

## CHAPTER 9: CONTINUOUS INVERSE PROBLEMS AND OTHER APPROACHES

### 9.1 Introduction

Until this point, we have only considered discrete inverse problems, either linear or nonlinear, that can be expressed in the form

$$\mathbf{d} = \mathbf{G}\mathbf{m} \quad (1.13)$$

We now turn our attention to another branch on inverse problems, called continuous inverse problems, in which at least the model parameter vector  $\mathbf{m}$  is replaced by a continuous function, and the matrix  $\mathbf{G}$  is replaced by an integral relationship. The general case of a linear continuous inverse problem involving continuous data and a continuous model function is given by a Fredholm equation of the first kind

$$g(y) = \int k(x, y)m(x)dx \quad \text{continuous data } g(y) \quad (9.1)$$

where the data  $g$  are a continuous function of some variable  $y$ , the model  $m$  is a continuous function of some other variable  $x$ , and  $k$ , called the data kernel or Green's function, is a function of both  $x$  and  $y$ .

In most situations, data are not continuous, but rather are a finite sample. For example, analog (continuous) seismic data is digitized to become a finite, discrete data set. In the case where there are  $N$  data points, Equation (9.1) becomes

$$g_j = \int k_j(x)m(x)dx \quad \text{discrete data, } j = 1, N \quad (9.2)$$

One immediate implication of a finite data set of dimension  $N$  with a continuous (infinite dimensional) model is that the solution is underconstrained, and if there is any solution  $m(x)$  that fits the data, there will be an infinite number of solutions that will fit the data as well. This basic problem of nonuniqueness was encountered before for the discrete problem in the minimum length environment (Chapter 3).

Given that all continuous inverse problems with discrete data are nonunique, and almost all real problems have discrete data, the goals of a continuous inverse analysis can be somewhat different than a typical discrete analysis. Three possible goals of a continuous analysis include: (1) find a model  $m(x)$  that fits the data  $g_j$ , also known as *construction*, (2) find unique properties or values of all possible solutions that fit the data by taking linear combinations of the data, also

known as *appraisal*, and (3) find the values of other linear combinations of the model using the data  $g_j$ , also known as *inference* (Oldenburg, 1984). There are many parallels, and some fundamental differences, between discrete and continuous inverse theory. For example, we have encountered the construction phase before for discrete problems whenever we used some operator to find a solution that best fits the data. The appraisal phase is most similar to an analysis of resolution and stability analysis for discrete problems. We have not encountered the inference phase before. Emphasis in this chapter on continuous inverse problems will be on the construction and appraisal phases, and the references, especially Oldenburg [1984] can be used for further information on the inference phase.

The material in this chapter is based primarily on the following references:

Backus, G. E. and J. F. Gilbert, Numerical application of a formalism for geophysical inverse problems, *Geophys. J. Roy. Astron. Soc.*, 13, 247–276, 1967.

Huestis, S. P., An introduction to linear geophysical inverse theory, unpublished manuscript, 1992.

Jeffrey, W., and R. Rosner, On strategies for inverting remote sensing data, *Astrophys. J.*, 310, 463–472, 1986.

Oldenburg, D. W., An introduction to linear inverse theory, *IEEE Trans. Geos. Remote Sensing*, Vol. GE-22, No. 6, 665–674, 1984.

Parker, R. L., Understanding inverse theory, *Ann. Rev. Earth Planet. Sci.*, 5, 35–64, 1977.

Parker, R. L., *Geophysical Inverse Theory*, Princeton University Press, 1994.

## 9.2 The Backus–Gilbert Approach

There are a number of approaches to solving Equations (9.1) or (9.2). This chapter will deal exclusively with one approach, called the Backus–Gilbert approach, which was developed by geophysicists in the 1960's. This approach is based on taking a linear combination of the data  $g_j$ , given by

$$\begin{aligned}
 l &= \sum_{j=1}^N \alpha_j g_j = \sum_{j=1}^N \alpha_j \int k_j(x) m(x) dx \\
 &= \int \left[ \sum_{j=1}^N \alpha_j k_j(x) \right] m(x) dx
 \end{aligned}
 \tag{9.3}$$

where the  $\alpha_j$  are as yet undefined constants. The essence of the Backus-Gilbert approach is deciding how the  $\alpha_j$ 's are chosen.

If the expression in square brackets,  $[\sum \alpha_j k_j(x)]$ , has certain special properties, it is possible in theory to construct a solution from the linear combination of the data. Specifically, if

$$\left[ \sum_{j=1}^N \alpha_j k_j(x) \right] = \delta(x - x_0) \quad (9.4)$$

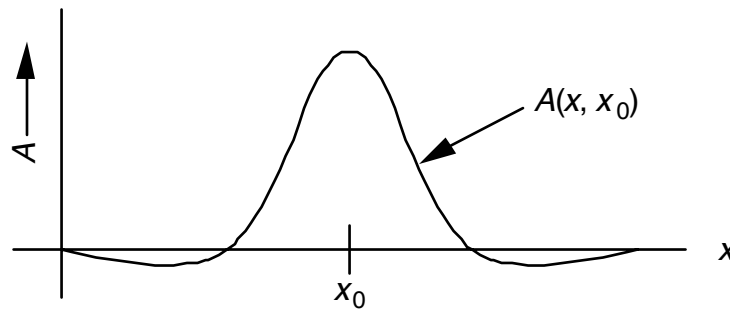
where  $\delta(x - x_0)$  is the Dirac delta function at  $x_0$ , then

$$l(x_0) = \left[ \sum_{j=1}^N \alpha_j(x_0) g_j \right] = \int \delta(x - x_0) m(x) dx = m(x_0) \quad (9.5)$$

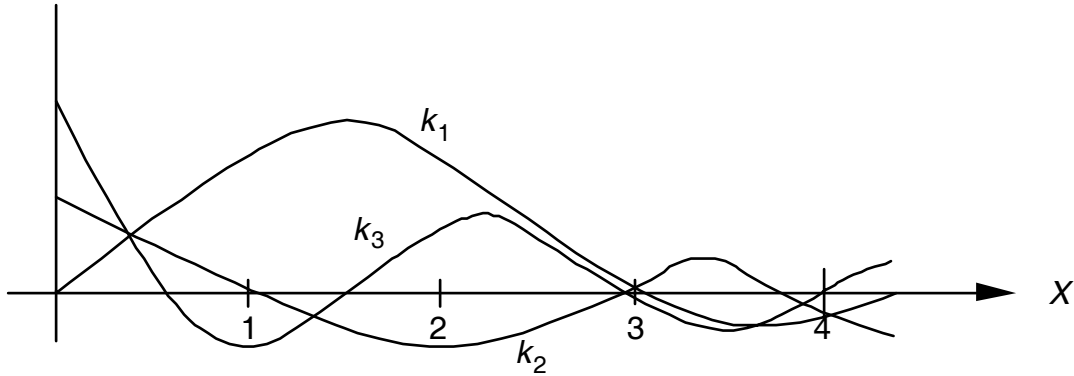
That is, choosing the  $\alpha_j$  such that  $[\sum \alpha_j k_j(x)]$  is as much like a delta function as possible at  $x_0$ , we can obtain an estimate of the solution at  $x_0$ . The expression  $[\sum \alpha_j k_j(x)]$  plays such a fundamental role in Backus–Gilbert theory that we let

$$\sum_{j=1}^N \alpha_j k_j(x) = A(x, x_0) \quad (9.6)$$

where  $A(x, x_0)$  has many names in the literature, including *averaging kernel*, *optimal averaging kernel*, *scanning function*, and *resolving kernel*. The averaging kernel  $A(x, x_0)$  may take many different forms, but in general is a function which at least peaks near  $x_0$ , as shown below



Recall from Equation (9.6) that the averaging kernel  $A(x, x_0)$  is formed as a linear function of a finite set of data kernels  $k_j$ . An example of a set of three data kernels  $k_i$  over the interval  $0 < x < 4$  is shown on the next page.



One of the fundamental problems encountered in the Backus-Gilbert approach is that the finite set of data kernels  $k_j, j = 1, N$  is an incomplete set of basis functions from which to construct the Dirac delta function. You may recall from Fourier analysis, for example, that a spike (Dirac delta) function in the spatial domain has a white spectrum in the frequency domain, which implies that it takes an infinite sum of sin and cosine terms (basis functions) to construct the spike function. It is thus impossible for the averaging kernel  $A(x, x_0)$  to exactly equal the Dirac delta with a finite set of data kernels  $k_j$ .

Much of Backus-Gilbert approach thus comes down to deciding how best to make  $A(x, x_0)$  approach a delta function. Backus and Gilbert defined three measures of the “deltaness” of  $A(x, x_0)$  as follows

$$J = \int [A(x, x_0) - \delta(x - x_0)]^2 dx \quad (9.7)$$

$$K = 12 \int (x - x_0)^2 [A(x, x_0) - \delta(x - x_0)]^2 dx \quad (9.8)$$

$$W = \int \left[ H(x - x_0) - \int A(x, x_0) dx \right]^2 dx \quad (9.9)$$

where  $H(x - x_0)$  is the Heaviside, or unit step, function at  $x_0$ .

The smaller  $J, K,$  or  $W$  is, the more the averaging kernel approaches the delta function in some sense. Consider first the  $K$  criterion:

$$K = 12 \int \left\{ (x - x_0)^2 [A(x, x_0)]^2 - 2(x - x_0)^2 A(x, x_0) \delta(x - x_0) + (x - x_0)^2 [\delta(x - x_0)]^2 \right\} dx \quad (9.10)$$

The second and third terms drop out because  $\delta(x - x_0)$  is nonzero only when  $x = x_0$ , and then the  $(x - x_0)^2$  term is zero. Thus

$$\begin{aligned}
 K &= 12 \int (x - x_0)^2 [A(x, x_0)]^2 dx \\
 &= 12 \int (x - x_0)^2 \left[ \sum_{j=1}^N \alpha_j k_j(x) \right]^2 dx
 \end{aligned} \tag{9.11}$$

We minimize  $K$  by taking the partials with respect to the  $\alpha_j$ 's and setting them to zero.

$$\begin{aligned}
 \frac{\partial K}{\partial \alpha_i} &= 12 \int \left\{ (x - x_0)^2 2 \left[ \sum_{j=1}^N \alpha_j k_j(x) \right] k_i(x) \right\} dx = 0 \\
 &= 24 \sum_{j=1}^N \alpha_j \int (x - x_0)^2 k_i(x) k_j(x) dx = 0
 \end{aligned} \tag{9.12}$$

Writing out the sum over  $j$  explicitly for the  $i$ th partial derivative gives

$$\begin{aligned}
 \frac{\partial K}{\partial \alpha_i} &= \left[ \int (x - x_0)^2 k_i(x) k_1(x) dx \right] \alpha_1 + \left[ \int (x - x_0)^2 k_i(x) k_2(x) dx \right] \alpha_2 + \cdots \\
 &\quad + \left[ \int (x - x_0)^2 k_i(x) k_N(x) dx \right] \alpha_N = 0
 \end{aligned} \tag{9.13}$$

Combining the  $N$  partial derivatives and using matrix notation this becomes

$$\begin{array}{c}
 \left[ \begin{array}{cccc}
 \int (x - x_0)^2 k_1 k_1 dx & \int (x - x_0)^2 k_1 k_2 dx & \cdots & \int (x - x_0)^2 k_1 k_N dx \\
 \int (x - x_0)^2 k_2 k_1 dx & \int (x - x_0)^2 k_2 k_2 dx & \cdots & \int (x - x_0)^2 k_2 k_N dx \\
 \vdots & \vdots & \ddots & \vdots \\
 \int (x - x_0)^2 k_N k_1 dx & \int (x - x_0)^2 k_N k_2 dx & \cdots & \int (x - x_0)^2 k_N k_N dx
 \end{array} \right] \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} = \begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \end{array} \\
 N \times N & \qquad \qquad \qquad N \times 1 \quad N \times 1
 \end{array} \tag{9.14}$$

or

$$\mathbf{B} \boldsymbol{\alpha} = \mathbf{0} \tag{9.15}$$

$\mathbf{B} \boldsymbol{\alpha} = \mathbf{0}$  has the trivial solution  $\boldsymbol{\alpha} = \mathbf{0}$ . Therefore, we add the constraint, with Lagrange multipliers, that

$$\int A(x, x_0) dx = 1 \tag{9.16}$$

which says the area under the averaging kernel is one, or that  $A(x, x_0)$  is a unimodular function. Adding the constraint to the original  $K$  criterion creates a new criterion  $K'$  given by

$$K' = 12 \int (x - x_0)^2 [A(x, x_0)]^2 dx + \lambda \left\{ \left[ \int A(x, x_0) dx \right] - 1 \right\} \quad (9.17)$$

which leads to

$$\begin{bmatrix} 24 \int (x - x_0)^2 k_1 k_1 dx & \cdots & 24 \int (x - x_0)^2 k_1 k_N dx & \int k_1 dx \\ \vdots & \ddots & \vdots & \vdots \\ 24 \int (x - x_0)^2 k_N k_1 dx & \cdots & 24 \int (x - x_0)^2 k_N k_N dx & \int k_N dx \\ \int k_1 dx & \cdots & \int k_N dx & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (9.18)$$

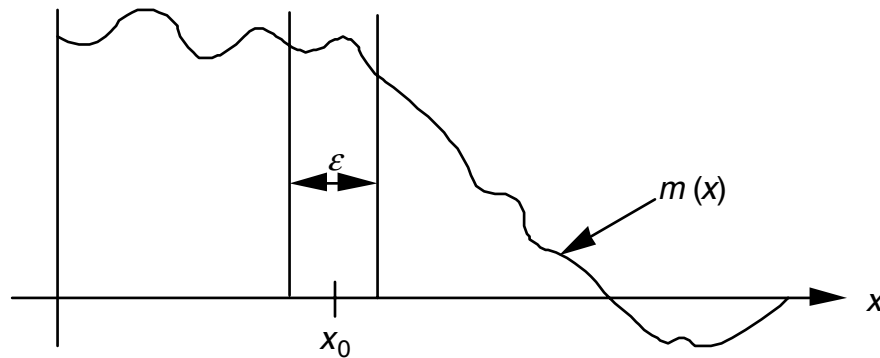
$(N + 1) \times (N + 1)$

or

$$C \begin{bmatrix} \boldsymbol{\alpha} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \quad (9.19)$$

Then the  $\alpha_j$ 's are found by inverting the square, symmetric matrix  $C$ .

The factor of 12 in the original definition of  $K'$  was added to facilitate the geometrical interpretation of  $K'$ . Specifically, with the factor of 12 included,  $K' = \varepsilon$  if  $A(x, x_0)$  is a unimodular (i.e.,  $\int A(x, x_0) dx = 1$ ) box car of width  $\varepsilon$  centered on  $x_0$ . In general, when  $K'$  is small, it is found (Oldenburg, 1984, p. 669) that  $K'$  is a good estimation of the *width* of the averaging function  $A(x, x_0)$  at half its maximum value: "Thus  $K'$  gives essential information about the resolving power of the data." If  $K'$  is large, then the estimate of the solution  $m(x)$  will be a smoothed average of the true solution around  $x_0$ , and the solution will have poor resolution. Of course, you can also look directly at  $A(x, x_0)$  and get much the same information. If  $A(x, x_0)$  has a broad peak around  $x_0$ , then the solution will be poorly resolved in that neighborhood. Features of the solution  $m(x)$  with a scale length less than the width of  $A(x, x_0)$  cannot be resolved (i.e., are nonunique). Thus, on the figure below, the high-frequency variations in  $m(x)$  are not resolved.



The analysis of the averaging kernel above falls within the appraisal phase of the possible goals of a continuous inverse problem. Since any solution to Equation (9.2) is nonunique, often times the most important aspect of the inverse analysis is the appraisal phase.

The above discussion does not include possible data errors. Without data errors, inverting  $C$  to get the  $\alpha_j$ 's gives you the solution. Even if  $C$  is nearly singular, then except for numerical instability, once you have the  $\alpha_j$ 's, you have a solution. If, however, the data contain noise, then near-singularity of  $C$  can cause large errors in the solution for small fluctuations in data values.

It may help to think of the analogy between  $k_j$  and the  $k$ th row of  $\mathbf{G}$  in the discrete case  $\mathbf{G}\mathbf{m} = \mathbf{d}$ . Then

$$A(x, x_0) = \sum_{j=1}^N \alpha_j k_j \tag{9.20}$$

is equivalent to taking some linear combination of the rows of  $\mathbf{G}$  (which are  $M$ -dimensional, and therefore represent vectors in model space) and trying to make a row-vector as close as possible to a row of the identity matrix. If there is a near-linear dependency of rows in  $G$ , then the coefficients in the linear combination will get large. One can also speak of the near interdependence of the data kernels  $k_j(x)$ ,  $j = 1, N$  in Hilbert space. If this is the case, then terms like  $\int k_i k_j dx$  can be large, and  $C$  will be nearly singular. This will lead to *large values for  $\alpha$*  as it tries to approximate  $\delta(x - x_0)$  with a set of basis functions (kernels)  $k_j$  that have near interdependence. Another example arises in the figure after Equation (9.6) with three data kernels that are all nearly zero at  $x = 3$ . It is difficult to approximate a delta function near  $x = 3$  with a linear combination of the data kernels, and the coefficients of that linear combination are likely to be quite large.

You can quantify the effect of the near singularity of  $C$  by considering the data to have some covariance matrix [cov  $\mathbf{d}$ ]. Then the variance of your estimate of the solution  $m(x_0)$  at  $x_0$  is

$$\sigma_{m(x_0)}^2 = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_N] [\text{cov } \mathbf{d}] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} \tag{9.21}$$

and if

$$[\text{cov } \mathbf{d}] = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_N^2 \end{bmatrix} \tag{9.22}$$

then

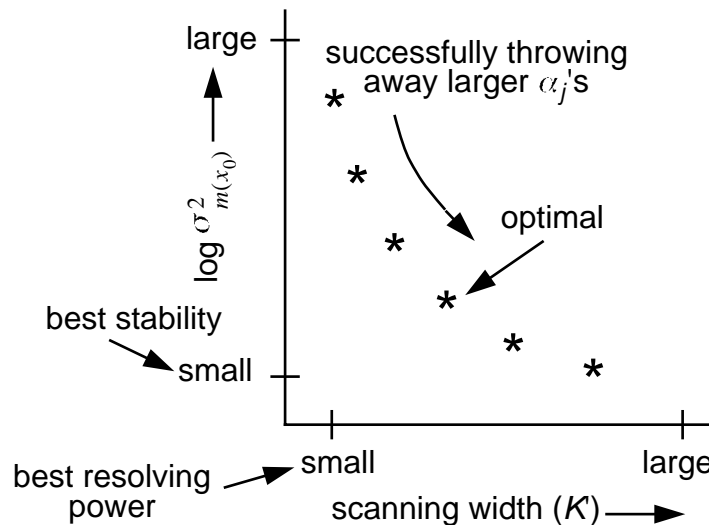
$$\sigma_{m(x_0)}^2 = \sum_{j=1}^N \alpha_j^2 \sigma_j^2 \tag{9.23}$$

[See Oldenburg, 1984, Equation (18), p. 670, or Jeffrey and Rosner, Equation (3.5), p. 467.]

If  $\sigma_{m(x_0)}^2$  is large, the solution is unstable. We now have two conflicting goals:

- (1) minimize  $K'$   
 versus (2) minimize  $\sigma_{m(x_0)}^2$

This leads to *trade-off* curves of stability (small  $\sigma_{m(x_0)}^2$ ) versus resolving power (small  $K'$ ). These trade-off curves are typically plotted as



Typically, one gains a lot of stability without too much loss in resolving power early on by throwing away the largest (few)  $\alpha_j$ 's.

Another way to accomplish the same goal is called *spectral expansion* or *spectral synthesis*. With this technique, you do singular-value decomposition on  $C$  and start throwing away the smallest singular values and associated eigenvectors until the error amplification (i.e.,  $\log \sigma_{m(x_0)}^2$ ) is sufficiently small. In either case, you give up resolving power to gain stability. Small singular values, or large  $\alpha_j$ , are associated with high-frequency components of the solution  $m(x)$ . As you give up resolving power to gain stability, the “width” of the resolving kernel increases. Thus, in general, the narrower the resolving kernel, the better the resolution but the poorer the stability. Similarly, the wider the resolving kernel, the poorer the resolution, but the better the stability.

The logic behind the trade-off between resolution and stability can be looked at another way. With the best resolving power you obtain the best fit to the data in the sense of minimizing the difference between observed and predicted data. However, if the data are known to contain

noise, then fitting the data exactly (or too well) would imply fitting the noise. It does not make sense to fit the data better than the data uncertainties in this case. We can quantify this relation for the case in which the data errors are Gaussian and uncorrelated with standard deviation  $\sigma_j$  by introducing  $\chi^2$

$$\chi^2 = \sum_{j=1}^N (g_j - \hat{g}_j)^2 / \sigma_j^2 \quad (9.24)$$

where  $\hat{g}_j$  is the predicted  $j$ th datum, which depends on the choice of  $\alpha$ .  $\chi^2$  will be in the range  $0 \leq \chi^2 \leq \infty$ . If  $\chi^2 = 0$ , then the data are fit perfectly, noise included. If  $\chi^2 \approx N$ , then the data are being fit at about the one standard deviation level. If  $\chi^2 \gg N$ , then the data are fit poorly. By using the trade-off curve, or the spectral expansion method, you affect the choice of  $\alpha_j$ 's, and hence the predicted data  $\hat{g}_j$  and ultimately the value of  $\chi^2$ . Thus the best solution is obtained by adjusting the  $\alpha_j$  until  $\chi^2 \approx N$ .

Now reconsider the  $J$  criterion:

$$\begin{aligned} J &= \int [A(x, x_0) - \delta(x - x_0)]^2 dx \\ &= \int [A^2(x, x_0) - 2A(x, x_0)\delta(x - x_0) + \delta^2(x - x_0)] dx \end{aligned} \quad (9.25)$$

Recall that for the  $K$  criterion, the second and third terms vanished because they were multiplied by  $(x - x_0)^2$ , which is zero at the only place the other two terms are nonzero because of the  $\delta(x - x_0)$  term. With the  $J$  criterion, it is minimized by taking partial derivatives with respect to  $\alpha_i$  and setting equal to zero. The  $A^2(x, x_0)$  terms are similar to the  $K$  criterion case, but

$$\begin{aligned} \frac{\partial}{\partial \alpha_i} \left[ \int -2A(x, x_0)\delta(x - x_0) \right] &= -2 \frac{\partial}{\partial \alpha_i} \int \sum_{j=1}^N \alpha_j k_j(x) \delta(x - x_0) dx \\ &= -2 \int k_i(x) \delta(x - x_0) dx = -2k_i(x_0) \end{aligned} \quad (9.26)$$

Thus the matrix equations from minimizing  $J$  become

$$\begin{bmatrix} \int k_1 k_1 & \int k_1 k_2 & \cdots & \int k_1 k_N \\ \int k_1 k_2 & \int k_2 k_2 & \cdots & \int k_2 k_N \\ \vdots & \vdots & \ddots & \vdots \\ \int k_1 k_N & \int k_2 k_N & \cdots & \int k_N k_N \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} = 2 \begin{bmatrix} k_1(x_0) \\ k_2(x_0) \\ \vdots \\ k_N(x_0) \end{bmatrix} \quad (9.27)$$

$N \times N \qquad N \times 1 \qquad N \times 1$

or

$$\mathbf{D}\alpha = 2\mathbf{k} \quad (9.28)$$

Notice now that  $\alpha = \mathbf{0}$  is not a trivial solution, as it was in the case of the  $K$  criterion. Thus, we do not have to use Lagrange multipliers to add a constraint to insure a nontrivial solution. Also, notice that  $\mathbf{D}$ , the matrix that must be inverted to find  $\alpha$ , no longer depends on  $x_0$ , and thus need only be formed once. This benefit is often sacrificed by adding the constraint that  $A(x, x_0)$  be unimodular anyway! Then the  $\alpha$ 's found no longer necessarily give  $m(x_0)$  corresponding to a solution of  $g_j = \int k_j(x)m(x) dx$  at all, but this may be acceptable if your primary goal is appraisal of the properties of solutions, rather than construction of one among many possible solutions.

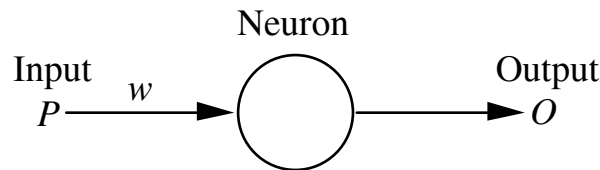
The  $J$  and  $K'$  criteria lead to different  $\alpha$ . The  $J$  criterion typically leads to narrower resolving kernels, but often at the expense of negative side lobes. These side lobes confuse the interpretation of  $A(x, x_0)$  as an “averaging” kernel. Thus, most often the  $K'$  criterion is used, even though it leads to somewhat broader resolving kernels.

### 9.3 Neural Networks

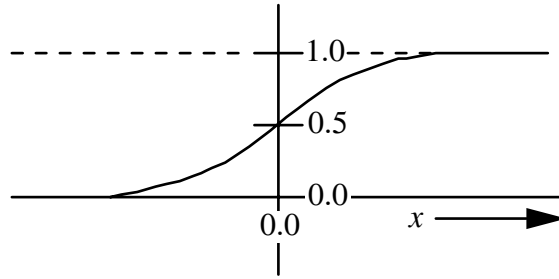
Neural networks, based loosely on models of the brain and dating from research in the 1940s (long before the advent of computers), have become very successful in a wide range of applications, from pattern recognition (one of the earliest applications) to aerospace (autopilots, aircraft control systems), defense (weapon steering, signal processing), entertainment (animation and other special effects), manufacturing (quality control, product design), medical (breast cancer cell analysis, EEG analysis, optimization of transplant times), oil and gas (exploration), telecommunications (image and data compression), and speech (speech recognition) applications.

The basic idea behind neural networks is to design a system, based heavily on a parallel processing architecture, that can learn to solve problems of interest.

We begin our discussion of neural networks by introducing the neuron model, which predictably has a number of names, including, of course, *neurons*, but also *nodes*, *units*, and *processing elements* (PEs),



where the neuron sees some input  $P$  coming, which is weighted by  $w$  before entering the neuron. The neuron processes this weighted input  $= Pw$  and creates an output  $O$ . The output of the neuron can take many forms. In early models, the output was always  $+1$  or  $-1$  because the neural network was being used for pattern recognition. Today, this neuron model is still used (called the step function, or Heaviside function, among other things), but other neuron models have the output equal to the weighted input to the neuron (called the linear model) and perhaps the most common of all, the sigmoid model, which looks like



often taken to be given by  $S(x) = 1/(1 + e^{-x})$ , where  $x$  is the weighted input to the neuron. This model has elements of both the linear and step function but has the advantage over the step function of being continuously differentiable.

So, how is the neuron model useful? It is useful because, like the brain, the neuron can be trained and can learn from experience. What it learns is  $w$ , the correct weight to give the input so that the output of the neuron matches some desired value.

As an almost trivial example, let us assume that our neuron in the figure above behaves as a linear model, with the output equal to the weighted input. We train the neuron to learn the correct  $w$  by giving it examples of inputs and output that are true. For example, consider examples to be

input =	1	output =	2
	10		20
	-20		-40
	42		84

By inspection we recognize that  $w = 2$  is the correct solution. However, in general we start out not knowing what  $w$  should be. We thus begin by assuming it to be some number, perhaps 0, or as is typically done, some random number. Let us assume that  $w = 0$  for our example. Then for the first test with input = 1, our output would be 0. The network recognizes that the output does not match the desired output and changes  $w$ . There are a world of ways to change  $w$  based on the mismatch between the output and the desired output (more on this later), but let us assume that the system will increase  $w$ , but not all the way to 2, stopping at 0.5. Typically, neural networks do not change  $w$  to perfectly fit the desired output because making large changes to  $w$  very often results in instability. Now our system, with  $w = 0.5$ , inputs 10 and outputs 5. It again increases  $w$  and moves on to the other examples. When it has cycled through the known input/output pairs once, this is called an *epoch*. Once it has gone through an epoch, it goes back to the first example and cycles again until there is an acceptably small misfit between all of the outputs and desired outputs for all of the known examples. In neural network programs, the codes typically go through all of the known examples randomly rather than sequentially, but the idea is the same. In our example, it will settle in eventually on  $w = 2$ . In more realistic examples, it takes hundreds to thousands of iterations (one iteration equals an epoch) to find an acceptable set of weights.

In our nomenclature, we have the system

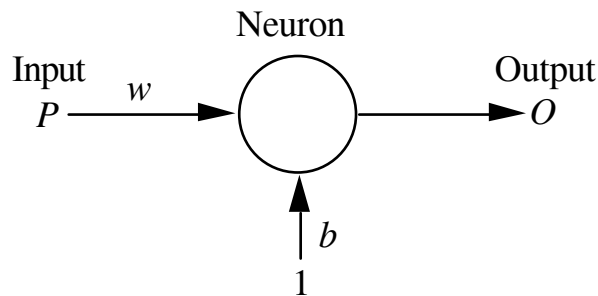
$$\mathbf{d} = \mathbf{Gm} \tag{1.13}$$

and in this example,  $G = 2$ . The beauty of the neural network is that it “learned” this relationship without even having to know it formally. It did it simply by adjusting weights until it was able to correctly match a set of example input/output pairs.

Suppose we change our example input/output pairs to

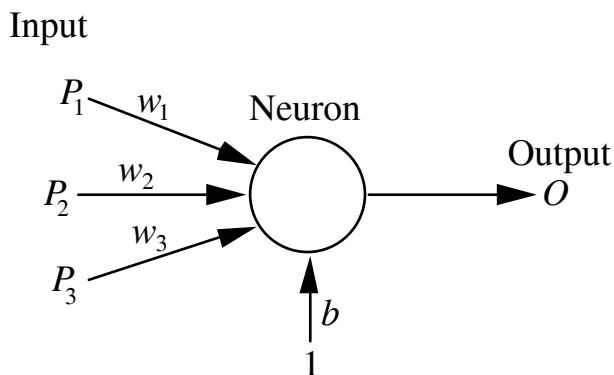
input =	1	output =	5
	10		23
	-20		-37
	42		87

where the output has been shifted 3 units from the previous example. You may recognize that we will be unable to find a single weight  $w$  to make the output of our neuron equal the desired output. This leads to the first additional element in the neuron model, called the *bias*. Consider the diagram below



where the bias is something added to the weighted input  $Pw$  before the neuron “processes” it to make output. It is convention that the bias has an input of 1 and another “weight” (bias)  $b$  that is to be determined in the learning process. In this example, the neuron will “learn” that, as before,  $w = 2$ , and now that the bias  $b$  is 3.

The next improvement to our model is to consider a neuron that has multiple inputs:

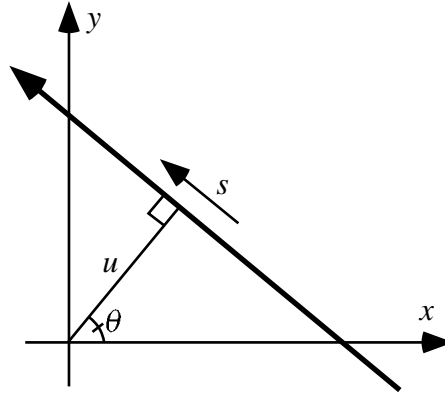


We now introduce the nomenclature that the process of the neuron is going to be some function of the sum of the weighted input vector and the bias  $b$ . That is, we describe the neuron by the function of  $(\mathbf{P} \cdot \mathbf{w} + b)$ , where

$$\mathbf{P} \cdot \mathbf{w} = P_1 w_1 + P_2 w_2 + \dots + P_N w_N = \sum_{i=1}^N P_i w_i \quad (9.29)$$

This is just a very incomplete introduction to neural networks. Maybe someday we will add more material.

## 9.4 The Radon Transform and Tomography (Approach 1)



### 9.4.1 Introduction

Consider a model space  $m(x, y)$  through which a straight-line ray is parameterized by the perpendicular distance  $u$  and angle  $\theta$ . Position  $(x, y)$  and ray coordinates  $(u, s)$  are related by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} u \\ s \end{bmatrix} \quad (9.30)$$

and

$$\begin{bmatrix} u \\ s \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (11.18 \text{ in Menke})$$

If  $m(x, y)$  is slowness ( $1/v$ ) model, and  $t(u, \theta)$  is the travel time along a ray at distance  $u$  and angle  $\theta$ , then

$$t(u, \theta) = \int_{-\infty}^{\infty} m(x, y) ds \quad (9.31)$$

is known as the Radon transform (RT). Another way of stating this is that  $t(u, \theta)$  is the "projection" of  $m(x, y)$  onto the line defined by  $u$  and  $\theta$ .

The inverse problem is: Given  $t(u, \theta)$  for many values of  $u$  and  $\theta$ , find the model  $m(x, y)$ , i.e., the inverse Radon transform (IRT).

Define a 2-D Fourier transform of the model:

$$\tilde{m}(k_x, k_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} m(x, y) e^{-2i\pi(k_x x + k_y y)} dx dy \quad (9.32)$$

$$m(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{m}(k_x, k_y) e^{-2i\pi(k_x x + k_y y)} dk_x dk_y \quad (9.33)$$

Now, define the 1-D Fourier Transform of the projection data:

$$\tilde{t}(k_u, \theta) = \int_{-\infty}^{\infty} t(u, \theta) e^{-2i\pi k_u u} du \quad (9.34)$$

$$t(u, \theta) = \int_{-\infty}^{\infty} \tilde{t}(k_u, \theta) e^{2i\pi k_u u} dk_u \quad (9.35)$$

Substituting Equation (9.31) into Equation (9.34) gives

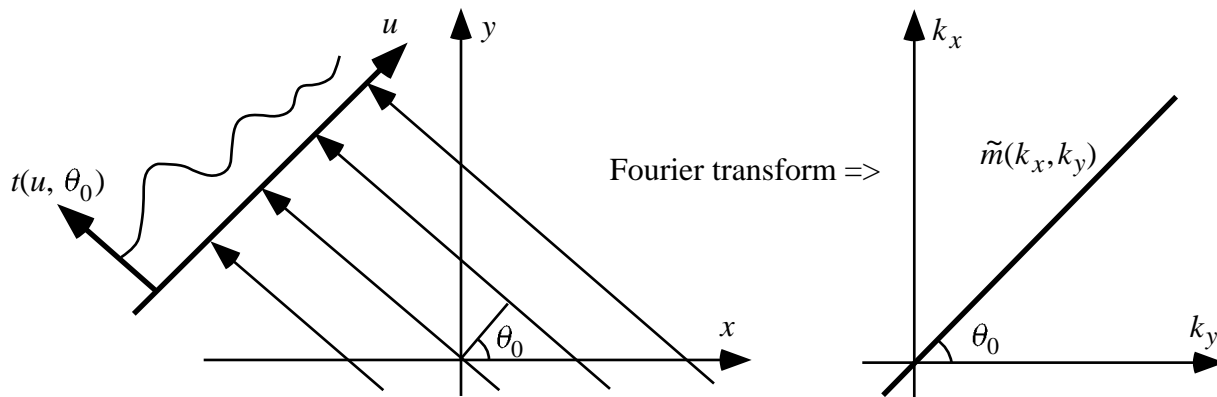
$$\tilde{t}(k_u, \theta) = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} m(x, y) ds \right] e^{-2i\pi k_u u} du \quad (9.36)$$

Making a change of variables  $ds du \rightarrow dx dy$  and using the fact that the Jacobian determinant is unity we have

$$\begin{aligned} \tilde{t}(k_u, \theta) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} m(x, y) e^{-2i\pi k_u (\cos \theta x + \sin \theta y)} dx dy \\ &= \tilde{m}(k_u \cos \theta, k_u \sin \theta) \end{aligned} \quad (9.37)$$

Equation (9.37) states that the 1-D Fourier transform of the projected data is equal to the 2-D Fourier transform of the model. This relationship is known as the Fourier (central) slice theorem because for a fixed angle  $\theta$ , the projected data provide the Fourier transform of the model slice through the origin of the wavenumber space, i.e.,

$$\begin{bmatrix} k_x \\ k_y \end{bmatrix} = k_u \begin{bmatrix} \cos \theta_0 \\ \sin \theta_0 \end{bmatrix}, \theta_0 \text{ fixed}$$



Now, we can invert the 2-D Fourier transform and recover the model,

$$m(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{m}[k_u(x \cos \theta + y \sin \theta)] e^{2i\pi(k_x x + k_y y)} dk_x dk_y \quad (9.38)$$

Change variables to polar coordinates gives

$$m(x, y) = \int_{-\infty}^{\infty} \int_0^{\pi} \tilde{m}(k_u, \theta) e^{2i\pi k_u(x \cos \theta + y \sin \theta)} |k_u| d\theta dk_u \quad (9.39)$$

where  $|k|$  arises because  $k|_{-\infty}^{\infty} \rightarrow d\theta|_0^{\pi}$ .

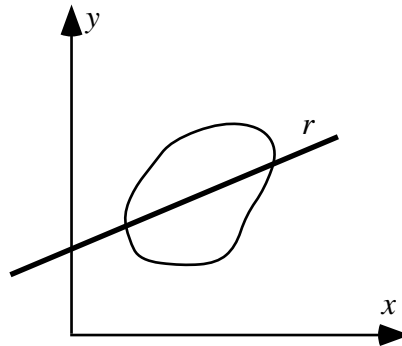
$$\begin{aligned} m(x, y) &= \int_0^{\pi} \int_{-\infty}^{\infty} \tilde{m}(k_u, \theta) |k| e^{2i\pi k_u(x \cos \theta + y \sin \theta)} dk_u d\theta \\ &= \int_0^{\pi} m^*(k_u(x \cos \theta + y \sin \theta), \theta) d\theta \end{aligned} \quad (9.40)$$

where  $m^*$  is obtained from the inverse Fourier transform of  $|k_u| \tilde{m}(k_u, \theta)$ .

The Radon Transform can be written more simply as

$$F(a, b) = \int_r f(x, a + bx) dr \quad (9.41)$$

shadow path model

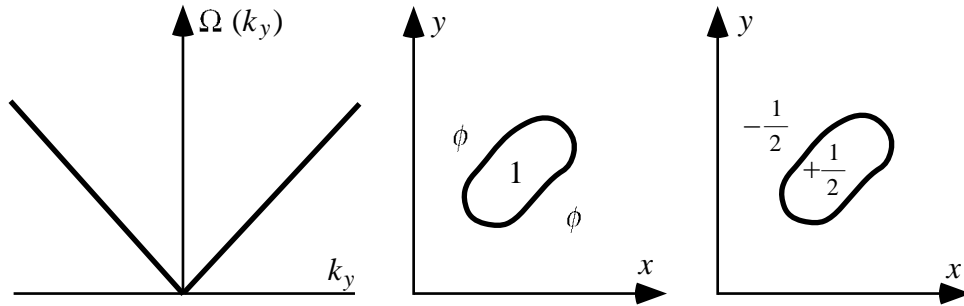


with the inverse

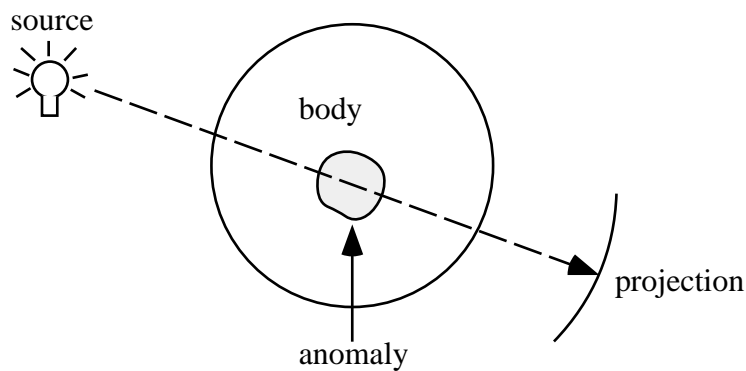
$$f(x, y) = \Omega(x, y) * \int_q F(y - bx, b) dq \quad (9.42)$$

model "rho filter" path shadows

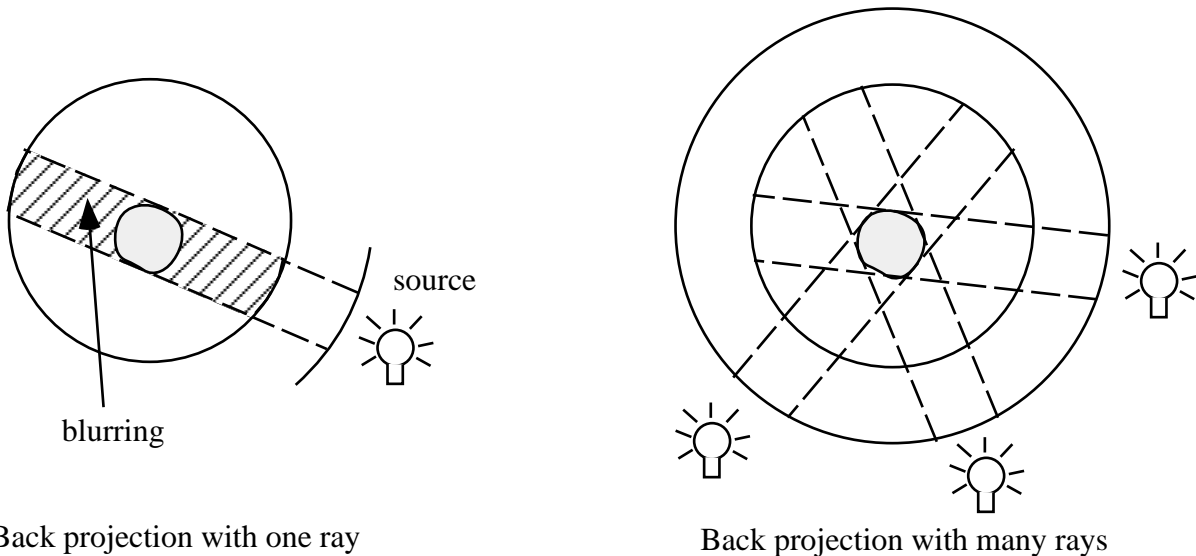
Note the similarity of the inverse and forward transforming with a change of sign in the argument of the integrand.



### 9.4.2 Interpretation of Tomography Using the Radon Transform



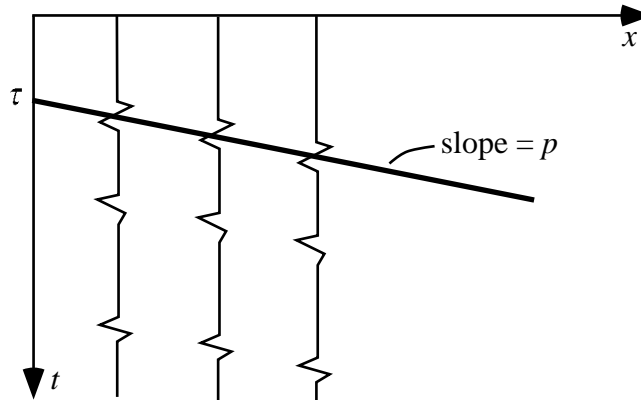
The resolution and accuracy of the reconstructed image are controlled by the coverage of the sources.



Because of this requirement for excellent coverage for a good reconstruction, the Radon transform approach is generally not used much in geophysics. One exception to this is in the area of seismic processing of petroleum industry data, where data density and coverage is, in general, much better.

9.4.3 Slant-Stacking as a Radon Transform (following Claerbout, 1985)

Let  $u(x, t)$  be a wavefield. An example would be as follows.

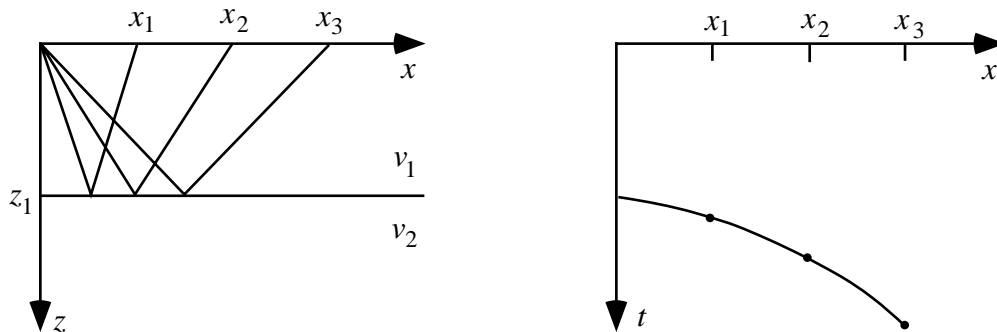


The *slant stack* of the wavefield is defined by

$$\bar{u}(p, \tau) = \int u(x, \tau + px) dx \tag{9.43}$$

The integral along  $x$  is done at constant  $\tau$ , which defines a slanting straight line in the  $x-t$  plane with slope  $p$ . Note the similarity of Equation (9.43) to Equation (9.41). Equation (9.43) is a Radon transform.

To get a better “feel” for this concept, let’s step back a little and consider the travel time equations of “rays” in a layered medium.



The travel time equation is

$$t = \sqrt{4z_1^2 + x^2} / v \text{ or } t^2 v^2 - x^2 = 4z_1^2 \tag{9.44}$$

Because the signs of the  $t^2$  and  $x^2$  terms are opposite, this is an equation of a hyperbola. Slant stacking is changing variables from  $t - x$  to  $\tau - p$  where

$$\tau = t - px \tag{9.45}$$

and

$$p = \frac{dt}{dx} \tag{9.46}$$

From Equation (9.44)

$$2t dt v^2 = 2x dx \tag{9.47}$$

so

$$\frac{dt}{dx} = \frac{x}{v^2 t} \tag{9.48}$$

The equations simplify if we introduce the parametric substitution.

$$\left. \begin{aligned} z &= vt \cos \theta \\ x &= vt \sin \theta \\ \text{so } x &= z \tan \theta \end{aligned} \right\} \tag{9.49}$$

Now, take the linear moveout Equation (9.46) and substitute for  $t$  and  $x$  (using  $p = (\sin \theta) / v$ ).

$$\begin{aligned} \tau &= \frac{z}{v \cos \theta} - \frac{\sin \theta}{v} z \tan \theta \\ &= \frac{z}{v} \cos \theta \end{aligned} \tag{9.50}$$

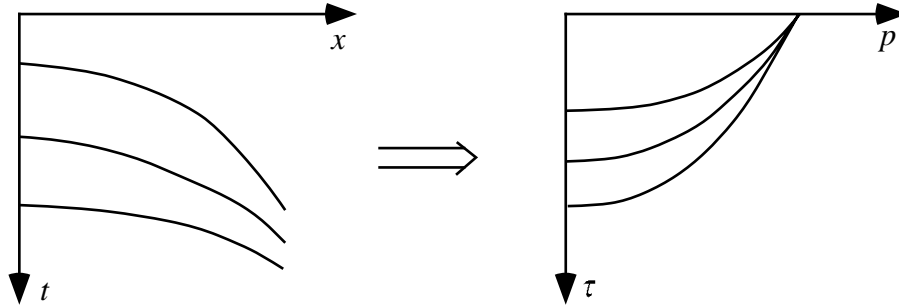
This can be written after some substitution as

$$\tau = \frac{z}{v} \sqrt{1 - p^2 v^2} \tag{9.51}$$

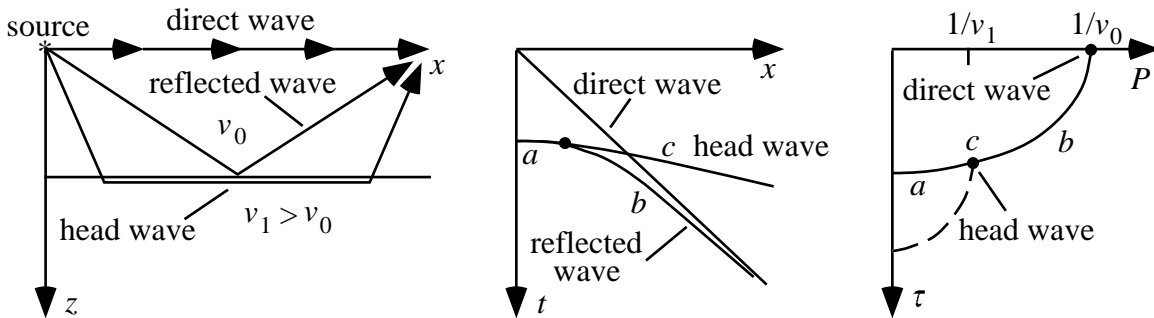
and finally,

$$\left( \frac{\tau}{x} \right)^2 + p^2 = \frac{1}{v^2} \tag{9.52}$$

This is an equation of an ellipse in  $\tau$ - $p$  space. All this was to show that hyperbolic curves in  $t$ - $x$  space transform to ellipses in  $\tau$ - $p$  space.



With this background, consider how the wavefield in a layer-over-halfspace slant stacks (or Radon transforms).



A couple of important features of the  $\tau$ - $p$  plot are as follows:

1. Curves cross one another in  $t$ - $x$  space but not in  $\tau$ - $p$  space.
2. The  $p$  axis has dimensions of  $1 / v$ .
3. The velocity of each layer can be read from its  $\tau$ - $p$  curve as the inverse of the maximum  $p$  value on its ellipse.
4. Head waves are points in  $\tau$ - $p$  space located where the ellipsoids touch.

Returning to our original slant-stack equation,

$$\bar{u}(p, \tau) = \int u(x, \tau + px) dx \tag{9.43}$$

This equation can be transformed into the Fourier space using

$$U(k, w) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x, t) e^{iwt - ikx} dx dt \tag{9.53}$$

Let  $p = k / w$

$$U(wp, w) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x, t) e^{iw(t - px)} dx dt \tag{9.54}$$

and change of variables from  $t$  to  $\tau = t - px$ .

$$U(wp, w) = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} u(x, t + px) dx \right] e^{iw\tau} d\tau \quad (9.55)$$

|<-----(9.43)----->|

Insert Equation (9.43) to get

$$U(wp, w) = \int_{-\infty}^{\infty} \bar{u}(p, \tau) e^{iw\tau} d\tau \quad (9.56)$$

Think of this as a 1-D function of  $w$  that is extracted from the  $k$ - $w$  plane along the line  $k = wp$ .

Finally, taking the inverse Fourier transform

$$\bar{u}(p, \tau) = \int_{-\infty}^{\infty} U(wp, w) e^{-iw\tau} dw \quad (9.57)$$

This result shows that a slant stack can be done by Fourier transform operations.

1. Fourier transform  $u(x, t)$  to  $U(k, w)$ .
2. Extract  $U(wp, w)$  from  $U(k, w)$  along the line  $k = wp$ .
3. Inverse Fourier transform from  $w$  to  $t$ .
4. Repeat for all interesting values of  $p$ .

Tomography is based on the same mathematics as inverse slant stacking. In simple terms, tomography is the reconstruction of a function given line integrals through it.

The inverse slant stack is based on the inverse Fourier transform of Equation (9.53),

$$u(x, t) = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} U(k, w) e^{ikx} dk \right] e^{-iwt} dw \quad (9.58)$$

substituting  $k = wp$  and  $dk = wpd$ . Notice that when  $w$  is negative, the integration with  $dp$  is from positive to negative. To keep the integration in the conventional sense, we introduce  $|w|$ ,

$$u(x, t) = \int_{-\infty}^{\infty} \left[ \int U(wp, w) |w| e^{iwp x} dp \right] e^{-iwt} dw \quad (9.59)$$

Changing the order of integration gives

$$u(x, t) = \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} [U(wp, w) e^{iwp x} |w|] e^{-iwt} dw \right\} e^{-iwt} dp \quad (9.60)$$

Note that the term in  $\{ \}$  contains the inverse Fourier transform of a product of three functions of frequency. The three functions are (1)  $U(wp, w)$ , which is the Fourier transform of the slant stack, (2)  $e^{iwp x}$ , which can be thought of as a delay operator, and (3)  $|w|$ , which is called

the “rho” filter. A product of three terms in Fourier space is a convolution in the original space. Let the delay  $px$  be a time shift in the argument, so,

$$u(x, t) = \text{rho}(t) * \int \bar{u}(p, t - px) dp \quad (9.61)$$

This is the inverse slant-stack equation. Comparing the forward and inverse Equations (9.41) and (9.58), note that the inverse is basically another slant stack with a sign change.

$$\bar{u}(p, \tau) = \int u(x, \tau + px) dx \quad (9.43)$$

$$u(x, t) = \text{rho}(t) * \int \bar{u}(p, t - px) dp \quad (9.61)$$

### 9.5 Review of the Radon Transform (Approach 2)

If  $m(x, y)$  is a 2-D slowness ( $1/v$ ) model, and  $t(u, \theta)$  is the travel time along a ray parameterized by distance  $u$  and angle  $\theta$ , then

$$t(u, \theta) = \int_{-\infty}^{\infty} m(x, y) ds \quad \text{is the Radon transform} \quad (9.62)$$

The inverse problem is: Given  $t(u, \theta)$  for many values of  $u$  and  $\theta$ , find the model  $m(x, y)$ . Take the 1-D Fourier transform of the projection data,

$$\tilde{t}(k_u, \theta) = \int_{-\infty}^{\infty} t(u, \theta) e^{-2i\pi k_u u} du \quad (9.63)$$

Substitute Equation (9.62),

$$\tilde{t}(k_u, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} m(x, y) ds e^{-2i\pi k_u u} du \quad (9.64)$$

Change variables  $ds du \rightarrow dx dy$

$$\tilde{t}(k_u, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} m(x, y) e^{-2i\pi k_u (\cos \theta x + \sin \theta y)} dx dy \quad (9.65)$$

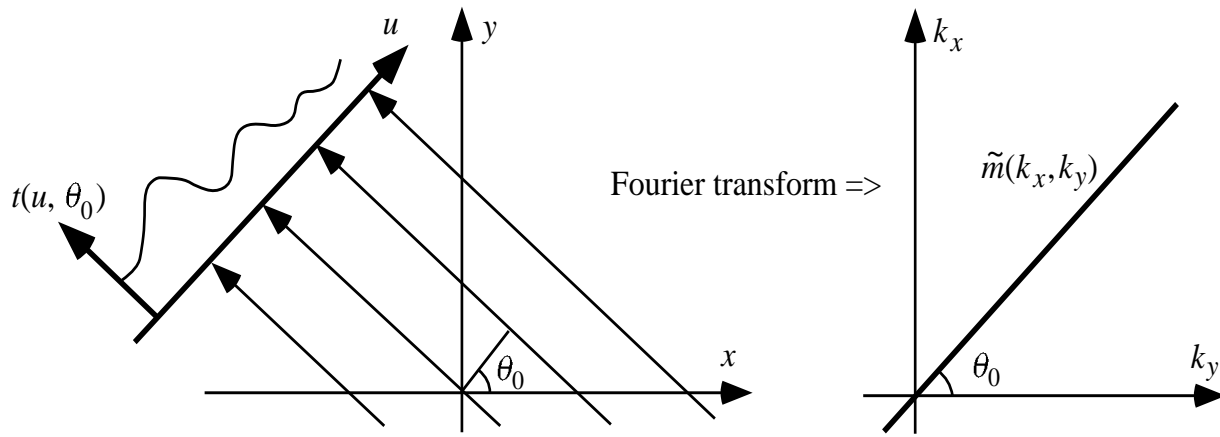
Recognize that the right-hand side is a 2-D Fourier transform:

$$\tilde{m}(k_x, k_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} m(x, y) e^{-2i\pi(k_x x + k_y y)} dx dy \quad (9.66)$$

So

$$\tilde{t}(k_u, \theta) = \tilde{m}(k_u \cos \theta, k_u \sin \theta) \quad (9.67)$$

In words: “The 1-D Fourier transform of the projected data is equal to the 2-D Fourier transform of the model evaluated along a radial line in the  $k_x$ - $k_y$  space with angle  $\theta$ .”

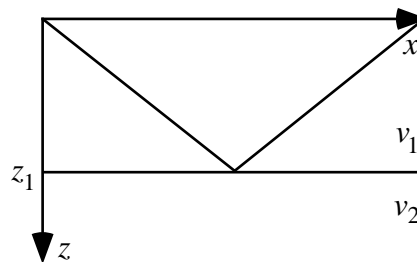


If the Radon transform is known for all values of  $(u, \theta)$ , then the Fourier transform image of  $m$  is known for all values of  $(u, \theta)$ . The model  $m(x, y)$  can then be found by taking the inverse Fourier transform of  $\tilde{m}(k_x, k_y)$ .

Slant-stacking is also a Radon transform. Let  $u(x, t)$  be a wavefield. Then the slant-stack is

$$\bar{u}(p, \tau) = \int u(x, \tau + px) dx \quad (9.68)$$

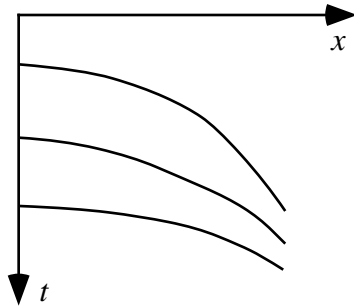
Now, consider a simple example of reflections in a layer.



Travel time ( $t-x$ ) equation is:

$$t^2 v^2 - x^2 = z^2$$

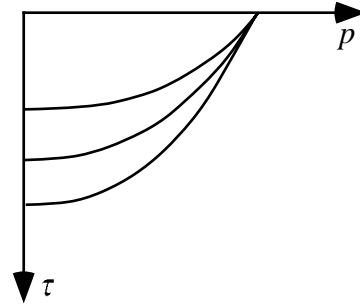
hyperbolas



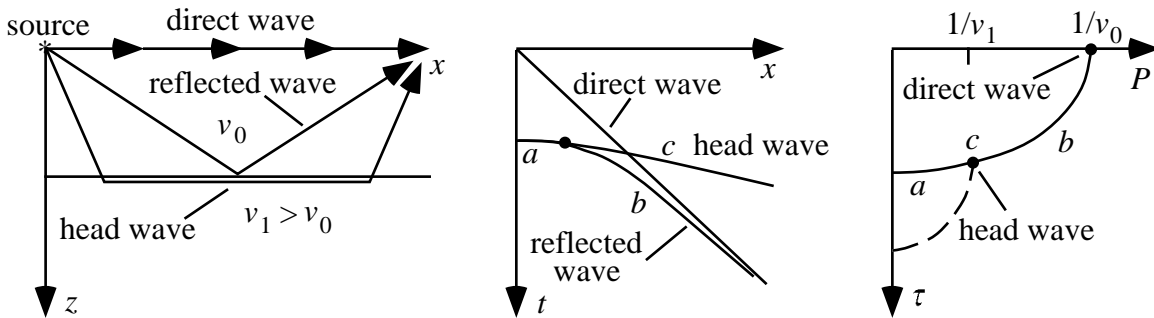
Slant-stack ( $p-\tau$ ) equation is:

$$(\tau / x)^2 + p^2 = 1 / v^2$$

ellipses



With this background, consider how the “complete” wavefield in a layer slant-stack:



The  $p-\tau$  equation can be transformed into  $k-w$  space by a 2-D Fourier transform:

$$U(k, w) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x, t) e^{iwt - ikx} dx dt \tag{9.69}$$

The inverse slant-stack (IRT) is based on the inverse Fourier transform of the above equation,

$$u(x, t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(k, w) e^{ikx} dx e^{-iwt} dw \tag{9.70}$$

Substitute  $k = wp$  and  $dk = w dp$ . Notice that when  $w$  is negative, the integral with respect to  $dp$  is from + to -. To keep the integration in the conventional sense, introduce  $|w|$ .

$$u(x, t) = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} U(wp, w) |w| e^{iwp x} dp \right] e^{-iwt} dw \tag{9.71}$$

Changing the order of integration,

$$u(x, t) = \int_{-\infty}^{\infty} \left\{ \left[ \int_{-\infty}^{\infty} U(wp, w) e^{iwp x} |w| \right] e^{-iwt} dw \right\} dp \quad (9.72)$$

The term in { } contains an inverse Fourier transform of a product of three functions of  $w$ .

1.  $U(wp, w)$  is the Fourier transform of the slant-stack evaluated at  $k = wp$ . So, the slant-stack is given by

$$\bar{u}(p, \tau) = \int_{-\infty}^{\infty} e^{-i w \tau} U(wp, w) dw \quad (9.73)$$

2. The  $e^{iwp x}$  term can be thought of a delay operator where  $\tau = t - px$ .
3. The  $|w|$  term is called the “rho” filter.

Now we can rewrite the above equation as

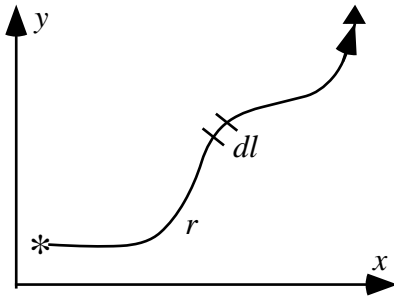
$$u(x, t) = \text{rho}(t) * \int \bar{u}(p, t - px) dp \quad \text{Inverse Radon Transform} \quad (9.74)$$

Compare with

$$\bar{u}(x, \tau) = \int u(x, \tau + px) dx \quad \text{Radon Transform} \quad (9.43)$$

### 9.6 Alternative Approach to Tomography

An alternative approach to tomography is to discretize the travel time equation, as follows.



$$m(x, y) = \frac{1}{v(x, y)} \quad (9.75)$$

$$t = \int_r m dl \quad (9.76)$$

With some assumptions, we can linearize and discretize this equation to

$$t_r = \sum_b l_{rb} m_b \quad (9.77)$$

- where
- $t_r$  = travel time of the  $r^{\text{th}}$  ray
  - $m_b$  = slowness of the  $b^{\text{th}}$  block
  - $l_{rb}$  = length of the  $r^{\text{th}}$  ray segment in the  $b^{\text{th}}$  block.

In matrix form,

$$\mathbf{d} = \mathbf{Gm} \quad \text{Aha!!} \quad (1.13)$$

We know how to solve this equation. But what if we have a 3-D object with 100 blocks on a side? Then  $M \approx (100)^3 = 10^6$ , and even  $\mathbf{G}^T\mathbf{G}$  is a matrix with  $M^2$  elements, or  $\sim 10^{12}$ . Try throwing that into your MATLAB program. So what can we do?

Let's start at the beginning again.

$$\mathbf{d} = \mathbf{Gm} \quad (1.13)$$

$$\mathbf{G}^T\mathbf{d} = \mathbf{G}^T\mathbf{Gm} \quad (9.78)$$

$$| \mathbf{R} |$$

In a sense,  $\mathbf{G}^T$  is an approximate inverse operator that transforms a data vector into model space. Also, in this respect,  $\mathbf{G}^T\mathbf{G}$  can be thought of as a resolution matrix that shows you the “filter” between your estimate of the model,  $\mathbf{G}^T\mathbf{d}$ , and the real model,  $\mathbf{m}$ . Since the ideal  $\mathbf{R} = \mathbf{I}$ , let's try inverting Equation (9.78) by using only the diagonal elements of  $\mathbf{G}^T\mathbf{G}$ . Then the solution can be computed simply.

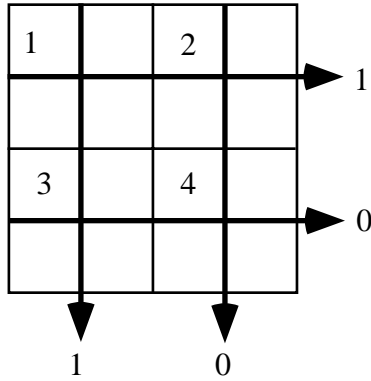
$$\begin{bmatrix} \sum_{i=1}^N l_{i_1} t_i \\ \sum_{i=1}^N l_{i_2} t_i \\ \vdots \\ \sum_{i=1}^N l_{i_M} t_i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N l_{i_1}^2 & 0 & 0 & 0 \\ 0 & \sum_{i=1}^N l_{i_2}^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sum_{i=1}^N l_{i_M}^2 \end{bmatrix} \mathbf{m} \quad (9.79)$$

so

$$m_b = \frac{\sum_{i=1}^N l_{ib} t_i}{\sum_{i=1}^N l_{ib}^2} \quad \text{"tomographic approximation"} \quad (9.80)$$

Operationally, each ray is back-projected, and at each block a ray hits, the contributions to the two sums are accumulated in two vectors. At the end, the contents of each element of the numerator vector are divided by each element of the denominator vector. This reduces storage requirements to  $\sim 2M$  instead of  $M^2$ .

Let's see how this works for our simple tomography problem.



$$\text{for } \mathbf{m} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (9.81)$$

$$\mathbf{m}_{ML} = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \mathbf{G}^T \mathbf{d} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (9.82)$$

Fits data

$$\mathbf{G}^T \mathbf{G} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \quad [\mathbf{G}^T \mathbf{G}]_d = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad (9.83)$$

The approximate equation is

$$\begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \mathbf{m} \Rightarrow \tilde{\mathbf{m}} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix} \quad (9.84)$$

Not bad, but this does not predict the data. Can we improve on this? Yes! Following Menke, develop an iterative solution.

$$[\mathbf{G}^T \mathbf{G}] \mathbf{m} = \mathbf{G}^T \mathbf{d} \quad (9.85)$$

$$[\mathbf{I} - \mathbf{I} + \mathbf{G}^T \mathbf{G}] \mathbf{m} = \mathbf{G}^T \mathbf{d} \quad (9.86)$$

$$\mathbf{m} - [\mathbf{I} - \mathbf{G}^T \mathbf{G}] \mathbf{m} = \mathbf{G}^T \mathbf{d} \quad (9.87)$$

so

$$\mathbf{m}^{\text{est}(i)} = \mathbf{G}^T \mathbf{d} + [\mathbf{I} - \mathbf{G}^T \mathbf{G}] \mathbf{m}^{\text{est}(i-1)} \quad (9.88)$$

[Menke's Equations (11) and (24)]

A slightly different iterative technique,

$$\mathbf{m}^{(k)} = \mathbf{D}^{-1}\mathbf{G}^T\mathbf{d} + [\mathbf{D} - \mathbf{G}^T\mathbf{G}]\mathbf{m}^{(k-1)} \quad (9.89)$$

where  $\mathbf{D}$  is the diagonalized  $\mathbf{G}^T\mathbf{G}$  matrix. Then for  $\mathbf{m}^0 = 0$ ,  $\mathbf{m}^1 = \mathbf{D}^{-1}\mathbf{G}^T\mathbf{d}$  is the “tomographic approximation” solution.

These are still relatively “rudimentary” iterative techniques. They do not always converge. Ideally, we want iterative techniques to converge to the least squares solution. A number of such techniques exist and have been used in “tomography.” They include the “simultaneous iterative reconstruction techniques,” or SIRT, and LSQR. See the literature.

This chapter is only an introduction to continuous inverse theory, neural networks, and the Radon transform. The references cited at the beginning of the chapter are an excellent place to begin a deeper study of continuous inverse problems. The Backus–Gilbert approach provides one way to handle the construction (i.e., finding a solution) and appraisal (i.e., what do we really know about the solution) phases of a continuous inverse analysis. In this sense, these are tasks that were covered in detail in the chapters on discrete inverse problems. For all the division between continuous and discrete inverse practitioners (continuous inverse types have been known to look down their noses at discrete types saying that they're not doing an *inverse* analysis, but rather parameter estimation; conversely, discrete types sneer and say that continuous applications are too esoteric and don't really work very often anyway), we have shown in this chapter that the goals of constructing (finding) a solution and appraising it are universal between the two approaches. We hope to add more material on these topics Real Soon Now.

## References

Claerbout, J. F., *Imaging the Earth's Interior*, 398 pp., Blackwell Scientific Publications, Oxford, UK, 1985.