Stanford Exploration Project, Report 73, November 18, 1997, pages 425–434

Filling data gaps using a local plane-wave model

Jon F. Claerbout[1]

keywords: *interpolation, deconvolution, modeling, two-dimensional*

ABSTRACT

I assemble a utility for trace interpolation assuming a local monoplanewave model. In local windows, many one-point filters are tested with various dips. These are trivial spatial prediction filters. These simple filters are easily interpolated leading to data interpolation. PVI introduces the idea of 2-D prediction-error filters to data interpolation. The method here reduces that concept to practice in an elementary way. This method is the first step of a family of inverse methods that I am developing.

INTRODUCTION

This article has practical goals, educational goals, and research goals. The practical goal is to illustrate a subroutine that interpolates missing seismograms by the most rudimentary, and generally effective method. The basic assumption is that the data can be locally (in space and time) fit to a plane wave which can be used for linear interpolation. It further assumes that the filling in of a gap between two seismograms can be based on only the two seismograms and that no additional seismograms are needed. Effectively, this assumes good quality data. Thus, it should be particularly effective for crossline marine surveys where the line separation is uncomfortably large and irregular, but the data quality is high because of in-line stacking.

The educational goal is to introduce what amounts to Burg handling of edge effects. Since dips change rapidly in time and space, and since we will work in small windows where we presume constant dip, edge effects can be overwhelming and great care is needed.

The research goal is to begin bridging the gap between conventional processing and general inversion. Since PVI (Claerbout, 1992) I am associating 2-D PEFs with (factorizations of) inverse covariance matrices that need to be estimated simultaneously with the missing data (or model parameters). This makes the inversion problem fundamentally nonlinear and dependent on a reasonable initial method. Ultimately, I plan to show that the interpolation method expounded here (which approximates

[1]**email:** not available

conventional processing) fits the framework of a general inverse problem. Thus this method, limited though it is, is a natural basis for iterative improvement.

RESULTS AND METHOD

The interpolation procedure is invoked with subroutine `gapfill()` on the current page. First it scans the seismogram array for seismograms that are entirely zero. These are taken as missing data and the rest of the data as known. Then it passes over the seismograms sending groups to subroutine `gapatch()` on this page. Each group is characterized by a known seismogram on the left and on the right, and a collection of unknown seismograms in between.

```
# Fill in missing traces by interpolating across gaps on best slopes.
#
subroutine gapfill( data, n1,n2, known    , maxslope)
integer                   n1,n2, known(n2), maxslope,   i1,i2, left, rite, w2
real                data( n1,n2),                       big
do i2= 1, n2 {
        big = 0.
        do i1= 1, n1
                if( big < abs( data(i1,i2)) )
                    big = abs( data(i1,i2))
        if( big > 0.) { known(i2) = 1 }
        else          { known(i2) = 0 }
        }                                       # Even if zero,
  left = 1                                      # take first trace known.
do rite= 2, n2 {
        if( known( rite) > 0 | rite == n2) {    # Take last trace known.
                w2 = rite - left + 1
                call gapatch( data(1,left), n1, w2, maxslope)
                left = rite
                }
        }
return; end
```

Testing `gapfill()` on the current page on synthetic data gave the result in Figure 1. This figure illustrates that the method generally works, failing mainly where you expect it to, along the unconformity and at the buried focus where there is wave triplication. The input data in this example appears to be irregular. Actually, it is on a mesh that is more densely sampled than normal leaving many more gaps than data locations. In practice, where there is more data than gaps, an alternate method that assumes imperfect data might be preferable. I should test it on some field data.

```
# make patches on 1-axis, each as wide as a gap, length depending on width.
#
subroutine gapatch( data, n1, n2, maxslope )
integer                   n1, n2, maxslope
```

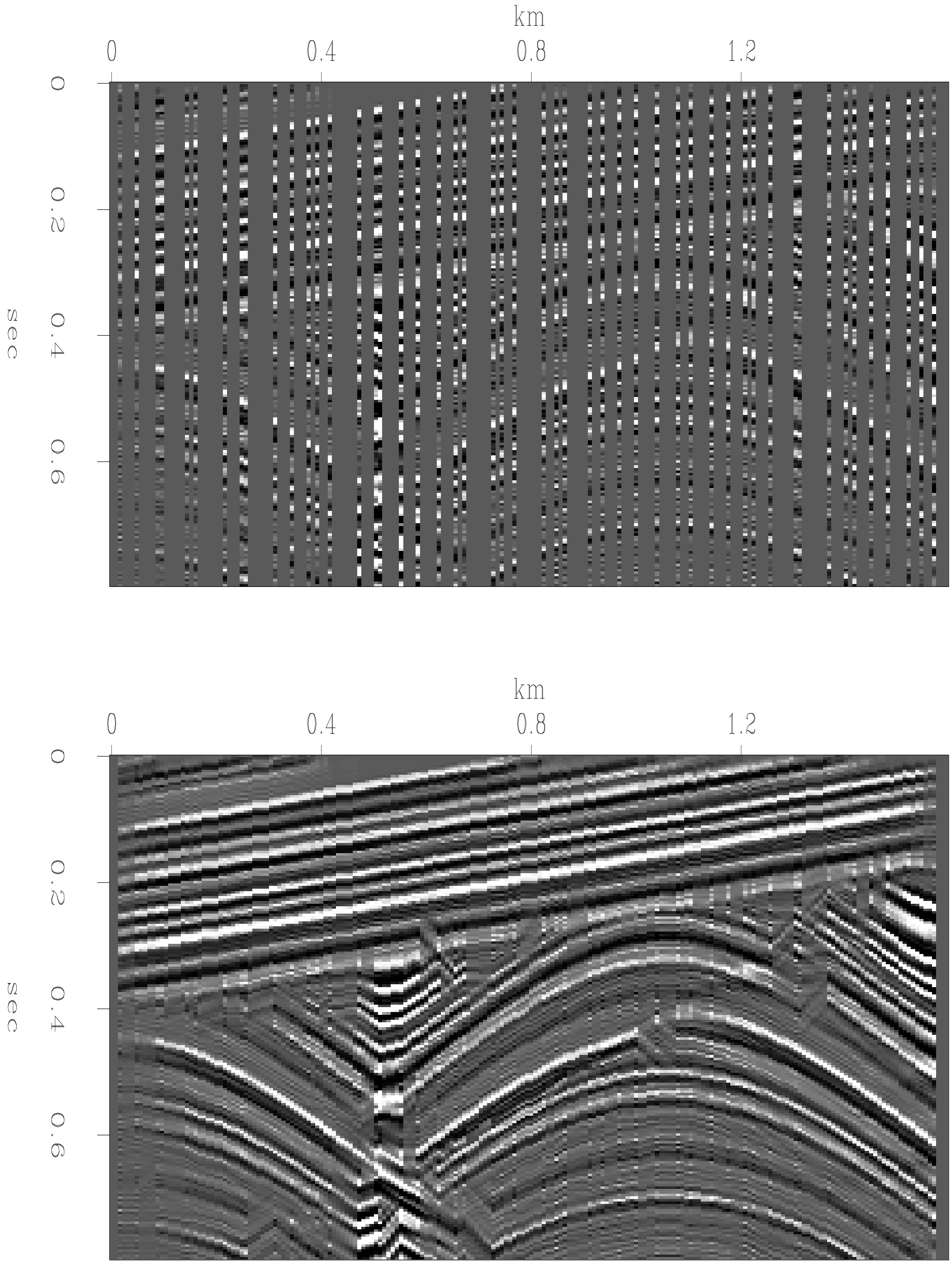

Figure 1: Top is synthetic data. Bottom is linear interpolation over gaps. jon5.gapfill-gapfill [ER]

```
real                data( n1, n2)
integer i1,i2, w1,w2,j1,j2,k1,k2, maxlag, w0
parameter ( w0 = 20)
temporary real      wind  ( w0+maxslope*(n2-1), n2), dcopy ( n1, n2)
temporary real      windwt( w0+maxslope*(n2-1), n2), wallwt( n1, n2)
maxlag =  maxslope * (n2-1)
w1 = w0 + maxlag        ;  k1 = 1 + (1.25 * n1) / w0
w2 = n2                 ;  k2 = 1
call zero( n1*n2, wallwt)
call zero( n1*n2, dcopy )
    j2= 1
do  j1= 1, k1 {
        call patch( 0, 0, j1,j2, k1,k2,   data, n1,n2,   wind, w1,w2)
        call gapinside( wind,  w1,w2, windwt, maxslope)
        call patch( 1, 1, j1,j2, k1,k2,  dcopy, n1,n2,   wind, w1,w2)
        call patch( 1, 1, j1,j2, k1,k2, wallwt, n1,n2, windwt, w1,w2)
        }
do i2= 1, n2
do i1= 1, n1
        if(                                   wallwt(i1,i2) != 0.)
                data( i1,i2)  =  dcopy( i1,i2) / wallwt(i1,i2)
return; end
```

Subroutine `gapatch()` on page 426 takes two seismograms surrounding a collection of zero seismograms—call this a "tower" of data—and slices the time axis to make "patches" of data. The patches are extracted and put back with subroutine `patch()` on page 416 described in an accompanying article (page **??**). These patches are passed along to subroutine `gapinside()` on the facing page for handling. It adds data onto the emptiness in the center of the tower and it also accumulates a weighting function for each patch from various triangles that taper data towards the top and bottom of the patch and taper the data as it is projected into the tower from either side. The weighting function for each patch typically differs from other patches (because of the various slopes encountered in each patch). The patches overlap, and after each one is processed, they must be reassembled seamlessly. This is done by accumulating a weighting function of the whole tower of data at the same time the patches are accumulated. The final step is to divide the accumulation by the tower weight (where it is nonzero).

Let us return to the parameter `w0` at the beginning of `gapatch()` on page 426. Increasing this parameter makes patches taller, thereby increasing the size of the disturbed zone near the unconformity. Decreasing this parameter increases the number of false picks, particularly near the ends of the dip range. Since the mispicks (not shown) are not uniformly distributed, but seem to scatter near the extremes of the dip range, it seems more theoretical work should go into the analysis of `burone()` on the facing page. The parameter `w0` relates to the patch time duration in the following way: At the largest dip there will be the fewest number of time points correlated across the gap. This number is `w0`. I was uncertain whether to base the number `k1` of patches on `w0` or on `w1`, the patch height. One seems an over estimate, the other an under estimate. I chose the cautious estimate but then cut the overlap factor to

25%. The actual overlap will be larger where the dips are modest.

Before we look at the messy internals of what happens in each patch, let us examine the lowest level subroutine `burone()` on the current page.

```
# residual(a) =  SELFDOT( x(g)-a*y(1) ) +  SELFDOT( y(1)-a*x(g) )
#
subroutine burone( x,   y,n, g, a )
integer                  n, g,          m
real               x(n),y(n),   a,      dot, aa,ra
m = n - g + 1
aa =   dot(m, y(1),y(1)) + dot(m, x(g),x(g)) + 1.e-20
ra =   dot(m, y(1),x(g)) + dot(m, x(g),y(1))
a  =   ra / aa
return; end
```

This routine is called with two traces, one from each side of the patch. It tests the correlation of the two traces at a specified relative time shift. The correlation is defined by the simultaneous prediction of the left trace from the right and the right trace from the left. If you know the Burg spectral estimation procedure (see FGDP) this is a familiar idea. Requiring the leftward predictor to equal the rightward predictor contributes to imposing the monoplanewave model upon the data. In FGDP p. 135-6 it is proven that the correlation coefficient computed this way has a magnitude that is always less than unity. Notice that `burone()` on this page presumes that the left seismogram `x(n)` carries its information *later* than the right seismogram `y(n)`. Since in practice either one could be later, it is necessary to call `burone()` on the current page twice, the second time with seismograms interchanged.

```
# Figure which direction works the best, then interpolate along it.
#
subroutine gapinside( data, n1,n2,  wiwt, maxslope )
integer                     n1,n2,        maxslope,  i1,i2
real                  data( n1,n2), wiwt(n1,n2),     aa, bb, coh, w, tri
integer                                          m, j1, lag,lagl,lagr, mid,best
temporary real  coherency( 1+2*maxslope*(n2-1))
call zero(                  n1*n2,  wiwt )
mid  = 1 +  maxslope * (n2-1)
do lag = 1, mid {
        call burone( data,      data(1,n2),n1, lag, coherency( mid+lag-1) )
        call burone( data(1,n2),data,      n1, lag, coherency( mid-lag+1) )
        }
best = mid
do lag = 1, 2*mid-1
        if(  coherency( best) < coherency(lag) )
                        best  =          lag
coh = amax1( coherency( best), 0.);     lag =  1 + iabs( mid - best)
do i2= 2, n2-1 {
        aa = (n2-i2) / float( n2-1);    lagr =  1.5 + (lag-1) * aa
        bb = (i2- 1) / float( n2-1);    lagl =  1.5 + (lag-1) * bb
        if( best < mid ) {
               m= n1-lagl+1;  do i1= 1,m {  j1= i1+lagl-1;       w= tri(i1,m)
```

```
                        data(j1,i2)= data(j1,i2) + data(i1,  1) * w * aa * coh
                        wiwt(j1,i2)= wiwt(j1,i2) +                w * aa
                        }
                m= n1-lagr+1;  do i1= 1,m {  j1= i1+lagr-1;       w= tri(i1,m)
                        data(i1,i2)= data(i1,i2) + data(j1, n2) * w * bb * coh
                        wiwt(i1,i2)= wiwt(i1,i2) +                w * bb
                        }
                }
        else {  m= n1-lagl+1;  do i1= 1,m {  j1= i1+lagl-1;       w= tri(i1,m)
                        data(i1,i2)= data(i1,i2) + data(j1,  1) * w * bb * coh
                        wiwt(i1,i2)= wiwt(i1,i2) +                w * bb
                        }
                m= n1-lagr+1;  do i1= 1,m {  j1= i1+lagr-1;       w= tri(i1,m)
                        data(j1,i2)= data(j1,i2) + data(i1, n2) * w * aa * coh
                        wiwt(j1,i2)= wiwt(j1,i2) +                w * aa
                        }
                }
        }
return; end
```

Subroutine `gapinside()` on the preceding page first calls `burone()` on the page before twice, once for each allowable lag. Then it finds the lag with the best coherence and uses that to set the dip. We can imagine the coherence between the two observed traces decaying steadily from one towards the other. Instead of attempting to allow for such a coherence model, I simply scaled all signals in the gap by the coherence measured across the gap. Finally is a sequence of four loops that I find very confusing. Two are needed because the lag may be positive or negative. Two are needed because information is projected from each seismogram toward the other. This projection is under a linear taper and there is an additional triangle taper on the time axis applied by the function `tri()` on this page.

```
# Triangle function (isoceles)
#
real function tri( i, n)
integer            i, n
real mid, wide, x
mid  = (n+1)/2.;
wide =   n /2.
x    =   abs( (i-mid) / wide )
tri  =  1. - abs( x)
return; end
```

IMPROVEMENTS

The fitting need not be done in discrete patches. The coherence estimated in subroutine `burone()` on the page before could be estimated in a continuously variable way by smoothing the quadratic found in its numerator and separately smoothing the denominator before forming the ratio. I do not see any reason why this should not be practical but I do not know how much quality difference would result from continuous gradation. It seems a worthwhile study.

Another view is that the innards of `burone()` on page 429 are nearly the conjugate of the four confusing loops, but the conjugacy is not explicitly developed here. In fact, the present treatment of weighting functions is *not* conjugate.

The approach in this paper assumes a single plane wave at every location. Although this assumption will surely be wrong in some places (where there will be none, or two, or more) it may be the most sensible launching pad for nonlinear iteration. An inversion that launches itself from zero-valued missing data, for example, might be doomed by the nonlinearity and distance from the final solution. It might be worthwhile to formally extend the model in that way in the direction of chapter 8 in PVI (1992) which handles several simultaneous dips.

RESEARCH GOALS

My ultimate goal includes all forms of parameter estimation. My immediate goal is data interpolation. Interpolation is a trivial consequence of a full inversion. I believe, however, that interpolation is an important stepping stone toward a full inversion.

In the code in this paper I have reformulated conventional practice to look like general inversion theory. To introduce the notion of covariance matrix into conventional practice, I replaced the notion of "best crosscorrelation" with the idea of "2-D prediction-error filter". The problem I have found with PEFs is they immediately suggest using a large number of undetermined coefficients, many more than required to parameterize the simple model we have in mind. These coefficients arise from the need to cover a large range of possible stepouts. Using them all allows for frequency dependent coherency between adjoining traces, and that is not the model we need here. Our first need is to reduce the number of coefficients. That is why I put a "1" on the trace on one side of the gap and a single adjustable coefficient on the other. Since many stepouts are possible, the question arises, "which lag will have the free parameter?" That is why I did a global search of all possible lags within some range to see which gives the best fit. You might recognize some similarity in what I am proposing here to the more conventional approach of finding the peak of the crosscorrelation. Although my real stratagy is to structure the method to match the conventional approach, you may notice that differences could arise when the two traces have differing autocorrelations. As with one-dimensional PEF theory (see FGDP) I want my 2-D PEF's to predict both ways. In other words, if you reverse both time and space axes, the stepout (or dip) remains unchanged and the same PEF applies.

In my beginning efforts I used a three-point filter, a "1" on one trace and two adjacent coefficients on the other. As I was finishing the coding but before I began debugging it, I judged the extra refinement to be unwarranted since this refined interpolation in data space will be overruled in the steps that I plan where I will invoke the monoplane principle in image space.

3-D EXTENSION PROPOSAL

Consider three seismograms on the earth surface that are not on a line. Let the seismograms be denoted as A, B, and C. Let seismogram A be located at $r_a = (x_a, y_a)$ with a likewise location for B and C. Take a monoplane model, const $= t - p_x x - p_y y$. Define three times which are linear functions of p_x and p_y.

$$\begin{aligned} \tau_a &= r_a \cdot p &= \tau_a(p_x, p_y) \qquad (1) \\ \tau_b &= r_b \cdot p &= \tau_b(p_x, p_y) \qquad (2) \\ \tau_c &= r_c \cdot p &= \tau_c(p_x, p_y) \qquad (3) \end{aligned}$$

Let Z be the familiar delay operator of Z-transform theory. Now, do a global scan over the two parameters (p_x, p_y) where for each value of (p_x, p_y) you solve the following minimization of the quadratic in three scalar variables.

$$\begin{aligned} Q(\alpha, \beta, \gamma) = \quad & \|A - \alpha Z^{(\tau_a - \tau_b)} B\| + \|B - \alpha Z^{(\tau_b - \tau_a)} A\| + \\ & + \|A - \beta Z^{(\tau_a - \tau_c)} C\| + \|C - \beta Z^{(\tau_c - \tau_a)} A\| + \\ & + \|C - \gamma Z^{(\tau_c - \tau_b)} B\| + \|B - \gamma Z^{(\tau_b - \tau_c)} C\| \qquad (4) \end{aligned}$$

where summation over a small time gate is denoted by $\|Q\|$. Absorbing the time shifts into the definitions of A, B, and C and minimizing Q by variation of α leads to $\alpha = 2(A \cdot B)/[(A \cdot A) + (B \cdot B)]$ with similar expressions for β and γ.

The value of (p_x, p_y) is found by the global search of $Q(p_x, p_y)$. A two-dimensional search on (p_x, p_y) space is not as costly as it sounds. We can first run `burone()` on page 429 for each of the three rows in equation (4) for all possible one-dimensional stepouts and tabulate the outputs. Then the two-dimensional global search requires only a few additions from the tables for each point in (p_x, p_y) space.

If we were to follow the philosophy of my 2-D code, the value of the interpolated trace is far less important than the fact that we have used the (p_x, p_y) from above. In other words, we can throw away (α, β, γ). The value of the unknown trace could be

$$U Z^{-\tau_u} \quad = \quad c(w_1 A Z^{-\tau_a} + w_2 B Z^{-\tau_b} + w_3 C Z^{-\tau_c}) \qquad (5)$$

where the w_i depend only on geometry and the scale, if any, is the overall coherence, say $c = 1 - \|Q\|/(\|A\| + \|B\| + \|C\|)$. Alternately, it is not difficult to make more elaborate models. [2]

There are many ways to bake the cake. I believe, however, the most important feature of any formulation is that it should maintain a very minimal number of adjustable parameters because we want to go to very small time gates. If the original data were on a regular rectangular mesh, instead of an arbitrary three locations, then

[2] Steve Cole has begun testing some of these 3-D ideas. He suggests (and I agree) that the code given here should not give up at the side boundaries. Sideward extension should taper off in strength consistent with the coherency between known traces.

we can reduce the three free parameters above to two. First, permit me to change variables so all the values of Z^τ are unity. Then use α for north-south correlation and β for east-west correlation.

$$\begin{aligned} Q(\alpha,\beta) \quad = \quad & \|A-\alpha B\| + \|B-\alpha A\| + \|C-\alpha D\| + \|D-\alpha C\| + \\ & + \|A-\beta C\| + \|C-\beta A\| + \|B-\beta D\| + \|D-\beta B\| \end{aligned} \tag{6}$$

REFERENCES

Claerbout, J. F., 1992, Earth Soundings Analysis: Processing Versus Inversion: Blackwell Scientific Publications.