# Lab #6. Nonlinear inversion

In this lab, you will practice two types of iterative inversion of nonlinear problems: the Gauss-Newton inversion and the simplex method. The application example will be the same Mauna Loa $CO_2$ dataset from lab 3. The forward model is (in the approximate sense I talk about in the first sections of the text)

$$d_i \approx A\cos(2\pi t) + B\sin(2\pi t) + C + De^{\frac{t_i - t_1}{\tau}}, \quad (1)$$

where $t_i$ is the time in years, $d_i$ is the $CO_2$ ppm value, and $A$, $B$, $C$, and $D$, and $\tau$ are the unknowns. Parameters $A$, $B$, $C$, and $D$ enter eq. (1) linearly, but for $\tau$, the equation is nonlinear, and this is why we need to use iterative inverse methods.

However, even when using nonlinear models, you need to consider how to define the model vector **m**. Note that instead of $\tau$ in eq. (1), it is easier and better to use parameter $\kappa = 1/\tau$ and form the model vector as $\mathbf{m} = \begin{pmatrix} A & B & C & D & \kappa \end{pmatrix}^T$. Equation (1) is then $\mathbf{d} \approx \mathbf{d}^{\text{pre}}(\mathbf{m})$, where:

$$d_i^{\text{pre}}(\mathbf{m}) = m_1\cos(2\pi t) + m_2\sin(2\pi t) + m_3 + m_4 e^{m_5(t_i - t_1)}, \quad (2)$$

and the function which we want to minimize is $\Phi(\mathbf{m}) = \sum_i \left[ d_i - d_i^{\text{pre}}(\mathbf{m}) \right]^2$.

For linear and particularly for nonlinear methods to work well numerically, it is important to properly precondition the forward problem. As a minimum, you need to ensure that all parameters $m_i$ have comparable impacts on the data (values of the Jacobian matrix **J**). From eq. (2), note that parameters $m_1$ to $m_4$ are measured in ppm, and the ranges of their variation (variances) are around ~2 ppm (see lab 3). By contrast, parameter $m_5$ is measured in 1/years, and its expected range of variation is roughly 1/(60 years). As a result, numerical values of $m_5$ are much smaller than those of $m_1$ to $m_4$, and the inversion may have hard time finding $m_5$ concurrently with other parameters. Therefore, for a better-conditioned inverse problem, you would need to rescale parameter $m_5$ so that its expected value would be around one. This can be done by parameterizing eq. (2), for example, like this:

$$d_i^{\text{pre}}(\mathbf{m}) = m_1\cos(2\pi t) + m_2\sin(2\pi t) + m_3 + m_4 e^{m_5 \cdot 0.01(t_i - t_1)}. \quad (2a)$$

This scaling can be interpreted as measuring $m_5$ in units of 1/(100 years), which is more natural for this model.

To minimize function $\Phi(\mathbf{m})$, you can start from some model $\mathbf{m}^0$ and linearly approximate $\mathbf{d}^{\text{pre}}\left(\mathbf{m}^0+\delta\mathbf{m}\right)\approx\mathbf{d}^{\text{pre}}\left(\mathbf{m}^0\right)+\mathbf{J}\delta\mathbf{m}$, where **J** (the "Jacobian" matrix) consists of the derivatives of each data point $d_i^{\text{pre}}$ with respect to each model parameter $m_j$: $J_{ij}=\dfrac{\partial d_i^{\text{pre}}}{\partial m_j}$. Alternatively, you can use nonlinear search methods like simplex, which require no calculation of the Jacobian. However, in all cases, it is important to estimate a starting model $\mathbf{m}_0$ which is reasonably close to the solution.

Thus, the steps of the lab are like this:

**Step 1**: calculate the Jacobian analytically (verify this):

$$\text{for any } i: \quad \begin{cases} J_{i1}=\cos\left(2\pi t\right), \\ J_{i2}=\sin\left(2\pi t\right), \\ J_{i3}=1, \\ J_{i4}=e^{m_5\left(t_i-t_1\right)}, \\ J_{i5}=0.01m_4\left(t_i-t_1\right)e^{0.01m_5\left(t_i-t_1\right)}. \end{cases} \tag{3}$$

The function being minimized is thus replaced with $\Phi\left(\mathbf{m}\right)=\sum_i\left[d_i-d_i^{\text{pre}}\left(\mathbf{m}^0\right)-\sum_j J_{ij}\delta m_j\right]^2$ which is the objective function of the least-squares problem for $\delta\mathbf{m}$:

$$\mathbf{J}\delta\mathbf{m}=\delta\mathbf{d}, \tag{4}$$

where $\delta\mathbf{d}=\mathbf{d}-\mathbf{d}^{\text{pre}}\left(\mathbf{m}^0\right)$ is the data misfit vector for your current iteration. This is often (and in this lab) an *over-determined* inverse problem for $d\mathbf{m}$. In the Gauss-Newton iterative inverse, $\delta\mathbf{m}$ is found from eq. (4) by using the least-squares method:

$$\delta\mathbf{m}=\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\delta\mathbf{d}\ . \tag{5}$$

Note that sometimes, the problem (4) might be under-determined, and then you would probably use the minimum-length inverse instead of (5).

Also note in eq. (5) that matrix $\mathbf{J}^T\mathbf{J} = \mathbf{H}$ is the "approximate Hessian" (matrix of second derivatives of the approximate $\Phi(\mathbf{m})$ with respect to the model: $H_{ij} = \dfrac{\partial^2 \Phi(\mathbf{m})}{\partial m_i \partial m_j}$, and $\mathbf{J}^{\mathrm{T}}\delta\mathbf{d}$ equals minus the gradient of $\Phi(\mathbf{m})$: $\mathbf{g} = \dfrac{\partial \Phi}{\partial \mathbf{m}} = -\mathbf{J}^T\left(\mathbf{d} - \mathbf{d}^{\mathrm{pre}}\right)$. Thus, the Gauss-Newton inverse (eq. 5) can also be written as $\delta\mathbf{m} = -\mathbf{H}^{-1}\mathbf{g}$.

**Step 2:** Add to the program you have been writing code to so perform iterated nonlinear least squares by the Gauss-Newton method, Use either the (**J**, $\delta$**d**) or (**H**, **g**) notation above.

**Step 3**: Starting from some reasonable $\mathbf{m}^0$ not far from he one found in lab 3, perform iterations to obtain solution of the nonlinear inverse problem. Plot the path of these iterations on the plane of parameters ($m_4$, $m_5$) (or also any other).

When should you stop iterating? Three considerations can guide you in this decision:

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 succeeding iterations.

You could insert an automatic stop based on these criteria, or preferably a pause with a query to continue or quit.

**Step 4:** Try starting with ever more outrageous initial guesses for $m_5 = \kappa$ (or $\tau$) to test the limits of convergence. Note that if you are using $\tau$ as the nonlinear parameter, then even very large values of $\tau$ represent *small* variations of the actually essential parameter $\kappa$ from zero. With $\kappa$, you can start iterations even from values $\kappa < 0$ and see how the method converges.

Next, try applying additional linear constraints when iteratively solving for a nonlinear inverse. For this type of model, only exact linear constraints make sense, because, for example. there are no parameters for which we could require smoothness. Let us impose a constraint requiring that the first data point in the dataset is known exactly: $d_1^{\text{pre}}(\mathbf{m}) = d_1$. From eqs. (2) or (2a), this constraint reads $d_1^{\text{pre}}(\mathbf{m}) = m_1\cos(2\pi t_1) + m_2\sin(2\pi t_1) + m_3 + m_4 = d_1$. Write it in matrix form **Bm** = **c** (section 7.1 in the notes and lab 5) and verify that:

$$\mathbf{B} = \begin{bmatrix} \cos(2\pi t_1) & \sin(2\pi t_1) & 1 & 1 & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{c} = d_1 . \tag{6}$$

For the model increment $\delta\mathbf{m}$, this equation becomes $\mathbf{B}\delta\mathbf{m} = d_1 - d_1^{\text{pre}}(\mathbf{m})$, where $d^{\text{pre}}(\mathbf{m})$ is the data predicted by the preceding iteration. Before starting the iterations using this equation, it is better to ensure that the starting model is close to satisfying the desired constraint **Bm** = $d_1$. This can be done by various selections of parameters **m**. For example, set all parameters $m_i$ as you found in lab 3, and then adjust the additive parameter $m_3$ to satisfy $d_1^{\text{pre}}(\mathbf{m}) = d_1$.

**Step 5:** Add the exact linear constraint $\mathbf{B}\delta\mathbf{m} = \mathbf{0}$ to eq. (5), similar to what was done in the last question of lab 5. Perform the iterative inverse and verify that $d_1^{\text{pre}}(\mathbf{m}) = d_1$ after each iteration. Plot the convergence path on the ($m_4$, $m_5$) plane.

In the remainder of the lab, examine the simplex algorithm on the same problem, using function `fminsearch` in Matlab or Octave. For a model with $N$ parameters (five in this case). this algorithm operates with $N$+1 trial models. At each step of iteration, `fminsearch` rejects the model that returns the largest data misfit, and based on the remaining $N$ models, proposes a replacement model with lower data misfit. The big advantage over the iterated Gauss-Newton method is that you do not need to calculate any derivatives and solve any least-squares or minimum-length problems. However, the execution time is generally longer and increases drastically with the number of model parameters.

**Step 6:** Solve the problem in eq. (2a) again using `fminsearch`.

In Gauss-Newton or any other variations of linearized inverses, the error of the inverted model parameters can be estimated by using data and model covariance matrices (see lab 4 and class notes). With the simplex algorithm, there is no such straightforward way to calculate the model error estimates. However, error estimates can still be obtained by a generating synthetic data, or by data bootstrapping (see class notes). In this lab, let us try straightforward Monte Carlo simulation of data errors.

The idea of Monte Carlo parameter-error testing consists in generating several sets of synthetic data **d***:

$$d_i^* = d_i^{\text{pre}}\left(\mathbf{m}^{\text{best}}\right) + \varepsilon_i, \tag{7}$$

where $\mathbf{m}^{\text{best}}$ is your best-fit model, and $\varepsilon_i$ is a random error drawn from the distribution of measurement errors. By using these **d*** as data, you would then repeat the inversion by any method, and the scatter of the resulting models $(\mathbf{m^*})^{\text{best}}$ would show the model-parameter uncertainty, and also the covariances between the different parameters.

**Step 7**. From your lab 3, find the standard deviation of $CO_2$ ppm data errors $\sigma$. Then, go through the simplex inversion procedure several times, each time generating a new Gaussian distribution with zero mean and variance $\sigma$. Add these errors to the best-fit model predictions as in eq. (7). Using these randomized data, produce inverses by the `fminsearch` algorithm.

Ten or twenty trials above should be enough to estimate the model parameter errors.

If you do not want to trust the results of lab 3 for determining $\sigma$ (might be a good idea), then you can repeat the analysis of data fitting from there. Plot the residuals of data errors $d_i - d_i^{\text{pre}}\left(\mathbf{m}^{\text{best}}\right)$ versus time of the observation, check whether there are any systematic trends, and then plot a histogram. Does it look Gaussian? Calculate the standard deviation $\sigma$.

**Step 8**. Lastly, implement the linear equality constraints on the first data point $d_1^{\text{pre}}(\mathbf{m}) = d_1$ in the simplex method. This constraint is again given as **Bm** = **c** (eq. 6).

As you will find, *exact* constraints cannot be included in the simplex method, because it does not allow restricting the possible values of arguments **m**. However, you can apply constraints in a "soft" way by adding them to the objective function (written in matrix form here):

$$\Phi(\mathbf{m}) = \frac{1}{2}\left[\mathbf{d} - \mathbf{d}^{\text{pre}}(\mathbf{m})\right]^T \left[\mathbf{d} - \mathbf{d}^{\text{pre}}(\mathbf{m})\right] + \frac{\beta}{2}(\mathbf{c} - \mathbf{Bm})^T (\mathbf{c} - \mathbf{Bm}). \quad (8)$$

As usual, some thought (and testing) needs to be given to the selection of parameter $\beta$.