Stanford Exploration Project, Report 100, April 20, 1999, pages 1–227

Short Note

Polarity and PEF regularization

Jon Claerbout[1]

keywords: *polarity PEF prediction error*

INTRODUCTION

We address the puzzle of seismic polarity. Why do we rarely observe it clearly and how could we be more systematic about trying to observe polarity? This puzzle will lead us to long prediction-error filters. Being long, they require many data samples. If such a filter is nonstationary, we might have an inadequate number of fitting equations. Then we need regularization. Here we consider some examples and consider an efficient way to regularize the filter estimation.

EXAMPLES

We sometimes observe seismic signal polarity. We rarely observe it in soft clastic areas like much of the Gulf of Mexico, but often see it in more hardened areas like the North Sea. Generally, polarity is much more likely to be recognized in the ideal conditions we encounter with marine data than with land data. Even where it is possible to recognize, it is not especially noticeable because even an ideal seismic impulse is spread out by the shot and receiver ghosts to the (low cut) three-phase event $(1, -2, 1)$. An attractive example that I recently encountered is that of Christine Ecker in Figure 1.

In principle, a causal double integration converts this triplet into an impulse which theoretically should make for ready recognition of polarity. On the other hand, we need to think about whether the recording equipment actually records the low frequencies that double integration would amplify and we need to think about low-frequency noise.

I have noticed that marine recording systems often record seismic energy right down to zero frequency. This is evidenced by the appearance of water surface gravity waves with speeds of about 30 km/hour, speeds so slow that we can hardly distinguish their speed from zero. I have not made quantitative measurements on Figure 2, but I believe it to be consistent with long ocean swells moving about 20 miles/hour or 30 km/hour. Distance traveled in

[1]**email:** jon@sep.stanford.edu, http://sepwww.stanford.edu/sep/jon/

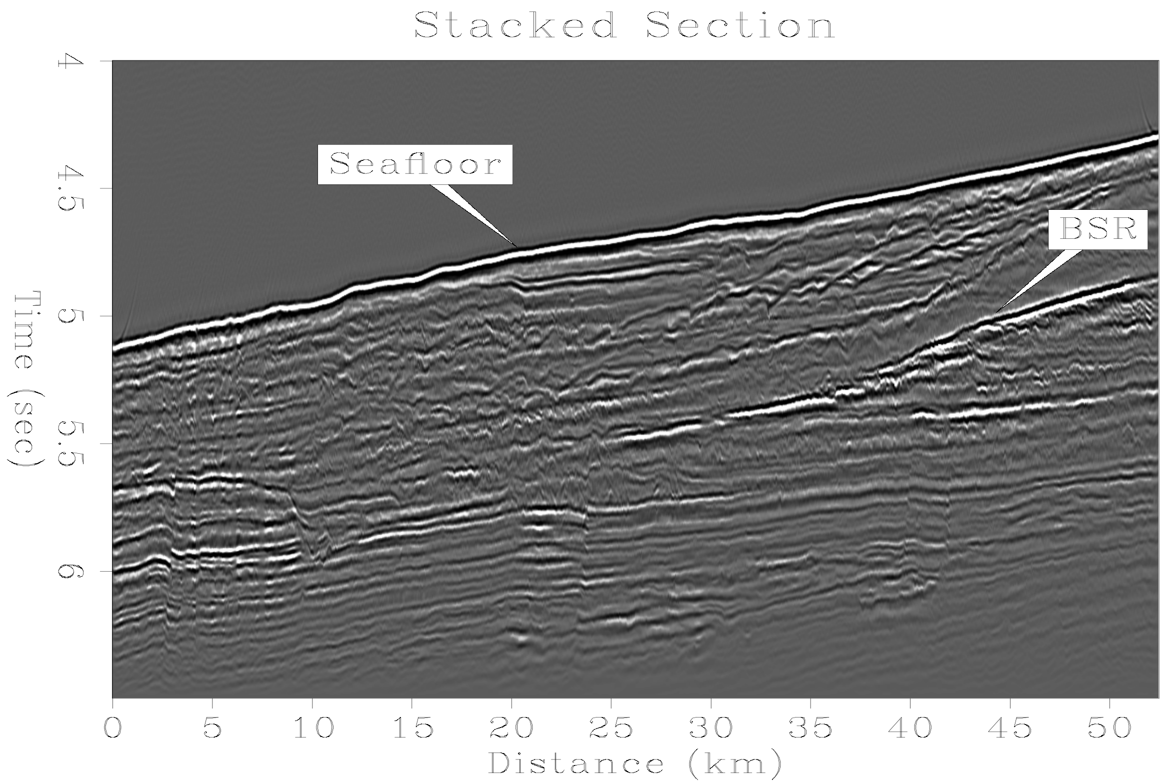

Figure 1: Christine Ecker's BSR data from her PhD disertation. Most reflectors are recognizeable as "black-white-black". The exception is the BSR reflector which is "white-black-white". jon3-christine [NR]

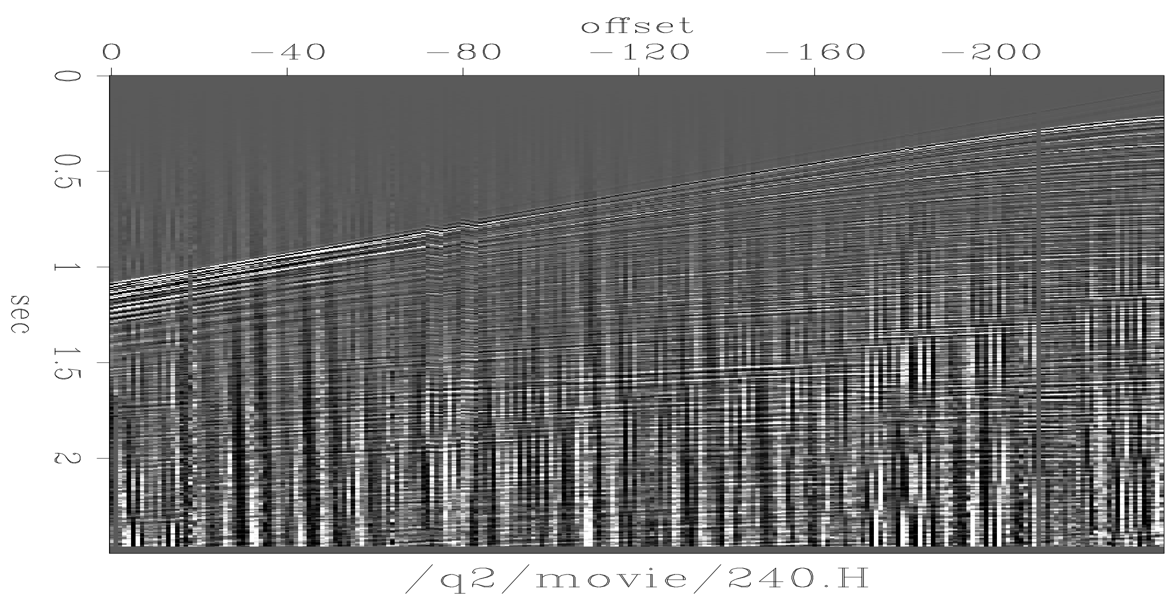

Figure 2: Marine data with t^2 gain. jon3-gravity [ER]

meters over four seconds will be $4 \times 30 \times 1000/3600$ = 33m. These become visible late on the data where the t^2 gain brings them up. Their time signature is roughly a growing ramp function.

From these two data sets we recognize two different goals: One data set says we should use double integration to convert the $(1, -2, 1)$ ghost to a pulse. The other data set says we should use double differentiation to suppress the growing ramp function of the surface gravity waves.

On more careful study we can see that the goals are not *exactly* opposite: The ramp function rises over the full two seconds of the data. Let us say its frequency is about .2 Hz. On the other hand, the triplet $(1, -2, 1)$ is not merely spread over three samples but perhaps 10 samples or 40ms so we associate it with a frequency of about 25 Hz. Now imagine a data set with both the phenomena we see on Figure 1 and on Figure 2. It would show both phenomena. We can analyze this in the frequency domain or the time domain. The composite spectrum would be the sum of their spectra. The spectrum would grow near zero frequency as $1/\omega^2$ and grow near .2 Hz as ω^2. We need to raise the spectrum between those two limits (or suppress it beyond those limits). In the time domain we notice that while the ramp function is approximately as long as the seismogram itself, the duration of the triplet $(1, -2, 1)$ is about 10 samples or 40ms. Perhaps the filter with the correct spectral characteristic in the time domain would begin as a ramp for 40ms and then gently bend toward negative values a second or so later. Notice that this is a fairly long filter

We could examine the specifics more carefully but fortunately, the autoregression method of deconvolution addresses the general problem without requiring specific information. Notice a key ingredient here: The required autoregression filter, the PEF, is *long* because any approximation to double integration must be long. The filter length raises interesting issues that I have previously not given adequate attention to.

Basically we are talking about a decon filter whose length is a significant fraction of the trace length. There is a real danger that we might "over fit", i.e., use insufficient data to estimate the filter and find our filter adapting to the geology instead of adapting to the data-recording environment. The way to overcome this problem is to use a lot of data.

MINIMUM-PHASE EQUIVALENT TRAINING DATA SET

This leads to a technique that is new to me. I'll describe it first in the one-dimensional world. (In real life, multidimensional cases might be more interesting, for example where dip spectra change rapidly.) The basic problem is to define the appropriate regularization for a prediction-error filter (PEF). Regularization is ordinarily regarded as supplying a prior statement about the model, in this case, about the autoregression filter. We don't think about PEFs as being "physical" and the correct prior model and its covariance are not immediately obvious. The answer is that the prior PEF is nothing more and nothing less than the solution to the autoregression equations for a prior "universal" data set. In practice, it amounts to having a "training" data set. I have noticed an efficient way to merge the information of the training data set with the "too-small" local data set. Given a data set packed in an operator

$\mathbf{D}$ and likewise a training data set $\mathbf{T}$, we formulate the fitting goals for finding the PEF $\mathbf{a}$ by using a constraint matrix $\mathbf{K}$ (an identity matrix except for the (1,1) element which is zero).

$$\begin{array}{rcl} \mathbf{0} & \approx & \mathbf{DKa} \\ \mathbf{0} & \approx & \mathbf{TKa} \end{array} \tag{1}$$

In principle, the training data (and hence the matrix $\mathbf{T}$) is very large. Consider however a spectral factorization of the training data set. Say $\mathbf{T}'\mathbf{T} = \mathbf{B}'\mathbf{B}$ where $\mathbf{b}$ is a minimum-phase spectral factorization of the training data set (and $\mathbf{B}$ is the packing of $\mathbf{b}$ into a convolution operator). For me, this is a new idea, that we express the prior information as a "training wavelet" $\mathbf{b}$ that we find by spectral factorization of a "universal" data set. The idea is that we then find our "local" PEF by fitting the goals

$$\begin{array}{rcl} \mathbf{0} & \approx & \mathbf{DKa} \\ \mathbf{0} & \approx & \mathbf{BKa} \end{array} \tag{2}$$

The result of fitting (2) is theoretically equal to that of (1) but computationally (2) is potentially much easier training wavelet is much more compact (because it is minimum-phase) than the full training data set.

DISCUSSION

Sometimes data is stationary. Then Fourier analysis gives a more efficient approach. The ideas above are more relevant to cases where stationarity is less valid. Likewise, when the required filter is very long, comparable to a trace length, Fourier analysis would be more appropriate.

On the other hand, with spatial filtering applications a local PEF is more appropriate. In my book GEE, I explain how to build a time-variable PEF. It seems an alternative could be based on a training data set that varies locally. I see this as perhaps theoretically superior. In the GEE example, the filter itself is stated to vary smoothly. Now I would be proposing that the training data set be varying smoothly. In general, PEFs tend to "look bad" because their frequency content is inverse to that of the signal. This could mean that smoothing a PEF is not nearly such a good idea as using a training data set. Imagine we seek a new PEF upon the arrival of each new trace, or perhaps even upon the arrival of each new data point. Naturally, the Wilson-Burg spectral factorization method might be helpful. Generally however, I am not sure how to proceed.