CHAPTER 5

Some extra notes on body waves and mantle tomography

1. The travel time inverse problem

In ray theory, the travel time of the ray is

$$T = \int_{\Gamma} \frac{1}{v}\, d\Gamma \tag{1}$$

where the integral is taken along the ray path and v is the seismic velocity. In global tomography, we usually start from a 1-D spherically symmetric Earth. Fermat's principle states that a ray path between two points is a path of stationary time. Thus the travel time will not change to first order if the ray path is slightly perturbed. If we make a small perturbation in velocity structure there will be a change in travel time due to the change in velocity structure *and* due to the change in the ray path but the latter term is of second order and so can be neglected. We can therefore differentiate equation 1 giving:

$$\delta T = -\int_{\Gamma} \frac{1}{v}\frac{\delta v}{v}\, d\Gamma \tag{2}$$

where we can use the spherical Earth ray path and v can be taken to be the 1-D velocity. More generally, we can write

$$\delta T = \int_{V} K(r,\theta,\phi)\delta m(r,\theta,\phi)\, dV \tag{3}$$

where δm is our model – in our case, the relative perturbation in velocity $\delta v/v$, and K is some kernel which in ray theory is just a delta function along the ray, but in more sophisticated theories which take into account finite frequency effects, will occupy a volume around the geometrical ray.

2. Long- and short-period travel times

Historically, seismograms were recorded either at "long" periods or "short" periods. The reason for this is that a major source of motion of the ground is the "microseisms" which are due to nonlinear interactions of ocean waves causing pressure variations on the ocean floor. Microseisms have a main peak at 14 second period and a secondary peak at 7 seconds. It is actually the secondary peak that is mainly seen on seismometers. In the past, seismic recording systems did not have the dynamic range to record both the microseisms and the small seismic signals which ride on them. Thus instruments were designed to see periods shorter than 7 seconds (usually peak response at about 1 second) or periods longer than about 15 seconds. Modern seismic recording systems have enough dynamic range to be able to record the microseisms (so-called broad-band recording) but, for most earthquakes, we must still filter out the microseisms so we can see the small seismic signals.

On the short period side, body waves of dominant period 1 second are seen and the first arriving P wave can be accurately picked. Scattering by short-wavelength heterogeneity causes large "codas" which can obscure secondary arrivals. Many observations have been made and are collected by the International Seismological Commission (ISC) which has used them to make a more comprehensive tabulation of earthquake locations. Such data are also used in tomography. P wave tomography using the ISC data has been quite successful but the S waves are more problematic. This is because S waves typically have a lower frequency content due to attenuation and are more poorly recorded by short period instruments. Furthermore, ISC picks are usually made from vertical component instruments so interference of S by the SKS phase at distances beyond 80 degrees is a problem. This makes it very difficult to image S velocity in the lowermost mantle from ISC S picks.

Long-period data offer some advantages over the ISC data – in particular, codas from scattering are nearly non-existant so later phases can be accurately picked.

3. Parameterization.

Consider a linearized inverse problem of the form:

$$d_i \pm \sigma_i = \int_0^R G_i \delta m \, dr \tag{4}$$

where d_i is a datum such as the difference between an observed frequency of free oscillation of the earth and one calculated for a starting model, G_i is some continuous kernel which we can compute and δm is a continuous model perturbation. When we have relatively few data, it is possible to avoid parameterization of the model and make an expansion of the form:

$$\delta m = \sum_{i=1,N} a_i G_i(r) \tag{5}$$

where N is the number of data. Inserting this into equation 4 gives

$$\mathbf{d} = \mathbf{\Gamma} \cdot \mathbf{a} \quad \text{where} \quad \Gamma_{ij} = \int_0^R G_i G_j \, dr \tag{6}$$

and $\mathbf{\Gamma}$ is a matrix which is of dimension $N \times N$. Equation 6 can be solved in a variety of ways (e.g., we can impose smoothness constraints on the model perturbation or on the total model) and we can explore the trade off with fit to the data. And we can look at the ability of our data to resolve features of our models in a quantitative way.

Unfortunately, once N exceeds a few thousand, the computational burden of dealing with huge matrices becomes too great. The conventional way around this is to parameterize the model by expanding it in a set of basis functions where the number of parameters is chosen to be computationally manageable:

$$\delta m = \sum_{i=1,M} a_i f_i(r) \tag{7}$$

where M is the number of parameters. Of course, in 3D tomography, the basis functions f are functions of r, θ, and ϕ. Substituting 7 into equation 4 gives

$$\mathbf{d} = \mathbf{A} \cdot \mathbf{a} \quad \text{where} \quad A_{ij} = \int_0^R G_i f_j \, dr \tag{8}$$

and $\mathbf{A}$ is a matrix which is of dimension $N \times M$.

The choice of basis functions in equation 7 can impact the kinds of models we can recover and can also impact the computational difficulty of solving equation 8. In global tomography, the choice of basis functions has, for many years, involved the use of spherical harmonics for parameterizing lateral variations:

$$\delta m = \sum_{s,t} \delta m_s^t(r) Y_s^t(\theta, \phi) \tag{9}$$

where the radial expansion coefficients $\delta m_s^t(r)$ are further parameterized either in global functions (e.g. Legendre polynomials or Chebychef polynomials) or as local functions (e.g. layers or B-splines. The reason for the choice of spherical harmonics is that these are efficient for parameterizing the long-wavelength structure which dominates many of the seismic datasets. Unfortunately, a consequence of using global bases is that every datum effectively becomes sensitive to the entire model so that the matrix $\mathbf{A}$ is very dense. This means that global models using spherical harmonics are typically limited to about 10,000 model parameters – if we divide the mantle up into roughly 20 layers (see below), each layer could have about 500 parameters. A spherical harmonic expansion up to degree l has $(l+1)^2$ expansion coefficients so this means l is limited to about 21. If we recall that

$$ka = l + \tfrac{1}{2} = \frac{2\pi a}{\lambda} \tag{10}$$

where k is wavenumber, λ is wavelength, and $2\pi a$ is the circumference of the earth (about 40,000 km), we find that the minimum wavelength we can capture is about 2000 km. Unfortunately, dynamically interesting structures such as slabs typically have much smaller dimensions than this (and many of our data are, in principle, sensitive to small wavelength structure). This has motivated the use of local bases in global tomography (e.g. blocks of uniform lateral dimension, equal area blocks, non-uniform distribution of blocks mimicking data sampling, tesselations, etc).

Why are local bases so useful? Let us suppose we have several hundred thousand travel time measurements. To a fairly good approximation, ray theory can be used to interpret such data so each datum is sensitive to only a small fraction of the total number of parameters in the model (i.e. along a particular ray). For example, using blocks of lateral dimension 4 degrees at the equator (this corresponds to a surface wavelength of about 880km or an l of about 45 if we had done a spherical harmonic expansion) gives roughly 2500 blocks per layer for a total of 50,000 model parameters for a 20 layer model. However, each datum samples only about 1% or less of the blocks so each row of the matrix $\mathbf{A}$ will have less then 500 non-zero entries. Sparse matrix techniques can then be efficiently used to solve equation 8.

4. Computing matrix elements for travel times using ray theory

Consider a ray through the Earth as shown in Figure 1.

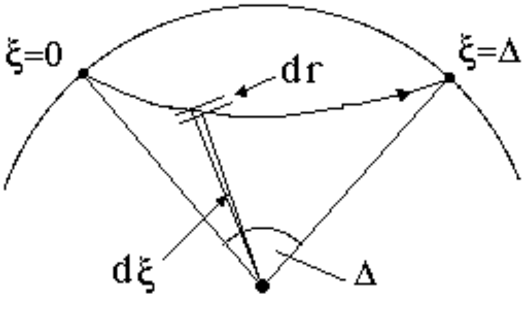

Fig 1

Now focus on the small segment of the ray which subtends the angle $d\xi$ at the center of the Earth (Fig. 2).

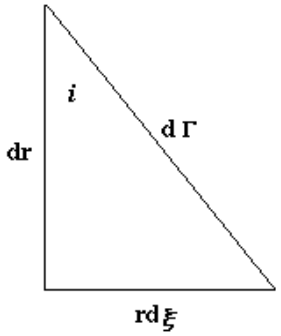

Fig 2

i is the angle the ray makes with the vertical and we know that the ray parameter is related to i by

$$p = \frac{r}{v} \sin i \tag{11}$$

where v is the velocity for the ray segment at radius r. Consider equation (2):

$$\delta T = -\int\limits_{\Gamma} \frac{1}{v} \frac{\delta v}{v} \, d\Gamma$$

From figure 2, we have

$$\sin i = \frac{r d\xi}{d\Gamma} \quad \text{so} \quad d\Gamma = \frac{r^2}{pv} d\xi \tag{12}$$

and we can rewrite equation (2) as an integral over distance:

$$\delta T = \int\limits_{0}^{\Delta} G(\xi) \frac{\delta v(\xi)}{v} \, d\xi \tag{13}$$

where

$$G(\xi) = -\frac{r^2}{pv^2} \tag{14}$$

This kernel is evaluated by keeping track of the depth of the ray for every position of arc length ξ. To do this we need an equation relating ξ to r. Reconsider Figure 2 and note that (using the equation 11 for p)

$$r\frac{d\xi}{dr} = \tan i = \frac{\frac{vp}{r}}{\left(1 - (\frac{vp}{r})^2\right)^{\frac{1}{2}}} \tag{15}$$

or

$$\frac{d\xi}{dr} = \frac{p}{r} \left(\frac{r^2}{v^2} - p^2 \right)^{-\frac{1}{2}} \tag{16}$$

On an aspherical Earth where we have used a local block parameterization, we step finely along in distance starting from a specific source position to a specific receiver position. At each point, we compute the radius we are at using equation 16 and then evaluate the kernel using equations 13 and 14. We also keep track of which block we are in at each step along the ray then integrate the contributions to each block at the end.

So far, we have been considering the effect of a "volume perturbation" in velocity. There may also be perturbations in the levels of discontinuities which, if the velocity is different on either side, produce travel time anomalies. The formula for δT for a transmitted ray if a boundary is moved by δr is

$$\delta T = -\frac{\delta r}{r}\left[\left(\frac{r^2}{v^2} - p^2\right)^{\frac{1}{2}}\right]_-^+ \tag{17}$$

where $[f]_-^+$ indicates the value of f below subtracted from the value of f above the discontinuity. For a reflected ray, we get

$$\delta T = -\frac{2\delta r}{r}\left(\frac{r^2}{v^2} - p^2\right)^{\frac{1}{2}} \tag{18}$$

for a topside reflection (so v is the velocity just above the discontinuity) and

$$\delta T = \frac{2\delta r}{r}\left(\frac{r^2}{v^2} - p^2\right)^{\frac{1}{2}} \tag{19}$$

for a bottomside reflection (so v is the velocity just below the discontinuity).

5. Finding a model

Let us suppose we are solving the problem

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{d} \tag{20}$$

for the vector $\mathbf{x}$. Further, we shall assume that we have divided each row of this system of equations by the observation error on the datum so that the data vector $\mathbf{d}$ has a covariance matrix which is just $\mathbf{I}$ (i.e. we are assuming our data are statistically independent from each other. If our system of equations (20) were well-conditioned, we might just find the least-squares solution:

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d} \tag{21}$$

which minimizes $(\mathbf{A} \cdot \hat{\mathbf{x}} - \mathbf{d})^2$. In reality, $\mathbf{A}$ is usually not well-conditioned and $\mathbf{A}^T\mathbf{A}$ is even worse (the condition number is effectively squared) so the solution (21) is rarely chosen. One way around squaring the condition number is to use a singular value decomposition (SVD) on equation 20. The matrix $\mathbf{A}$ is decomposed into singular values and matrices of left and right eigenvectors (here, we assume $M < N$):

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^T \tag{22}$$

where $\mathbf{U}$ has dimension $N \times M$ and $\mathbf{V}$ has dimension $M \times M$ and $\boldsymbol{\Lambda}$ is a $M \times M$ with non-zero diagonal elements. Note that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. The least-squares solution in terms of the SVD is

$$\hat{\mathbf{x}} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T\mathbf{d} = \mathbf{A}^+\mathbf{d} \tag{23}$$

where $\mathbf{A}^+$ can be thought of as the (generalized) inverse of $\mathbf{A}$. If $\mathbf{A}$ is not well-conditioned, it will have some small singular values which will generally lead to some poorly determined contributions to $\hat{\mathbf{x}}$. To see why this is so, consider the covariance matrix of the model. To get the model we are taking a linear combination of data: $\mathbf{A}^+\mathbf{d}$. Now $\mathbf{d}$ has covariance matrix $\mathbf{I}$ so $\hat{\mathbf{x}}$ has covariance matrix:

$$\mathbf{A}^+\mathbf{I}(\mathbf{A}^+)^T = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T\mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Lambda}^{-2}\mathbf{V}^T \tag{24}$$

The square roots of the diagonal elements of this matrix are the errors on our model parameters. Clearly, small singular values are going to make these errors large. One way to avoid this is to exclude small singular values from the sums implicit in equations 23 and 24 but this will mean that $\mathbf{A}^+\mathbf{A}$ will no longer be $\mathbf{I}$. in fact, substituting 20 into 23 gives

$$\hat{\mathbf{x}} = \mathbf{A}^+\mathbf{A}\mathbf{x} = \mathbf{R}\mathbf{x} \tag{25}$$

and the matrix $\mathbf{R} = \mathbf{A}^+\mathbf{A}$ is sometimes called the "resolution matrix". In a perfectly resolved system, $\mathbf{R} = \mathbf{I}$ but, in general, each model element estimated will be a linear combination of all the model elements. For the truncated SVD approximation to the generalized inverse, $\mathbf{R} = \mathbf{V}\mathbf{V}^T$. We use the resolution matrix to estimate how much we are "blurring" the model.

The process of throwing away small singular values is an example of "regularization" of the inverse problem. It is not a commonly used method because the model we end up with doesn't satisfy any particularly sensible optimization criterion. Usually we seek a model which has some property optimized and still adequately satisfies the data. For example, we might seek a model which has minimum first or second derivative. Let $\mathbf{D}$ be some "roughening" operation on the model. Then we might want to minimize

$$f = (\mathbf{A}\mathbf{x} - \mathbf{d})^T(\mathbf{A}\mathbf{x} - \mathbf{d}) + \lambda(\mathbf{D}\mathbf{x})^T\mathbf{D}\mathbf{x} \tag{26}$$

where the parameter λ controls the degree of smoothing. Expanding out the brackets and taking the derivative with respect to $\mathbf{x}$ and setting to zero gives

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D}^T\mathbf{D})^{-1}\mathbf{A}^T\mathbf{d} \tag{27}$$

Clearly, setting λ to zero gives us our least-squares result. Comparing equations 23 and 27 gives $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D}^T\mathbf{D})^{-1}\mathbf{A}^T$ and we can use 24 and 25 to estimate the model covariance matrix and the resolution matrix. Increasing λ will result in models which have a smaller value of $\mathbf{x}^T\mathbf{D}^T\mathbf{D}\mathbf{x}$. One choice for $\mathbf{D}$ is $\mathbf{I}$ which results in a process called "ridge regression" and ends up minimizing the Euclidean length of the solution vector. This turns out to be a bad thing to do in tomography as it results in models which have wildly underestimated amplitudes. A good choice for $\mathbf{D}$ is the first difference operator which in 1D looks like:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & ... \\ 0 & 1 & -1 & 0 & ... \\ 0 & 0 & 1 & -1 & ... \end{pmatrix} \tag{28}$$

In tomography, we use this for for smoothing in the radial direction and we use a form which minimizes the sum of the first differences between a block and its four nearest neighbors laterally for lateral smoothing (an approximation to the Laplacian). In practice, very different degrees of radial and lateral smoothing are required in the tomography problem because radial and lateral length scales are so different for mantle structure.

We have already complained about forming matrix products like $\mathbf{A}^T\mathbf{A}$ when the matrices are ill-conditioned and, in any case, making $\mathbf{A}^T\mathbf{A}$ can itself be time consuming (and may remove the sparsity). In practice, we construct the following equivalent system:

$$\begin{pmatrix} \mathbf{A} \\ \lambda^{\frac{1}{2}}\mathbf{D} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{d} \\ \mathbf{0} \end{pmatrix} \tag{29}$$

and solve this rectangular system using SVD – or more likely a solver which takes advantage of the sparseness of the matrices $\mathbf{A}$ and $\mathbf{D}$.

One final technical point about solving equation 29 is that we can help the conditioning of the system by solving a slightly different system:

$$\mathbf{C}\mathbf{y} = \begin{pmatrix} \mathbf{A} \\ \lambda^{\frac{1}{2}}\mathbf{D} \end{pmatrix} \mathbf{W}\mathbf{y} = \begin{pmatrix} \mathbf{d} \\ \mathbf{0} \end{pmatrix} \quad \text{where} \quad \mathbf{y} = \mathbf{W}^{-1}\mathbf{x} \tag{30}$$

for $\mathbf{y}$ then getting $\mathbf{x}$ from $\mathbf{x} = \mathbf{W}\mathbf{y}$. $\mathbf{W}$ can be chosen in a variety of ways – one is to make it a diagonal matrix such that the Euclidean lengths of the columns of $\mathbf{C}$ are the same – this makes the range of singular values of $\mathbf{C}$ much less extreme and also speeds up convergence of some of the iterative techniques we discuss in the next section. This process of weighting is sometimes called "preconditioning" of the system and whole books have been written on the topic.

We now consider some "iterative" techniques for solving large systems of (hopefully) sparse equations. Such techniques can operate on one row of the matrix at a time (and are sometimes called row-action methods)

6. True iterative techniques

For simplicity, we go back to equation 20: $\mathbf{A}\mathbf{x} = \mathbf{d}$ though we are more likely to be solving something like equation 30 in practice. Let $\mathbf{x}^q$ be the q'th iterate and define the residual vector

$$\mathbf{r}^q = \mathbf{d} - \mathbf{A} \cdot \mathbf{x}^q \tag{31}$$

Now we want to perturb $\mathbf{x}^q$ to get a better answer. One way to do this is to work one equation at a time. Let $\Delta\mathbf{x}^q$ be the desired perturbation. We choose $\Delta\mathbf{x}^0$ to be the perturbation that makes the first element of $\mathbf{r}^0$ be zero, $\Delta\mathbf{x}^1$ is chosen to make the second element of $\mathbf{r}^1$ zero and so on – we then cycle through the equations until we get convergence. To get a unique perturbation, we choose the one that has $\|\Delta\mathbf{x}^q\|$ minimized. Thus we minimize

$$\left(A_{ij}\Delta x_j^q - r_i^q\right)^2$$

Then

$$\Delta x_j = \frac{A_{ij} r_i}{\sum_k A_{ik}^2} \tag{32}$$

This is the original procedure of Kaczmarz and is not terribly efficient. One popular modification to this is to compute the correction for each row (as above) and then average all the corrections to get a mean $\Delta\mathbf{x}$:

$$\Delta x_j = \frac{1}{M} \sum_{i=1}^{M} \frac{A_{ij} r_i}{\sum_k A_{ik}^2} \tag{33}$$

where M is the number of non-zero elements in A_{ij}. This process is called the Simultaneous Iterative Reconstruction Technique (SIRT) and is still commonly used. Some modifications are described in Hager and Clayton, 1989. A general family of SIRT methods is given by

$$\Delta x_j = \frac{\Omega}{\gamma_j} \sum_{i=1}^{M} \frac{A_{ij} r_i}{\rho_i}$$

where

$$\gamma_j = \sum_i |A_{ij}|^\alpha, \quad \rho_i = \sum_k |A_{ik}|^{2-\alpha}$$

with $0 < \Omega < 2$ and $0 < \alpha < 2$. Hager et al use $(\alpha = 1, \Omega = 1)$. It turns out that SIRT as described above converges to a solution which is not the least squares solution of the original system of equations and some weighting must be applied to correct this (van der Sluis and van der Vorst, 1987). SIRT works well in practice but it is now more common to use a conjugate gradient method – one particular variant called LSQR has become popular in seismic tomography.

7. Gradient (Projection) techniques

Consider the function defined by

$$f(\mathbf{x}) = \tfrac{1}{2}\,(\mathbf{A}\cdot\mathbf{x}-\mathbf{d})^2 \tag{34}$$

In two dimensions ($\mathbf{x} = x_1, x_2$), f is a surface which has hills and valleys. Expanding out this function gives

$$\begin{aligned} f &= \tfrac{1}{2}\,(\mathbf{A}\cdot\mathbf{x}-\mathbf{d})^T(\mathbf{A}\cdot\mathbf{x}-\mathbf{d}) \\ &= \tfrac{1}{2}\,[\mathbf{d}^T\cdot\mathbf{d}+\mathbf{x}^T\cdot\mathbf{A}^T\cdot\mathbf{A}\cdot\mathbf{x}-2\mathbf{x}^T\cdot\mathbf{A}^T\cdot\mathbf{d}] \end{aligned}$$

Now define the square symmetric matrix $\mathbf{B} = \mathbf{A}^T\cdot\mathbf{A}$ and the vector $\mathbf{b} = \mathbf{A}^T\cdot\mathbf{d}$ then

$$f = \tfrac{1}{2}\,[\mathbf{d}^T\cdot\mathbf{d}+\mathbf{x}^T\mathbf{B}\cdot\mathbf{x}-2\mathbf{x}^T\cdot\mathbf{b}]$$

The first term on the right is just the length of the data vector so we define the misfit function $\phi(\mathbf{x})$ as the last two terms:

$$\phi(\mathbf{x}) = \tfrac{1}{2}\,\mathbf{x}^T\mathbf{B}\cdot\mathbf{x}-\mathbf{x}^T\cdot\mathbf{b} \tag{35}$$

(This is the same function as f with all the same hills and valleys but with an offset removed.)
The gradient of ϕ with respect to $\mathbf{x}$ is simply

$$\nabla\phi(\mathbf{x}) = \mathbf{B}\cdot\mathbf{x}-\mathbf{b} \tag{36}$$

At any point $\mathbf{x}_k$ on the surface, the downhill slope is given by

$$-\nabla\phi(\mathbf{x}_k) = \mathbf{b}-\mathbf{B}\cdot\mathbf{x}_k = \mathbf{r}_k \tag{37}$$

and is actually zero at a solution which fits the data ($\mathbf{B}\cdot\mathbf{x}-\mathbf{b}=0$)

Our procedure is to find $\mathbf{x}$ by moving in a sequence of directions which take us down the misfit surface. Let

$$\mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{u}_k \tag{38}$$

where $\mathbf{u}_k$ is a direction we choose to go in. We can find the value of λ_k (assuming $\mathbf{u}_k$ is specified) that minimizes

$$\phi(\mathbf{x}_k+\lambda_k\mathbf{u}_k)$$

$$\phi = \tfrac{1}{2}\,(\mathbf{x}_k+\lambda_k\mathbf{u}_k)^T\cdot\mathbf{B}\cdot(\mathbf{x}_k+\lambda_k\mathbf{u}_k)-(\mathbf{x}_k+\lambda_k\mathbf{u}_k)^T\cdot\mathbf{b}$$

so

$$\frac{\partial\phi}{\partial\lambda_k} = \mathbf{u}_k^T\cdot\mathbf{B}\cdot\mathbf{x}_k+\lambda_k\mathbf{u}_k^T\cdot\mathbf{B}\cdot\mathbf{u}_k-\mathbf{u}_k^T\cdot\mathbf{b} = 0$$

so

$$\mathbf{u}_k^T\cdot(\mathbf{B}\cdot\mathbf{x}_k-\mathbf{b})+\lambda_k\mathbf{u}_k^T\cdot\mathbf{B}\cdot\mathbf{u}_k = 0$$

$$\lambda_k = \frac{\mathbf{u}_k^T\cdot\mathbf{r}_k}{\mathbf{u}_k^T\cdot\mathbf{B}\cdot\mathbf{u}_k} \tag{39}$$

The next question is how to specify $\mathbf{u}_k$. If we choose $\mathbf{u}_k = \mathbf{r}_k$ we get the "steepest descent algorithm" (remember $\mathbf{r}$ is the local downhill direction – see equation 37):

$$\mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{r}_k \quad \text{where} \quad \lambda_k = \frac{\mathbf{r}_k^T\cdot\mathbf{r}_k}{\mathbf{r}_k^T\cdot\mathbf{B}\cdot\mathbf{r}_k} \tag{40}$$

This isn't always a very good idea since it is possible to go from one side of the valley to another – rather than going down the middle. A better method is to chose directions so that they are "conjugate" (perpendicular in some sense) to all previous directions.

Reconsider equation 38:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{u}_k$$

Note that $\mathbf{x}_{k+1}$ is actually a linear combination of all the directions taken to date: $\mathbf{u}_1 ... \mathbf{u}_k$ – if there are N model parameters, then the final $\mathbf{x}$ can be completely specified by an expansion in N (orthogonal) directions:

$$\mathbf{x} = \lambda_1 \mathbf{u}_1 + \lambda_2 \mathbf{u}_2 + \cdots + \lambda_N \mathbf{u}_N$$

If the directions were truly orthogonal to each other, we could just dot this equation with the transpose of the j'th $\mathbf{u}$ and that would pick out the j'th term. It turns out that this isn't computationally helpful – but it is helpful to make the directions "B-orthogonal" which means that

$$\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_j = 0$$

Applying this to the above equation gives

$$\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{x} = \mathbf{u}_k^T \cdot \mathbf{b} = \lambda_k \mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k$$

A conjugate-gradient algorithm can now be developed. We start with $\mathbf{x}_1 = 0$ and compute $\mathbf{r}_1 = \mathbf{b}$. For the first direction, we choose steepest descent so $\mathbf{u}_1 = \mathbf{r}_1$ and we get λ_1 from equation 39. We are now at point $\mathbf{x}_2$ and can compute $\mathbf{r}_2$. In steepest descents, $\mathbf{r}_2$ would be our next direction but this is not "B-orthogonal" to the previous direction. To achieve this, we let the new direction be

$$\mathbf{u}_{k+1} = \mathbf{r}_{k+1} + \gamma_k \mathbf{u}_k \tag{41}$$

Dotting through by $(\mathbf{B} \cdot \mathbf{u}_k)^T$ gives

$$\gamma_k = -\frac{\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{r}_{k+1}}{\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k}$$

This form for γ_k is not computationally optimal as we shall see. To get our final algorithm, we first note that the $\mathbf{r}$'s can be computed recursively. Multiply equation 38 by $\mathbf{B}$ and subtract $\mathbf{b}$ from both sides:

$$\mathbf{B} \cdot \mathbf{x}_{k+1} - \mathbf{b} = \mathbf{B} \cdot \mathbf{x}_k - \mathbf{b} + \lambda_k \mathbf{B} \cdot \mathbf{u}_k$$

so

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \lambda_k \mathbf{B} \cdot \mathbf{u}_k \tag{42}$$

We can further manipulate the above formulae to get some identities which allow us to compute λ_k and γ_k more efficiently. First, note that we recover equation 39 from equation 42 if we require $\mathbf{u}_k^T \cdot \mathbf{r}_{k+1} = 0$. Forcing this to be true and dotting $\mathbf{r}_{k+1}^T$ into equation 41 gives the result that $\mathbf{r}_k^T \cdot \mathbf{u}_k = \mathbf{r}_k^T \cdot \mathbf{r}_k$. Furthermore, if we dot $\mathbf{r}_{k+1}^T$ into 42 and use equation 39 for λ_k and the above formula for γ_k, we get

$$\mathbf{r}_{k+1}^T \cdot \mathbf{r}_{k+1} = \mathbf{r}_{k+1}^T \cdot \mathbf{r}_k - \lambda_k \mathbf{r}_{k+1}^T \cdot \mathbf{B} \cdot \mathbf{u}_k = \mathbf{r}_{k+1}^T \cdot \mathbf{r}_k + \gamma_k \mathbf{u}_k^T \cdot \mathbf{r}_k \tag{43}$$

Similarly, dotting $(\mathbf{B} \cdot \mathbf{u}_{k+1})^T$ into 41 shows that $\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k = \mathbf{r}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k$. Dotting $\mathbf{r}_k^T$ into 42 and using this result allow us to show that $\mathbf{r}_{k+1}^T \cdot \mathbf{r}_k = 0$. These identities allow us to compute γ_k and λ_k as

$$\gamma_k = \frac{\mathbf{r}_{k+1}^T \cdot \mathbf{r}_{k+1}}{\mathbf{r}_k^T \cdot \mathbf{r}_k} \qquad \lambda_k = \frac{\mathbf{r}_k^T \cdot \mathbf{r}_k}{\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k} \tag{44}$$

The algorithm can now be written (taking $\mathbf{x}_1 = 0$)

```
$k = 0$
$\mathbf{r}_1 = \mathbf{b}$
$\mathbf{u}_1 = \mathbf{r}_1$
$\mathbf{x}_1 = 0$
$begin \quad loop$
    $k = k + 1$
    $\mathbf{w} = \mathbf{B} \cdot \mathbf{u}_k$
    $\lambda = \mathbf{r}_k^T \cdot \mathbf{r}_k / \mathbf{u}_k^T \cdot \mathbf{w}$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda \mathbf{u}_k$
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \lambda \mathbf{w}$
    $\gamma = \mathbf{r}_{k+1}^T \cdot \mathbf{r}_{k+1} / \mathbf{r}_k^T \cdot \mathbf{r}_k$
    $\mathbf{u}_{k+1} = \mathbf{r}_{k+1} + \gamma \mathbf{u}_k$
$end \quad loop$
```

Note that there is only one matrix-vector multiply per iteration. M iterations of this process would give the exact solution (in the absence of roundoff) but it is anticipated that much fewer than M iterations will be required to get an acceptable solution.

The algorithm described above is the standard CG algorithm – Golub and Van Loan (Chapter 10) 1996 give an extensive discussion of the theory. This is not in the best form for numerical application since it uses the "normal" equations $\mathbf{B} \cdot \mathbf{x} - \mathbf{b}$ which, as we have already noted, can square the condition number and introduce instability. We would like to go back to the rectangular system in equation 20. Remember, even just forming $\mathbf{B}$ can turn a sparse $\mathbf{A}$ matrix into a dense $\mathbf{B}$ matrix though the sparseness can be retained by computing $\mathbf{B} \cdot \mathbf{u}$ as $\mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{u})$. An equivalent sparse square system can be written down:

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}$$

and used to develop algorithms which do not implicitly use the normal equations and which are stable when systems are not well-conditioned (e.g. LSQR). We leave this as an exercise to the reader.

One final point: knowing when to stop iterative techniques can be a bit of an art form. Typically, much of the misfit to the data is taken up in the first few iterations but convergence to a stable model can take much longer. In particular, where we include a smoother (as in equation 30), it seems that the effect of the smoother becomes more apparent at later iterations even though the fit to the data does not change much. Several stopping criteria for LSQR have been suggested (see original papers by Paige and Saunders) but it pays to be conservative and to iterate longer than you think you need to! I have found that stopping when the norm of the model vector has reached a stable value works best.

8. Resolution and error analysis when the generalized inverse is not available

In section 5, we discussed resolution and error and gave results in terms of the generalized inverse of $\mathbf{A}$ (equations 24 and 25). How do we go about computing resolution and error when $\mathbf{A}^+$ is not available (as when using an iterative technique). One way of estimating the resolution matrix is to do an inversion where we set the m'th element of the model vector $\mathbf{x}$ to one and all the others to zero – call this vector $\mathbf{x}_m$. Now, compute $\mathbf{d}_m = \mathbf{A}\mathbf{x}_m$ and solve $\mathbf{A}\mathbf{x} = \mathbf{d}_m$ using exactly the same iterative algorithm as you used to get your true model. This process computes a single row (and column) of the resolution matrix corresponding to the m'th model element. The complete resolution matrix can be computed by performing M such inversions – one for each model parameter. Clearly this is infeasible if we are talking about 50,000 model parameters but we can focus on key areas of the model where we are particularly interested in the resolvability of a particular structure.

A modification of the above process (which is sometimes called a "spike test") is to solve for some pattern to test resolution over a broad region. A common choice is to use a checkerboard pattern in one of the layers of the model. A synthetic data set is computed for this checkerboard model and then inverted using exactly the same iterative algorithm used to get the real model. The recovered checkerboard can indicate areas of problematic recovery in the layer being tested and can show leakage into adjacent layers above and below.

The estimation of the covariance matrix of the model can also be problematic but usually we are satisfied with the diagonal elements (the square roots of which are the standard deviations of the model paramters). It turns out that the best way to estimate these is to add a noise vector to the data vector $\mathbf{d} = \mathbf{d} + \mathbf{e}$ where the elements of $\mathbf{e}$ are randomly chosen from a normal distribution with a unit standard deviation (remember, we divided all data by their errors initially). We then solve for a model using this perturbed data vector in our iterative procedure. We repeat this process many times (100 say) and then look at the standard deviations of the elements of the 100 models we have generated. Tests show that this process produces an excellent estimate of the diagonal elements of the model covariance matrix.

9. Importance of earthquake location in tomography

It turns out that our (in)ability to locate earthquakes accurately means that we have a source of noise in our tomographic problem which can rival the signal from 3D structure (at least for P-wave tomography). We can estimate the uncertainty due to event mislocation by considering the following equation

$$\delta t = \frac{\partial t}{\partial x}\delta x + \frac{\partial t}{\partial y}\delta y + \frac{\partial t}{\partial z}\delta z + \delta t_0,$$

where $\delta x, \delta y, \delta z$ are errors in event location, δt_0 is the error in origin time, and δt is the resulting error in travel time. Analyses of mislocations of events located by independent means leads to estimates of the length of a typical mislocation vector, ϵ_X of $\simeq$14–18 km . To convert this number to a typical change in epicentral distance, we assume that the stations are uniformly distributed around the event so that stations in a direction perpendicular to the mislocation vector see no change in epicentral distance while stations in the direction of mislocation will see the full value. Assuming a cosinusoidal dependence as a function of azimuth suggests that, on average, the error in epicentral distance is $\simeq \epsilon_X/\sqrt{2}$. Assuming that δx and δy do not co-vary (as suggested by an analysis of the differences of the NEIC and ISC locations), the error in the travel time due to the error in each of x and y is

$$\frac{p\epsilon_X}{\sqrt{2}},$$

where p is the ray parameter. It is well known that errors in depth and origin time do covary with $\delta t_0 \simeq \delta z/9$ (depth in kilometers). Since $\partial t/\partial z$ is negative, the errors in origin time and depth tend to cancel in their contribution to the total error and the errors in x and y dominate the error budget. We now assume a typical depth uncertainty of about 10 km and find that σ_X is .6–1.2 seconds for P waves at epicentral distances of about 70° for mislocation vectors of length 10–20 km. The corresponding estimate of σ_X for S waves is 1.6–2.5 seconds. These numbers rival the signals from 3D structure.

These results mean that we cannot ignore earthquake mislocation in our tomography and we must either relocate events or make our data insensitive to event location. Consider the travel time residuals for one event:

$$\delta\mathbf{t} = \mathbf{A}\delta\mathbf{h} + \mathbf{B}\delta\mathbf{v} \tag{45}$$

We could iteratively solve this equation first by relocating the events then solving separately for velocity structure then reloacting again but now including the new velocity structure. Convergence is usually attained after a few iterations – but often to an incorrect solution. Alternatively, we can seek linear combinations of the data for each event which, to first order, are insensitive to the event location. This reduces to finding $\mathbf{P}$ such that

$$\mathbf{P}\delta\mathbf{t} = \mathbf{P}\mathbf{A}\delta\mathbf{h} + \mathbf{P}\mathbf{B}\delta\mathbf{v} = \mathbf{P}\mathbf{B}\delta\mathbf{v} \tag{46}$$

i.e., we want $\mathbf{PA} = 0$. Note that if $\mathbf{A}$ has the SVD $\mathbf{A} = \mathbf{U\Lambda V}^T$ then $\mathbf{P} = \mathbf{G}(\mathbf{I} - \mathbf{UU}^T)$ where $\mathbf{G}$ is any matrix. We choose $\mathbf{G}$ so that the new data $\delta\mathbf{t}' = \mathbf{P}\delta\mathbf{t}$ are statistically independent. If $\delta\mathbf{t}$ has a covariance matrix $\mathbf{I}$ then $\delta\mathbf{t}'$ has covariance matrix $\mathbf{G}(\mathbf{I} - \mathbf{UU}^T)\mathbf{G}^T$ (since $(\mathbf{I} - \mathbf{UU}^T) = (\mathbf{I} - \mathbf{UU}^T)^T$ and $(\mathbf{I} - \mathbf{UU}^T)(\mathbf{I} - \mathbf{UU}^T) = (\mathbf{I} - \mathbf{UU}^T)$). Thus, if $(\mathbf{I} - \mathbf{UU}^T)$ has the eigenvalue decomposition $\mathbf{R\Omega R}^T$ then choosing $\mathbf{G} = \mathbf{\Omega}^{-\frac{1}{2}}\mathbf{R}^T$ leads to the desired covariance matrix which is $\mathbf{I}$. It is interesting that the eigenvalues of $(\mathbf{I} - \mathbf{UU}^T)$ are one or zero and we lose four eigenvalues during the projection process – we have effectively used up four data to remove sensitivity to location.

The alternative process is to relocate initially, solving

$$\delta\mathbf{t} = \mathbf{A}\delta\mathbf{h} \tag{47}$$

which, if we have used a SVD would lead to a mislocation vector

$$\delta\hat{\mathbf{h}} = \mathbf{V\Lambda}^{-1}\mathbf{U}^T\delta\mathbf{t} \tag{48}$$

and equation 45 would become

$$\delta\mathbf{t} - \mathbf{A}\delta\hat{\mathbf{h}} \equiv (\mathbf{I} - \mathbf{UU}^T)\delta\mathbf{t} = \mathbf{B}\delta\mathbf{v} \tag{49}$$

Note this is similar to the projection method where $\mathbf{P} = \mathbf{G}(\mathbf{I} - \mathbf{UU}^T)$ except that we have not taken account of the fact that we have "used some data up" in doing the relocation and we have not consistently operated on the $\mathbf{B}$ part of equation 45 as we did with the projection method. Ignoring these niceties does still leave us with a $\mathbf{B}$ matrix which is sparse whereas, in the projection method, each new travel time is a linear combination of all the travel times for that event so that $\mathbf{PB}$ is no longer as sparse as we would like. Unfortunately, alternating between solving for locations then 3D velocity structure can result in convergence to the wrong answer – so this technique should be used with great care.

It is also possible to solve equation 45 directly which means expanding the unknown vector we are solving for by four times the number of events. Since we are often dealing with 10,000 events, this means we are adding 40,000 more unknowns to the problem. However, the matrices remain extremely sparse so this approach is entirely feasible from a computational point of view.