Geol 483.3

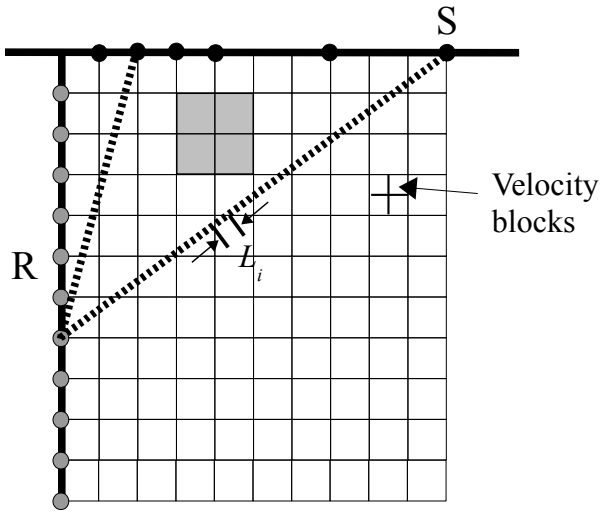
Lab Project #1: Offset VSP travel-time modelling and tomography

This project consists of two parts. First, you will set up a simplified synthetic cross-well model and use Matlab to compute the first-arrival travel times in it. After this, you will use these synthetic travel times to invert for seismic velocities in a procedure called seismic tomography.

VSP model (30% of the mark)

In offset VSP imaging, a string of geophones is installed in the borehole, or a tool is raised through the borehole for each source position.. In travel-time imaging, first-arrival source-receiver travel times are recorded for each position of the sources and receivers. These times are utilized to invert for the velocities.

Consider a simplified experiment geometry shown below. 11 receivers are located within the borehole at 20-, 40-, 60-, ..., and 220-m depths. Six shots are fired: four at distances 20, 40, 60, and 80 m, and another two at 140 and 200 m from the borehole head. The velocity model is gridded into 10×10-m blocks, so that as each receiver-spacing size cell contains four velocity blocks:



To show that shots do not have to be located within the model, the last shot is outside the modelled/inverted area.

Note that even before you start, you should expect that the lower-right part of the model will not be constrained by the travel-time data. Therefore, your inversion will be under-constrained and will require regularization. The ray coverage is also poorer within the upper-right part of the model.

The velocity within each square cell is constant, and all rays are assumed straight, as shown in the plot above. Model velocities need to be provided in an array, and you need to compute 121 travel times from each source to each receiver. To achieve this, you will need:

1) Write a Matlab subroutine to initialize the velocity array. Note that velocities need to be stored in a ONE-dimensional array, so some numbering convention (e. g., by scanning the grid) is advisable. First, set the velocities equal 2 m/ms everywhere (use meters to measure distances and milliseconds for times).
2) Write a Matlab function to return, for a given source and receiver, the lengths of the segments of the corresponding ray (L_i in the plot above) in each of the model cells. For most cells (not crossed by the ray) these lengths will be 0.
3) Using the lengths of ray segments, compute the total ray travel time for each ray. Plot the travel times (as functions of receiver depth) for source positions at the left, middle, and right edges of the source spread.

Now apply a +10% velocity anomaly within the shaded area in the plot and repeat the steps above. How different are the travel times? Why?

4) Save the resulting travel times. Again, for the inversion below, it is better to store all of the resulting travel times in a single array of 121 values. You can put into the array, say, first all times from the shot at depth 0, followed by all travel-times from shot at 20 m, and so on.
   - Create and save another travel-time dataset, for velocity anomaly shifted into the upper-left corner of the model.

Tomography (40%):

Once you have all of your velocity model values in an array $\{V_i\}$ and travel times in array $\{t_j\}$, the inversion proceeds as follows. Given some velocity distribution, predicting travel times are:

$$t_i = \sum_k L_{ik} p_k , \qquad (1)$$

where k is the index of model cell, L_{ik} is the ray path of i-th ray in this cell, and $p_k = 1/V_k$ is its "slowness". In matrix form:

$$\begin{pmatrix} t_1 \\ t_2 \\ \dots \\ t_{121} \end{pmatrix} = (L) \begin{pmatrix} p_1 \\ p_2 \\ \dots \\ p_{100} \end{pmatrix},$$

where **L** now is a 100×121 matrix. It is not directly invertible (there are more equations than unknowns), yet it can be inverted in the Least Squares sense by multiplying the above equation by the transposed matrix $\mathbf{L}^T$:

$$L^T \begin{pmatrix} t_1 \\ t_2 \\ \dots \\ t_{121} \end{pmatrix} = L^T L \begin{pmatrix} p_1 \\ p_2 \\ \dots \\ p_{100} \end{pmatrix},$$

and therefore,

$$\begin{pmatrix} p_1 \\ p_2 \\ \dots \\ p_{100} \end{pmatrix} = (L^T L)^{-1} L^T \begin{pmatrix} t_1 \\ t_2 \\ \dots \\ t_{121} \end{pmatrix}. \qquad (2)$$

This matrix equation is readily programmable in Matlab or Octave. For each of the two travel-time datasets you created:

6) Look at the matrix $\mathbf{L}^T\mathbf{L}$ (print it out in Matlab), and try computing its inverse. Comment why the inverse fails. What properties of the matrix and of the model cause this failure?
   To accomplish approximate inversion, you need to *regularize* the inverse (2) by replacing it with:

$$\begin{pmatrix} p_1 \\ p_2 \\ \dots \\ p_{100} \end{pmatrix} = (L^T L + \epsilon I)^{-1} L^T \begin{pmatrix} t_1 \\ t_2 \\ \dots \\ t_{121} \end{pmatrix}. \qquad (3)$$

   where ε is some small number. Try using ε equal ~0.01 of the typical value of the diagonal values of $\mathbf{L}^T\mathbf{L}$.

7) Implement the regularized (also called "*damped*") matrix inversion (3),
8) Plot horizontal and vertical velocity cross-sections across the anomaly and compare to the actual (10-%) anomaly.

9) Discuss the difference in the shapes of the anomalies recovered by the inversion in the two cases. Which anomaly is recovered better, at the center or corner of the model? What could be the reason for this?
10) Use Matlab to plot the resulting 2-D velocity distributions in (X,Z) grey-shade plot similar to the one above.
11) Plot the diagonal of the resolution matrix in grey scale. Describe its appearance.
12) Plot (in grey scale, again) the middle row (the one corresponding to one of the shaded cells in the plot above) of the resolution matrix.

Checkerboard resolution test (30%):

Calculation of the resolution matrix above is often impossible or impractical. For example, such would be the case if we iterated the inversion above in order to account for non-uniform velocity and bending rays. N such cases, regardless of the way inversion is performed, resolution of the model can be measured by conducting a so-called checkerboard test.

12) To perform the test, create a synthetic model with positive and negative ±10% anomalies alternating in a checkerboard pattern. The anomalies can consist of a single cell or be 2×2, 3×3, etc., cell blocks.
13) Predict the travel times through the model using matrix equations (1).
14) Invert these synthetic travel times. The resulting inverted model should correspond to the original checkerboard model. Are all the anomalies equally well recovered? Why? How does this change if you increase the sizes of the input anomalies?

Hand in:

Codes, report in a Word file, screen captures or PostScript/PDF plots.