at a lower temporal frequency than P because its velocity is lower. The scalar wave equation explains it $\omega^2 = v^2 k^2$.

The multichannel structure of Figure 5.1 arises in diverse physical settings. Not only does the Earth contain pressure waves and shear waves, where we measure vertical and horizontal motions, additionally, ocean bottom recordings contain pressure as well as three component velocity sensors. It is useful to extract upgoing from downgoing waves. Because pressure and velocity are sensed in different but overlapping frequency bands, the idea of b_{11} and b_{22} having different passbands is another valuable aspect of this model.

Fourier analysis suggests a crude approach to Figure 5.1. For scalar waves, given the spectrum $Y(\omega)^*Y(\omega)$, the solution to the problem is $A(\omega) = 1/\sqrt{Y(\omega)^*Y(\omega)}$. But, a symmetric function of frequency implies a symmetric function of time which is not causal. Fourier space requires stationary statistics, and forbids ℓ_1 -norm. The square root of a matrix of Fourier functions is easily found, but the disadvantages of Fourier space are overwhelmed by the simplicity of the time domain. Causality is easily expressed with Z -transforms, equivalently either as a matrix of polynomials or as a polynomial of matrix coefficients.

5.1 MULTI CHANNEL PEF

This mathematical model applies to one point in space, where it is based on causality and simultaneity of the two channels responding to the world around. The two-component signal model herein is not suitable for two scalar signals recorded at separate locations. At separate locations, there naturally would be time delays between the locations. If the underlying model \mathbf{B} were to introduce delay, its hypothetical inverse \mathbf{A} would need to contain inverse delay (anticausality!). Because \mathbf{A} , a PEF, is casual by construction, it cannot function anticausally. Whatever \mathbf{A} would come out of this process, it could not satisfy $\mathbf{BA} = \mathbf{I}$. In other words, there are many ways \mathbf{B} could contain delays without changing its covariance \mathbf{BB}^* . Our inverse operator \mathbf{A} is fundamentally based on \mathbf{BB}^* , which contains no phase. We get phase by insisting on causality for \mathbf{A} .

If you are processing a string of multicomponent recorders (e.g., down a well) each multicomponent recorder yields statistics that may be shared and averaged with neighboring recorders, but the signals themselves do not mix. The process described herein is simply a vector-valued, time variable linear operator. The same process could be independently applied to other channels.

Delay causes the method of this paper to fail in principle. In marginal cases (tiny delay) the notion of sparsity has helped for scalar signals (Claerbout and Guitton, 2013). There is an example in Chapter 1. Minuscule delays are a promising area beyond our present scope. Differential equations apply to a point in space. Their finite difference representations cover slightly more than a point. There may be some ticklish but promising aspects of merging finite difference operators with vector signals.

The multichannel model would seem to extend to three and more physical dimensions though we will never know until we try. Whether or not it is suitable for many channel market signals, I cannot predict.

5.1.1 Vector signal scaling

When components of data or model are out of scale with one another, bad things happen, such as the adjoint operator will not be a good approximation to the inverse, physical units may be contradictory, and the steepest descent method creep along slowly. These dangers would arise with vector-valued signals if the observations y_1 and y_2 had different physical units such as pressure and velocity recorded from up-going and down-going waves, or, uncalibrated vertical and horizontal seismograms.

We need to prepare ourselves for channels being out of scale with one another. Thus, we scale each component of data \mathbf{y} and residual \mathbf{r} by dividing out their variances. Recall that any component of a gradient may be scaled by any positive number. Such scaling is merely a change in coordinates.

With scalar signals, we updated using $\Delta \mathbf{a} = -(\epsilon r / \sigma_y^2) y_{t-\tau}$. With multiple channels, we are a bit more cautious and allow for data variance to differ from prediction-error variance. More importantly, the two components of \mathbf{y} might have differing physical units. Let $\sigma_{\mathbf{r}}$ be an estimate of the standard deviation of the prediction error in each channel. The following

code resembles this update

$$\Delta \mathbf{a} = - \left(\frac{\epsilon \mathbf{r}}{\sigma_{\mathbf{r}} \sigma_{\mathbf{y}}} \right) \mathbf{y}_{t-\tau} \quad (5.1)$$

Our original code contained leaky integrations for $\sigma_{\mathbf{y}}$ and $\sigma_{\mathbf{r}}$, but we had no vision of data to test that aspect. It also gave odd behavior when we adapted too rapidly. Because we had more pressing areas in which to direct our attention, the code exposition below simply replaces $\sigma_{\mathbf{y}}$ and $\sigma_{\mathbf{r}}$ by their global averages.

5.1.2 Pseudocode for vector signals

Compared with earlier pseudocode for scalar signals in which the gradient is a scaled adjoint, the gradient herein divides out the variances $\sigma_{\mathbf{r}}$ and $\sigma_{\mathbf{y}}$. That because we may always scale gradient components by positive numbers, say **sigy** and **sigr**. Look at the code below for the four **do** loops following **Happy streaming**. You see a matrix full of PEFs at work. The three loops next below the PEF filtering are simply its adjoint (allowing for the complication of the $\sigma_{\mathbf{r}}$ and $\sigma_{\mathbf{y}}$ scaling)—something you easily recognize by the interchange of inputs and outputs, **r** and **a**.

```
#                CODE = PREDICTION ERROR FOR VECTOR SIGNALS
#
integer it, nt=1000, tau, ntau=10, gap=0, ic, jc, nc=2
real y(nc,nt), r(nc,nt), aa(nc,nc,na), sige(nc), sigy(nc), eps
  e (*,*) = 0.
  aa(*,*,*) = 0.
do ic=1,nc {
  aa(ic,ic,0) = 1.          # Make a 2x2 identity matrix.
}
read input y(nc,nt)        # Read multichannel data.
#
do ic=1,nc {                # Initial variance estimates.
  sumsq=0
  do it=0,nt
    sumsq += y(ic,it)**2
  sigy(ic) = sqrt(sumsq/nt)
  sigr(ic) = sigy(ic)/2.
}
#                          Here we go! Happy streaming. Wheee!
do it= ntau, nt {
  do tau=1,ntau {           # lag axis.
    do ic =1,nc {           # Take a signal vector into a filter matrix.
      do jc =1,nc {        #
        r(ic,it) += aa(ic,jc,tau) * y(jc, it-tau)
      }}}
# Optionally update sigy and sige
  do tau=gap+1, ntau {     # adjoint = r * y' (outer product)
    do ic= 1, nc {         #
      do jc= 1, nc {       #
        aa(ic,jc,tau) -= eps * (r(ic,it)/sigr(ic)) * ( y(jc, it-tau) /sigy(jc))
      }}}
}
}
```

Now, it is easy to say that the code above is really quite trivial, but I breathed a sigh of relief when Kaiwen showed me the first results. (It worked on the first try!) Before I conceived the calculation as explained above, I had quite a struggle attempting the derivative of a quadratic form by a matrix filter, and even more doubts that I would be able to explain my analysis to other people, as well as a debt to Mohammed Hadidi, whose derivation showed that my derivative was the transpose of the correct one. Then I tried thinking carefully about Figure 5.1. But, it was better not to think at all; instead simply code the modeling, its adjoint, and stuff in the residual! Phew.

5.1.3 How the conjugate gradient method came to be oversold

Textbooks often illustrate the solution to a two component regression by comparing the steepest-descent method to the conjugate-gradient method. Conjugate gradient winningly obtains the exact solution on the second iteration while steepest descent plods along zig-zagging an infinite number of iterations. But, is this a fair comparison? Is it not true that axis stretching completely alters the picture? So, what exactly is the axis stretching that makes a more fair comparison? I suspect it is the kind of stretching done in the preceding code with variance divisors.

5.1.4 The PEF output is orthogonal to its inputs

Let us try to understand what this program has accomplished. If the program ran a long time in a stationary environment with a tiny ϵ `eps`, the filter \mathbf{A} , namely `aa(*,*,*)` would no longer be changing. The last line of the code would then say the residual `r(ic,it)` is orthogonal to the fitting functions `y(jc,it-tau+1)`. We would have a square matrix full of such statements. The fitting functions are all channel combinations of the shifted data. That is the main ingredient to Levin's whiteness proof for scalar signals in Chapter 7. I believe it means we can presume Levin's whiteness proof applies to vector signals. As we subsequently see, however, the situation at zero lag does bring up something new (Cholesky factorization).

5.1.5 Restoring source spectra

White signals are not ideal for display. Before corruption from channel 2, channel 1 had the spectrum of b_{11} . Consider restoring \mathbf{r}_1 to the original spectrum, namely b_{11} . Because $\mathbf{B} = \mathbf{A}^{-1}$, we can deduce b_{11} .

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \quad (5.2)$$

Under the assumption that the crossover filters are less significant than the pass-through filters, we may simplify the result for initial trials:

$$b_{11} = a_{22}/(a_{11}a_{22} - a_{21}a_{12}) \approx 1/a_{11} \quad (5.3)$$

$$b_{22} = a_{11}/(a_{11}a_{22} - a_{21}a_{12}) \approx 1/a_{22} \quad (5.4)$$

The result of polynomial division $\hat{x}(Z) = r(Z)/A(Z)$ is recognizable in the code by $\hat{\mathbf{x}}_t = \text{xhat}(\text{ichan}, t)$. Here is the polynomial division code fragment:

```
#                               CODE = POLYNOMIAL DIVISION
      xhat(1,t) = r(1,t)
do tau=1,ntau                    # xhat1(Z) = r1(Z)/a11(Z)
      xhat(1,t) -= aa(1,1,tau) * xhat(1,t-tau)

      xhat(2,t) = r(2,t)
do tau=1,ntau                    # xhat2(Z) = r2(Z)/a22(Z)
      xhat(2,t) -= aa(2,2,tau) * xhat(2,t-tau)
}
```

We have been doing this polynomial division for some time with no stability issues yet.

5.2 CHOLESKY DECORRELATING AND SCALING

The two independent channels of unit-variance random numbers in \mathbf{r} *entering* filter \mathbf{B} in Figure 5.1 have the identity matrix \mathbf{I} as a covariance. Herein we arrange to have the same identity covariance for the values \mathbf{r} *exiting* from \mathbf{A} on the right.

By construction, the multicomponent PEF output chews up nonzero lagged correlations within and among channels. By construction, it does not chew up correlations among channels at zero lag. With two components we are left at the zero lag with a nice 2×2 matrix of prediction-error variances \mathbf{W} .

$$\mathbf{W}(\tau = 0) = \begin{bmatrix} \sigma_{r_{11}}^2 & \sigma_{r_{12}}^2 \\ \sigma_{r_{21}}^2 & \sigma_{r_{22}}^2 \end{bmatrix} \approx \begin{bmatrix} (\mathbf{r}_1 \cdot \mathbf{r}_1) & (\mathbf{r}_1 \cdot \mathbf{r}_2) \\ (\mathbf{r}_2 \cdot \mathbf{r}_1) & (\mathbf{r}_2 \cdot \mathbf{r}_2) \end{bmatrix} \quad (5.5)$$

Consider the expectation (leaky sum over time) $E[\mathbf{r}\mathbf{r}^*]$. Theoretically it is a three component (3-C) function of lag and the two channels. We are going to assume our PEFs do their job, so, it is no longer a function of lag. Thus, we presume that $E[\mathbf{r}\mathbf{r}^*]$ is like the $\mathbf{W}(\tau = 0)$ we computed with Equation (5.5) at zero lag τ .

Use the Cholesky method to factor \mathbf{W} into a triangular matrix \mathbf{V} times its transpose. We express this as: $\mathbf{W} = \mathbf{V}\mathbf{V}^*$. (The Cholesky method is nearly trivial: [1] write a triangular matrix of unknown elements, [2] multiply it by its transpose, and [3] notice a sequential method that unravels the unknown elements.) Starting from $\mathbf{W} = \mathbf{V}\mathbf{V}^*$ we have:

$$\mathbf{W} = \mathbf{V}\mathbf{V}^* \quad (5.6)$$

$$\mathbf{V}^{-1}\mathbf{W}(\mathbf{V}^*)^{-1} = \mathbf{I} \quad (5.7)$$

$$\mathbf{C}\mathbf{W}\mathbf{C}^* = \mathbf{I} \quad (5.8)$$

where we have defined $\mathbf{C} = \mathbf{V}^{-1}$. Using this new matrix operator \mathbf{C} we get a new vector signal \mathbf{q} .

$$\mathbf{q} = \mathbf{C}\mathbf{r} \quad (5.9)$$

Using Equation 5.8 the expectation of this new variable \mathbf{q} is as follows:

$$E[\mathbf{q}\mathbf{q}^*] = E[\mathbf{C}\mathbf{r}\mathbf{r}^*\mathbf{C}^*] = \mathbf{C}E[\mathbf{r}\mathbf{r}^*]\mathbf{C}^* = \mathbf{C}\mathbf{W}\mathbf{C}^* = \mathbf{I} \quad (5.10)$$

This proves Cholesky meets our goals: (1) it descales, and (2) it decorrelates \mathbf{r} at zero lag.

5.3 ROTATING FOR SPARSITY

Intrigue is what comes last, something wholly unfamiliar. As the universe marches on, things get mixed and entropy increases. We seek the opposite. Even after solving the problem posed in Figure 5.1, the solution is unique only within an arbitrary unitary matrix. (With scalar signals the arbitrariness is in a scale factor $e^{i\phi}$.) We get to choose the unitary matrix \mathbf{U} having minimum entropy \mathbf{r} output. Luckily, this two-channel problem, although nonlinear, is easily amenable to a one-parameter exhaustive search. That search can be done to maximize sparsity of the final signals. We humans love the simplest representation of our data. This should be it. Hooray!

Rotations and reflections are called “unitary operators.” For now, we are ignoring reflections (polarity changes). (Consider that to be an application labeling issue.) Scanning a single parameter θ through all angles allows us to choose the one with the most sparsity (least clutter). A general form for a 2×2 rotation operator is

$$\mathbf{U} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \quad (5.11)$$

We will meet our goal of finding \mathbf{A} and \mathbf{r} of Figure 5.1 with the following:

$$\mathbf{r} = \mathbf{U}\mathbf{q} = \mathbf{U}\mathbf{C}\mathbf{r} = \mathbf{U}\mathbf{C}\mathbf{E}\mathbf{y} = \mathbf{A}\mathbf{y} \quad (5.12)$$

A unitary operator \mathbf{U} does not change the length of any vector. It satisfies $\mathbf{U}^*\mathbf{U} = \mathbf{I}$, therefore for any \mathbf{v} we see $(\mathbf{U}\mathbf{v})^*\mathbf{U}\mathbf{v} = \mathbf{v}^*\mathbf{U}^*\mathbf{U}\mathbf{v} = \mathbf{v}^*\mathbf{v}$. Let us check that the covariance of $\mathbf{r} = \mathbf{U}\mathbf{q}$ is constant independent of θ . Equation (5.10) leads to $\mathbf{r}\mathbf{r}^* = \mathbf{U}\mathbf{E}[\mathbf{q}\mathbf{q}^*]\mathbf{U}^* = \mathbf{U}\mathbf{I}\mathbf{U} = \mathbf{I}$, which says the energy stays constant as we sweep through θ .

5.3.1 Finding the angle of maximum sparsity (minimum entropy)

Given any angle θ for Equation (5.11), we have $\mathbf{r} = \mathbf{U}\mathbf{q}$. We can scan θ over one degree increments. Defining the entropy at any particular time as $(|r_1| + |r_2|)/\sqrt{r_1^2 + r_2^2}$, we easily choose the angle of minimum entropy for that time. We may define the entropy for the entire time range of the signal as follows:

$$\text{Entropy}(\theta) = \frac{\sum_t^\infty (|r_1(t)| + |r_2(t)|)}{\sqrt{\sum_t^\infty (r_1^2(t) + r_2^2(t))}} \quad (5.13)$$

Because the denominator should be a constant function of θ , we may as well define entropy simply by the numerator $\text{Entropy}(\theta) = \sum_t^\infty (|r_1(t)| + |r_2(t)|)$.

Retrospectively, the authors have come to understand that the unitary operator \mathbf{U} is not only a mathematical tool, but, it also models rotation in the physical world. It should be done at beginning of the process (as well as again at the end) because it often has the power to diagonalize the matrices right at the beginning.

Why the scan works

Why does this \mathbf{U} process of scanning θ lead to sparsity? Suppose the vector signal element \mathbf{q}_N at time at $t = N$ has all its energy in its first component. Say the vector signal is

$[-1, 0]^*$ with energy and magnitude both now equal unity. The rotated signal is now as follows:

$$\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\cos \theta \\ \sin \theta \end{bmatrix} \quad (5.14)$$

Let the rotation angle be 45° so sine and cosine are both $1/\sqrt{2}$. The sum of the magnitudes becomes $2/\sqrt{2} = \sqrt{2} > 1$. As expected the rotation took away the original sparsity.

We experimented with taking the matrix \mathbf{U} to be time variable. That has pitfalls we are not yet prepared to explain.

5.3.2 3-component vector data

For 3-component vectors, the scan would run over two angles; therefore the `u(itheta)` would be expanded to `u(itheta, iphi)`.

5.3.3 Channel order and polarity

Although our first synthetic data had the strongest pressure wave on the first channel, our first successful run yielded the pressure wave on the second channel. The channel flip operation is as follows:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (5.15)$$

Now, we flip channels when we find the expression $|\mathbf{r}_1 \cdot \mathbf{y}_1| + |\mathbf{r}_2 \cdot \mathbf{y}_2| < |\mathbf{r}_1 \cdot \mathbf{y}_2| + |\mathbf{r}_2 \cdot \mathbf{y}_1|$.

Our initial P-wave result had a flipped polarity. The operation for flipping the polarity for Channel 1 is as follows:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad (5.16)$$

We change the polarity of Channel 1 when $(\mathbf{y}_1 \cdot \mathbf{r}_1) < 0$ and likewise for Channel 2.

It is easy to show for signals with an identity \mathbf{I} correlation matrix, that channel flip and polarity change operations do not change the \mathbf{I} correlation matrix. It is easy to imagine situations in which flip and polarity should change with time. For example, there may be more than two wave types present. One may die out, while another grows. We have not yet synthesized such data for testing and are unclear how we might proceed. We will, no doubt, be strongly influenced by the data at hand.

5.4 RESULTS OF KAIWEN WANG

Figure 5.2 is our first test data, synthetic data with a vertical component and a horizontal component. Both a P wave and an S wave are emerging at a fairly steep angle; so the vertical is mostly a P is corrupted by a little S, while on the horizontal it is the opposite.

On Figure 5.3, we notice that the spike estimates become sharper and sharper with time as the filter \mathbf{A} adapts with time. Oddly, there is some crosstalk on the P channel that does

Figure 5.2: Synthetic data input is vertical and horizontal components. Model is a mix of sharp, unipolar P waves and S waves of lower frequency with alternating polarity. Stronger P waves on the vertical, and stronger S waves on the horizontal. (Kaiwen Wang) `vector/. y-cropped`

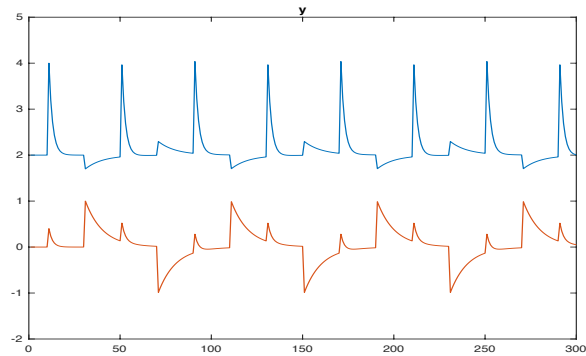
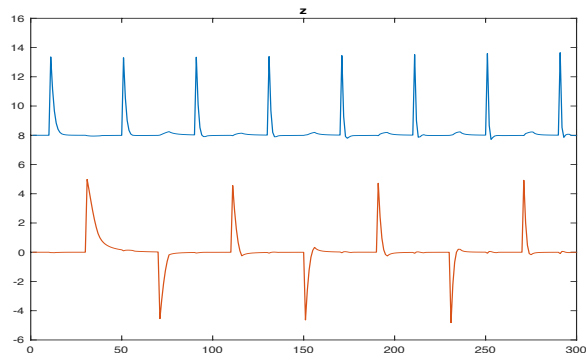


Figure 5.3: Output results: Deconvolved P wave on vertical component (top), S on horizontal (bottom). Spiking improves with time. (Kaiwen Wang) `vector/. z-cropped`



not seem to be diminishing with time. I do not know why that is. Perhaps, we should repeatedly run the program over the panel.

On Figure 5.4, the P and S channels contain two signals—the original spikes and their estimates. We notice that crosstalk nearly diminishes to zero on the P channel, likewise on the S channel.

Figure 5.5 is like Figure 5.4 but with denser spikes—a spike every 4 pixels, each spike topped by a small circle. Vertical lines primarily connect to the dots. Ideally, between the dots are vertical lines of zero height, the nonzero height exhibiting the limitations of the overall process.

Notice the vertical trace (top in upper panel) being dominated by P waves is a higher frequency than the horizontal trace “H” (top in lower panel) which is dominated by S waves. Results are about the same quality as Figure 5.4—proving that having so much wavelet overlap creates no real problems. Fitting on the S channel (bottom in lower panel) gets much better with time. Fitting on the P channel is so good near the beginning that we hardly notice improvement with time.

REFERENCES

- Claerbout, J. and A. Guitton, 2013, Ricker-compliant deconvolution: SEP-Report, **150**, 1–12.
- Claerbout, J. and K. Wang, 2017, Multichannel data: separating independent causes : SEP-Report, **170**, 189–206.

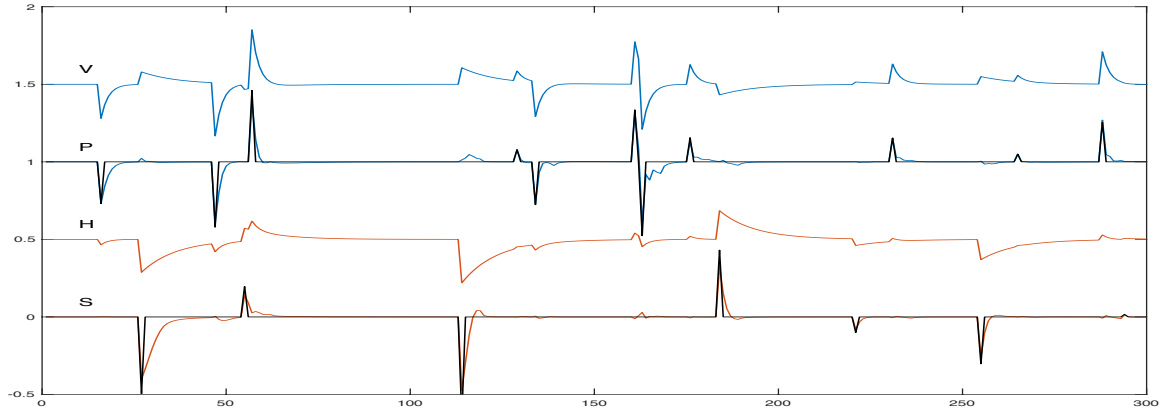


Figure 5.4: V=vertical, H=horizontal. The traces P and S are overlays of the original impulsive waves and their attempted reconstruction from (V,H). The pulses get sharper with time as the PEFs adapt. (Kaiwen Wang) `vector/. tracesOrdered-cropped`

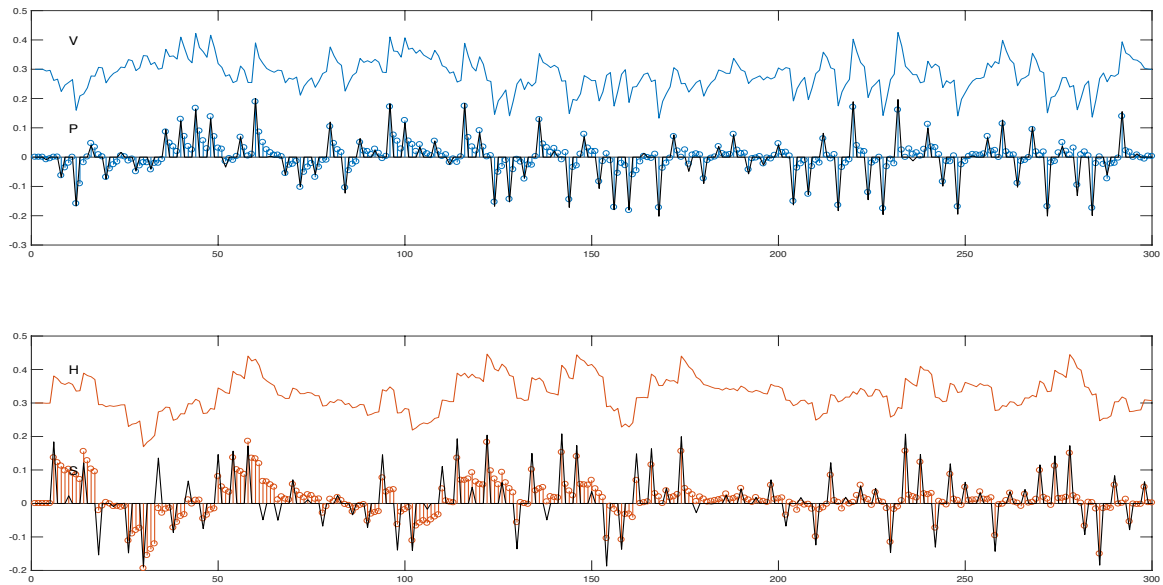


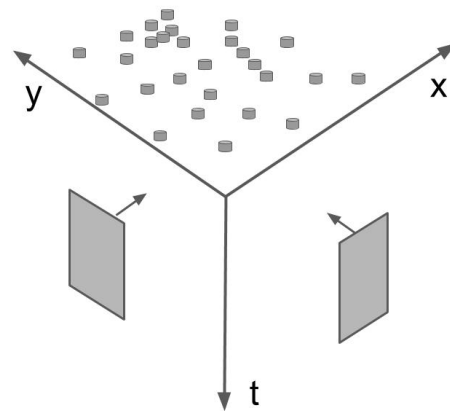
Figure 5.5: The top panel refers to the vertical motions V and the pressure waves P. The second signal in that panel is a superposition of the sparse original impulses (tiny circles) that made the data and the pulses as estimated by the entire process. These should match. They mostly do match, but small nonzero values appear between the dots. The lower panel is likewise for the horizontal H seismograph and the S wave (Kaiwen Wang) `vector/. denseSpikes-cropped`

Chapter 6

Inverse interpolation

Figure 6.1 illustrates a **universal problem in Geophysics** and in many other fields. We wish a dense uniform grid on the Earth surface from which linear interpolation would give our raw data found sprinkled on the surface. (Reflection seismology using physics and math explains the transformation $t \rightarrow z$).

Figure 6.1: Please, pretty please, build me a dense uniform grid on the Earth surface (x, y) plane. From that grid, I want to draw by interpolation my observed data sprinkled in the (x, y) plane. Those two gray boxes must be magic 2-D PEFs. They sweep through the entire volume, updating themselves as they go. [uniform/. WorldOfSignals5](#)



To achieve IID estimation, we can always use PEFs on *model space* (since we define it), but we often wish likewise for *data space* where PEFs could fill data gaps. Our goal here is to make pseudo data on a uniform grid from the real data sprinkled about. Since this is an inversion problem, the pseudo data is the model space. The model \mathbf{m} is located at $x_i = x_0 + i \Delta x$, namely $\mathbf{x} = \mathbf{x}_0 + i\mathbf{x} \cdot \mathbf{dx}$. Components of the observed signal data \mathbf{d} each have with them a location \mathbf{x}_d , namely $\mathbf{xx}(\text{id})$ —likewise for 2-D space (\mathbf{x}, \mathbf{y}). Generally, the pseudo data \mathbf{m} is gridded into somewhat more locations than the real data \mathbf{d} so regularization is essential.

The 1-D linear operator \mathbf{L} is defined by the following code. (2-D is similar.) Code elements \mathbf{dd} and \mathbf{mm} are 1-D arrays of signals.

```
#                               CODE = LINEAR INTERPOLATION OF 1-D SIGNALS
integer 0 <= d <= nd             # nd data signals
integer 0 <= m <= nm             # nm grid locations
real mm(m)                       # components of mm are signals on a uniform grid
```

```

real dd(d)           # components of dd are signals, recorded data.
real xx(d)           # locations of dd signal raw data recordings.
real ox, dx          # origin and delta of x grid coordinates.

do d = 0, nd          # Data scan over all recorded signals.
  x = (xx(d)-x0)/dx  # the value x points within the grid.
  ix = integer(x)    # the value ix points to a place on the grid.
  if ( 0<ix< nm-1 ) # include only data signals inside the grid
    f = x-ix         # 0<f<1. closeness to ix+1
    g = 1.-f         # 0<g<1. closeness to ix      f+g=1.
    do t = 0, nt     # Both dd and mm are functions of time.
      if forward
        dd(d)      += ( g * mm(ix) + f * mm(ix+1))
      else adjoint
        mm(ix)     += g * dd(d)
        mm(ix+1)   += f * dd(d)
    end do
  end do

```

Geophysics requires data, most often acquired on land (although also often at sea or in space). On land it is often difficult or impossible to acquire data on a regular grid, because we have limited access to land. But, mathematical algorithms are normally expressed in a form requiring a regular grid. And, PEFs require a uniform Cartesian grid. And more, PEFs are the only easy, large scale method of achieving IID. (Singular-value decomposition is much slower, suitable only for much smaller problems.) Resolving the data/theory grid conflict requires a process to synthesize pseudo data on a regular grid, from the given signals on a non regular grid. Such processes are a class of “inverse problems.”

6.0.1 Sprinkled signals go to a uniform grid via PEFed residuals

Sprinkled signals \mathbf{d} means at arbitrary (x_i, y_i) lies your i^{th} signal $\mathbf{d} = d_{i,t}$. Herein we make synthetic signals $\mathbf{m} = m_t(x_0 + j\Delta x, y_0 + k\Delta y)$. The algorithm for building \mathbf{m} is the following:

- 1: **Background**
- 2: $\mathbf{m}_1 = \text{Random}$ trial model
- 3: $\mathbf{d}_1 = \mathbf{L}\mathbf{m}_1$ trial data
- 4: $\mathbf{m}_2 = \mathbf{L}^* \mathbf{d}_1$ implied model
- 5: $\mathbf{r} = \mathbf{m}_1 - \alpha \mathbf{m}_2$ model residual, but α unknown
- 6: $0 = d(\mathbf{r} \cdot \mathbf{r})/d\alpha$
- 7: $0 = \mathbf{m}_2 \cdot (\mathbf{m}_1 - \alpha \mathbf{m}_2)$
- 8: $\alpha = (\mathbf{m}_2 \cdot \mathbf{m}_1)/(\mathbf{m}_2 \cdot \mathbf{m}_2)$ α is now a known property of \mathbf{L} .
- 9: **Iteration**
- 10: $\mathbf{r} = \mathbf{d} - \mathbf{L}\mathbf{m}$ residual update rule
- 11: $\mathbf{m} \leftarrow \mathbf{m} + \epsilon_d \alpha \mathbf{L}^* \mathbf{r}$ use data to expand the model (fitting)
- 12: $\mathbf{m} \leftarrow \mathbf{m} - \epsilon_m \mathbf{A}\mathbf{m}$ use PEF to shrink the model (regularizing)

Regularization being the flat-earth model

We could debug code starting signals of a single time value, so $\mathbf{m} = m$. To see energy spreading out from a signal to surrounding model space locations, take the PEF \mathbf{A} to be simply the space derivative $d\mathbf{m}/dx$. We may call $d\mathbf{m}/dx \approx 0$ the “flat-earth” fitting goal.

$$\mathbf{r} \leftarrow \mathbf{d} - \mathbf{L}\mathbf{m} \quad (6.1)$$

$$\mathbf{m} \leftarrow \mathbf{m} + \epsilon_d \alpha \mathbf{L}^* \mathbf{r} - \epsilon_m \frac{d}{dx} \mathbf{m} \quad (6.2)$$

To simplify testing codes, we may use signals each consisting of a single scalar value.

Once the preceding code works on scalar valued signals, we can upgrade to signal duration longer than a single scalar value. Signals would be placed somewhat randomly along a 1-D line on the earth surface. The test data \mathbf{d} might be dipping layers. Some layers would be thin (short on the time axis) others fat; some steeply dipping, some gentle. On \mathbf{m} space, fat gentle bedding should interpolate smoothly and continuously in space. Thin steep bedding would break up into a series of fattened dots.

Learning the PEF while using it to interpolate

Going beyond the flat-earth assumption, let us interpolate a seismic receiver line. The wave arrival slope changes with time and space. Remember from page 16 that 2-D PEFs can kill linear events like wavefronts. Waves of differing slopes and differing frequencies often arrive at the same time. We need local PEFs to handle these complications occurring all together. Think of this:

$$\mathbf{r} \leftarrow \mathbf{d} - \mathbf{L}\mathbf{m} \quad (6.3)$$

$$\mathbf{m} \leftarrow \mathbf{m} + \epsilon_d \alpha \mathbf{L}^* \mathbf{r} - \epsilon_m \mathbf{A}\mathbf{m} \quad (6.4)$$

Consider the effect of the two terms, $\mathbf{L}^*\mathbf{r}$ and $\mathbf{A}\mathbf{m}$. First, \mathbf{r} is the raw data minus the data our model predicts. If our model \mathbf{m} is too weak, its predicted data will be too weak, so the term $\mathbf{L}^*\mathbf{r}$ will push more raw data into \mathbf{m} . While the ϵ_d term adds essentials to the model, the ϵ_m term cuts back some “bad” spectrum from the model—here is how: The PEF \mathbf{A} has removed the dominant spectrum, the good, from \mathbf{m} , so what comes out of $\mathbf{A}\mathbf{m}$ is its bad spectrum, that to be subtracted. (This term also obligates us to the side project estimating the PEF \mathbf{A} from \mathbf{m} .)

I suggest the PEF estimation be done in a subroutine where its residual \mathbf{r} is kept internal, so not to be confused with the present residual \mathbf{r} going into \mathbf{L}^* .

We may wish the PEF \mathbf{A} be derived in the dilation invariant manner of page 20.

Manufacturing super-resolution does not work, but we can go far.

Mathematically, the pulling apart of the product $\mathbf{A}\mathbf{m}$ is a nonlinear activity, therefore it is susceptible to multiple solutions. That happens with too fine a grid. An attractive always-available starting solution is defining an initial \mathbf{m} on a coarse grid, and interpolating that.

We cannot build spatial resolution that is not in the data, however, the tacit assumption that we envision the world being made up of planes (because our physics gives us plane waves) has saved us from needing a 3-D PEF. This leads to some magic: Without going into a lengthy discussion, in reflection seismology we often encounter very slow waves (ground roll) that are adequately sampled on the time axis, but inadequately sampled on a distance axis. Never-the-less, after we nail down the velocity (slope), the space axis comes easily from the neighboring time axis. Good understanding of one dimension is valuable, but not fully adequate to understand higher dimensional spaces.

We give up on recursion because our gaps are small.

Take data organized somewhat like the model space, but with a substantial gap of missing signals in it. Enough iterations of (6.3)-(6.4) should eventually fill the gap, albeit somewhat tediously. Stationary theory has a seductive method of filling long gaps commonly known as recursion or polynomial division. This method is fast for covering long gaps, such as at cable ends. But in most applications, we have more modest goals, such as data sampling irregularities and gaps the size of streamer separations. Moreover, the speed of the method herein might render itself irrelevant, even on larger gaps. Do not give much credence to synthetic data far from real data. My dear old Russian friend Boris would say, “Do not trust data what you have not paid for.”

3-D flat-earth regularization

For 3-D data, an (x, y) -plane of signals, we penalize slopes in both x and y with the following iteration:

$$\mathbf{m} \leftarrow \mathbf{m} + \epsilon_d \alpha \mathbf{L}^* \mathbf{r} - \epsilon_m \begin{bmatrix} \frac{d}{dx} & \frac{d}{dy} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix} \mathbf{m} \quad (6.5)$$

This fills holes with the 3-D flat-earth model.

3-D locally constant dip regularization

For the first time now, we do that which is not easy to do by any other method. Use two 2-D PEFs, \mathbf{A} and \mathbf{B} , one for the (t, x) -plane, the other for the (t, y) -plane. In principle, a 10×2 PEF in the (t, x) plane, likewise for the (t, y) -plane, adapts to dipping planes. In practice, 10×3 might work better. This and longer filters on the space axes allow for several plane wave angles appearing simultaneously on the time axis. The fitting iterations are:

$$\mathbf{m} \leftarrow \mathbf{m} + \epsilon_d \alpha \mathbf{L}^* \mathbf{r} - \epsilon_m [\mathbf{A} \ \mathbf{B}] \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix} \mathbf{m} \quad (6.6)$$

We have not discussed the double PEF estimation algorithm necessary in this circumstance. Well, I need to leave some fun examples for my readers to map out. To get started, recall Figure 6.1.

Seismologists (they work to map $t \rightarrow z$) who have lived for years in (x, t) space, upon arriving in (x, y, t) space find themselves in awe at how much different the world feels.

Without me speculating more on why, (which I easily could), I feel users of Equation (6.6) will be amazed when they first encounter results of Equation (6.6). Compare a solitary picture on your retina, to a radiologist swimming throughout your body with a PET scan. She can glide anywhere she likes, all the while viewing down, forward, and sideways.

6.1 REPAIRING THE NAVIGATION

Occasionally, data location is inaccurate. Historically, we have often seen that. Today navigation is usually quite good, but not universally so. Multicomponent seismometers along with a pressure gauge are called “nodes.” Nodes may be placed on the ocean bottom with a *Remote Operated Vehicle* (ROV), or alternately with a manned underwater vehicle. The surface boat knows its location well enough, but it may not be very certain where the node is. I’m willing to work on this problem, but not until after I find colleagues to work on it with me.

The Galilee data set in *GIEE* is an example of data that gave me good reason to doubt the navigation. But, it is 1990 vintage data with pre-satellite navigation.

6.2 IMAGING

The bulk of the work at SEP concerns imaging. Imaging problems are also regressions, as is the main theme of this booklet. Thus in principle imaging problems are amenable to the descent methodology of this booklet. First impressions are that even easy trial applications such as time migration and velocity analysis, might be too slow using the technology of this booklet. The IID aspect is not slow, but slowness might result from the huge number of locations in model space. Perhaps we should contemplate highly aliased model spaces? Modern computer technology offers extreme parallelism. What applications using our “gradient hopping” might benefit from such parallelism? These topics are suitable for Friday afternoon casual discussions (with beer). Something of value might be learned from simple test cases.

In a world consisting of (1) knowns, (2) known unknowns, and (3) unknown unknowns, imaging by gradient hopping is among the unknown unknowns. If any place remains for breakthroughs in the seemingly mature field of seismic imaging, we should ramble among the unknown unknowns.

6.3 DAYDREAMS

I like to daydream about equation (6.6) and its relationship to the land surface of the USA. Many kinds of geophysical recorders lay sparse and irregular on the ground, so the factor $(\mathbf{d} - \mathbf{F}\mathbf{m})$ seems central to our efforts. Of course we need to flatten the Earth sphere leading us to wonder whether PEF concepts are limited to Cartesian spaces. The land surface \mathbf{m} is somewhat smooth in the plains whereas rough in the mountains. Where \mathbf{m} in the plains is very smooth, there \mathbf{A} must turn out to be a powerful roughener. There can be the occasional sharply defined texture in the plains, so we will want *softclip*($\mathbf{A}\mathbf{m}$) in the plains as much as in the mountains.

Have a look with Google Earth or satellite maps. In the Appalachians there is a pattern to the mountains not found in the Rockies. Follow the track from Harrisburg, Pennsylvania to Birmingham Alabama. Occasionally these rolling mountains are broken through by rivers. After the land, look at the bottom of the oceans.

Ocean bottoms are tough places to get data. Many kinds of data (and data gaps!) affect the images we are able to see of the ocean floor. Given Google Maps, even the casual viewer should recognize many data acquisition limitations. Everywhere there are stories to be told, half geological, and half about data acquisition limitations. Awesome! Let your imagination run.

Chapter 7

Appendices

7.1 WHY PEFs HAVE WHITE OUTPUT

It is somewhat intuitive that 1-D PEFs have a white output, but it is really amazing that 2-D PEFs tend to spectral whiteness in a 2-D space; yet, this whiteness is extensively demonstrated in *GIEE* (Claerbout, 2014), while herein it is simply introduced and has its whiteness proven. Consequently (shown here), on a cartesian grid the PEF times its adjoint plays the role of the inverse covariance matrix demanded by inverse theory.

7.1.1 Why 1-D PEFs have white output

¹The basic idea of least-squares fitting is that the residual is orthogonal to each of the fitting functions. Applied to the PEF, this idea means the output of the PEF is orthogonal to lagged inputs. The orthogonality applies only for lags in the past, because prediction knows only the past while it aims to the future. What we soon see herein is different; namely, the output is uncorrelated with *itself* (as opposed to the input) for lags in *both* directions; therefore, the autocorrelation of the output is a delta function and the output spectrum is white. Knowing the PEF and having output whiteness has many applications.

Let \mathbf{d} be a vector with components containing a time function. Let $Z^n\mathbf{d}$ represent shifting the components to delay the signal in \mathbf{d} by n samples. The definition of a PEF is that it minimizes $\|\mathbf{r}\|$ by adjusting filter coefficients a_k . The PEF output is as follows:

$$\mathbf{r} = \mathbf{d} + a_1 Z^1 \mathbf{d} + a_2 Z^2 \mathbf{d} + a_3 Z^3 \mathbf{d} + \dots \quad (7.1)$$

We set out to choose the best a_k by setting to zero the derivative of $(\mathbf{r} \cdot \mathbf{r})$ by a_k . After the best a_k are chosen, the residual is perpendicular to each of the fitting functions as follows:

$$0 = \frac{d}{da_k} (\mathbf{r} \cdot \mathbf{r}) \quad (7.2)$$

$$0 = \mathbf{r} \cdot \frac{d\mathbf{r}}{da_k} = \mathbf{r} \cdot Z^k \mathbf{d} \quad \text{for } k > 0. \quad (7.3)$$

¹This subsection draws from Levin et al. (2013), and is also included in Claerbout (2014).

Given that $0 = \mathbf{r} \cdot Z^k \mathbf{d}$, we examine $\mathbf{r} \cdot Z^k \mathbf{r}$ and see that it also vanishes. Using Equation (7.1), we have for any autocorrelation lag $k > 0$,

$$\begin{aligned} \mathbf{r} \cdot Z^k \mathbf{r} &= \mathbf{r} \cdot (Z^k \mathbf{d} + a_1 Z^{k+1} \mathbf{d} + a_2 Z^{k+2} \mathbf{d} + \dots) \\ &= \mathbf{r} \cdot Z^k \mathbf{d} + a_1 \mathbf{r} \cdot Z^{k+1} \mathbf{d} + a_2 \mathbf{r} \cdot Z^{k+2} \mathbf{d} + \dots \\ &= 0 + a_1 0 + a_2 0 + \dots \\ &= 0. \end{aligned}$$

Because the autocorrelation is symmetric, $\mathbf{r} \cdot Z^{-k} \mathbf{r}$ is also zero for $k < 0$; therefore, the autocorrelation of \mathbf{r} is an impulse. In other words, the spectrum of the time function r_t is white. Thus, \mathbf{d} and \mathbf{a} have mutually inverse spectra. Because the output of a PEF is white, the PEF itself has a spectrum inverse to its input.

7.1.2 The PEF gives the inverse covariance matrix.

Put any residual \mathbf{r} into its PEF \mathbf{A} and receive a white output \mathbf{Ar} .

$$\begin{aligned} \mathbf{I} &= \mathbf{E}[\mathbf{Ar}(\mathbf{Ar})^*] && \text{The autocorrelation is an impulse.} && (7.4) \\ &= \mathbf{E}[\mathbf{A} \mathbf{r} \mathbf{r}^* \mathbf{A}^*] \\ &= \mathbf{A} \mathbf{E}[\mathbf{r} \mathbf{r}^*] \mathbf{A}^* \\ \mathbf{A}^{-1}(\mathbf{A}^*)^{-1} &= \mathbf{E}[\mathbf{r} \mathbf{r}^*] \\ \mathbf{A}^* \mathbf{A} &= (\mathbf{E}[\mathbf{r} \mathbf{r}^*])^{-1} && \text{The PEF gives the inverse covariance.} \\ &\text{Q.E.D.} \end{aligned}$$

If you have not been introduced to expectations ($\mathbf{E}[\]$), think of Equation (7.4) as meaning

$$\mathbf{E} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{bmatrix} y_1 & y_2 \end{bmatrix} = \mathbf{E} \begin{bmatrix} y_1^2 & y_1 y_2 \\ y_2 y_1 & y_2^2 \end{bmatrix} = \begin{bmatrix} \sum_t y_t^2 & \sum_t y_t y_{t+1} \\ \sum_t y_{t+1} y_t & \sum_t y_{t+1}^2 \end{bmatrix} \quad (7.5)$$

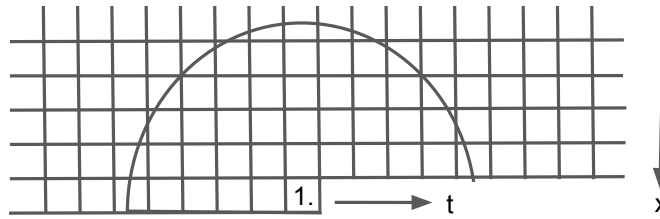
Notice the zero lag of the autocorrelation is on the main diagonal and the first lag above and below it.

7.1.3 Why 2-D PEFs have white output

Chapter 4 in my *GIEE* book (Claerbout, 2014) extends 1-D signal analysis to 2-D and 3-D physical space. There are also many examples in *GIEE* Chapter 7. In summary, to visualize the 2-D idea of a 1-D PEF, wrap a long rope tightly spiraling around a silo inching upward covering many revolutions. The surface of the silo and coiled rope are 2-D spaces for our 2-D imaging games. Let the silo hold the 2-D data and the rope hold the filter. Let the rope be slippery so it can slide over the silo in a 2-D space. Such sliding may be along the axis of the silo, or along the rope or any direction in the 2-D surface.

Figure 7.1 shows how you can think of the rope as either a 1-D or a 2-D filter. At the end of the rope, one filter coefficient is constrained to be a “1.” Filter coefficients in the semicircle near the “1.” in the 2-D space are typically the most significant ones because being nearby the “1.” they most likely give the best predictions of what lies under the “1.”

Figure 7.1: The “1.” at the end of a 1-D rope wrapped on a silo. We consider only the filter coefficients inside the semicircle, outside coefficients supposedly negligible. `appendix/. ropeEnding`



In principle all the coefficients outside the semicircle vanish. For coding convenience, the non-vanishing coefficients commonly lie in a box not a semicircle.

Stew Levin points out that once you have mastered the 1-D whiteness proof, you do not need the 2-D proof in *GIEE* if you know about the helix. Why? Because wrapping one side of a long, long 1-D autocorrelation spike many turns around the helix on the silo shows you a 2-D spike of an autocorrelation which implies 2-D spectral whiteness.

I do not like proving theorems, especially those with negative consequences, but I may save you some trouble if I tell you a curious fact. If you put adjustable (by least squares) coefficients on both sides of the “1,” you spoil the whiteness of the output.

7.2 THE HEART OF NONSTATIONARY PEF USING CALCULUS

²Suppose we have a PEF that represents all previous moments in time. Call it $\bar{\mathbf{a}} = (1, \bar{a}_1, \bar{a}_2, \bar{a}_3, \dots)$. Say that $\bar{\mathbf{a}}$ represents the PEF (inverse spectrum) of the data values $(d_1, d_2, d_3, \dots, d_{98})$. We seek to define the \mathbf{a} that represents the PEF with an appended data value d_{99} . Consider the regression as follows:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} d_{99} & d_{98} & d_{97} & d_{96} \\ \gamma & \cdot & \cdot & \cdot \\ \cdot & \gamma & \cdot & \cdot \\ \cdot & \cdot & \gamma & \cdot \\ \cdot & \cdot & \cdot & \gamma \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} - \gamma \begin{bmatrix} 0 \\ 1 \\ \bar{a}_1 \\ \bar{a}_2 \\ \bar{a}_3 \end{bmatrix} \quad (7.6)$$

The top row says we are trying to fit a new data point d_{99} . The bottom block says the new PEF \mathbf{a} should be highly similar to the PEF that fit earlier data, $\bar{\mathbf{a}}$. The parameter γ should be big enough that the new data point d_{99} does not change \mathbf{a} very much. Rewrite Equation (7.6) as follows:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} d_n & d_{n-1} & d_{n-2} \\ \gamma & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} - \begin{bmatrix} -d_{n+1} \\ \gamma \bar{a}_1 \\ \gamma \bar{a}_2 \\ \gamma \bar{a}_3 \end{bmatrix} \quad (7.7)$$

or in a shortened block-matrix notation, we have the residual to minimize

$$\mathbf{0} \approx \mathbf{r} = \begin{bmatrix} \mathbf{d}^* \\ \gamma \mathbf{I} \end{bmatrix} \mathbf{a} - \begin{bmatrix} -d_{n+1} \\ \gamma \bar{\mathbf{a}} \end{bmatrix}, \quad (7.8)$$

²This section drawn on Fomel et al. (2016) and Claerbout (2017).

where \mathbf{I} is the identity matrix and

$$\mathbf{d} = \begin{bmatrix} d_n \\ d_{n-1} \\ d_{n-2} \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix},$$

For decades Bernard “Bernie” Widrow (Wikipedia) attacked problems of this nature by defining a quadratic form and finding its gradient. (Actually, he thinks in terms of circuit diagrams.) Then he repeatedly made small steps down the gradient (not up). How big are the small steps? Experience teaches.

The quadratic form is $\mathbf{r}^* \mathbf{r}$. We take its derivative to find the search direction.

$$\Delta \mathbf{a} = - (\text{some constant}) \left. \frac{\partial}{\partial \mathbf{a}^*} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \mathbf{r}^* \mathbf{r} \quad (7.9)$$

Form the transpose of the Residual (7.8) and then, differentiate by \mathbf{a}^* . (By \mathbf{a}^* , we mean the complex conjugate transpose of \mathbf{a} .)

$$\frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*} = \frac{\partial}{\partial \mathbf{a}^*} \{ \mathbf{a}^* [\mathbf{d} \ \gamma \mathbf{I}] - [-d_{n+1} \ \gamma \bar{\mathbf{a}}] \} = [\mathbf{d} \ \gamma \mathbf{I}] \quad (7.10)$$

and multiply that onto \mathbf{r} from Equation (7.8) keeping in mind that $\mathbf{d}^* \bar{\mathbf{a}}$ is a scalar.

$$\Delta \mathbf{a} \propto \frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*} \mathbf{r} = [\mathbf{d} \ \gamma \mathbf{I}] \left\{ \begin{bmatrix} \mathbf{d}^* \\ \gamma \mathbf{I} \end{bmatrix} \mathbf{a} - \begin{bmatrix} -d_{n+1} \\ \gamma \bar{\mathbf{a}} \end{bmatrix} \right\} \quad (7.11)$$

$$= \mathbf{d}(\mathbf{d}^* \mathbf{a}) + \gamma^2 \mathbf{a} + \mathbf{d} d_{n+1} - \gamma^2 \bar{\mathbf{a}} \quad (7.12)$$

$$\Delta \mathbf{a} \propto \left. \frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \mathbf{r} = (\mathbf{d}^* \bar{\mathbf{a}} + d_{n+1}) \mathbf{d} \quad (7.13)$$

$$\Delta \mathbf{a} = -\epsilon r_t \mathbf{d} \quad (7.14)$$

It is certainly surprising that the analytic solution to the Regression (7.6) computationally amounts to a single step of the optimization strategy (7.13), a strategy so crude as to be absent from textbooks; yet true (Fomel et al., 2016). Experimentalists first notice that Equation (7.6) demands we supply a not-given constant γ while (1.6) or (7.14) demands a not-given constant ϵ (or λ).

REFERENCES

- Claerbout, J., 2014, Geophysical image estimation by example: Lulu.com.
 ———, 2017, Fitting while whitening nonstationary residuals: SEP-Report, **168**, 255–262.
 Fomel, S., J. Claerbout, S. Levin, and R. Sarkar, 2016, Streaming nonstationary prediction error (II): SEP-Report, **163**, 271–277.
 Levin, S. A., J. Claerbout, and E. R. Martin, 2013, Shortest path to whiteness: SEP-Report, **150**, 13–16.