Surface-consistent prediction-error filtering

Stewart A. Levin

INTRODUCTION

In SEP-44 I described a method for surface-consistent trace balancing. In that article I set up a nonlinear least-squares problem and solved it for surface-consistent balance factors by Newton and quasi-Newton iterations. I indicated that these same ideas could also be applied to designing prediction-error deconvolution filters. Here I will detail this application and show some surface-consistent results.

WHAT IS J TRANSPOSE?

Let me review the basic framework I used in SEP-44. Nonlinear least-squares tries to minimize a functional of the form

$$1/2 \quad || f ||^2 \quad . \tag{1}$$

Setting its derivative to zero produces

$$J^T f \;=\; 0 \quad . \tag{2}$$

The second derivative of the function may be written

$$J^T J + K \tag{3}$$

where

$$K \;=\; \sum_i f_i \nabla^2 f_i \quad . \tag{4}$$

The Newton equations

$$(J^T J + K)\, \delta x \;=\; -J^T f \tag{5}$$

determine the location of a zero of the quadratic approximation to the functional obtained by truncating a Taylor expansion after second order.

In the Gauss-Newton method one assumes that K is small compared to $J^T J$ and looks instead at the equations

$$J^T J\; \delta x \;=\; -J^T f \tag{6}$$

which are the normal equations for the classic linear least-squares problem

$$J\; \delta x \;\approx\; -f \quad . \tag{7}$$

Now let's see what some of these objects are for some simple deconvolution problems.

1) Single Channel deconvolution

Here the vector f is given by

$$f = d - \mathbf{SH}(d)b \tag{8}$$

where d is the data trace, b is the unknown debubble filter, and $\mathbf{SH}(d)$ is a matrix of shifted copies of d:

$$\begin{pmatrix} 0 & 0 & 0 & . \\ d_o & 0 & 0 & . \\ d_1 & d_o & 0 & . \\ d_2 & d_1 & d_o & . \\ . & d_2 & d_1 & . \\ . & . & d_2 & . \\ . & . & . & . \\ 0 & . & . & . \\ 0 & 0 & . & . \\ 0 & 0 & 0 & . \end{pmatrix}$$

The gradient equation (2) may be written

$$0 = -\mathbf{SH}(d)^T \, (d - \mathbf{SH}(d)b) \tag{9}$$

which are the normal equations for the least squares problem

$$d \approx \mathbf{SH}(d)b \quad . \tag{10}$$

The second derivative Hessian matrix is

$$\mathbf{SH}(d)^T \, \mathbf{SH}(d) \tag{11}$$

which is $J^T \, J$ and so K is zero in this (linear) problem. From this we see the Newton step satisfies

$$\mathbf{SH}(d)^T \, \mathbf{SH}(d) \, \delta b = \mathbf{SH}(d)^T \, (d - \mathbf{SH}(d)b) \tag{12}$$

again normal equations for

$$\mathbf{SH}(d)\delta b \approx d - \mathbf{SH}(d)b \tag{13}$$

identical to the equation $\nabla\phi = 0$ since the problem was linear. For the same reason the Gauss-Newton equations are identical to the Newton equations.

2) Multichannel deconvolution

By this I mean designing a single filter for a gather of traces. Simply concatenate the data traces d_i and the convolutional matrices $\mathbf{SH}(d_i)$ and the single channel derivation above now applies. To avoid bias, each data trace should be scaled to the same RMS amplitude. This is one form of preconditioning. Other preconditioners are divergence correction and prewhitening.

3) Simultaneous pre- and post-NMO deconvolution (Claerbout, SEP-42)

Here

$$f \;=\; (1 - rev\,) NMO\;\; t\,(1 - bub\,) data \tag{14}$$

and J is

$$-\Big[\;\mathbf{SH}_{rev}\,(NMO\;\; t\,(1 - bub\,)data\,)\;,\;(1 - rev\,)NMO\;\; t\,\mathbf{SH}_{bub}\,(data\,)\;\Big]\;. \tag{15}$$

The Gauss-Newton step is computed from

$$\begin{aligned}&\mathbf{SH}_{rev}\,(NMO\;\; t\,(1 - bub\,)data\,)\delta_{rev}\; + \\ &\qquad (1 - rev\,)NMO\;\; t\,\mathbf{SH}_{bub}\,(data\,)\delta_{bub} \;=\; (1 - rev\,)NMO\;\; t\,(1 - bub\,)data\end{aligned} \tag{16}$$

or more loosely

$$\begin{aligned}&\delta_{rev}\,NMO\;\; t\,(1 - bub\,)data\; + \\ &\qquad (1 - rev\,)NMO\;\; t\;\; \delta_{bub}\,data \;\approx\; (1 - rev\,)NMO\;\; t\,(1 - bub\,)data \quad .\end{aligned} \tag{17}$$

With initial guess $bub\; = rev\; = 0$ this reduces to

$$\delta_{rev}\,NMO\;\; t\;\; data\; + NMO\;\; t\;\; \delta_{bub}\,data \;\approx\; NMO\;\; t\;\; data$$

as in Claerbout's article. In that investigation he elected not to proceed beyond the first Gauss-Newton step, merely indicating how one might proceed.

Forming $J^T J$ we see involves auto- and cross-correlation of data partially deconvolved by either rev or bub but not both. The second derivative is more complex than $J^T J$ because the cascade of rev and bub filters is nonlinear. But there is a component of the nonlinear term K that is independent of the current estimates of the filters rev and bub. The entries of K are the dot products of the various derivatives of J with the vector f. Examining J above we can see immediately that the rev derivatives of the first half of J and the bub derivatives of the second half of J are identically zero. The remaining crossterms are

$$\frac{\partial f}{\partial rev\;\partial bub} \;=\; \mathbf{SH}_{rev}\,(\;NMO\;\; t\;\; \mathbf{SH}_{bub}\,(data\,)\,) \tag{18}$$

Thus K takes the form of a symmetric 2×2 block matrix with the diagonal blocks zero and the off-diagonal blocks are dot products of delayed copies of moveout and gain corrected delayed copies of the data with the current estimate of the deconvolved data. That this is not just correlation of the input and output of decon reflects Jon's underlying observation that neither NMO nor gain correction commutes with debubble (or dereverb) filtering.

4) Surface-consistent deconvolution

I discuss the simplest case and defer simultaneous pre- and post-NMO filtering to a future report. So assume one filter for each shot and one for each receiver. The appropriate generalization of single gather deconvolution is to minimize the norm of f where the component traces of f are

$$f_{ij} = (1 - s_i)(1 - g_j)\, d_{ij} \quad . \tag{19}$$

The gradient J of f takes the form

$$-\left[\ \mathbf{SH}_s(\ (1-g_j)d_{ij}\ )\ ,\ (1-s_i)\mathbf{SH}_g(d_{ij})\ \right] \tag{20}$$

with zero blocks for all other i, j in each row. Invoking commutativity of convolution, we can rewrite this as

$$-\left[\ \mathbf{SH}_{s_i}(\ (1-g_j)d_{ij}\ )\ ,\ \mathbf{SH}_{g_j}(\ (1-s_i)d_{ij}\ )\ \right] \tag{21}$$

With this latter formulation it is immediately apparent that the Gauss-Newton step is found by solving the least squares system

$$\delta s_i(1 - g_j)d_{ij} + \delta g_j(1 - s_i)d_{ij} \approx (1 - g_j)(1 - s_i)d_{ij} \tag{22}$$

Simple enough. But wait. For a conjugate-gradient solution I have to be able to multiply by J and J^T. The dimensions of J are big: NSHOT*LSFILT+NGEO*LGFILT columns and NTRACE*LTRACE rows. Assuming 10 point filters designed for the 275 52-trace CDP gathers I worked with for trace balancing in SEP-42 and SEP-44, this works out to be 3,110 columns and 8,967,168 rows. You can't keep very many 8,967,168 element vectors in core (one, perhaps two, on the Convex). This means that if I want to work directly on the overdetermined system (22), I have to anticipate and manage I/O to and from secondary storage in addition to all the computation involved in this sparse matrix multiply. Alternatively, I can sacrifice the superior numerical conditioning of the QR least-squares approach and use $J^T J$ with, say, SYMMLQ (Paige and Saunders, 1975) to keep all dimensions down to 3,110. In the latter case $y = J^T Jx$ looks like:

```
do itrace=1,ntrace
    strace=conv(1-g(j),trace(itrace))
    gtrace=conv(1-s(i),trace(itrace))
    temptr=conv(xs(i),strace)+conv(xg(j),gtrace)
    ys(i)=ys(i)+xcorr(temptr,strace)
    yg(j)=yg(j)+xcorr(temptr,gtrace)
end do
```

where **conv** is convolution and **xcorr** is cross-correlation. This involves external I/O as well but only to read the traces once per pass.

How much of a sacrifice is it to take the $J^T J$ route? Conventional wisdom is that the number of conjugate-gradient iterations required for convergence increases and, to a lesser extent, numerical accuracy decreases. I tested this experimentally for the problem of designing a single filter for the field profile shown in Figure 1. I chose to design a 132 msec filter with a 60 msec gap. Figure 2 exhibits the filters designed using LSQR and SYMMLQ. Figure 3 displays the corresponding deconvolved gathers. LSQR needed seven iterations to generate its filter; SYMMLQ required ten iterations with comparable tolerances. The low frequency content of the two prediction filters is markedly different but the deconvolutions are nearly identical. This reflects the lack of corresponding low frequencies in the data; my filter design was unconstrained outside the bandwidth of the data. Conclusions: 1) I don't require the additional numerical accuracy of LSQR in order to obtain a reasonable deconvolution of these data, 2) the moderate increase in the number of iterations required by the SYMMLQ design is a modest price to pay for the corresponding simplification in program design and I/O management.

EXAMPLES

I applied the Gauss-Newton method to the normal equations for the least squares system (22) to design shot and geophone consistent filters to a Central Valley dataset. This is the same data I used in SEP-44 for surface-consistent trace balancing although here I used the unbalanced data. I chose, for no compelling reason, to design shot-consistent filters with the same 132 msec length and 60 msec lag I used in the single gather experiments described previously and geophone-consistent filters of length 80 msec with a gap of 44 msec.

Figure 4 is a stack of these data prior to deconvolution. For Figure 5 I designed only shot-consistent filters (that is one filter for each shot profile), deconvolved with them and stacked. In Figure 6 I followed this by an additional pass of designing and applying geophone-consistent prediction-error filters. This is the procedure I used in SEP-41, Fig. 4, abbreviated there as SG. The corresponding filters are shown in Figure 7. For Figure 8 I used the Gauss-Newton method to simultaneously estimate and then apply shot- and geophone-consistent filters (shown in Figure 9) prior to stack. For this dataset there are no significant differences between the cascaded deconvolution of Figure 6 and the simultaneous deconvolution of Figure 8. Playback clips, based on the 99th quantile of amplitudes, are within three percent of each other. From this I find no reason to modify my previous conclusion that for these data surface-consistent deconvolution is no improvement over ordinary single-channel prediction error filtering.

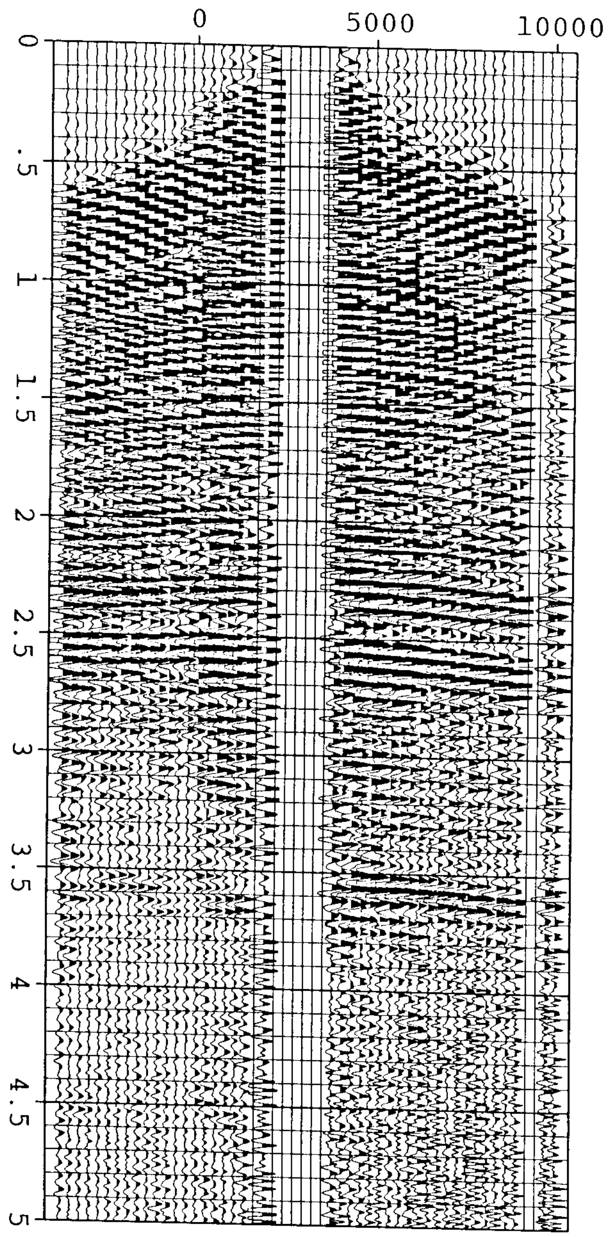
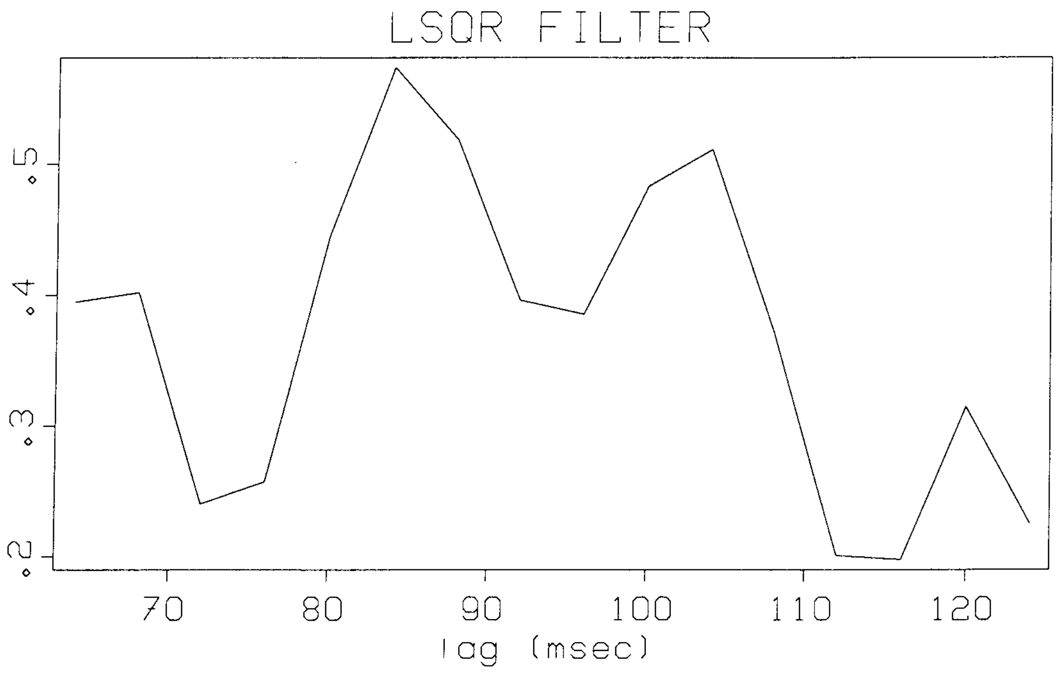


FIG. 1. Field gather from the Central Valley of California I used to compare two prediction-error filter design methods.



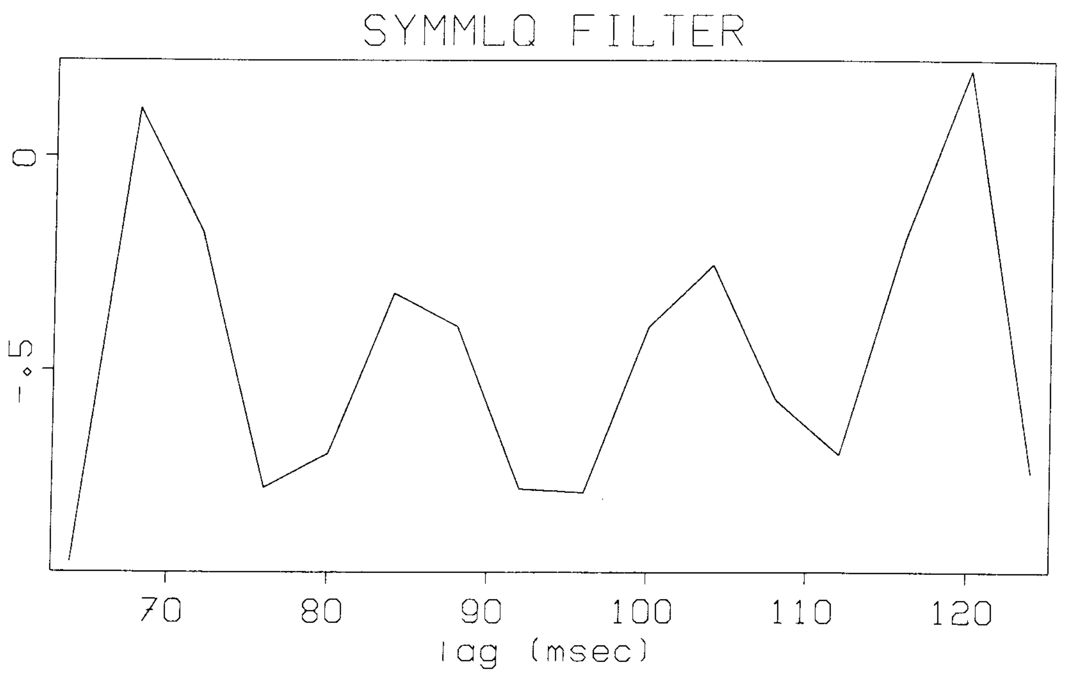
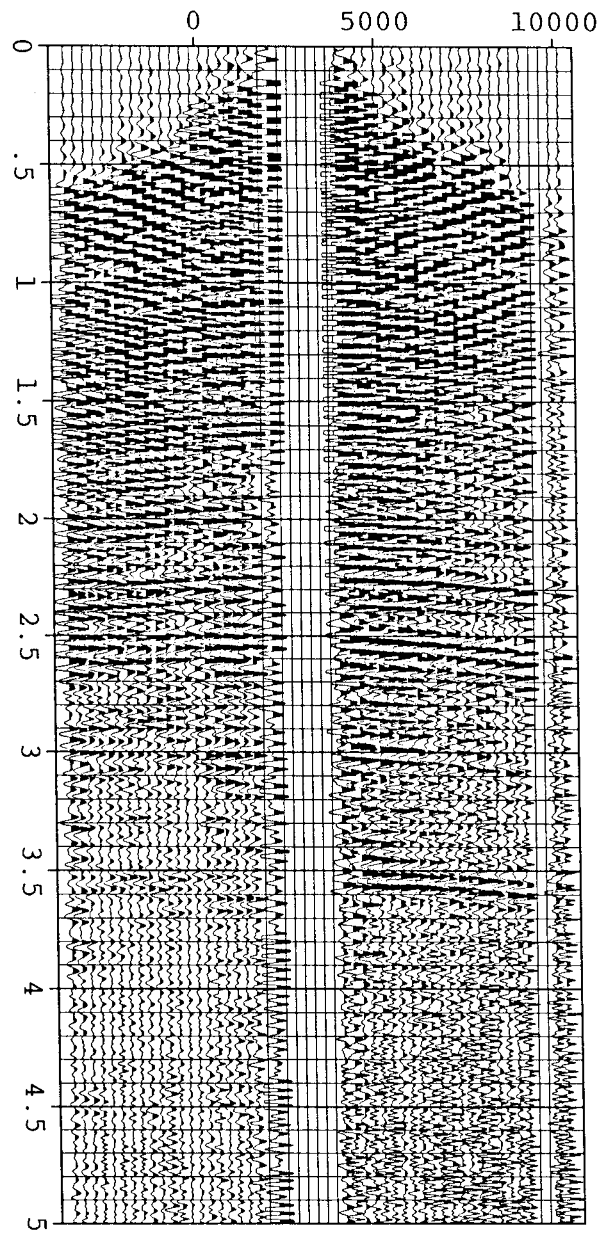
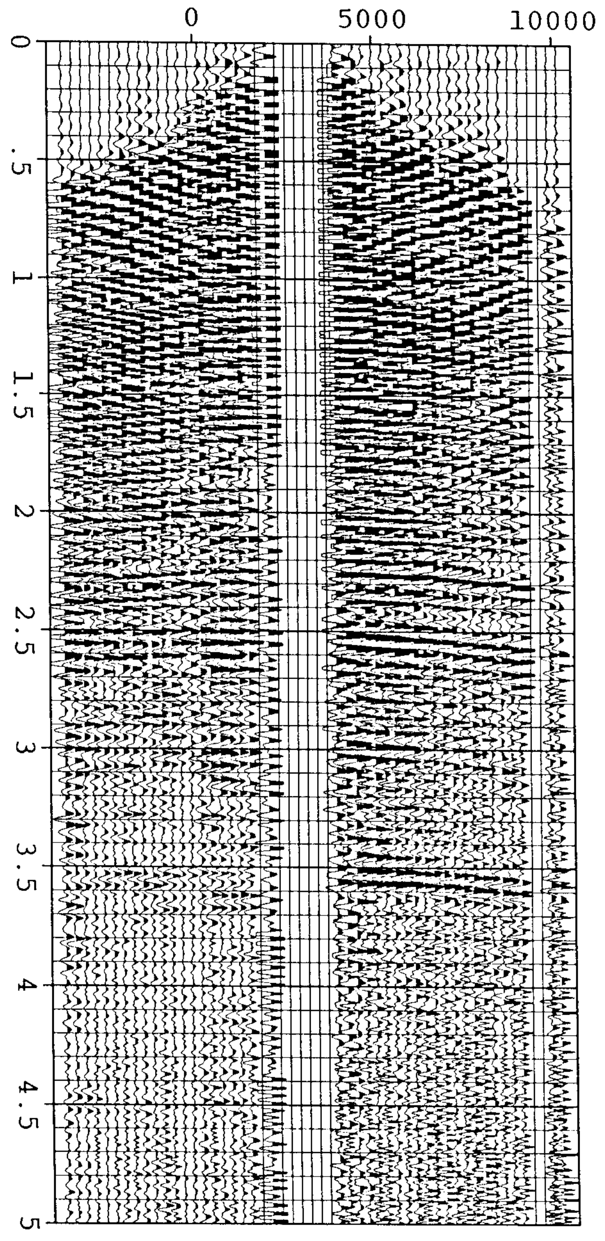


FIG. 2. Prediction filters designed to deconvolve the field gather of Fig. 1. The LSQR filter is an iterative solution to the overdetermined least squares system of equation (10). The SYMMLQ filter is an iterative solution to the corresponding normal equations (9).



(a)



(b)

FIG. 3. Field gather of Fig. 1 deconvolved with (a) the LSQR prediction error filter and (b) with the SYMMLQ prediction error filter.

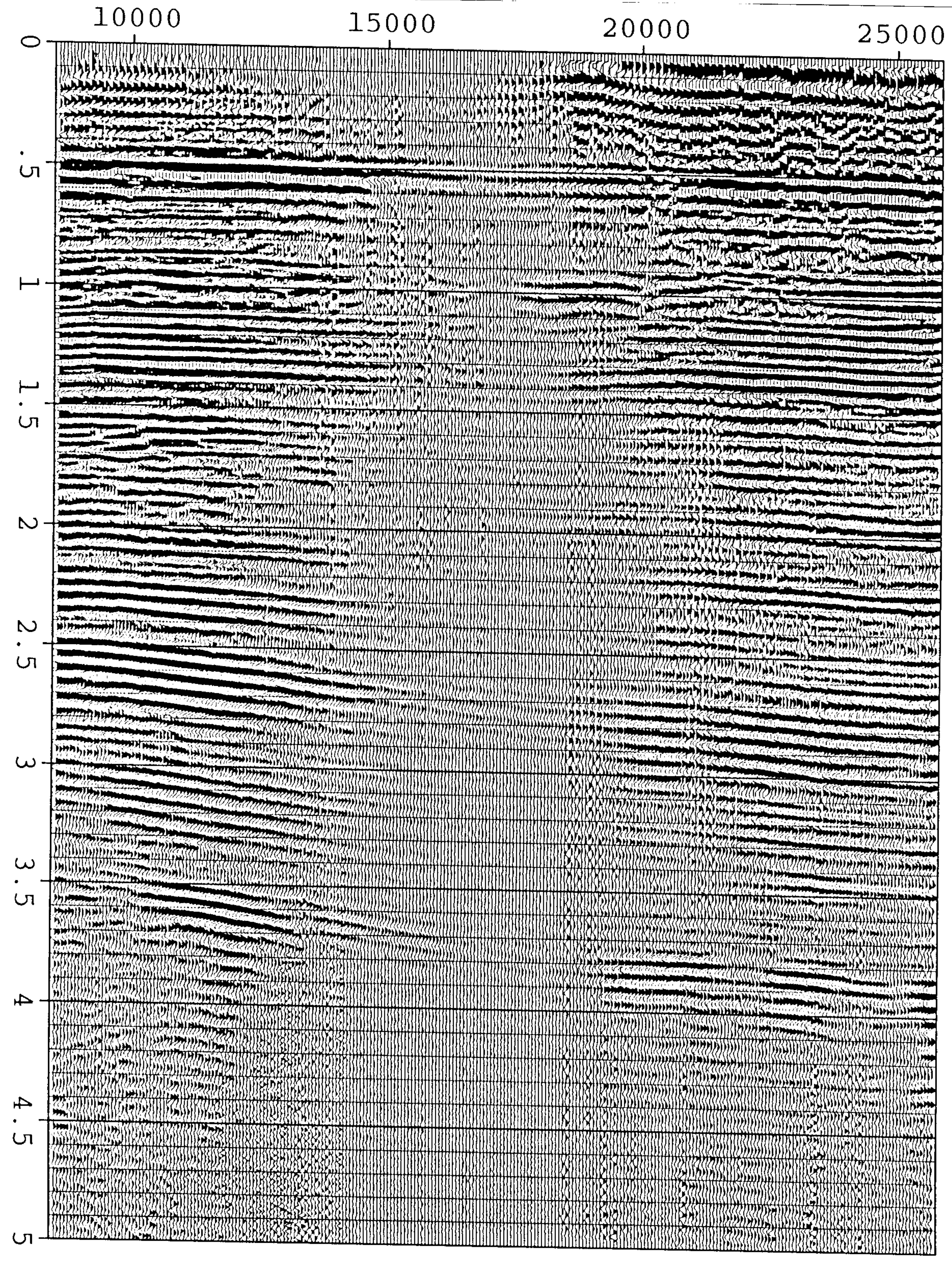


FIG. 4. Stack of Central Valley data. A t-squared gain function has been applied for playback. The amplitudes in the central portion of the line are attenuated by a (low velocity) gas seep in that area (Toldi, 1985).

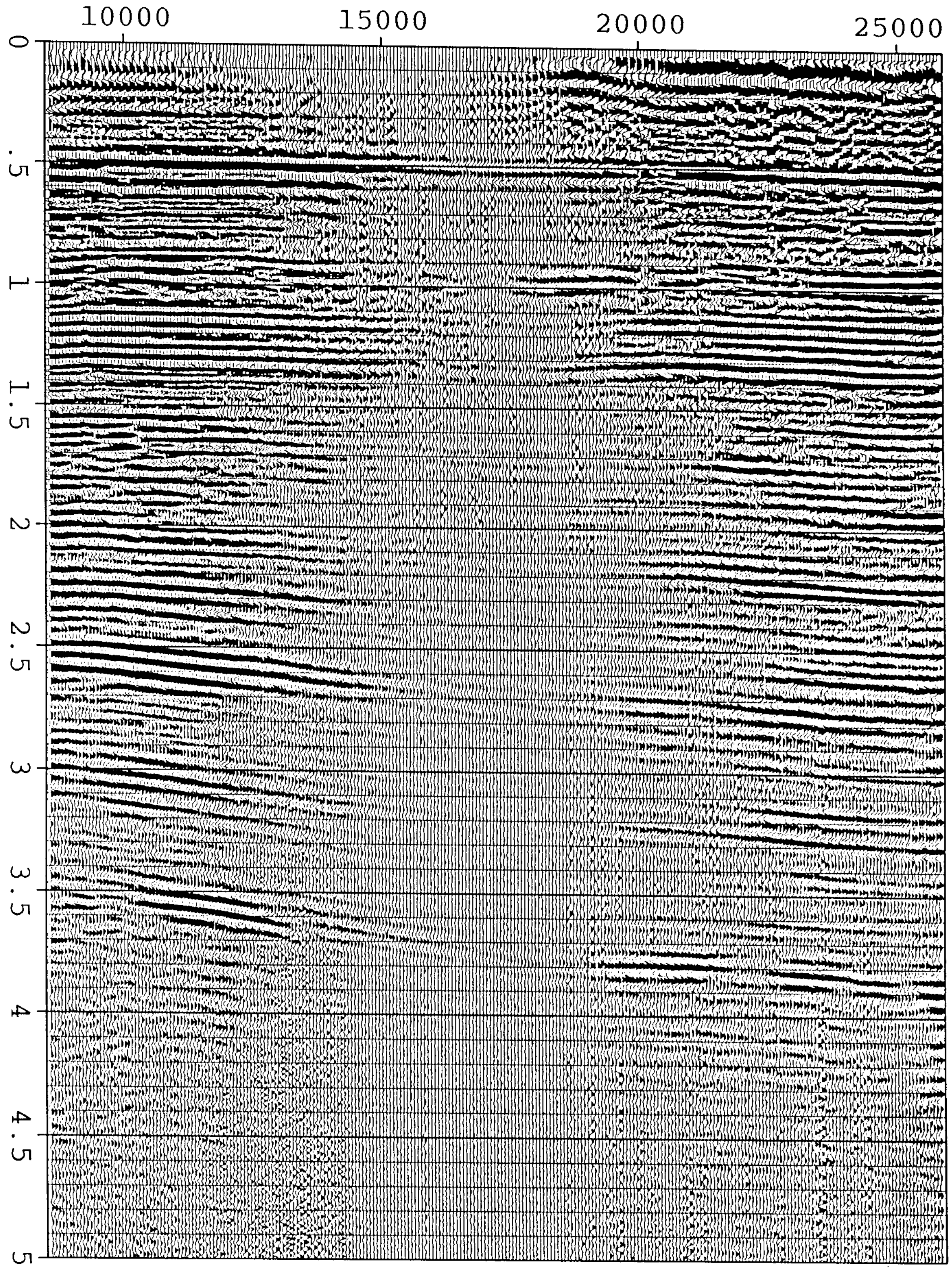


FIG. 5. Stack after shot-consistent prediction-error filtering. Filters were 132 msec with a 60 msec gap.

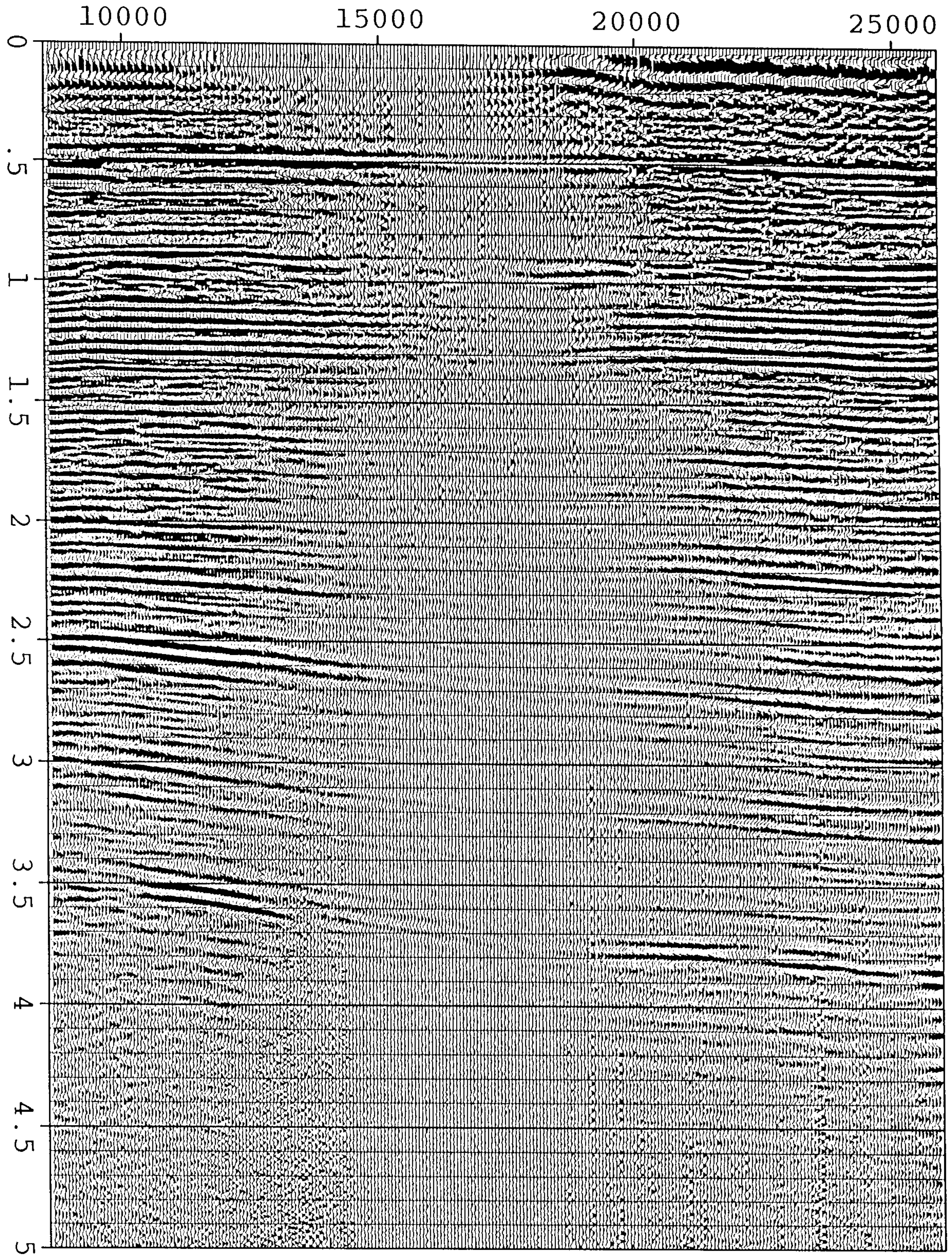
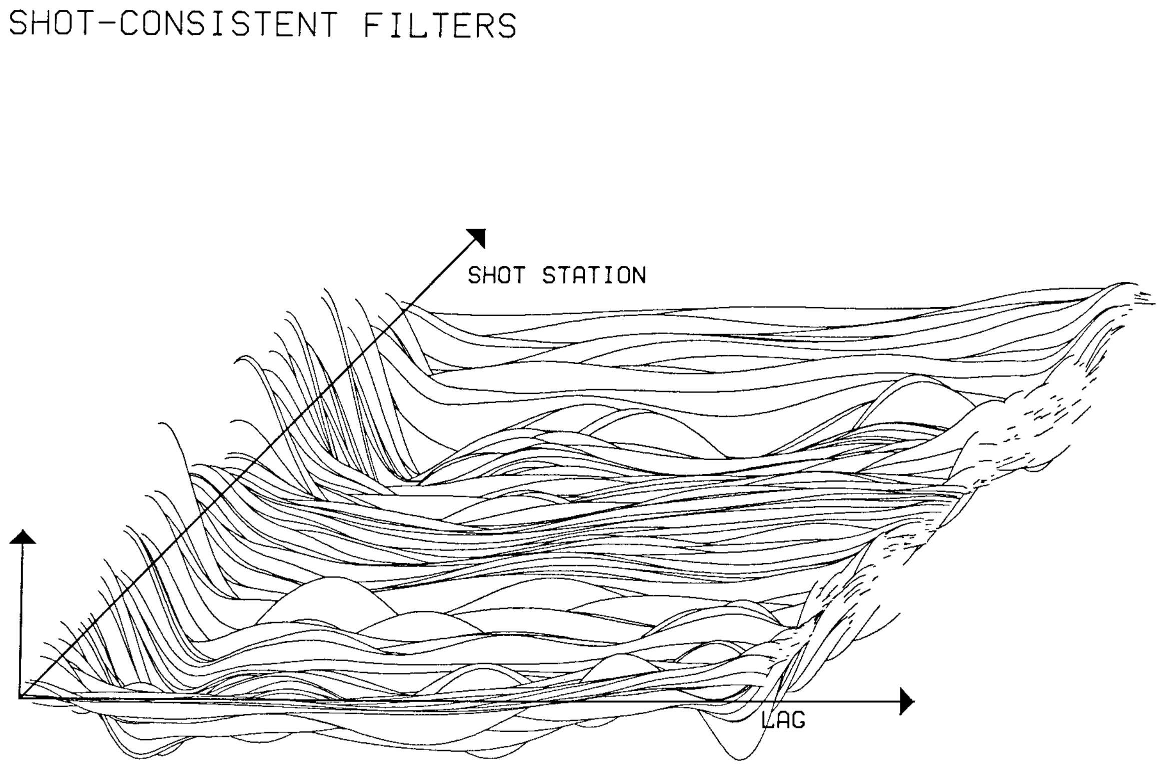


FIG. 6. Stack after cascading shot-consistent and geophone-consistent filtering. The geophone filters were 80 msec in length with a 44 msec gap.



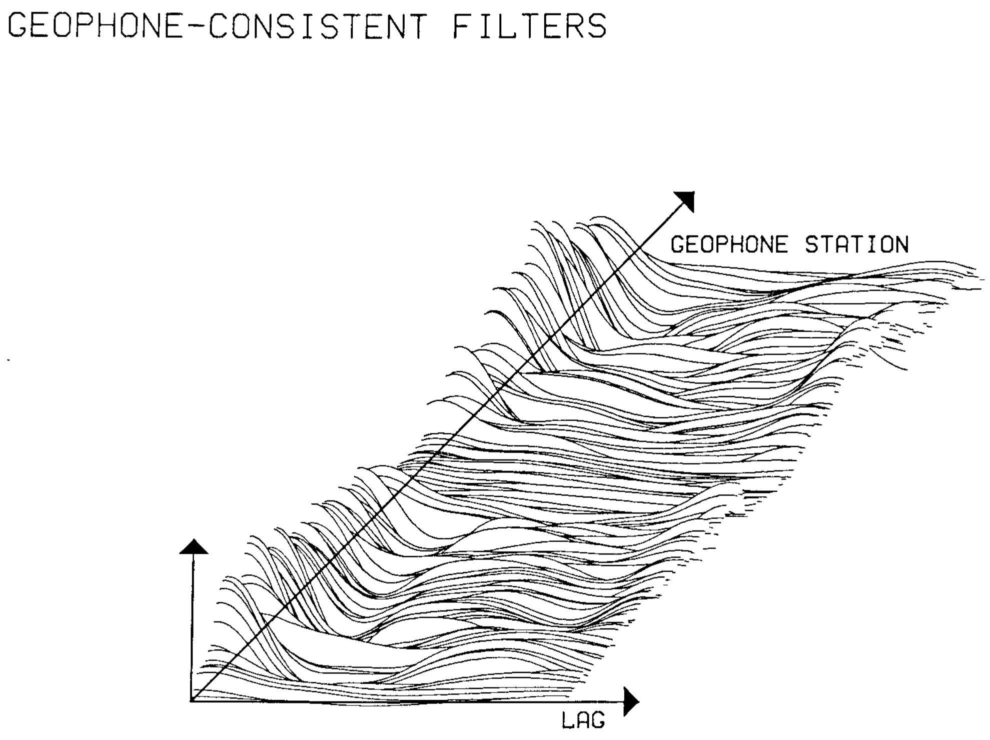


FIG. 7. Shot- and geophone-consistent filters estimated consecutively for the stack of Fig. 6.

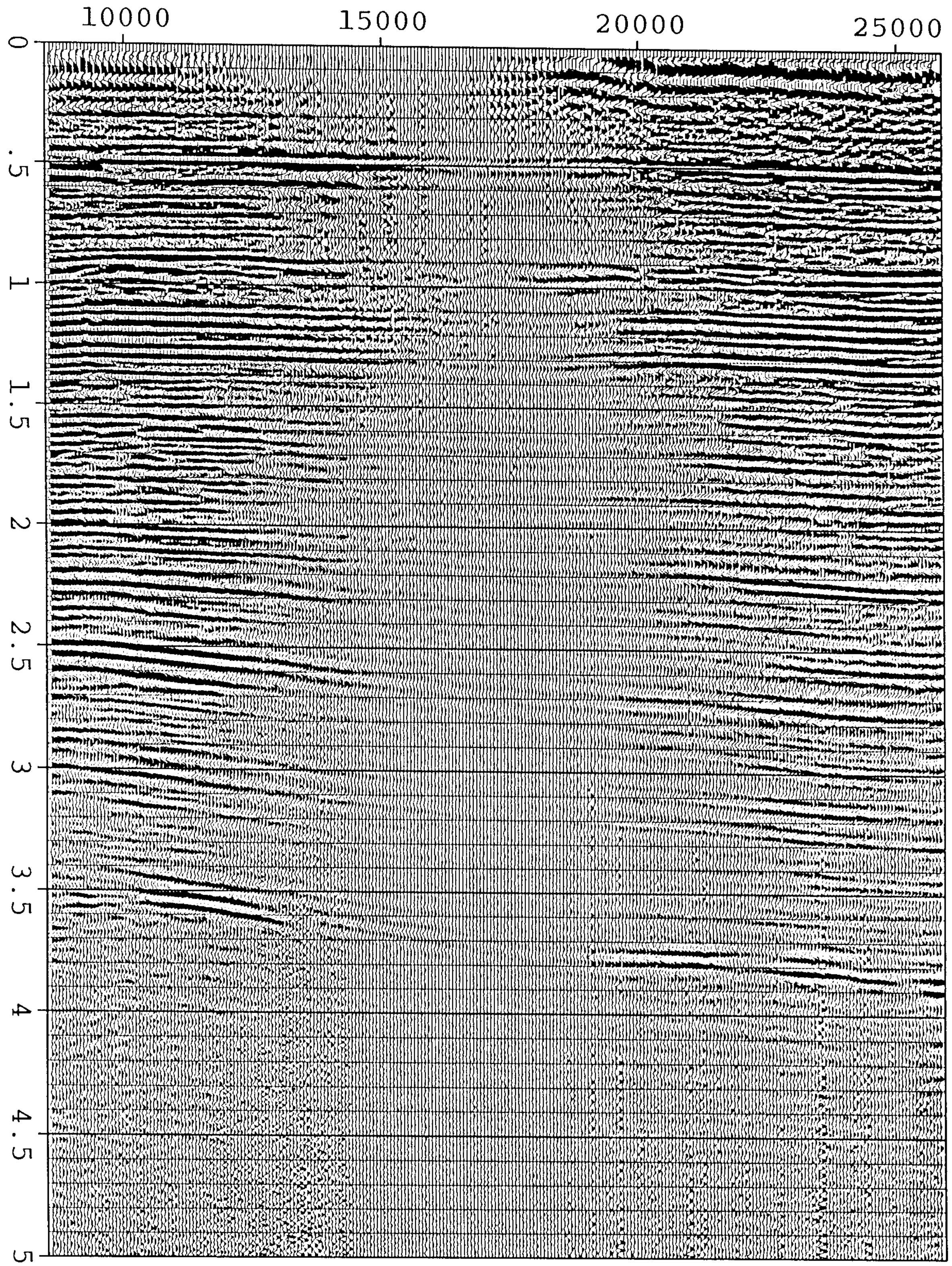


FIG. 8. Stack with shot- and geophone-consistent filters estimated simultaneously. This nonlinear least-squares solution was obtained by the Gauss-Newton method. This stack is very similar to that in Figure 6.

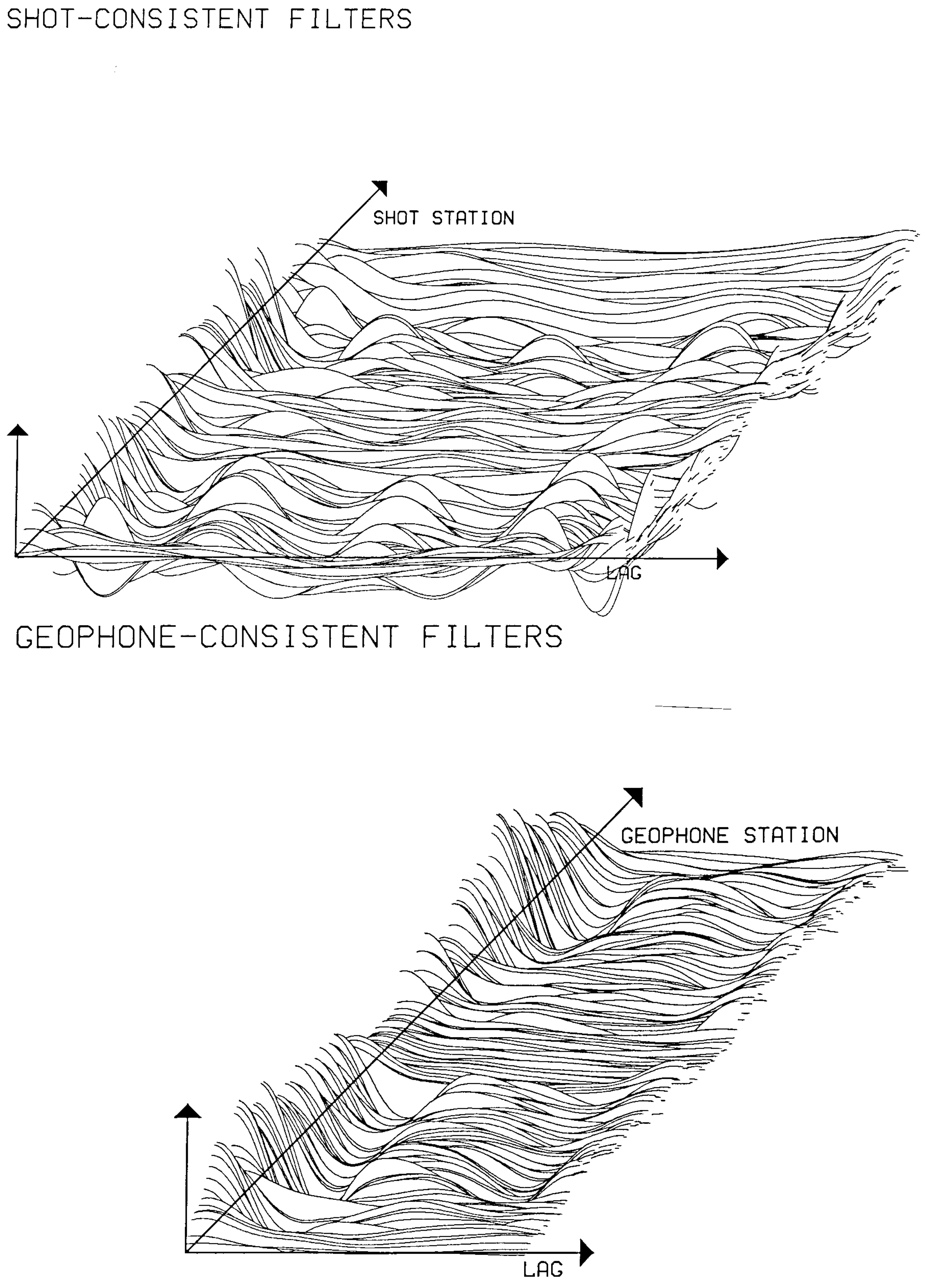


FIG. 9. Shot- and geophone-consistent filters estimated simultaneously for the stack of Fig. 8.

SUMMARY

Simultaneous design of surface-consistent prediction-error filters is eminently feasible even without using the second derivative acceleration of the Newton iteration. The data I've used to show this have been useful in debugging my procedures but did not profit from surface-consistent deconvolution. It's time to look for some good data examples.

REFERENCES

Claerbout, J.F., 1985, Simultaneous pre- and post-NMO deconvolution: SEP-42, 25-43.

Levin, S.A., 1984, Surface-consistent deconvolution: SEP-41, 1-26.

Levin, S.A., 1985, Newton trace balancing II: SEP-44, 145-159.

Paige, C.C. and Saunders, M.A., 1975, Solution of sparse indefinite systems of linear equations: SIAM J. Numer. Anal., 12, 617-629.

Toldi, J.L., 1985, Velocity analysis without picking: Stanford University Ph.D. thesis (SEP-44) 106 p.

Weird code, part 4

Even little programs can be weird. One of even tested this one!

4. The best "small" program:
(submitted by Jack Applin [with help from Robert Heckendorn]
<hplabs!hp-dcd!jack>)

```
main(v,c)char**c;{for(v[c++]="Hello, world!\n)";(!!c)[*c]&&(v--||--c&&execlp(*c,
*c,c[!!c]+!!c,!c));**c=!c)write(!!*c,*c,!!**c);}
```