Nonstationary Signal Tutorial
(beta release 1.01, May 2018)

Jon Claerbout with Kaiwen Wang

Stanford University

SEP-172

Contents

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* (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). Not only that, but, as I was finishing *GIEE*, I was horrified to realize that I had invested much in managing *scalar* data on an irregular grid, but the method did not scale well to *signal* data. Altogether, I could not resist embarking on this booklet. And it worked.

I am now ready to share further development with any and all. Any participant is welcome to contribute illustrations (and ideas)—perhaps becoming a coauthor, even taking over this manuscript. (I'm 80 years old.) The first need now is more examples. Ultimately, all the examples should be presented in reader rebuildable form.

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 booklet on PEFs, (3) GIEE conjugate gradients and subsequent chapters.

Early beta versions of this booklet 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 booklet out out of beta when 50% of the illustrations can be destroyed and rebuilt by readers.

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 May 16, 2018.

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 4, 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 three reasons: First, to enable pleasing displays of them. Second, the equations of physics are generally represented in computers using *finite differences*. Third, we learn by subtracting observed data from model derived data. That difference we call *the residual*. From the residual we are able to make discoveries. Before residuals are minimized to learn the best fitting model, a principle of statistics says residuals should be scaled to uniform strength in both physical space and in Fourier space. We achieve this by a means that requires uniform sampling of data. Since spatial sampling uniformity often cannot be achieved with real data, this booklet 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.

This booklet is novel by attacking data what is nonstationary, meaning that its statistical characterization is not constant in time and space. This approach works by including a new data value to a previously solved imaging problem. 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.

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 $\vec{\mathbf{y}}_t$.

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 both are equivalent to knowledge of a Prediction Error Filter (PEF). PEFs share many properties with differential equations. A PEF is like a differential equation that when driven by a random noise source, it produces data with the spectrum of interest. The word *spectrum* refers not only the temporal spectrum, but also to 3-D spatial spectra. The PEF is derived from data by least squares procedures that we soon see.

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 for each energy source plus another two for each signal receiver.) I expected the study of submarine warfare to result in conceptual advances with acoustic antennas; etc, but, I am not aware of whatever fundamentals it may have uncovered.

0.3.2 PEFs present and future

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. Since my 2014 book, Sergey Fomel and I have found a way to extend the PEF concept to nonstationary data. Not only have we discovered an extension, it is also easier to understand and to code! This ease promises quick results and, it looks like fun! Although I recently turned 80, 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 Learning 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 have fun.

0.4 CREDITS AND THANKS

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. 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.

Chapter 1

Nonstationary scalar signals

1.0.1 Why should residuals be IID?

A principle of Statistics is that residuals of fitting models to data should be Independent and Identically Distributed (IID), meaning in practice that the residuals should be scaled up to be easily visible everywhere in both physical space and Fourier space, this before adjusting model parameters to minimize a sum of residuals. This ensures that all aspects of the data have been probed. Scaling in physical space is easy. For Fourier space we need Prediction Error Filters (PEFs). They come next.

1.0.2 Prediction-error filtering (deconvolution)

A widespread generic model for signal and image data is that the data originated from independent causes that were somehow smoothed, filtered, or convolved before reaching us as data. Because the PEF returns us to uncorrelated and apparently independent sources, it is said to be "deconvolved."

Start with a channel of data (a signal of many thousands of values). We denote these data numbers by $\mathbf{y} = (y_0, y_1, y_2, \cdots)$. A little patch of numbers that we call a "filter" is denoted by $\mathbf{a} = (a_0, a_1, a_2, \cdots, a_{n_\tau})$. In pseudo code these filter numbers are denoted by `a(0), a(1),...,a(ntau)`. Likewise code for the data. The filter numbers slide across the data numbers with the leader being `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 `x+=y`. We use this notation herein, likewise `x-=y` for subtraction. Pseudo code for finding `r(t)` is:

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

With each step in time we prepare to change the filter `a(tau)` a tiny bit. To specify the change, we need a goal for the filter outputs `r(t)` to have minimum energy. To prevent the filter **a** from becoming all zeros, we constrain the first filter coefficient to be unity.

$$\mathbf{a} = [\,1,\, a_1, a_2, a_3, \cdots] \tag{1.1}$$

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 *r*esidual 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 data negative.

Intuitively, PEF output has sucked all the predictability from its input. Appendix 5.1.1 *Why 1-D PEFs have white output* shows 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 state. It cannot restore any delays—because the PEF is *causal*, meaning it has only knowledge of the past; therefore $[\cdots, a_{-2}, a_{-1}] \;=\; \mathbf{0}$. P.E. filtering is sometimes called *blind* deconvolution—stressing that **a** is estimated as well as applied.

For now y_t and a_τ are scalar time functions at a point in space. Before we finish, they will become time functions in a physical space like the quadruple indexed array $y_t(x, y, z)$. Beyond that we will revert to a point in space, but extend to a vector-valued signal. Theory proceeds somewhat like with scalar signals, but data and prediction error are vector functions like $\vec{\mathbf{r}}_t = (u_t, v_t, w_t)$, where the PEF begins not with a 1.0 but with a three component identity matrix **I**.

1.1 THE HEART OF NONSTATIONARY PEF WITH NO CALCULUS

At any moment in time, we may think of the PEF output $r_t = \sum_\tau a_\tau y_{t-\tau}$, as a dot product of the filter **a** onto some backwards piece of input data. Denote that backwards piece by **d**. (Other moments in time have other values in the **d** vector.) At that moment, the PEF output is $\mathbf{a} \cdot \mathbf{d}$. Consider the exploratory filter $\mathbf{a} - \epsilon\,\mathbf{d}$, where ϵ is a tiny positive number. Its output r_t is $(\mathbf{a} - \epsilon\,\mathbf{d}) \cdot \mathbf{d} = (\mathbf{a} \cdot \mathbf{d}) - \epsilon\,(\mathbf{d} \cdot \mathbf{d})$. To reduce the size $|r_t|$ of the new output residual, these two terms must have opposite polarity; but $r_t = (\mathbf{a} \cdot \mathbf{d})$ may have either polarity. Try instead the filter update $(\mathbf{a} - \epsilon\, r_t \mathbf{d})$. Its output is $(\mathbf{a} \cdot \mathbf{d}) - \epsilon\, r_t (\mathbf{d} \cdot \mathbf{d})$, which on rearrangement is $(\mathbf{a} \cdot \mathbf{d})(1 - \epsilon\,(\mathbf{d} \cdot \mathbf{d}))$, is easily assured smaller than $r_t = (\mathbf{a} \cdot \mathbf{d})$. Thus,

$$\Delta\mathbf{a} \;=\; -\,\epsilon\, r_t\, \mathbf{d} \;=\; -\,\epsilon\, r_t\, y_{t-\tau} \tag{1.2}$$

In summary:

Filter	Definition	Output
		$r_t = \sum_\tau a_\tau y_{t-\tau}$
$\mathbf{a} = a_\tau$	$\mathbf{d} = y_{t-\tau}$	$r_t = \mathbf{a} \cdot \mathbf{d}$
$\mathbf{a} - \epsilon\,\mathbf{d}$	First trial	$\mathbf{a} \cdot \mathbf{d} - \epsilon\,(\mathbf{d} \cdot \mathbf{d})$
$\mathbf{a} - \epsilon\; r_t\; \mathbf{d}$	Revision	$\mathbf{a} \cdot \mathbf{d} - \epsilon\; r_t\,(\mathbf{d} \cdot \mathbf{d})$
$\mathbf{a} + \Delta\mathbf{a}$	Success!	$(\mathbf{a} \cdot \mathbf{d})(1 - \epsilon\,(\mathbf{d} \cdot \mathbf{d}))$

The last line says the filter change reduces the size of the residual $r_t = (\mathbf{a} \cdot \mathbf{d})$. Clearly we want to choose $0 < \epsilon < 1/(\mathbf{d} \cdot \mathbf{d})$, but might want it much smaller than that, because our numerical choice of ϵ balances fitting the present versus fitting the past.

We arrived at Equation (1.2) by guesswork and logic, no calculus. Appendix 5.2 *The heart of nonstationary PEF using calculus* derives the same result with two pages of calculus, but it does so under the limitation of the ℓ_2 norm. You might notice that r_t and each component of $\mathbf{d}$ in the expression $\epsilon\, r_t\, \mathbf{d}$ can be stretched or shrunken by any polarity preserving function, such as $r \leftarrow r/|r|$. After we finish with the basics, we will return to specialized applications that take advantage of this extra flexibility. (Hint: We will go beyond ℓ_1.)

1.1.1 Code for prediction error = deconvolution = autoregression

The following code does "deconvolution," also known as "autoregression."

```
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 `#forward` line in the code preceeding applies the filter to get the residual. The `#adjoint` line in the code is building Equation (1.2). The adjoint operation is also called back-projection. The code preceeding, based on little more than the definition of dot product, is a example of a deeper principle in classroom mathematics. The line `#forward` is a matrix times a vector. The line `#adjoint` is also a matrix times a vector. It is the same matrix `y(t-tau)`, but one `tau` loop does the matrix transpose of the other because one carries `tau` space to `t` space, while the other carries `t` space to `tau` space. The transpose of any matrix is $\mathbf{M}^*_{ij} = \mathbf{M}_{ji}$. The line `da(0)=0` is a constraint to prevent changing the `a(0)=1` maintaining the definition of `r(t)` as a residual. Common sense has given us the above example of classroom fundamentals: Put the residual into the adjoint (transpose) to get the gradient; then go down. We got the gradient without ever calculating a derivative, without needing a function to take the derivative of! If coding adjoints is new to you, I recommend Chapter 1 in *GIEE* (Claerbout, 2014). It is free on the internet.

Suppose while running along time `t`, we find the line in the code above computing $\Delta\mathbf{a}$ saying for practical purposes `da(tau)=r(t)*y(t-tau)` vanishes for all `tau>0`. This statement would delight any stationary theorist, because the gradient vanishing $\Delta\mathbf{a} = \mathbf{0}$ says we are finished. It is saying the residual $\mathbf{r} = r_t$ is orthogonal to all the fitting functions $\mathbf{y}_\tau = y_{t-\tau}$. (A particular fitting function is $\mathbf{y}_9 = y_{t-9}$.) With our nonstationary technology, we do not expect $\Delta\mathbf{a}$ ever to be exactly zero, but we do expect it to get small and then

bounce 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 how to characterize nonstationarity; but, we will come up with a good starting guess for ϵ. While theorizing, there is much we will learn by experience.

1.1.2 The outside world—real estate

The regression updating approach introduced here is not limited to convolutional 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 first column contains the recorded sales prices. The next column contains the square footages, the third column contains 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 a learning machine.

1.1.3 Why does the residual into the adjoint give the gradient?

Basic geophysical model $\mathbf{m}$ estimation is summarized by the residual minimization $\mathbf{0} \approx \mathbf{r}(\mathbf{m}) = \mathbf{F}\mathbf{m} - \mathbf{d}$. In the special case in which $\mathbf{F}$ is a convolution matrix (downshifted columns of data $\mathbf{d}$), this formulation fits the estimation of a prediction filter $\mathbf{m}$. But, for many applications we want a prediction-error filter $\mathbf{a}$. Thus, we think of PEF estimation as the constraint $a_0 = 1$ along with the augmented matrix $\mathbf{Y} = [\mathbf{d}|\mathbf{F}]$ where $\mathbf{Y}$ is also a convolution matrix.

This tutorial document shows twelve pseudo codes for diverse applications of PEFs; therefore let us be sure that everyone is onboard with the idea that the gradient is a residual dumped into a transposed modeling operator. Chapter 2 of my textbook *GIEE* (Claerbout, 2014) guides you through every step with a 2×3 matrix. In summary, the quadratic form you are minimizing is $\mathbf{r} \cdot \mathbf{r} = (\mathbf{m}^*\mathbf{F}^* - \mathbf{d}^*)(\mathbf{F}\mathbf{m} - \mathbf{d})$ with derivative by $\mathbf{m}^*$ being $\Delta\mathbf{m} = \mathbf{F}^*\mathbf{r}$. Likewise, the derivative of $\mathbf{a}^*\mathbf{Y}^*\mathbf{Y}\mathbf{a}$ by $\mathbf{a}^*$ is $\Delta\mathbf{a} = \mathbf{Y}^*\mathbf{r}$.

1.2 ESTIMATING TOGETHER MISSING DATA WITH 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 code preceeding. The `#forward` line is the usual computation of the prediction error. To understand that the lines labeled `#adjoint` are adjoints (transposes), compare each to the forward line, and observe that input and output spaces have been swapped. At the code's bottom are the three lines for missing-data estimation. The code "looks canonical" (by sticking a residual into an adjoint), but what is it doing?

```
#                 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
   }
```

We can easily say what the program does, but that does little to explain why it is doing something logical. It is easier to assert that it must work because a residual `r(t)` is going into the transpose of forward modeling, the spaces swapped being `r(t)` and `y(t-tau)`.

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.

PEF estimation proceeds quickly on early parts of the data. 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 **y** and **r** after each pass.) To speed the code, one might capture the `t` values that are affected by missing data, thereafter iterating only on those.

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

It is a near certainty this method works fine if a small percentage of data values are missing. But, what if a large percentage of values were missing? It might work, or it might fail. There should be strategies to help it work better. There are valuable uses for data restoration. Figure 2.4 illustrates the idea.

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.

1.2.1 Old 1-D examples I have done in the stationary world

Figure 1.1 shows an appealing test case. The conclusion to draw is that PEF interpolation preserves the character of the given data, unlike linear or cubic interpolation. a PEF resembles a differential equation (more like the finite difference representation of a differential equation) which may account for the more "physical" look of the interpolation.

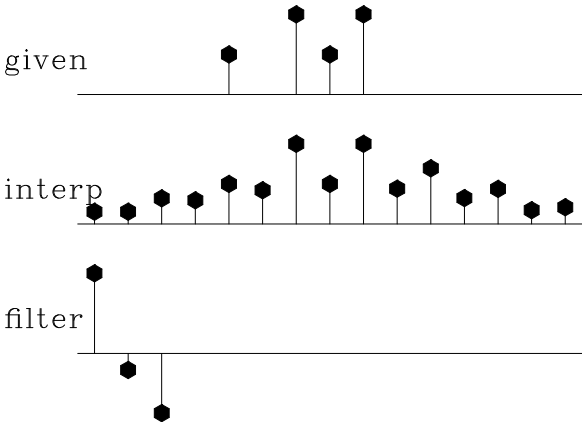

Figure 1.1: Top is given data, taken to be zeros off the ends of the axis. Middle is the given data with interpolated values. The restored data has the character of the given data. Bottom shows the best fitting filter. Its output (not shown) has minimum energy. (Claerbout, PVI) signal/. missif

Another problem of missing data with unknown PEF that I once solved is copied in Figure 1.2. It clearly shows interpolation beyond aliasing. I took it from page 197 of 2012 version of *GIEE.* A sinusoid was sampled densely on the left and sparsely on the right. Toward the right, the interpolated function is well sampled despite widely separated data values, i.e., data sampled beyond aliasing.

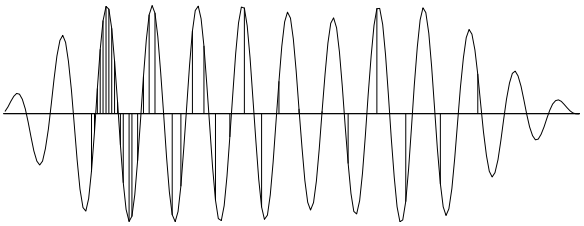

Figure 1.2: Simultaneous estimation of PEF and stationary missing data (taken from the 2012 version of *GIEE,* on page 197) (Claerbout) signal/. subsine390

1.3 CHOOSING THE STEP SIZE

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:

$$\Delta\mathbf{a} \;=\; -\,\frac{\epsilon\, r_t}{\sigma_y^2}\,\mathbf{d} \tag{1.3}$$

That is the story for `epsilonA` in the code above. For the missing data adaptation rate, `epsilonY`, no normalization is required because `r(t)` and `y(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`.

Epsilon ϵ 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

$$\epsilon \;=\; 1/\lambda \tag{1.4}$$

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

I often think of the memory function $(1-\epsilon)^t$ as 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 roughly exceed the number of filter coefficients `ntau`. To avoid overfitting, I suggest beginning with $\lambda = 100 \times$ `ntau`.

There is a pitfall in the paragraph above. 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):

$$m_t \quad = \quad \frac{t-1}{t}\, m_{t-1} \;+\; \frac{1}{t}\, r_t \quad = \quad \frac{1}{t} \sum_{k=1}^{t} r_k \tag{1.5}$$

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 might like to skip to Chapter 2.

1.4 HYPERBOLIC PENALTY FUNCTION

Most people do data fitting by minimizing the sum of the squared residuals—called "least squares" or the ℓ_2-norm approach. Computations are generally easy, but a single outlandish residual ruins everything. The ℓ_1-norm approach minimizes the sum of absolute values of residuals. The median is a child of ℓ_1. Occasional humongous residuals detract little from the solutions. Regressions solved by ℓ_1-norm fitting are described as "robust."

GIEE has many examples of practical 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 75^{th} 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	$q = r$	$\lvert q\rvert$	$q/\lvert q\rvert$	$\text{sgn}(\mathbf{q})$
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."

1.4.1 How can the nonstationary PEF operator be linear?

Formally, finding the PEF is $\mathbf{a} = \text{argmin}_{\mathbf{a}}(\mathbf{Ya})$ subject to $a_0 = 1$, while using it is $\mathbf{r} = \mathbf{Ay}$. 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 $\Delta a_\tau = 0$. There is an example in Chapter 2, Figure 2.1.

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 $\mathbf{d}^*\mathbf{B}^*\mathbf{Bd}$ 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 (1.2) 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, $sgn(r) = r/|r|$, or the derivative of the hyperbolic penalty function, $softclip(r)$. Imagine an arbitrary squeezing function *RandSqueeze()* that squeezes its argument by an arbitrary polarity-preserving squeezing function. Each τ might have its own $RandSqueeze_\tau()$ mixing $signum()$ and $softclip()$ and the like. The possibilities are bewildering. We could update PEFs with the following:

$$\Delta a_\tau \;=\; -\,\epsilon\ \text{RandSqueeze}(r_t)\ \text{RandSqueeze}_\tau(y_{t-\tau}) \tag{1.6}$$

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 $RandSqueeze_\tau()$. Given enough data, how would we identify the $RandSqueeze_\tau()$ 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.

Convex functions do not have banana-shaped contours, a problem for many methodologies, but not a problem herein. However, arbitrary squeezing and stretching functions could lead to multiple minima.

1.5.5 The wide world of applications

What is the most general formulation? With vector-valued signals after Chapter 3 we may find unexpected opportunities, such as vorticity in the ocean[1], or Alfven waves in the ionosphere. In time, diverse applications will crop up.

1.5.6 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 1.3 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.

1.5.7 Sparse decon results that I aspire to reconquer

Antoine Guitton (Guitton and Claerbout, 2015) analyzed five data sets getting amazing results on all five. Two are shown in Figures 1.4 and 1.5. The clarity of polarity in every case is wonderful for geologic interpretation.

[1] See the youtube for "Perpetual ocean."

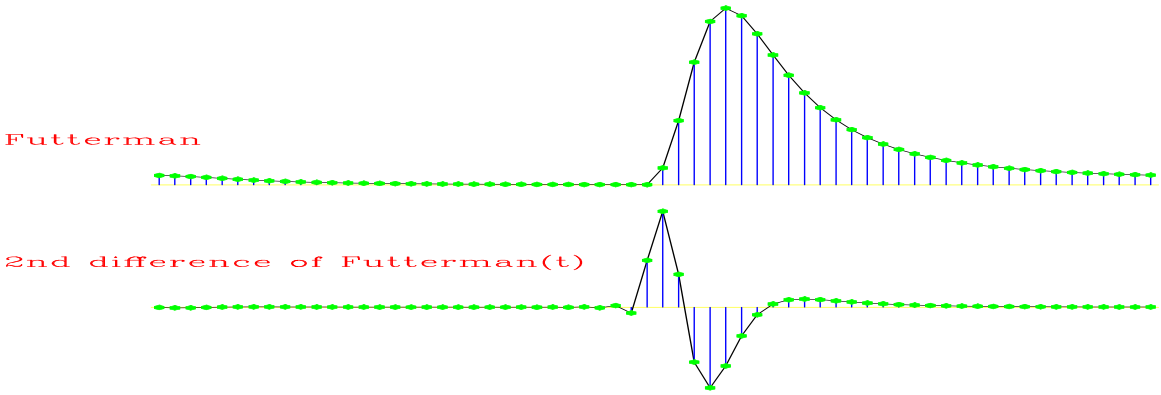

Figure 1.3: The causal constant Q response and its second finite difference. The first two lobes are approximately the same height, but the middle lobe has more area. That third lobe is really small. Its smallness explains why the water bottom could seem a Ricker wavelet (second derivative of a Gaussian) while the top of salt would seem a doublet. (Claerbout) signal/. futter

Guitton's examples were done with a stationary theory that allows the inverse shot wavelet being slightly noncausal. This suggests we should always try to spike not at the wavelet onset (which is what PEFs do), but somewhere more like the center lobe of the Gaussian, namely, at the second lobe of the second derivative of the Futterman.

The nonstationary method has the ability to seek sparseness as Guitton did with the hyperbolic penalty function. Unfortunately for us (and many others before us) PEF outputs lose their whiteness when extended non causally. But, the goal of lucid polarity is a truly rewarding one.

Guitton's work did not extend readily to wider shot-hydrophone separations, a feature naturally exploitable by the nonstationary approach. If we could get everything together, we could reasonably hope to make millions. We should begin guessing!

As a side issue, for each travel time depth $\tau = z/v$, we wish the phase correction of $\exp(-|\omega|\tau/Q)$. Is that easy (like Stolt migration) or harder like downward continuation? Hmm. In any case it is not difficult.

REFERENCES

Claerbout, J., 2014, Geophysical image estimation by example: Lulu.com.

Guitton, A. and J. Claerbout, 2015, Nonminimum phase deconvolution in the log domain: A sparse inversion approach: GEOPHYSICS, **80**, WD11–WD18.

Resnick, M., 2017, Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play: The MIT Press, Cambridge, MA.

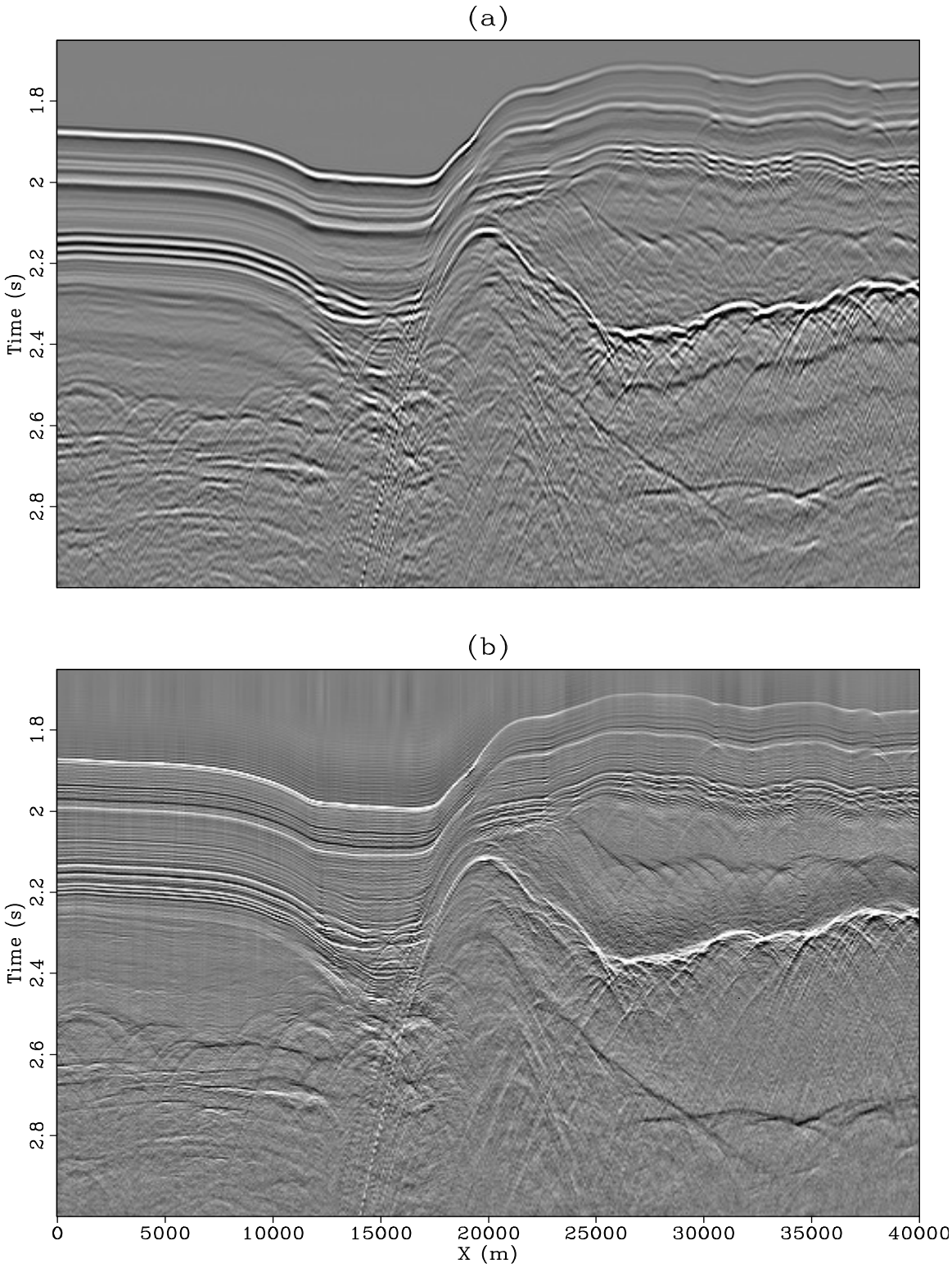

Figure 1.4: Gulf of Mexico. Top is before sparse decon, bottom after. Between 2.25s to 2.70s, the right side is salt (no reflectors). Notice salt top reflection is white, bottom black. Notice that sparse decon has eliminated bubble reverberation in the reflection-free salt zone (as well as elsewhere).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

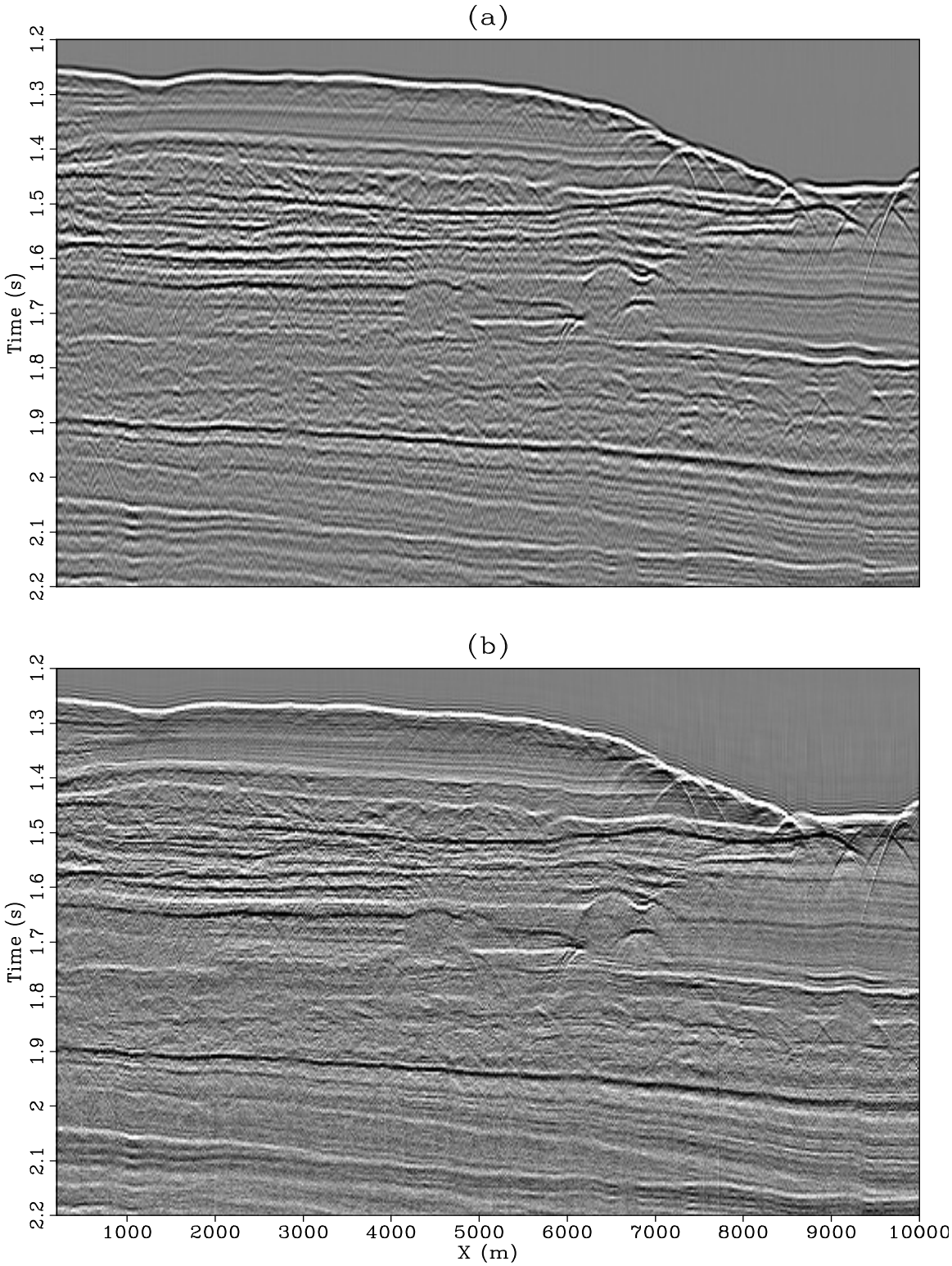

Figure 1.5: Offshore west Australia. Notice how the sparse decon creates many events that are pure white or pure black. White denotes a hard reflector, black a soft one. signal/. antoineAustralia


I flipped
my class

Chapter 2

Spatial deconvolution

2.1 AVERAGING OVER TIME AND SPACE

[1]A streaming 1-D prediction filter is a decaying average of earlier prediction filters; however, these earlier filters need not all be saved in memory. Because they vary smoothly, we may simply use the most recent one. Call it $\bar{\mathbf{a}}$. In two dimensions, $\bar{\mathbf{a}}$ becomes some average of its previous value on each of its two axes. For example, instead of updating from the previous moment $\mathbf{a}(t-\Delta t, x)$, we could update from the previous location $\mathbf{a}(t, x-\Delta x)$. That would be learning over x while filtering over t. More generally, an update could leap from a base that is a weighted average over time and space. We would update $\mathbf{a} \leftarrow \bar{\mathbf{a}} + \Delta\mathbf{a}$ with the following:

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

Notice that the weights sum to unity. The averaging region is an area roughly $\lambda_x\lambda_t$ pixels squared in size. The coding requires not only saving $\mathbf{a}$ at the previous time, it requires at the previous x, namely at $x-\Delta x$, all lags of $\mathbf{a}$ saved over all time. The memory cost is $n_t \times n_\tau$, not bad.

In 3-D, it looks like we will need a plane of saved PEFs. In higher dimensional spaces, we need store PEFs only in the zone of the transition from the filtered to the unfiltered. Thus, in 5-D, we need to store a 4-D volume of PEFs. Do not let that trouble you though. Because the PEFs are generally smoothly variable, they can be linearly interpolated from a sparse grid.

PEFs on the previous trace $\mathbf{a}(t, x-\Delta x)$ can be smoothed symmetically on the time axis. Such smoothing expands the averaging region from the quadrant behind (t, x) to the halfspace behind x.

Stationary decon should remove a shot waveform. Nonstationary decon starts from there but has the added opportunity of removing the waveform of the propagating wave. It evolves with travel time (Q and forward scattered multiples). It also evolves with space, especially shot to receiver separation.

[1]Drawn from Fomel et al. (2016).

2.1.1 Bubble removal

The internet easily yields 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 2.1 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, $\mathbf{\Delta a} = (1, 0, 0, \text{more zeros}, 0, a_{20}, a_{21}, \cdots, a_{80})$.

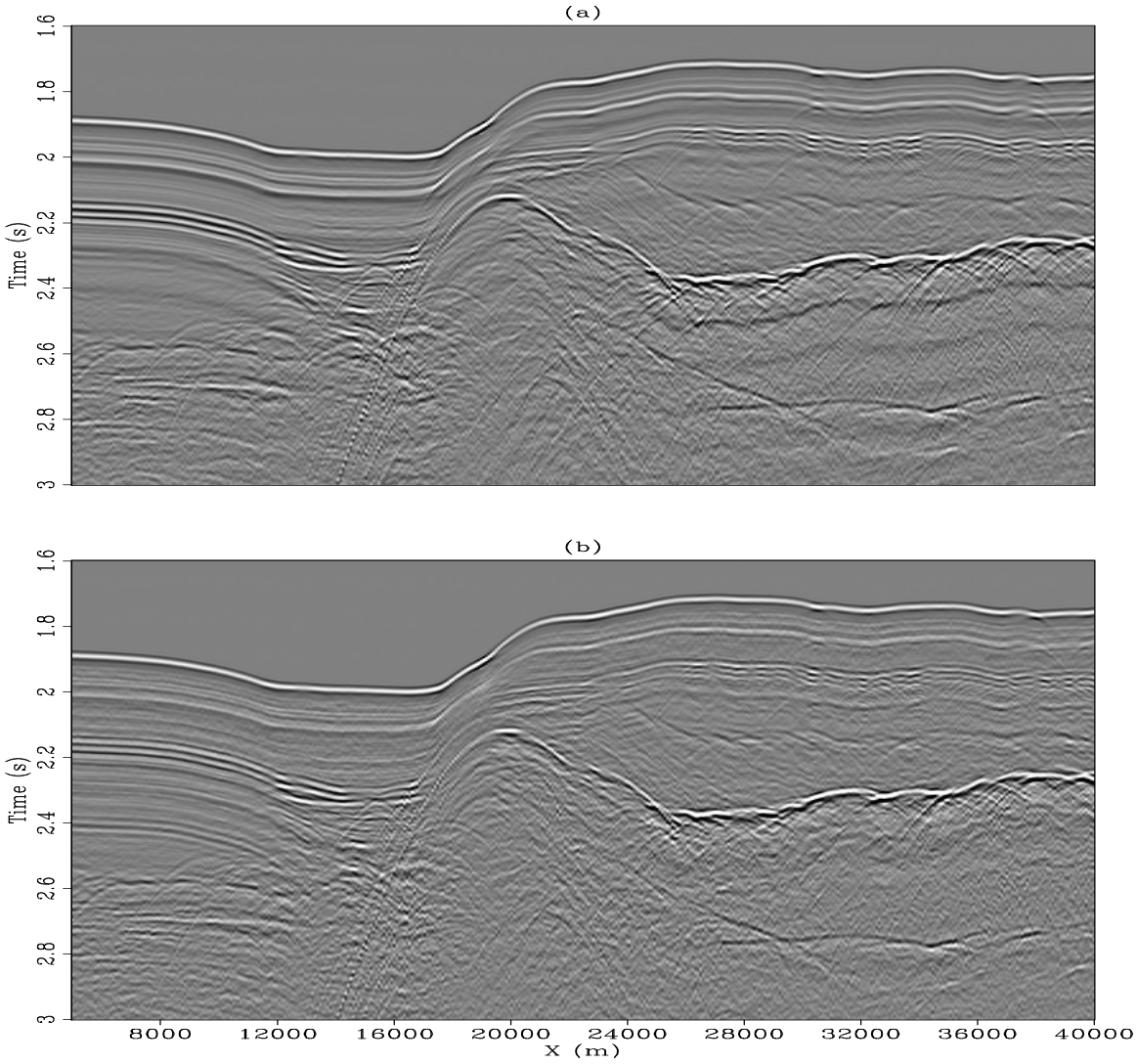

Figure 2.1: Debubble done by the nonstationary method. Original (top), debubbled (bottom). On the right third of the top plot, prominent bubbles appear as three quasihorizontal black bands between times 2.4s and 2.7s. Blink overlay display would make it more evident that there is bubble removal everywhere. (Joseph Jennings) image/. debubble-ovcomp

2.1.2 Two-dimensional PEF

We have seen 1-D PEFs applied to 2-D data. Now for 2-D PEFs. Signal analysis extends to image analysis quite easily except for the fact that the spike on the PEF is not in the middle or on a corner of the 2-D filter array but on its side. This old knowledge is summarized in Appendix 5.1.2 *Why 2-D PEFs have white output.*

Figure 2.2: A PEF is a function of lag `a(tl,xl)`. It is lying backward herein—shown as crosscorrelating seismic data with t down, x to the right. On the filter, τ runs up, x runs left. (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 the 1-D code herein, we 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-end 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 below the lag loops `(tl,xl)`, cover the entire filter. First, the residual `r(t,x)` calculation (`# Filter`) is simply the usual 1-D convolution seen additionally on the 2-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 preceeding it.) Finally, the update line `a-=da` is trivial.

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

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

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

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

This code whitens (flattens) nonstationary spectra in the 2-D frequency (ω, k_x)-space. The local autocorrelation tends to a delta function in 2-D lag (`tl,tx`)-space. Everybody's 2-D image estimations need code like this code to achieve IID residuals.

2.1.3 2-D PEFs as plane wave destructors and plane wave builders

Two dimensional PEFs are useful in seismology. Convolving an image with the PEF in Figure 2.3 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.

Convolving two PEFs with two different slopes makes a 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.

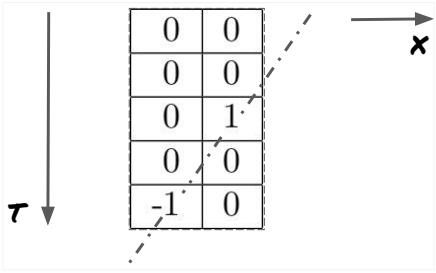

Figure 2.3: Plane wave destructor for events of slope 2. Applied to data it destroys that slope in the data. Used in a missing data program, that slope is produced where the data is missing. (Claerbout) image/. DippingPEF5

A PDF can be specified, as I did in making Figure 2.3, 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 5. Using the PEF in Figure 2.3 in a filtering program, that slope is destroyed. Using that PEF in a missing data program, that dip is built. (Outside our present topic of nonstationary data, traditional stationary methods can fill large holes much more rapidly than herein by using polynomial division.)

Figure 2.4 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.

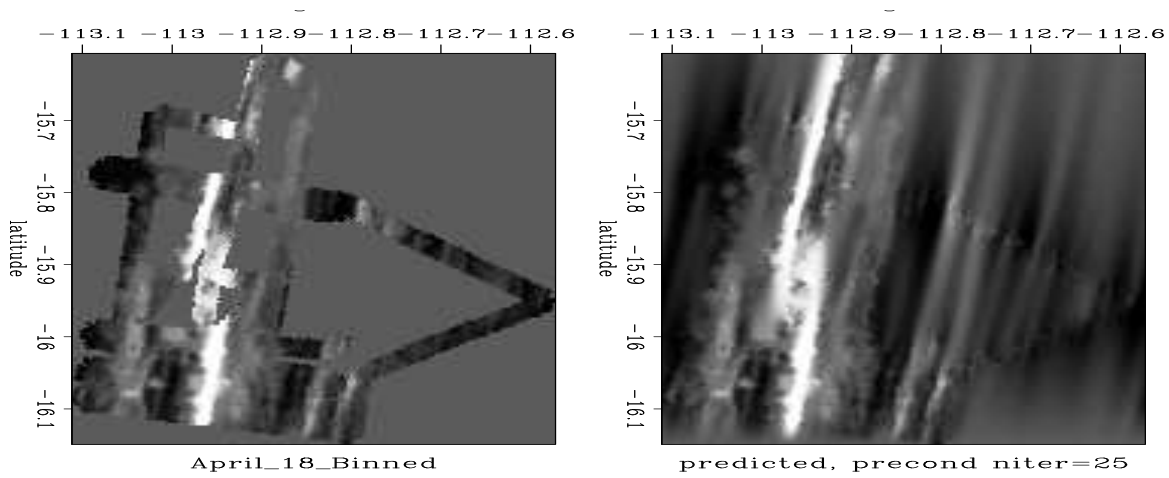

Figure 2.4: (left) Seabeam data of mid-Pacific transform fault. (right) After interpolation by stationary 2-D PEF. The purpose herein is to guess what the ship would have recorded if there were more hours in a day. *(GIEE)* image/. seapef90

2.1.4 Why 2-D PEFs improve gradients

This example shows why PEFs improve gradients. Figure 2.5 shows a shot gather **d** before and after *stationary* PEFing **Ad**. Notice the back scattered energy. Near zero offset, it almost vanishes on the raw data whereas it is prominent after the PEF. The backscatter energy tells us a great deal about reflectors topping near 2.5-2.8s. This 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 over interpreted.

It disappoints me that I am not aware of formal tests of the assertion that PEFs improve model fitting. Sensible priors for any test may be expressed by regularization, or by a suitably gapped PEF (because we are not seeking to model near the Nyquist). How might such tests be framed objectively?

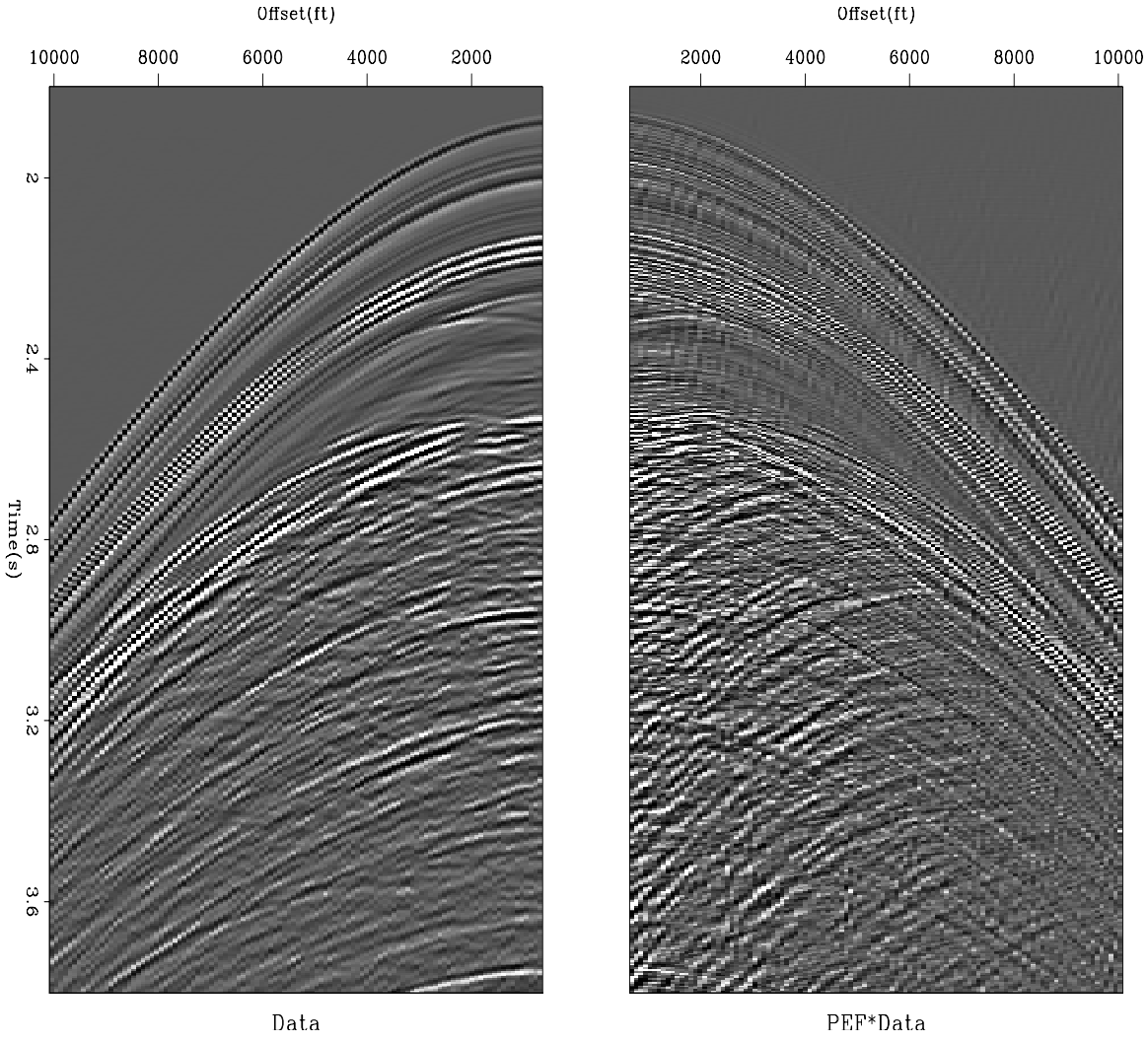

Figure 2.5: (left) Shot gather; (right) mirror imaged after global 2D PEF (20×5). (Antoine Guitton, GIEE) image/. antoinedecon2

2.2 INTERPOLATION BEYOND ALIASING

Wavefields are parameterized by their velocity, namely, their slope in (x,t)-space. Slope is a one parameter space. PEFs in (x,t)-space have two parameters. 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 measured 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 2.6. It shows badly spatially aliased data processed to interpolate three intermedi-

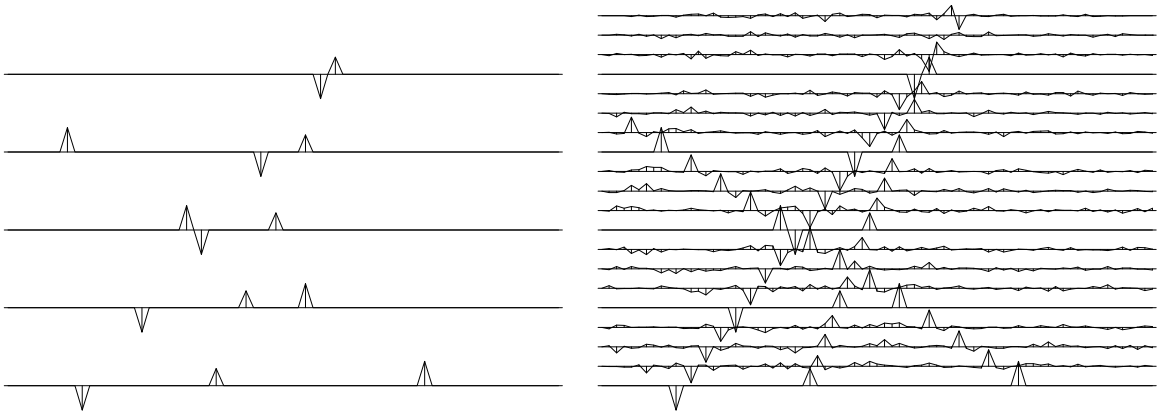

Figure 2.6: Left is five signals, each showing three arrivals. An expanded PEF from the left was compressed to create interpolated data on the right. There are three new traces between the given traces. The original traces are preserved. (Claerbout, PVI) image/. lace3

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 5, 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.

2.2.1 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 `t` and `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 (tl,xl):
(1)                r1(t ,x )  +=  a(tl,xl) * y(t-tl  , x-xl  )
(2)                r2(t ,x )  +=  a(tl,xl) * y(t-tl*2, x-xl*2)              # Dilated PEF

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

2.2.2 Multiscale missing data estimation

Observe the form of missing data updates in one dimension from pseudocode on page 5. Express it in two dimensions, without and with trace skipping.

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

(I am embarrassed that I am unsure of the negative signs in (5) and (6), likewise in Chapter 1. Can anyone help?)

We intend to use only lines (2), (4), and (5), with all the usual looping statements and constraints.

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

Our goal is finding data for the odd-numbered channels, originally zero valued. The code should use only the data on even-numbered channels to find the PEF. Lines (2) and (4) give us a 2-D PEF. Line (5) uses that PEF and requires the presumption $\mathbf{r}_1 \approx \mathbf{r}_2$ to spray out data to the odd numbered channels. The success of this method depends on $\mathbf{r}_2$ being a satisfactory proxy for $\mathbf{r}_1$; it depends on dilation invariance.

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 2.6 suggests a pathway remains nearby to be found.

2.3 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 **a** to predict signal **x** from signal **y**. Either signal might lag the other. Take the filter to be two-sided, `[a(-9),a(-8),...,a(0),a(1),...,a(9)]`. Let us begin from `a(0)=1`, but not hold that as a constraint.

```
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 from the filters a time `shift`. Here I have simply computed the moment. That would be correct if signals **x** and **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 this code multiple times while decreasing epsilon.

Besides time shifting, the filtering operator has the power of gaining and of changing color. Suppose, for example that brother **y** and sister **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!

2.4 DISJOINT REGIONS OF SPACE

2.4.1 Geostatistics

Figure 2.7 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.

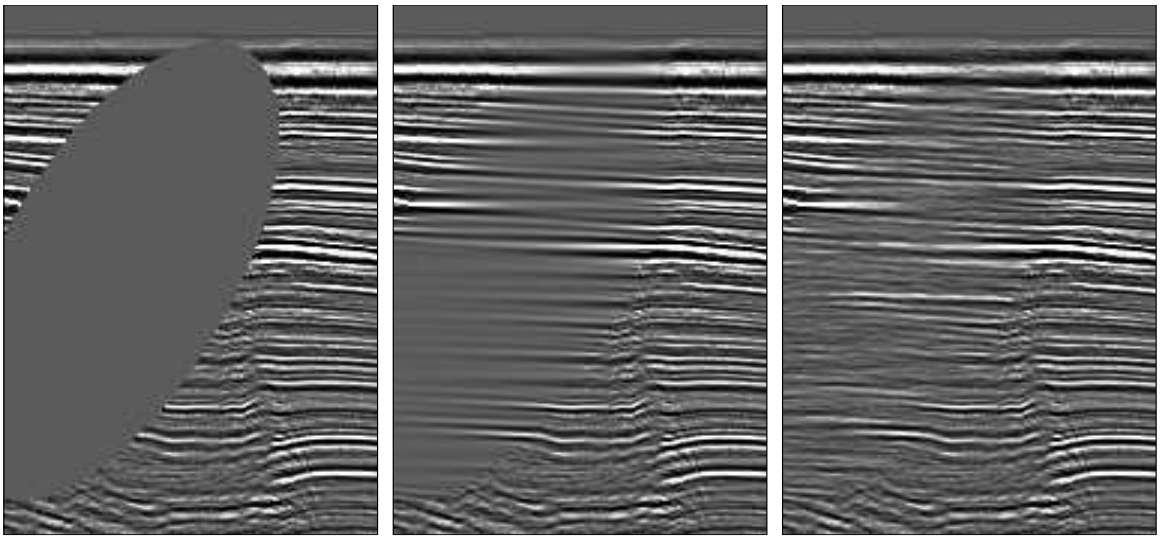

Figure 2.7: A 2-D stationary example from *GIEE*. A CDP stack with a hole punched in it. The center panel attempts to fill the hole by methodology similar to herein. The right panel uses random numbers inverse to the PEF to create panel fill with the global spectrum while assuring continuity at the hole 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 is a solution to a

problem such as $\mathbf{0} \approx \mathbf{Fm} - \mathbf{d}$ with regularization $\mathbf{0} \approx \mathbf{Am}$, The solution to the right panel uses a different regularization, $\mathbf{0} \approx \mathbf{Am} - \mathbf{r}$, 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 missing data program. Debug your code. Clean it up a little and give it to me. Maybe I can use it. Better yet, make me a nice illustration and your name will go in the caption. Still better, make sure your illustration and code is presented in reader rebuildable form.

2.4.2 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.

2.4.3 Rapid recognition of a spectral change

This booklet begins with 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

$$(...., T, T, T, o, o, o, T, T, T, o, o, o, T, T, T, o, o, T, T, o, o, T, T, o, o, T, T, o, o, T, T, o, o....)$$

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.

$$\mathbf{a} \quad = \quad \frac{\sigma_{\text{right}}}{\sigma_{\text{right}} + \sigma_{\text{left}}} \, \mathbf{a}_{\text{left}} \; + \; \frac{\sigma_{\text{left}}}{\sigma_{\text{right}} + \sigma_{\text{left}}} \, \mathbf{a}_{\text{right}} \tag{2.2}$$

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 2.9.

2.4.4 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 2.8) 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.

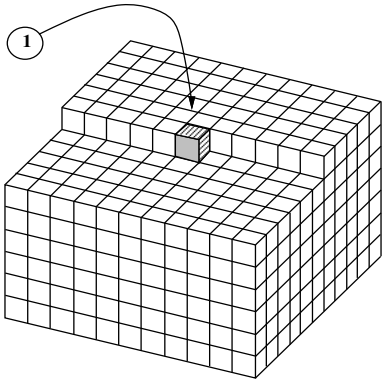

Figure 2.8: The coefficients in a 3-D PEF. (*GIEE*) image/. 3dpef

You are trying to remove noise from the test photo of Lenna (Figure 2.9). 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.

2.4.5 What physical phenomena gives the spectra of a 3-D PEF?

Although it is clear how to fit a single 3-D PEF to data, it might not be relevant to seismic data. Waves fill a volume with pancakes, not with noodles. When I see 3-D data, $y(t, x, y)$,

Figure 2.9: Lenna, a widely known photo used for testing engineering objectives in photometry. (Wikipedia) image/. Lenna

I visualize it containing planes. A plane in 3-D looks like a line in both (t, x) and (t, y) space. It is more efficient to fit two planes each with a 2-D PEF $[a(t, x), b(t, y)]$ than with a single 3-D PEF $a(t, x, y)$. If you have been thinking about a regularization, it now becomes two regularizations. What physical 3-D fields call for 3-D PEFs? I could guess, but this is not the time and place.

REFERENCES

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

Fomel, S., J. Claerbout, S. Levin, and R. Sarkar, 2016, Streaming nonstationary prediction error (II): SEP-Report, **163**, 271–277.

Chapter 3

Vector-valued signals

[1]We 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 3.1 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.

3.0.6 Multi channels = vector-valued signals

Widespread adaptation of multicomponent recorders leads to new opportunities indicated by the lower bulk of Figure 3.1. 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

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

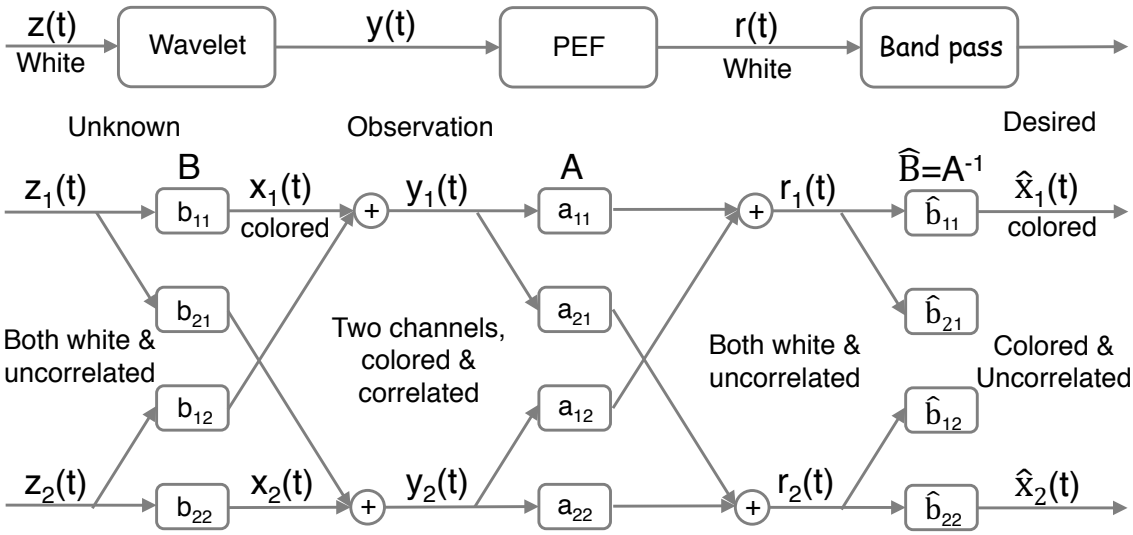

Figure 3.1: Top is scalar decon. Bottom is vector decon. In nature, two uncorrelated white random signals $\mathbf{z}$ get colored thereby creating $\mathbf{x}$, which then gets mixed and creates our observations $\mathbf{y}$. Vector decon converts $\mathbf{y}$ to uncorrelated white signals $\mathbf{r}$, which hopefully are a reasonable approximation to $\mathbf{z}$. If $\mathbf{r} \approx \mathbf{z}$, then $\mathbf{AB} \approx \mathbf{I}$, therefore, recoloring $\mathbf{r}$ without mixing gives us $\hat{\mathbf{x}}$, which should match the original colored signals $\mathbf{x}$. (Kaiwen Wang) vector/. diagram

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 3.1 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 3.1. 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.

3.1 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.

3.1.1 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 $\Delta\mathbf{a} = -\left(\epsilon\, r/\sigma_y^2\right) y_{t-\tau}$. 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

$$\Delta \mathbf{a} \quad = \quad - \left(\frac{\epsilon \, r}{\sigma_{\mathbf{r}} \sigma_{\mathbf{y}}} \right) \, \mathbf{y}_{t-\tau} \tag{3.1}$$

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.

3.1.2 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 3.1. But, it was better not to think at all; instead simply code the modeling, its adjoint, and stuff in the residual! Phew.

3.1.3 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.

3.1.4 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 5. 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).

3.1.5 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}.

$$\mathbf{B} \;=\; \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \;=\; \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} \;=\; \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{3.2}$$

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

$$b_{11} \;=\; a_{22}/(a_{11}a_{22} - a_{21}a_{21}) \;\approx\; 1/a_{11} \tag{3.3}$$
$$b_{22} \;=\; a_{11}/(a_{11}a_{22} - a_{21}a_{21}) \;\approx\; 1/a_{22} \tag{3.4}$$

The result of polynomial division $\hat{x}(Z) = r(Z)/A(Z)$ is recognizable in the code by $\hat{\mathbf{x}}_t =$ `xhat(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.

3.2 CHOLESKY DECORRELATING AND SCALING

The two independent channels of unit-variance random numbers in $\mathbf{r}$ *entering* filter $\mathbf{B}$ in Figure 3.1 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 $\mathbf{W}$.

$$\mathbf{W}(\tau = 0) \quad = \quad \begin{bmatrix} \sigma^2_{r_{11}} & \sigma^2_{r_{12}} \\ \sigma^2_{r_{21}} & \sigma^2_{r_{22}} \end{bmatrix} \quad \approx \quad \begin{bmatrix} (\mathbf{r}_1 \cdot \mathbf{r}_1) & (\mathbf{r}_1 \cdot \mathbf{r}_2) \\ (\mathbf{r}_2 \cdot \mathbf{r}_1) & (\mathbf{r}_2 \cdot \mathbf{r}_2) \end{bmatrix} \tag{3.5}$$

Consider the expectation (leaky sum over time) $\mathrm{E}[\,\mathbf{r}\,\mathbf{r}^*]$. 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 $\mathrm{E}[\,\mathbf{r}\,\mathbf{r}^*]$ is like the $\mathbf{W}(\tau = 0)$ we computed with Equation (3.5) 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:

$$\mathbf{W} \quad = \quad \mathbf{V}\,\mathbf{V}^* \tag{3.6}$$

$$\mathbf{V}^{-1}\mathbf{W}(\mathbf{V}^*)^{-1} \quad = \quad \mathbf{I} \tag{3.7}$$

$$\mathbf{C}\mathbf{W}\mathbf{C}^* \quad = \quad \mathbf{I} \tag{3.8}$$

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

$$\mathbf{q} \quad = \quad \mathbf{C}\,\mathbf{r} \tag{3.9}$$

Using Equation 3.8 the expectation of this new variable $\mathbf{q}$ is as follows:

$$\mathrm{E}[\mathbf{q}\mathbf{q}^*] \;=\; \mathrm{E}[\mathbf{C}\,\mathbf{r}\mathbf{r}^*\,\mathbf{C}^*] \;=\; \mathbf{C}\,\mathrm{E}[\mathbf{r}\mathbf{r}^*]\,\mathbf{C}^* \;=\; \mathbf{C}\mathbf{W}\mathbf{C}^* \;=\; \mathbf{I} \tag{3.10}$$

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

3.3 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 3.1, 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

$$\mathbf{U} \quad = \quad \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{3.11}$$

We will meet our goal of finding $\mathbf{A}$ and $\mathbf{r}$ of Figure 3.1 with the following:

$$\mathbf{r} = \mathbf{U}\mathbf{q} = \mathbf{U}\mathbf{C}\mathbf{r} = \mathbf{U}\,\mathbf{C}\,\mathbf{E}\,\mathbf{y} = \mathbf{A}\,\mathbf{y} \tag{3.12}$$

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 (3.10) leads to $\mathbf{r}\mathbf{r}^* = \mathbf{U}\ \mathrm{E}[\mathbf{q}\mathbf{q}^*]\ \mathbf{U}^* = \mathbf{U}\mathbf{I}\mathbf{U} = \mathbf{I}$, which says the energy stays constant as we sweep through θ.

3.3.1 Finding the angle of maximum sparsity (minimum entropy)

Given any angle θ for Equation (3.11), 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:

$$\text{Entropy}(\theta) \quad = \quad \frac{\sum_t^\infty |r_1(t)| + |r_2(t)|}{\sqrt{\sum_t^\infty r_1^2(t) + r_2^2(t)}} \tag{3.13}$$

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:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad = \quad \begin{bmatrix} -\cos\theta \\ \sin\theta \end{bmatrix} \tag{3.14}$$

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.

3.3.2 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)`.

3.3.3 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:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.15}$$

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:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.16}$$

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.

3.4 RESULTS OF KAIWEN WANG

Figure 3.2 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 3.3, 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

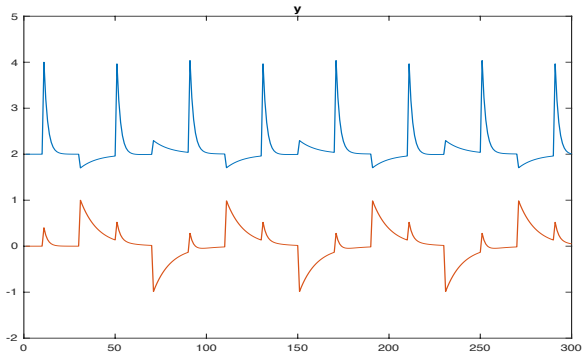

Figure 3.2: Synthetic data input is vertical and horizontal components. Model is a mix of sharp, unipolar P waves and S waves of lower frequency with alternating polarity. Stronger P waves on the vertical, and stronger S waves on the horizontal. (Kaiwen Wang) vector/. y-cropped

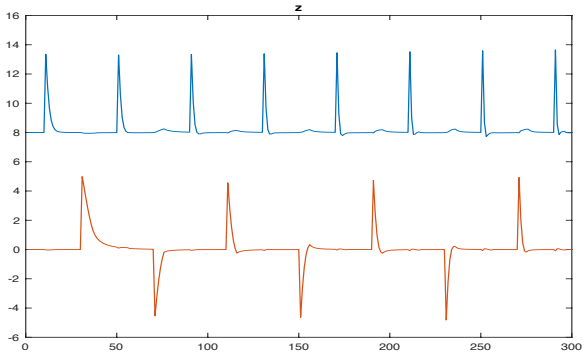

Figure 3.3: Output results: Deconvolved P wave on vertical component (top), S on horizontal (bottom). Spiking improves with time. (Kaiwen Wang) vector/. z-cropped

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 3.4, 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 3.5 is like Figure 3.4 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 3.4—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.

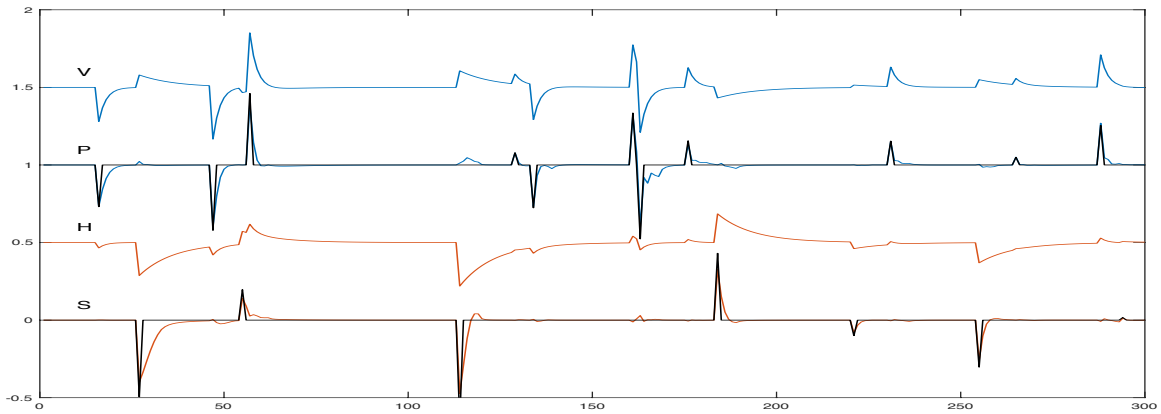

Figure 3.4: V=vertical, H=horizontal. The traces P and S are overlays of the original impulsive waves and their attempted reconstruction from (V,H). The pulses get sharper with time as the PEFs adapt. (Kaiwen Wang) vector/. tracesOrdered-cropped

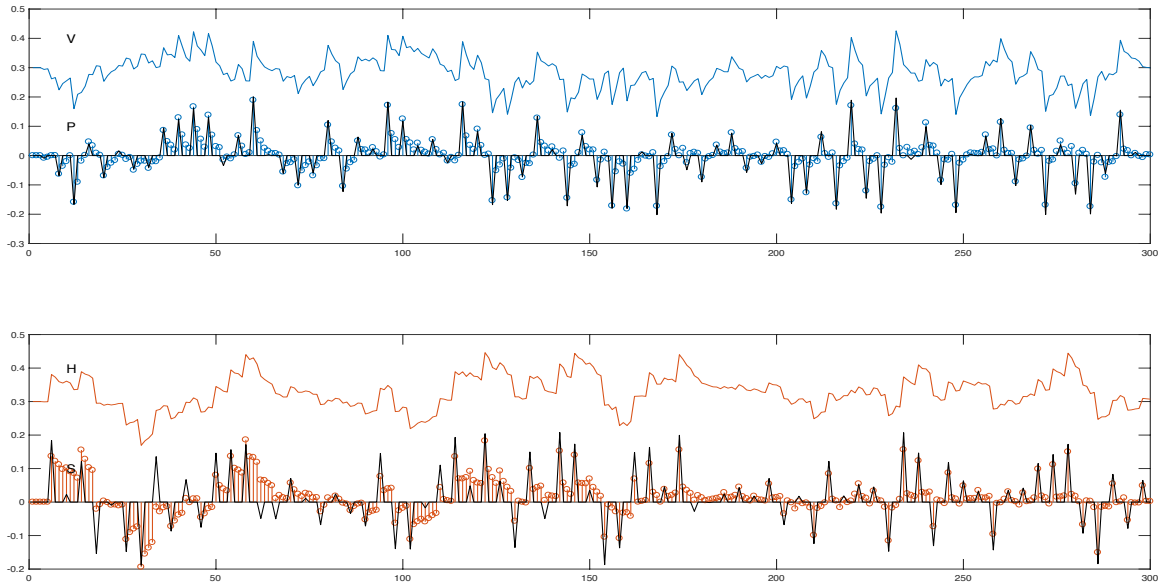

Figure 3.5: The top panel refers to the vertical motions V and the pressure waves P. The second signal in that panel is a superposition of the sparse original impulses (tiny circles) that made the data and the pulses as estimated by the entire process. These should match. They mostly do match, but small 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 4

Universal problems in Geophysics

Until now, we have limited our operators to convolutions. In reality, Physics gives us many other operators, say $\mathbf{F}$, in which we are fitting $\mathbf{0} \approx \mathbf{r}(\mathbf{m}) = \mathbf{Fm} - \mathbf{d}$. While pursuing such a fitting, we should also be finding a PEF $\mathbf{A}$ to achieve IID residuals, so we should be fitting $\mathbf{0} \approx \mathbf{q}(\mathbf{m}) = \mathbf{A}(\mathbf{Fm} - \mathbf{d})$ simultaneously finding $\mathbf{m}$ while finding $\mathbf{A}$. Beyond this problem is a second problem that for logistical reasons, Geophysical field data $\mathbf{d}$ often fails to occupy a regular grid, mostly because many locations on the earth are not available for measurements. For this second problem we assemble a process to create pseudo data on a regular grid from the reality of "sprinkled" data.

4.1 UPDATING MODELS WHILE UPDATING THE PEF

Let the misfit of theoretical data $\mathbf{Fm}$ to field data $\mathbf{d}$ define the raw residual $\mathbf{r} = \mathbf{Fm} - \mathbf{d}$. Statistical principles assert we should put $\mathbf{r}$ through a PEF $\mathbf{A}$ before minimizing some norm of $\mathbf{q} = \mathbf{Ar}$. So, we minimize $\mathbf{q}(\mathbf{m}) = \mathbf{A}\,(\mathbf{Fm} - \mathbf{d})$. For the special case $\mathbf{m} = \mathbf{0}$, this regression $\mathbf{0} \approx \mathbf{Ad}$ is none other than the PEF problem that we solved in earlier chapters. The energy E in the residual $\mathbf{q}(\mathbf{m})$ is expressed as:

$$E \;=\; \mathbf{q}\cdot\mathbf{q} \;=\; \mathbf{q}^*\,\mathbf{q} \;=\; (\mathbf{m}^*\mathbf{F}^* - \mathbf{d}^*)\mathbf{A}^*\mathbf{A}(\mathbf{Fm} - \mathbf{d})). \tag{4.1}$$

The model update direction is the negative of the energy gradient.

$$-\,\Delta\mathbf{m} \quad = \quad \frac{\partial E}{\partial \mathbf{m}^*} \quad = \quad \mathbf{F}^*\mathbf{A}^*\mathbf{A}(\mathbf{Fm} - \mathbf{d}) \quad = \quad \mathbf{F}^*\mathbf{A}^*\mathbf{Ar}. \tag{4.2}$$

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

$$\mathbf{r} \;=\; (\mathbf{Fm} - \mathbf{d}) \tag{4.3}$$

$$\mathbf{q} \;=\; \mathbf{A}(\mathbf{Fm} - \mathbf{d}) \;=\; \mathbf{A}\,\mathbf{r} \tag{4.4}$$

$$\mathbf{s} \;=\; \mathbf{A}^*\mathbf{A}(\mathbf{Fm} - \mathbf{d}) \;=\; \mathbf{A}^*\mathbf{q} \;=\; \mathbf{A}^*\mathbf{Ar} \tag{4.5}$$

$$-\Delta\mathbf{m} \;=\; \mathbf{F}^*\mathbf{A}^*\mathbf{A}(\mathbf{Fm} - \mathbf{d}) \;=\; \mathbf{F}^*\mathbf{s} \tag{4.6}$$

Equations above are in code below for computing $\mathbf{s} = \mathbf{A}^*\mathbf{Ar}$ while finding $\mathbf{A}$.

Regularization augments the data fitting penalty (4.1) with another PEF $\mathbf{B}$ for the regularization $\epsilon^2\mathbf{m}^*\mathbf{B}^*\mathbf{Bm}$. The role of $\mathbf{B}^*\mathbf{B}$ resembles that of an inverse Hessian.

4.1.1 Applying the adjoint of a streaming filter

We often think of adjoint filtering as running the filter backward on the time or space axes. That view arises with recursive filters in which the adjoint must indeed run backward. With nonrecursive filters, such as the prediction error filter, there is a more basic view. In a (nonrecursive) linear operator code, the inputs and outputs can simply be exchanged to produce the adjoint output. For example, the following pseudocode applies a PEF `a(tau)` to the physical residual `r(t)` to get a statistical (whitened) residual `q`. We get the adjoint by the usual process of swapping spaces getting `s`. The time `t` loop could run forward or backward.

```
#                            CODE = CONVOLUTION AND ITS ADJOINT
do t= ntau, nt
    do tau = 0, na
        if( forward operator )
                      q(t)     += r(t-tau) * a(tau)    # one output q(t) pulls many
        if( adjoint )
                      s(t-tau) += q(t)     * a(tau)    # one input  q(t) pushes many
```

4.1.2 Code for applying $\mathbf{A}^*\mathbf{A}$ while estimating $\mathbf{A}$

```
#                  CODE = DATA FITTING WITH PEFed RESIDUALS.
a(*) = 0; da(*) = 0;  a(0) = 1.
r(*) = 0;  q(*) = 0;  s(*) = 0                              # You compute r=Fm-d.
do t= ntau, nt
        do tau = 0, ntau
                      da(tau)   =  0
                      q(t)     +=  a(tau)  * r(t-tau)    #  q = A r
        do tau = 0, ntau
                      da(tau)  +=  q(t)    * r(t-tau)    # da = q r
        do tau = 0, ntau
                      s(t-tau) +=  q(t)    * a(tau)      #  s = A' A r
        do tau = 1, ntau
                      a(tau)   -=  da(tau) * epsilon     # Update the filter
                                                         # You apply F' to s
```

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.

4.1.3 Streaming

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. For the overall process, streaming depends on the physics operator $\mathbf{F}$ allowing it.

4.2 REGRIDDING: INVERSE INTERPOLATION OF SIGNALS

Figure 4.1 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 $t \rightarrow z$).

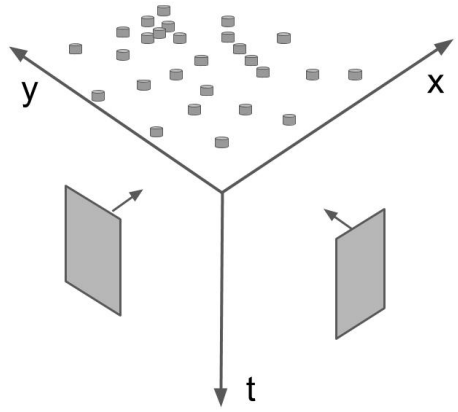

Figure 4.1: Please, pretty please, build me a dense uniform grid on the Earth surface (x, y) plane. From that grid, I want to draw by interpolation my observed data sprinkled in the (x, y) plane. Those two gray boxes must be magic 2-D PEFs. They sweep through the entire volume, updating themselves as 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 `x=x0+ix*dx`. Components of the observed signal data $\mathbf{d}$ each have with them a location $\mathbf{x}_d$, namely `xx(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 `dd` and `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)
```

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."

4.2.1 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:	$\mathbf{m}_1 = \texttt{Random}$	trial model
3:	$\mathbf{d}_1 = \mathbf{L}\mathbf{m}_1$	trial data
4:	$\mathbf{m}_2 = \mathbf{L}^* \, \mathbf{d}_1$	first guess model
5:	$\mathbf{m}_3 = \alpha \, \mathbf{m}_2$	second guess, but α unknown
6:	$\mathbf{r} = \mathbf{m}_1 - \mathbf{m}_3$	test model residual
7:	$\mathbf{r} = \mathbf{m}_1 - \alpha \, \mathbf{m}_2$	test model residual, but α unknown
8:		$0 \;=\; d(\mathbf{r} \cdot \mathbf{r})/d\alpha$
9:		$0 \;=\; \mathbf{m}_2 \cdot (\mathbf{m}_1 - \alpha \mathbf{m}_2)$
10:	$\alpha = (\mathbf{m}_2 \cdot \mathbf{m}_1)/(\mathbf{m}_2 \cdot \mathbf{m}_2)$	α is now a known property of $\mathbf{L}$.
11:	**Initialization**	
12:	$\mathbf{m} = \alpha \, \mathbf{L}^* \, \mathbf{d}$	estimated $\mathbf{m}$ assuming $\alpha, \mathbf{L}^*$ and raw data $\mathbf{d}$
13:	$\hat{\mathbf{d}} = \mathbf{L}\mathbf{m}$	estimated data
14:	$\mathbf{r} = \mathbf{d} \;-\; \hat{\mathbf{d}}$	residual is raw minus estimated data
15:	$\mathbf{r} = \mathbf{d} \;-\; \mathbf{L}\mathbf{m}$	Given any $\mathbf{m}$, this is the residual update rule.
16:	**Iteration**	
17:	$\mathbf{r} = \mathbf{d} \;-\; \mathbf{L}\mathbf{m}$	residual update rule
18:	$\mathbf{m} \leftarrow \mathbf{m} \;+\; \epsilon_d \alpha \mathbf{L}^* \mathbf{r}$	use data to expand the model (fitting)
19:	$\mathbf{m} \leftarrow \mathbf{m} \;-\; \epsilon_m \mathbf{A}\mathbf{m}$	use PEF to shrink the model (regularizing)

Regularization being the flat-earth model

To see energy spreading out from a signal to surrounding model space locations, take the PEF $\mathbf{A}$ to be simply the space derivative dm/dx. We may call $dm/dx \approx 0$ the "flat-earth" fitting goal.

$$\mathbf{r} \;\leftarrow\; \mathbf{d} \;-\; \mathbf{L}\mathbf{m} \tag{4.7}$$

$$\mathbf{m} \;\leftarrow\; \mathbf{m} \;+\; \epsilon_d \, \alpha \, \mathbf{L}^* \, \mathbf{r} \;-\; \epsilon_m \, \frac{d}{dx} \, \mathbf{m} \tag{4.8}$$

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

Once the preceeding 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 **d** might be dipping layers. Some layers would be thin (short on the time axis) others fat; some steeply dipping, some gentle. On **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 18 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:

$$\mathbf{m} \quad \leftarrow \quad \mathbf{m} \;+\; \epsilon_d \,\alpha\, \mathbf{L}^* \,\mathbf{r} \;-\; \epsilon_m \,\mathbf{A}\mathbf{m} \tag{4.9}$$

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 dominent 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 of updating the PEF $\mathbf{A}$.)

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 $\mathbf{L}^*$.

A wistful note: The 2-D PEF captures two frequencies, while many applications are concerned only with their ratio. In seismology's (x, t) space the frequency ratio is the velocity.

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 (4.9) 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 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:

$$\mathbf{m} \quad \leftarrow \quad \mathbf{m} \;+\; \epsilon_d \, \alpha \, \mathbf{L}^* \, \mathbf{r} \;-\; \epsilon_m \, [\frac{d}{dx} \;\; \frac{d}{dy}] \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix} \mathbf{m} \tag{4.10}$$

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:

$$\mathbf{m} \quad \leftarrow \quad \mathbf{m} \;+\; \epsilon_d \, \alpha \, \mathbf{L}^* \, \mathbf{r} \;-\; \epsilon_m \, [\mathbf{A} \;\; \mathbf{B}] \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix} \mathbf{m} \tag{4.11}$$

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 4.1.

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 (4.9) will be amazed when they first encounter results of Equation (4.11). 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.

4.2.2 Is the theory worthwhile?

Questions

1. Are these really *universal* problems in Geophysics?

2. Is the foregoing method of data infill worthwhile?

3. At each iteration, PEFs go one direction. Should they be going opposite ways on alternate iterations?

4. What are conventional methods of infill?

5. We could average nearby data signals weighted inversely with distance. What is wrong with that?

6. When traces have dipping events on them, how should they be interpolated?

7. Near a growth fault various dips may be simultaneously present. Does it work?

8. Does the new method act like dip or azimuthal moveout?

9. On a very dense grid in model space, we could nearest neighbor the data, then something...

Answers

No, satellite data has good spatial coverage. I don't know. I don't know. I dunno. I dunno. I dunno. I dunno. I dunno. Looks like we are not going to find out, either.

4.2.3 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 maned 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.

4.2.4 Daydreams

I like to daydream about equation (4.9) 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 $(\mathbf{Fm}-\mathbf{d})$ seems central to our efforts. Of course we need to flatten the Earth sphere giving 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 $softclip(\mathbf{Am})$ 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. Everywhere there are stories to be told, half geological, and half about data acquisition limitations. Awesome! Let your imagination run.

Chapter 5

Appendices

5.1 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.

5.1.1 Why 1-D PEFs have white output

[1]The 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_τ. The PEF output is as follows:

$$\mathbf{r} = \mathbf{d} + a_1 Z^1\mathbf{d} + a_2 Z^2\mathbf{d} + a_3 Z^3\mathbf{d} + \cdots \tag{5.1}$$

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

$$0 = \frac{d}{da_\tau}(\mathbf{r}\cdot\mathbf{r}) \tag{5.2}$$

$$0 = \mathbf{r}\cdot\frac{d\mathbf{r}}{da_\tau} = \mathbf{r}\cdot Z^\tau\mathbf{d} \qquad \text{for } \tau > 0. \tag{5.3}$$

Given that $0 = \mathbf{r}\cdot Z^\tau\mathbf{d}$, we examine $\mathbf{r}\cdot Z^\tau\mathbf{r}$ and see that it also vanishes. Using Equation (5.1), we have for any autocorrelation lag $k > 0$,

$$\mathbf{r}\cdot Z^k\mathbf{r} = \mathbf{r}\cdot(Z^k\mathbf{d} + a_1 Z^{k+1}\mathbf{d} + a_2 Z^{k+2}\mathbf{d} + ...)$$

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

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

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.

5.1.2 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 down by 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.

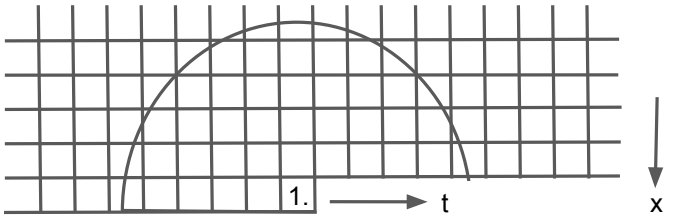

Figure 5.1: The "1." at the end of a 1-D rope wrapped on a silo. We consider only the filter coefficients inside the semicircle, outside coefficients supposedly negligible. appendix/. ropeEnding

Figure 5.1 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." In principle all the coefficients outside the semicircle vanish. For coding convenience, the nonvanishing 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.

5.2 THE HEART OF NONSTATIONARY PEF USING CALCULUS

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

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \approx \left[\begin{array}{cccc} d_{99} & d_{98} & d_{97} & d_{96} \\ \hline \gamma & \cdot & \cdot & \cdot \\ \cdot & \gamma & \cdot & \cdot \\ \cdot & \cdot & \gamma & \cdot \\ \cdot & \cdot & \cdot & \gamma \end{array}\right] \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} - \gamma \left[\begin{array}{c} 0 \\ \hline 1 \\ \bar{a}_1 \\ \bar{a}_2 \\ \bar{a}_3 \end{array}\right] \tag{5.4}$$

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 (5.4) as follows:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} d_n & d_{n-1} & d_{n-2} \\ \gamma & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} - \begin{bmatrix} -d_{n+1} \\ \gamma\,\bar{a}_1 \\ \gamma\,\bar{a}_2 \\ \gamma\,\bar{a}_3 \end{bmatrix} \tag{5.5}$$

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

$$\mathbf{0} \approx \mathbf{r} = \begin{bmatrix} \mathbf{d}^* \\ \gamma\,\mathbf{I} \end{bmatrix} \mathbf{a} - \begin{bmatrix} -d_{n+1} \\ \gamma\,\bar{\mathbf{a}} \end{bmatrix}, \tag{5.6}$$

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.

$$\Delta\mathbf{a} = -(\text{some constant}) \left.\frac{\partial}{\partial \mathbf{a}^*}\right|_{\mathbf{a}=\bar{\mathbf{a}}} \mathbf{r}^*\mathbf{r} \tag{5.7}$$

Form the transpose of the Residual (5.6) and then, differentiate by $\mathbf{a}^*$. (By $\mathbf{a}^*$, we mean the complex conjugate transpose of $\mathbf{a}$.)

$$\frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*} = \frac{\partial}{\partial \mathbf{a}^*} \{\mathbf{a}^*[\mathbf{d} \quad \gamma\mathbf{I}] - [-d_{n+1} \quad \gamma\bar{\mathbf{a}}]\} = [\mathbf{d} \quad \gamma\mathbf{I}] \tag{5.8}$$

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

and multiply that onto $\mathbf{r}$ from Equation (5.6) keeping in mind that $\mathbf{d}^*\bar{\mathbf{a}}$ is a scalar.

$$\Delta\mathbf{a} \quad \propto \quad \frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*} \quad \mathbf{r} \quad = \quad [\mathbf{d} \quad \gamma\mathbf{I}] \left\{ \begin{bmatrix} \mathbf{d}^* \\ \gamma\,\mathbf{I} \end{bmatrix} \mathbf{a} \; - \; \begin{bmatrix} -d_{n+1} \\ \gamma\,\bar{\mathbf{a}} \end{bmatrix} \right\} \tag{5.9}$$

$$= \quad \mathbf{d}(\mathbf{d}^*\mathbf{a}) + \gamma^2\mathbf{a} + \mathbf{d}d_{n+1} - \gamma^2\bar{\mathbf{a}} \tag{5.10}$$

$$\Delta\mathbf{a} \quad \propto \quad \frac{\partial \mathbf{r}^*}{\partial \mathbf{a}^*}\bigg|_{\mathbf{a}=\bar{\mathbf{a}}} \mathbf{r} \quad = \quad (\mathbf{d}^*\bar{\mathbf{a}} + d_{n+1})\,\mathbf{d} \tag{5.11}$$

$$\Delta\mathbf{a} \quad = \quad -\,\epsilon\, r_t\,\mathbf{d} \tag{5.12}$$

It is certainly surprising that the analytic solution to the Regression (5.4) computationally amounts to a single step of the optimization strategy (5.11), a strategy so crude as to be absent from textbooks; yet true (Fomel et al., 2016). Experimentalists first notice that Equation (5.4) demands we supply a not-given constant γ while (1.3) or (5.12) 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.