# CHAPTER 2

## *Geophysical inverse problems*

***2.1 Introduction .*** The art of geophysics is to make inferences of the properties of the Earth from measurements made at the Earth's surface. An example of such a problem is the inference of the velocity structure as a function of depth from travel time curves. Can we mathematically answer the question of what is really resolved by the data we have? These notes are directed towards addressing this question.

The solution of an inverse problem can be formally split into four parts:

1) existence
2) uniqueness
3) construction
4) evaluation.

Most geophysicists are obsessed with part 3 – finding a model which is consistent with the data. The first two parts are of interest primarily from a mathematical point of view and part 4 – model evaluation is extremely important from a practical point of view.

To solve an inverse problem we must already have solved the *forward problem, i.e.*, we must have developed the machinery to compute predicted data from a model. This requires a mathematical model and there will always be assumptions built into this. Suppose we have measured the mass and moment of inertia of the Earth. If the Earth has a spherically symmetric density distribution we can write

$$M = 4\pi \int_0^R \rho(r) r^2 \, dr \tag{2.1}$$

and

$$C = \frac{8\pi}{3} \int_0^R \rho(r) r^4 \, dr \tag{2.2}$$

By assuming that the Earth has a spherically symmetric density distribution we have simplified our problem considerably. These equations represent the forward problem – given $\rho(r)$, compute $M$ and $C$. The *inverse problem* is – given $M$ and $C$, compute $\rho(r)$. This type of problem is the simplest to solve because it is *linear*. If the density is doubled everywhere, $M$ and $C$ are doubled. Quite a lot of theory exists for linear problems and we shall look at this type of problem in detail. Most geophysical inverse problems are non-linear and we often linearize about some starting model which is (we hope) close to the final model. Very little can be said in general about non-linear problems except that there may be multiple solutions to the problem which are not close to one another in solution space. Some techniques have been developed which can efficiently search model space (e.g. genetic algorithms, evolutionary programming, etc) and can handle strongly non-linear problems providing the number of model parameters is not too large. We shall not consider these further here.

***2.2 Linear inverse problems .*** We shall first write our linear problem in a more general way:

$$d_i \pm \sigma_i = \int_0^R G_i(r) m(r)\, dr \tag{2.3}$$

$d_i$ is the $i^{th}$ datum and has an error $\sigma_i$, $G_i(r)$ is a "data kernel" for the $i^{th}$ datum and $m(r)$ is the model. In our example we have two data:

$$d_1 = M \quad \text{and} \quad d_2 = C$$

$$G_1 = 4\pi r^2 \quad \text{and} \quad G_2 = \frac{8\pi}{3} r^4$$

$$m(r) = \rho(r)$$

The first question we shall consider is one of existence. Is the data consistent with our mathematical formulation of the problem? In our example the question is – does any density distribution $\rho(r)$ exist which would predict our data? It should be obvious to you that it does – however, there are some cases where this may not be true – our mathematical formulation may be inadequate.

The question of uniqueness can be simply answered for the linear inverse problem. If there is one solution consistent with a finite dataset, there are an infinite number of solutions consistent with the data. In practical problems we always have a finite amount of data. Sometimes we can artificially generate an infinite dataset. For example we may have a travel-time ($T - X$) curve which we have obtained by fitting a polynomial to some data. $T$ is then formally defined for every $X$ so we have an infinite dataset. It turns out that in this problem a unique solution exists, *i.e.*, there is a unique velocity-depth distribution which exactly fits our $T - X$ curve. It also turns out that this is a very bad way of solving the problem. The model generated is very sensitive to details of the $T - X$ curve and can lead to some strange unphysical features, *e.g.*, velocity triplications in depth!

This leads us to the question of model construction. This is the easiest part and there exist many methods of generating continuous functions of depth (say) from a finite dataset. Generally they involve some additional subjective constraints, *e.g.*, the model must be geophysically reasonable. Often the model is parameterized in some way and we consider this in detail next.

## 3. The need for parameterization.

Consider a generic linearized inverse problem:

$$d_i \pm \sigma_i = \int_0^R G_i \delta m\, dr \tag{2.4}$$

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 $\delta m$ 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:

$$\delta m = \sum_{i=1,N} a_i G_i(r) \tag{2.5}$$

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

$$\mathbf{d} = \mathbf{\Gamma} \cdot \mathbf{a} \quad \text{where} \quad \Gamma_{ij} = \int_0^R G_i G_j \, dr \tag{2.6}$$

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. We can also look at the ability of our data to resolve features of our models using Backus Gilbert theory which, though we do not have time to cover it here, is discussed in detail in an accompanying pdf by Masters and Gubbins (2003).

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:

$$\delta m = \sum_{i=1,M} a_i f_i(r) \tag{2.7}$$

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

$$\mathbf{d} = \mathbf{A} \cdot \mathbf{a} \quad \text{where} \quad A_{ij} = \int_0^R G_i f_j \, dr \tag{2.8}$$

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:

$$\delta m = \sum_{s,t} \delta m_s^t(r) Y_s^t(\theta, \phi) \tag{2.9}$$

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 and are the natural basis for interpreting mode structure coefficients. 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, 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

$$ka = l + \tfrac{1}{2} = \frac{2\pi a}{\lambda} \tag{2.10}$$

where $k$ is wavenumber, $\lambda$ 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 mimicing 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. Finding a model

Let us suppose we are solving the problem

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{d} \tag{2.11}$$

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 sytem of equations (11) were well-conditioned, we might just find the least-squares solution:

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d} \tag{2.12}$$

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 (12) is rarely chosen. One way around squaring the condition number is to use a singular value decomposition (SVD) on equation 11. The matrix $\mathbf{A}$ is decomposed into singular values and matrices of left and right eigenvectors:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{2.13}$$

where $\mathbf{U}$ has dimension $N \times N$ and $\mathbf{V}$ has dimension $M \times M$ and $\mathbf{\Lambda}$ is a $M \times N$ 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

$$\hat{\mathbf{x}} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{U}^T\mathbf{d} = \mathbf{A}^+\mathbf{d} \tag{2.14}$$

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:

$$\mathbf{A}^+\mathbf{I}(\mathbf{A}^+)^T = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{U}^T\mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{V}^T = \mathbf{V}\mathbf{\Lambda}^{-2}\mathbf{V}^T \tag{2.15}$$

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 14 and 15 but this will mean that $\mathbf{A}^+\mathbf{A}$ will no longer be $\mathbf{I}$. in fact, substituting 11 into 14 gives

$$\hat{\mathbf{x}} = \mathbf{A}^+\mathbf{A}\mathbf{x} = \mathbf{R}\mathbf{x} \tag{2.16}$$

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. The we might want to minimize

$$f = (\mathbf{A}\mathbf{x} - \mathbf{d})^T(\mathbf{A}\mathbf{x} - \mathbf{d}) + \lambda(\mathbf{D}\mathbf{x})^T\mathbf{D}\mathbf{x} \tag{2.17}$$

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

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D}^T\mathbf{D})^{-1}\mathbf{A}^T\mathbf{d} \tag{2.18}$$

Clearly, setting $\lambda$ to zero gives us our least-squares result. Comparing equations 14 and 18 gives $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D}^T\mathbf{D})^{-1}\mathbf{A}^T$ and we can use 15 and 16 to estimate the model covariance matrix and the resolution matrix. Increasing $\lambda$ 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:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \dots \\ 0 & 1 & -1 & 0 & \dots \\ 0 & 0 & 1 & -1 & \dots \end{pmatrix} \tag{2.19}$$

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. 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:

$$\begin{pmatrix} \mathbf{A} \\ \lambda^{\frac{1}{2}}\mathbf{D} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{d} \\ \mathbf{0} \end{pmatrix} \tag{2.20}$$

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 20 is that we can help the conditioning of the system by solving a slightly different system:

$$\mathbf{C}\mathbf{y} = \begin{pmatrix} \mathbf{A} \\ \lambda^{\frac{1}{2}}\mathbf{D} \end{pmatrix} \mathbf{W}\mathbf{y} = \begin{pmatrix} \mathbf{d} \\ \mathbf{0} \end{pmatrix} \quad \text{where} \quad \mathbf{y} = \mathbf{W}^{-1}\mathbf{x} \tag{2.21}$$

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)

## 5. True iterative techniques

For simplicity, we go back to equation 11: $\mathbf{Ax} = \mathbf{d}$ though we are more likely to be solving something like equation 21 in practice. Let $\mathbf{x}^q$ be the $q$'th iterate and define the residual vector

$$\mathbf{r}^q = \mathbf{d} - \mathbf{A} \cdot \mathbf{x}^q \tag{2.22}$$

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

$$\Delta x_j = \frac{A_{ij} r_i}{\sum_k A_{ik}^2} \tag{2.23}$$

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 $\Delta\mathbf{x}$:

$$\Delta x_j = \frac{1}{M} \sum_{i=1}^{M} \frac{A_{ij} r_i}{\sum_k A_{ik}^2} \tag{2.24}$$

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, probably because it was popularized in the mid 80's by Guust Nolet.

## 6. Gradient (Projection) techniques

Consider the function defined by

$$f(\mathbf{x}) = \tfrac{1}{2}\,(\mathbf{A}\cdot\mathbf{x}-\mathbf{d})^2 \tag{2.25}$$

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:

$$\phi(\mathbf{x}) = \tfrac{1}{2}\,\mathbf{x}^T\mathbf{B}\cdot\mathbf{x}-\mathbf{x}^T\cdot\mathbf{b} \tag{2.26}$$

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

$$\nabla\phi(\mathbf{x}) = \mathbf{B}\cdot\mathbf{x}-\mathbf{b} \tag{2.27}$$

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

$$-\nabla\phi(\mathbf{x}_k) = \mathbf{b}-\mathbf{B}\cdot\mathbf{x}_k = \mathbf{r}_k \tag{2.28}$$

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

$$\mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{u}_k \tag{2.29}$$

where $\mathbf{u}_k$ is a direction we choose to go in. We can find the value of $\lambda_k$ (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

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

$$\lambda_k = \frac{\mathbf{u}_k^T \cdot \mathbf{r}_k}{\mathbf{u}_k^T \cdot \mathbf{B} \cdot \mathbf{u}_k} \tag{2.30}$$

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 28):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{r}_k \quad \text{where} \quad \lambda_k = \frac{\mathbf{r}_k^T \cdot \mathbf{r}_k}{\mathbf{r}_k^T \cdot \mathbf{B} \cdot \mathbf{r}_k} \tag{2.31}$$

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 29:

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

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 $\lambda_1$ from equation 30. 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

$$\mathbf{u}_{k+1} = \mathbf{r}_{k+1} + \gamma_k \mathbf{u}_k \tag{2.32}$$

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 $\gamma_k$ 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 29 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

$$\mathbf{r}_{k+1}=\mathbf{r}_k-\lambda_k\mathbf{B}\cdot\mathbf{u}_k \tag{2.33}$$

We can further manipulate the above formulae to get some identities which allow us to compute $\lambda_k$ and $\gamma_k$ more efficiently. First, note that we recover equation 30 from equation 33 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 32 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 30 and use equation 30 for $\lambda_k$ and the above formula for $\gamma_k$, we get

$$\mathbf{r}_{k+1}^T\cdot\mathbf{r}_{k+1}=\mathbf{r}_{k+1}^T\cdot\mathbf{r}_k-\lambda_k\mathbf{r}_{k+1}^T\cdot\mathbf{B}\cdot\mathbf{u}_k=\mathbf{r}_{k+1}^T\cdot\mathbf{r}_k+\gamma_k\mathbf{u}_k^T\cdot\mathbf{r}_k \tag{2.34}$$

Similarly, dotting $(\mathbf{B}\cdot\mathbf{u}_{k+1})^T$ into 32 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 33 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 $\gamma_k$ and $\lambda_k$ as

$$\gamma_k=\frac{\mathbf{r}_{k+1}^T\cdot\mathbf{r}_{k+1}}{\mathbf{r}_k^T\cdot\mathbf{r}_k}\qquad\lambda_k=\frac{\mathbf{r}_k^T\cdot\mathbf{r}_k}{\mathbf{u}_k^T\cdot\mathbf{B}\cdot\mathbf{u}_k} \tag{2.35}$$

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 11. 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 21), 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 we typically use a criterion based on convergence of the model vector (rather than some data misfit criterion).

## 7. Resolution and error analysis

In section 4, we discussed resolution and error and gave results in terms of the generalized inverse of $\mathbf{A}$ (equations 15 and 16). How do we go about computing resolution and error when $\mathbf{A}^+$ is not available (as when using an iterative technique). Some have suggested using a rough estimate of $\mathbf{A}^+$ (e.g. Nolet et al,1999, GJI,v138,p36) using a one-step back projection which gives

$$\mathbf{A}^+ \simeq \mathbf{A}^T \boldsymbol{\Omega} \tag{2.36}$$

where $\boldsymbol{\Omega}$ is a diagonal matrix and

$$\Omega_{kk} = \frac{(\mathbf{A}\mathbf{A}^T)_{kk}}{\sum_{i=1}^{N} (\mathbf{A}\mathbf{A}^T)^2_{ik}} \tag{2.37}$$

It is not clear to us how well this performs in practice but we are often only interested in the overall nature of the resolution matrix and not precise values for its elements. Perhaps this is adequate for this.

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. Some people also use complicated test structures (such as slabs) to see how well recovered such structures are but I worry that this kind of test can cover up smearing – particularly if the test structure is chosen to look like a structure recovered in the inversion.

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.

We shall illustrate some of these techniques with an inversion of surface wave phase data.