Spatial Whitening Improves the Temporal Decon Filter

Jon F. Claerbout

Current industrial practice is to compute the temporal deconvolution filter from the spectrum of the data itself. An alternate approach is to compute the temporal deconvolution filter from the temporal spectrum of the *spatial* second difference (1,−2,1) of the data. The second difference is a simple spatial whitener based on the assumption that the data shows lateral continuity. The technique may also be extended to profiles and gathers before moveout correction. Then the $-\partial_{xx}$ operation is not done at constant t, but the operator is convolved along the unmoved-out hyperbola. Conceptually this is like doing moveout, convolving (−1,2,1) over offset at constant time, and then doing inverse moveout.

Preliminary tests are very encouraging.

Operator Definition

It is well known that the unit-span prediction error filter has a spectrum *inverse* to that of the data itself. Because this implies a white output, which no one wants to see, it is customary to post filter, or instead to use a gapped prediction error filter. Alternately -- and this amounts to about the same thing as gapping -- a disposable copy of the data may be prewhitened with a short (say 5 term) preliminary deconvolution before the final deconvolution filter is computed from the prewhitened copy. The idea being proposed by this paper is to add another stage of prewhitening, a spatial whitening. Because prewhitening weakens signal in the disposable copy and because the final filter spectrum is *inverse* to that of the copy, signal emerging from the final filter is strengthened. An acronym for these filters is SWED filters -- Spatially Whitened Estimated Deconvolution filters.

Since we are merely testing the spatial whitening concept, not optimizing it, the simplest spatial whitening filter will suffice. The data copy will be whitened by convolving on

the horizontal space axis with the (1,−2,1)-operator. This should destroy lateral continuity leaving a whiter spatial spectrum. We will first ignore moveout by examining a constant offset section, a previously moveout corrected gather, and a CDP stack. Then we will test raw unmoved-out profiles.

I was originally drawn to this idea by noticing that nearby traces with equal power in one time window may have unequal power in a later time window. Two ideas to explain this observation were (1) some kind of focusing due to lateral velocity variation or more simply, (2) interference of quasimonochromatic waves changing laterally as the pseudorandom reflection coefficient series changes. If the problem is one of interference, then the solution is one of deconvolution. The observation suggests designing the deconvolution filter to try to make each trace equal to the average of its neighbors, in other words, to minimize the power in $\partial_{xx} P$ instead of that in P itself.

Results

This idea was conceived shortly before the report deadline. There was no time to tune parameters or select the best of many trials. All results are shown. They may not turn out to be typical, but they certainly are realistic. Selectable parameters are the filter length *nfpost* and the length of the temporal prewhitening filter *nfpre*. Prewhitening was done with a five term filter in every case. *Nfpost* was either twenty points or forty points, depending on the circumstances. When a single filter is being used for all traces then the filter length defaults to forty. When a separate filter is being computed for each trace then the filter length defaults to twenty. My imagined "industry standard" program computes a separate filter of length twenty for each trace. All programs were based on the Burg method (see FGDP and SEP-10). All the programs prefer gain control to be done first, and where this was not already done, I used t^2. Ultimately, filtering should be done before, not after, t^2 scaling, but this degree of perfection does not yet exist in my programs.

Constant-Offset Section Example

The first example tested was a constant-offset section from the California Central Valley. It is a subset of a data set that John Toldi got from Chevron. The data exhibits little dip, but it has substantial statics and lateral velocity variation. Statics corrections were not made before these tests. The energy source was Vibroseis (registered trademark of CONOCO). I don't mind the symmetry of the Vibroseis wavelet autocorrelation, because I believe it to be short compared to the downgoing waveform. We are trying to deconvolve something much more like the downgoing waveform than like the autocorrelation of the

Vibroseis wavelet. Experience bore out this idea. I selected the particular offset at random from the middle of the spread. The data had already been multiplied by t^2 but not moveout corrected.

Figure 1a shows the input constant-offset section (COS). Figure 1b shows the output of the presumed industry-standard decon. Figure 1c shows SWED with a single filter being applied to all traces. Figure 1d shows SWED with a separate filter for each trace. I was delighted by these first results, especially figure 1c between shotpoints 0-20 from 2-3sec near 3 sec. Blinking the CRT display between figures 1a and 1c shows the deconvolution moving energy packets up the time section as much as 200ms. Surprisingly, even packet onsets seemed to move up 20-40ms. Examining the time scales carefully you can see that this is so. (The plots were separately scaled). Since the raw data has a quasimonochromatic look, a large motion of the envelope should not be surprising. But the first arrival shouldn't move, so I double checked that the program does indeed apply a causal filter. You will be able to confirm causality of the program on a later hard bottom marine profile where arrivals are more abrupt. I was delighted to get these results on the first data tested.

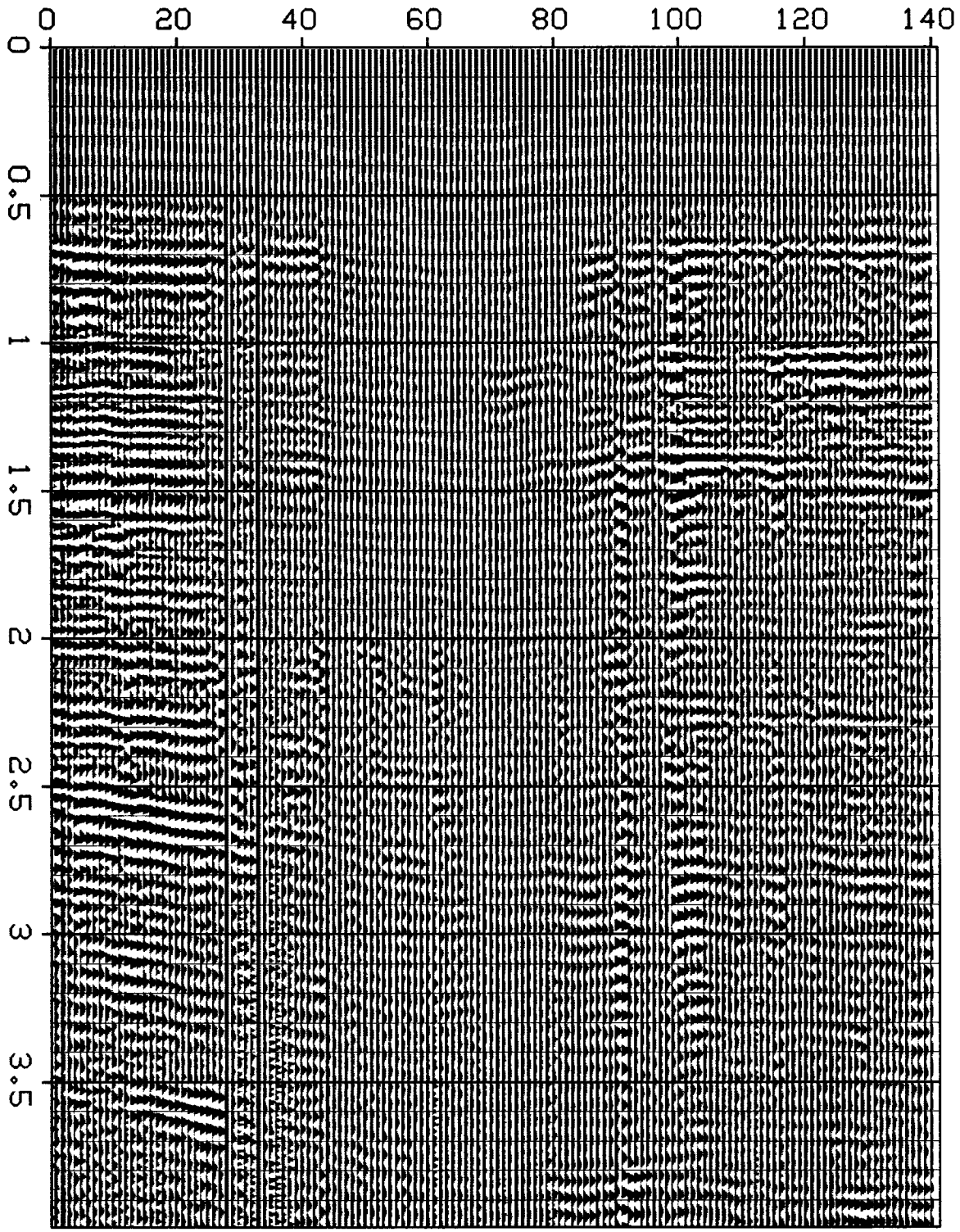


FIG. 1a. Constant offset section, California Central Valley.

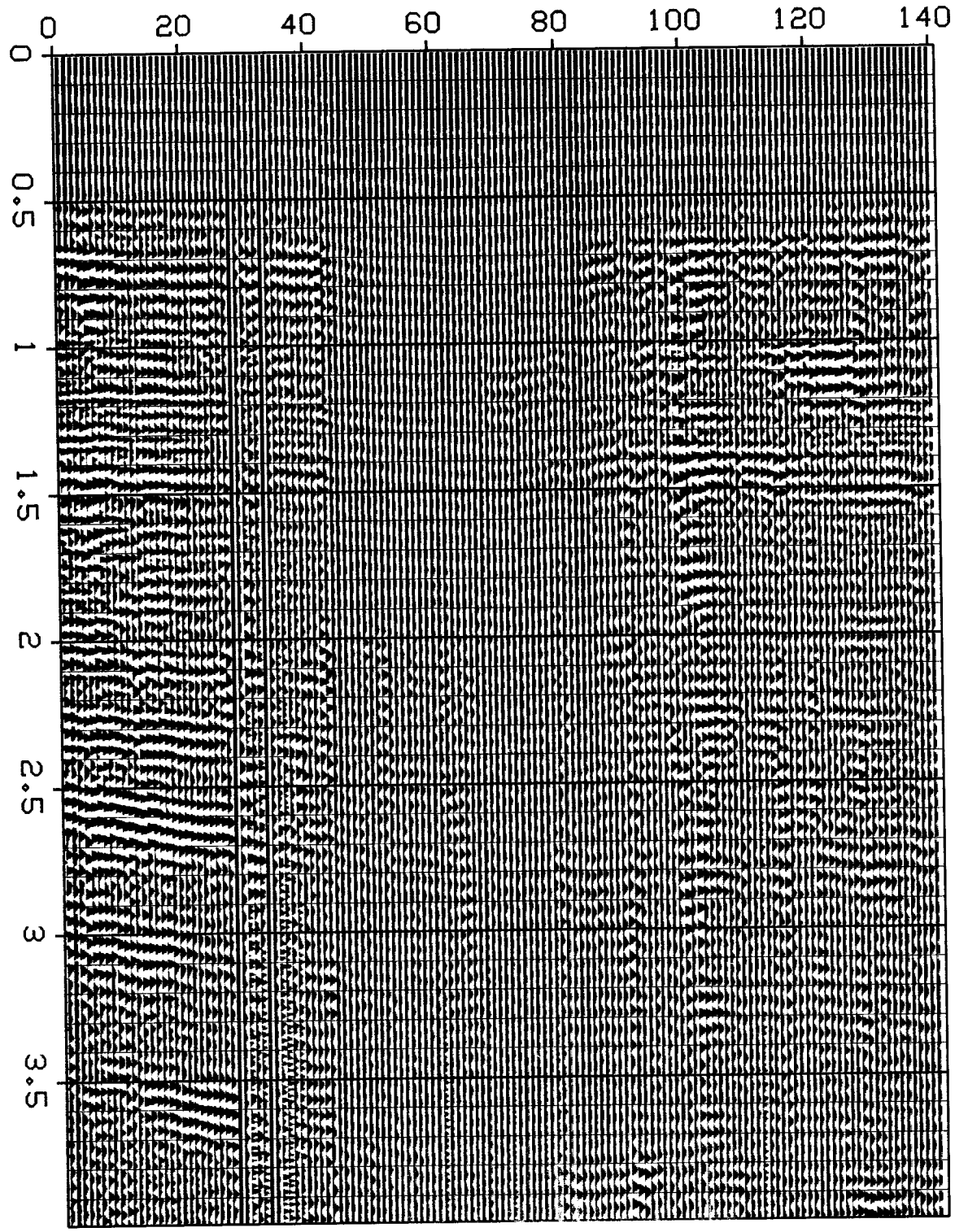


FIG. 1b. Processed by standard decon.

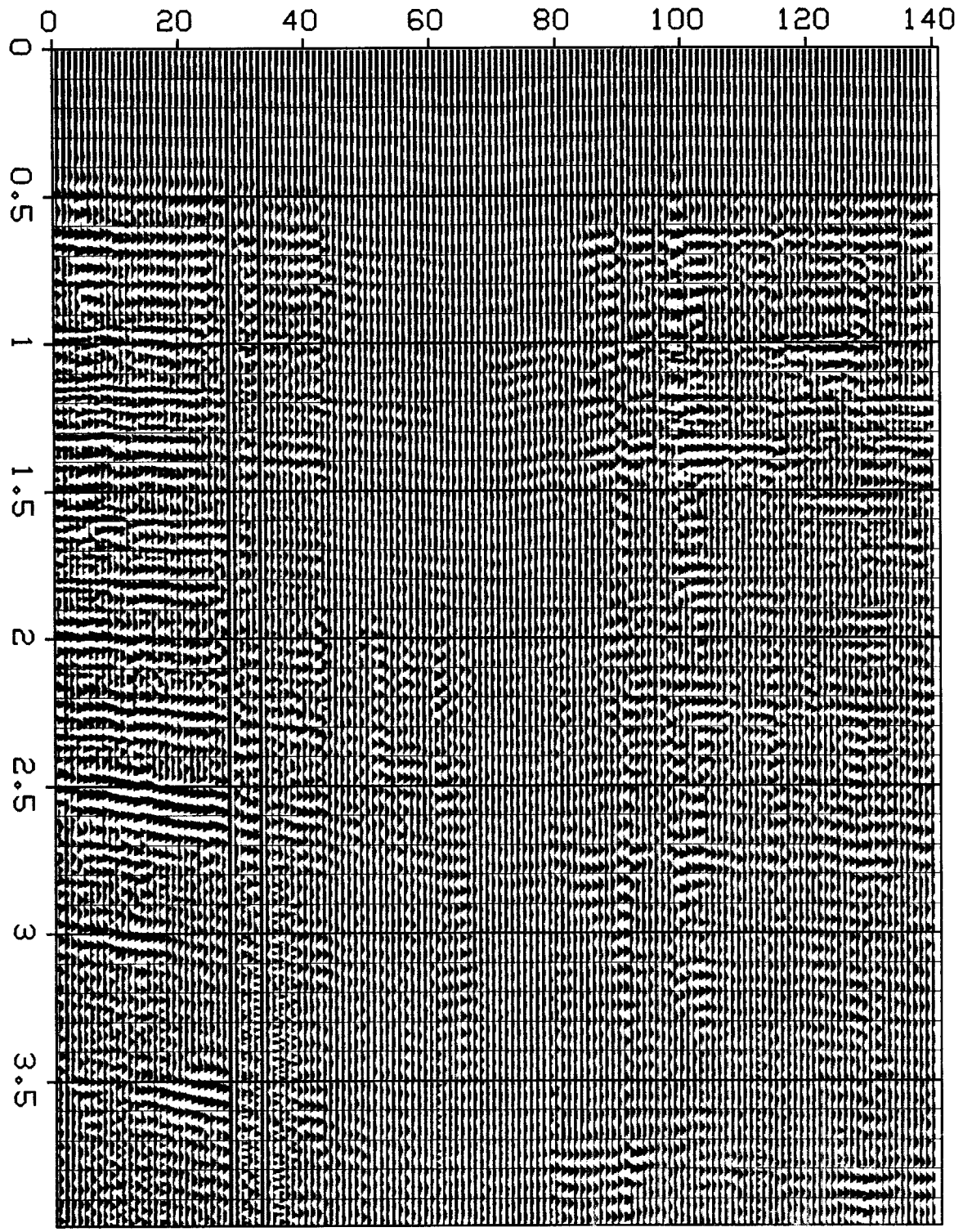


FIG. 1c. SWED with one filter for the whole frame.

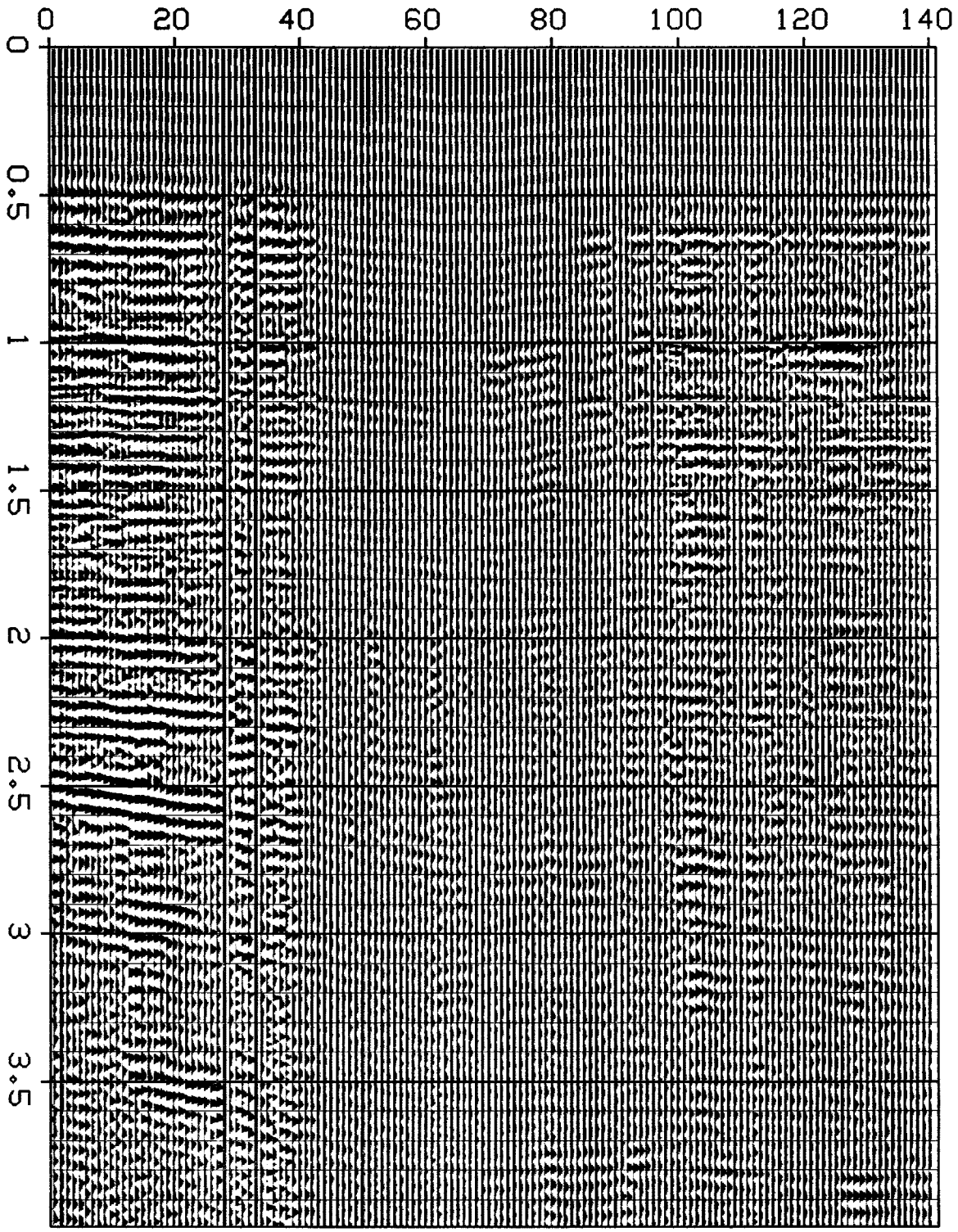


FIG. 1d. SWED with a different filter for each trace.

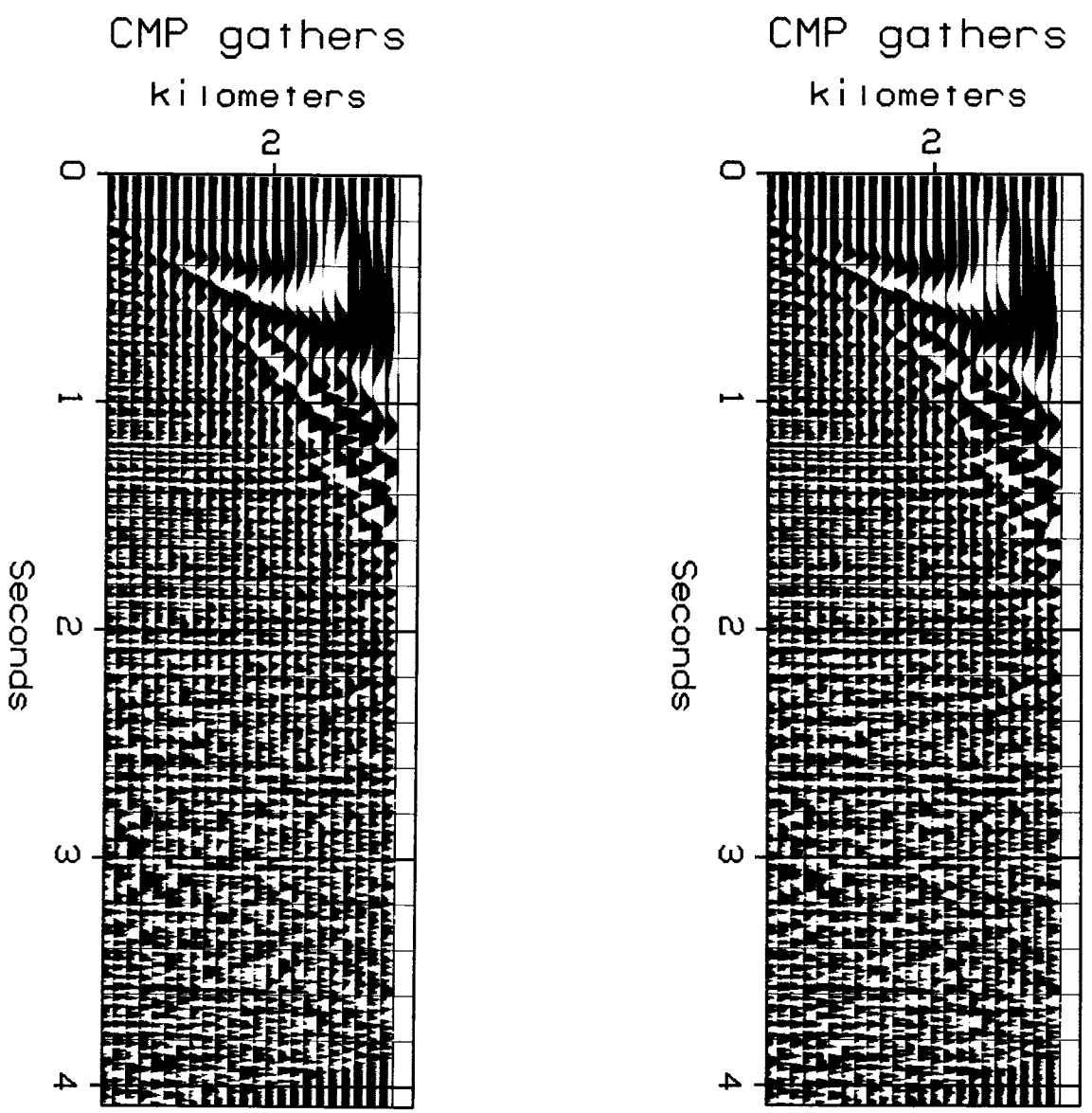


FIG. 2ab. (a) Hales CDP Gather. (b) Conventional decon.

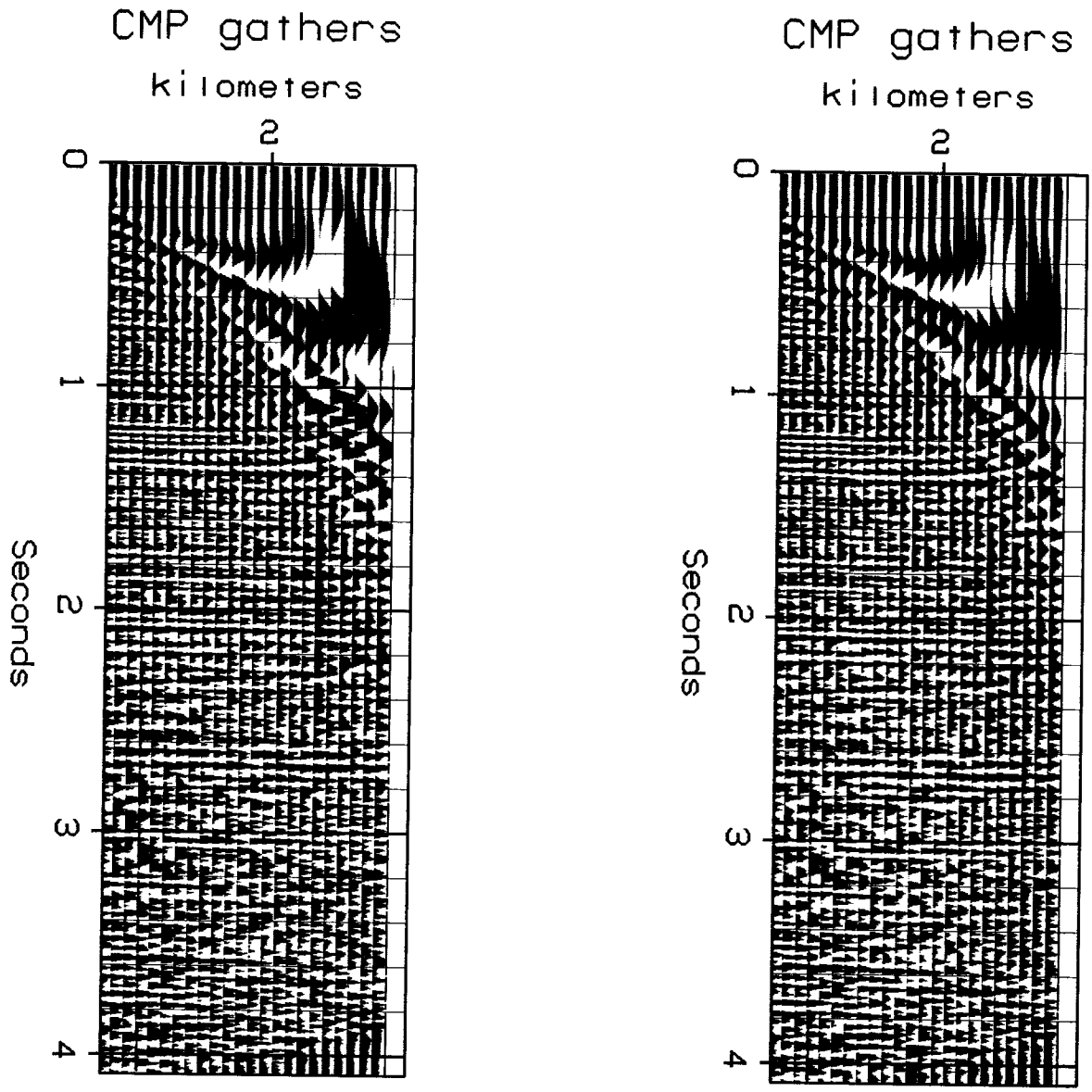


FIG. 2cd. (c) SWED, one filter/plane. (d) SWED, one filter/trace.

Moveout-Corrected CDP-Gather Example

The CDP gather selected from the SEP files originally came from Western Geophysical, and had been previously used by Dave Hale (see SEP-36, p. 32) in his dip moveout work. The data has a previous gain and decon history which is uncertain, but it may be Western's standard. The data was moveout corrected. A fault plane nearby shows an overcorrected arrival on figure 2a at about 2.2sec. Figure 2b shows the industry standard. Figure 2c shows SWED with a single filter for all traces. Figure 2d shows SWED with a separate filter for each trace. All three decon processes suppress a low frequency slow noise that may be seen on figure 2a and in Hale's original plots. Subjectively we found that the data was more clearly seen on the video terminals than on any hard copy I could produce. Indeed some adjustment of plotting parameters was required in order to get the hard copy results as clear as they are. From the hard copy results I cannot say you should prefer figure 2c but that seemed to be our conclusion from viewing the video screen and blinking back and forth.

CDP Stack Example

Shuki Ronen got some data from Israel and used it for Statics studies. This data was previously AGCed, statics shifted, and stacked. Because of numerous near surface problems, and messy initial results, only the last half of the time axis was used. In spite of this heavy-handed muting, we all agreed that results were poor. None-the-less, they were instructive. Figure 3a is the stack. Figure 3b is the conventional decon Figure 3c is SWED with a single filter for all traces. Figure 3d is SWED with a separate filter for each trace. Although the results are messy, all agreed that the industry standard was better than the two SWED cases. This bad result warrants a thorough analysis of this example.

First notice that the conventional decon in figure 3b has introduced a certain roughness from trace to trace. This is because a separate decon filter is being designed for each trace. SWED in figure 3d has the same problem but with SWED it is more severe, disturbingly so on video displays. Now we must recognize that there are many valid reasons why an ideal decon filter would change from trace to trace. It seems from this example, and it may be more generally so, that a SWED filter requires more spatial averaging than a conventional one. To limit the growing scope of the study I decided to eliminate the question of spatial variation in the filter itself. This question raises too many side issues of how to smooth. Henceforth, in this paper, my definition of "conventional decon" means that a single filter will be used on the entire data plane under examination. To further ensure comparability, the industry and the SWED decon programs were merged into one, with an internal switch *qdxx* for the spatial prewhitening. The programs with filters varying from trace to trace were set aside.

Another question is why figure 1b, SWED with a single filter, came out so low frequency and thus apparently inferior to conventional. The overall frequency content is not a valid objection to either SWED or conventional since it is more or less predetermined by the selection of the parameter *nfpre*.

Another question is whether we should deconvolve CDP stacks at all. Theoretically NMO stretch distorts a consistent temporal wavelet. I realize that in practice people do deconvolve CDP stacks and get good results, but that may be incidental. Conventional decon of CDP stacks may be little more than a conveniently parameterized filter. The parameters are partly determined by the data itself and partly determined by the operator. On a CDP stack, conventional decon may be only remotely related to waveform estimation. Let us face headlong the the question of NMO.

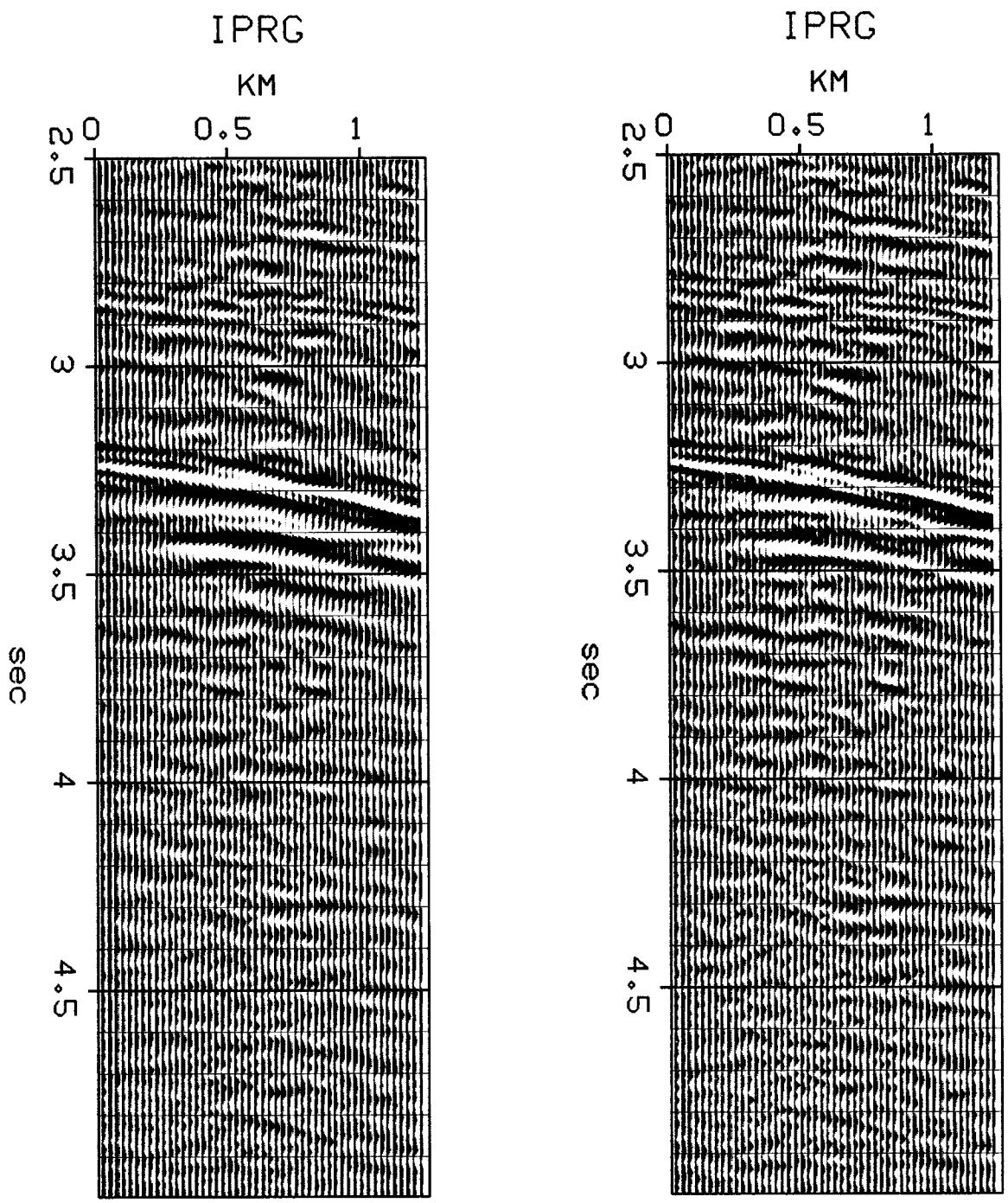


FIG. 3ab. (a) Part of CDP stack. (b) Conventional decon.

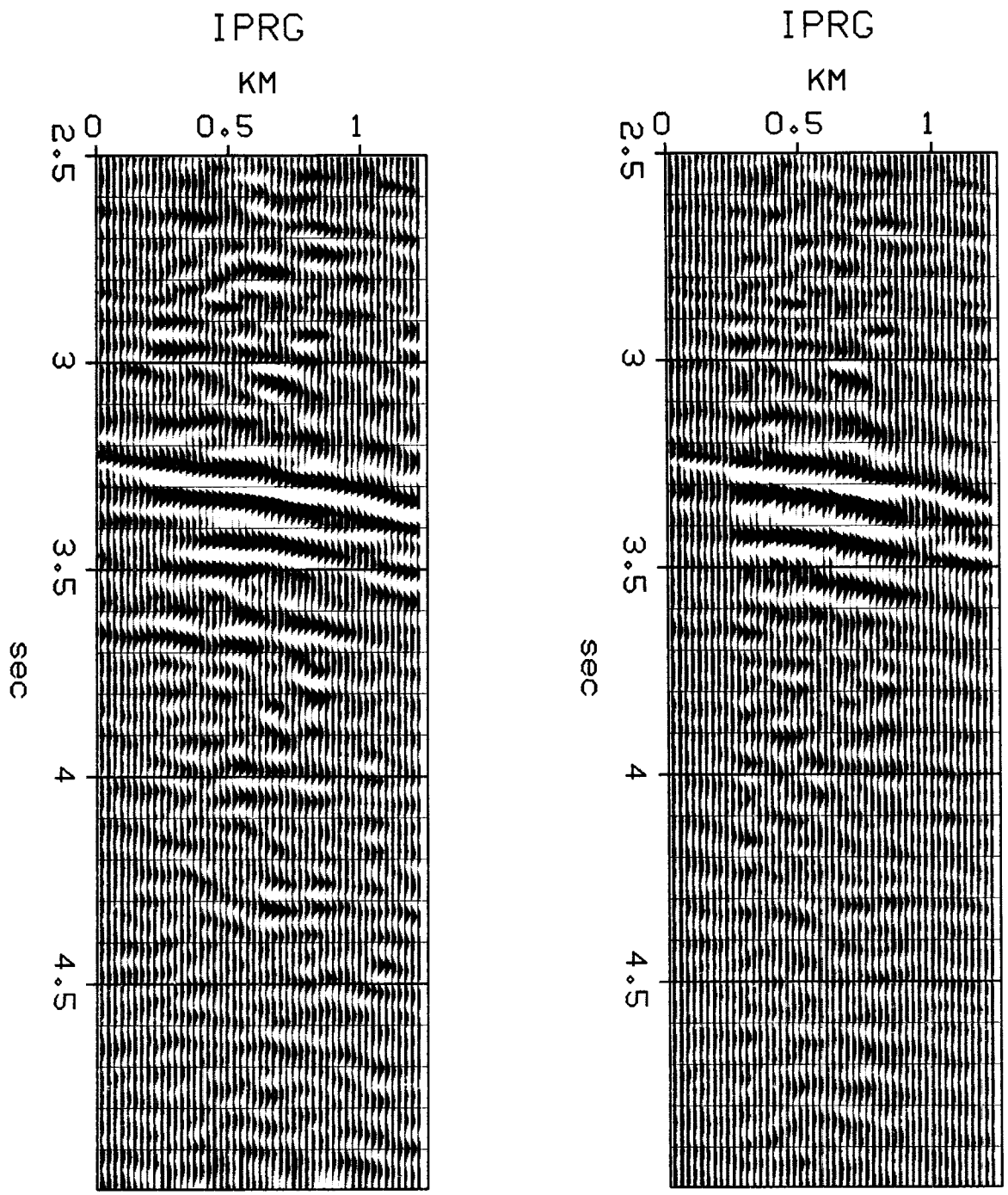


FIG. 3cd. (c) SWED, one filter/plane. (d) SWED, one filter/trace.

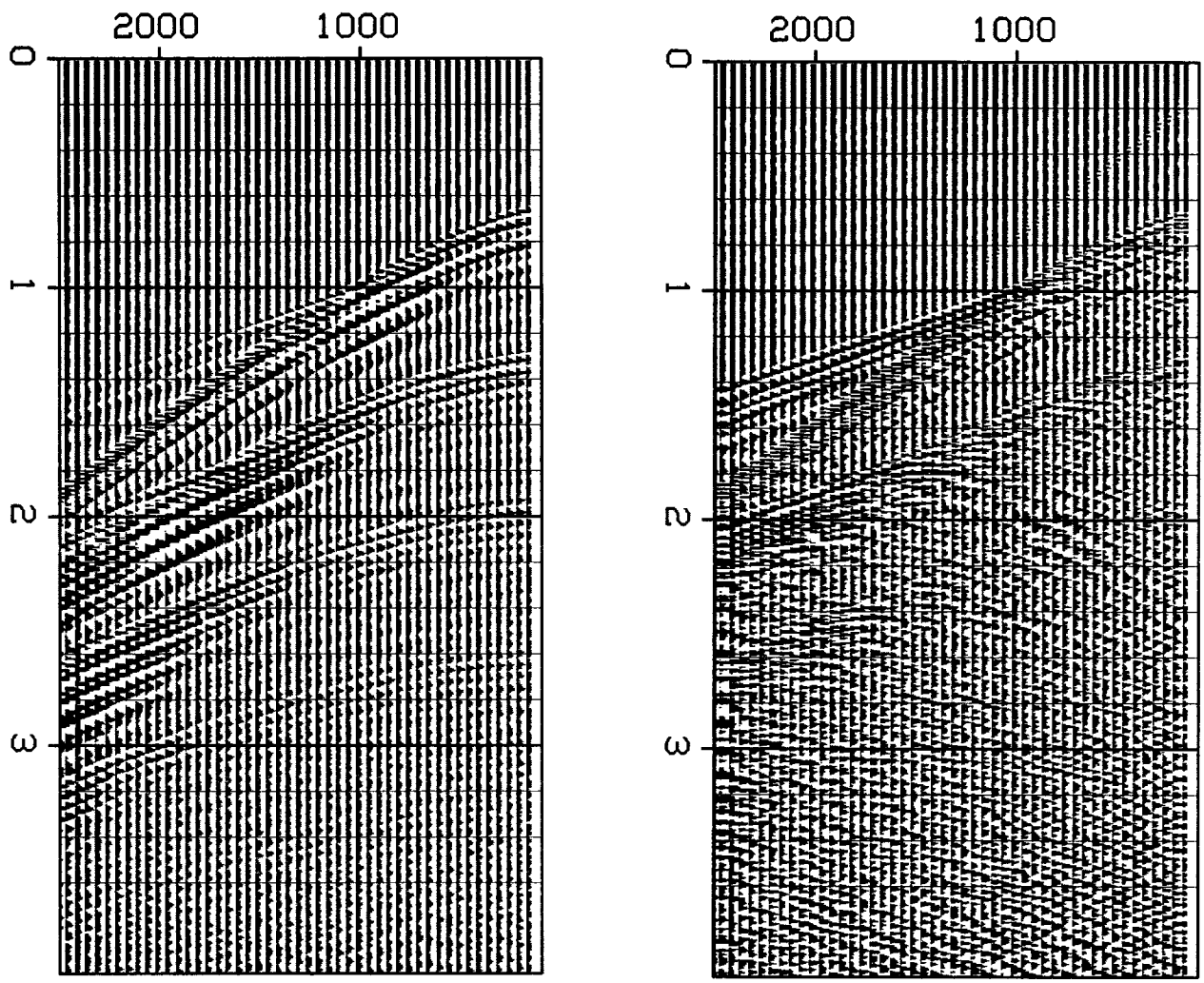


FIG. 4ab. (a) Marine profile (b) Spatially whitened at 1500m/s.

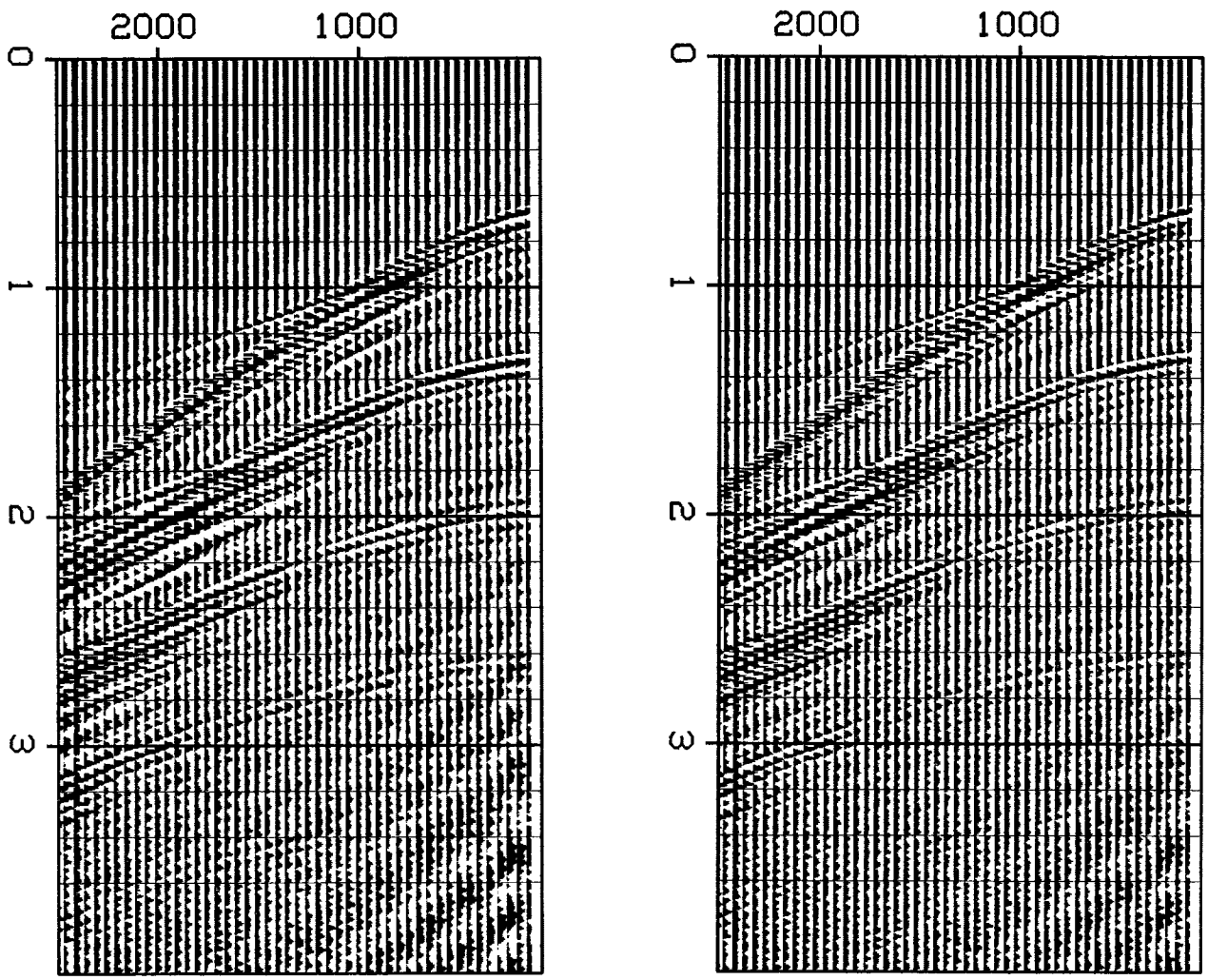


FIG. 5ab. (a) Conventional decon. (b) SWED.

Spatial Whitening with NMO

Spatial whitening of unmoveout-corrected profiles means the suppression of the dominant hyperbolas. NMO will certainly be important when we get around to using SWED to estimate surface-consistant filters. At first glance, data exhibiting NMO would seem to present a problem for the simplest SWED spatial operator $-\partial^2/\partial x^2$. That operator may whiten hyperbola tops, but leaves a mess on the flanks. Conceptually, the solution is to do NMO, apply the spatial whitening operator, and then do inverse NMO, before decon filter design. In practice, I found it easier and better to "bend" the operator and apply it on the original unmoved-out space. I loop over the data space applying the +2 everywhere. To find the place where the -1's belong, I move away from the +2 with the well-known relation

$$dt = \frac{x}{v_{RMS}^2\, t}\, dx \tag{1}$$

Then I linearly interpolate the time axis. This procedure requires neither interval velocity nor square root computations.

Figure 4a shows a hard bottom marine data set used for testing. This data came from GECO and has been previously used by Jeff Thorson. Figure 4b shows the same data after rejecting water velocity. Most of the water hyperbolas have gone leaving a variety of interesting refracted and diffracted events and noises. I experimented a bit with various velocities and found the total power on the gather to be a sensitive function of the velocity. This suggests an alternative to velocity analysis by semblance. Instead of passing hyperbolas and plotting power, we could *reject* the hyperbolas and plot the power inverse. This scheme, like Burg's spectral scheme, should be better than conventional when observed functions are badly truncated at the ends of the observation interval. I suspect that CDP stacking would work better on figure 4b than on the raw data 4a because cable truncations of water velocity events should be much weaker.

Marine Shot Profile Deconvolution

Figure 5a and 5b show deconvolutions of figure 4a. In both cases a single filter was used on the entire frame. Conventional decon is in 5a. The SWED result is in 5b. To repeat, the SWED method is just conventional deconvolution, but the filter is designed on figure 4b instead of 4a. All observers agreed that both deconvolutions are better than the original data. The results of the two methods are significantly different. All local observers of the video displays agreed that SWED was significantly better than the conventional. Notice the noise level between the water bottom events at wide offsets. Even the head wave seems more parsimonious with SWED. In the hard copies you may notice the two sedimentary

primaries at 1.33sec and 1.43sec and their peglegs at 2.15sec and 2.23sec. Unfortunately, these primaries don't seem to extend into the zone where SWED has cleaned the space between the multiples.

Conclusion

I am pleased by these results and am looking forward to more applications.

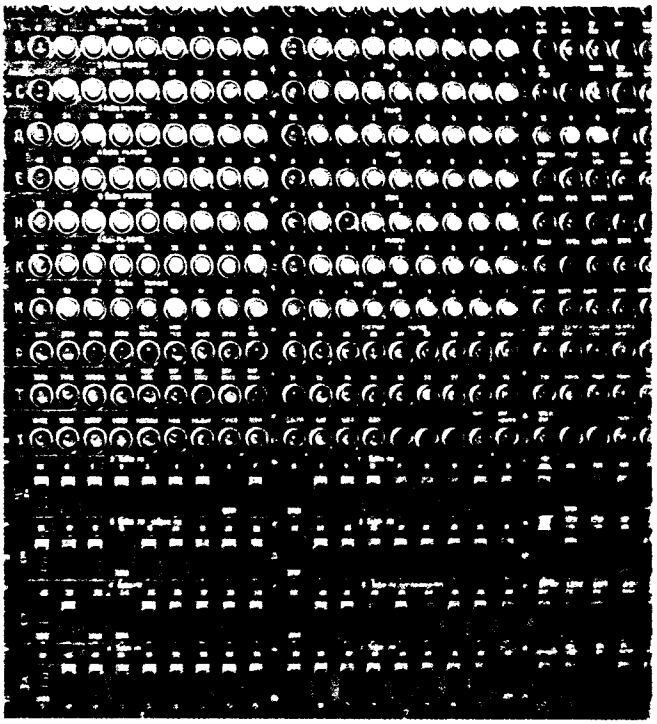
REFERENCES

Claerbout, J.F., 1976, LEVITY: Levinson Recursion Reprogrammed, SEP-10, p 90.
Claerbout, J.F., 1982, Envelope Sensing Decon, SEP-30, p. 121

Programs

```
# Spatial whitening decon. Jon Claerbout. ref. SEP 38.
# ep=input ep=output (em,fp,fm)=temps
subroutine deconx (qdxx,slow,t0,dt,x0,dx,nfpre,nfpost,nt,nx,ep,em,fp,fm)
real slow, t0, dt, x0, dx
real ep(nt,nx), em(nt,nx), fp(nt,nx), fm(nt,nx), top, bot, dot, adot, c
integer qdxx, it, nt, ix, nx, j, nfpre, nfpost
real hp(1000), hm(1000)                 # good for filters to 500 points.
do it = 1, 2*nfpost-1
        hp(it) = 0.
hp(nfpost) = 1.                         # The decon filter.
do it = 1, 2*nfpost-1
        hm(it) = hp(it)
do ix=1,nx {
        do it=1,nt {
                em(it,ix) = ep(it,ix)
                fp(it,ix) = ep(it,ix)
                fm(it,ix) = ep(it,ix)
                }
        }
if( qdxx != 0 )                         # Spatial prewhiten if they want.
        call dxx(slow,t0,dt,x0,dx,nt,nx,fp,fm)
do ix=2,nx-1 {
        call norm (nt, fm(1,ix) )       # Don't get wiped out by wild traces.
        do it=1,nt
                fp(it,ix) = fm(it,ix)
        }
do j = 2, nfpre {               # temporal prewhitening.
        top=0.;                 bot=0.
        do ix=1,nx {
                top = top + dot(nt-j+1, fp(j,ix), fm(1,ix) )
                bot = bot + dot(nt-j+1, fp(j,ix), fp(j,ix) )
                bot = bot + dot(nt-j+1, fm(1,ix), fm(1,ix) )
                }
        if ( bot == 0. )
                call err("deconx: Data vanishes")
        c = 2 * top / bot
        do ix=1,nx
                call scub( c, nt-j+1, fm(1,ix), fp(j,ix) )
        }
# Filter design and apply
do j = nfpre+1, nfpost {
        top=0.;                 bot=0.
        do ix=2, nx-1 {
                top = top + adot( nt-j+1, fm(1,ix), fp(j,ix) )
                top = top + adot( nt-j+1, fp(j,ix), fm(1,ix) )
                bot = bot + adot( nt-j+1, fp(j,ix), fp(j,ix) )
                bot = bot + adot( nt-j+1, fm(1,ix), fm(1,ix) )
                }
        if ( bot == 0. )
                call err("deconx: Data vanishes")
        c = top / bot
        do ix=2,nx-1
                call scub( c, nt-j+1, fp(j,ix), fm(1,ix) )
        do ix=1,nx
                call scub( c, nt-j+1, ep(j,ix), em(1,ix) )
        call scub (c, (2*nfpost-1)-j+1, hp(j), hm(1) )
        }
return
end

subroutine scub(c, nt, y, x)
real c, y(nt), x(nt), tempe
integer it, nt
do it = 1, nt {
        tempe = y(it) - c * x(it)
        x(it) = x(it) - c * y(it)
        y(it) = tempe
        }
return
end

real function dot(nt,x,y)
integer it,nt
real x(nt), y(nt), sum
sum = 0.
do it=1,nt
        sum = sum + x(it) * y(it)
dot=sum
return
end
```

```
real function adot(nt,x,y)
# Statistically a more robust inner product.
integer it, nt
real x(nt), y(nt), sum
sum = 0.
do it=1,nt
        if ( x(it)  > 0.0 )
                sum = sum + y(it)
        else
                sum = sum - y(it)
adot=sum
return
end

subroutine norm( nt, v )
integer it, nt
real v(nt), big
big = 1.e-20
do it=1,nt
        if( abs(v(it)) > big )
                big = abs(v(it))
big = 1. / big
do it=1,nt
        v(it) = v(it) * big
return
end

# Spatially whiten an unmoved-out gather as though it were moved out.
#  by Q = {partial sup 2 P} over {partial x sup 2} at constant z
# I.e  Q  =  InvNMO (- Partial xx) NMO  P
#  apology -- This routine is slow, but it goes as nt*nx
#               and will be with stuff going as nt*nx*nfilt
subroutine dxx( slow, t0, dt, x0, dx, nt, nx, p, q )
real slow, t0, dt, x0, dx, p(nt,nx), q(nt,nx)
integer it, nt, ix, nx, itl, itr
real t, x, delt, tl, tr, tlo, tro, pl, pr
if( nx < 3 )
        call err( "dxx:  Not enough channels")
if( dt == 0.)
        call err("dxx: dt = 0")
if( slow == 0. ) {
        do ix=2, nx-1
                do it=1,nt
                        q(it,ix) = -p(it,ix-1) + 2.*p(it,ix) - p(it,ix+1)
        }
else {
        do ix = 2, nx-1 {  # Two nested loops, order insignificant.
        do it = 1, nt   {  # Two nested loops, order insignificant.
                x = x0 + (ix-1)*dx
                t = t0 +  it   *dt
                delt = ( x * slow  * slow / t ) * dx
                tl = t-t0 - delt;  itl = tl/dt;  tlo=itl*dt;
                tr = t-t0 + delt;  itr = tr/dt;  tro=itr*dt;
                if ( itl+1 >nt | itl <1 | itr+1>nt | itr<1 ) # off mesh
                        q(it,ix) = 0.
                else {                  # Interpolate
                        pl = (dt+tlo-tl) * p(itl,ix-1) + (tl-tlo) * p(itl+1,ix-1)
                        pr = (dt+tro-tr) * p(itr,ix+1) + (tr-tro) * p(itr+1,ix+1)
                        q(it,ix) = 2.*p(it,ix) - (pl + pr) / dt
                        }
                }}
        }
do it = 1, nt {
        q(it, 1)  = q(it, 2)
        q(it, nx) = q(it, nx-1)
        }
return
end
```



СИБИРСКОЕ ОТДЕЛЕНИЕ
АКАДЕМИИ НАУК СССР

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

НОВОСИБИРСК

SIBERIAN DIVISION
OF THE USSR
ACADEMY OF SCIENCES

COMPUTING CENTER

NOVOSIBIRSK