INVERSION **ASSIGNMENT 6**

NON-LINEAR LEAST SQUARES

Non-linear least squares can be done with either an iterated linear least squares, or a specialized inversion like simplex.

In iterated non-linear least squares you need to linearize the model about a first guess, solve for adjustments to the model, and then repeat the process with the new model. So if m_o is the initial guess for the model, and $f(m^{true}) = d$, we need to minimize

$$d^{obs} - f(m_o)$$

or the difference between the observed data and what the initial model would predict. Linearizing the model, we have

$$\begin{aligned}\frac{\partial f}{\partial m}\delta m + f(m_o) =& d + \epsilon \\ \frac{\partial f}{\partial m}\delta m =& d - f(m_o) + \epsilon\end{aligned}$$

where δm are adjustments to the initial model that result in a smaller data misfit. The partial wrt the model parameters is then the K matrix, which I will write with a prime as a reminder that this is the derivative of K.

$$K'\delta m = d^{obs} - f(m_o);$$

A single step in the sequence of iterations has thus been cast in the familiar form of a linear inverse problem except K has been replaced by K', and the observed data are now the observed data minus the data predictions based on the current estimate of the model.

$$K'_{i,j} = \frac{\partial f(m)}{\partial m_j}|m_o$$

1) Add to the program you have been writing so that it will do iterated nonlinear least squares.

In a previous assignment you looked at CO2 data from Mauna Loa. Re-visit that problem now and solve for the time constant. The model was:

$$Acos(2\pi t) + Bsin(2\pi t) + C + De^{\frac{t-t(1)}{\tau}}$$

where τ was set at reasonable guess. A cursory examination of the data suggests an initial guess for all parameters.

τ appears in this equation as a non-linear factor, so it cannot be evaluated in the same way as A, B, C, and D. Taking partial derivatives

$$\begin{aligned}
K'_{i,1} =& cos(2\pi t_i) \\
K'_{i,2} =& sin(2\pi t_i) \\
K'_{i,3} =& 1 \quad \text{all i} \\
K'_{i,4} =& exp^{\frac{t-t(1)}{\tau_o}} \\
K'_{i,5} =& -D_o exp^{\frac{t-t(1)}{\tau_o}} \frac{t-t(1)}{\tau_o^2}
\end{aligned}$$

and each i refers to a different time.

So the problem to solve in the first iteration is:

$$K' \begin{bmatrix} \delta A \\ \delta B \\ \delta C \\ \delta D \\ \delta\tau \end{bmatrix} = d^{obs} - f(m_o)$$

and out of this you get corrections to your first estimate of the five model parameters, which you add to the model parameters to form a guess for the second iteration.

When should you quit iterating? There are three considerations that guide you here:

1) when the standard deviation of the data misfit approaches the known standard deviation of the observations,
2) when the standard deviation of the data misfit reaches a level beyond which further iterations produce no further improvement,
3) when the model parameters have converged and show no further changes with succeding iterations. You could insert an automatic stop based on these criteria, or preferably a pause with a query to continue or quit.

Try starting with ever more outrageous initial guesses for τ to test the limits of convergence.

We also saw earlier that some of these parameters could be combined in a linear equality constraint. Add this linear equality constraint and solve again.

The simplex algorithm is implemented in *FMINSEARCH* in matlab. *FMINSEARCH* operates with N+1 models, where N is the number of parameters, and at each step rejects the model that returns the largest data misfit and proposes a replacement model to be used in the next iteration based on the remaining models. The big advantage over iterated Gauss-Newton is that you do not need to calculate the derivatives, but the execution time is generally longer and increases drastically with the number of model parameters.

Solve the above problem again using *FMINSEARCH.*

In G-N inversion, or any of the variations of G-N we have looked at, the estimated error on the model parameters was:

$$\Delta m^{mest} = (K^t K)^{-1} K^t \delta d^{obs}$$

With the Simplex algorithm there is no such straightforward way to calculate error estimates on parameters. Error estimates can still be obtained, but with a little more work, through a Monte Carlo simulation. If you know the standard deviation of each observation then add a random number with zero mean and this standard deviation to every observation, thus generating an observed sequence with the same statistical properties as the original data. *Each sample in this new sequence could have been the result of the same measurement process.* Now go through the inversion procedure again generating a new set of model parameters which are the best fit to the new data. This procedure is repeated until sufficient trials have been executed that a statistically meaningful standard deviation of model parameters can be estimated. Ten or twenty trials should be enough to estimate the parameter errors.

If you do not know the error on each observation, then you can estimate it from the residuals of the observed data, that is, the standard deviation of

$$d^{obs} - d^{pre}$$

is an estimate of the standard deviation on the observations. How good an estimate it is depends on how well your model predicts all the variance in observed data. If the residuals display any trend, or do not look random, then you will over-estimate the observation errors, and over-estimate the parameter error as well. One very important test is to do a histogram of these residuals. Does it look Gaussian?

Lastly, Implement the linear equality constraints in simplex.