Objective Mapping: What to Do When the Matrices Are Too Big to Invert

The basic objective mapping algorithm allows you to map data ϕ to a grid of output points, provided that you know the data-data covariance $\mathbf{A}$, the noise-to-signal ratio ϵ and the data-grid covariance $\mathbf{P}$. Then

$$\hat{\boldsymbol{\psi}} = \mathbf{P}(\mathbf{A} + \epsilon\mathbf{I})^{-1}\boldsymbol{\phi}. \tag{1}$$

One challenge with this algorithm is the requirement to invert the matrix $(\mathbf{A} + \epsilon\mathbf{I})$. What do you do when the matrix is too big for your computer to invert easily in Matlab? Here's a laundry list of possibilities:

(a) *Speed up your algorithm.* Matlab tends to be particularly slow because of the way it allocates memory. If Matlab needs more memory than your computer has, your computer will try to swap information in and out of active memory. This slows down the calculations dramatically. If you can control your memory requirements, you can speed up your code. Some possibly strategies are:

- Use the "pack" command to clear memory in your current Matlab session. (Note: this won't help if your problems stem from the size of your matrices rather than the complexity of your Matlab session.)
- Use the "single" command to force single precision calculations. (Note: this may have serious repercussions, if it makes your resulting output less accurate.)
- Pre-allocate memory for your arrays by creating an initial matrix of zeros that is the size of the matrix you eventually will need to fill. See http://www.mathworks.com/support/solutions/en/data/1-18150/ for a discussion of how and why to do this.
- Force the matrix $(\mathbf{A} + \epsilon\mathbf{I})$ to be sparse by setting all near-zero elements to zero (while retaining symmetry across the diagonal) and then using the "sparse" command. Then for the matrix inversion, you can try "lsqr" or another sparse operator.

(b) *Find more computer resources.* If you're short of memory and especially if you're relying on a laptop for your calculations, you can probably speed things up by finding an available large desktop or server-class machine. Or you can resign yourself to running a job that will take many hours, but look for a second machine for running extra cases. This strategy is probably not useful for a class assignment with a short time horizon, but it is a standard approach for research.

(c) *Use a compiled language rather than Matlab.* Since Matlab poses memory allocation issues, you may be able to speed your code quite a bit by rewriting it in Fortran, using the Linpack/Lapack matrix inversion routines. This is also a strategy that is pertinent for research but less relevant for a class assignment.

(d) *Make the problem smaller.* If you simply need to debug code or demonstrate functionality for an algorithm, you can speed things up quite a bit by choosing a smaller domain for your research. You don't have to examine the entire historical data record in order to demonstrate that your code works.

(e) *Average data together.* One standard strategy is to pre-average data points together (e.g. David, R. E., J. Geophys. Res., 1998).

(f) *Solve small regions and patch the solutions together.* If you are reluctant to combine data without using the objective mapping machinery, you can also break the problem into smaller geographic domains. The mapped grids do not need to overlap, but the data used to produce each patch should overlap by at least one full decorrelation radius, and you will want to verify that when you patch results together, you do not create discontinuities at the boundaries. (See e.g. Gille, S. T., J. Phys. Oceanogr., 2003.)

(g) *Solve the problem with some of the data, and then use a recursive update procedure to incorporate in additional observations.* Wunsch (1996, pp. 208-209) describes the essence of the recursive estimate under the assumption that the error covariance matrices for the two estimates are completely independent. Here are detailed notes on implementation of recursive update approaches.

In real world applications, error covariance matrices are not generally independent. For example, ALACE floats never sample under the sea ice surrounding Antarctica, so errors associated with ice-covered points are uniformly defined to be 100% of the a priori error. Similarly, ALACE coverage is much denser in the core of the ACC than it is in the more quiescent region to the north. As a result, in independent objective maps, the distribution of large-error regions is likely to be highly correlated. Simple application of the recursive update technique will result in error covariances that are too small by as much as a factor of two.

This problem is easily corrected by rederiving the recursive update equations for the special case where the error covariance matrices are correlated. Thus

$$\mathbf{x}_{new} = \mathbf{x}_1 + (\mathbf{P}_2 - \mathbf{P}_{12})(\mathbf{P}_1 + \mathbf{P}_2 - 2\mathbf{P}_{12})^{-1}(\mathbf{x}_2 - \mathbf{x}_1) \tag{2}$$

and

$$\mathbf{P}_{new} = \mathbf{P}_1 - (\mathbf{P}_1 - \mathbf{P}_{12})(\mathbf{P}_1 + \mathbf{P}_2 - 2\mathbf{P}_{12})^{-1}(\mathbf{P}_1 - \mathbf{P}_{12})^T \tag{3}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the independent prior estimates of the mapped quantity and may represent temperature, streamfunction, or any other quantity, $\mathbf{x}_{new}$ is the updated estimate of the mapped quantity. $\mathbf{P}_1$ and $\mathbf{P}_2$ represented the prior estimates of the error covariance for the two fields, $\mathbf{P}_{12}$ is the covariance of the two fields together, and $\mathbf{P}_{new}$ is the error covariance corresponding to the new mapped field $\mathbf{x}_{new}$. For this study, we use an approximation $\mathbf{P}_{12} \approx \mathbf{P}_1^4\mathbf{P}_2^4$. This expression is nearly 1 where both error covariance matrices P_1 and P_2 are near 1 and is small elsewhere. The resulting estimates of the height map do not differ substantially from heights estimated by inverting the full error covariance matrix at once.