Chapter 6

Model fitting by least squares

Two sections omitted here may be found in SEP-61 p 375. They are "Multivariate least squares" and "Conjugate-gradient descent". We will build here on the subroutine `cgstep()` defined there on page 384. (The order of the arguments was revised. It now is `cgstep( iter, n, x, g, s, m, res, gg, ss)`).

MODELING BY RESTORATION OF MISSING DATA

We take *data space* to be a uniform mesh on which some values are given and some are missing. We rarely have missing values on a time axis, but commonly have missing values on a space axis, i.e. missing seismograms. Missing seismograms (traces) happen occasionally for miscellaneous reasons and they happen systematically from aliasing and truncation. The aliasing arises from economic reasons—saving instrumentation by running receivers far apart. Truncation arises at the ends of any survey, which like any human activity, must be finite. Beyond the survey lies more hypothetical data. The traces we will find for the missing data are not as good as real observations, but they are closer to reality than supposing unmeasured data is zero-valued. Making an image with a single application of a conjugate operator amounts to assuming data vanishes beyond its given locations. Dealing with missing data is a step beyond this and, as we'll see, reduces the need for arbitrary model filtering.

The first principle we use for restoring missing data is that the restored data, after specified filtering, have minimum energy. Specifying the filter chooses the interpolation philosophy. Generally the filter is a *roughening* filter. When a roughening filter goes off the end of smooth data it typically produces a big edge transient. Minimizing energy implies a choice for unknown data values at the edge to minimize the transient. We'll examine five cases and then make some generalizations.

Let m denote a missing value. The data set on which the examples are based is $(\cdots, m, m, 1, m, 2, 1, 2, m, m, \cdots)$. Using a program described later, values were found to replace the missing m values so that the power in the filtered data is minimized.

Figure 1 shows interpolation of the data set with $1 - Z$ as a roughening filter. The interpolated data matches the given data where it is given.

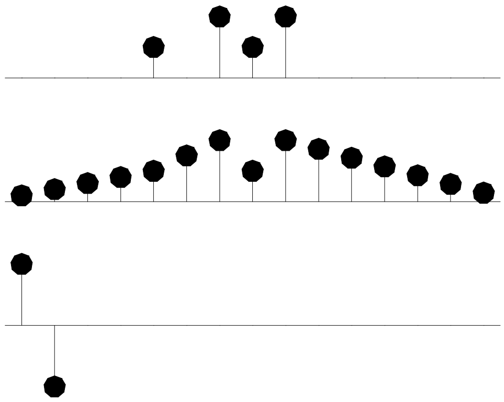

Figure 1: Top is given data. Middle is given data with interpolated values. Missing values seem to be interpolated by straight lines. Bottom shows the filter $(1, -1)$. jon2-mlines [NR]

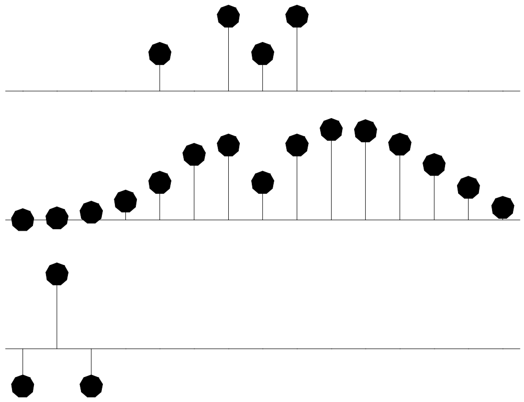

Figure 2: Top is the same input data as in Figure 1. Middle is interpolated. Bottom shows the filter $(-1, 2, -1)$. Missing data seems to be interpolated by parabolas. jon2-mparab [NR]

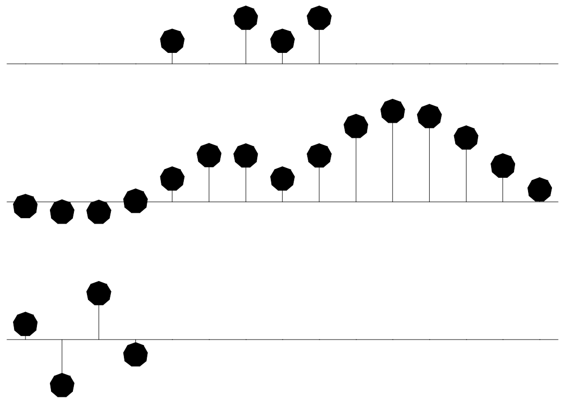

Figure 3: Top is the same input. Middle is interpolated. Bottom shows the filter $(1, -3, 3, -1)$. Missing data is very smooth. It shoots upward high off the right edge of the observations apparently to match data slope there. jon2-mseis [NR]

Figures 1-5 illustrate the rougher the filter the smoother the interpolated data, and vice versa. It is as if the product of the filter spectrum with the data spectrum is roughly constant. Switch attention from the residual spectrum to the residual itself. The residual for Figure 1 is the *slope* of the signal (because the filter $1 - Z$ is a *first derivative*) and the slope is constant (uniformly distributed) along the straight lines where the least-squares procedure is choosing signal values. So these examples confirm

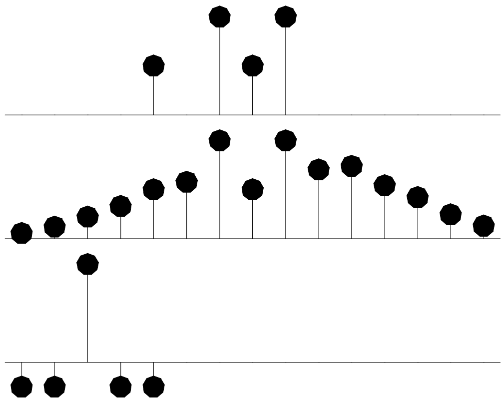

Figure 4: Top is the same. Middle is interpolated. Bottom shows the filter $(-1, -1, 4, -1, -1)$. Interpolation is with stiff lines like the straight lines of Figure 1 but they project through a cluster of given values instead of projecting to the nearest given value. Thus this interpolation tolerates noise in the given data better than Figure 3. $\boxed{\text{jon2-msmo}}$ [NR]

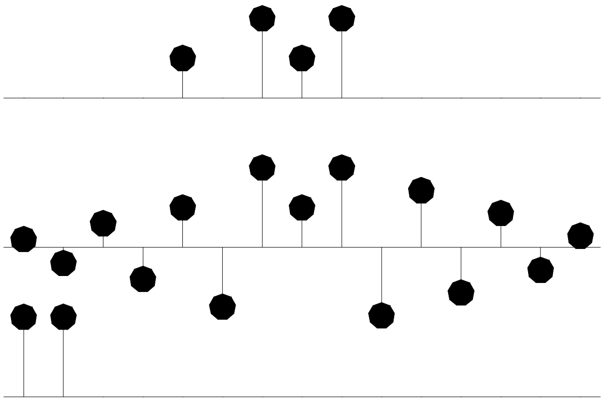

Figure 5: Top is the same. Middle is interpolated. Bottom shows the filter $(1, 1)$. The interpolation is rough. Like the given data itself it has much energy at the Nyquist frequency. But unlike the given data it has little zero frequency energy. $\boxed{\text{jon2-moscil}}$ [NR]

the idea that the least-squares method abhors large values (because they are squared) and thus it tends to uniformly distribute residuals in both time and frequency to the extent the constraints allow. This idea helps us answer the question, “What is the best filter to use?” Formally the answer is that the filter should have an amplitude spectrum that is inverse to that we wish for the interpolated data. A systematic approach is developed in chapter 8 but I’ll offer a simple subjective analysis here. Looking at the data I see all points are positive so it seems the data is rich in low frequencies so the filter should contain something like $(1 - Z)$ which vanishes at zero frequency. Likewise the data seems to contain Nyquist frequency so the filter should contain $(1 + Z)$. The result of the filter $(1 - Z)(1 + Z) = 1 - Z^2$ is in Figure 6. I regard this as the “best” interpolation based on the idea that the missing data should look like the given data.

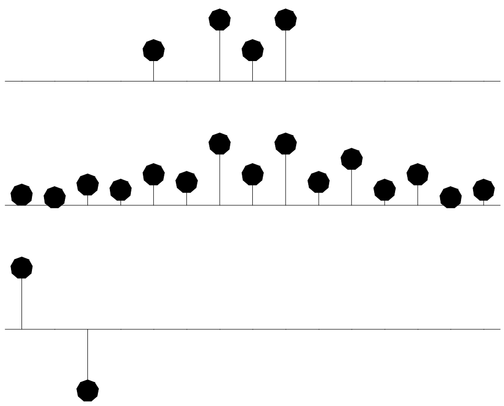

Figure 6: Top is the same. Middle is interpolated. Bottom shows the filter $(1, 0, -1)$ which comes from the coefficients of $(1 - Z)(1 + Z)$. Both the given data and the interpolated data have significant energy at both zero and Nyquist frequencies. jon2-mbest [NR]

Missing data program

This method of finding missing data values uses conjugate gradients and it introduces the notion of *simple constraints* to least squares fitting. The addition to the basic CG algorithm is uncomplicated, regardless of the amount of missing data or the complexity of its geometrical arrangement. You will see the program that made Figures 1-5 is pleasingly concise.

Formally, our problem is $\mathbf{0} \approx \mathbf{F}(\mathbf{k} + \mathbf{m}) = \mathbf{F}\mathbf{k} + \mathbf{F}\mathbf{m}$ where $\mathbf{F}$ is a filter (a shifted column matrix like in (**??**)), and $\mathbf{k}$ and $\mathbf{m}$ have elements in data space. Every point in data space is either given or missing so where a component in the vector of known givens, $\mathbf{k}$, is nonzero, that component in $\mathbf{m}$, must be zero, and vice versa. Recall that the program `cgmeth()` solves sets like $\mathbf{y} \approx \mathbf{A}\mathbf{x}$. So for that program $\mathbf{A}$ is $\mathbf{F}$, $\mathbf{x}$ is $\mathbf{m}$, and $\mathbf{y}$ is $-\mathbf{F}\mathbf{k}$. The trouble with that program is that it will change the known data values unless we recode the filter operator to exclude them. Instead we’ll reorganize the CG algorithm to include explicitly these constraints so then we can use any filter program from our library.

The vector $\mathbf{y}$ is the entire data space $\mathbf{k} + \mathbf{m}$. It is input to the missing-data program containing $\mathbf{k} + \mathbf{0}$. The program begins by loading the negative filtered $\mathbf{y}$ into a residual. The iterations proceed as with `cgmeth()` except that the matrix multiply routine is replaced by the convolution routine `contran()`. The new ingredient in the missing-data subroutine `miss1()` is the *simple constraints* that the given data cannot be changed. Thus after the gradient is computed, the components that correspond to given data values are set to zero.

```
# fill in missing data on 1-axis by minimizing power out of a given filter.
#
subroutine miss1( na, a, ny, y, copy, niter)
integer na,  iy,ny,  ir,nr,  iter,niter
real    y(ny)           # in: known data with zeros for missing values.
                        # out: known plus missing data.
real    copy(ny)        # in: copy(iy) vanishes where y(iy) is a missing value.
real    a(na)           # in: roughening filter
temporary real  dy(ny),sy(ny),  r(ny+na-1),dr(ny+na-1),sr(ny+na-1)
nr = ny+na-1
call contran( 0, na, a, ny, y, r)              # r = a*y   convolution
do ir= 1, nr
        r(ir) = - r(ir)                        # residual = - filtered data
do iter = 0, niter {                           # niter= number missing or less
        call contran( 1, na,a, ny,dy, r)       # dy(a,r)   correlation
        do iy= 1, ny
                if( copy(iy) != 0.)            # missing data where copy(iy)==0
                        dy(iy) = 0.            # can't change known data
        call contran( 0, na,a, ny,dy, dr)      # dr=a*dy   convolution
        call cgstep( iter, ny,y,dy,sy, nr,r,dr,sr)      # y=y+dy; r=r-dr
        }
return; end
```

That prevents changes to the given data by motion any distance along the gradient. Likewise motion along previous steps cannot perturb the given data values. So the CG method (finding the minimum power in the plane spanned by the gradient and the previous step) leads to minimum power while respecting the constraints.

Missing data with other operators

This book has many programs for conjugate-pair transformations. More transformations are described in pseudocode and more multidimensional transformations are found in my imaging book. Operators can be cascaded with one another yielding a greater multitude of conjugate-pair operators. From this multitude the missing data program shown used the convolution subroutine `contran()` but another operator from the multitude could have been chosen. The convolution operator was a good choice because convolution amounts to the family of ordinary differential equations—a big family of applications. What other operators could lead to useful results? This is food for thought.

Obviously the same missing data program could be used for two-dimensional data if we replaced the one-dimensional convolution subroutine by two-dimensional convolution. Simple filters could be interpreted as partial differential equations such as for plate bending or for a tent over poles. Large filters would be more costly, but they might be represented as a separable approximation. Theoretically the CG method could require many iterations—as many as there are missing values. This creates the potential for burdensome computational costs but convergence is often much faster than the theoretical limit. If you have a convergence speed problem you might get some guidance from the literature on solving elliptic equations.

What the missing data program does is provide missing values in data space by spreading out residuals in another space. Let us think about other transformations and spaces.

Missing data in tomography

Medical tomography avoids a problem that is unavoidable in earth-science tomography. In medicine it is not difficult to surround the target with senders and receivers. In earth science it is nearly impossible. It is well known that our reconstructions tend to be indeterminate along the dominant ray direction. Customarily the indeterminacy is resolved by minimizing power in a roughened image. The roughening filter should be inverse in spectrum to the desired image spectrum. Unfortunately that spectrum is unknown and arbitrary. Perhaps we can replace this arbitrary image smoothing by something more rational in the space of the missing data.

Recall the well-to-well tomography problem. Given a sender at depth z_s in one well, a receiver at depth z_g in the other well, and given travel times $t_k(z_s, z_g)$, the rays are predominantly horizontal. Theory says we need some rays around the vertical. Imagine the two vertical axes of the wells being supplemented by two horizontal axes, one connecting the tops of the wells and one connecting the bottoms, with missing data traveltimes $t_m(x_s, x_g)$. From any earth model, t_k and t_m are predicted. But what principles can give us t_m from t_k? Obviously something like we used in Figures 1-5. But data for the tomographic problem is two-dimensional: Let the source location be measured as the distance along the perimeter of a box, where the two sides of the box are the two wells. Likewise receivers may be placed along the perimeter. Analogous to the *midpoint* and *offset* axes of surface seismology (see my other book "Imaging the Earth's Interior"), we have midpoint and offset along the perimeter. Obviously there are discontinuities at the corners of the box, and everything is not as nice as in medical imaging with a circle where the source and receiver locations as measured by angles, angles measured from an origin at the center of the box. Anyway, the box gives us a plane in which to layout the data, not just the recorded data, but all the data that we think is required to represent the image.

A simple idea to fill in the missing data is to represent the two-dimensional data plane as an array of one-dimensional signals. Then the program that produced Figures 1-5 is directly usable. Of course there is the issue of whether the one-dimensional signals should be laid out along the receiver (perimeter) axis or the sender axis, or the midpoint or the offset axis. Also, if we hope to be able to extrapolate around the corners of the box we had better eliminate discontinuities in traveltime slope by doing mormal moveout—traveltimes should be divided by the distance of the corresponding ray across the box.

There are other possible ways to interpolate the data. Some are intrinsically two-dimensional and some may use the tomographic operator. Alas, all this effort cannot produce information where none was recorded. But it should yield an image that is not overwhelmed by anisotropy from the obvious heterogeneity of the data-collection geometry.

With the filling of data space will it still be necessary to smooth the model explicitly (by minimizing energy in a roughened model)? Mathematically the question is one of "completeness" of the data space. I believe there are analytic solutions well known in medical imaging that prove a circle of data is enough information to specify completely the image. So we can expect that little or no arbitrary image smoothing is required to resolve the indeterminacy—it should be essentially resolved by the assertion that statistics gathered from the known data are applicable to the missing data. I regret I have not carried this concept beyond developing the hypothetical processing scheme, but chapter 7 does show you many completed field-data examples in filter theory.

Extending the program to more general constraints

It is not difficult to include more general constraints in the above program. Imagine a constraint that the sum of the data points must be zero. This would be implemented by first seeing that the mean began at zero and then at each iteration removing the mean from the gradient. Then any amount of the gradient can added to the solution and the data would retain a zero mean. Other constraints could be handled likewise because the mean is merely an *unweighted* sum and the same idea applies to weighted sums. Thus multiple linear constraints can be handled, so long as there are not so many of them that they bog down the calculation.

"Useful" constraint equations

There is a theory for general constraints in quadratic form minimization. I haven't found the theory to be useful in any application I've run into so far, but it should be useful for writing erudite theoretical articles.

Constraint equations are an underdetermined set of equations, say $\mathbf{d} = \mathbf{G}\mathbf{x}$ (the number of components in $\mathbf{x}$ exceeds that in $\mathbf{d}$), that must be solved exactly while some other set is solved in the least-squares sense, say $\mathbf{y} \approx \mathbf{B}\mathbf{x}$. This is formalized as

$$\min_{\mathbf{x}} \{Q_C(\mathbf{x}) \;=\; \lim_{\epsilon\to 0} [(\mathbf{y}-\mathbf{B}\mathbf{x})'(\mathbf{y}-\mathbf{B}\mathbf{x}) + \frac{1}{\epsilon}(\mathbf{d}-\mathbf{G}\mathbf{x})'(\mathbf{d}-\mathbf{G}\mathbf{x})]\} \qquad (1)$$

In my first book, "Fundamentals of Geophysical Data Processing," I minimized Q_C by power series letting $\mathbf{x} = \mathbf{x}^{(0)} + \epsilon\,\mathbf{x}^{(1)}$ and hence $Q_C = Q^{(0)} + \epsilon Q^{(1)} + \cdots$. I minimized both $Q^{(0)}$ and $Q^{(1)}$ w.r.t. $x^{(0)}$ and $x^{(1)}$. With a page of algebra this leads to the system of equations

$$\begin{bmatrix} \mathbf{B}'\mathbf{B} & \mathbf{G}' \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{B}'\mathbf{y} \\ \mathbf{d} \end{bmatrix} \tag{2}$$

where $\mathbf{x}^{(1)}$ has been superceded by the variable $\lambda = \mathbf{G}\mathbf{x}^{(1)}$ which has fewer components and where $\mathbf{x}^{(0)}$ has simply been replaced by $\mathbf{x}$. The second of the two equations by itself shows that the constraints are satisfied. But from (2) it is not clear that (1) is minimized.

The great mathematician Lagrange apparently looked at the result, equation (2), and realized that he could have it far more simply by extremalizing the following quadratic form

$$Q_L(\mathbf{x}, \lambda) = (\mathbf{y} - \mathbf{B}\mathbf{x})'(\mathbf{y} - \mathbf{B}\mathbf{x}) + (\mathbf{d} - \mathbf{G}\mathbf{x})'\lambda + \lambda'(\mathbf{d} - \mathbf{G}\mathbf{x}) \tag{3}$$

as you can quickly verify by setting to zero the derivatives w.r.t. $\mathbf{x}'$ and λ'. Naturally everyone prefers to handle constraints by Lagrange's method. Unfortunately Lagrange failed to pass on to the teachers of this world an intuitive reason *why* extremalizing (3) gives the same result as extremalizing (1). Lagrange's quadratic form is not even positive definite (it cannot be written as something times its conjugate). In honor of Lagrange, the variables λ have come to be known as Lagrange multipliers.

When I sat down to write this book I promised myself to include no equations without a practical use, so now I'll explain how you can use these equations to prepare pedantic articles on geophysical inversion. You can derive marvelously complicated equations by formally solving equation (2) as you would any 2×2 set. The solution balloons up in size, particularly since the elements of equation (2) are submatrices and they cannot be commuted. You justify these equations by stating that geophysical measurements are finite in number and thus all equations with measurements are constraints to an infinite-dimensional optimization problem for finding the earth model which is a continuous function. You can obfuscate further by introducing weighting functions that are inverse covariance matrices of the unknowns, and thus fully as unknown as the unknowns themselves.