Contents

0.1 ABSTRACT ........................................................................................................ i
0.2 PREFACE .......................................................................................................... i
0.3 INTRODUCTION .............................................................................................. i

1 Nonstationary scalar signals .............................................................................. 1
  1.0.1 Why should residuals be IID? ................................................................. 1
  1.0.2 Prediction-error filtering (deconvolution) ............................................. 1
  1.0.3 The heart of a nonstationary PEF without serious math .................. 2
  1.0.4 Prediction-error code ............................................................................... 3
  1.0.5 Why does the residual into the adjoint give the gradient? ............... 4
  1.0.6 The outside world ................................................................................. 4
  1.0.7 Some application aspects of these fundamentals ......................... 5
  1.0.8 Hyperbolic penalty function .................................................................. 5
  1.0.9 Other norms and penalties .................................................................... 6
  1.0.10 Simultaneous estimation of the PEF and the missing data ............ 6
  1.0.11 Old 1-D examples I've done in the stationary world ................... 7
  1.0.12 Epsilon ................................................................................................. 8
  1.0.13 How can the nonstationary PEF operator be linear? ..................... 9
  1.0.14 Sparse decon ....................................................................................... 10

2 Spatial deconvolution ....................................................................................... 13
  2.0.15 Averaging in time and space ............................................................... 14
  2.0.16 Example of a PEF applied to a seismic section .............................. 16
  2.0.17 Opportunities in a time-variable epsilon .......................................... 17
  2.0.18 Rapid recognition of a spectral change ............................................ 17
  2.0.19 Boundaries between regions of constant spectrum ..................... 18
## CONTENTS

2.0.20 What physical phenomena gives the spectra of a 3-D PEF? 19

3 Fitting while whitening residuals 21

3.0.21 Applying the adjoint of a streaming filter 21
3.0.22 Applying the PEF with its adjoint 22
3.0.23 PEFs for both fitting and styling 23
3.0.24 Signal regridding 23

4 Multi channels = Vector-valued signals 25

4.0.25 No time delays please 26
4.0.26 Vector signal scaling 27
4.0.27 Pseudo code for vector signals 28
4.0.28 The PEF output is orthogonal to its inputs 29
4.0.29 Restoring source spectra 29

4.1 CHOLESKY DECORRELATING AND SCALING 29

4.2 ROTATING FOR SPARSITY 30

4.2.1 Finding the angle of maximum sparsity (minimum entropy) 31
4.2.2 3-component vector data 32
4.2.3 Channel order and polarity 32

4.3 RESULTS OF KAIWEN WANG 32

5 Appendices 35

5.1 WHY PEFs HAVE WHITE OUTPUT 35

5.1.1 Why 1-D PEFs have white output 35
5.1.2 Why 2-D PEFs have white output 36

5.2 DERIVATIVE OF A NEW DATA RESIDUAL BY A PEF 38
Front matter

0.1 ABSTRACT

This instructional booklet teaches geophysical (especially acoustic and seismic) image and voxel model building focusing on codes for nonstationary (time-variable statistics) data signals. Signals and signal arrays, both scalar and vector-valued, are attacked with prediction-error filters to develop codes for whitening residuals in geophysical model building, for filling gaps in data, and for compensating irregular recording locations.

0.2 PREFACE

My previous 2014 book GIEE (Geophysical Image Estimation by Example) is widely referred to herein. It 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.

Alfa and early beta versions of this booklet will fail to provide rebuildable illustrations. I’m no longer coding myself, so if there are ever to be rebuildable illustrations, I will need coauthors. I set myself the goal to take it out of beta when 50% of the illustrations can be destroyed and rebuilt by readers.

Putting this out at my 80th birthday I am not planning another book, but since 2014 I stumbled on an approach to much geophysical data fitting that is greatly simpler than traditional approaches. Even better, it avoids the often unreasonable academic presumption of stationarity (time invariant statistics). Hence I embarked on this booklet. Any participant is welcome to contribute illustrations (and ideas) perhaps becoming a coauthor, even eventually taking it over. The first need now is more examples. Ultimately, all the examples should be presented in rebuildable form.

0.3 INTRODUCTION

In the nonstationary approach we pretend to have already solved an imaging problem, but a newly arrived data value requires us to make a small adjustment to it. The stationary approach is to assemble signals and gather their statistics. Then apply those statistics to old and new data. Both approaches use the non-predictability of data as the statistical background for more complicated model building, especially image and voxel model building.
Although we begin here narrowly with 1-D scalar signals $y_t$, we soon expand broadly to $y_t(x, y, z)$ with multidimensional data (images and voxels) and then multicomponent signals $\mathbf{\tilde{y}}_t$ (i.e. vector valued).

Most of the time-series literature presumes data spectra are time invariant (aka stationary) while most applications involve spectra that change in time and space. Hence there is a strong need for nonstationary tools. These capabilities are particularly important in higher dimensional problems, slightly in time, more strongly in space. A basic problem in geophysics is lack of adequately dense data sampling in space, and wide-spread nonstationarity there too. Such nonstationarity arises both for statistical reasons, and for physical reasons like dip versus offset.

Sergey Fomel triggered this direction of research when he solved the nonstationarity problem I had posed but could not solve. Bob Clapp ran an inspiring summer research group. Stew Levin provided me with 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 commented on early versions of the multichannel chapter. Jason Chang assisted me with LaTeX. Thanks to all.
Chapter 1

Nonstationary scalar signals

It’s a principle of statistics (and common sense) that residuals in model building should be IID (Independent, Identically Distributed). In seismic signal practice the “ID” means gain should be chosen to give residuals a generally uniform variance over physical space; it is simply a matter of scaling. The remaining “I” in IID means “independent,” our topic here. Autocorrelations should tend to impulses meaning spectra should tend to whiteness (uniform variance), these accomplished here by prediction-error filtering (to be defined).

1.0.1 Why should residuals be IID?

With least squares fitting, small residuals are squared, so they tend to be ignored. The entire physical space and fourier space of residuals should be scaled up to easily visible levels to ensure all aspects of the data have been probed. The statistical principle that model fitting residuals should be IID says mostly every image estimation situation calls for using PEFs (Prediction Error Filters), to be defined next.

1.0.2 Prediction-error filtering (deconvolution)

Start with data (a signal or signals of many thousands of values). In mathematics we denote these data numbers \( y = (y_0, y_1, y_2, \ldots, y_\infty) \). In pseudo code they are \( y(it) = [y(1), y(2), y(3), \ldots, nt] \). (In signal analysis, subscripts begin from zero. In code fragments here, subscripts begin from one.) Take a little patch of numbers that are not data. Mathematically we’ll call this patch \( a = (a_0, a_1, a_2, \ldots, a_9) \), with pseudo code referring to it as \( a(1), a(2), \ldots, a(10) \). This patch may be called a filter. This patch of numbers will slide across the data numbers. 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.

An equation for sliding the filter numbers across the data numbers obtaining the output \( r_t \) is \( r_t = \sum_{\tau=0}^{9} a_\tau y_{t-\tau} \). Pseudo code for finding \( r(t) \) is:
Before we finish they will become time functions in an N-dimensional physical space like the four component array $y_t(x, y, z)$. Conceptually beyond that, we 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 will be vector functions like $\mathbf{r}_t = (u_t, v_t, w_t)$, where the PEF begins not with a 1.0 but with a three component identity matrix $I$.

### 1.0.3 The heart of a nonstationary PEF without serious math

At any one moment in time we may think of the PEF output $\sum_{\tau} a_{\tau} y_{t-\tau}$ 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 will have other values in the $d$ vector.) At that moment the PEF output is $a \cdot d$. Consider the exploratory filter $a + \epsilon d$. Its output $r_t$ would be $(a + \epsilon d) \cdot d = (a \cdot d) + \epsilon (d \cdot d)$. To reduce the new output $r_t$ these two terms must have opposite polarity, but $r_t = (a \cdot d)$ may have either polarity. Try instead the filter update $(a - \epsilon r_t d)$. Its output is $(a \cdot d) - \epsilon r_t (d \cdot d)$ which on rearrangement is $(a \cdot d)(1 - \epsilon (d \cdot d))$, easily assured smaller than $r_t = (a \cdot d)$. Thus

$$\Delta a = -\epsilon r_t d = -\epsilon r_t y_{t-\tau}$$

(1.2)
In summary:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Definition</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_t = \sum_{\tau} a_{t-\tau} y_{t-\tau} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( a = a_t )</td>
<td>( d = y_{t-\tau} )</td>
<td>( r_t = a \cdot d )</td>
</tr>
<tr>
<td>( a + \epsilon \cdot d )</td>
<td>First trial</td>
<td>( a \cdot d + \epsilon (d \cdot d) )</td>
</tr>
<tr>
<td>( a - \epsilon r_t d )</td>
<td>Revision</td>
<td>( a \cdot d - \epsilon r_t (d \cdot d) )</td>
</tr>
<tr>
<td>( a + \Delta a )</td>
<td>Success!</td>
<td>((a \cdot d)(1 - \epsilon (d \cdot d)))</td>
</tr>
</tbody>
</table>

The polarity of components of \( d \) are all that matters. Their magnitude does not. We can relate this conclusion to basic mathematics. Take anyone’s favorite penalty function. Take its gradient \( g \) of the newest data value residual with respect to the PEF coefficients \( a \). One characteristic of the gradient of any penalty function is that moving along any gradient component increases the penalty. (Its negative decreases penalty). Thus, the result of your mathematical exercise must conclude with components of \( g \) that are sign-preserving functions of components of \( d \) such as \( d \) itself. For example we could update with \( \Delta a = \epsilon(r_t/|r_t|)d \) which has an \( \ell_1 \)-norm flavor. An appendix has a two-page calculation of the derivative showing that the \( \ell_2 \)-norm gradient of including a new data value \( y_t \) is \( \Delta a_{t-\tau} = -\epsilon r_t y_{t-\tau} \), namely, our equation (1.2). Penalty functions need not be convex(!), but they should have a single unique minimum. Convex functions do not have banana-shaped contours, a problem for many methodologies, but not here.

1.0.4 Prediction-error code

In 1-D filtering applications the universal need for IID residuals gives much value to this code. It is called “deconvolution.”

```plaintext
r(...) = 0.  # CODE = NONSTATIONARY PREDICTION ERROR
a(...) = 0.
a( 1 ) = 1.0

do over time t {# r(t) = nonstationary prediction error.
    do tau= 1, na
        da(tau) = 0
        r(t) += a(tau) * y(t-tau+1)  # forward
    da(tau) += r(t) * y(t-tau+1)  # adjoint
    da(1) = 0.  # constraint
    do tau= 1, na
        a(tau) -= da(tau) * epsilon
    }
```

The `#forward` line in the code above 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 above, based on little more than the definition of dot product, is a demonstration 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 \( y(t-tau+1) \) 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 \( M_{ij}^* = M_{ji} \). The line `da(1)=0` is a constraint to prevent changing the \( a(1)=1 \) 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
CHAPTER 1. NONSTATIONARY SCALAR SIGNALS

the gradient; then go down. We got the gradient without ever calculating a derivative! If coding adjoints is new to you, I recommend chapter 1 in GIEE (Claerbout, 2014). It’s free on the internet.

Suppose while running along time \( t \) we find the line in the code above computing \( \Delta a \) saying for practical purposes \( da(tau) = r(t) \cdot y(t - tau + 1) \) vanishes for all \( tau > 1 \). This statement would delight any stationary theorist because the gradient vanishing \( \Delta a = 0 \) says we’re finished. Equivalently, it says the residual \( r \) is orthogonal to all the fitting functions \( y_\tau = y_{t - \tau} \). (A particular fitting function is \( y_{t - 9} \).) With our nonstationary technology, we don’t expect \( \Delta a \) ever to be exactly zero, but we do expect it to get small and then bounce around. The fluctuation in size of \( |\Delta a| \) 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 \( \epsilon \), how much bouncing around to expect, or how to characterize nonstationarity, but we will come up with a good starting guess for \( \epsilon \). You can learn by experience while I speculate on epsilon varying with time, and speculate more wildly on epsilon being data dependent in a 2-D space.

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

Basic geophysical model \( m \) estimation is summarized by the residual minimization \( 0 \approx r(m) = Fm - d \). When \( F \) is a convolution matrix (downshifted columns of data \( d \)), this formulation fits the estimation of a prediction filter \( m \). But, for most applications we want a prediction-error filter \( a \). Thus we think of PEF estimation as the constraint \( a_0 = 1 \) along with the augmented matrix \( Y = [d | F] \) where \( Y \) is also a convolution matrix. PEF estimation coding is repetitions of \( r = Ya \), followed by \( \Delta a = -Y^*r \), followed by the constraint \( \Delta a_0 = 0 \), followed by \( a = a + \epsilon \Delta a \).

This tutorial document shows five codes for diverse applications of PEFs, so let’s be sure that everyone is on board 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 \( 2 \times 3 \) matrix. In short, the quadratic form you are minimizing is \( r \cdot r = (m^*F^* - d^*)(Fm - d) \) with derivative by \( m^* \) being \( \Delta m = F^*r \). Likewise, the derivative of \( a^*Y^*Ya \) by \( a^* \) is \( \Delta a = Y^*r \).

1.0.6 The outside world

This approach applies to all “infinitely tall” regression equations. It’s not limited to convolutional matrices. For each new regression row, add a tiny suitably-scaled copy of the new row into the solution. Move along; keep doing it. When you run out of equations, you can recycle the old ones. Cycling around an infinite number of times with an epsilon tending to zero you converge to the stationary solution. This should be some long-known principle in mathematics. I have stumbled upon something called the \textit{Widrow-Hoff learning rule} which feels just like this.

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 the square footages, the third column the number of bathrooms, etc. Since 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 code above finds \( a_i \) coefficients to (negatively) predict the signal. Associating lags with real-estate aspects, the code would predict (the negative of) (maybe the logarithm of) the sales price. You’ve made the first step towards a learning machine.

### 1.0.7 Some application aspects of these fundamentals

Consider weighting functions such as \( d^\top Wd \) where \( W \) is a positive definite matrix. Take a diagonal \( W \) with some components vanishing. Those represent constraints that some filter coefficients remain zero. Think “gapped filter,” or “debubble.”

Here is another example: In reflection seismology \( t^2 \) gain and debubble do not commute. Do the physics right by applying debubble first; get a bad answer (because late data has been ignored). Do the statistics right; apply gain first; 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 \( \epsilon \).

### 1.0.8 Hyperbolic penalty function

Most people do most data fitting by minimizing the sum of the squared residuals. This is called “least squares” or the \( \ell_2 \)-norm approach. Computations are generally easy but a single outlandish residual ruins everything. The \( \ell_1 \)-norm approach minimizes the sum of absolute values of residuals. The advantage of \( \ell_1 \) is that occasional infinite residuals detract little from the solution. Thus, \( \ell_1 \) is described as “robust”.

GIEE has many examples of practical use of the hyperbolic penalty function. Loosely, we call it \( \ell_h \). For small residuals it is like \( \ell_2 \), and for large ones it is like \( \ell_1 \). Happily, the nonstationary approach allows easy mixing and switching among norms. Hooray!

A marvelous feature of \( \ell_1 \) and \( \ell_h \) emerges on model space regularizations. They encourage models to contain many zero or small values leaving the essence of the model in a small number of locations. Hence sparse models, the essence of Occam’s razor. Wow! In

<table>
<thead>
<tr>
<th>name</th>
<th>scalar residual</th>
<th>scalar penalty</th>
<th>scalar gradient</th>
<th>vector penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_2 )</td>
<td>( q )</td>
<td>( q^2/2 )</td>
<td>( q )</td>
<td>( q )</td>
</tr>
<tr>
<td>( \ell_1 )</td>
<td>( q )</td>
<td>(</td>
<td>q</td>
<td>)</td>
</tr>
<tr>
<td>( \ell_h )</td>
<td>( q = r/\bar{r} )</td>
<td>( (1 + q^2)^{1/2} - 1 )</td>
<td>( q/(1 + q^2)^{1/2} )</td>
<td>( \text{softclip}(q) )</td>
</tr>
</tbody>
</table>

From the table, observe at large \( q \), \( \ell_h \) tends to \( \ell_1 \). At small \( q \), \( \ell_h \) tends to \( q^2/2 \) which matches \( \ell_2 \). To see a hyperbola \( h(q) \), set \( h - 1 \) equal to the scalar penalty in the table, getting \( h^2 = 1 + q^2 \). Coding requires a model gradient \( \Delta m \) or \( \Delta a \) which you form by putting the vector penalty into the adjoint of the modeling operator, then taking the negative, for example \( \Delta a = -Y^*\text{softclip}(q) \).

An attribute of \( \ell_1 \) and \( \ell_2 \) fitting is that \( \|\alpha r\| = \alpha \|r\| \). This attribute is not shared by \( \ell_h \). Technically it is not a norm; it should be called a “measure.” Results are critically dependent on the choice of \( \bar{r} \). It locates the transition between \( \ell_1 \) and \( \ell_2 \) behavior. I have often chosen \( \bar{r} \) to be at the 75th percentile of the residuals.
1.0.9 Other norms and penalties

Besides the “non mathematical” update equation (1.2) which amounts to a least-squares ($\ell_2$-norm) approach (see Appendix: Derivative of a PEF by a new data value), an $\ell_1$-norm style update is

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

With this update we might be able to deconvolve data containing a sprinkling of infinite spikes. More likely, a soft-clip function would work better than the hard-clip sgn() function in equation (1.3).

Any polarity preserving stretching functions may be applied to $r_t$ and $y_{t-\tau}$. This is clear from our original non-mathematical derivation of equation (1.2), but you may also notice the general principle that scaling axes on a penalty function leads to scaled components of gradients, not polarity changes. The main property of gradients is that each component is pulling the penalty in the desired direction. We don’t even need to know a penalty function or its derivative. For the filter update we only need polarity-preserving functions of the residuals and the data.

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.

In practice, how should we play the statistics of $r$ and $d$? I’ve never undertaken to play with with $\text{sgn}(y_{t-\tau})$, so I have little feel for its opportunities. To fuel our creative imaginations perhaps we need interactive programs to play with diverse data sets. I also need an energetic student to test my explanations and perhaps reject some of my claims. Since an infinite data value is easy to recognize, perhaps we should simply mark such a location as missing data and switch to the method of the next section.

What might be the most general formulation? When we get to vector-valued signals we might find unexpected opportunities such as vorticity in the ocean\[1]\ or Alfven waves in the ionosphere. In time, diverse applications will crop up.

1.0.10 Simultaneous estimation of the PEF and the missing data

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

“That’s too tough,” he replied.

We do it easily now by appending three lines to code above. 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 bottom are three lines for missing-data estimation. The code “looks canonical” (by sticking a residual into an adjoint), but what is it doing?

\[1\] See the youtube for “Perpetual ocean”
# CODE = ESTIMATING PEF AND MISSING DATA TOGETHER

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

Where data values are missing they are changed so that the output residual is reduced. Updates are produced by laying the PEF a(tau) onto the data allowing changes only where data is missing. I believe this code works, but actually understanding it is not as easy as verifying that a residual is stuffed into transposed modeling.

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 $\epsilon_A$, $\epsilon_Y$, and our $\Delta t$ viewing rate.

Still more fun to have this happening on images (next section). Did you know that playing with your constructions induces creative thinking? Check the book, “Lifelong Kindergarten,” by Mitchel Resnick. He also urges sharing your rebuildable projects with peers.

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 should run the time loop backwards on alternate passes. There should be some other way to organize this calculation to avoid passing over large areas that have no missing data. But, it runs so fast that perhaps no one cares about efficiency. Wizards running exotic chips with parallel architectures will appreciate the simplicity of this code.

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.

I wondered whether our missing data code would work in the wider world of applications, the world beyond mere signals. Most likely not. A missing signal value affects a regression equation for each filter coefficient while a missing home square footage affects only one regression equation.

### 1.0.11 Old 1-D examples I’ve done in the stationary world

Figure 1.1 is shows a nice test case. The result shown solved for missing data given the PEF of (1, 0, −1). When I solved for the optimal PEF (many years ago) both it and the interpolated data was much the same as shown here. The conclusion to draw is that PEF interpolation preserves the character of the given data, unlike linear or cubic interpolation.
CHAPTER 1. NONSTATIONARY SCALAR SIGNALS

Figure 1.1: Top is given data. It is taken to be known zeros off the ends of the axis. Middle is the given data with interpolated values. Thusly restored data has the character of the given data. Bottom shows the filter \((1, 0, -1)\). Its output (not shown) has minimum energy. (Claerbout)

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 the 2012 version of GIEE, on page 197. A sinusoid was sampled densely on the left and sparsely on the right. You see it nicely restores data beyond aliasing.

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

1.0.12 Epsilon

An application parameter like \texttt{epsilon} requires some practitioner to choose its numerical value. This is best rationalized by making sure \(\epsilon\) is free from physical units, so let us now attend to that. From the past of \(y\) the filter \(a\) predicts the future of \(y\), so \(a\) itself must be without physical units. The data \(y_t\) might have units of voltage, and so its prediction error \(r_t\) has the same units. To repair the units in \(\epsilon\) we need something with units of voltage squared for the denominator. Let’s call it the variance \(\sigma_y^2\). You can compute it globally for your whole data set \(y\) or you can compute it by leaky integration so it adjusts itself with the nonstationary changes in data \(y_t\). The filter update \(\Delta a\) with a unit-free \(\epsilon\) and correct polarity is

\[ \Delta a = -\frac{\epsilon r_t}{\sigma_y^2} d \] (1.4)

That’s the story for \texttt{epsilonA} in the code above. For the missing data adaptation rate, \texttt{epsilonY}, no normalization is required because \(r(t)\) and \(y(t)\) have the same physical units while \(a(tau)\) is unitless.

Epsilon \(\epsilon\) 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 augmented by \(\epsilon\) times a current estimate of it. After \(\lambda\) steps the influence of any original time is reduced by the factor \((1-\epsilon)^\lambda\). Setting that to \(1/e = 1/2.718\) says \(1/e = (1-\epsilon)^\lambda\).
Taking the natural logarithm, \(-1 = \lambda \ln(1 - \epsilon) \approx -\lambda \epsilon\), so to good approximation
\[
\epsilon = 1/\lambda
\] (1.5)

By the well known property of exponentials, half the area in the decaying signal appears before the distance \(\lambda\), the other half after.

I often think of the memory function \((1 - \epsilon)^t\) as a rectangle function of length \(\lambda\). Least squares analysis begins with the idea there should be more regression equations than unknowns. So \(\lambda\) should roughly exceed the number of filter coefficients \(na\). To avoid overfitting, I'd begin with \(\lambda = 100 \times na\).

There is a pitfall above. With synthetic data you may have runs of zero values. These don’t count as data. Then you need a bigger \(\lambda\) because the zeros don’t provide the needed information.

Mathematicians are skilled at dealing with the stationary case. They will be inclined to consider all residuals \(r_t\) to carry equal information. They may keep a running average \(m_t\) of a residual \(r_t\) by the identity (proof by induction)
\[
m_t = \frac{t - 1}{t} m_{t-1} + \frac{1}{t} r_t = \frac{1}{t} \sum_{k=1}^{t} r_k
\] (1.6)

This suggests an \(\epsilon\) decreasing proportional to \(1/t\) (which is like \(\lambda\) proportional to \(t\)) may in some instances be a guide to practice although it offers no guidance for nonstationarity other than that nonstationarity calls for an \(\epsilon\) larger than proportional to \(1/t\).

Given an immense amount of data, a “learning machine” should be able to come up with a way of choosing the adaptivity rate \(\epsilon\). 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.

1.0.13 How can the nonstationary PEF operator be linear?

Formally, finding the PEF is \(a = \arg\min_a (Ya)\) subject to \(a_0 = 1\), while using it is \(r = Ay\). The combination is a nonlinear function of the data \(y\). But it is nearly linear, even strictly linear in a certain sense. Notice that \(A\) could have been built entirely from spatially nearby data, not at all from \(y\). Then \(A\) would be nonstationary, yet a perfectly linear operator on \(y\).

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. This 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.)
1.0.14 Sparse decon

We 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 happens twice, once at the air gun (about 15 meters deep) and once again at the hydrophones yielding roughly a second finite-difference response called a “ghost”. Modelers often choose the second derivative of a Gaussian (called a Ricker wavelet) for their limited-bandwidth source wavelet. The Gaussian function is not causal (not vanishing before \( t = 0 \)). So, a more natural way to limit bandwidth is with the Futterman wavelet (GIEE), a causal representation of the spectrum \( \exp(-|\omega|t/Q) \) where \( Q \) is the quality constant of rock. See Figure 1.3.

![Figure 1.3: The causal constant Q response and its second finite difference. This explains why the water bottom could seem a Ricker wavelet (second derivative of a Gaussian) while the top of salt would seem a doublet. That third lobe is small. The first two lobes are about the same height but the middle lobe has more area. (Claerbout)](chap1/futterman.png)

Antoine Guittton [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. (Unfortunately, Antoine and I did not appreciate the Futterman effect when we did the sparse deconvolutions.) Antoine’s examples were done with a stationary theory that allows the inverse shot wavelet being slightly non-causal. Too bad PEFs lose their whiteness when we extend them noncausally. I think we should try to spike not at the wavelet onset (which is what PEFs do), but somewhere more like the center lobe of the Gaussian or the second lobe of the 2nd derivative of the Futterman.

Unfortunately, the nonstationary track we travel in this tutorial may not bring us directly to such grand results. However Antoine’s work did not extend easily to wider shot-hydrophone separations, so he needed something like our current nonstationary approach.

If we could get this all together, we could reasonably hope to make millions. Let us begin guessing! For each travel time depth \( \tau = z/v \) we need the phase correction of \( \exp(-|\omega|\tau/Q) \). Is that easy (like Stolt migration) or harder like downward continuation? Hmm.

REFERENCES

Figure 1.4: Gulf of Mexico. Top is before sparse decon, bottom after. Between 2.25-2.70s 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. (Antoine Guitton)
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. Such unambiguous geologic interpretation is worth money. (Antoine Guitton)
Chapter 2

Spatial deconvolution

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

Figure 2.1: (left) Seabeam data of mid-Pacific transform fault. (right) After interpolation by stationary 2-D PEF. (GIEE)

Signal analysis extends to image analysis quite easily except for the fact that the spike on the PEF is not on a corner of the 2-D filter array but on its side. This old knowledge is summarized in the appendix: Why 2-D PEFs have white output.

Figure 2.2: A PEF in the code is a function of lag $a(t_1,x_1)$. It is lain backwards here so we can visualize it crosscorrelating seismic data ($t$ down, $x$ to the right). (Claerbout)

Unlike the 1-D code, here we use zero and negative subscripts on time. Like in 1-D, the
CHAPTER 2. SPATIAL DECONVOLUTION

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 \((t_l, x_l)\) cover the entire filter. First the residual \( r(t,x) \) calculation (# Filter) is simply the usual 1-D convolution seen again on the 2-axis. Next the adjoint follows the usual rule of swapping input and output spaces. Next the constraint line preserves not only the 1.0, but also the zeros before it. Finally, the update line \( a -= da \) is trivial.

```plaintext
# 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
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 \( (\omega,k_x) \)-space. The local auto-correlation tends to a delta function in \((\tau,\text{lag}_x)\) space. Everybody’s 2-D image estimations need this to achieve IID residuals.

Figure 2.3 illustrates using PEF technology refilling a hole in an image of the Gulf of Mexico. This illustration (taken from GIEE) uses stationary technology.

2.0.15 Averaging in time and space

A streaming 1-D prediction filter is a decaying average of earlier prediction filters, however these earlier filters need not be all saved in memory. Since they vary smoothly we may simply use the most recent one. Call it \( \bar{a} \). In two dimensions \( \bar{a} \) becomes some average of its previous value on each of its two axes. For example instead of updating from the old filter \( a(t - \Delta t, x) \), we could update from the old filter located at \( a(t, x - \Delta x) \). 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 \( a \leftarrow \bar{a} + \Delta a \) with

\[
\bar{a} = a(t - \Delta t, x) \frac{\lambda_t^2}{\lambda_t^2 + \lambda_x^2} + a(t, x - \Delta x) \frac{\lambda_x^2}{\lambda_t^2 + \lambda_x^2}
\]

\( ^1 \text{Drawn from Fomel et al. (2016).} \)
Figure 2.3: 2-D stationary example from GIEE. CDP stack with a hole punched in it. The rightmost frame shows something not discussed here. (It should be.) Random noise into the inverse PEF is added into the gap in such a way as to assure continuity at the gap boundary. (GIEE)
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 a saved $\mathbf{a}$ for every time at the previous $x$, namely at $x - \Delta x$.

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. Don’t let that trouble you though. Since the PEFs are all smoothly variable they could be linearly interpolated from a sparse mesh. PEFs on the previous trace $\mathbf{a}(t, x - \Delta x)$ might be smoothed symmetrically on the time axis so the region of averaging expands from a quadrant to a halfspace.

Stationary decon should remove a shot waveform. Nonstationary decon starts from there but has the added possibility of removing the waveform of the outgoing wave. It evolves with traveltimes ($Q$ and forward scattered multiples). It also evolves with space, especially shot to receiver offset. We could build such nonstationary filters on either field data or synthetic data, then apply them to field data. The relations among pressure, velocity, upcoming, and downgoing waves vary systematically with offset. You could work out theoretical expressions for these relations, but instead you could see how this data fitting code would handle it.

**2.0.16 Example of a PEF applied to a seismic section**

![Figure 2.4: A CDP stack before and after 2-D decon. Its goal is to suppress predictable events. (Clapp et al)](chap2_. wgstack)
Figure 2.4 shows the world’s first “industrial scale” nonstationary 2-D prediction error. It was the principal product of a summer research group led by Bob Clapp ([Ruan et al., 2015]). It demonstrates a 2-D nonstationary deconvolution based on the idea of using a coarse mesh to store many 2-D PEFs. This data set and others at that reference are available for testing the streaming ideas of this tutorial. Give it a try!

2.0.17 Opportunities in a time-variable epsilon

Examine the logic leading up to equation (1.2). Recall that \( d \) is the chunk of data lying under the filter, so \( d \) is actually a function of time \( d_t \). The logic at equation (1.2) says we should assure \( |1 - \epsilon (d_t \cdot d_t)| < 1 \). But, for any fixed \( \epsilon \) there will surely be moments in time when \( |d_t| \) is large enough to violate this inequality. Instead of taking an \( \epsilon \) constant in time we might consider this time-variable epsilon \( \epsilon_t \).

\[
\epsilon_t = 2 \eta / (d_t \cdot d_t) \quad \text{for} \quad 0 < \eta < 1 \tag{2.2}
\]

However, since \( \epsilon_t \) may now change rapidly with time, the overall process has become a dynamical system whose behavior is difficult to anticipate, though easy enough to try out. It is even extensively more flexible. Eta \( \eta \) could be a function of lag \( \tau \). Is there an opportunity here? We don’t know.

2.0.18 Rapid recognition of a spectral change

This booklet begins with the goal of escaping the strait jacket of stationarity, intending merely to allow for slowly variable spectral change. Real life, of course has many important examples where 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. 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 one-dimensional example: Let \( a = 1 \) and \( b = -1 \). The time function

\[
(...,a,a,b,b,a,a,a,b,b,a,a,a,b,b,a,a,a,b,b,a,a,a,b,b,a,a,a,b,b,a,a,b,b,a,a,a,b,b,a,a,a,b,b,...)
\]

begins with period 6 and abruptly switches to period 4. The magnitude of the prediction error running to the right is quite different from that running to the left. Running right the prediction error is about 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 \( \sigma_{\text{right}} \) and going left be \( \sigma_{\text{left}} \). The local spectrum will be defined by a weighted average of the two PEFs. The two weights are first \( w_{\text{left}} = \sigma_{\text{right}} / (\sigma_{\text{right}} + \sigma_{\text{left}}) \) and then \( w_{\text{right}} = \sigma_{\text{left}} / (\sigma_{\text{right}} + \sigma_{\text{left}}) \). The weights sum to unity. 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 PEFs adapt slowly to spectral changes. Amazing! This completes your preparation for the image of Lenna.

Evolving document. Save the link, not the PDF. February 13, 2018
2.0.19 Boundaries between regions of constant spectrum

There is no direct application to predicting financial markets. But, with recorded data one could experiment with predictions forwards and backwards in time. 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 2.5) 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 backwards, even at other angles.

Figure 2.5: The coefficients in a 3-D PEF. (GIEE)

Figure 2.6: Lenna, a widely known photo used for testing engineering objectives in photometry. (Wikipedia)

You are trying to remove noise from the well-known “test photo of Lenna” (figure 2.6).
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. Bingo!

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

Although it’s 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, $y(t,x,y)$, I visualize it containing planes. A plane in 3-D looks like a line in both $(t,x)$ and $(t,y)$ space. It’s more efficient to fit two planes each with a 2-D PEF $[a(t,x), a(t,y)]$ 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

Chapter 3

Fitting while whitening residuals

A typical geophysical data fitting problem may be expressed as $0 \approx r = Fm - d$, where $d$ is data, $m$ is a model, $r$ is a residual, and $F$ is an operator representing physics. For statistical reasons (to achieve IID) we also need a PEF applied to residuals, say $A$. You find here the code for finding the nonstationary PEF $A$ while finding the model $m$ in

$$0 \approx q(m) = A (Fm - d)$$

Later a regularization term is included. The approach here is called “streaming,” meaning that including $A$ does not prevent streaming, that the entire data volume need not be kept in memory—it all flows through the box defined by codes here. Whether your entire process does allow streaming naturally depends on whether your $F$ operator in $0 \approx Fm - d$ allows it.

It is shown in an appendix that PEFs whiten their output signals. Starting out on a model fitting project we’d like to know the PEF that whitens the ultimate fitting residuals that we don’t have yet. So, let us change the PEF at every iteration. By whitening our current residuals we whiten our final residuals. This is more proper than guessing that the final residual spectrum is the same as the data spectrum! (Theory states that iteration-variable weighting harms conjugate gradient solvers while not bothering conjugate direction solvers.) What happens to people who use suboptimal weights and PEFs? They do not use their data “efficiently” (a statistical term). We are ready to go, wondering if we might consistently outperform our forebears in iteration count as well as solution quality. Even if not, we remain enthusiastic. The tool here handles both nonstationarity and spatial aliasing, massive practical issues in seismic imaging.

3.0.21 Applying the adjoint of a streaming filter

Those of us with a long history of filtering think of a filter adjoint as running the filter backwards. That view arises with recursive filters whose adjoint must indeed run backwards. With nonrecursive filters, such as prediction error, there is a more basic view. In a (nonrecursive) linear operator program, the inputs and outputs can be exchanged to produce the
adjoint. For example the pseudocode below applies a filter $a$ to the physical residual $y$ to get a statistical (whitened) residual $r$. We get the adjoint by swapping spaces.

```plaintext
# CODE = CONVOLUTION AND ITS ADJOINT
do it= na, nt {
do tau = 1, na {
    if( operator itself)
        r(it) += y(it-tau+1) * a(tau)  # one output r(t) pulls many
    if( adjoint )
        y(it-tau+1) += r(it) * a(tau)  # one input r(t) pushes many
    }
}

Observe the time axis runs forward for both the operator and its adjoint $A^*$. It could as well run backwards.

### 3.0.22 Applying the PEF with its adjoint

The residual after PEF whitening a simple data fitting problem is $r = A(Fm - d)$. The energy in it is $E = (m^*F^* - d^*)A^*A(Fm - d)$). The gradient of $\partial E/\partial m^*$ is $\Delta m = F^*A^*A(Fm - d)$. Here are the steps to compute it.

\[
\begin{align*}
y &= (Fm - d) \\
r &= A(Fm - d) = Ay \\
q &= A^*A(Fm - d) = A^*r \\
\Delta m &= F^*A^*A(Fm - d) = F^*q
\end{align*}
\]

Equations above are in code below for $q = A^*Ay$. Your job is to make $y$ and $F^*q$.

```plaintext
# CODE = DATA FITTING WITH PEFed RESIDUALS.
a(*) = 0;  a(1) = 1.  # You compute y=Fm-d.
r(*) = 0
q(*) = 0
do t= na, nt {
do tau = 1, na {
    r(t) += y(t-tau+1) * a(tau)  # Make statistical residual
  }
do tau = 1, na {
    q(t-tau+1) += r(t) * a(tau)  # Apply adjoint
  }
do tau = 2, na {
    a(tau) -= y(t-tau+1) * r(t) * epsilon  # Update filter
  }
  # You apply F' to q.
}

Only the middle $\tau$ loop is new. This code is untested. Notice that the program also works when the time axis is run backwards. In two dimensions, either or both the axes may be run backwards. Flipping axes flips the region in which statistics are gathered.

*Evolving document. Save the link, not the PDF. February 13, 2018*
3.0.23 PEFs for both fitting and styling

Having PEF A on the regularization and PEF B on the fitting, the gradient is

$$\Delta m = F^* B^* Br + \epsilon^2 A^* Am \quad (3.6)$$

which may be coded as we did equations (3.2)-(3.5). Naturally, the PEF B is applicable only on those data-space axes that are regularly sampled such as on a marine cable.

We have ignored preconditioning. It’s important for covering large 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 this method might render preconditioning irrelevant even on larger gaps.

3.0.24 Signal regridding

Like equations (3.2)-(3.5) and (3.6) we may deduce PEFs B and A and the gradient $\Delta m$

$$\Delta m = L^* B^* B (Lm - d) + \epsilon^2 A^* Am \quad (3.7)$$

where L is the linear interpolation operator. Equation (3.7) applies to 2-D and 3-D data regularization. I believe it should also be able to fill holes.

REFERENCES

Chapter 4

Multi channels = Vector-valued signals

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 multi-component 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, shear waves on horizontals, but dipping waves corrupt each channel with the other. The main goal here is to disentangle this channel crosstalk.

Scalar blind deconvolution is widely used in the seismic survey industry. It is shown in the top line of Figure 4.1. Oversimplifying, the idea is that earth layers have random densities hence 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 airbubbles, 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 $r_t$. As an afterthought, this is often followed by a bandpass filter, used because the PEF output is subjectively too white. The whiteness emphasizes low and high frequencies that are usually noise. People usually think of a PEF as a spectral whitening process. I often gap a PEF, meaning that I specify some zeros following the initial impulse. These zeros broaden the central central spike in the autocorrelation.

Widespread adaptation of multicomponent recorders leads to new opportunities indicated by the lower part of Figure 4.1. 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 here the theory of matrix valued PEFs, we design a matrix of filters, say $A = a_{i,j}$ attempting to achieve the original purity of $z$. Normally we do not wish to achieve the pure whiteness of $z$. Rather than apply a bandpass filter here, we use our estimates $\hat{b}_{11}$ and $\hat{b}_{22}$ to find $\hat{x}$ as our attempt to restore the original colored signals $x$.

Others may make other choices, but we are choosing to display $\hat{x}$ for a reason. We want tests of whether 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. This because the earth acts

---

1This chapter draws from [Claerbout and Wang, 2017].
Figure 4.1: Top is scalar decon. Bottom is vector decon. In nature two uncorrelated white random signals $z$ get colored creating $x$ which then get mixed creating our observations $y$. Vector decon converts $y$ to uncorrelated white signals $r$ which hopefully are a reasonable approximation to $z$. If $r \approx z$, then $AB \approx I$, so recoloring $r$ without mixing gives us $\hat{x}$, which should match the original colored signals $x$. (Kaiwen Wang)

as a wavelength filter. It’s generally believed the earth dissipates waves proportional to their spatial frequencies. Cutting both P and S at the same spatial frequency implies S will cut 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 4.1 arises in diverse physical settings. Not only does the earth contain pressure waves and shear waves while 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. Since 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 4.1. 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 this implies a symmetric function of time, not causal. Fourier space requires stationary statistics, forbids $\ell_1$-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.

4.0.25 No time delays please

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 here 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 (anti-causality!). Since \( \mathbf{A} \), a PEF, is casual by construction, it cannot function anti-causally. Whatever \( \mathbf{A} \) would come out of this process, it could not satisfy \( \mathbf{B} \mathbf{A} = \mathbf{I} \). In other words, there are many ways \( \mathbf{B} \) could contain delays without changing its covariance \( \mathbf{B}^* \mathbf{B} \).

Our inverse operator \( \mathbf{A} \) is fundamentally based on \( \mathbf{B} \mathbf{B}^* \) which contains no phase. We get phase by insisting on causality in \( \mathbf{A} \).

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

Although 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]. 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 such notions.

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

### 4.0.26 Vector signal scaling

When components of data or model are out of scale with one another, bad things happen: The adjoint operator will not be a good approximation to the inverse. Physical units may be contradictory. Steepest descent creeps 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- and down-going waves. Or such as uncalibrated vertical and horizontal seismograms.

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

With scalar signals we updated using \( \Delta \mathbf{a} = - (\epsilon/e^2) y_{t-\tau} \). With multiple channels we will be 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 \( \sigma_y \) be an estimate of the standard deviation of the prediction error in each channel. The code below resembles this update

\[
\Delta \mathbf{a} = - \left( \frac{\epsilon}{\sigma_y} \right) \mathbf{y}_{t-\tau} \quad (4.1)
\]

Our original code contained leaky integrations for \( \sigma_y \) and \( \sigma_r \), but we had no vision of data to test that aspect. It also gave odd behavior when we adapted too rapidly. Since
we had more pressing areas to direct our attention to, the code exposition below simply replaces $\sigma_y$ and $\sigma_r$ by their global averages.

4.0.27 Pseudo code for vector signals

Compared with earlier pseudocode for scalar signals where the gradient is a scaled adjoint, the gradient here has divided out the variances $\sigma_r$ and $\sigma_y$. That because we may always scale gradient components by positive numbers, say $\text{sigy}$ and $\text{sigr}$. Look at the code below for the four do loops following Happy streaming. You see a matrix full of PEFs at work. Next, the three loops below the PEF filtering are simply its adjoint (except for the complication of the $\sigma_r$ and $\sigma_y$ 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, na=10, gap=1, ic, jc, nc=2
real y(nc,nt), r(nc,nt), aa(nc,nc,na), sige(nc), sigy(nc), eps
e(1,*) = 0.
aa(*,1,1) = 0.    # Make an identity matrix.
do ic=1,nc {
    aa(ic,ic,1) = 1.
}
read input y(nc,nt)  # Read multichannel data.
#
do ic=1,nc {  # Initial variance estimates.
    sumsq=0
    do it=1,nt
        sumsq += y(ic,it)**2
        sigy(ic) = sqrt(sumsq/nt)
        sigr(ic) = sigy(ic)/2.
    }
    # Here we go! Happy streaming. Wheee!
do it= na, nt {
    do tau=1,na {  # 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+1)
            }
        }
    }
    # Optionally update sigy and sige
    do tau=gap+1, na {  # 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+1) /sigy(jc))
            }
        }
    }
}
```

Now it’s 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 a quadratic form, even more doubts that I’d be able to explain my analysis to other people, and 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 4.1 But, better not to think at all, instead, simply code the adjoint! Phew.
4.0.28 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 $\epsilon = \text{eps}$ the filter $A$ namely $aa(\ast,\ast,\ast)$ 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’d have a square matrix full of such statements. The fitting functions are all channel combinations of the shifted data. That’s the main ingredient to Levin’s whiteness proof [Levin et al., 2013] for scalar signals. I believe this means we can presume Levin’s whiteness proof applies to vector signals. As we’ll see a bit later, however, the situation at zero lag does bring up something new (Cholesky, etc).

4.0.29 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 to the white output $r_1$ the original spectrum, namely $b_{11}$. Since $B = A^{-1}$ we can deduce $b_{11}$.

\[
\begin{bmatrix}
  b_{11} & b_{12} \\
  b_{21} & b_{22}
\end{bmatrix}
= 
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}^{-1}
= 
\frac{1}{a_{11}a_{22} - a_{21}a_{21}}
\begin{bmatrix}
  a_{22} & -a_{12} \\
  -a_{21} & a_{11}
\end{bmatrix}
\]

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

\[
\begin{align*}
  b_{11} &= a_{22}/(a_{11}a_{22} - a_{21}a_{21}) \approx 1/a_{11} \quad (4.3) \\
  b_{22} &= a_{11}/(a_{11}a_{22} - a_{21}a_{21}) \approx 1/a_{22} \quad (4.4)
\end{align*}
\]

I believe we should test the stability of the simplified approximation before thinking of adding in the complicating terms. We do this by appending some code to our SEP 170 code. The result of polynomial division $\hat{x}(Z) = r(Z)/A(Z)$ is recognizable in the code by $\hat{x}_t = xhat(ichan,t)$. Here is the polynomial division code fragment.

```python
# CODE = Polynomial division
xhat(1,t) = r(1,t)
do tau=2,na # xhat1(Z) = r1(Z)/a11(Z)
xhat(1,t) -= aa(1,1,tau) * xhat(1,t-tau+1)

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

4.1 CHOLESKY DECORRELATING AND SCALING

The two independent channels of unit-variance random numbers in $r$ entering filter $B$ in Figure 4.1 have the identity matrix $I$ as a covariance. Here we arrange to have the same identity covariance for the values $r$ exiting from $A$ on the right.

Evolving document. Save the link, not the PDF. February 13, 2018
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 $2 \times 2$ matrix of prediction-error variances $W$.  

$$W(\tau = 0) = \begin{bmatrix} \sigma_{r_1}^2 & \sigma_{r_2}^2 \\ \sigma_{r_1}^2 & \sigma_{r_2}^2 \end{bmatrix} \approx \begin{bmatrix} (r_1 \cdot r_1) & (r_1 \cdot r_2) \\ (r_2 \cdot r_1) & (r_2 \cdot r_2) \end{bmatrix} \quad (4.5)$$

Consider the expectation (leaky sum over time) $E[rr^*]$. Theoretically it’s a 3-D function of lag and the two channels. We’re going to assume our PEFs are perfect so that it is no longer a function of lag. Thus we presume that $E[rr^*]$ is like the $W$ we computed with equation (4.5) at zero lag $\tau$.

Use the Cholesky method to factor $W$ into a triangular matrix $V$ times its transpose, so $W = VV^*$. (The Cholesky method is nearly trivial: Write a triangular matrix of unknown elements. Multiply it by its transpose. Notice a sequential method that unravels the unknown elements.)

$$W = VV^* \quad (4.6)$$

$$V^{-1}W(V^*)^{-1} = I \quad (4.7)$$

$$CWC^* = I \quad (4.8)$$

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

$$q = Cr \quad (4.9)$$

The expectation of this new variable $q$ is

$$E[qq^*] = E[Crr^* C^*] \quad (4.10)$$

$$= CE[rr^*] C^* \quad (4.11)$$

$$E[qq^*] = CW C^* = I \quad (4.12)$$

This shows Cholesky does for us two things: (1) it descales, and (2) it decorrelates $r$ at zero lag.

### 4.2 ROTATING FOR SPARSITY

This merging of two write-ups contains duplication that I need to remove.

Intriguing is what comes last, something wholly unfamiliar. Even after solving the problem posed in Figure 4.1 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 $U$ having minimum entropy $r$ output. Unexpected. Intriguing! 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!

The most intriguing part of the entire process arrives at this the last stage. As the universe marches on, things get mixed and entropy increases. We seek the opposite.
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 $\theta$ through all angles allows us to choose the one with the most sparsity (least clutter). A general form for a $2 \times 2$ rotation operator is

$$U = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

(4.13)

We will meet our goal of finding $A$ and $r$ of Figure 4.1 with:

$$r = Uq = UCr = UCEy = Ay$$

(4.14)

A unitary operator $U$ does not change the length of any vector. It satisfies $U^*U = I$, so for any $v$ we see $(Uv)^*Uv = v^*U^*Uv = v^*v$. Let us check that the covariance of $r = Uq$ is constant independent of $\theta$. Equation (4.12) leads to $rr^* = UE[qq^*]U^* = UIU = I$. This says the energy stays constant as we sweep through $\theta$.

### 4.2.1 Finding the angle of maximum sparsity (minimum entropy)

Given any angle $\theta$ for equation (4.13) we have $r = Uq$. We can scan $\theta$ 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

$$\text{Entropy}(\theta) = \frac{\sum_{t=1}^{\infty} |r_1(t)| + |r_2(t)|}{\sqrt{\sum_{t=1}^{\infty} r_1^2(t) + r_2^2(t)}}$$

(4.15)

Since the denominator should be a constant function of $\theta$ we may as well define entropy by

$$\text{Entropy}(\theta) = \sum_{t} |r_1(t)| + |r_2(t)|$$

(4.16)

### Why the scan works

Why does this $U$ process of scanning $\theta$ lead to sparsity? Suppose the vector signal element $q_N$ at time $t = N$ 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

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

(4.17)

Let the rotation angle be $45^\circ$ 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 $U$ to be time variable. There are some pitfalls we are not yet prepared to explain.

Evolving document. Save the link, not the PDF. February 13, 2018
4.2.2 3-component vector data

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

4.2.3 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

\[
\begin{bmatrix}
0 & 1 \\
1 & 0 \\
\end{bmatrix}
\]  \hspace{1cm} (4.18)

Now we flip channels when we find the expression 
\[ |r_1 \cdot y_1| + |r_2 \cdot y_2| < |r_1 \cdot y_2| + |r_2 \cdot y_1|. \]

Our initial P-wave result had a flipped polarity. The operation for flipping the polarity for channel 1 is

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]  \hspace{1cm} (4.19)

We change the polarity of channel 1 when \( (y_1 \cdot r_1) < 0 \) and likewise for channel 2.

It is easy to show for signals with an identity \( I \) correlation matrix, the channel flip and polarity change operations do not change the \( I \) correlation matrix.

It is easy to imagine situations where 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.

4.3 RESULTS OF KAIWEN WANG

Figure 4.2: 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) 

Figure 4.2 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.
4.3. RESULTS OF KAIWEN WANG

Figure 4.3: Output results: Deconvolved P wave on vertical component (top), S on horizontal (bottom). Spiking improves with time. (Kaiwen Wang)

On Figure 4.3 we notice that the spike estimates become sharper and sharper with time as the filter $A$ adapts with time. Oddly, there is some crosstalk on the P channel that does not seem to be diminishing with time. I don’t know why that is. Perhaps I should run the program over the panel a zillion times, at each end, capturing the filter and reinstalling it at the beginning.

Figure 4.4: $V=$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)

On Figure 4.4 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 4.5 is like figure 4.4 but denser spikes, a spike every 4 pixels, so three quarters of the dots should be zero valued with the rest at spike tops. Notice the vertical trace (top) being dominated by P waves is higher frequency than the horizontal trace "H" which is dominated by S waves. Results are about the same quality showing that having lots of wavelet overlap causes no real problems. Fitting on the S channel gets much better with time. Fitting on the P channel is so good near the beginning that we hardly notice improvement with time.
CHAPTER 4. MULTI CHANNELS = VECTOR-VALUED SIGNALS

Figure 4.5: Illustration with denser spikes. (Kaiwen Wang) [chap4/. denseSpikes]

REFERENCES

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 is extensively demonstrated in GIEE \cite{Claerbout2014}, while it is simply introduced and proven here now.

5.1.1 Why 1-D PEFs have white output

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 here is different; namely, the output is uncorrelated with itself (as opposed to the input) for lags in both directions; hence 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 $d$ be a vector with components containing a time function. Let $Z^n d$ represent shifting the components to delay the signal in $d$ by $n$ samples. The definition of a PEF is that it minimizes $||r||$ by adjusting filter coefficients $a_\tau$. The PEF output is:

$$r = d + a_1 Z^1 d + a_2 Z^2 d + a_3 Z^3 d + \ldots$$  \hspace{1cm} (5.1)

We set out to choose the best $a_\tau$ by setting to zero the derivative of $(r \cdot r)$ by $a_\tau$. After the best $a_\tau$ are chosen, the residual is perpendicular to each of the fitting functions:

$$0 = \frac{d}{da_\tau} (r \cdot r)$$  \hspace{1cm} (5.2)

$$0 = r \cdot \frac{dr}{da_\tau} = r \cdot Z^\tau d \quad \text{for } \tau > 0.$$  \hspace{1cm} (5.3)

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

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

\footnote{This subsection draws from \cite{Levin2013}, also included in \cite{Claerbout2014}.}
= \textbf{r} \cdot Z^k \textbf{d} + a_1 \textbf{r} \cdot Z^{k+1} \textbf{d} + a_2 \textbf{r} \cdot Z^{k+2} \textbf{d} + ... \\
= 0 + a_1 0 + a_2 0 + ... \\
= 0 .

Because the autocorrelation is symmetric, \( \textbf{r} \cdot Z^{-k} \textbf{r} \) is also zero for \( k < 0 \); therefore, the autocorrelation of \( \textbf{r} \) is an impulse. In other words, the spectrum of the time function \( r_t \) is white. Thus, \( \textbf{d} \) and \( \textbf{a} \) 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. In summary, to visualize the 2-D notion of a 1-D PEF, wrap a long rope tightly spiraling around a huge culvert (pipe) covering many revolutions. The surface of the culvert and rope coils are 2-D spaces for our 2-D imaging games. Let the culvert hold the 2-D data and the rope hold the filter. Let the rope be slippery so it can slide over the pipe in a 2-D space. Such sliding may be along the axis of the culvert, or along the rope, or any direction in the 2-D surface. 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 “1.0.” Filter coefficients near the “1.0” in the 2-D space are typically the most significant ones because they most likely give the best predictions of what lies under the “1.0.” These most significant coefficients lie inside the semicircles in Figure 5.1. So, in 2-D a PEF is a little patch of numbers along one side of the “1.0”.

Figure 5.1: A 2-D whitening filter template with output at “A”, and a copy of itself lagged with output at “B”. At “A” and “B” the filter coefficient is the constrained “1.0.” When the semicircles are viewed as having infinite radius, the B filter is contained in the A filter. Because the output at A is orthogonal to all its inputs, which include all inputs of B, the output at A is orthogonal to the output of B. (Claerbout) 

Figure 5.1 shows the input plane with a 2-D filter on top at two possible locations. The filter shape is a semidisk, which you should imagine being of infinitely large radius. Notice that semidisk A includes all the points in B. The output of disk A is next shown to be orthogonal to the output of disk B. Conventional least squares theory says the coefficients of the filter are designed so that the output of the filter is orthogonal to each of the inputs.
to that filter (except for the input under the “1.0,” because any nonzero signal cannot be orthogonal to itself). Recall that if a given signal is orthogonal to each in a given group of signals, then the given signal is orthogonal to all linear combinations within that group. The output at B is a linear combination of members of its input group, which is included in the input group of A, already orthogonal to A. Therefore the output at B is orthogonal to the output at A. In summary,

\[
\begin{align*}
\text{residual} & \perp \text{fitting function} \\
\text{output at A} & \perp \text{each input to A} \\
\text{output at A} & \perp \text{each input to B} \\
\text{output at A} & \perp \text{linear combination of all parts of B} \\
\text{output at A} & \perp \text{output at B}
\end{align*}
\]

The essential meaning is that a particular lag of the output autocorrelation function vanishes.

Study Figure [5.1] to see for what lags all the elements of the B filter are wholly contained in the A filter. These are the lags in which we have shown the output autocorrelation to be vanishing. Notice another set of lags in which we have proven nothing (where B is moved to the right of A). Autocorrelations are centrosymmetric, which means that the value at any lag is the same as the value at the negative of that lag, even in 2-D and 3-D in which the lag is a vector quantity. Previously, we have shown that a halfplane of autocorrelation values vanishes. By the centrosymmetry, the other half must also vanish. Thus, the autocorrelation of the PEF output tends to an impulse function, so its 2-D spectrum tends to white.

I don’t 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 1.0, you spoil the whiteness of the output.
5.2 DERIVATIVE OF A NEW DATA RESIDUAL BY A PEF

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

\[
\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix} \approx \begin{bmatrix}
d_{99} & d_{98} & d_{97} & d_{96} \\
\gamma & \cdot & \cdot & \cdot \\
\cdot & \gamma & \cdot & \cdot \\
\cdot & \cdot & \gamma & \cdot \\
\cdot & \cdot & \cdot & \gamma
\end{bmatrix} \begin{bmatrix}
1 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix} - \gamma \begin{bmatrix}
0 \\
\bar{a}_1 \\
\bar{a}_2 \\
\bar{a}_3
\end{bmatrix}
\]  

(5.4)

The top row says we are trying to fit a new data point \( d_{99} \). The bottom block says the new PEF \( a \) should be pretty similar to the PEF that fit earlier data, \( \bar{a} \). The parameter \( \gamma \) should be big enough that the new data point \( d_{99} \) does not change \( a \) very much. Rewrite equation (5.4) as

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

(5.5)

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

\[
0 \approx r = \begin{bmatrix}
d^* \\
\gamma I
\end{bmatrix} a - \begin{bmatrix}
-d_{n+1} \\
\gamma \bar{a}
\end{bmatrix},
\]  

(5.6)

where \( I \) is the identity matrix and

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

For decades Bernard “Bernie” Widrow (Wikipedia) attacked problems of this nature by defining a quadratic form and finding its gradient. Then he repeatedly made small steps down the gradient (not up). How big are the small steps? Experience teaches. The quadratic form is \( r^*r \). We take its derivative to find the search direction.

\[
\Delta a = - (\text{some constant}) \frac{\partial}{\partial a^*} \bigg|_{a = \bar{a}} r^*r
\]  

(5.7)

Form the transpose of the residual (5.6) and then differentiate by \( a^* \). (By \( a^* \) we mean the complex conjugate transpose of \( a \).)

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

(5.8)

\(^2\text{This section drawn on Fomel et al. (2016) and Claerbout (2017).}\)
and multiply that onto \( r \) from (5.6) keeping in mind that \( d^*a \) is a scalar.

\[
\Delta a \propto \frac{\partial r^*}{\partial a^*} \quad r = [d \quad \gamma I] \left\{ \left[ d^* \begin{bmatrix} I \\ \gamma I \end{bmatrix} a - \begin{bmatrix} -d_{n+1} \\ \gamma \bar{a} \end{bmatrix} \right] \right\} \\
= d(d^*a) + \gamma^2 a + dd_{n+1} - \gamma^2 \bar{a} \quad (5.9)
\]

\[
\Delta a \propto \left. \frac{\partial r^*}{\partial a^*} \right|_{a=\bar{a}} \quad r = (d^*\bar{a} + d_{n+1}) \quad d \quad (5.10)
\]

\[
\Delta a \quad (5.11) \quad \Delta a = -\epsilon r_t \quad d \quad (5.12)
\]

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 that is so (Fomel et al., 2016). Experimentalists will first notice that (5.4) demands we supply a not-given constant \( \gamma \) while (1.4) or (5.12) demands a not-given constant \( \epsilon \) (or \( \lambda \)).

REFERENCES
