Chapter 8

Nonlinear problems

All the least squares problems in earlier chapters are merely the first step in a wider class of problems—nonlinear problems. Nonlinear problems typically arise when the unknowns come in two flavors. The slogan "don't add apples to oranges" applies in reverse, if you have apples and oranges, it is unlikely they will be combined by simple addition—a more complicated *nonlinear* relationship likely relates the unknowns. Linear relationships apply when we pretend that apples are constant while we solve for the oranges, and vice versa. The nonlinear relationships we will study here are little more than a systematic way of letting everything vary at once.

OPTIMUM FILTER WITH MISSING DATA

Recall the missing data figures beginning with Figure **??**. There the filters were taken as known and the unknowns were only the missing data. Now instead of having a predetermined filter we'll solve for the filter along with the missing data. The principle used is that the output power is minimized while the filter is constrained to have one nonzero coefficient (else all the coefficients would go to zero). First we'll look at some results and then see how they were found.

In Figure 1 the filter is constrained to be of the form $(1, a_1, a_2)$. The result is

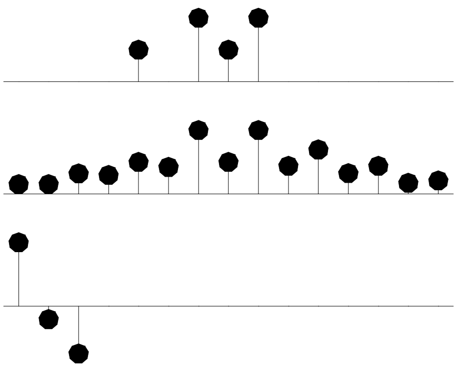

Figure 1: Top is known data. Middle includes the interpolated values. Bottom is the filter with the leftmost point constrained to be unity. jon3-missif [NR]

pleasing in that the interpolated traces have the same general character as the given values. The filter came out slightly different from the $(1, 0, -1)$ that I suggested based on a subjective analysis for Figure ??. Curiously, constraining the filter to be of the form $(a_{-2}, a_{-1}, 1)$ in Figure 2 yields the same interpolated missing data as in the previous figure. I understand the sum squared of the coefficients of $A(Z)Y(Z)$ is the same as that of $A(1/Z)Y(Z)$ but don't see why that would imply the same interpolated data.

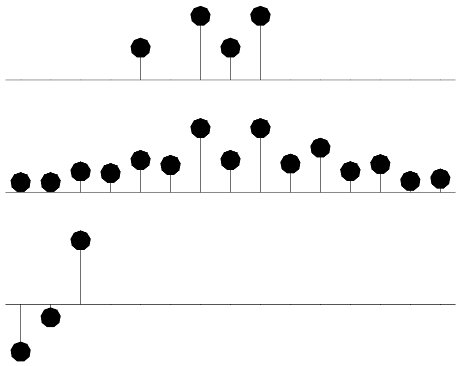

Figure 2: Top is known data. Middle includes the interpolated values. Bottom is the filter with the rightmost point constrained to be unity. jon3-backwards [NR]

Changing the filter constraint to be of the form $(a_{-1}, 1, a_1)$ yields a much different interpolation shown in Figure 3. The interpolated data is much smoother than the given data. This is because the missing data is produced from an interpolation-error filter instead of a prediction-error filter. A fact not evident from the figures is that the value of the output power was smaller for interpolation than for prediction. I cannot explain why the filter turned out to be imperceptibly different from a second difference operator, but somewhere I think I have a proof that it is a symmetrical operator.

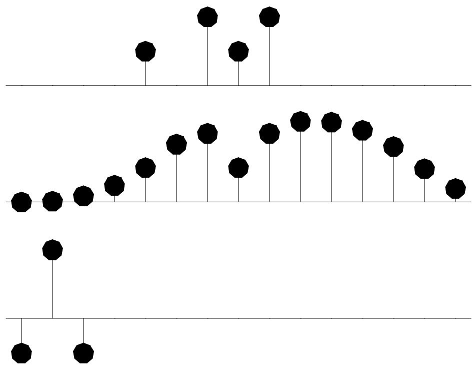

Figure 3: Top is known data with missing data represented by zeros. Middle includes the interpolated values. Bottom is the filter. jon3-center [NR]

Packing both missing data and filter into a CG vector

Define a roughening filter $A(Z)$ and a data signal $Y(Z)$ at some stage of interpolation. The regression is $0 \approx A(Z)Y(Z)$ where the filter $A(Z)$ has at least one coefficient constrained to be nonzero and the data contains both known and missing values.

Think of perturbations ΔA and ΔY. Neglecting the nonlinear term $\Delta A\,\Delta Y$ we have $-A(Z)Y(Z) \approx A(Z)\,\Delta Y(Z) + Y(Z)\,\Delta A(Z)$. To make a program such as `miss1()` from chapter 6. we need to pack both unknowns into a single vector $\Delta X = (\Delta Y, \Delta A)$ before calling the conjugate-gradient program. Likewise the resulting filter and data coming out must be unpacked. Also, the gradient now has two contributions, one from $A\,\Delta Y$ and one from $Y\,\Delta A$ and these must be added. These book-keeping operations are done in subroutine `cgpack()`.

```
subroutine cgpack( iter, ny,y,dy, na,a,da, nr,r,dry,dra, s, ss)
integer iter, iy, ny, ia,na, ir,nr,  cg
real y(ny),dy(ny),dry(nr),  a(na),da(na),dra(nr), r(nr), s(ny+na),ss(ny+na-1)
temporary real x(ny+na), g(ny+na), gg(nr)
do iy=1,ny { x(iy)    =  y(iy);         g(iy)    = dy(iy) }
do ia=1,na { x(ny+ia) =  a(ia);         g(ny+ia) = da(ia) }
do ir=1,nr { gg(ir) = dry(ir) + dra(ir) }
#
call cgstep( iter, na+ny, x, g, s, nr, r, gg, ss)
#
do iy=1,ny { y(iy) = x(iy);             dy(iy) = g(iy) }
do ia=1,na { a(ia) = x(ny+ia);          da(ia) = g(ny+ia) }
return; end
```

The program, `missif()` that made Figures 1 thru 3, closely resembles `miss1()`. Besides using `cgpack()` the difference is that to avoid accumulation of errors from the neglect of the nonlinear product $\Delta A\,\Delta Y$, the residual is recalculated inside the iteration loop instead of once at the beginning.

```
# find filter and missing data on 1-axis by minimizing power out.
#
subroutine missif( na, lag, a, ny, y, copy, niter)
integer i, ia, lag, na,  iy,ny,  ir,nr,  iter,niter
real    y(ny)           # input: known data with zeros for missing values.
                        # output: known plus missing data.
real    copy(ny)        # copy(iy) vanishes where y(iy) is a missing value.
real    a(na)           # input and output: roughening filter
temporary real  da(na), sa(na), dy(ny), sy(ny)
temporary real  dry(ny+na-1), dra(ny+na-1), dr(ny+na-1), r(ny+na-1)
temporary real  s(ny+na), ss(ny+na-1)
nr = ny+na-1
if( a(lag) == 0. )      call erexit('missif: input needs a(lag)!= 0.')
do iter = 0, niter {                            # niter= number missing about
        call contran( 0, na,a, ny,y,  r)        # r = a*y
        do ir=1,nr
                r(ir) = - r(ir)                 # residual = - filtered data
        call contran( 1, na,a, ny,dy, r)        # dy(a,r)
        call contran( 1, ny,y, na,da, r)        # da(y,r)
        do iy=1,ny
                if( copy(iy) != 0.)             # missing data where copy(iy)==0
                        dy(iy) = 0.             # can't change known data
        da(lag) = 0.                            # constrained filter point
        call contran( 0, na,a, ny,dy, dry)      # dry=a*dy
```

```
        call contran( 0, ny,y, na,da, dra)      # dra=y*da
        call cgpack( iter, ny,y,dy, na,a,da, nr,r,dry,dra, s, ss)
        }
return; end
```

I'll probably return to `missif()` and do something about preconditioning. The convergence can be very slow or fail if `a()` and `y()` are much out of scale with each other.

I have noticed that the convergence no longer appears to be quadratic, so I'll be checking into "partan" and other methods of optimization found in Luenburger's book. I believe quadratic convergence can be extremely valuable and is very little extra effort and should not be neglected. The Polak-Ribiere method described in Luenburger is now being used by Biondo and Jos.

REFERENCES

Luenberger, D. G., 1973, Introduction to linear and nonlinear programming: Addison-Wesley.

INTERPOLATION BY PLANE WAVE DESTRUCTION

Program `iner()` (SEP-61 page 399) found prediction and interpolation filters by minimizing the output power for a given input data. Program `miss1()` found partly unknown data by minimizing output power from a given filter. Program `missf()` found unknown data and an unknown filter by minimizing output power. Now we'll do that again except the filter will be constrained to be of the form of a plane-wave destructor (PWD) which is a simple two-dimensional filter. It is little inconvenience that a PWD is a two dimensional filter. A more serious inconvenience is that in practice the data space is a wall of data and the plane-wave model is only valid in regions of limited extent. We have already solved this problem without the missing data. Effectively the p for each window is independent of the others. The residuals were another matter, we had to add together the partially overlapping residuals, keeping track of how many windows overlapped each point in space. Now the question is, can I splice all this stuff together?

PLANE WAVE IDENTIFICATION BEYOND ALIASING

A central problem in seismology is that we can't afford to cover the earth with instrumentation. Thus our receivers (and senders) tend to be widely spaced, often so widely spaced that signals are aliased in space. I adapted some synthetic seismograms from Spitz in Figure 4. Signals that are plotted entirely as zeros are taken to be missing.

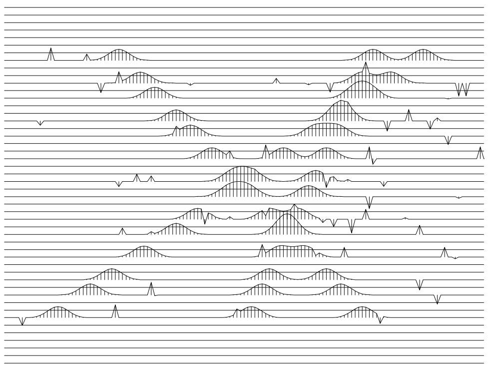

Figure 4: Modified Spitz model. Three ideal plane waves with impulsive noise and irregularly spaced missing traces. jon3-spitz [NR]

Besides missing traces on either side are irregularly interleaved zero signals depicting missing traces. Here we set for ourselves the goal of filling in the missing traces in Figure 4 as well as finding the three slopes, p_1, p_2, and p_3.

The problem of finding the wave slopes in data such as Figure 4 is an old one in seismology. A tremendous volume of human intellect has been brought to bear upon it. To begin with, the problem is widely solved using an intuitive approach which is simply the conjugate operator. In other words, the data can be regarded as a superposition of plane waves of all possible angles. The conjugate operation, called "slant stack" is to sum the data sample over many angles. Slant stack has an analytical inverse (see for example "Imaging the Earth's Interior"). The trouble is that theoretical inverse hinges on absence of aliasing and truncation—essential features of Figure 4. A problem with the conjugate is that its resolution is not sharp. When one of the plane waves is much stronger than the others then truncation and aliasing of the strong event can hide the weaker events. The method we'll develop here should be able to see weak events in the presence of strong ones.

REFERENCES

Spitz, Simon, 1989, Trace Interpolation beyond aliasing and without picking, Expanded abstracts of the SEG 1989 meeting in Dallas.

DIP FILTER DEFINITION

A dip filter is a two-dimensional filter that is intended to weight information depending on its dip and not on its frequency content. Dip filters may be designed by Fourier analysis which tends to make filters of great extent, or by partial differential operators which makes filters that are short, but imperfect near Nyquist frequencies. For example the 1-D filter $(1, -1)$ is like a time derivative only far from Nyquist. Likewise the 2-D filter made by expressing the partial differential operator $\partial_x - p\partial_t$ as a differencing star on a mesh has its expected dip behavior only far from Nyquist. The derivative filter creates an output with a vanishing zero-frequency component.

The dip filter creates an output which vanishes at dip p in dip space. (For more details about dip space, see IEI.)

When the mesh is refined, say by interlacing a new row and a new column between each existing row and column, two interesting questions arise. First we'll consider the questions independently, and then we'll consider them together. The first question is how to reexpress the filter on the refined mesh. Since refining both the t mesh and the x mesh simultaneously preserves the apparent dip, the values on the differencing star of a dip filter should remain unchanged—you just push them closer together. The second question is how to find new data values that interlace the given data values. This missing-data problem is one of linear least squares that is straightforward in principle. As I described in an earlier chapter the method fills out the data space by minimizing the output energy from the given filter. Interpolating the time axis is a boring question, but interpolating the space axis in seismology is a question of great interest because we often encounter spatially aliased data which frustrates many of our data analysis procedures.

INTERPOLATION WITH P.D.E. DIP FILTERS

For difference representations of partial differential operators, mesh refinement is straightforward because expressions for the differencing star include Δt and Δx which can simply be halved. The star for $\partial_x - p\partial_t$ is *unchanged* if Δt and Δx are simultaneously halved. If a dipping wavefield $u(x - p_1 t)$ is input to this operator the output vanishes if $p = p_1$, so this operator deserves the name "dip filter" or "dip rejection filter" or "plane-wave destruction filter". As deconvolution offers well-known opportunities with narrow-banded frequency spectra, the possibility of destroying a perfect plane wave with only two seismograms offers similar opportunities.

Spatial interpolation with $\partial_x - p\partial_t$

Given data on a mesh and the dip filter form $\partial_x - p\partial_t$ we can find the best fitting value of p by a univariate linear-least squares procedure in chapter 4. In summary, take the filter output $v(t,x) = \partial_x u(t,x) - p\partial_t u(t,x)$ and minimize it, $v(t,x) \approx 0$, in the least-squares sense by variation of the scalar variable p. The resulting p is simply a ratio of dot products (summation over t and x) with a numerator of u_x dotted into u_t and a denominator of u_t dotted into itself.

The difference star representing $\partial_x - p\partial_t$ is unchanged when the mesh is interleaved simultaneously on t and x. Finding the missing values on the t-axis is straightforward by any interpolation method. Finding the missing values on the x-axis seems straightforward by the CG method. Those with wave-equation solving experience might suggest a systematic procedure to have $v(t,x) = 0$ vanish exactly on the new traces. I'll outline the procedure to give confidence that the least-squares task need not be a horrific one, despite the huge number of unknowns. The differencing star is 2×2. Suppose the right side is placed on a known trace, the left side multiplies an unknown trace. The unknown trace can be found by solving a bi-diagonal set of simultaneous equations. This will involve recursion up or down the time axis depending on the sign of p. Since p will typically vary with t and take both signs in the range, well-known methods are needed. Hopefully the CG method takes care of this in a fairly automatic way. This deserves investigation. Potentially the CG method requires as many iterations as there are data points, so this could be prohibitive when the task is set up in a huge window of data. If the CG method is slow here, we'll gain some valuable insights on how to speed it up because we think we understand the mechanisms so well in this example. On the other hand, the CG method does not try for zero error when extrapolating from right to left, but for minimum averaged error when extrapolating both ways. Unfortunately, I have burned myself out on programming, so I have no examples to show you.

PDE to reject two dips

You can reject two dips with the operator

$$(\partial_x - p_1\partial_t)(\partial_x - p_2\partial_t) \tag{1}$$

Finding the two values p_1 and p_2 is a nonlinear problem that is easy. Let u be the input signal and v be the output signal. Consider

$$\left(\frac{\partial^2}{\partial x^2} + a\frac{\partial^2}{\partial x \partial t} + b\frac{\partial^2}{\partial t^2} \right) u(t,x) = v(t,x) \quad \approx \quad 0 \tag{2}$$

Now recognize that (2) (which is a separate equation at each point in the (t, x) space of $v(t, x)$) is an overdetermined set of linear equations for the two unknowns a and b. It is easy to find a and b which by comparison with equation (1) gives p_1 and p_2 by the nonlinear but easy equations $a = p_1 + p_2$ and $b = p_1p_2$. The minimum signal required is three seismograms (on which to express ∂_{xx}).

To recapitulate, first a simple procedure gives us the required coefficients for a filter that fits two waves to the dataset $u(t, x)$. Second, the same dip filter coefficients can be applied on a mesh in which t and x are interleaved, thus introducing new data locations on which we need data. Third, we interpolate t by any method. Fourth, we find missing traces by minimizing the power in $v(t, x)$.

Does it work?

The really interesting issue is whether this method works for data that is inadequately sampled in x, i.e. when the spectrum of the underlying wavefield is nonvanishing above the spatial Nyquist frequency of the sampled field. I think it might work if the spectrum has energy below spatial Nyquist as well as above. The method that chooses p might be insensitive to signal above spatial Nyquist because temporal frequencies below Nyquist are all agreed on the same p whereas temporal frequencies above the spatial Nyquist fit different values of p. After the wavefields are interpolated, perhaps by repetitive interlacing, the question of aliasing no longer arises. So, if you can bootstrap yourself up to the right solution, I think you will stay there.

INTERPOLATION WITH SPATIAL PREDICTORS

Because dip filters based on P.D.E.s deal poorly with aliased information we'll look at a more general kind of filter that can be a dip filter even for data that is spatially aliased with *no* signal satisfying the spatial Nyquist condition.

Spatial prediction filters as dip filters

A spatial prediction filter can take the form of a dip filter. Using linear least squares we can estimate the coefficients (a, b, c, d, e) in the 2D filter

$$\begin{array}{cc} & \mathrm{a} \\ & \mathrm{b} \\ 1 & \mathrm{c} \\ & \mathrm{d} \\ & \mathrm{e} \end{array}$$

Fitting the filter to two neighboring traces that are identical but for a time shift, the filter (a, b, c, d, e) should turn out to be the negative of an interpolation filter. Ideally you might see $(-1, 0, 0, 0, 0)$ or $(0, 0, -.5, -.5, 0)$. But if the two channels are not fully coherent you expect to see something like $(-.9, 0, 0, 0, 0)$ or $(0, 0, -.4, -.4, 0)$.

Refining both t and x with a spatial predictor

Having determined a filter, say $(0, 0, -.5, -.5, 0)$ on the original data mesh, we can now interleave both t and x and expect to use the identical filter. Everyone knows how to interpolate data on the time mesh. The above filter interpolates space by extrapolating leftward. To extrapolate right, we simply reverse t and x in the star. So the spatial interpolation problem seems to be solved whenever adjoining traces are perfectly coherent, regardless of the signal shift. A "spatial aliasing" issue does not arise.

When the coherence of the channels is less than perfect we have an interesting question remaining, but it does not seem to be a critical one.

The prediction form of a two-dip filter

Consider a uniform mesh on the space axis and a continuum on the time axis. A simple dip destruction filter operating in (t, x)-space is $\delta_{0,0} - \delta_{p_1 \Delta x, \Delta x}$. Convolving this two-pulse filter on any (t, x)-data plane destroys the p_1 dip component in that plane. (Of course if wavelengths are less than $2\Delta x$, more than only the dip p_1 will be destroyed.)

The operator that destroys two dips is a product in Fourier space or a convolution in (t, x)-space. The convolution is

$$\delta_{0,0} - \delta_{p_1 \Delta x, \Delta x} - \delta_{p_2 \Delta x, \Delta x} + \delta_{(p_1+p_2)\Delta x, 2\Delta x}. \tag{3}$$

The problem of estimating the filter in (3) is dreadfully non-linear, though a global 2-D search might work. But (3) suggests the 2D filter

$$\begin{array}{ccc} & a & v \\ & b & w \\ 1 & c & x \\ & d & y \\ & e & z \end{array}$$

After finding the coefficients, this filter can be applied on a mesh with both t and x interleaved. We are somewhat at the mercy of the regression procedure that determines all the coefficients in the filter. We have not imposed the constraint that the filter be a dip filter, so maybe it will be something undesirable in addition to being a dip filter. If the dip spectrum has nonzero bandwidth, the filter can't be exactly right, but it may perform adequately. We should try it. Perhaps it will work the way it is or perhaps we need to find some way to limit the number of free parameters in the differencing star to force it to fit the mold of a narrow banded dip filter.

Nonlinear least squares

The estimation methodology so far is linear least squares. First we found a spatial predictor on a mesh. Then we introduce missing data by interlacing the mesh on both axes. Then we assert the two-dimensional filter is a dip filter and that it is appropriate for use on the doubly refined mesh. With this operator, we then solve the linear least squares problem for the missing data.

We made an approximation when we presumed the 2D filter was essentially a dip filter and that the imperfect coherency was not significant. Now is the time to patch up that approximation by solving the nonlinear least-squares problem allowing both the filter and the missing data to readjust on the principle that the output power be minimized. Presumeably we are already close to the solution so we can expect rapid convergence.

CLASH IN PHILOSOPHIES

One philosophy of geophysical data analysis called "inverse theory" says that missing data is irrelevant. According to this philosophy a good geophysical model only needs to fit the real data, not interpolated or extrapolated data, so why bother with interpolated or extrapolated data? It must be a hoax. Most academic inverse theorists as well as our own Peter Mora seem to belong to this school of thought. Even some experienced practitioners belong to this school of thought. My good friend Boris Zavalishin says, "Don't trust the data you haven't paid for."

Another philosophy is that 40 years of geophysical data processing cannot be all wrong. There must be some good reason why we interpolate and extrapolate data though perhaps we do not clearly explain why. Let me cite some examples:

- Spectral analysis with prediction-error filters shows that unexpectedly high resolution can be achieved reliably with certain kinds of data sets. Conceptually this amounts to extending the data set with nonzero values beyond the recorded region. Familiar statments about limited resolving power generally presume data vanishes beyond where it is recorded.
- By now we are all accustomed migration side-boundary artifacts. We eliminate these by either tapering data near the sides (falsifying the data) or by extending nonzero data into the region beyond where data is recorded.
- Everyone does velocity analysis by hyperbola scan whereas most people do migration by Fourier analysis or finite differences. Since they are mathematically equivalent, why is the velocity analysis *never* implemented by the other two methods? Maybe it has something to do with the spatial sampling of the wide-offset data.
- Certain idealized data sets can be much more effectively interpolated and extrapolated by human beings than by any known mathematical analysis. We frequently see data with a limited dip spectrum but seem unable to exploit it because of our clumsiness with spatially aliased data.

Are we wasting our time interpolating and extrapolating the observed data field, or is some good purpose being served, if so, what? How can the conflict between the "inversion" school of thought and the "missing data" school of thought be resolved?

Relying on interpolated data?

Let us look at the issue of "relying on data not paid for". First we'll see that interpolation and extrapolation need do no harm. Conventional inversion can be expressed by minimizing the quadratic form

$$Q(\mathbf{x}) = \sum_k (d_k - \mathbf{a}_k \cdot \mathbf{x})^2 \ + \ \lambda\, \mathbf{x}' \cdot \mathbf{x} \tag{4}$$

where $\mathbf{x}$ is the model vector, $\mathbf{a}_k$ is the kth row of a partial derivative matrix, d_k is the kth $k$nown data point, λ is a scalar, and to avoid clutter I have neglected the residual weighting. The missing data philosophy is to add terms involving the missing data d_m, namely

$$Q(\mathbf{x}, d_m) = \sum_k (d_k - \mathbf{a}_k \cdot \mathbf{x})^2 \; + \; \sum_m (d_m - \mathbf{a}_m \cdot \mathbf{x})^2 \; + \; \lambda \, \mathbf{x}' \cdot \mathbf{x} \tag{5}$$

Since the missing data d_m are freely adjusted variables, it is evident that the value of Q at the minimum is the same both with and without the missing data. Missing data interpolated or extrapolated in this way cannot perturb the solution $\mathbf{x}$. Of course there are many other ways to interpolate and extrapolate, and they are generally less sophisticated than the model-based method I am describing. How can they be of value?

Rate of convergence

Perhaps we should consider the *rate of convergence* of the optimization method. Glass will turn to crystal if you wait long enough. But we never can wait long enough. Inversion methods are notorious for slow convergence. Consider that matrix inversion costs are proportional to the cube of the number of unknowns; our most powerful computers balk when the number of unknowns goes above the mid hundreds; and our images generally have millions. Perhaps the crystal quartz analogy is not far fetched.

The question is whether explicitly including the missing data slows things down by adding more unknowns and complexity, or whether it can speed things up. Matrix inversion by iterative methods amounts to iterative application of an operator and its conjugate. Given $\mathbf{d} \approx \mathbf{A}\mathbf{x}$, the solution $\mathbf{x}$ is implicitly developed in a power series

$$\mathbf{x} = \left[\sum_n \alpha_n (\mathbf{A}'\mathbf{A})^n \right] \mathbf{A}'\mathbf{d} \tag{6}$$

where the α_n are implicitly determined by the numerical method (such as steepest descent or conjugate gradients). The operator $\mathbf{A}$ has the rows $\mathbf{a}_k$ on top and under those rows are optional rows of $\mathbf{a}_m$ and under that are the optional stabizing rows of $\sqrt{\lambda}\,\mathbf{I}$. Why should convergence be faster when the matrix $\mathbf{A}$ includes missing data rows? I'll suggest that when the data space is completed (made large enough to fully determine the solution) then the conjugate operator $\mathbf{A}'$ has a better chance of being close to the inverse operator. Many of our data processing procedures are so huge that we do little more than apply the conjugate operator, and the conjugate to the missing-data-extended operator is better than the conjugate to the data-truncated operator.

Estimating the factors of the covariance matrix

Philosophically, what is the role of the nonlinear step on the missing data estimation? The conventional wisdom is that before you begin linear estimation you somehow measure the covariance matrix of the noise. The covariance matrix of the solution is taken to be a prior(!). During linear estimation these covariance matrices are constants. In real life, by which I mean (1) practical deconvolution filter design, (2) the missing data estimation I showed a few weeks ago in my textbook, and (3) in this paper, and probably other places too, the estimation of the covariance matrices proceeds *simultaneously* with the optimization, and it too is an optimization problem. The nonlinear step accomodates the limitation of the maximum-liklihood estimation paradyme that the residual covariance matrix be a prior. The Burg procedure is linear only because it is a sequence of linear estimations of reflection coefficients, each based on the previous. The missing-data-unknown-filter program `missif()` is intrinsically nonlinear (given the prior constraint that a filter be of prediction error form and given some of the data values are known, find the filter and the missing data values to minimize output power). The sparse dip spectrum problem addressed in this paper is essentially nonlinear, which is why this paper gives so much emphasis on good starting estimates. Note that we do not really find the inverse covariance matrix, but an approximate factorization of it, the prediction-error filter (prediction in space). Both problems are characterized by a vanishing of the model $\mathbf{x}$ and the partial derivative matrix $\mathbf{A}$. There is nothing left here but the missing data and the unknown filter (which is the factored covariance matrix). So this activity doesn't fit academic inversion theory at all.

Throw out the model covariance matrix

I won't say the covariance of the model can always be ignored, but since it must be a prior given, I would like to find some way around it. Thus I suggest that every data space be augmented till it has the dimensionality and completeness required to determine a solution. It this cannot be done fully, it should still be done to the extent feasible. For example we might omit evanescent waves, but not waves blocked by an aperture. Reinaldo and I are doing just that in tomography.

The covariance matrix of the residual in data space (missing and observed) seems a reasonable thing to estimate—unlike the covariance matrix of the model. I think this matrix should not be thought of as a covariance matrix of the solution, but as an interpolation function for plotting the solution.