NULL-SPACE NIGHTMARES

Twice in my life I landed in a pitfall that kept me back multiple months. When engulfed in a complicated application anyone might overlook some academic fundamentals. Since you've already read the title of this section, you should recognize the pitfall much quicker than I did. To make this more memorable, I present you the puzzle before the solution giving you a chance to guess it.

One of the pitfall situations is in the Galilee study coming in Chapter **??**. We have there a water-depth survey of a lake. The survey was done at unknown intervals over a couple seasons. We discovered our water-bottom solution showed ship's tracks in it, so we concluded the water surface had been changing during the survey. Consequently we set to find both the water top as well as its bottom. We considered the problem formulation

$$\mathbf{0} \;\approx\; \mathbf{Bh} \;+\; \mathbf{Iu} \;-\; \mathbf{d} \tag{1}$$

where $\mathbf{B}$ (adjoint binning) extracts from the (unknown) water depth map $\mathbf{h} = h(x, y)$ onto the data track $\mathbf{d} = d(s)$ where s is the measurement number mapped in some unknown way to calendar and clock time. The new unknown is water surface altitude $\mathbf{u} = u(s)$.

An obvious difficulty with the regression (1) is that we have introduced as many unknowns in $\mathbf{u}$ as we have data values in $\mathbf{d}$. To address this fact, we define $u(s)$ in such a way that it must be a slowly variable function of s. Let $\mathbf{L}$ be a low pass filter on some unknown random noise (new unknowns) $\mathbf{u} = \mathbf{Ln}$. That's preconditioning the surface altitude drift. We also precondition the map with a smoother $\mathbf{S}$ like the inverse helix operator, say $\mathbf{h} = \mathbf{Sp}$. Because we are still likely under-determined we are going to need some regularizations, so let's add one for $\mathbf{p}$ and another for $\mathbf{n}$.

$$\mathbf{0} \;\approx\; \mathbf{r} \;=\; \mathbf{BSp} \;+\; \mathbf{Ln} \;-\; \mathbf{d} \tag{2}$$

$$\mathbf{0} \;\approx\; \epsilon_p \, \mathbf{p} \tag{3}$$

$$\mathbf{0} \;\approx\; \epsilon_n \, \mathbf{n} \tag{4}$$

Keeping in mind there is a null-space issue and that we really don't know what the epsilons should be, we reason that the only real issue is their ratio $\lambda = \epsilon_h / \epsilon_n$. So, using the "faking the epsilon" trick we set out iterating in the direction

$$\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{n} \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{S}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} \\ \lambda\mathbf{L}^{\mathrm{T}} \end{bmatrix} \mathbf{r} \tag{5}$$

The code seemed to converge quite rapidly, but the result contained bad news. This 40 meter deep lake had a water surface that in the middle of the lake was depressed by one meter! Playing with the λ and the low pass filter $\mathbf{L}$ we could make the surface depression go away, but then the ship tracks reappeared in the water bottom map. No parameter adjustments would get both simultaneously correct.

Your turn to think about this for a while before I remind you of the mathematical fundamentals.

The answer (I hope!)

Recall the null space for a problem of fitting one data point d using two model points.

$$d \quad = \quad \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \tag{6}$$

The null space is any solution $\mathbf{m}$ that produces no data. Here it is $(m_1, m_2) = (1, 1)$. You can add an arbitrary amount of it to any solution getting another solution as good as the first. Likewise, any constant added to the water bottom and subtracted from its top has an identical fit to the data, so there is here some kind of low-frequency null space.

The most basic 2-D null-space problem is a parabolic penalty on one spatial axis with no penalty on the other axis. Imagine a house facing north-east with a parabolic rain gutter mounted perfectly horizontally on one edge of the house roof. The null space is anywhere on the center line along the bottom of the gutter. Anywhere you begin, steepest descent brings you immediately to the gutter bottom in a location that depends on where you began. Now tilt the gutter a little bit so the water drips off one end of the rain drain. Steepest descent now begins a slow zig-zag path to that end. Conjugate directions would bring you there quicker in the 2-D problem, but in the 150,000 dimensional lake bottom problem, conjugate directions taken only a few dozen iterations is unlikely to bring you to the actual bottom. The incipient null-space casts considerable doubt on our assumption that we can get a suitable answer in a few dozen iterations. So what can we do?

Early arrival in a good place requires only that we begin from a good place, and that is easy to do. It's a pitfall to begin from $(\mathbf{p}, \mathbf{n}) = (\mathbf{0}, \mathbf{0})$ and then run in the direction (5). Instead, set $\lambda = 0$ and run till apparent convergence getting the map with the tracks. Then turn λ on and run more iterations to estimate the little bit of surface drift required to remove the tracks.

Lesson learned

It is easy to imagine a null-space hazard in any situation like this,

$$\mathbf{d} \quad = \quad \begin{bmatrix} \frac{\partial \mathbf{d}}{\partial m_1} & \frac{\partial \mathbf{d}}{\partial m_2} \end{bmatrix} \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \end{bmatrix} \tag{7}$$

for example in seismology finding a velocity image along with an anisotropy image.

The decon null-space pitfall

The first time I was multi-month frustrated by a null-space pitfall was in a complicated non-linear research problem not in this book. Students were getting beautiful

solutions to a classic problem, but for reasons we did not understand, the beautiful results sometimes came with the wrong polarity and sometimes with unexpected time shifts. Aaack! We thought it had something to do with the non-linearity of the problem and spent a lot of time thinking about how to guide the descent. Then we learned we were getting excellent solutions early on, but by running many iterations the solution would drift away in an unpredictable manner. This woke us up to our need for regularization. We needed to stop while we had a good solution! We might have recognized that earlier had we plotted the residual as a function of iteration. We were solving for the complex logarithm of a filter, something with a hard-to-understand relationship to our final result. The needed regularization was not the usual weighting function. Struggle led me to finally grasp that we needed a weight upon the antisymmetric part of the imaginary part of what we were solving for. Learning that, however, did lead to a delightful publication with good data results. Had I grasped months earlier that I should forget nonlinearity issues and needed to concentrate on the regularization, I should have found it much earlier.

Old, old earthquake example

In the dawn of the era of computerized earthquake seismology someone decided to add earthquake depth to their catalog. Traditionally, they had solved for only three things, latitude, longitude, and time of source at the source, i.e. origin time. Now they would add a fourth, the depth. They wrote down the 4×4 system of equations and solved it. Erratic results. So then they froze the depth at zero, solved for the old three variables, and then introduced the depth. Problem solved. Is this a null-space problem or a non-linear problem? They pointed out that when any seismometer is far from the earthquake, the waves arrive nearly straight up which means a depth shift is a lot like an origin-time shift.

> Don't go after the little stuff until after you are close-in with the big.