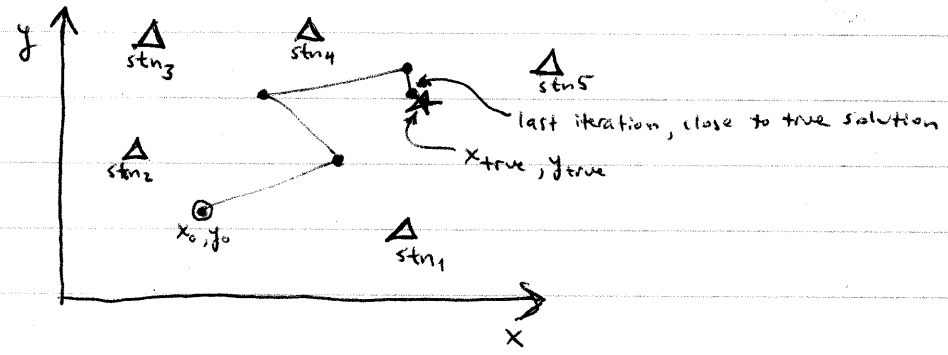
Lab #5 for Geophysical Inverse Theory ESS523, Fall 2005, Univ. of WA.
TA presenting this lecture: Andy Ganse
Course professor: Ken Creager

15 Nov 2005

Objective Surfaces in Inverse + Parameter Estimation Problems

For last week's lab regarding the EQ location parameter estimation problem, we created map plots showing a path in x, y space as our Matlab script iterated to a solution. (We actually estimated the source time t_s too, but we didn't plot that.)



What we'll concentrate on this week is what it is that this path is traversing over — this landscape is called the "objective surface" or "objective function". The objective function is a scalar real function of N variables, and in our EQ location example N=3 (x_s, y_s, t_s are the variables) although we only plot its 2D projection onto the x_s, y_s plane here. Will discuss later how we choose to do that projection.

note I tend to use these terms interchangably →

Whether we've been talking about linear or nonlinear problems, parameter estimation problems or parametrized continuous models in inverse problems, we keep talking about norms of the data residuals and of the model size.

So we can consider such a norm as a function of candidate model parameters, like this:

$$objf(\underline{m}) = \| \underline{d} - \underline{\underline{G}}(\underline{m}) \|^2$$

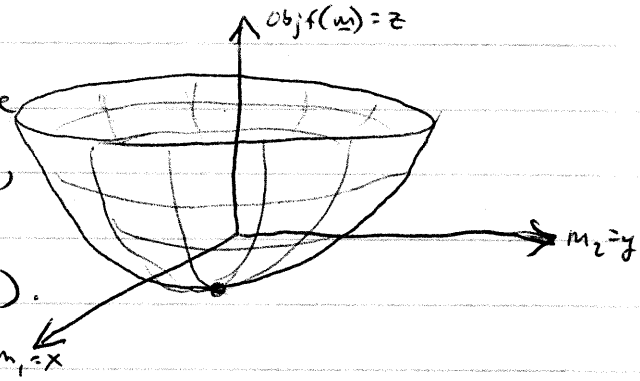
or if we're regularizing in an inverse problem maybe it looks like this:

$$objf(\underline{m}) = \| \underline{d} - \underline{\underline{G}}(\underline{m}) \|^2 + \nu \| \underline{m} \|^2$$

Objf($\underline{m}$) is the objective function, and if we were to plot it as a (scalar real) z-value for every choice of of model parameters in $\underline{m}$, we'd see some surface with various dips and valleys (in general), or one big bowl (specifically in the examples above since they're linear).

Just for some extra terminology, a given z-value (i.e. objf value) is called an "objective value" and the model parameters in $\underline{m}$ are sometimes called "objective variables". These extra terms are useful to know because optimization software may help you to find the minimum (ie. solution point) of one of these objective surfaces in a hard nonlinear problem, and these words are used in the optimization community.

So note if the goal is to minimize the norm or sum of norms above, then the solution value of $\underline{m}$ is at the minimum of objf($\underline{m}$).



Let's concentrate on <u>linear problems</u> for a few pages here... Now forgetting the model-size/inverse problem part for a bit and just concentrating on the residual norm:

$$objf(\underline{m}) = \| \underline{d} - \underline{\underline{G}}(\underline{m}) \|_2^2$$

← note narrowing down to L2 norm here

We can think about it for a few minutes and see that if $\underline{\underline{G}}(\underline{m})$ is linear, so that $\underline{\underline{G}}(\underline{m}) = \underline{\underline{H}}\underline{m}$, and if we use the L2 norm (sum of squares), then components of vector $\underline{m}$ get squared and the objective function is quadratic, eg:

(assume $\underline{\underline{H}}$ is "nice" here → not singular and not even ill-conditioned)

$$\underline{d} = \underline{\underline{H}}\,\underline{m} \quad \rightarrow \quad \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

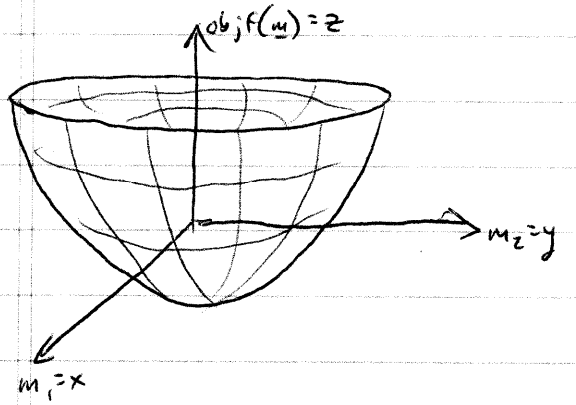
$$objf(\underline{m}) = \sum_i (d_i - H_{i,\cdot}\,\underline{m})^2 = (d_1 - h_{11}m_1 - h_{12}m_2)^2 + (d_2 - h_{21}m_1 - h_{22}m_2)^2$$

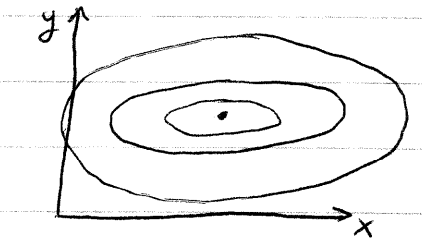
↑ ie i^{th} row of H

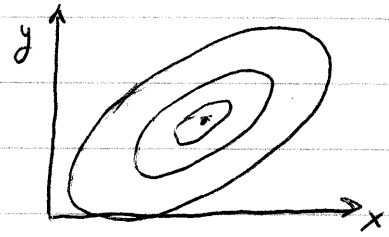
$$= d_1^2 - 2d_1h_{11}m_1 + h_{11}^2m_1^2 - 2d_1h_{12}m_2 + 2h_{11}h_{12}m_1m_2 + h_{12}^2m_2^2$$
$$+ d_2^2 - 2d_2h_{21}m_1 + h_{21}^2m_1^2 - 2d_2h_{22}m_2 + 2h_{21}h_{22}m_1m_2 + h_{22}^2m_2^2$$

So our objective surface for a linear problem with the L2 norm is a paraboloid (rotated parabola, or higher dimensional equivalent of rotated parabola). This paraboloid is a bowl (or higher dimensional equivalent of a bowl), and we know it's a bowl and not a hill because we see that the 2nd degree terms in the equation for $objf(\underline{m})$ above are positive, so the 2nd derivative is positive.

Note however that the paraboloid may be scaled differently in x than in y (or z, etc.), or perhaps rotated with respect to the x and y (etc) axes:







Notice that the contours of the paraboloid are ellipses (since the level curves of a paraboloid are ellipses) — this is why the confidence regions in linear problems with the L2 norm are ellipses.

Again often the model vector $\underline{m}$ is longer than just x and y and in that case this stuff just generalizes to higher dimensional equivalents. But this one I can draw!

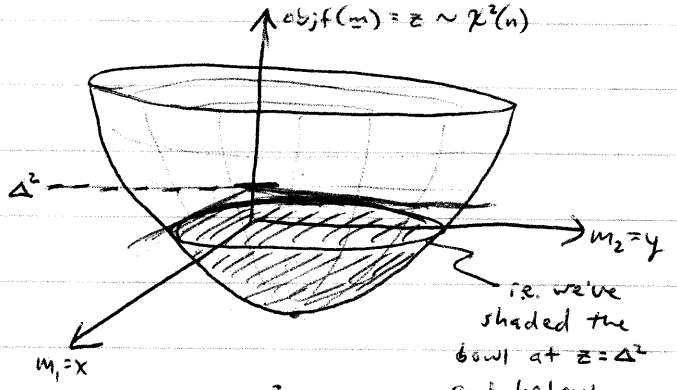
One more note about the error ellipses, and noise in the data. In class (and text) we saw that if we normalize the data noise covariance out of the data and linear forward problem matrix H, the transformed data $\hat{d}$ has noise with 0 mean and covariance $\underline{\underline{I}}$, so the data misfit (residual) norm is a chi-squared

→

distributed random variable, with n degrees of freedom where n is the number of data points.

$$\hat{\underline{d}} = \underline{\underline{C}}^{-1/2}\,\underline{d} \quad , \quad \hat{\underline{\underline{H}}} = \underline{\underline{C}}^{-1/2}\,\underline{\underline{H}}$$

$$objf(\underline{m}) = \left\| \hat{\underline{d}} - \hat{\underline{\underline{H}}}\,\underline{m} \right\|_2^2 \sim \chi^2(n)$$



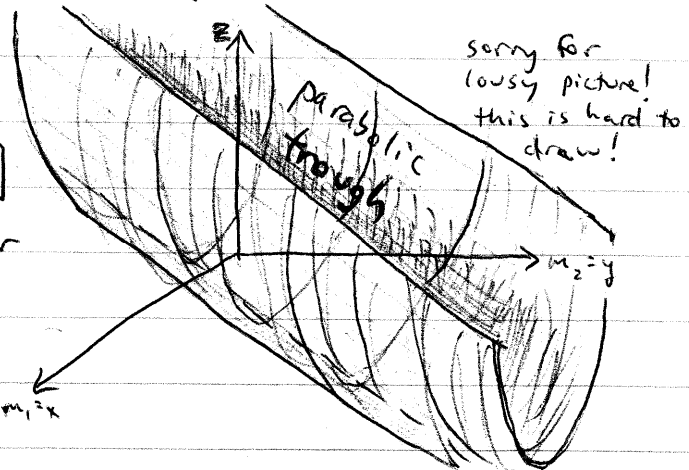
Now if we're interested in model estimates of x and y within a 95% confidence region, that corresponds to a particular value of our χ^2-distributed objf value, call it Δ^2.

the 95th percentile of $\chi^2(n)$, taken from a table

To be within the confidence interval then, we want to include all the x,y values on that bowl for $z \leq \Delta^2$. And that's the interior of that error ellipse. In the limit as the data noise covariance goes to 0, Δ^2 slides down the z axis and the error ellipse converges on the bottom point of the bowl, x^*, y^*, the least squares solution fit. That all works because we assumed back on page 3 that $\underline{\underline{H}}$ was not singular or ill-conditioned, that it was "nice."

recall that $(x^*, y^*) = \underline{m}^* = (\hat{\underline{\underline{H}}}^T \hat{\underline{\underline{H}}})^{-1} \hat{\underline{\underline{H}}}^T \hat{\underline{d}}$

But what if it's not "nice"? Say it's something like $\underline{\underline{H}} = \begin{bmatrix} -1 & 2 \\ -1 & 2 \end{bmatrix}$ (i.e. not full rank). Then our parabolic bowl degenerates into a parabolic trough.



Unlike the parabolic bowl which has one minimum point which corresponds to the unique LS solution, the parabolic trough comes from a problem which didn't have enough information to have one best model x,y point which fits the data — instead there are infinity of them along the bottom of the trough which equally fit the data in that rank-deficient example.

This simple example is an analogy to inverse problems, which by nature of trying to estimate a continuous model function from a finite set of data points is inherently underdetermined, on top of further ill-posedness that often comes with the problem itself.

How do we narrow the infinity of solutions along the bottom of the trough down to one? We regularize by adding an additional constraint such as the one mentioned earlier for model size $\|\underline{m}\|_2^2$:

$$objf(\underline{m}) = \|\underline{\hat{d}} - \underline{\underline{\hat{H}}}\,\underline{m}\|_2^2 + \nu\|\underline{m}\|_2^2 \quad , \quad \nu > 0$$

so a bowl again

terminology note: this form of regularization with model size $\|\underline{m}\|_2^2$ is also called "Ridge Regression".

This constraint doesn't rely on $\underline{\underline{H}}$ and brings $objf(\underline{m})$ back into a bowl/paraboloid. (Note objf is still a paraboloid with this constraint because it's essentially a sum of two paraboloids.) In this formulation now, the bottom of the new parabolic bowl is at:

$$(x^*, y^*) = \underline{m}^* = \left(\underline{\underline{\hat{H}}}^T \underline{\underline{\hat{H}}} + \nu\,\underline{\underline{I}}\right)^{-1} \underline{\underline{H}}^T \underline{\hat{d}}$$

Notice the tradeoff parameter ν that was in there as we've discussed in class regarding regularization, and we see that it affects the x^*, y^* solution as well. Just as earlier, for a given choice of ν we can construct a confidence ellipse around x^*, y^* based on $z = \Delta^2$. But note that changing ν changes the shape of the bowl making it steeper or shallower, so that changes the ellipse that came from the intersection of the bowl with the Δ^2 level. So not only does the choice of ν affect the solution point, it also affects its uncertainty. This matches what we heard in class that ν governs the trade-off point between uncertainty and resolution of the model we're estimating. (We're only considering x and y here for simplicity, so we don't consider the resolution part here.)

Back to nonlinear problems now...

The above discussion was all for linear problems, but the concepts in general still apply to nonlinear problems, regarding the objective surface (function). The catch in a nonlinear problem is that we no longer have that nice machinery to immediately and analytically tell us the location of the bottom of a parabola, and the shape of the confidence region associated with it, because we no longer have a parabola for the objective function in a nonlinear problem.

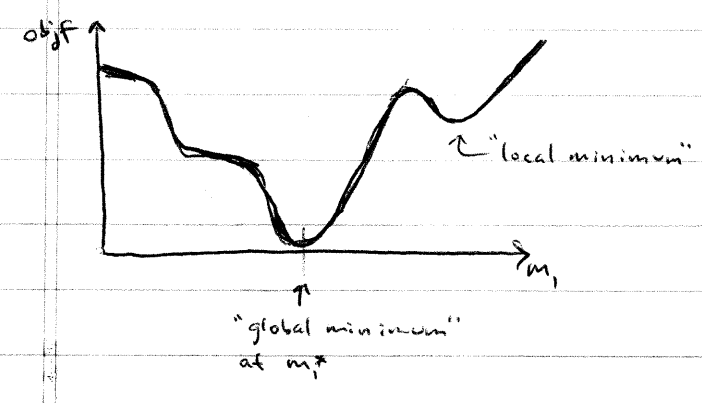
Since in a nonlinear problem the norms are no longer of the form $\|\underline{d} - \underline{\underline{G}}\underline{m}\|_2^2$ but instead now $\|\underline{d} - \underline{G}(\underline{m})\|_2^2$, we no longer have (necessarily) a quadratic expression for $\text{obj}f(\underline{m})$. It could now include powers higher than 2 in m_i, or trig functions or exponents of m_i, or whatever. This means the objective surface is no longer a nice parabolic bowl; we have a whole range of possible shapes from "almost paraboloid" to frightening jaggy surfaces. Depending on the shape of the objective surface, our nonlinear inverse problems (or parameter estimation problems) could be fairly easy to solve or close to impossible to solve.

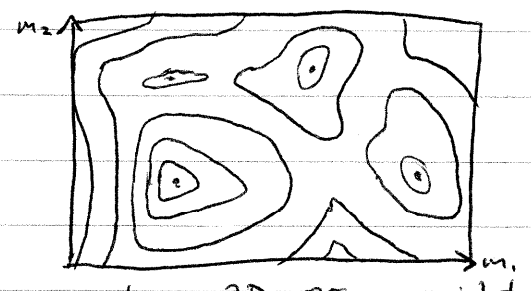
The relative smoothness vs. jagginess of the objective surface is an indicator of the degree of nonlinearity of a problem. A fairly smooth (although still not paraboloid) surface corresponds to a "weakly nonlinear" problem while a jaggy surface corresponds to a more "strongly nonlinear" problem. In either case, nonlinear problems lack the uniqueness in their solutions that linear problems have — in linear problems the bowl has one and only one minimum, corresponding to the unique solution. But in nonlinear problems the objective surface may have many minima, giving a solution that is both nonunique and confusing in its statistics.

If it's only a few minima as in a weakly nonlinear problem, it may not be too bad. Conveniently, some useful geophysical problems are only weakly nonlinear, including estimation of wave slownesses (1/velocities) as a function of location based on seismic wave travel times. The source location problem in this lab (and the previous one) is also weakly nonlinear. Recall in the previous lab that for some layouts of receiver stations, the solution converged to two different locations depending on the choice of initial estimate. When we plot the corresponding objective surface to that problem in this lab, we'll see the two distinct minima.

Now let's look at how our local linearization technique we learned last time works in terms of these objective surfaces. I'm going to mostly draw 1D models here due to my artistic limitations (and to try to show it clearly), but try to imagine the concepts applied to higher dimensional "surfaces".

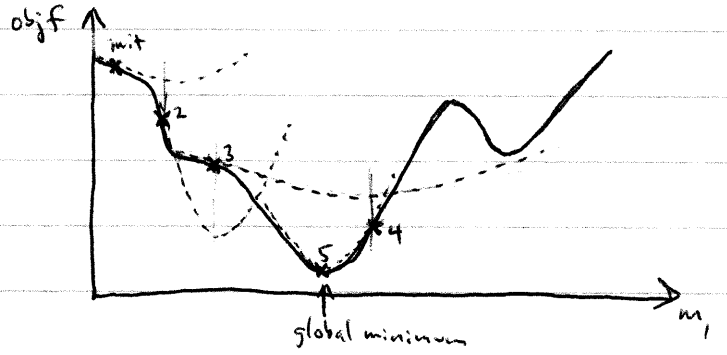
Say we have an objective surface like this:



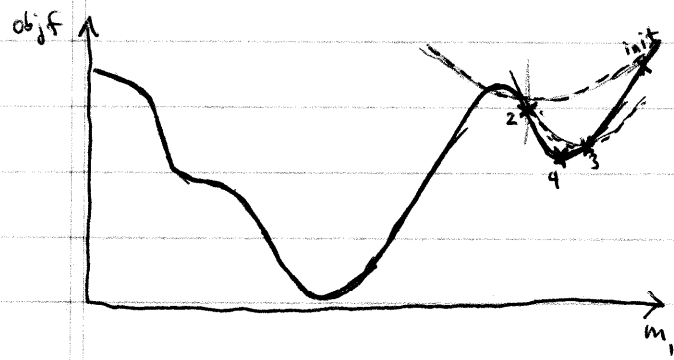


maybe a 2D version might look something like this in contours...

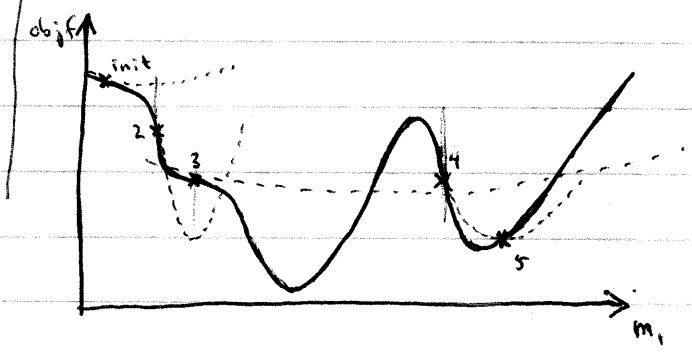
Our local linearization technique we learned about last time starts at an initial estimate of m_j and iterates to the global minimum in a series of steps. At each step we fit a parabola to that location m_j (based on the derivatives at m_j and the data points) and analytically jump to the location of that parabola's minimum. Usually this method takes you downhill toward the global minimum:



But we can see here how starting at a poor initial estimate can make you end up in a local minimum, as we saw last lab with the receiver stations in a line:



Or maybe there could even be some problem where you have just the right conditions to knock you over to the wrong local minimum...



So those are some of the issues it's helpful to be aware of when dealing with nonlinear problems, which an understanding of objective surfaces may help you analyze. For most problems you won't be able to look at that objective surface directly as in this lab, because to do this we must grid up the model space really finely and run the forward problem for each and every combination of model parameters on that grid and plot the resulting χ^2 value (or $\chi^2 + \alpha\|m\|_2^2$ value). Then you could just see where the minima are and the shape of the confidence regions around them. But for (most?) many problems that would take huge amounts of computation time that we don't have (although it would be the ideal solution!). So we try to get by with techniques that hope to find a solution cleverly with a minimum of computation.

However, the forward problem in this lab is so simple and quick that we <u>can</u> run it zillions of times, so that's why we use it here to explore plots of the objective surface. We're going to build upon the code we made for last week's lab, and what really want to see in this week's assignment is a comparison of the error ellipses we'll calculate at the solution point (we're going to add noise into the problem) with the shape of the objective surface plotted underneath. To clarify, typically in this "local linearization" technique we then locally linearize one final time at the solution point, and

from that compute a model estimate covariance matrix just as we did for linear problems. As in lab #2 we can compute confidence ellipses from this covariance matrix. However, unlike linear problems, here that's only an approximation to the shape of the confidence region, since the objective surface is not paraboloid. We justify its use though by saying that close to the solution point the objective surface is paraboloid (in the same sense that close to a local expansion point the problem can be approximated as linear). So if the uncertainties aren't too large then the ellipses should roughly match the objective surface, and if the uncertainties are large then we might expect a poor approximation of the statistics by the ellipses.

We'll augment last week's plots with an image or contour plot of the objective surface in the background, and two confidence ellipses. So do the following:

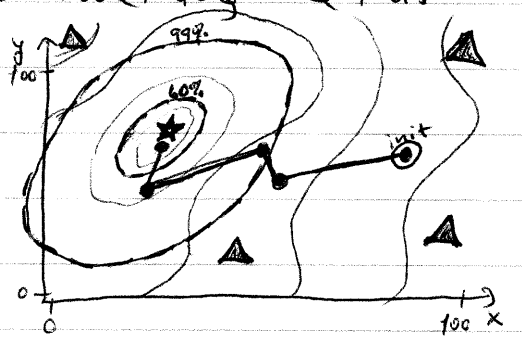
1.) take last week's script and add data noise ($\sim N(0, \underline{\underline{C}}_d)$) to the problem — pick some simple covariance like $\underline{\underline{C}}_d = \sigma_d^2 \underline{\underline{I}}$. Compute $\hat{sd}$ and $\hat{H}(\underline{m})$ every iteration (since $\underline{\underline{G}}(\underline{m})$ and $\underline{\underline{H}}$ will change with each new $\underline{m}$).

2.) at the final solution point found in your script, linearize one final time and calculate $\hat{H}$ once more to compute the covariance of $\underline{m}$ at the solution point. (call it $\underline{\underline{C}}_m$)

3.) From $\underline{\underline{C}}_m$ calculate the 60% and 99% confidence ellipses as done in lab #2.

4.) Calculate the objective surface for the problem (note computational hint below).

5.) Plot the objective surface, with seismometer stations, iteration path, initial and final location estimates on top. Add the 60% and 99% confidence intervals on top of that. (Be sure to check out the "imagesc", "contour", "axis xy", "axis ij", and "hold on" commands in Matlab). We want to see something like this:



6.) This should all be done up in a Matlab script so it's easy to change station locations and σ_d.

7.) Compare the shape of the ellipses to the shape of the objective surface underneath. Do they line up exactly? Does one ellipse line up at least a little better than the other? How does this change with a second choice of station locations or σ_d?

8.) Calculate the eigenvalues of the cov matrix $\underline{\underline{C}}_m$ at the solution estimate, and compare the eigenvalues to the size/shape of the error ellipses. Also, how's the eigenvalue for the unplotted source-time parameter compare to that of the locations?

Lastly, a computational hint to quickly produce the objective surface so you can concentrate on the rest. To account for the unplotted source time parameter t_s, let's remove its effect from the objective surface at each x_s, y_s point by subtracting off the mean of the data residuals. Here's Matlab code to produce the objective surface plot such that it lines up with last week's plot:

```
xmax = 100; xmin = 0; ymax = 100; ymin = 0;
for i = 1 : 100
    for j = 1 : 100
        xi(i) = i * (xmax - xmin)/100 + xmin;
        yj(j) = j * (ymax - ymin)/100 + ymin;
        t = fwdprob([xi(i), yj(j), m_true(3)]); % same fwdprob as last week...
        errors = (d - t) / sigmad;   % sigmad = data noise stdev
        errors = errors - mean(errors);
        obj(i,j) = dot(errors, errors);
    end
end
imagesc(xi, yj, obj');   % note transpose on obj...
axis xy;                 % and this command, to orient correctly
hold on;
% now add last week's plotting on top of obj surface...
```

note that m_true = [10,20,30]'; (or whatever values) is used to produce the travel times...

Hopefully this code here turns this week's assignment from what initially looked enormous to something manageable involving mixing code from labs #2 + #4 + above...