Stanford Exploration Project, Report 103, April 27, 2000, pages 205–309

Patching and micropatching in seismic data interpolation

Sean Crawley[1]

ABSTRACT

I interpolate CMP gathers with PEFs arranged on a dense, radial grid. The radial grid facilitates preconditioning by radial smoothing, and enables the use of relatively large grid cells, which we refer to as micropatches. Even when the micropatches contain enough data samples that the PEF calculation problem appears overdetermined, radial smoothing still noticeably improves the interpolation, particularly on noisy data.

INTRODUCTION

In filtering applications, input seismic data are commonly divided into smaller subsets which we refer to as patches (which are also referred to as windows, gates, and other things). The data are assumed to be approximately stationary within a patch, but due to practical limits on patch size, it may not be possible to avoid nonstationarity and poor results in some patches. This is often true where the data are strongly curved (for spatial filtering) or noisy. An alternative to independent patches is nonstationary filtering. SEP has applied nonstationary filtering to numerous problems in recent reports, including groundroll suppression (Brown et al., 1999), multiple suppression (Clapp and Brown, 1999), tomography regularization (Clapp and Biondi, 1998), deconvolution (Claerbout, 1997), and interpolation (Fomel, 1999; Crawley, 1999). Nonstationary PEFs do not have patch-size limitations, so we may shrink patches down to arbitrary size and shape. To control the potentially huge null space, we regularize the set of filters to ensure that PEFs located at similar data coordinates have similar coefficients. This implements the assumption that dips in the data may change everywhere, but do so smoothly. Besides being small, these new patches are fundamentally different in that they are not independent problems, but related to each other via the regularization. To distinguish them from the old independent patches, we call them "micropatches". We have a great deal of freedom in deciding the size and shape of our micropatches, and in implementing the regularization. This paper motivates and describes my implementation.

[1]**email:** sean@sep.stanford.edu

FORMULATION

A standard formulation for calculating PEFs from known data is to solve a linear least-squares problem like

$$\mathbf{0} \approx \mathbf{YCa} + \mathbf{r}_0, \tag{1}$$

where **a** is a vector containing the PEF coefficients, **C** is a filter coefficient selector matrix, and **Y** denotes convolution with the input data. The coefficient selector **C** is like an identity matrix, with a zero on the diagonal placed to prevent the fixed 1 in the zero lag of the PEF from changing. The $\mathbf{r}_0$ is a vector that holds the initial value of the residual, $\mathbf{Ya}_0$. If the unknown filter coefficients are given initial values of zero, then $\mathbf{r}_0$ contains a copy of the input data. $\mathbf{r}_0$ makes up for the fact that the 1 in the zero lag of the filter is not included in the convolution (it is knocked out by **C**). When there are many coefficients, as when PEFs are spread densely on the data grid, it makes sense to add damping equations and/or precondition the problem. Inserting the preconditioned variable **Sp** (where **S** is a somewhat arbitrary smoother) for **a** and adding the also somewhat arbitrary roughener $\mathbf{R} = \mathbf{S}^{-1}$ to regularize the model, gives a formulation like

$$\mathbf{0} \approx \mathbf{YKSp} + \mathbf{r}_0 \tag{2}$$

$$\mathbf{0} \approx \epsilon\, \mathbf{Ip} \tag{3}$$

In many cases we can set $\epsilon = 0$ and just use equation (2), being careful not to let it go for too many iterations. We still have to define **S** (or **R**).

RADIAL SMOOTHING

We have to choose **S**. Inserting a smoother signifies the assertion that the dips in seismic data should change in a gradual way. Choosing an isotropic smoother means we expect the dips to vary similarly in all directions. However, we know that the dip spectrum of the data probably changes more quickly in some directions than in others. We want to smooth most heavily along directions where the dip is nearly constant. In a constant-velocity, flat-layered earth, events fall along hyperbolas like

$$t^2 = \tau^2 + \frac{x^2}{v^2},$$

where x is offset, v is stacking velocity, τ is zero-offset time. The time dip of an event is dt/dx. If velocity is constant, differentiating gives

$$\frac{dt}{dx} = \frac{x}{v^2 t},$$

which means that the dip does not change along radial lines, where x/t is constant. In a real earth, we suppose that dips will change, but slowly. Real earth velocity may change quickly in depth, but hyperbola trajectories are functions of RMS velocity, which is smooth. We want a smoother with an impulse response which is highly elongated in the radial direction. To get

a big impulse response cheaply, I apply the inverse of a directional derivative, pointed in the radial direction. To directly apply the inverse, the roughener has to be causal, which means that the inverse will only smooth in one direction (Claerbout, 1998). We want **S** to have an impulse response which is smoothed both in towards zero radius and out towards large radius, so we make it the cascade of the causal smoother and its anticausal adjoint.

Patches, micropatches, pixels

Having chosen the radial direction, we can think of some different ways of implementing our radially-smoothed filters. An obvious one is putting a PEF at every point on the data grid, and devising a derivative filter which adjusts its direction to point at the origin. An alternative is to overlay a radial grid on the data grid, and arrange PEFs on the radial grid. Here we compare the two smoothing schemes. Our goal is to assume stationarity in a small enough region that we can interpolate well where the data do not fit a plane-wave model. In the method of independent patches, individual patches are treated as separate problems. A patch can not be arbitrarily small, because it must provide enough fitting equations that the filter coefficients are well overdetermined. In 1-D, filtering with a PEF looks like this:

$$\hat{u}(i) = \sum_{k=0}^{p} a(k)u(i-k), \qquad p + i_{\text{patchmin}} \le i \le i_{\text{patchmax}}. \tag{4}$$

The patch boundaries are i_{patchmin} and i_{patchmax}, $\mathbf{p}$ is the number of adjustable coefficients, $a(0) = 1$. The lower limit on i is to prevent the filter from running off the end of the data and encountering implicit zero values. The patches are designed to overlap, and the outputs are normalized to hide the patch boundaries. In moving to the method of gradually-varying PEFs, we replace the notion of extracting a subset of the data with that of dereferencing the data coordinates to find the appropriate filter, as in

$$\hat{u}(i) = \sum_{k=0}^{p} a_i(k)u(i-k), \qquad p + 1 \le i \le i_{\text{datamax}}. \tag{5}$$

The index of the data sample i dereferences the set of filters $a_i(k)$. The data boundaries $i = 1$ and i_{datamax} replace the patch boundaries. We have lots of freedom in dereferencing a_i. In the limiting cases, all the data may share one PEF, or we can choose a_i to be a different set of coefficients for each data point. In the case where we have a PEF at every data point, we call **S** pixel-wise smoothing. Choosing a separate PEF for every input sample is a possibility, but not necessary. Our motivation for moving away from independent patches was to use one PEF in a region small enough that we do not have trouble with nonstationarity. Some amount of patching may still make sense, provided the patches may be small. It is easy to implement small patches as a generalization of the case above where each data sample has its own PEF. A particular a_i can be the same for any number of values of i without complication. Because they may be small, we refer to the new patches as "micropatches" to distinguish between them and independent patches. To subdivide a CMP gather into micropatches, we choose a web-like grid made up of radial lines and circular lines. Radial lines are a natural choice because we

want to smooth in the radial direction. Circles are somewhat arbitrary; we could choose flat lines or reflection-like hyperbolas to cross the radial lines. Circles have the attractive property that they make equal-area micropatches at a given radius.

Smoothing pixels versus smoothing micropatches

We can choose pixel-wise smoothing or micropatch smoothing. An easy argument favoring micropatches over pixel-wise smoothing says that putting a filter at every data sample is a tremendous waste of memory. If the data are predictable at all, they are probably not so nonstationary that they need a separate PEF at each sample. A single 3-D PEF has easily 20 or more adjustable coefficients, so allocating the set of PEFs requires 20 times the storage of the input data. Even very small micropatches require much less memory. Micropatches also have some simplifying side effects that make them preferable to pixel-wise smoothing. One is apparent from examining Figures 1 and 2. Figure 1 shows smoothing in micropatches and Figure 2 shows pixel-wise smoothing. In each figure, the values represent filter coefficients displayed in data coordinates. The axes are time and offset. The top halves show a set of impulses, labeled **d**. **Md** is the impulses binned into micropatches, while **Pd** is the impulses binned into pixels (naturally, $\mathbf{Pd} = \mathbf{d}$). **F** and $\mathbf{F}'$ are pixel-wise smoothers pointed towards and away from zero radius, respectively. **C** and $\mathbf{C}'$ are the micropatched smoothers. In this case, the two are similar, though the pixel-wise smoother obviously produces a higher-resolution picture (though the micropatches could be made much smaller). The bottom halves show the same treatment applied to a constant function, labeled **1**. **M1** has an angular limit applied. Pixel-wise smoothing creates some very large ridge artifacts, visible in **FP1** and $\mathbf{F}'\mathbf{FP1}$, where the angle between a data sample and the origin corresponds to an integer slope. Also, where the constant function is smoothed in towards zero radius, **FP1**, energy concentrates in a huge spike at the origin. $\mathbf{F}'\mathbf{F}$ and $\mathbf{C}'\mathbf{C}$ can be thought of as weighting functions in equation (2) (either $\mathbf{F}'\mathbf{F}$ or $\mathbf{C}'\mathbf{C}$ is used for **S** in that equation). It is desirable to have the flatter weighting function. It is also simpler to implement. Producing the many different angles in Figure 2 requires that the smoothers **F** and $\mathbf{F}'$ be made up of many different filters, oriented in a continuous sweep between a spatial derivative and a time derivative. **C** and $\mathbf{C}'$ produce the same range of angles in Figure 1 using a single, radial derivative filter. The PEFs in micropatches are regularly gridded in angle and radius, so they are easily smoothed in those directions with old-fashioned stationary 1-D derivative filters. PEFs at every pixel are instead regularly sampled in time and offset, so working in polar coordinates requires some work. **F** uses many coefficients, **C** uses two.

Is smoothing necessary?

Using the weblike pattern seen in Figure 1, it is often possible to use fairly large micropatches. An alternative to radial smoothing may be to simply lengthen the micropatches in the radial direction, and not bother smoothing at all. I test this in Figures 3 and 4. It seems that smoothing may have some important effects beyond just statistically compensating for the small size of a micropatch. Even with very elongated patches, such that the area of a patch is more than large

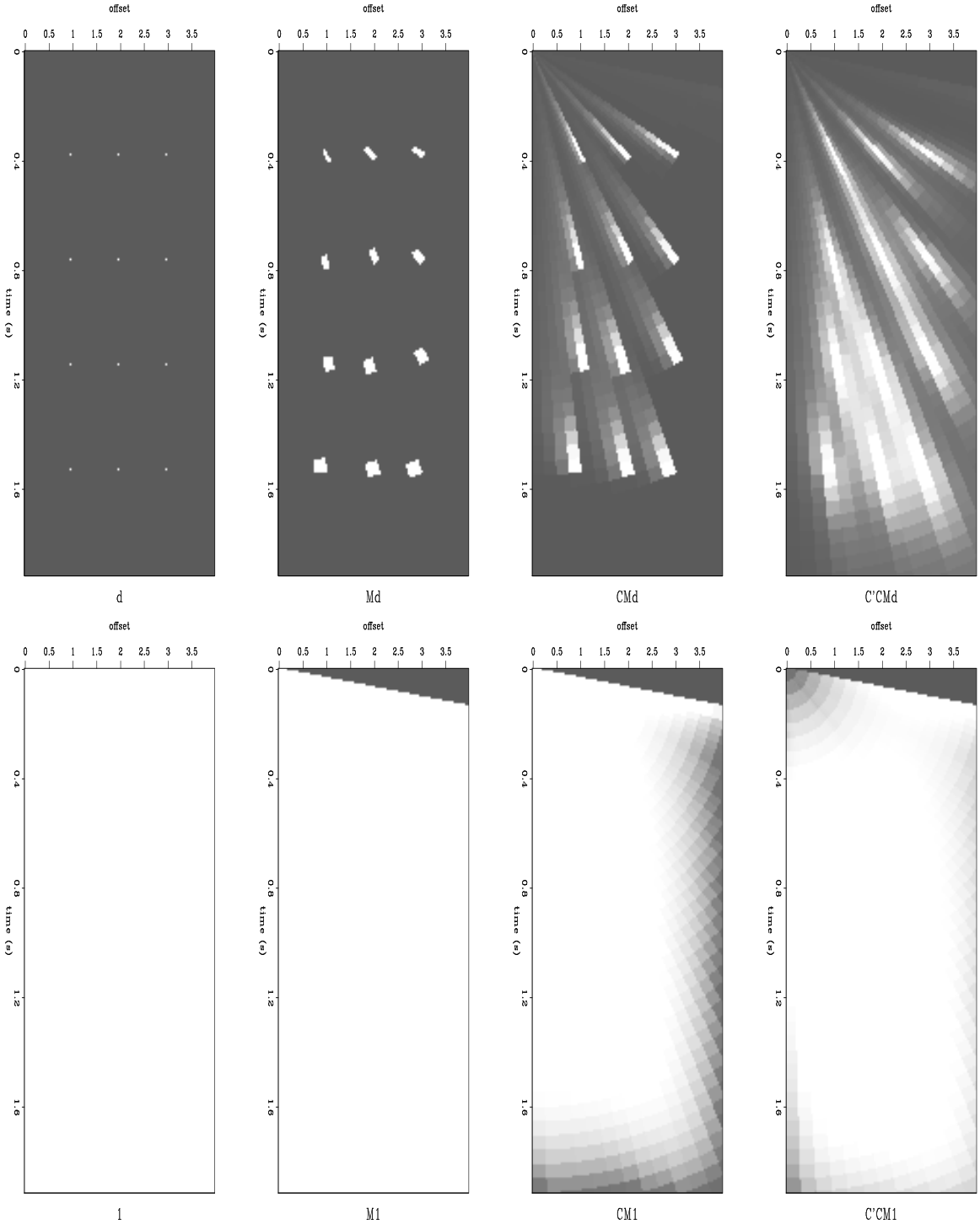

Figure 1: Illustration of micropatched radial filter coefficient smoothing. sean1-curtSmear8 [ER]

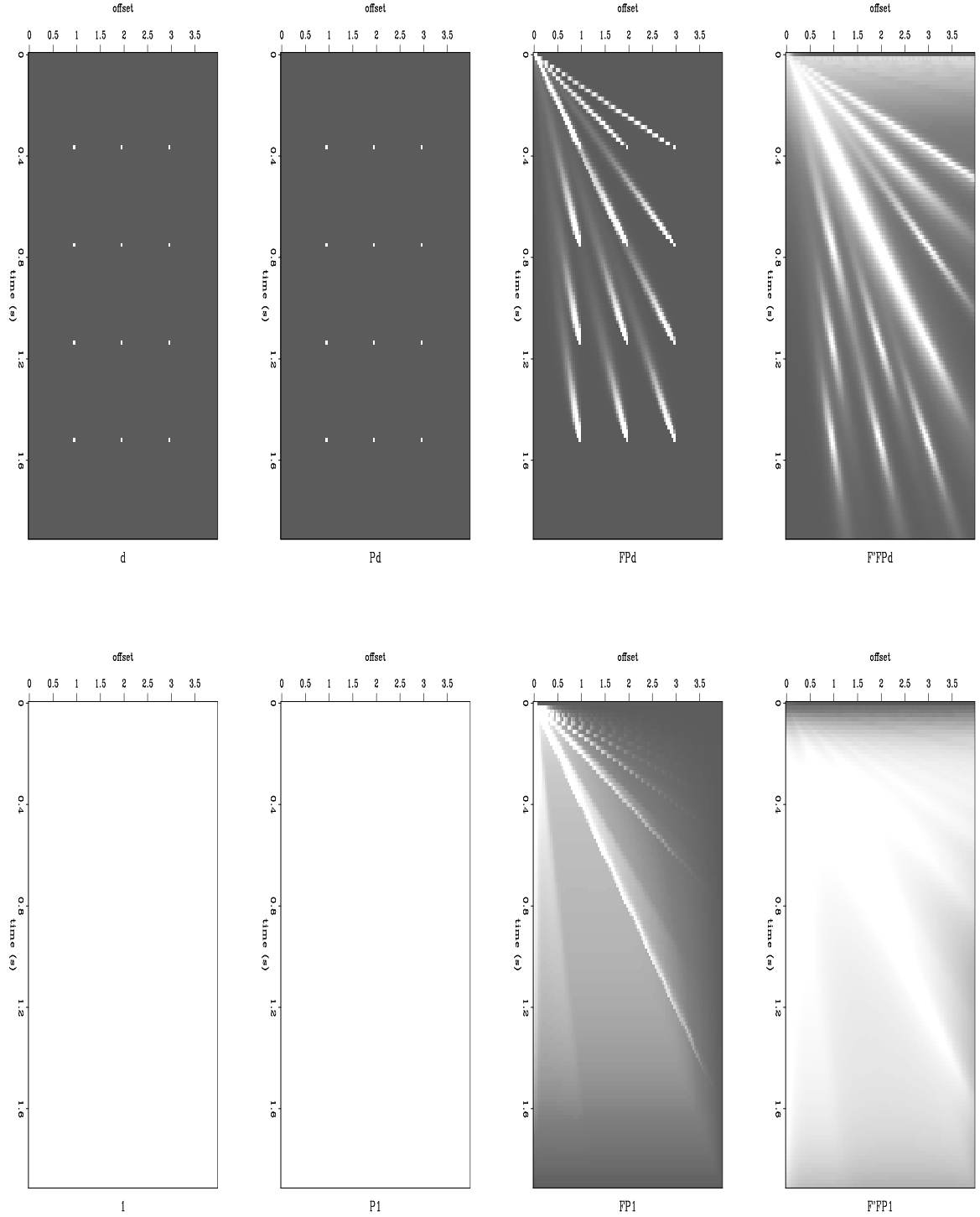

Figure 2: Illustration of pixel-wise radial filter coefficient smoothing. sean1-random8 [ER]

enough for the number of adjustable filter coefficients, smoothing noticeably improves the final result, particularly where the data have many dips or are noisy. One possible explanation is that where the data are incoherent, the change in a particular filter coefficient at each iteration is just an average of data samples, which is approximately zero. With the addition of the smoother, the change in nearby filter coefficients fills in. Figure 3 shows a section of noise-free data with many dips, interpolated with PEF smoothing on the left and without on the right. The top two panels show the interpolated traces (the known input traces are windowed out). The bottom two panels show the differences between the interpolated traces and the original traces which were thrown out to make the input. The two panels are similar, though the left side is noticeably better on some events. Figure 4 shows two more interpolation results. In this case the data is land data, and much noisier. Both known and interpolated traces are shown. Because the data is somewhat noisy, it is easier to distinguish between the coherency of the two panels than picking out differences between particular events. The result using PEF smoothing, in the left panel, is noticeably more coherent, particularly between 1.2 and 1.6 seconds.

CONCLUSION

We have a great deal of freedom in choosing how to distribute PEFs in the data coordinates, and in choosing how to implement the smoother between PEFs. Choosing the web-like arrangement of micropatches and smoothing PEFs in radial coordinates gives a nice flat smoother response, and the ability to use relatively large micropatches without trouble related to nonstationarity. The large size of the micropatches calls into question whether smoothing is actually necessary. It turns out that, especially where data are noisy, smoothing continues to be important.

REFERENCES

Brown, M., Clapp, R. G., and Marfurt, K., 1999, Predictive signal/noise separation of groundroll-contaminated data: SEP–**102**, 111–128.

Claerbout, J. F., 1997, Geophysical exploration by example: Environmental soundings image enhancement: Stanford Exploration Project.

Claerbout, J. F., 1998, Multi-dimensional recursive filtering via the helix: Geophysics, **63**, no. 5, 1532–1541.

Clapp, R. G., and Biondi, B. L., 1998, Regularizing time tomography with steering filters: SEP–**97**, 137–146.

Clapp, R. G., and Brown, M., 1999, Applying sep's latest tricks to the multiple suppression problem: SEP–**102**, 91–100.

Crawley, S., 1999, Interpolation with smoothly nonstationary prediction-error filters: SEP–**100**, 181–196.

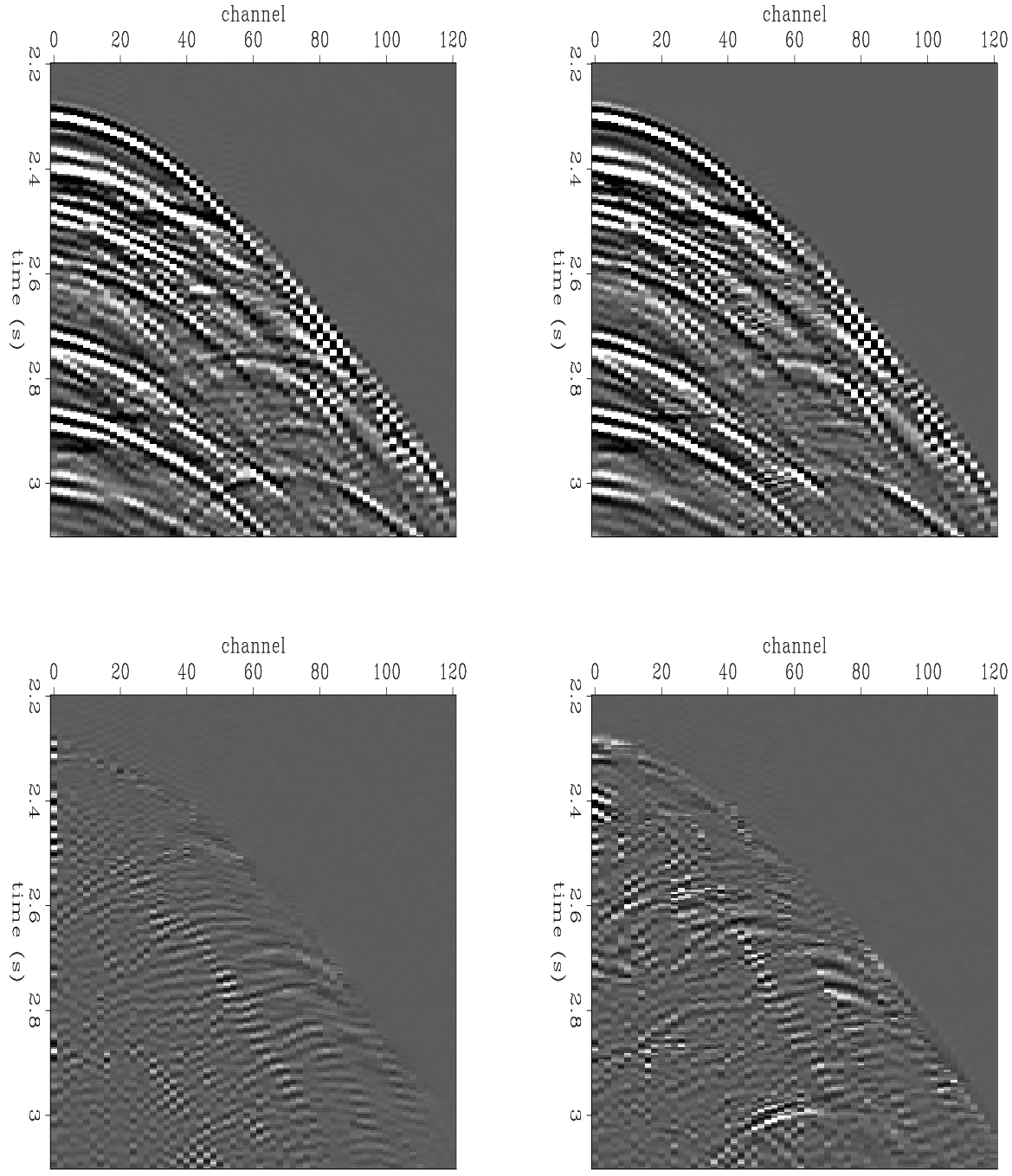

Figure 3: Noise-free interpolated traces and difference from original data. Traces were interpolated with PEF smoothing on the left, without on the right. Known data is not shown, to make the differences easier to see. The results are similar, but the result with smoothing is better. sean1-smonosmo [ER]

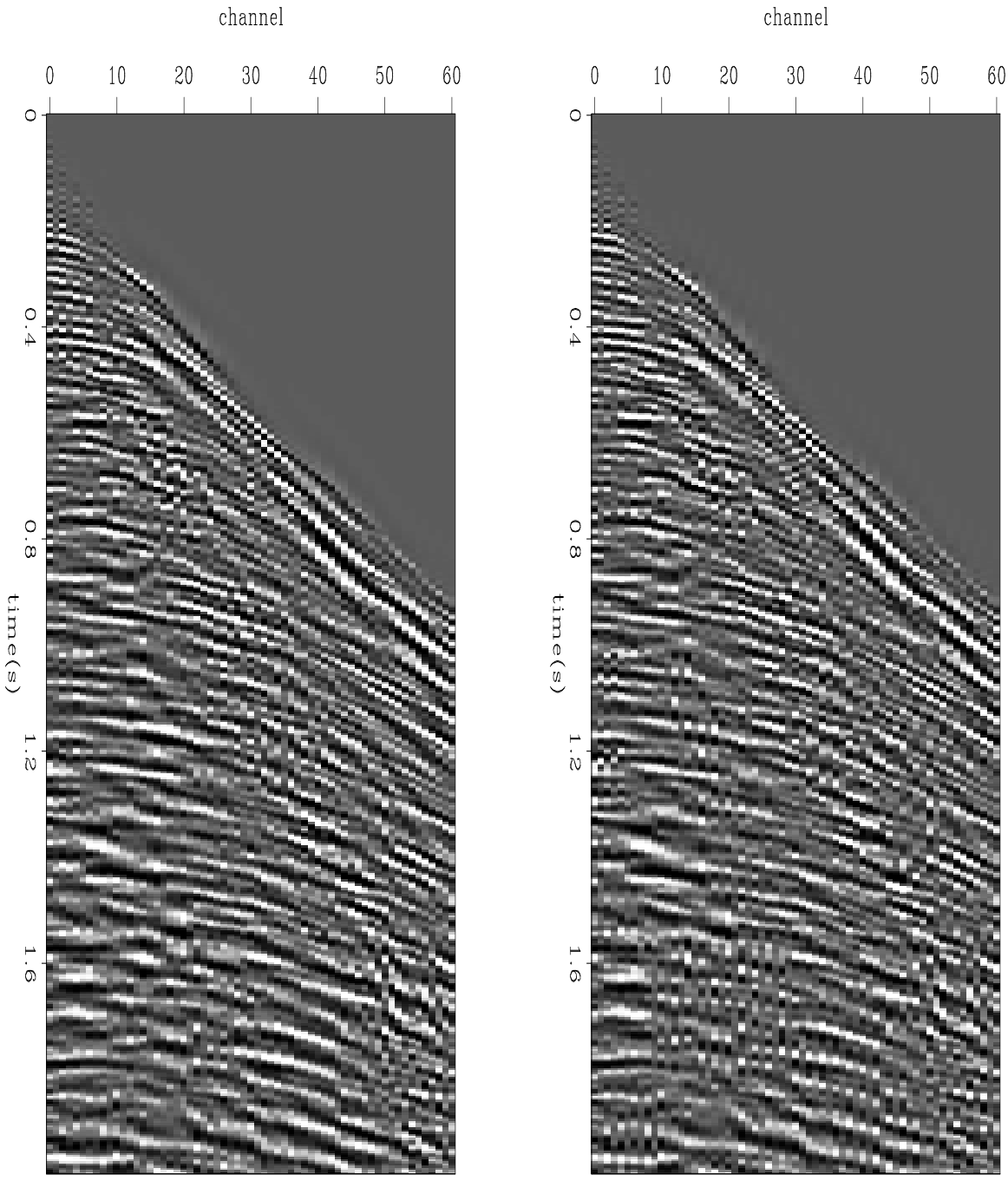

Figure 4: Noisy interpolated traces and differences. Traces were interpolated with PEF smoothing on the left, without on the right. Differences are more visible here on noisy data than on noise-free data. Traces interpolated without PEF smoothing abruptly change from coherent to incoherent along certain micropatch boundaries. sean1-smonosmo2 [ER]

Fomel, S., 1999, Plane wave prediction in 3-D: SEP–**102**, 101–110.