

Describing Inverse Problems

1.1 FORMULATING INVERSE PROBLEMS

The starting place in most inverse problems is a description of the data. Since in most inverse problems the data are simply a list of numerical values, a vector provides a convenient means of their representation. If N measurements are performed in a particular experiment, for instance, one might consider these numbers as the elements of a vector \mathbf{d} of length N .

The purpose of data analysis is to gain *knowledge* through systematic examination of data. While knowledge can take many forms, we assume here that it is primarily numerical in nature. We analyze data so to infer, as best we can, the values of numerical quantities—*model parameters*. Model parameters are chosen to be *meaningful*; that is, they are chosen to capture the essential character of the processes that are being studied. The model parameters can be represented as the elements of a vector \mathbf{m} , which is of length M

$$\begin{aligned} \text{data: } \mathbf{d} &= [d_1, d_2, d_3, d_4, \dots, d_N]^T \\ \text{model parameters: } \mathbf{m} &= [m_1, m_2, m_3, m_4, \dots, m_M]^T \end{aligned} \quad (1.1)$$

Here, T signifies transpose.

The basic statement of an inverse problem is that the model parameters and the data are in some way related. This relationship is called the *quantitative model* (or *model*, or *theory*, for short). Usually, the model takes the form of one or more formulas that the data and model parameters are expected to follow.

If, for instance, one were attempting to determine the density of an object, such as a rock, by measuring its mass and volume, there would be $N=2$ data—mass and volume (say, d_1 and d_2 , respectively)—and $M=1$ unknown model parameter, density (say, m_1). The model would be the statement that density times volume equals mass, which can be written compactly by the vector equation $d_2 m_1 = d_1$. Note that the model parameter, density, is more meaningful than either mass or volume, in that it represents an intrinsic property of a substance that is related to its chemistry. The data—mass and volume—are easy to measure, but they are less fundamental because they depend on the size of the object, which is usually incidental.

In more realistic situations, the data and model parameters are related in more complicated ways. Most generally, the data and model parameters might be related by one or more implicit equations such as

$$\begin{aligned} f_1(\mathbf{d}, \mathbf{m}) &= 0 \\ f_2(\mathbf{d}, \mathbf{m}) &= 0 \\ &\vdots \\ f_L(\mathbf{d}, \mathbf{m}) &= 0 \end{aligned} \quad \text{or} \quad \mathbf{f}(\mathbf{d}, \mathbf{m}) = 0 \quad (1.2)$$

where L is the number of equations. In the above example concerning the measuring of density, $L = 1$ and $d_2 m_1 - d_1 = 0$ would constitute the one equation of the form $f_1(\mathbf{d}, \mathbf{m}) = 0$. These implicit equations, which can be compactly written as the vector equation $\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0$, summarize what is known about how the measured data and the unknown model parameters are related. The purpose of inverse theory is to solve, or “invert,” these equations for the model parameters, or whatever kinds of answers might be possible or desirable in any given situation.

No claims are made either that the equations $\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0$ contain enough information to specify the model parameters uniquely or that they are even consistent. One of the purposes of inverse theory is to answer these kinds of questions and provide means of dealing with the problems that they imply. In general, $\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0$ can consist of arbitrarily complicated (nonlinear) functions of the data and model parameters. In many problems, however, the equation takes on one of several simple forms. It is convenient to give names to some of these special cases, since they commonly arise in practical problems; we shall give them special consideration in later chapters.

1.1.1 Implicit Linear Form

The function \mathbf{f} is linear in both data and model parameters and can therefore be written as the matrix equation

$$\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0 = \mathbf{F} \begin{bmatrix} \mathbf{d} \\ \mathbf{m} \end{bmatrix} = \mathbf{F}\mathbf{x} \quad (1.3)$$

where \mathbf{F} is an $L \times (M + N)$ matrix and the vector $\mathbf{x} = [\mathbf{d}^T, \mathbf{m}^T]^T$ is a concatenation of \mathbf{d} and \mathbf{m} , that is, $\mathbf{x} = [d_1, d_2, \dots, d_N, m_1, m_2, \dots, m_M]^T$.

1.1.2 Explicit Form

In many instances, it is possible to separate the data from the model parameters and thus to form $L = N$ equations that are linear in the data (but still nonlinear in the model parameters through a vector function \mathbf{g}).

$$\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0 = \mathbf{d} - \mathbf{g}(\mathbf{m}) \quad (1.4)$$

1.1.3 Explicit Linear Form

In the explicit linear form, the function \mathbf{g} is also linear, leading to the $N \times M$ matrix equation (where $L=N$)

$$\mathbf{f}(\mathbf{d}, \mathbf{m}) = 0 = \mathbf{d} - \mathbf{G}\mathbf{m} \quad (1.5)$$

This form is equivalent to a special case of the matrix \mathbf{F} in [Section 1.1.1](#):

$$\mathbf{F} = [\mathbf{I}, -\mathbf{G}] \quad (1.6)$$

1.2 THE LINEAR INVERSE PROBLEM

The simplest and best-understood inverse problems are those that can be represented with the explicit linear equation $\mathbf{G}\mathbf{m} = \mathbf{d}$. This equation, therefore, forms the foundation of the study of discrete inverse theory. As will be shown below, many important inverse problems that arise in the physical sciences involve precisely this equation. Others, while involving more complicated equations, can often be solved through linear approximations.

The matrix \mathbf{G} is called the data kernel, in analogy to the theory of integral equations, in which the analogs of the data and model parameters are two continuous functions $d(x)$ and $m(x)$, where x is some independent variable. Continuous inverse theory lies between these two extremes, with discrete data but a continuous model function.

Discrete inverse theory:

$$d_i = \sum_{j=1}^M G_{ij} m_j \quad (1.7a)$$

Continuous inverse theory:

$$d_i = \int G_i(x) m(x) dx \quad (1.7b)$$

Integral equation theory:

$$d(y) = \int G(y, x) m(x) dx \quad (1.7c)$$

The main difference among discrete inverse theory, continuous inverse theory, and integral equation theory is whether the model m and data d are treated as continuous functions or discrete parameters. The data d_i in inverse theory are necessarily discrete, since inverse theory is concerned with deducing knowledge from observational data, which always has a discrete nature. Both continuous inverse problems and integral equations can be converted to discrete

inverse problems by approximating the model $m(x)$ as a vector of its values at a set of M closely spaced points

$$\mathbf{m} = [m(x_1), m(x_2), m(x_3), \dots, m(x_M)]^T \quad (1.8)$$

and the integral as a Riemann summation (or by some other quadrature formula).

1.3 EXAMPLES OF FORMULATING INVERSE PROBLEMS

1.3.1 Example 1: Fitting a Straight Line

Suppose that N temperature measurements T_i are made at times t_i in the atmosphere (Figure 1.1). The data are then a vector \mathbf{d} of N measurements of temperature, where $\mathbf{d} = [T_1, T_2, T_3, \dots, T_N]^T$. The times t_i are not, strictly speaking, data. Instead, they provide some auxiliary information that describes the geometry of the experiment. This distinction will be further clarified below.

Suppose that we assume a model in which temperature is a linear function of time: $T = a + bt$. The intercept a and slope b then form the two model parameters

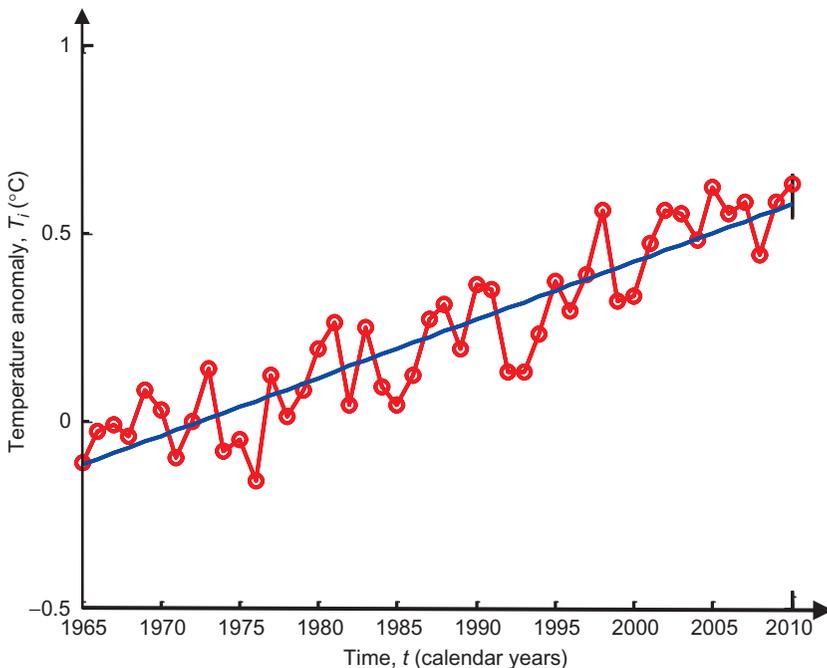


FIGURE 1.1 (Red) Average global temperature for the time period, 1965–2010. The inverse problem is to determine the rate of increase of temperature and its confidence interval. (Blue) Straight line fit to data. The slope of the line is 0.015 ± 0.002 (95%) $^{\circ}\text{C}/\text{year}$. Data from Hansen et al. 2010. *MatLab* script gda01_01.

of the problem, $\mathbf{m} = [a, b]^T$. According to the model, each temperature observation must satisfy $T_i = a + bt_i$:

$$\begin{aligned} T_1 &= a + bt_1 \\ T_2 &= a + bt_2 \\ &\vdots \\ T_N &= a + bt_N \end{aligned} \quad (1.9)$$

These equations can be arranged as the matrix equation $\mathbf{d} = \mathbf{Gm}$

$$\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{bmatrix} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_N \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \quad (1.10)$$

In *MatLab*, the matrix \mathbf{G} is computed as:

$$\mathbf{G} = [\text{ones}(N, 1), \mathbf{t}];$$

(*MatLab* script gda01_01)

1.3.2 Example 2: Fitting a Parabola

If the model in Example 1 is changed to assume a quadratic variation of temperature with depth of the form $T = a + bt + ct^2$, then a new model parameter c is added to the problem, and $\mathbf{m} = [a, b, c]^T$. The number of model parameters is now $M = 3$. The data and model parameters are supposed to satisfy

$$\begin{aligned} T_1 &= a + bt_1 + ct_1^2 \\ T_2 &= a + bt_2 + ct_2^2 \\ &\vdots \\ T_N &= a + bt_N + ct_N^2 \end{aligned} \quad (1.11)$$

These equations can be arranged into the matrix equation

$$\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (1.12)$$

This matrix equation has the explicit linear form $\mathbf{d} = \mathbf{Gm}$. Note that, although the equation is linear in the data and model parameters, it is not linear in the auxiliary variable t .

The equation has a very similar form to the equation of the previous example, which brings out one of the underlying reasons for employing matrix notation: it can often emphasize similarities between superficially different problems. In *MatLab*, the matrix \mathbf{G} is computed as:

$$\mathbf{G} = [\text{ones}(N, 1), \mathbf{t}, \mathbf{t}.^2];$$

(*MatLab* script gda01_02)

Note the use of the element-wise power, signified “ $.$ ” to compute t_i^2 .

1.3.3 Example 3: Acoustic Tomography

Suppose that a wall is assembled from a rectangular array of bricks (Figure 1.2) and that each brick is composed of a different type of clay. If the acoustic velocities of the different clays differ, one might attempt to distinguish the different kinds of bricks by measuring the travel time of sound across the various rows and columns of bricks in the wall. The data in this problem are $N=8$ measurements of travel times, $d=[T_1, T_2, T_3, \dots, T_8]^T$. The model assumes that each brick is composed of a uniform material and that the travel time of sound across each brick is proportional to the width and height of the brick. The proportionality factor is the brick's *slowness* s_i , thus giving $M=16$ model parameters $\mathbf{m}=[s_1, s_2, s_3, \dots, s_{16}]^T$, where the ordering is according to the numbering scheme of the figure. The data and model parameters are related by

$$\begin{aligned} \text{row 1 : } T_1 &= hs_1 + hs_2 + hs_3 + hs_4 \\ \text{row 2 : } T_2 &= hs_5 + hs_6 + hs_7 + hs_8 \\ &\vdots \\ \text{column 4 : } T_8 &= hs_4 + hs_8 + hs_{12} + hs_{16} \end{aligned} \quad (1.13)$$

and the matrix equation is

$$\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_8 \end{bmatrix} = h \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{16} \end{bmatrix} \quad (1.14)$$

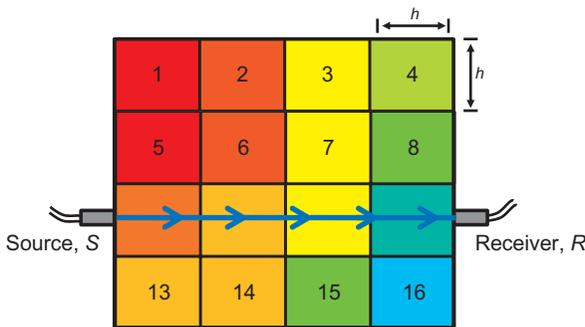


FIGURE 1.2 The travel time of acoustic waves (blue line) through the rows and columns of a square array of bricks is measured with acoustic source S and receiver R placed on the edges of the square. The inverse problem is to infer the acoustic properties of the bricks, here depicted by the colors. Although the overall pattern is spatially variable, individual bricks are assumed to be homogeneous.

Here, the bricks are assumed to be of width *and* height h . The *MatLab* code for constructing \mathbf{G} is:

```
G=zeros(N,M);
for i = [1:4]
for j = [1:4]
% measurements over rows
k = (i-1)*4 + j;
G(i,k)=h;
% measurements over columns
k = (j-1)*4 + i;
G(i+4,k)=h;
end
end
```

(*MatLab* script gda01_03)

1.3.4 Example 4: X-ray Imaging

Tomography is the process of forming images of the interior of an object from measurements made along rays passed through that object (“tomo” comes from the Greek word for “slice”). The computerized tomography scanner is an X-ray imaging device that has revolutionized the diagnosis of brain tumors and many other medical conditions. The scanner solves an inverse problem for the X-ray opacity of body tissues using measurements of the amount of radiation absorbed from many crisscrossing beams of X-rays (Figure 1.3).

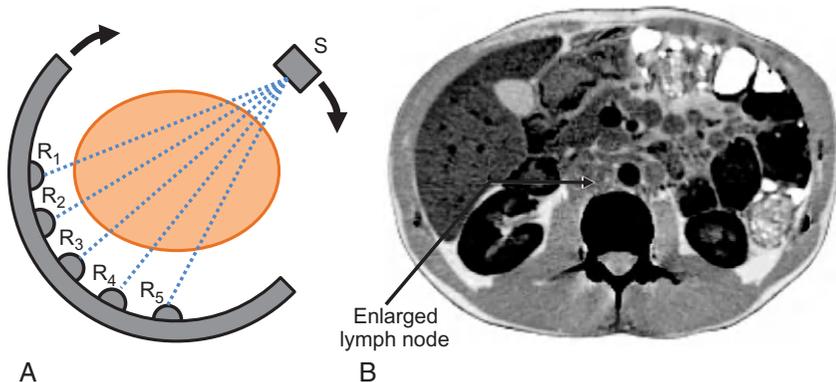


FIGURE 1.3 (A) An idealized computed tomography (CT) medical scanner measures the X-ray absorption along lines (blue) passing through the body of the patient (orange). After a set of measurements are made, the source S and receivers R_i are rotated, and the measurements are repeated so that data along many crisscrossing lines are collected. The inverse problem is to determine the X-ray opacity as a function of position in the body. (B) Actual CT image of a patient infected with *Mycobacterium genavense* (from de Lastours et al., 2008).

The basic physical model underlying this device is the idea that the intensity of X-rays diminishes with the distance traveled, at a rate proportional to the intensity of the X-ray beam, and an absorption coefficient that depends on the type of tissue:

$$\frac{dI}{ds} = -c(x, y) I \quad (1.15)$$

Here, I is the intensity of the beam, s the distance along the beam, and $c(x, y)$ the absorption coefficient, which varies with position. If the X-ray source has intensity I_0 , then the intensity at the i th detector is

$$I_i = I_0 \exp\left\{-\int_{\text{beam } i} c(x, y) ds\right\} \approx I_0 \left\{1 - \int_{\text{beam } i} c(x, y) ds\right\} \quad (1.16a)$$

$$I_0 - I_i = I_0 \int_{\text{beam } i} c(x, y) ds \quad (1.16b)$$

Note that Equation (1.16a) is a nonlinear function of the unknown absorption coefficient $c(x, y)$ and that the absorption coefficient varies continuously along the beam. This is a nonlinear problem in continuous inverse theory. However, it can be linearized, for small net absorption, by approximating the exponential with the first two terms in its Taylor series expansion, that is, $\exp(-x) \approx 1 - x$.

We now convert this problem to a discrete inverse problem of the form $\mathbf{Gm} = \mathbf{d}$. We assume that the continuously varying absorption coefficient can be adequately represented by a grid of many small square boxes (or *pixels*), each of which has a constant absorption coefficient. With these pixels numbered 1 through M , the model parameters are then the vector $\mathbf{m} = [c_1, c_2, c_3, \dots, c_M]^T$. The integral can then be written as the sum

$$\Delta I_i = \frac{I_0 - I_i}{I_0} = \sum_{j=1}^M \Delta s_{ij} c_j \quad (1.17)$$

Here, the data $d_i = \Delta I_i$ represent the differences between the X-ray intensities at the source and at the detector, and $G_{ij} = \Delta s_{ij}$ is the distance the i th beam travels in the j th pixel.

The inverse problem can then be summarized by the matrix equation

$$\begin{bmatrix} \Delta I_1 \\ \Delta I_2 \\ \vdots \\ \Delta I_N \end{bmatrix} = \begin{bmatrix} \Delta s_{11} & \Delta s_{12} & \cdots & \Delta s_{1M} \\ \Delta s_{21} & \Delta s_{22} & \cdots & \Delta s_{2M} \\ \cdots & \cdots & \cdots & \cdots \\ \Delta s_{N1} & \Delta s_{N2} & \cdots & \Delta s_{NM} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_M \end{bmatrix} \quad (1.18)$$

Since each beam passes through only a few of the many boxes, many of the Δs_{ij} are zero. Such a matrix is said to be *sparse*.

Computations with sparse matrices can be made extremely efficient by storing only the nonzero elements and by never explicitly multiplying or adding the zero elements (since the result is a foregone conclusion). However, special software support is necessary to gain this efficiency, since the computer must keep track of the zero elements. In *MatLab*, matrices need to be declared as sparse:

```
G = spalloc( N, M, MAXNONZEROELEMENTS );
(MatLab script gda01_03)
```

Once so defined, many normal matrix operations, including addition and multiplication, are efficiently computed without further user intervention. We will discuss this technique further in subsequent chapters, for its use makes practical the solving of very large inverse problems (say, with millions of model parameters). Further examples are given in [Menke and Menke \(2011\)](#).

1.3.5 Example 5: Spectral Curve Fitting

Not every inverse problem can be adequately represented by the discrete linear equation $\mathbf{Gm} = \mathbf{d}$. Consider, for example, a spectrogram containing a set of emission or absorption peaks, that vary with some auxiliary variable z ([Figure 1.4](#)). The positions f , area A , and width c of the peaks are of interest because they

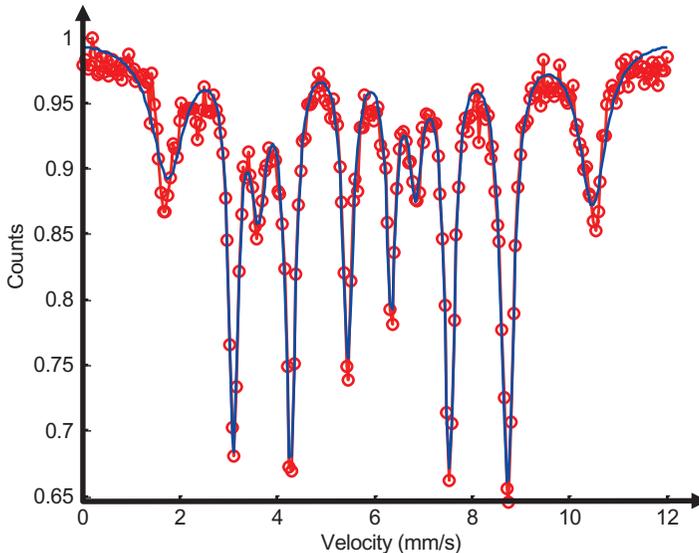


FIGURE 1.4 Example of a Mossbauer spectroscopy experiment performed by the *Spirit* rover on Martian soil. (Red) Absorption peaks reflect the concentration of different iron-bearing minerals in the soil. The inverse problem is to determine the position and area of each peak, which can be used to determine the concentration of the minerals. (Blue) The sum of ten Lorentzian curves fit to the data. Data courtesy of NASA and the University of Mainz. *MatLab* script gda01_04.

reflect the chemical composition of the sample. Denoting the shape of peak j as $p(z, f_j, A_j, c_j)$, the model is that the spectrum consists of a sum of q such peaks

$$d_i = \sum_{j=1}^q p(z_i, f_j, A_j, c_j) = \sum_{j=1}^q \frac{A_j c_j^2}{(z_i - f_j)^2 + c_j^2} \quad (1.19)$$

Here, the peak shape $p(z, f_j, A_j, c_j)$ is taken to be a *Lorentzian*. The data and model are therefore related by the *nonlinear* explicit equation $\mathbf{d} = \mathbf{g}(\mathbf{m})$, where \mathbf{m} is a vector of the position, area, and width of each peak. This equation is inherently nonlinear.

1.3.6 Example 6: Factor Analysis

Another example of a nonlinear inverse problem is that of determining the composition of chemical end members on the basis of the chemistry of a suite of mixtures of the end members. Consider a simplified “ocean” (Figure 1.5) in which sediments are composed of mixtures of several chemically distinct rocks eroded from the continents. One expects the fraction of chemical j in the i th sediment sample S_{ij} to be related to the amount of end-member rock in sediment sample i (C_{ik}) and to the amount of the j th chemical in the end-member rock (F_{kj}) as

$$\begin{bmatrix} \text{sample} \\ \text{composition} \end{bmatrix} = \sum_{\text{end members}} \begin{bmatrix} \text{amount of} \\ \text{end member} \end{bmatrix} \begin{bmatrix} \text{end member} \\ \text{composition} \end{bmatrix} \quad (1.20)$$

$$S_{ij} = \sum_{k=1}^p C_{ik} F_{kj} \quad \text{or} \quad \mathbf{S} = \mathbf{C}\mathbf{F}$$

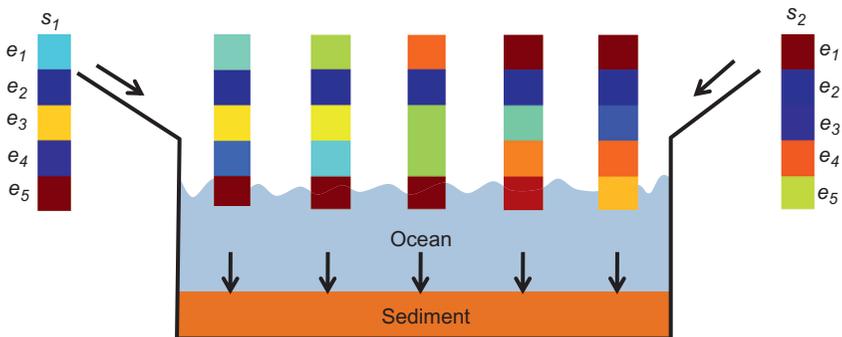


FIGURE 1.5 Sediment on the floor of this idealized ocean is a mixture of rocks eroded from several sources s_j . The sources are characterized by chemical elements, e_1 through e_5 , depicted here with color bars. The chemical composition of the sediments is a simple mixture of the composition of the sources. The inverse problem is to determine the number and composition of sources from observations of the composition of the sediments. *MatLab* script gda01_05.

In a typical experiment, the number of end members p , the end-member composition \mathbf{F} , and the amount of end members in the samples \mathbf{C} are all unknown model parameters. Since the data \mathbf{S} are on one side of the equations, this problem is also of the explicit nonlinear type. Note that basically the problem is to factor a matrix \mathbf{S} into two other matrices \mathbf{C} and \mathbf{F} . This factoring problem is a well-studied part of the theory of matrices, and methods are available to solve it. As will be discussed in [Chapter 10](#), this problem (which is often called *factor analysis*) is very closely related to the algebraic eigenvalue problem.

1.4 SOLUTIONS TO INVERSE PROBLEMS

We shall use the terms *solution* and *answer* to indicate broadly whatever information we are able to determine about the problem under consideration. As we shall see, there are many different points of view regarding what constitutes a solution to an inverse problem. Of course, one generally wants to know the numerical values of the model parameters (we call this kind of answer an *estimate* of the model parameters). Unfortunately, this issue is made complicated by the ubiquitous presence of measurement error and also by the possibility that some model parameters are not constrained by any observation. The solution of inverse problems rarely leads to exact information about the values of the model parameters. More typically, the practitioner of inverse theory is forced to make various compromises between the kind of information he or she actually wants and the kind of information that can in fact be obtained from any given data set. These compromises lead to other kinds of “answers” that are more abstract than simple estimates of the model parameters. Part of the practice of inverse theory is identifying what features of a solution are most valuable and making the compromises that emphasize these features. Some of the possible forms an “answer” to an inverse problem might take are described below.

1.4.1 Estimates of Model Parameters

The simplest kind of solution to an inverse problem is an estimate \mathbf{m}^{est} of the model parameters. An estimate is simply a set of numerical values for the model parameters, $\mathbf{m}^{\text{est}} = [1.4, 2.9, \dots, 1.0]^T$, for example. Estimates are generally the most useful kind of solution to an inverse problem. Nevertheless, in many situations, they can be very misleading. For instance, estimates in themselves give no insight into the quality of the solution. Depending on the structure of the particular problem, measurement errors might be averaged out (in which case the estimates might be meaningful) or amplified (in which case the estimates might be nonsense). In other problems, many solutions might exist. To single out arbitrarily only one of these solutions and call it \mathbf{m}^{est} gives the false impression that a unique solution has been obtained.

1.4.2 Bounding Values

One remedy to the problem of defining the quality of an estimate is to state additionally some bounds that define its certainty. These bounds can be either absolute or probabilistic. Absolute bounds imply that the true value of the model parameter lies between two stated values, for example, $1.3 \leq m_1 \leq 1.5$. Probabilistic bounds imply that the estimate is likely to be between the bounds, with some given degree of certainty. For instance, $m_1^{\text{est}} = 1.4 \pm 0.1$ (95%) might mean that there is a 95% probability that the true value of the model parameter m_1^{true} lies between 1.3 and 1.5.

When they exist, bounding values can often provide the supplementary information needed to interpret properly the solution to an inverse problem. There are, however, many instances in which bounding values do not exist.

1.4.3 Probability Density Functions

A generalization of the stating of bounding values is the stating of the complete probability density function $p(\mathbf{m})$ for model parameters, either as an analytic function or as values on an M -dimensional grid. The usefulness of this technique depends, in part, on the complexity of $p(\mathbf{m})$. If the probability density functions $p(m_i)$ for an individual model parameter m_i has only one peak (Figure 1.6A), then it provides little more information than an estimate based on the position of the peak's center with error bounds based on the peak's shape. On the other hand, if the probability density function is very complicated (Figure 1.6C), it is basically uninterpretable (except in the sense that it implies that the model parameter cannot be well estimated). Only in those exceptional instances in which it has some intermediate complexity (Figure 1.6B) does it really provide information toward the solution of an inverse problem.

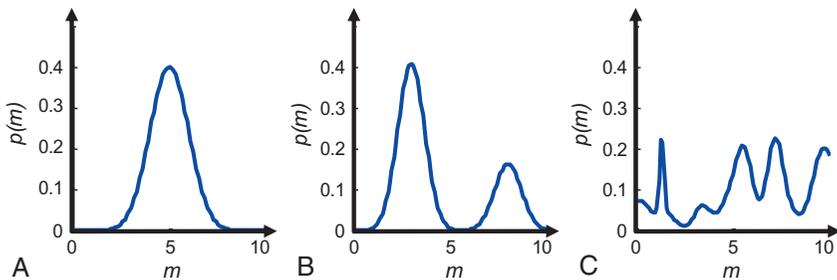


FIGURE 1.6 Three hypothetical probability density functions for a model parameter, m . (A) The first is so simple that its properties can be summarized by its central position, at $m=5$, and the width of its peak. (B) The second implies that the model parameter has two probable ranges of values, one near $m=3$ and the other near $m=8$. (C) The third is so complicated that it provides no easily interpretable information about the model parameter. *MatLab* script gda01_06.

1.4.4 Sets of Realizations of Model Parameters

Except in the well-understood Gaussian (Normal) case, which we will discuss later in the book, most probability density functions are exceedingly difficult to compute. A large set of realizations $\mathbf{m}^{(i)}$ of model parameter vectors drawn from $p(\mathbf{m})$ is somewhat easier to compute and can serve as an alternative. The set, itself, might be considered the solution to the inverse problem, since many of the properties of the probability density function can be inferred from it. However, this set might need to be extremely large to capture the properties of $p(\mathbf{m})$, especially when the number M of model parameters is large. Deriving useful knowledge from, say, a billion examples of possible \mathbf{m} s is a challenging task.

1.4.5 Weighted Averages of Model Parameters

In many instances, it is possible to identify combinations or averages of the model parameters that are in some sense better determined than the model parameters themselves. For instance, given $\mathbf{m} = [m_1, m_2]^T$, it may turn out that $\langle m \rangle = 0.2m_1 + 0.8m_2$ is better determined than either m_1 or m_2 . Unfortunately, one might not have the slightest interest in such an average, be it well determined or not, because it may not have physical significance. It may not contribute useful knowledge.

Averages *can* be of considerable interest when the model parameters represent a discretized version of some continuous function. If the weights are large only for a few physically adjacent parameters, then the average is said to be *localized*. The meaning of the average in such a case is that, although the data cannot resolve the model parameters at a particular point, they can resolve the average of the model parameters in the *neighborhood* of that point.

In the following chapters, we shall derive methods for determining each of these different kinds of solutions to inverse problems. We note here, however, that there is a great deal of underlying similarity between these types of “answers.” In fact, it will turn out that the same numerical “answer” will be interpretable as any of several classes of solutions.

1.5 PROBLEMS

- 1.1** Suppose that you determine the masses of 100 objects by weighing the first, then weighing the first and second together, and then weighing the rest in triplets: the first, second, and third; the second, third, and fourth; and so forth. (A) Identify the data and model parameters in this problem. How many of each are there? (B) Write down the matrix \mathbf{G} in the equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ that relates the data to the model parameters. (C) How sparse is \mathbf{G} ? What percent of it is zero?
- 1.2** Suppose that you determine the height of 50 objects by measuring the first, and then stacking the second on top of the first and measuring their combined height, stacking the third on top of the first two and measuring their

- combined height, and so forth. (A) Identify the data and model parameters in this problem. How many of each are there? (B) Write down the matrix \mathbf{G} in the equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ that relates the data to the model parameters. (C) How sparse is \mathbf{G} ? What percent of it is zero?
- 1.3** Write a *MatLab* script to compute \mathbf{G} in the case of the cubic equation, $T = a + bz + cz^2 + dz^3$. Assume that 11 z s are equally spaced from 0 to 10.
- 1.4** Let the data \mathbf{d} be the running average of the model parameters, \mathbf{m} , computed by averaging groups of three neighboring points; that is, $d_i = (m_{i-1} + m_i + m_{i+1})/3$. (A) What is the matrix \mathbf{G} in the equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ in this case? (B) What problems arise at the top and bottom rows of the matrix and how can you deal with them? (C) How sparse is \mathbf{G} ? What percent of it is zero?
- 1.5** Simplify Equation (1.20) by assuming that there is only one sample S_{1j} whose composition is measured. Consider the case where the composition of the p factors is known, but their proportions in the sample are unknown, and rewrite Equation (10.2) in the form $\mathbf{d} = \mathbf{G}\mathbf{m}$. Hint: You might start by taking the transpose of Equation (1.20).

REFERENCES

- de Lastours, V., Guillemin, R., Mainardi, J.-L., Aubert, A., Chevalier, P., Lefort, A., Podglajen, I., 2008. Early diagnosis of disseminated *Mycobacterium genavense* infection. *Emerg. Infect. Dis.* 14 (2), 346.
- Hansen, J., Ruedy, R., Sato, M., Lo, K., 2010. Global surface temperature change. *Rev. Geophys.* 48, RG4004, doi:10.1029/2010RG000345.
- Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc., Oxford, UK 263pp.

Some Comments on Probability Theory

2.1 NOISE AND RANDOM VARIABLES

In the preceding chapter, we represented the results of an experiment as a vector \mathbf{d} whose elements were individual measurements. Usually, however, a single number is insufficient results of an experiment represented the \mathbf{d} whose elements as a vector to represent a single observation. Measurements contain noise, and if an observation were to be performed several times, each measurement would be different (Figure 2.1). Information about the range and shape of this scatter must also be provided to characterize the data completely.

The concept of a *random variable* is used to describe this property. Each random variable has definite and precise properties, governing the range and shape of the scatter of values one observes. These properties cannot be measured directly; however, one can only make individual measurements, or *realizations*, of the random variable and try to estimate its true properties from these data.

The true properties of the random variable d are specified by a *probability density function* $p(d)$ (abbreviated *p.d.f.*). This function gives the probability that a particular realization of the random variable will have a value in the neighborhood of d . The probability that the measurement is between d and $d + dd$ is $p(d) dd$ (Figure 2.2). (Our choice of the variable name “ d ” for “data” makes the differential “ dd ” look a bit funny, but we will just have to live with it.)

Since each measurement must have some value, the probability that d lies somewhere between $-\infty$ and $+\infty$ is complete certainty (usually given the value of 100% or unity), which is written as

$$\int_{-\infty}^{+\infty} p(d)dd = 1 \quad (2.1)$$

The probability P that d lies in some specific range, say between d_1 and d_2 , is the integral of $p(d)$ over that range:

$$P(d_1, d_2) = \int_{d_1}^{d_2} p(d)dd \quad (2.2)$$

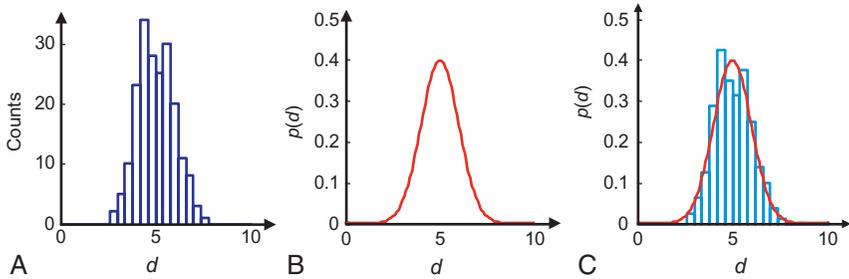


FIGURE 2.1 (A) Histogram showing data from 200 repetitions of an experiment in which datum d is measured. Noise causes observations to scatter about their mean value, $\langle d \rangle = 5$. (B) Probability density function (p.d.f.), $p(d)$, of the data. (C) Histogram (blue) and p.d.f. (red) superimposed. Note that the histogram has a shape similar to the p.d.f. *MatLab* script gda02_01.

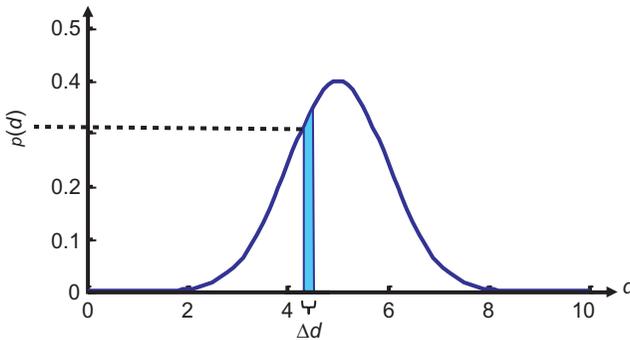


FIGURE 2.2 The shaded area $p(d)\Delta d$ of the probability density function $p(d)$ gives the probability, P , that the observation will fall between d and $d + \Delta d$. *MatLab* script gda02_02.

The special case of $d_1 = -\infty$, $d_2 = d$, which represents the probability that the value of the random variable is less than or equal to a given value d , is called the *cumulative distribution function* $P(d)$ (abbreviated *c.d.f.*). Note that the numerical value of P represents an actual probability, while the numerical value of p does not.

In *MatLab*, we use a vector \mathbf{d} evenly spaced values with sampling Δd to represent the random variable and we use a vector \mathbf{p} to represent the probability density function at corresponding values of d . The total probability P_{total} (which should be unity) and the cumulative probability distribution \mathbf{P} are calculated as

$$P_{\text{total}} = \Delta d * \text{sum}(\mathbf{p}) ;$$

$$\mathbf{P} = \Delta d * \text{cumsum}(\mathbf{p}) ;$$

(*MatLab* script gda02_03)

Here we are employing the Riemann approximation for an integral, $\int p(d) dd \approx \Delta d \sum_i p(d_i)$. Note that the `sum()` function returns a scalar, the sum of the elements of `p`, whereas the `cumsum()` function returns a vector, the running sum of the elements of `p`.

The probability density function $p(d)$ completely describes the random variable, d . Unfortunately, it is a continuous function that may be quite complicated. A few numbers that summarize the major properties of the probability density function can be very helpful. One such kind of number indicates the typical numerical value of a measurement. The most likely measurement is the one with the highest probability, that is, the value of d at which $p(d)$ is peaked (Figure 2.3). However, if the distribution is skewed, this *maximum likelihood point* may not be a good indication of the typical measurement, since a wide range of other values also has high probability. In such instances, the *mean*, or *expected* measurement, $\langle d \rangle$, is a better characterization of a typical measurement. This number is the “balancing point” of the distribution and is given by

$$\langle d \rangle = E(d) = \int_{-\infty}^{+\infty} d p(d) dd \quad (2.3)$$

Another property of a distribution is its overall width. Wide distributions imply very noisy data, and narrow ones imply relatively noise-free data. One way of measuring the width of a distribution is to multiply it by a function that is zero near the center (peak) of the distribution and that grows on either side of the peak (Figure 2.4). If the distribution is narrow, then the resulting function will be everywhere small; if the distribution is wide, then the result will be large.

A quantitative measure of the width of the peak is the area under the resulting function. If one chooses the parabola $(d - \langle d \rangle)^2$ as the function, where $\langle d \rangle = E(d)$ is the expected value of the random variable, then this measure is called the *variance* σ^2 of the distribution and is written as

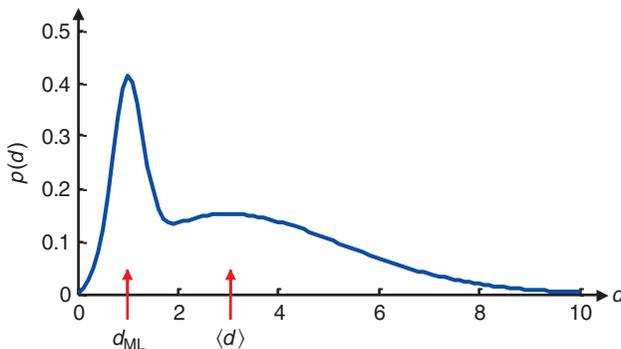


FIGURE 2.3 The maximum likelihood point d_{ML} of the probability density function $p(d)$ gives the most probable value of the datum d . In general, this value can be different than the mean datum $\langle d \rangle$ which is at the “balancing point” of the distribution. *MatLab* script `gda02_04`.

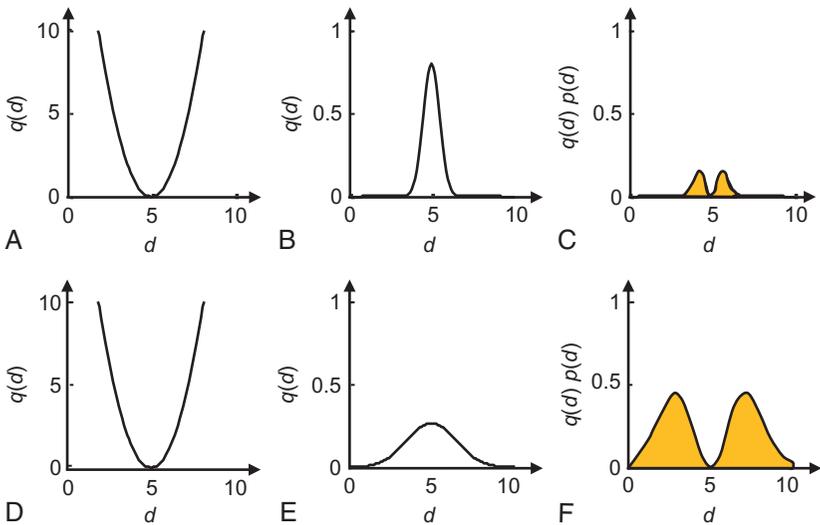


FIGURE 2.4 (A and D) Parabola of the form $q(d) = (d - \langle d \rangle)^2$ is used to measure the width of two probability density functions $p(d)$ (B and E), which have the same mean $\langle d \rangle$ but different widths. The product qp is everywhere small for the narrow function (C) but had two large peaks for the wider distribution (F). The area (shaded orange) under qp is a measure of the width of the function $p(d)$ and is called the variance. The variances of (A) and (F) are $(0.5)^2$ and $(1.5)^2$, respectively. *MatLab* script gda02_05.

$$\sigma^2 = \int_{-\infty}^{+\infty} (d - \langle d \rangle)^2 p(d) dd \quad (2.4)$$

The square root of the variance, σ , is a measure of the width of the distribution. In *MatLab*, the expected value and variance are computed as

```
Ed = Dd * sum(d .* p) ;
sigma2 = Dd * sum((d - Ed).^2 .* p) ;
```

(*MatLab* script gda02_05)

Here d is a vector of equally spaced values of the random variable d , with spacing Δd , and p is the corresponding value of the probability density function.

As we will discuss further in [Chapter 5](#), the mean and variance can be estimated from a set of N realizations of data d_i as

$$\langle d \rangle^{\text{est}} = \frac{1}{N} \sum_{i=1}^N d_i \quad \text{and} \quad (\sigma^2)^{\text{est}} = \frac{1}{N-1} \sum_{i=1}^N (d_i - \langle d \rangle^{\text{est}})^2 \quad (2.5)$$

The quantity $\langle d \rangle^{\text{est}}$ is called the *sample mean* and the quantity σ^{est} is called the *sample standard deviation*. In *MatLab*, these estimates can be computed using the `mean(dx)` and `std(dx)` functions, where dx is a vector of N realizations of the random variable d .

2.2 CORRELATED DATA

Experiments usually involve the collection of more than one datum. We therefore need to quantify the probability that a set of random variables will take on a given value. The joint probability density function $p(\mathbf{d})$ is the probability that the first datum will be in the neighborhood of d_1 , that the second will be in the neighborhood of d_2 , etc. If the data are independent—that is, if there are no patterns in the occurrence of the values between pairs of random variables—then this joint distribution is just the product of the individual distributions (Figure 2.5)

$$p(\mathbf{d}) = p(d_1) p(d_2) p(d_3) \cdots p(d_N) \quad (2.6)$$

The probability density function for a single random variable, say d_i , irrespective of all the others, is computed by integrating $p(\mathbf{d})$ over all the other variables:

$$p(d_i) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} p(\mathbf{d}) dd_j dd_k \cdots dd_l \quad (2.7)$$

($N - 1$ integrals)

In some experiments, measurements *are* correlated. High values of one datum tend to occur consistently with either high or low values of another datum (Figure 2.6). The joint distribution for such data must be constructed to take this correlation into account. Given a joint distribution $p(d_1, d_2)$ for two random variables d_1 and d_2 , one can test for correlation by selecting a function that divides the (d_1, d_2) plane into four quadrants of alternating sign, centered on the mean of the distribution (Figure 2.7). If one multiplies the distribution by this function, and then sums up the area, the result will be zero for uncorrelated distributions, since they tend to lie equally in all four quadrants. Correlated distributions will have either positive or negative area, since they tend to be concentrated in two

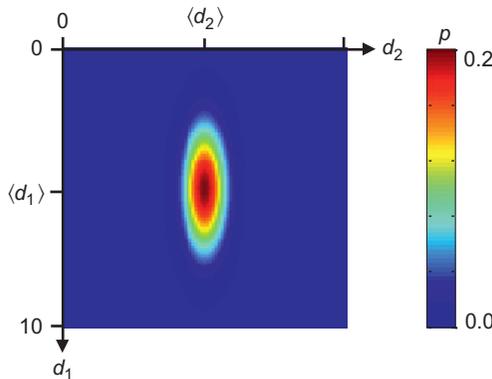


FIGURE 2.5 The probability density function $p(d_1, d_2)$ is displayed as an image, with values given by the accompanying color bar. These data are uncorrelated, since especially large values of d_2 are no more or less likely if d_1 is large or small. In this example, the variance of d_1 and d_2 are $\sigma_1^2 = (1.5)^2$ and $\sigma_2^2 = (0.5)^2$, respectively. *MatLab* script gda02_06.

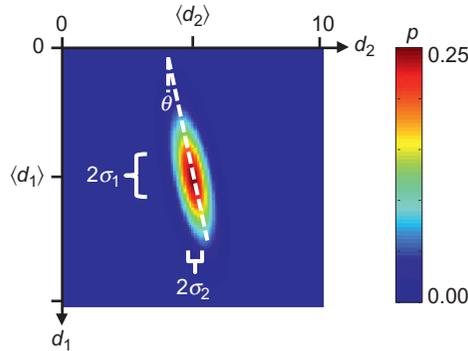


FIGURE 2.6 The probability density function $p(d_1, d_2)$ is displayed as an image, with values given by the accompanying color bar. These data are positively correlated, since large values of d_2 are especially probable if d_1 is large. The function has means $\langle d_1 \rangle = 5$ and $\langle d_2 \rangle = 5$ and widths in the coordinate directions $\sigma_1 = 1.5$ and $\sigma_2 = 0.5$. The angle θ is a measure of the degree of correlation and is related to the covariance $\text{cov}(d_1, d_2) = 0.4$. *MatLab* script `gda02_07`.

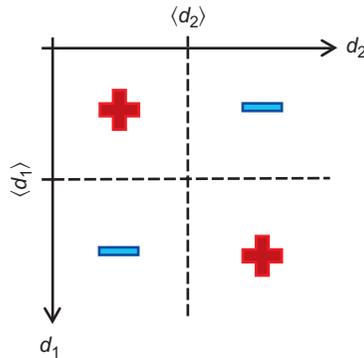


FIGURE 2.7 The function $q(d_1, d_2) = (d_1 - \langle d_1 \rangle)(d_2 - \langle d_2 \rangle)$ divides the (d_1, d_2) plane into four quadrants of alternating sign.

opposite quadrants (Figure 2.8). If $[d_1 - \langle d_1 \rangle][d_2 - \langle d_2 \rangle]$ is used as the function, the resulting measure of correlation is called the covariance:

$$\text{cov}(d_1, d_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} [d_1 - \langle d_1 \rangle][d_2 - \langle d_2 \rangle] p(d_1, d_2) dd_1 dd_2 \quad (2.8)$$

Note that the covariance of a datum with itself is just the variance. The covariance, therefore, characterizes the basic shape of the joint distribution.

When there are many data given by the vector \mathbf{d} , it is convenient to define a vector of expected values and a matrix of covariances as

$$\begin{aligned} \langle d \rangle_i &= \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} d_i p(\mathbf{d}) dd_1 \cdots dd_N \\ [\text{cov } \mathbf{d}]_{ij} &= \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} [d_i - \langle d_i \rangle][d_j - \langle d_j \rangle] p(\mathbf{d}) dd_1 \cdots dd_N \end{aligned} \quad (2.9)$$

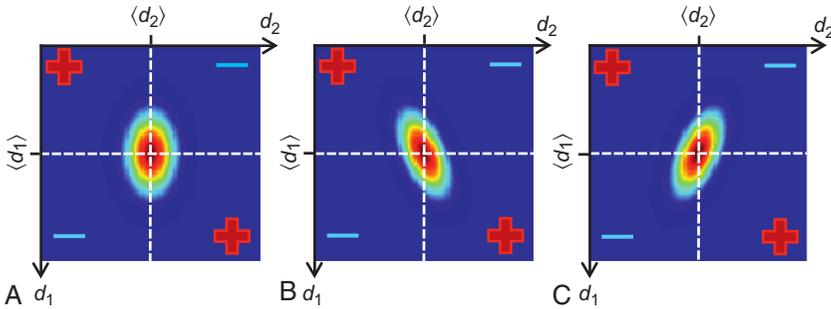


FIGURE 2.8 The probability density function $p(d_1, d_2)$ displayed as an image, when the data are (A) uncorrelated, (B) positively correlated, and (C) negatively correlated. The dashed lines indicated the four quadrants of alternating sign used to determine the correlation (see Figure 2.7). *MatLab* script gda02_08.

Henceforth, we will abbreviate these multidimensional integrals as $\int d^N d$. The diagonal elements of the covariance matrix are variances. They are measures of the scatter in the data. The off-diagonal elements are covariances. They indicate the degree to which pairs of data are correlated. Notice that the integral for the mean can be written in terms of the univariate probability density function $p(d_i)$ and the integral for the variance can be written in terms of the bivariate probability density function $p(d_i, d_j)$, since the other dimension of $p(\mathbf{d})$ are just “integrated away” to unity:

$$\begin{aligned} \langle d \rangle_i &= \int_{-\infty}^{+\infty} d_i p(d_i) dd_i \\ [\text{cov } \mathbf{d}]_{ij} &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} [d_i - \langle d_i \rangle][d_j - \langle d_j \rangle] p(d_i, d_j) dd_i dd_j \end{aligned} \quad (2.10)$$

The covariance matrix can be estimated from a set of N realizations of data. Suppose that there are N different types of data and that K realizations of them have been observed. The data can be organized into a matrix \mathbf{D} , with the N columns referring to the different data types and the K rows to the different realizations. The *sample covariance* is then

$$[\text{cov } \mathbf{d}]_{ij}^{\text{est}} = \frac{1}{K} \sum_{k=1}^K (D_{ki} - \langle D_i \rangle^{\text{est}})(D_{kj} - \langle D_j \rangle^{\text{est}}) \quad (2.11)$$

Here $\langle D_i \rangle^{\text{est}}$ is the sample mean of the i th data type. The *MatLab* function `cov(D)` implements this formula.

2.3 FUNCTIONS OF RANDOM VARIABLES

The basic premise of inverse theory is that the data and model parameters are related. Any method that solves the inverse problem—that estimates a model parameter on the basis of data—will map errors from the data to the estimated

model parameters. Thus the *estimates* of the model parameters are themselves random variables, which are described by a distribution $p(\mathbf{m}^{\text{est}})$. Whether or not the *true* model parameters are random variables depends on the problem. It is appropriate to consider them deterministic quantities in some problems and random variables in others. *Estimates* of the model parameters, however, are always random variables.

We need the tools to transform probability density functions from $p(\mathbf{d})$ to $p(\mathbf{m})$ when the relationship $\mathbf{m}(\mathbf{d})$ is known. We start simply and consider just one datum and one model parameter, related by the simple function $m(d) = 2d$. Now suppose that $p(d)$ is *uniform* on the interval $(0, 1)$; that is, d has equal probability of being anywhere in this range. The probability density function is constant and must have amplitude $p(d) = 1$, since the total probability must be unity (width \times height $= 1 \times 1 = 1$). The probability density function $p(m)$ is also uniform, but on the interval $(0, 2)$, since m is twice d . Thus, $p(m) = 1/2$, since its total probability must also be unity (width \times height $= 2 \times 1/2 = 1$) (Figure 2.9). This result shows that $p(m)$ is not merely $p[d(m)]$, but rather must include a factor that accounts for the stretching (or shrinking) of the m -axis with respect to the d -axis.

This stretching factor can be derived by transforming the integral for total probability:

$$1 = \int_{d_{\min}}^{d_{\max}} p(d) dd = \int_{d(m_{\min})}^{d(m_{\max})} p[d(m)] \frac{dd}{dm} dm = \int_{m_{\min}}^{m_{\max}} p(m) dm \quad (2.12)$$

By inspection, $p(m) = p[d(m)] dd/dm$, so the stretching factor is dd/dm . The limits (d_{\min}, d_{\max}) transform to (m_{\min}, m_{\max}) . However, depending upon the function $m(d)$, we may find that $m_{\min} > m_{\max}$; that is, the direction of integration might be reversed ($m(d) = 1/d$ would be one such case). We handle this problem by adding an absolute value sign

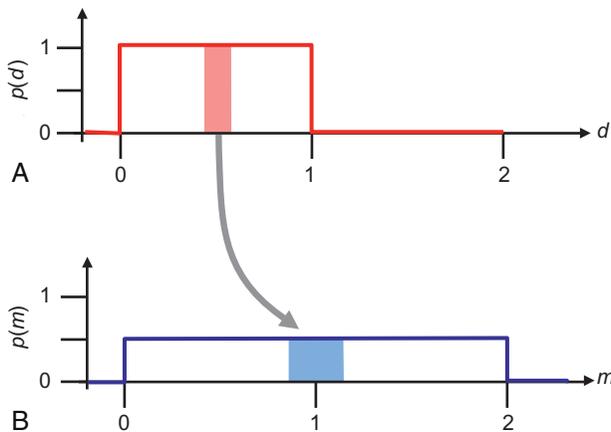


FIGURE 2.9 (A) The uniform probability density function $p(d) = 1$ on the interval $0 < d < 1$. (B) The transformed probability density function $p(m)$, given the relationship $m = 2d$. Note that a patch (shaded rectangle) of probability in m is wider and lower than the equivalent patch in d .

$$p(m) = p[d(m)] \left| \frac{dd}{dm} \right| \quad (2.13)$$

together with the understanding that the integration is always performed in the direction of positive m . Note that in the case above, with $p(d) = 1$ and $m(d) = 2d$, we find $dd/dm = 1/2$ and (as expected) $p(m) = 1 \times 1/2 = 1/2$.

In general, probability density functions change shape when transformed from d to m . Consider, for example, the uniform probability density function $p(d) = 1$ on the interval $(0, 1)$ together with the function $m(d) = d^2$ (Figure 2.10). We find $d = m^{1/2}$, $dd/dm = 1/2m^{-1/2}$, and $p(m) = 1/2m^{-1/2}$, with m defined on the interval $(0, 1)$. Thus, while $p(d)$ is uniform, $p(m)$ has a peak (actually an integrable singularity) at $m = 0$ (Figure 2.10B).

The general case of transforming $p(\mathbf{d})$ to $p(\mathbf{m})$, given the functional relationship $\mathbf{d}(\mathbf{m})$, is more complicated but is derived using the rule for transforming multidimensional integrals that is analogous to Equation (2.13). This rule states that the volume element transforms as $d^N d = J(\mathbf{m}) d^N m$ where $J(\mathbf{m}) = |\det(\partial \mathbf{d} / \partial \mathbf{m})|$ is the *Jacobian determinant*, that is, the absolute value of the determinant of the matrix whose elements are $[\partial \mathbf{d} / \partial \mathbf{m}]_{ij} = \partial d_i / \partial m_j$:

$$\begin{aligned} 1 &= \int p(\mathbf{d}) d^N d = \int p[\mathbf{d}(\mathbf{m})] \left| \det \left[\frac{\partial \mathbf{d}}{\partial \mathbf{m}} \right] \right| d^N m = \int p[\mathbf{d}(\mathbf{m})] J(\mathbf{m}) d^N m \\ &= \int p(\mathbf{m}) d^N m \end{aligned} \quad (2.14)$$

Hence, by inspection, we find that the probability density function transforms as

$$p(\mathbf{m}) = p[\mathbf{d}(\mathbf{m})] \left| \det \left[\frac{\partial \mathbf{d}}{\partial \mathbf{m}} \right] \right| = p[\mathbf{d}(\mathbf{m})] J(\mathbf{m}) \quad (2.15)$$

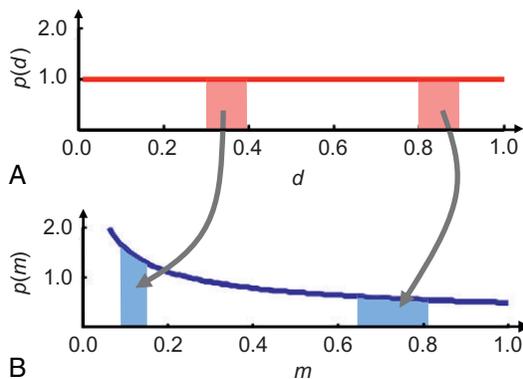


FIGURE 2.10 (A) The uniform probability density function $p(d) = 1$ on the interval $0 < d < 1$. (B) The transformed probability density function $p(m)$, given the relationship $m = d^2$. Note areas of equal probability, which are of equal height and width in the variable d are transformed into areas of unequal height and width in the variable, m . *MatLab* script gda02_09.

Note that for the linear transformation $\mathbf{m} = \mathbf{M}\mathbf{d}$, the Jacobian is constant, with the value $J = |\det(\mathbf{M}^{-1})| = |\det(\mathbf{M})|^{-1}$. As an example, consider a two-dimensional probability density function that is uniform on the intervals $(0, 1)$ for d_1 and $(0, 1)$ for d_2 , together with the transformation $m_1 = d_1 + d_2$, $m_2 = d_1 - d_2$. As is shown in Figure 2.11, $p(\mathbf{d})$ corresponds to a square of unit area in the (d_1, d_2) plane and $p(\mathbf{m})$ corresponds to a square of area 2 in the (m_1, m_2) plane. In order that the total area be unity in both cases, we must have $p(\mathbf{d}) = 1$ and $p(\mathbf{m}) = 1/2$. The transformation matrix is

$$M = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{so } |\det(M)| = 2 \quad \text{and } J = \frac{1}{2} \quad (2.16)$$

Thus, by Equation (2.15), we find that $p(\mathbf{m}) = p(\mathbf{d})J = 1 \times 1/2 = 1/2$, which agrees with our expectations.

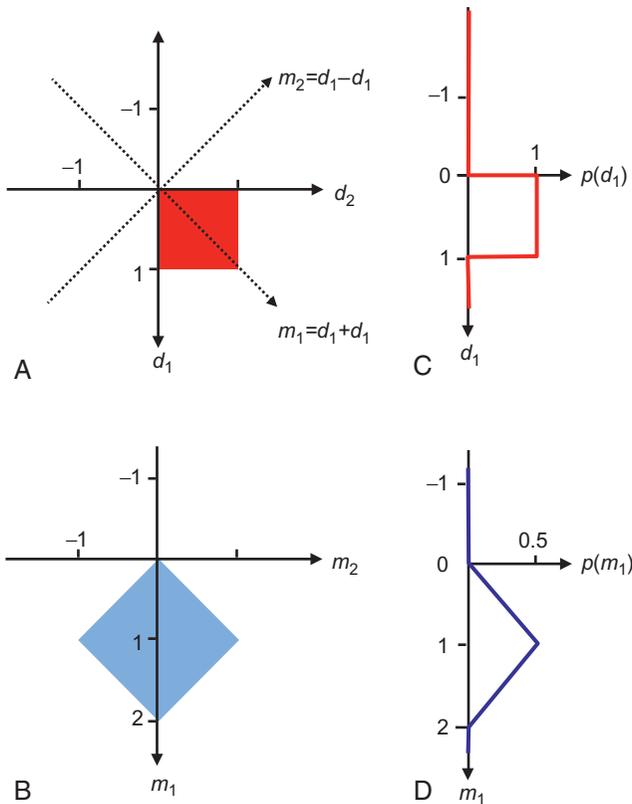


FIGURE 2.11 (A) The uniform probability density function $p(d_1, d_2) = 1$ on the interval $0 < d_1 < 1$, $0 < d_2 < 1$. Also shown are (m_1, m_2) axes, where $m_1 = d_1 + d_2$ and $m_2 = d_1 - d_2$. (B) The transformed probability density function $p(m_1, m_2) = 0.25$. (C) The univariate distribution $p(d_1)$ is formed by integrating $p(d_1, d_2)$ over d_2 . It is a uniform distribution of amplitude 1. (D) The univariate distribution $p(m_1)$ is formed by integrating $p(m_1, m_2)$ over m_2 . It is a triangular distribution of peak amplitude 0.5.

Note that we can convert $p(\mathbf{d})$ to a univariate distribution $p(d_1)$ by integrating over d_2 . Since the sides of the square are parallel to the coordinate axes, the integration yields the uniform probability density function, $p(d_1) = 1$ (Figure 2.11C). Similarly, we can convert $p(\mathbf{m})$ to a univariate distribution $p(m_1)$ by integrating over m_2 . However, because the sides of the square are oblique to the coordinate axes, $p(m_1)$ is a triangular—not a uniform—probability density function (Figure 2.11D).

Transforming a probability density function $p(\mathbf{d})$ is straightforward, but tedious. Fortunately, in the case of the linear function $\mathbf{m} = \mathbf{M}\mathbf{d} + \mathbf{v}$, where \mathbf{M} and \mathbf{v} are an arbitrary matrix and vector, respectively, it is possible to make some statements about the properties of the results without explicitly calculating the transformed probability density function $p(\mathbf{m})$. In particular, the mean and covariance can be shown, respectively, to be

$$\langle \mathbf{m} \rangle = \mathbf{M}\langle \mathbf{d} \rangle + \mathbf{v} \quad (2.17a)$$

and

$$[\text{cov } \mathbf{m}] = \mathbf{M}[\text{cov } \mathbf{d}]\mathbf{M}^T \quad (2.17b)$$

These rules are derived by transforming the definition of the mean and variance:

$$\begin{aligned} \langle m \rangle_i &= \int m_i p(\mathbf{m}) d^N m = \int \sum_j M_{ij} d_j p[\mathbf{d}(\mathbf{m})] \left| \det \left[\frac{\partial \mathbf{d}}{\partial \mathbf{m}} \right] \right| \left| \det \left[\frac{\partial \mathbf{m}}{\partial \mathbf{d}} \right] \right| d^N d \\ &= \sum_j M_{ij} \int d_j p(\mathbf{d}) d^N d = \sum_j M_{ij} \langle d_j \rangle \end{aligned} \quad (2.18)$$

$$\begin{aligned} [\text{cov } \mathbf{m}]_{ij} &= \\ &= \int (m_i - \langle m \rangle_i)(m_j - \langle m \rangle_j) p(\mathbf{m}) d^N m \\ &= \int \sum_p (M_{ip} d_p - M_{ip} \langle d_p \rangle) \sum_q (M_{jq} d_q - M_{jq} \langle d_q \rangle) \\ &\quad p[\mathbf{d}(\mathbf{m})] \left| \det \left[\frac{\partial \mathbf{d}}{\partial \mathbf{m}} \right] \right| \left| \det \left[\frac{\partial \mathbf{m}}{\partial \mathbf{d}} \right] \right| d^N d \\ &= \sum_p \sum_q M_{ip} M_{jq} \int (d_p - \langle d_p \rangle)(d_q - \langle d_q \rangle) p(\mathbf{d}) d^N d \\ &= \sum_p \sum_q M_{ip} [\text{cov } \mathbf{d}]_{pq} M_{jq} \end{aligned} \quad (2.19)$$

Equation (2.17b) is very important, because the covariance of the data is a measure of the amount of measurement error. The equation functions as a rule for *error propagation*; that is, given $[\text{cov } \mathbf{d}]$ representing measurement error, it

provides a way to compute $[\text{cov } \mathbf{m}]$ representing the corresponding error in the model parameters. While the rule requires that the data and the model parameters be linearly related, it is independent of the functional form of the probability density function $p(\mathbf{d})$. Furthermore, it can be shown to be correct even when the matrix \mathbf{M} is not square.

As an example, consider a model parameter m_1 , which is linearly related to the data by

$$m_1 = \frac{1}{N} \sum_{i=1}^N d_i = \frac{1}{N} [1, 1, 1, \dots, 1] \mathbf{d} \quad (2.20)$$

Note that this formula is the sample mean, as defined in Equation (2.5). This formula implies that matrix $\mathbf{M} = [1, 1, 1, \dots, 1]/N$ and vector $\mathbf{v} = 0$. Suppose that the data are uncorrelated and all have the same mean $\langle d \rangle$ and variance σ_d^2 . Then we see that $\langle m_1 \rangle = \mathbf{M} \langle \mathbf{d} \rangle + \mathbf{v} = \langle d \rangle$ and $\text{var}(m_1) = \mathbf{M} [\text{cov } \mathbf{d}] \mathbf{M}^T = \sigma_d^2/N$. The model parameter m_1 has a probability density function $p(m_1)$ with the same mean as \mathbf{d} ; that is, $\langle m_1 \rangle = \langle d \rangle$. Hence it is an estimate of the mean of the data. Its variance $\sigma_m^2 = \sigma_d^2/N$ is less than the variance of \mathbf{d} . The square root of the variance, which is a measure of the width of the $p(m_1)$, is proportional to $N^{-1/2}$. Thus, accuracy of determining the mean of a group of data increases as the number of observations increases, albeit slowly (because of the square root).

In the case of uncorrelated data with uniform variance (that is, $[\text{cov } \mathbf{d}] = \sigma_d^2 \mathbf{I}$), the covariance of the model parameters is $[\text{cov } \mathbf{m}] = \mathbf{M} [\text{cov } \mathbf{d}] \mathbf{M}^T = \sigma_d^2 \mathbf{M} \mathbf{M}^T$. In general, $\mathbf{M} \mathbf{M}^T$, while symmetric, is not diagonal. Not only do the model parameters have unequal variance, but they are also correlated. Strongly correlated model parameters are usually undesirable, but (as we will discuss later) good experimental design can sometimes eliminate them.

2.4 GAUSSIAN PROBABILITY DENSITY FUNCTIONS

The probability density function for a particular random variable can be arbitrarily complicated, but in many instances, data possess the rather simple *Gaussian* (or *Normal*) probability density function

$$p(d) = \frac{1}{(2\pi)^{1/2} \sigma} \exp \left[-\frac{(d - \langle d \rangle)^2}{2\sigma^2} \right] \quad (2.21)$$

This probability density function has mean $\langle d \rangle$ and variance σ^2 (Figure 2.12). The Gaussian probability density function is so common because it is the limiting probability density function for the sum of random variables. The *central limit theorem* shows (with certain limitations) that regardless of the probability density function of a set of independent random variables, the probability density function of their sum tends to a Gaussian distribution as the number of summed variables increases. As long as the noise in the data comes from several sources of comparable size, it will tend to follow a Gaussian probability density function. This behavior is exemplified by the sum of the two uniform

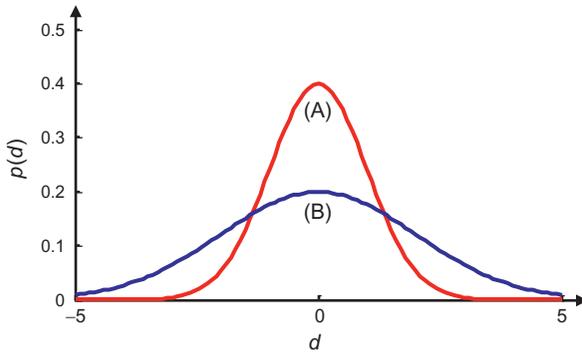


FIGURE 2.12 Gaussian (Normal) distribution with zero mean and $\sigma=1$ for curve (A) and $\sigma=2$ for curve (B). *MatLab* script gda02_10.

probability density functions in [Section 2.3](#). The probability density function of their sum is more nearly Gaussian than the individual probability density functions (it being triangular instead of rectangular).

The joint probability density function for two independent Gaussian variables is just the product of two univariate probability density functions. When the data are correlated (say, with mean $\langle \mathbf{d} \rangle$ and covariance $[\text{cov } \mathbf{d}]$), the joint probability density function is more complicated, since it must express the degree of correlation. The appropriate generalization can be shown to be

$$p(\mathbf{d}) = \frac{1}{(2\pi)^{N/2} (\det[\text{cov } \mathbf{d}])^{1/2}} \exp\left(-\frac{1}{2} [\mathbf{d} - \langle \mathbf{d} \rangle]^T [\text{cov } \mathbf{d}]^{-1} [\mathbf{d} - \langle \mathbf{d} \rangle]\right) \quad (2.22)$$

Note that this probability density function reduces to [Equation \(2.21\)](#) in the special case of $N=1$ (where $[\text{cov } \mathbf{d}]$ becomes σ_d^2). It is perhaps not apparent that the general case has an area of unity, a mean of $\langle \mathbf{d} \rangle$ and a covariance matrix of $[\text{cov } \mathbf{d}]$. However, these properties can be derived by inserting [Equation \(2.22\)](#) into the relevant integral and by transforming to the new variable $\mathbf{y} = [\text{cov } \mathbf{d}]^{-1/2} [\mathbf{d} - \langle \mathbf{d} \rangle]$ (whence the integral becomes substantially simplified).

When $p(\mathbf{d})$ ([Equation 2.22](#)) is transformed using the linear rule $\mathbf{m} = \mathbf{M}\mathbf{d}$, the resulting $p(\mathbf{m})$ is also Gaussian in form with mean $\langle \mathbf{m} \rangle = \mathbf{M}\langle \mathbf{d} \rangle$ and covariance matrix $[\text{cov } \mathbf{m}] = \mathbf{M}[\text{cov } \mathbf{d}]\mathbf{M}^T$. Thus, all linear functions of Gaussian random variables are themselves Gaussian.

In [Chapter 5](#), we will show that the information contained in each of two probability density functions can be combined by multiplying the two distributions. Interestingly, the product of two Gaussian probability density functions is itself Gaussian ([Figure 2.13](#)). Given Gaussian $p_A(\mathbf{d})$ with mean $\langle \mathbf{d}_A \rangle$ and covariance $[\text{cov } \mathbf{d}]_A$ and Gaussian $p_B(\mathbf{d})$ with mean $\langle \mathbf{d}_B \rangle$ and covariance $[\text{cov } \mathbf{d}]_B$, the product $p_C(\mathbf{d}) = p_A(\mathbf{d}) p_B(\mathbf{d})$ is Gaussian with mean and variance (e.g., [Menke and Menke, 2011](#), their [Section 5.4](#))

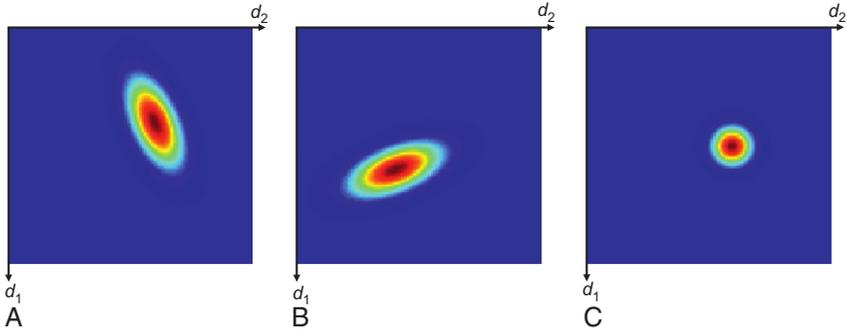


FIGURE 2.13 (A) A Normal probability density function $p_A(d_1, d_2)$. (B) Another Normal probability density function $p_B(d_1, d_2)$. (C) The product of these two functions $p_C(d_1, d_2) = p_A(d_1, d_2)p_B(d_1, d_2)$ is Normal. *MatLab* script gda02_11.

$$\langle \mathbf{d}_C \rangle = \left([\text{cov } \mathbf{d}]_A^{-1} + [\text{cov } \mathbf{d}]_B^{-1} \right)^{-1} \left([\text{cov } \mathbf{d}]_A^{-1} \langle \mathbf{d}_A \rangle + [\text{cov } \mathbf{d}]_B^{-1} \langle \mathbf{d}_B \rangle \right)$$

$$[\text{cov } \mathbf{d}]_C^{-1} = [\text{cov } \mathbf{d}]_A^{-1} + [\text{cov } \mathbf{d}]_B^{-1} \quad (2.23)$$

The idea that the model and data are related by an explicit relationship $\mathbf{g}(\mathbf{m}) = \mathbf{d}$ can be reinterpreted in light of this probabilistic description of the data. We can no longer assert that this relationship can hold for the data themselves, since they are random variables. Instead, we assert that this relationship holds for the mean data: $\mathbf{g}(\mathbf{m}) = \langle \mathbf{d} \rangle$. The distribution for the data can then be written as

$$p(\mathbf{d}) = \frac{1}{(2\pi)^{N/2} (\det[\text{cov } \mathbf{d}])^{1/2}} \exp\left(-\frac{1}{2} [\mathbf{d} - \mathbf{g}(\mathbf{m})]^T [\text{cov } \mathbf{d}]^{-1} [\mathbf{d} - \mathbf{g}(\mathbf{m})]\right) \quad (2.24)$$

The model parameters now have the interpretation of a set of unknown quantities that define the shape of the distribution for the data. One approach to inverse theory (which will be pursued in [Chapter 5](#)) is to use the data to determine the distribution and thus the values of the model parameters.

For the Gaussian distribution ([Equation 2.24](#)) to be sensible, $\mathbf{g}(\mathbf{m})$ must not be a function of any random variables. This is why we differentiated between data and auxiliary variables in [Chapter 1](#); the latter must be known exactly. If the auxiliary variables are themselves uncertain, then they must be treated as data and the inverse problem becomes an implicit one with a much more complicated distribution than the above problem exhibits.

As an example of constructing the distribution for a set of data, consider an experiment in which the temperature d_i in some small volume of space is measured N times. If the temperature is assumed not to be a function of time and space, the experiment can be viewed as the measurement of N realizations of the same random variable or as the measurement of one realization of N distinct

random variables that all have the same distribution. We adopt the second viewpoint.

If the data are independent Gaussian random variables with mean $\langle \mathbf{d} \rangle$ and variance σ_d^2 so that $[\text{cov } \mathbf{d}] = \sigma_d^2 \mathbf{I}$, then we can represent the assumption that all the data have the same mean by an equation of the form $\mathbf{Gm} = \mathbf{d}$:

$$\begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} [m_1] = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_N \end{bmatrix} \quad (2.25)$$

where m_1 is a single model parameter. We can then compute explicit formulas for the expressions in $p(\mathbf{d})$ as

$$\begin{aligned} (\det[\text{cov } \mathbf{d}]^{-1})^{1/2} &= (\sigma_d^{-2N})^{1/2} = \sigma_d^{-N} \\ [\mathbf{d} - \mathbf{Gm}]^T [\text{cov } \mathbf{d}]^{-1} [\mathbf{d} - \mathbf{Gm}] &= \sigma_d^{-2} \sum_{i=1}^N (d_i - m_1)^2 \end{aligned} \quad (2.26)$$

The joint distribution is therefore

$$p(\mathbf{d}) = \frac{\sigma_d^{-N}}{(2\pi)^{N/2}} \exp \left[-\frac{1}{2} \sigma_d^{-2} \sum_{i=1}^N (d_i - m_1)^2 \right] \quad (2.27)$$

2.5 TESTING THE ASSUMPTION OF GAUSSIAN STATISTICS

In the following chapters, we shall derive methods of solving inverse problems that are applicable whenever the data exhibit Gaussian statistics. In many instances, the assumption that the data follow this distribution is a reasonable one; nevertheless, having some means to test it is important.

First, consider a set of N Gaussian random variables x_i each with zero mean and unit variance. Suppose we construct a new random variable

$$\chi_K^2 = \sum_{i=1}^K x_i^2 \quad (2.28)$$

by summing squares of x_i . The function relating the x_i to χ_K^2 is nonlinear, so χ_K^2 does not have a Gaussian probability density function, but rather a different one (which we will not derive here) with the functional form

$$p(\chi_K^2) = \frac{1}{2^{K/2} (\frac{K}{2} - 1)!} [\chi_K^2]^{(K/2)-1} \exp \left(-\frac{1}{2} \chi_K^2 \right) \quad (2.29)$$

It is called the *chi-squared probability density function*. It can be shown to be unimodal with mean K and variance $2K$. We shall make use of it in the discussion to follow.

We begin by supposing that we have some method of solving the inverse problem for the estimated model parameters. Assuming further that the model is explicit, we can compute the variation of the data about its estimated mean—a quantity we refer to as the error $\mathbf{e} = \mathbf{d} - \mathbf{g}(\mathbf{m}^{\text{est}})$. Does this error follow an uncorrelated Gaussian distribution with uniform variance?

To test the hypothesis that it does, we first make a histogram of the N errors e_i , in which the histogram intervals have been chosen so that there are about the same number of errors e_i in each interval. This histogram is then normalized to unit area, and the area A_i^{est} of each of the, say, p intervals is noted. We then compare these areas (which are all in the range from zero to unity) with the areas A_i predicted by a Gaussian distribution with the same mean and variance as the e_i . The overall difference between these areas can be quantified by using

$$(\chi_K^2)^{\text{est}} = N \sum_{i=1}^p \frac{(A_i^{\text{est}} - A_i)^2}{A_i} \quad (2.30)$$

If the data followed a Gaussian distribution exactly, then $(\chi_K^2)^{\text{est}}$ should be close to zero (it will not *be* zero since there are always random fluctuations). We therefore need to inquire whether the $(\chi_K^2)^{\text{est}}$ measured for any particular data set is sufficiently far from zero that it is improbable that the data follow the Gaussian distribution. This is done by computing the theoretical distribution of $(\chi_K^2)^{\text{est}}$ and testing whether $(\chi_K^2)^{\text{est}}$ is probable. The usual rule for deciding that the data do not follow the assumed distribution is that values greater than or equal to $(\chi_K^2)^{\text{est}}$ occur less than 5% of the time (if many realizations of the entire experiment were performed).

The quantity $(\chi_K^2)^{\text{est}}$ can be shown to follow approximately a χ_K^2 distribution with $K = p - 3$ degrees of freedom, regardless of the type of distribution involved. The reason that the degrees of freedom are $p - 3$ rather than p is that three constraints have been introduced into the problem: that the area of the histogram is unity and that the mean and variance of the Gaussian distribution match those of the data. This test is known as *Pearson's chi-squared test* (Figure 2.14). In *MatLab*, the probability P that χ_K^2 is greater than or equal to $(\chi_K^2)^{\text{est}}$ is computed as

$$P = 1 - \text{chi2cdf}(x2\text{est}, K);$$

(*MatLab* script gda02_12)

Here $\text{chi2cdf}()$ is the cumulative chi-squared distribution, that is, the probability that χ_K^2 is less than or equal to $(\chi_K^2)^{\text{est}}$.

2.6 CONDITIONAL PROBABILITY DENSITY FUNCTIONS

Consider a scenario in which we are measuring the diameter d_1 and weight d_2 of sand grains drawn randomly from a pile of sand. We can consider d_1 and d_2 random variables described by a joint probability density function $p(d_1, d_2)$. The variables d_1 and d_2 will be correlated, since large grains will also tend to be heavy. Now, suppose that $p(d_1, d_2)$ is known. Once we draw a sand grain from the pile and weigh it, we already know something about its diameter, since

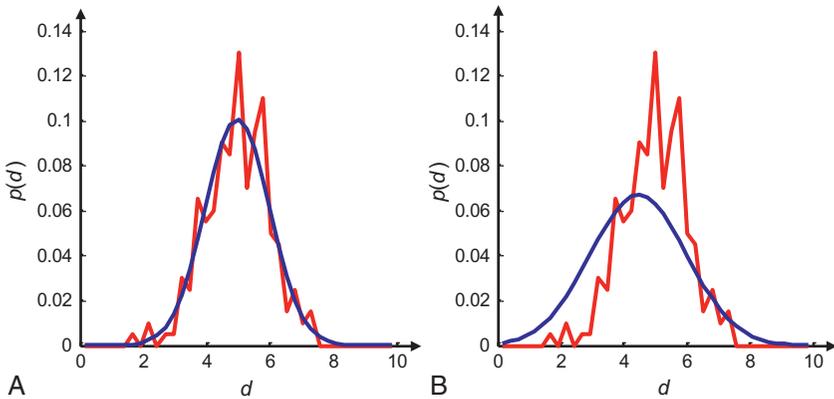


FIGURE 2.14 Example of Pierson's chi-squared test. The red curve is a probability density function (p.d.f.) estimated by binning 200 realizations of a random variable d drawn from a Gaussian population with a mean of 5 and a variance of 1^2 . (A) Gaussian p.d.f. with the same mean and variance as the empirical one. (B) Gaussian p.d.f. with a mean of 4.5 and a variance of 1.5^2 . According to the test, χ^2 values exceeding the observed value occur extremely frequently (75% of the time) for (A) but extremely infrequently (0.003%) for (B). *MatLab* script gda02.12.

diameter is correlated with weight. The quantity that embodies this information is called the *conditional* probability density function of d_1 , given d_2 , and is written $p(d_1|d_2)$.

The *conditional* probability density function of $p(d_1|d_2)$ is not the same as $p(d_1, d_2)$, although it is related to it. The key difference is that $p(d_1|d_2)$ is really only a probability density function in the variable d_1 , with the variable d_2 just providing auxiliary information. Thus, the integral of $p(d_1|d_2)$ with respect to d_1 needs to be unity, regardless of the value of d_2 . Thus, we must normalize $p(d_1, d_2)$ by dividing it by the total probability that d_1 occurs, given a specific value for d_2

$$p(d_1|d_2) = \frac{p(d_1, d_2)}{\int p(d_1, d_2) dd_1} = \frac{p(d_1, d_2)}{p(d_2)} \quad (2.31)$$

Here we have used the fact that $p(d_2) = \int p(d_1, d_2) dd_1$. The same logic allows us to calculate the conditional probability density function for d_2 , given d_1

$$p(d_2|d_1) = \frac{p(d_1, d_2)}{\int p(d_1, d_2) dd_2} = \frac{p(d_1, d_2)}{p(d_1)} \quad (2.32)$$

See [Figure 2.15](#) for an example. Combining these two equations yields

$$p(d_1, d_2) = p(d_1|d_2)p(d_2) = p(d_2|d_1)p(d_1) \quad (2.33)$$

This result shows that the two conditional probability density functions are related but that they are *not* equal: $p(d_1|d_2) \neq p(d_2|d_1)$. The two equations can be further rearranged into a result called *Bayes Theorem*

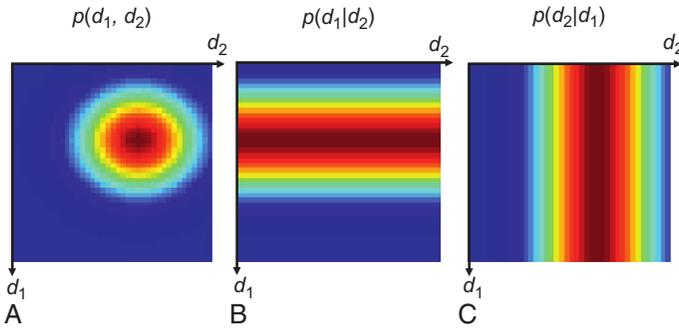


FIGURE 2.15 Example of conditional probability density functions. (A) A Gaussian joint probability density function $p(d_1, d_2)$. (B) The corresponding conditional probability density function $p(d_1|d_2)$. (C) The corresponding conditional probability density function $p(d_2|d_1)$. *MatLab* script gda02_13.

$$\begin{aligned}
 p(d_1|d_2) &= \frac{p(d_2|d_1)p(d_1)}{p(d_2)} = \frac{p(d_2|d_1)p(d_1)}{\int p(d_1, d_2) dd_1} = \frac{p(d_2|d_1)p(d_1)}{\int p(d_2|d_1)p(d_1) dd_1} \\
 p(d_2|d_1) &= \frac{p(d_1|d_2)p(d_2)}{p(d_1)} = \frac{p(d_1|d_2)p(d_2)}{\int p(d_1, d_2) dd_2} = \frac{p(d_1|d_2)p(d_2)}{\int p(d_1|d_2)p(d_2) dd_2}
 \end{aligned} \tag{2.34}$$

Note that only the denominators of the three fractions in each equation are different. They correspond to three different but equivalent ways of writing $p(d_1)$ and $p(d_2)$.

As an example, consider the case where diameter can take on only two discrete values, small (S) and big (B), and when weight can take on only two values, light (L) and heavy (H). A hypothetical joint probability function is

$$P(d_1, d_2) = \begin{bmatrix} d_1|d_2 & L & H \\ S & 0.8000 & 0.0010 \\ B & 0.1000 & 0.0990 \end{bmatrix} \tag{2.35}$$

In this scenario, about 90% of the small sand grains are light, about 99% of the large grains are heavy, and small/light grains are much more common than big/heavy ones. Univariate distributions are computed by summing over rows or columns, and in [Equation \(2.7\)](#):

$$P(d_1) = \begin{bmatrix} d_1 \\ S & 0.8010 \\ B & 0.1990 \end{bmatrix} \quad \text{and} \quad P(d_2) = \begin{bmatrix} d_2 & L & H \\ & 0.9000 & 0.1000 \end{bmatrix} \tag{2.36}$$

According to [Equation \(2.34\)](#), the conditional distributions are

$$P(d_1|d_2) = \begin{bmatrix} d_1|d_2 & L & H \\ S & 0.8888 & 0.0100 \\ B & 0.1111 & 0.9900 \end{bmatrix} \quad \text{and} \quad P(d_2|d_1) = \begin{bmatrix} d_1|d_2 & L & H \\ S & 0.9986 & 0.0012 \\ B & 0.5025 & 0.4974 \end{bmatrix} \tag{2.37}$$

Now suppose that we pick one sand grain from the pile, measure its diameter, and determine that it is big. What is the probability that it is heavy? We may be tempted to think that the probability is very high, since weight is highly correlated to size. But this reasoning is incorrect because heavy grains are about equally divided between the big and small size categories. The correct probability is given by $P(H|B)$, which is 49.74%.

Bayes theorem offers some insight into what is happening. Equation (2.34), adapted for discrete values by interpreting the integral as a sum, becomes

$$\begin{aligned} P(H|B) &= \frac{P(B|H)P(H)}{P(B|L)P(L) + P(B|H)P(H)} = \frac{0.9900 \times 0.1000}{0.1111 \times 0.9000 + 0.9900 \times 0.1000} \\ &= \frac{0.0990}{0.1000 + 0.0990} = \frac{0.0990}{0.1990} = 0.4974 \end{aligned} \quad (2.38)$$

The numerator of Equation (2.38) represents the big, heavy grains and the denominator represents *all* the ways that one can get big grains, that is, the sum of big, heavy grains and big, light grains. In the scenario, light grains are extremely common, and although only a small fraction of them are heavy, their number affects the probability very significantly.

The above analysis, called *Bayesian Inference*, allows us to assess the importance of any given measurement. Before having measured the size of the sand grain, our best estimate of whether it is heavy is 10%, because heavy grains make up 10% of the total population (that is, $P(H) = 0.10$). After the measurement, the probability rises to 49.74%, which is about a factor of five more certain. As we will see in Chapter 5, Bayesian Inference plays an important role in the solution of inverse problems.

2.7 CONFIDENCE INTERVALS

The confidence of a particular observation is the probability that one realization of the random variable falls within a specified distance of the true mean. Confidence is therefore related to the distribution of area in $p(d)$. If most of the area is concentrated near the mean, then the interval for, say, 95% confidence will be very small; otherwise, the confidence interval will be large. The width of the confidence interval is related to the variance. Distributions with large variances will also tend to have large confidence intervals. Nevertheless, the relationship is not direct, since variance is a measure of width, not area. The relationship is easy to quantify for the simplest univariate distributions. For instance, Gaussian probability density functions have 68% confidence intervals 1σ wide and 95% confidence intervals 2σ wide. Other types of simple distributions have similar relationships. If one knows that a particular Gaussian random variable has $\sigma = 1$, then if a realization of that variable has the value 50, one can state that there is a 95% chance that the mean of

the random variable lies between 48 and 52. One might symbolize this by $\langle d \rangle = 50 \pm 2$ (95%).

The concept of confidence intervals is more difficult to work with when one is dealing with joint probability density functions of several correlated random variables. One must define some volume in the space of data and compute the probability that the true means of the data are within the volume. One must also specify the shape of that volume. The more complicated the distribution, the more difficult it is to choose an appropriate shape and calculate the probability within it.

Even in the case of the Gaussian multivariate probability density functions, statements about confidence levels need to be made carefully, as is illustrated by the following scenario. Suppose that the Gaussian probability density function $p(d_1, d_2)$ represents two measurements, say the length and diameter of a cylinder, and suppose that these measurements are uncorrelated with equal variance, σ_d^2 . As we might expect, the univariate probability density function $p(d_1) = \int p(d_1, d_2) dd_2$ has variance, σ_d^2 , and so the probability, P_1 , that d_1 falls between $d_1 - \sigma_d$ and $d_1 + \sigma_d$, is 0.68 or 68%. Similarly, the probability, P_2 , that d_2 falls between $d_2 - \sigma_d$ and $d_2 + \sigma_d$, is also 68%. But P_1 represents the probability of d_1 , irrespective of the value of d_2 , and P_2 represents the probability of d_2 , irrespective of the value of d_1 . The probability, P , that *both* d_1 and d_2 simultaneously fall within their respective one-sigma confidence intervals is $P = P_1 P_2 = (0.68)^2 = 0.46$ or 46%, which is significantly smaller than 68%.

One occasionally encounters a journal article containing a table of many (say 100) estimated parameters, each one with a stated 2σ error bound. The probability that *all one hundred* measurements fall within their respective bounds is $(0.95)^{100}$ or 0.6%—which is pretty close to zero!

2.8 COMPUTING REALIZATIONS OF RANDOM VARIABLES

The ability to create a vector of realizations of a random variable is very important. For instance, it can be used to simulate noise when testing a data analysis method on synthetic data (that is, artificially prepared data with well-controlled properties). And it can be used to generate a suite of possible models, to test against data.

MatLab provides a function `random()` that can generate realizations drawn from many different probability density functions. For instance,

```
m = random('Normal', mbar, sigma, N, 1);
```

(*MatLab* script gda02_14)

creates a vector `m` of `N` Gaussian-distributed (Normally distributed) data with mean `mbar` and variance `sigma^2`.

In cases where no predefined function is available, it is possible to transform an available distribution, say $p(d)$, into the desired distribution, say $q(m)$, using the transformation rule

$$p[d(m)] \frac{dd}{dm} = q(m) \quad (2.39)$$

Most software environments provide a predefined function for realizations of a uniform distribution on the interval (0,1). Then, since $p(d) = 1$, Equation (2.39) is a differential equation for $d(m)$

$$\frac{dd}{dm} = q(m) \quad \text{or} \quad d = \int q(m) dm = Q(m) \quad (2.40)$$

Here, $Q(m)$ is the cumulative probability distribution corresponding to $q(m)$. The transformation is then $m = Q^{-1}(d)$; that is, one must invert the cumulative probability distribution to give the value of m for which the probability d occurs. Thus, the transformation requires that the inverse cumulative probability distribution be known.

MatLab provides a `norminv()` function that calculates the inverse cumulative probability distribution in the Gaussian case, as well as a `random('unif',...)` function that returns realizations of the uniform probability density function. Thus,

```
d = random('unif', 0, 1, N, 1);
m = norminv(d, mbar, sigma);
```

(*MatLab* script gda02_14)

creates N realizations of a Gaussian probability density function (Figure 2.16). Such an approach offers no advantage in the Gaussian case, since the `random('Normal',...)` function is available. It is of practical use in cases not supported by *MatLab*, as long as an appropriate `Qinv()` function can be provided.

Another method of producing a vector of realizations of a random variable is the *Metropolis-Hastings algorithm*. It is a useful alternative to the transformation method described above, especially since it requires evaluating only the probability density function $p(d)$ and not its cumulative inverse. It is an iterative algorithm that builds the vector \mathbf{d} element by element. The first element, d_1 , is set to an arbitrary number, such as zero. Subsequent elements are generated in sequence, with an element d_i , generating a successor d_{i+1} according to this algorithm: First, randomly draw a *proposed* successor d' from a conditional probability density function $q(d'|d_i)$. The exact form of $q(d'|d_i)$ is arbitrary; however, it must be chosen so that d' is typically in the neighborhood of d_i . One possible choice is the Gaussian function

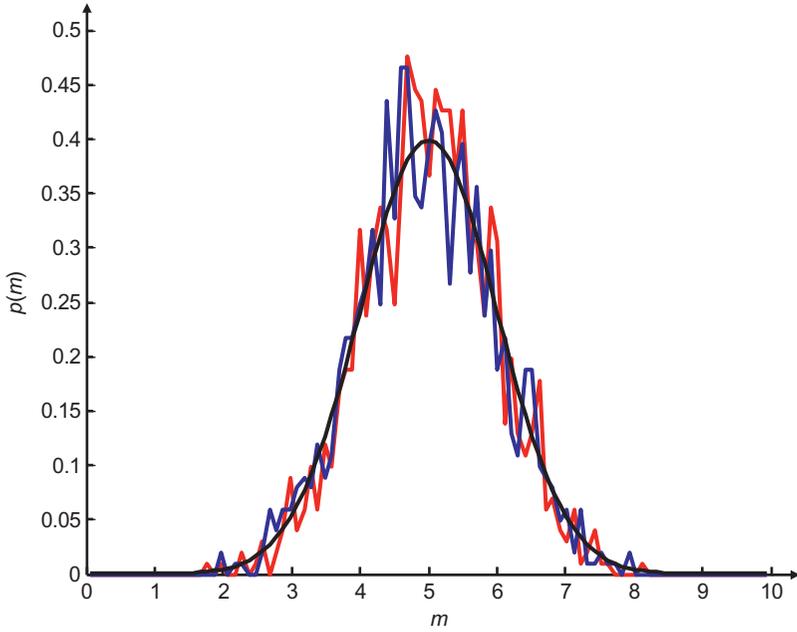


FIGURE 2.16 Gaussian probability density function $p(m)$ with mean 5 and variance 1^2 . (Red curve) Computed by binning 1000 realizations of a random variable generated using *MatLab*'s random ("Normal",...) function. (Blue) Computed by binning 1000 realizations of a random variable generated by transforming a uniform distribution. (Black) Exact formula. *MatLab* script gda02_14.

$$q(d'|d_i) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left\{-\frac{(d' - d_i)^2}{2\sigma^2}\right\} \quad (2.41)$$

Here, σ represents the size of the neighborhood, that is, the typical deviation of d' away from d_i . Second, generate a random number α drawn from a uniform distribution on the interval (0,1). Third, accept the proposed successor and set $d_{i+1} = d'$ if

$$\alpha < \frac{p(d')q(d_i|d')}{p(d_i)q(d'|d_i)} \quad (2.42)$$

Otherwise, set $d_{i+1} = d_i$. When repeated many times, this algorithm leads to a vector \mathbf{d} that has approximately the probability density function $p(d)$ (Figure 2.17). Note that the conditional probability density functions cancels from Equation (2.42) when $q(d'|d_i) = q(d_i|d')$, as is the case for the Gaussian in Equation (2.41).

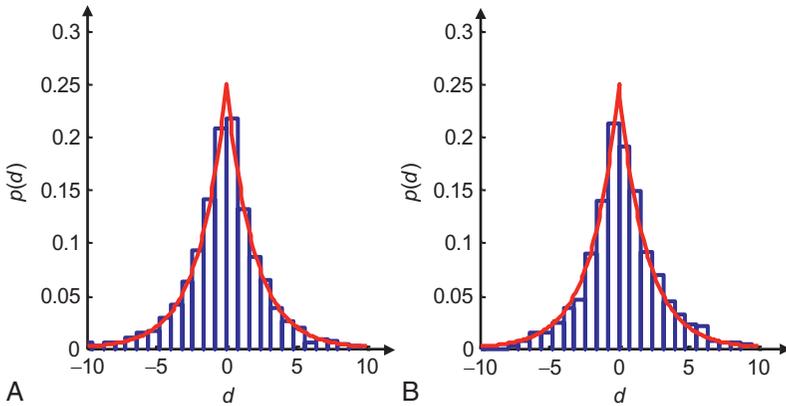


FIGURE 2.17 Histograms (blue curves) of 5000 realizations of a random variable d for the probability density function (red curves) $p(d) = \frac{1}{2}c \exp(-|d|/c)$ with $c=2$. (A) Realizations computed by transforming data drawn from a uniform distribution and (B) realizations computed using the Metropolis-Hastings algorithm. *MatLab* script gda02_14.

2.9 PROBLEMS

- 2.1.** What is the mean and variance of the uniform distribution $p(d) = 1$ on the interval $(0,1)$?
- 2.2.** Suppose d is a Gaussian random variable with zero mean and unit variance. What is the probability density function of $E = e^2$? Hint: Since the sign of d gets lost when it is squared, you can assume that $p(d)$ is one-sided, that is, defined for only $d \geq 0$ and with twice the amplitude of the usual Gaussian.
- 2.3.** Write a *MatLab* script that uses the `random()` function to create a vector \mathbf{d} of $N = 1000$ realizations of a Gaussian-distributed random variable with mean $\langle d \rangle = 4$ and variance $\sigma_d^2 = 2^2$. Count up the number of instances where $d_i > (\langle d \rangle + 2\sigma_d)$. Is this about the number you expected?
- 2.4.** Suppose that the data are uncorrelated with uniform variance, $[\text{cov } \mathbf{d}] = \sigma_d^2 \mathbf{I}$, and that the model parameters are linear functions of the data, $\mathbf{m} = \mathbf{M}\mathbf{d}$. (A) What property must \mathbf{M} have for the model parameters to be uncorrelated with uniform variance σ_m^2 ? (B) Express this property in terms of the rows of the \mathbf{M} .
- 2.5.** Use the transformation method to compute realizations of the probability density function $p(m) = 3m^2$ on the interval $(0,1)$, starting from realizations of the uniform distribution $p(d) = 1$. Check your results by plotting a histogram.

REFERENCES

Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc, Oxford, UK 263pp.

Solution of the Linear, Gaussian Inverse Problem, Viewpoint 1: The Length Method

3.1 THE LENGTHS OF ESTIMATES

The simplest of methods for solving the linear inverse problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ is based on measures of the size, or length, of the estimated model parameters \mathbf{m}^{est} and of the predicted data $\mathbf{d}^{\text{pre}} = \mathbf{G}\mathbf{m}^{\text{est}}$.

To see that measures of length can be relevant to the solution of inverse problems, consider the simple problem of fitting a straight line to data (Figure 3.1). This problem is often solved by the so-called method of least squares. In this method, one tries to pick the model parameters (intercept and slope) so that the predicted data are as close as possible to the observed data. For each observation, one defines a prediction error, or misfit, $e_i = d_i^{\text{obs}} - d_i^{\text{pre}}$. The best-fit line is then the one with model parameters that lead to the smallest overall error E , defined as

$$E = \sum_{i=1}^N e_i^2 = \mathbf{e}^T \mathbf{e} \quad (3.1)$$

The total error E (the sum of the squares of the individual errors) is exactly the squared Euclidean length of the vector \mathbf{e} , or $E = \mathbf{e}^T \mathbf{e}$.

The method of least squares estimates the solution of an inverse problem by finding the model parameters that minimize a particular measure of the length of the prediction error, $\mathbf{e} = \mathbf{d}^{\text{obs}} - \mathbf{d}^{\text{pre}}$, namely, its Euclidean length. As will be detailed below, it is the simplest of the methods that use measures of length as the guiding principle in solving an inverse problem.

3.2 MEASURES OF LENGTH

Note that although the Euclidean length is one way of quantifying the size or length of a vector, it is by no means the only possible measure. For instance, one could equally well quantify length by summing the absolute values of the elements of the vector.

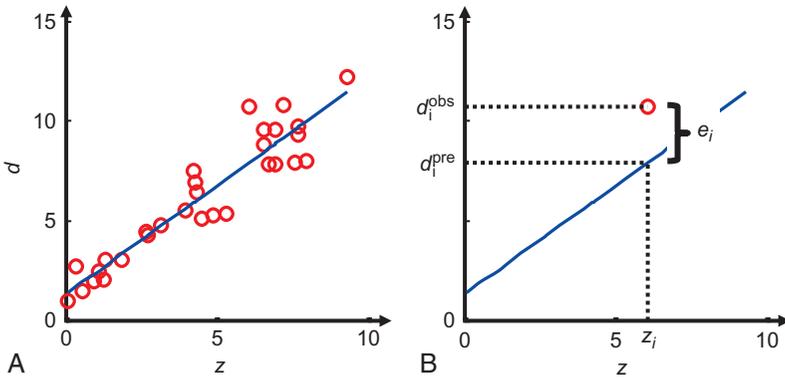


FIGURE 3.1 (A) Least squares fitting of a straight line to (z, d) data. (B) The error e_i for each observation is the difference between the observed and predicted datum: $e_i = d_i^{obs} - d_i^{pre}$. *MatLab* script gda03_01.

The term *norm* is used to refer to some measure of length or size and is indicated by a set of double vertical bars; that is, $\|\mathbf{e}\|$ is the norm of the vector \mathbf{e} . The most commonly employed norms are those based on the sum of some power of the elements of a vector and are given the name L_n , where n is the power:

$$L_1 \text{ norm: } \|\mathbf{e}\|_1 = \left[\sum_i |e_i|^1 \right] \quad (3.2a)$$

$$L_2 \text{ norm: } \|\mathbf{e}\|_2 = \left[\sum_i |e_i|^2 \right]^{1/2} \quad (3.2b)$$

$$L_n \text{ norm: } \|\mathbf{e}\|_n = \left[\sum_i |e_i|^n \right]^{1/n} \quad (3.2c)$$

Successively higher norms give the largest element of \mathbf{e} successively larger weight. The limiting case of $n \rightarrow \infty$ gives nonzero weight to only the largest element (Figure 3.2); therefore, it is equivalent to the selection of the vector element with largest absolute value as the measure of length and is written as

$$L_\infty \text{ norm: } \|\mathbf{e}\|_\infty = \max_i |e_i| \quad (3.2d)$$

The method of least squares uses the L_2 norm to quantify length. It is appropriate to inquire why this, and not some other choice of norm, is used. The answer involves the way in which one chooses to weight data *outliers* that fall far from the average trend (Figure 3.3). If the data are very accurate, then the fact that one prediction falls far from its observed value is important. A high-order norm is

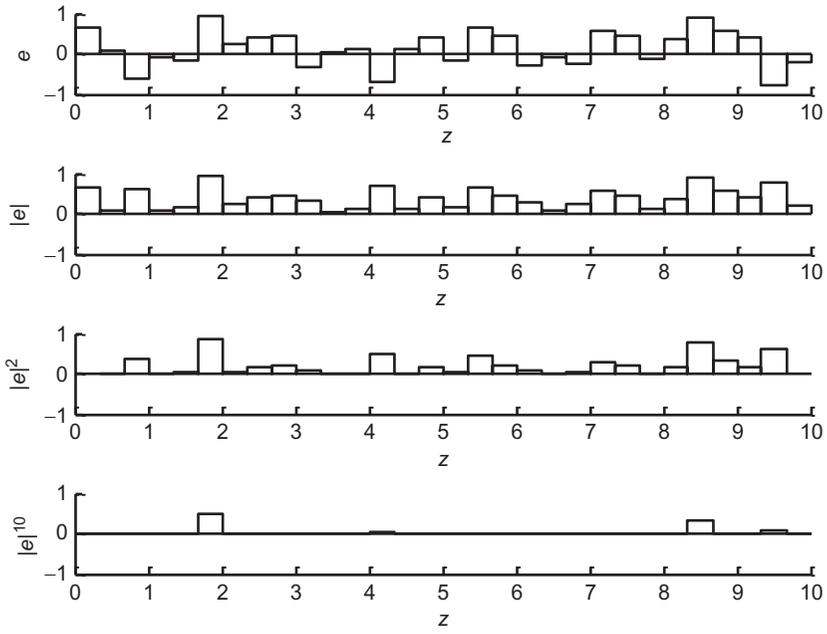


FIGURE 3.2 Hypothetical prediction error, $e_i(z_i)$, and its absolute value, raised to the powers of 1, 2, and 10. While most elements of $|e_i|$ are numerically significant, only a few elements of $|e_i|^{10}$ are. *MatLab* script gda03_02.

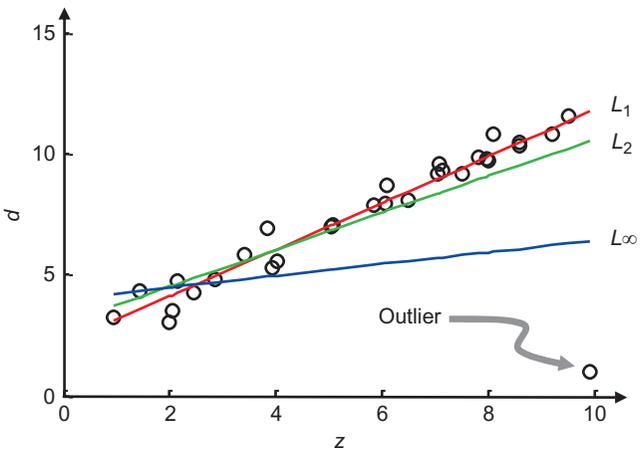


FIGURE 3.3 Straight line fits to (z, d) pairs where the error is measured under the L_1 , L_2 and L_∞ norms. The L_1 norm gives the least weight to the one outlier. *MatLab* script gda03_03.

used, since it weights the larger errors preferentially. On the other hand, if the data are expected to scatter widely about the trend, then no significance can be placed upon a few large prediction errors. A low-order norm is used, since it gives more equal weight to errors of different sizes.

As will be discussed in more detail later, the L_2 norm implies that the data obey Gaussian statistics. Gaussians are rather short-tailed functions, so it is appropriate to place considerable weight on any data that have a large prediction error.

The likelihood of an observed datum falling far from the trend depends on the shape of the distribution for that datum. Long-tailed distributions imply many scattered (improbable) points. Short-tailed distributions imply very few scattered points (Figure 3.4). The choice of a norm, therefore, implies an assertion that the data obey a particular type of statistics.

Even though many measurements have approximately Gaussian statistics, most data sets generally have a few outliers that are wildly improbable. The occurrence of these points demonstrates that the assumption of Gaussian statistics is in error, especially in the tails of the distribution. If one applies least squares to this kind of problem, the estimates of the model parameters can be completely erroneous. Least squares weights large errors so heavily that even one “bad” data point can completely throw off the result. In these situations, methods based on the L_1 norm give more reliable estimates. (Methods that can tolerate a few bad data are said to be *robust*.)

Matrix norms can be defined in a manner similar to vector norms (see Equation (3.3d)). Vector and matrix norms obey the following relationships:

Vector norms:

$$\|\mathbf{x}\| > 0 \text{ as long as } \mathbf{x} \neq 0 \quad (3.3a)$$

$$\|a\mathbf{x}\| = |a|\|\mathbf{x}\| \quad (3.3b)$$

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad (3.3c)$$

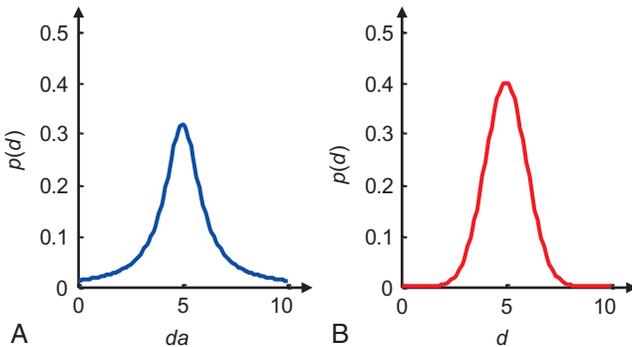


FIGURE 3.4 (A) Long-tailed probability density function. (B) Short-tailed probability density function. *MatLab* script gda03_04.

Matrix norms:

$$\|\mathbf{A}\|_2 = \left(\sum_{i=1}^N \sum_{j=1}^N A_{ij}^2 \right)^{1/2} \quad (3.3d)$$

$$\|c\mathbf{A}\| = |c| \|\mathbf{A}\| \quad (3.3e)$$

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \quad (3.3f)$$

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\| \quad (3.3g)$$

$$\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\| \quad (3.3h)$$

Equations (3.3c) and (3.3h) are called *triangle inequalities* because of their similarity to Pythagoras's law for right triangles.

3.3 LEAST SQUARES FOR A STRAIGHT LINE

The elementary problem of fitting a straight line to data illustrates the basic procedures applied in this technique. The model is the assertion that the data can be described by the linear equation $d_i = m_1 + m_2 z_i$. Note that there are two model parameters, $M=2$, and that typically there are many more than two data, $N > M$. Since a line is defined by precisely two points, it is clearly impossible to choose a straight line that passes through every one of the data, except in the instance that they all lie precisely on the same straight line. Collinearity rarely occurs when measurements are influenced by noise.

As we shall discuss in more detail below, the fact that the equation $d_i = m_1 + m_2 z_i$ cannot be satisfied for every i means that the inverse problem is *overdetermined*; that is, it has no solution for which $\mathbf{e} = 0$. One therefore seeks values of the model parameters that solve $d_i = m_1 + m_2 z_i$ approximately, where the goodness of the approximation is defined by the error

$$E = \mathbf{e}^T \mathbf{e} = \sum_{i=1}^N (d_i - m_1 - m_2 z_i)^2 \quad (3.4)$$

This problem is then the elementary calculus problem of locating the minimum of the function $E(m_1, m_2)$ and is solved by setting the derivatives of E to zero and solving the resulting equations.

$$\begin{aligned} \frac{\partial E}{\partial m_1} &= \frac{\partial}{\partial m_1} \sum_{i=1}^N [d_i - m_1 - m_2 z_i]^2 = 2Nm_1 + 2m_2 \sum_{i=1}^N z_i - 2 \sum_{i=1}^N d_i = 0 \\ \frac{\partial E}{\partial m_2} &= \frac{\partial}{\partial m_2} \sum_{i=1}^N [d_i - m_1 - m_2 z_i]^2 = 2m_1 \sum_{i=1}^N z_i + 2m_2 \sum_{i=1}^N z_i^2 - 2 \sum_{i=1}^N z_i d_i = 0 \end{aligned} \quad (3.5)$$

These two equations are then solved simultaneously for m_1 and m_2 , yielding the classic formulas for the least squares fitting of a line.

3.4 THE LEAST SQUARES SOLUTION OF THE LINEAR INVERSE PROBLEM

Least squares can be extended to the general linear inverse problem in a very straightforward manner. Again, one computes the derivative of the error E with respect to one of the model parameters, say, m_q , and sets the result to zero. The error E is

$$E = \mathbf{e}^T \mathbf{e} = (\mathbf{d} - \mathbf{Gm})^T (\mathbf{d} - \mathbf{Gm}) = \sum_{i=1}^N \left[d_i - \sum_{j=1}^M G_{ij} m_j \right] \left[d_i - \sum_{k=1}^M G_{ik} m_k \right] \quad (3.6)$$

Note that the indices on the sums within the parentheses are different dummy variables, to prevent confusion. Multiplying out the terms and reversing the order of the summations lead to

$$E = \sum_{j=1}^M \sum_{k=1}^M m_j m_k \sum_{i=1}^N G_{ij} G_{ik} - 2 \sum_{j=1}^M m_j \sum_{i=1}^N G_{ij} d_i + \sum_{i=1}^N d_i d_i \quad (3.7)$$

The derivatives $\partial E / \partial m_q$ are now computed. Performing this differentiation term by term gives

$$\begin{aligned} \frac{\partial}{\partial m_q} \left[\sum_{j=1}^M \sum_{k=1}^M m_j m_k \sum_{i=1}^N G_{ij} G_{ik} \right] &= \sum_{j=1}^M \sum_{k=1}^M [\delta_{jq} m_k + m_j \delta_{kq}] \sum_{i=1}^N G_{ij} G_{ik} \\ &= 2 \sum_{k=1}^M m_k \sum_{i=1}^N G_{iq} G_{ik} \end{aligned} \quad (3.8)$$

for the first term. Since the model parameters are independent variables, derivatives of the form $\partial m_i / \partial m_j$ are either unity, when $i = j$, or zero, when $i \neq j$. Thus, $\partial m_i / \partial m_j$ is just the Kronecker delta δ_{ij} (see Section I.4) and the formula containing it can be simplified trivially. The second term gives

$$-2 \frac{\partial}{\partial m_q} \left[\sum_{j=1}^M m_j \sum_{i=1}^N G_{ij} d_i \right] = -2 \sum_{j=1}^M \delta_{jq} \sum_{i=1}^N G_{ij} d_i = -2 \sum_{i=1}^N G_{iq} d_i \quad (3.9)$$

Since the third term does not contain any ms , it is zero as

$$\frac{\partial}{\partial m_q} \left[\sum_{i=1}^N d_i d_i \right] = 0 \quad (3.10)$$

Combining the three terms gives

$$\frac{\partial E}{\partial m_q} = 0 = 2 \sum_{k=1}^M m_k \sum_{i=1}^N G_{iq} G_{ik} - 2 \sum_{i=1}^N G_{iq} d_i \quad (3.11)$$

Writing this equation in matrix notation yields

$$\mathbf{G}^T \mathbf{G} \mathbf{m} - \mathbf{G}^T \mathbf{d} = 0 \quad (3.12)$$

Note that the quantity $\mathbf{G}^T \mathbf{G}$ is a square $M \times M$ matrix and that it multiplies a vector \mathbf{m} of length M . The quantity $\mathbf{G}^T \mathbf{d}$ is also a vector of length M . This equation is therefore a square matrix equation for the unknown model parameters. Presuming that $[\mathbf{G}^T \mathbf{G}]^{-1}$ exists (an important question that we shall return to later), we have the following estimate for the model parameters:

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d} \quad (3.13)$$

which is the least squares solution to the inverse problem $\mathbf{G} \mathbf{m} = \mathbf{d}$.

When the dimensions of \mathbf{G} are small (say, N and M less than a few hundred), the least squares solution is computed as

$$\text{mest} = (\mathbf{G}' * \mathbf{G}) \setminus (\mathbf{G}' * \mathbf{d}); \quad (\text{MatLab script gda03_05})$$

However, for larger problems, the computational cost of computing $\mathbf{G}^T \mathbf{G}$ can be prohibitive. Furthermore, $\mathbf{G}^T \mathbf{G}$ is rarely as sparse as \mathbf{G} itself. In this case, an iterative matrix solver, such as the *biconjugate gradient algorithm*, is preferred. This algorithm only requires products of the form $\mathbf{G}^T \mathbf{G} \mathbf{v}$, where \mathbf{v} is a vector constructed by the algorithm, and this product can be performed as $\mathbf{G}^T (\mathbf{G} \mathbf{v})$ so that $\mathbf{G}^T \mathbf{G}$ is never explicitly calculated (e.g., [Menke and Menke, 2011](#), their [Section 5.8](#)). *MatLab* provides a `bicg()` function, which is as follows:

$$\begin{aligned} \text{tol} &= 1\text{e-}6; \\ \text{maxit} &= 3 * N; \\ \text{mest2} &= \text{bicg}(\text{@least_squaresfcn}, \mathbf{G}' * \mathbf{d}, \text{tol}, \text{maxit}); \end{aligned} \quad (\text{MatLab script gda03_05})$$

The algorithm involves iteratively improving a solution, with the iterations terminating when the error is less than a tolerance `tol` or when `maxit` iterations have been performed (whichever comes first). The first argument is a *handle* to a user-provided `least_squaresfcn()` function that calculates $\mathbf{G}^T (\mathbf{G} \mathbf{v})$:

$$\begin{aligned} \text{function } y &= \text{least_squaresfcn}(v, \text{transp_flag}) \\ \text{global } G; \\ \text{temp} &= G * v; \\ y &= G' * \text{temp}; \\ \text{return} \end{aligned}$$

(*MatLab* script gda03_05)

This function is stored in the file `leastsquaresfcn.m`. In order for this function to perform correctly, *MatLab* must be instructed that the \mathbf{G} being used within it is the same as is defined in the main script. This is accomplished by placing the commands

```
clear G;
global G;
```

(*MatLab* script `gda03_05`)

at the beginning of the main script. Although this algorithm will work for any \mathbf{G} , it is particularly useful when \mathbf{G} has been defined as sparse using the `spalloc()` function. An example is given in *MatLab* script `gda03_06`.

3.5 SOME EXAMPLES

3.5.1 The Straight Line Problem

In the straight line problem, the model is $d_i = m_1 + m_2 z_i$, so the equation $\mathbf{Gm} = \mathbf{d}$ has the form

$$\begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \quad (3.14)$$

In *MatLab*, the matrix \mathbf{G} can be created with the command

```
G = [ones(N,1), z];
```

(*MatLab* script `gda03_05`)

The matrix products required by the least squares solution are

$$\mathbf{G}^T \mathbf{G} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_N \end{bmatrix} \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} = \begin{bmatrix} N & \sum_{i=1}^N z_i \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 \end{bmatrix} \quad (3.15)$$

and

$$\mathbf{G}^T \mathbf{d} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_N \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N d_i z_i \end{bmatrix} \quad (3.16)$$

This gives the least squares solution

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d} = \begin{bmatrix} N & \sum_{i=1}^N z_i \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N d_i z_i \end{bmatrix} \quad (3.17)$$

3.5.2 Fitting a Parabola

The problem of fitting a parabola is a trivial generalization of fitting a straight line (Figure 3.4). Now the model is $d_i = m_1 + m_2 z_i + m_3 z_i^2$, so the equation $\mathbf{Gm} = \mathbf{d}$ has the form

$$\begin{bmatrix} 1 & z_1 & z_1^2 \\ 1 & z_2 & z_2^2 \\ \vdots & \vdots & \vdots \\ 1 & z_N & z_N^2 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \quad (3.18)$$

In *MatLab*, the matrix \mathbf{G} can be created with the command

```
G = [ones(N,1), z, z.^2];
```

(*MatLab* script gda03_07)

The matrix products required by the least squares solution are as follows:

$$\begin{aligned} \mathbf{G}^T \mathbf{G} &= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_N \\ z_1^2 & z_2^2 & \cdots & z_N^2 \end{bmatrix} \begin{bmatrix} 1 & z_1 & z_1^2 \\ 1 & z_2 & z_2^2 \\ \vdots & \vdots & \vdots \\ 1 & z_N & z_N^2 \end{bmatrix} \\ &= \begin{bmatrix} N & \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 & \sum_{i=1}^N z_i^3 \\ \sum_{i=1}^N z_i^2 & \sum_{i=1}^N z_i^3 & \sum_{i=1}^N z_i^4 \end{bmatrix} \end{aligned} \quad (3.19)$$

and

$$\mathbf{G}^T \mathbf{d} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_N \\ z_1^2 & z_2^2 & \cdots & z_N^2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N z_i d_i \\ \sum_{i=1}^N z_i^2 d_i \end{bmatrix} \quad (3.20)$$

giving the least squares solution

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d} = \begin{bmatrix} N & \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 & \sum_{i=1}^N z_i^3 \\ \sum_{i=1}^N z_i^2 & \sum_{i=1}^N z_i^3 & \sum_{i=1}^N z_i^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N z_i d_i \\ \sum_{i=1}^N z_i^2 d_i \end{bmatrix} \quad (3.21)$$

An example of using a quadratic fit to examine Kepler's third law is shown in Figure 3.5.

3.5.3 Fitting a Plane Surface

To fit a plane surface, two auxiliary variables, say, x and y , are needed. The model is

$$d_i = m_1 + m_2 x_i + m_3 y_i \quad (3.22)$$

so the equation $\mathbf{Gm} = \mathbf{d}$ has the form

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \quad (3.23)$$

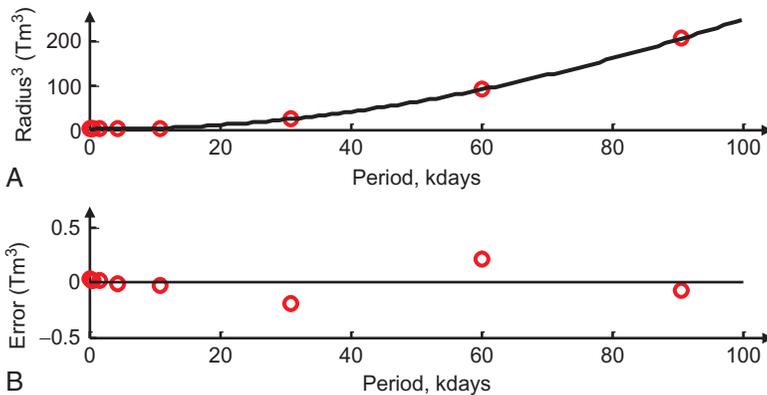


FIGURE 3.5 Test of Kepler's third law, which states that the cube of the orbital radius of a planet equals the square of its orbital period. (A) Data (red circles) for our solar system are least squares fit with a quadratic formula $d_i = m_1 + m_2 z_i + m_3 z_i^2$, where d_i is radius cubed and z_i is period. (B) Error of fit. A separate graph is used so that the error can be plotted at a meaningful scale. Data courtesy of Wikipedia. *MatLab* script gda03_07.

In *MatLab*, the matrix \mathbf{G} can be created with the command

$$\mathbf{G} = [\text{ones}(N, 1), \mathbf{x}, \mathbf{y}];$$

(*MatLab* script gda03_08)

forming the matrix products $\mathbf{G}^T \mathbf{G}$

$$\begin{aligned} \mathbf{G}^T \mathbf{G} &= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix} \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix} \\ &= \begin{bmatrix} N & \sum_{i=1}^N x_i & \sum_{i=1}^N y_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 & \sum_{i=1}^N x_i y_i \\ \sum_{i=1}^N y_i & \sum_{i=1}^N x_i y_i & \sum_{i=1}^N y_i^2 \end{bmatrix} \end{aligned} \quad (3.24)$$

and

$$\mathbf{G}^T \mathbf{d} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N x_i d_i \\ \sum_{i=1}^N y_i d_i \end{bmatrix} \quad (3.25)$$

giving the least squares solution

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d} = \begin{bmatrix} N & \sum_{i=1}^N x_i & \sum_{i=1}^N y_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 & \sum_{i=1}^N x_i y_i \\ \sum_{i=1}^N y_i & \sum_{i=1}^N x_i y_i & \sum_{i=1}^N y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N x_i d_i \\ \sum_{i=1}^N y_i d_i \end{bmatrix} \quad (3.26)$$

An example of using a planar fit to examine earthquakes on a geologic fault is shown in [Figure 3.6](#).

3.6 THE EXISTENCE OF THE LEAST SQUARES SOLUTION

The least squares solution arose from consideration of an inverse problem that had no exact solution. Since there was no exact solution, we chose to do the next best thing: to estimate the solution by those values of the model parameters that gave the best approximate solution (where “best” meant minimizing the L_2 prediction error). By writing a single formula $\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d}$, we implicitly assumed that there

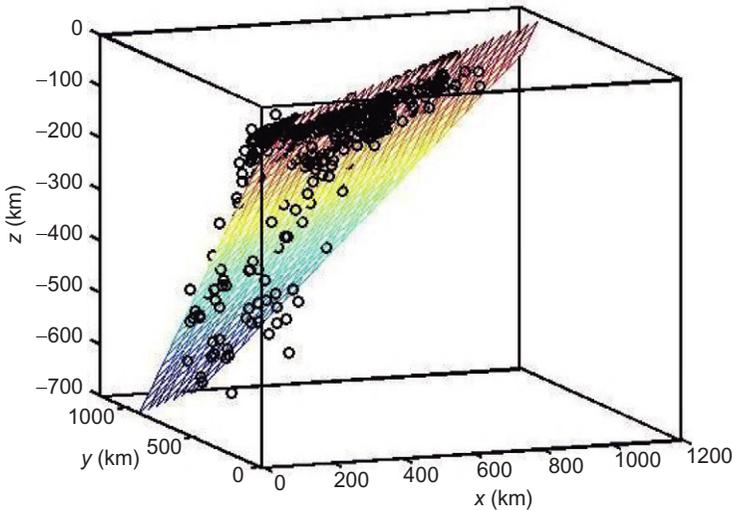


FIGURE 3.6 (Circles) Earthquakes in the Kurile subduction zone, northwest Pacific Ocean. The x -axis points north and the y -axis east. The earthquakes scatter about a dipping planar surface (colored grid), determined using least squares. Data courtesy of the U.S. Geological Survey. *MatLab* script gda03_08.

was only one such “best” solution. As we shall prove later, least squares fails if the number of solutions that give the same minimum prediction error is greater than one.

To see that least squares fails for problems with nonunique solutions, consider the straight line problem with only one data point (Figure 3.7). It is clear that this problem is nonunique; many possible lines can pass through the point, and each has zero prediction error. The solution then contains the following expression:

$$[\mathbf{G}^T \mathbf{G}]^{-1} = \begin{bmatrix} N & \sum_{i=1}^N z_i \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 \end{bmatrix}^{-1} \rightarrow \begin{bmatrix} 1 & z_1 \\ z_1 & z_1^2 \end{bmatrix}^{-1} \quad (3.27)$$

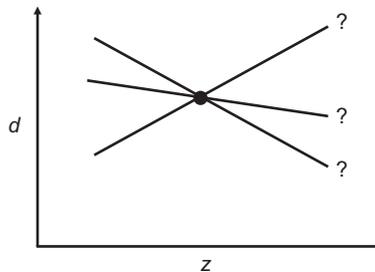


FIGURE 3.7 An infinity of different lines can pass through a single point. The prediction error for each is $E=0$.

The inverse of a matrix is proportional to the reciprocal of the determinant of the matrix so that

$$[\mathbf{G}^T \mathbf{G}]^{-1} \propto \left(\det \begin{bmatrix} 1 & z_1 \\ z_1 & z_1^2 \end{bmatrix} \right)^{-1} = \frac{1}{z_1^2 - z_1^2} \quad (3.28)$$

This expression clearly is singular. The formula for the least squares solution fails.

The question of whether the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ provides enough information to specify uniquely the model parameters serves as a basis for classifying inverse problems. A classification system based on this criterion is discussed in Sections 3.6.1–3.6.3.

3.6.1 Underdetermined Problems

When the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ does not provide enough information to determine uniquely all the model parameters, the problem is said to be *underdetermined*. As we saw in the example above, this can happen if there are several solutions that have zero prediction error. From elementary linear algebra, we know that underdetermined problems occur when there are more unknowns than data, that is, when $M > N$. We must note, however, that there is no special reason why the prediction error must be zero for an underdetermined problem. Frequently, the data uniquely determine some of the model parameters but not others. For example, consider the acoustic experiment in Figure 3.8. Since no measurements are made of the acoustic slowness in the second brick, it is clear that this model parameter is completely unconstrained by the data. In contrast, the acoustic slowness of the first brick is *overdetermined*, since in the presence of measurement noise, no choice of s_1 can satisfy the data exactly. The equation describing this experiment is

$$h \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \quad (3.29)$$

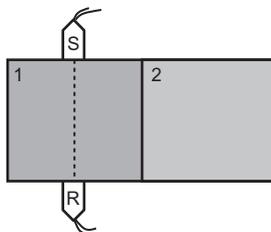


FIGURE 3.8 An acoustic travel time experiment with source S and receiver R in a medium consisting of two discrete bricks. Because of poor experimental geometry, the acoustic waves (dashed line) pass only through brick 1. The slowness of brick 2 is completely underdetermined.

where s_i is the slowness in the i th brick, h the brick width, and the d_i the measurements of travel time. If one were to attempt to solve this problem with least squares, one would find that the term $[\mathbf{G}^T\mathbf{G}]^{-1}$ is singular. Even though $M < N$, the problem is still underdetermined since the data kernel has a very poor structure. Although this is a rather trivial case in which only some of the model parameters are underdetermined, in realistic experiments, the problem arises in more subtle forms.

We shall refer to underdetermined problems that have nonzero prediction error as *mixed-determined problems*, to distinguish them from *purely underdetermined problems* that have zero prediction error.

3.6.2 Even-Determined Problems

In even-determined problems, there is exactly enough information to determine the model parameters. There is only one solution, and it has zero prediction error.

3.6.3 Overdetermined Problems

When there is too much information contained in the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ for it to possess an exact solution, we speak of it as being *overdetermined*. This is the case in which we can employ least squares to select a “best” approximate solution. Overdetermined problems typically have more data than unknowns, that is, $N > M$, although for the reasons discussed above it is possible to have problems that are to some degree overdetermined even when $N < M$ and to have problems that are to some degree underdetermined even when $N > M$.

To deal successfully with the full range of inverse problems, we shall need to be able to characterize whether an inverse problem is under- or overdetermined (or some combination of the two). We shall develop quantitative methods for making this characterization in [Chapter 7](#). For the moment, we assume that it is possible to characterize the problem intuitively on the basis of the kind of experiment the problem represents.

3.7 THE PURELY UNDERDETERMINED PROBLEM

Suppose that an inverse problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ has been identified as one that is purely underdetermined. For simplicity, assume that there are fewer equations than unknown model parameters, that is, $N < M$, and that there are no inconsistencies in these equations. It is therefore possible to find more than one solution for which the prediction error E is zero. (In fact, we shall show that underdetermined linear inverse problems have an infinite number of such solutions.) Although the data provide information about the model parameters, they do not provide enough to determine them uniquely.

We must have some means of singling out precisely one of the infinite number of solutions with zero prediction error E to obtain a unique solution \mathbf{m}^{est} to

the inverse problem. To do this, we must add to the problem some information not contained in the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$. This extra information is called *a priori* information (Jackson, 1979). *A priori* information can take many forms, but in each case, it quantifies expectations about the character of the solution that are not based on the actual data.

For instance, in the case of fitting a straight line through a single data point, one might have the expectation that the line also passes through the origin. This *a priori* information now provides enough information to solve the inverse problem uniquely, since two points (one datum, one *a priori*) determine a line.

Another example of *a priori* information concerns expectations that the model parameters possess a given sign or lie in a given range. For instance, suppose the model parameters represent density at different points in the earth. Even without making any measurements, one can state with certainty that the density is everywhere positive, since density is an inherently positive quantity. Furthermore, since the interior of the earth can reasonably be assumed to be rock, its density must have values in some range known to characterize rock, say, between 1000 and 10,000 kg/m³. If one can use this *a priori* information when solving the inverse problem, it may greatly reduce the range of possible solutions—or even cause the solution to be unique.

There is something unsatisfying about having to add *a priori* information to an inverse problem to single out a solution. Where does this information come from, and how certain is it? There are no firm answers to these questions. In certain instances, one might be able to identify reasonable *a priori* assumptions; in other instances, one might not. Clearly, the importance of the *a priori* information depends greatly on the *use* one plans for the estimated model parameters. If one simply wants one example of a solution to the problem, the choice of *a priori* information is unimportant. However, if one wants to develop arguments that depend on the uniqueness of the estimates, the validity of the *a priori* assumptions is of paramount importance. These problems are the price one must pay for estimating the model parameters of a nonunique inverse problem. As will be shown in Chapter 6, there are other kinds of “answers” to inverse problems that do not depend on *a priori* information (localized averages, for example). However, these “answers” invariably are not as easily interpretable as estimates of model parameters.

The first kind of *a priori* assumption we shall consider is the expectation that the solution to the inverse problem is *simple*, where the notion of simplicity is quantified by some measure of the length of the solution. One such measure is simply the Euclidean length of the solution, $L = \mathbf{m}^T \mathbf{m} = \sum m_i^2$. A solution is therefore defined to be simple if it is small when measured under the L_2 norm. Admittedly, this measure is perhaps not a particularly realistic measure of simplicity. It can be useful occasionally, and we shall describe shortly how it can be generalized to more realistic measures. One instance in which solution length may be realistic is when the model parameters describe the velocity of various points in a moving fluid. The length L is then a measure of the kinetic energy of

the fluid. In certain instances, it may be appropriate to find that velocity field in the fluid that has the smallest possible kinetic energy of those solutions satisfying the data.

We pose the following problem: Find the \mathbf{m}^{est} that minimizes $L = \mathbf{m}^T \mathbf{m} = \sum m_i^2$ subject to the constraint that $\mathbf{e} = \mathbf{d} - \mathbf{G}\mathbf{m} = \mathbf{0}$. This problem can easily be solved by the method of Lagrange multipliers (see Section 14.1). We minimize the function as

$$\Phi(\mathbf{m}) = L + \sum_{i=1}^N \lambda_i e_i = \sum_{i=1}^M m_i^2 + \sum_{i=1}^N \lambda_i \left[d_i - \sum_{j=1}^M G_{ij} m_j \right] \quad (3.30)$$

with respect to m_q , where λ_i are the Lagrange multipliers. Taking the derivatives yields

$$\frac{\partial \Phi}{\partial m_q} = \sum_{i=1}^M 2 \frac{\partial m_i}{\partial m_q} m_i - \sum_{i=1}^N \lambda_i \sum_{j=1}^M G_{ij} \frac{\partial m_j}{\partial m_q} = 2m_q - \sum_{i=1}^N \lambda_i G_{iq} \quad (3.31)$$

Setting this result to zero and rewriting it in matrix notation yield the equation $2\mathbf{m} = \mathbf{G}^T \boldsymbol{\lambda}$, which must be solved along with the constraint equation $\mathbf{G}\mathbf{m} = \mathbf{d}$. Plugging the first equation into the second gives $\mathbf{d} = \mathbf{G}\mathbf{m} = \mathbf{G}[\mathbf{G}^T \boldsymbol{\lambda} / 2]$. We note that the matrix $\mathbf{G}\mathbf{G}^T$ is a square $N \times N$ matrix. If its inverse exists, we can then solve this equation for the Lagrange multipliers, $\boldsymbol{\lambda} = 2[\mathbf{G}\mathbf{G}^T]^{-1} \mathbf{d}$. Then inserting this expression into the first equation yields the solution

$$\mathbf{m}^{\text{est}} = \mathbf{G}^T [\mathbf{G}\mathbf{G}^T]^{-1} \mathbf{d} \quad (3.32)$$

We shall discuss the conditions under which this solution exists later. As we shall see, one condition is that the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ be purely underdetermined—that it contain no inconsistencies.

3.8 MIXED-DETERMINED PROBLEMS

Most inverse problems that arise in practice are neither completely overdetermined nor completely underdetermined. For instance, in the X-ray tomography problem, there may be one box through which several rays pass (Figure 3.9A). The X-ray opacity of this box is clearly overdetermined. On the other hand, there may be boxes that have been missed entirely (Figure 3.9B). These boxes are completely undetermined. There may also be boxes that cannot be individually resolved because every ray that passes through one also passes through an equal distance of the other (Figure 3.9C). These boxes are also underdetermined, since only their mean opacity is determined.

Ideally, we would like to sort the unknown model parameters into two groups: those that are overdetermined and those that are underdetermined. Actually, to do this, we need to form a new set of model parameters that are linear combinations of the old. For example, in the two-box problem above, the average

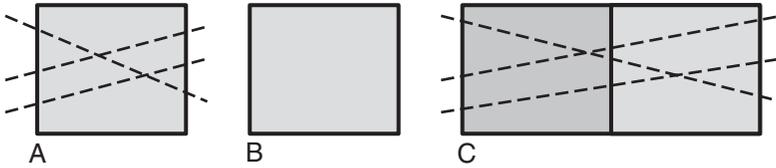


FIGURE 3.9 (A) The X-ray opacity of the brick is overdetermined, since measurements of X-ray intensity are made along three different paths (dashed lines). (B) The opacity is underdetermined, since no measurements have been made. (C) The average opacity of these two bricks is overdetermined, but since each path has an equal length in either brick, the individual opacities are underdetermined.

opacity $m'_1 = \frac{1}{2}(m_1 + m_2)$ is completely overdetermined, whereas the difference in opacity $m'_2 = \frac{1}{2}(m_1 - m_2)$ is completely underdetermined. We want to perform this partitioning from an arbitrary equation $\mathbf{G}\mathbf{m} = \mathbf{d} \rightarrow \mathbf{G}'\mathbf{m}' = \mathbf{d}'$, where \mathbf{m}' is partitioned into an upper part \mathbf{m}'^o that is overdetermined and a lower part \mathbf{m}'^u that is underdetermined:

$$\begin{bmatrix} \mathbf{G}'^o & 0 \\ 0 & \mathbf{G}'^u \end{bmatrix} \begin{bmatrix} \mathbf{m}'^o \\ \mathbf{m}'^u \end{bmatrix} = \begin{bmatrix} \mathbf{d}'^o \\ \mathbf{d}'^u \end{bmatrix} \quad (3.33)$$

If this can be achieved, we could determine the overdetermined model parameters by solving the upper equations in the least squares sense and determine the underdetermined model parameters by finding those that have minimum L_2 solution length. In addition, we would have found a solution that added as little *a priori* information to the inverse problem as possible.

This partitioning process can be accomplished through singular-value decomposition of the data kernel, a process that we shall discuss in [Chapter 7](#). Since it is a relatively time-consuming process, we first examine an approximate process that works if the inverse problem is not too underdetermined.

Instead of partitioning \mathbf{m} , suppose that we determine a solution that minimizes some combination Φ of the prediction error and the solution length for the unpartitioned model parameters:

$$\Phi(\mathbf{m}) = E + \varepsilon^2 L = \mathbf{e}^T \mathbf{e} + \varepsilon^2 \mathbf{m}^T \mathbf{m} \quad (3.34)$$

where the weighting factor ε^2 determines the relative importance given to the prediction error and solution length. If ε is made large enough, this procedure will clearly minimize the underdetermined part of the solution. Unfortunately, it also tends to minimize the overdetermined part of the solution. As a result, the solution will not minimize the prediction error E and will not be a very good estimate of the true model parameters. If ε is set to zero, the prediction error will be minimized, but no *a priori* information will be provided to single out the underdetermined model parameters. It may be possible, however, to find some compromise value for ε that will approximately minimize E while approximately minimizing the length of the underdetermined part of the

solution. There is no simple method of determining what this compromise ε should be (without solving the partitioned problem); it must be determined by trial and error. By minimizing $\Phi(\mathbf{m})$ with respect to the model parameters in a manner exactly analogous to the least squares derivation, we obtain

$$[\mathbf{G}^T\mathbf{G} + \varepsilon^2\mathbf{I}]\mathbf{m}^{\text{est}} = \mathbf{G}^T\mathbf{d} \quad \text{or} \quad \mathbf{m}^{\text{est}} = [\mathbf{G}^T\mathbf{G} + \varepsilon^2\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{d} \quad (3.35)$$

This estimate of the model parameters is called the *damped least squares* solution. The concept of error has been generalized to include not only prediction error but also error in fitting the *a priori* information (that the solution length is zero, in this case). The underdeterminacy of the inverse problem is said to have been damped.

3.9 WEIGHTED MEASURES OF LENGTH AS A TYPE OF A *PRIORI* INFORMATION

There are many instances in which $L = \mathbf{m}^T\mathbf{m}$ is not a very good measure of solution simplicity. For instance, suppose that one were solving an inverse problem for density fluctuations in the ocean. One may not want to find a solution that is smallest in the sense of closest to zero but one that is smallest in the sense that it is closest to some other value, such as the average density of sea water. The obvious generalization of L is then

$$L = (\mathbf{m} - \langle \mathbf{m} \rangle)^T (\mathbf{m} - \langle \mathbf{m} \rangle) \quad (3.36)$$

where $\langle \mathbf{m} \rangle$ is the *a priori* value of the model parameters (a known typical value of sea water, in this case).

Sometimes the whole idea of length as a measure of simplicity is inappropriate. For instance, one may feel that a solution is simple if it is flat or if it is smooth. These measures may be particularly appropriate when the model parameters represent a discretized continuous function such as density or X-ray opacity. One may have the expectation that these parameters vary only slowly with position. Fortunately, properties such as *flatness* can be easily quantified by measures that are generalizations of length. For example, the flatness of a continuous function of space can be quantified by the norm of its first derivative, which is a measure of *steepness* (the opposite of flatness). For discrete model parameters, one can use the difference between physically adjacent model parameters as approximations of a derivative. The steepness \mathbf{l} of a vector \mathbf{m} is then

$$\mathbf{l} = \frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \ddots \\ & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_M \end{bmatrix} = \mathbf{D}\mathbf{m} \quad (3.37)$$

where \mathbf{D} is the *steepness matrix*. Other methods of simplicity can also be represented by a matrix multiplying the model parameters. For instance, solution smoothness can be implemented by quantifying the *roughness* (the opposite of smoothness) by the second derivative. The matrix multiplying the model parameters would then have rows containing $(\Delta x)^{-2}[\dots 1 - 2 \ 1 \dots]$. The overall steepness or roughness of the solution is then just the length

$$L = \mathbf{1}^T \mathbf{1} = [\mathbf{D}\mathbf{m}]^T [\mathbf{D}\mathbf{m}] = \mathbf{m}^T \mathbf{D}^T \mathbf{D} \mathbf{m} = \mathbf{m}^T \mathbf{W}_m \mathbf{m} \quad (3.38)$$

The matrix $\mathbf{W}_m = \mathbf{D}^T \mathbf{D}$ can be interpreted as a weighting factor that enters into the calculation of the length of the vector \mathbf{m} . Note, however, that $\|\mathbf{m}\|_{\text{weighted}}^2 = \mathbf{m}^T \mathbf{W}_m \mathbf{m}$ is *not* a proper norm, since it violates the positivity condition given in Equation (3.3a); that is, $\|\mathbf{m}\|_{\text{weighted}}^2 = 0$ for some nonzero vectors (such as the constant vector). This behavior usually poses no insurmountable problems, but it can cause solutions based on minimizing this norm to be nonunique.

The measure of solution simplicity can therefore be generalized to

$$L = [\mathbf{m} - \langle \mathbf{m} \rangle]^T \mathbf{W}_m [\mathbf{m} - \langle \mathbf{m} \rangle] \quad (3.39)$$

By suitably choosing the *a priori* model vector $\langle \mathbf{m} \rangle$ and the weighting matrix \mathbf{W}_m , we can quantify a wide variety of measures of simplicity.

Weighted measures of the prediction error can also be useful. Frequently some observations are made with more accuracy than others. In this case, one would like the prediction error e_i of the more accurate observations to have a greater weight in the quantification of the overall error E than the inaccurate observations. To accomplish this weighting, we define a generalized prediction error

$$E = \mathbf{e}^T \mathbf{W}_e \mathbf{e} \quad (3.40)$$

where the matrix \mathbf{W}_e defines the relative contribution of each individual error to the total prediction error. Normally we would choose this matrix to be diagonal. For example, if $N=5$ and the third observation is known to be twice as accurately determined as the others, one might use

$$\mathbf{W}_e = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \quad (3.41)$$

The inverse problem solutions stated above can then be modified to take into account these new measures of prediction error and solution simplicity. The derivations are substantially the same as for the unweighted cases but the algebra is lengthy.

3.9.1 Weighted Least Squares

If the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ is completely overdetermined, then one can estimate the model parameters by minimizing the generalized prediction error $E = \mathbf{e}^T \mathbf{W}_e \mathbf{e}$. This procedure leads to the solution

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{W}_e \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{W}_e \mathbf{d} \quad (3.42)$$

3.9.2 Weighted Minimum Length

If the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ is completely underdetermined, then one can estimate the model parameters by choosing the solution that is simplest, where simplicity is defined by the generalized length $L = [\mathbf{m} - \langle \mathbf{m} \rangle]^T \mathbf{W}_m [\mathbf{m} - \langle \mathbf{m} \rangle]^T$. This procedure leads to the solution

$$\mathbf{m}^{\text{est}} = \langle \mathbf{m} \rangle + \mathbf{W}_m^{-1} \mathbf{G}^T [\mathbf{G} \mathbf{W}_m^{-1} \mathbf{G}^T]^{-1} [\mathbf{d} - \mathbf{G} \langle \mathbf{m} \rangle] \quad (3.43)$$

3.9.3 Weighted Damped Least Squares

If the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ is slightly underdetermined, it can be solved by minimizing a combination of prediction error and solution length, $\Phi(\mathbf{m}) = E + \varepsilon^2 L$ (Franklin, 1970; Jackson, 1972; Jordan and Franklin, 1971). The parameter ε is chosen by trial and error to yield a solution that has a reasonably small prediction error. The equation for the solution, obtained by minimizing Φ with respect to \mathbf{m} , is then

$$[\mathbf{G}^T \mathbf{W}_e \mathbf{G} + \varepsilon^2 \mathbf{W}_m] \mathbf{m}^{\text{est}} = \mathbf{G}^T \mathbf{W}_e \mathbf{d} + \varepsilon^2 \mathbf{W}_m \langle \mathbf{m} \rangle \quad (3.44)$$

or

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{W}_e \mathbf{G} + \varepsilon^2 \mathbf{W}_m]^{-1} [\mathbf{G}^T \mathbf{W}_e \mathbf{d} + \varepsilon^2 \mathbf{W}_m \langle \mathbf{m} \rangle] \quad (3.45)$$

This equation appears to be rather complicated. However, it can be vastly simplified by noting that it is equivalent to solving the equation

$$\mathbf{F} \mathbf{m}^{\text{est}} = \mathbf{f} \quad \text{with} \quad \mathbf{F} = \begin{bmatrix} \mathbf{W}_e^{1/2} \mathbf{G} \\ \varepsilon \mathbf{D} \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} \mathbf{W}_e^{1/2} \mathbf{d} \\ \varepsilon \mathbf{D} \langle \mathbf{m} \rangle \end{bmatrix} \quad \text{and} \quad \mathbf{W}_m = \mathbf{D}^T \mathbf{D} \quad (3.46)$$

by simple least squares; that is, the equation $\mathbf{F}^T \mathbf{F} \mathbf{m}^{\text{est}} = \mathbf{F}^T \mathbf{f}$, when multiplied out, is identical to Equation (3.45). As explained previously, the weight matrix \mathbf{W}_e is typically diagonal. In that case, its square root, $\mathbf{W}_e^{1/2}$, is also diagonal with elements that are the square roots of the corresponding elements of \mathbf{W}_e .

Equation (3.46) has a very simple interpretation: its top row is the data equation $\mathbf{G}\mathbf{m}^{\text{est}} = \mathbf{d}$, with both sides multiplied by the weight matrix $\mathbf{W}_e^{1/2}$, and its bottom row is the *a priori* equation, $\mathbf{m}^{\text{est}} = \langle \mathbf{m} \rangle$, with both sides multiplied by the *a priori* matrix, $\varepsilon \mathbf{D}$. Note that the data and *a priori* information play completely symmetric roles in this equation.

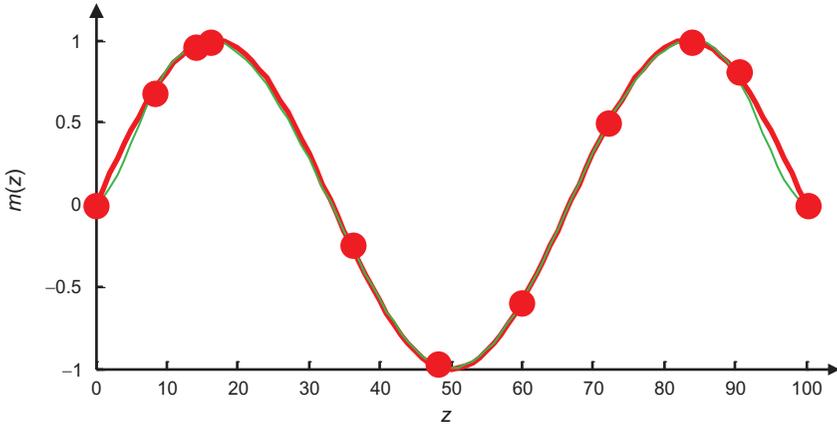


FIGURE 3.10 Example of weighted damped least squares. (Red curve) The true model, sampled with $\Delta z = 1$, is a sinusoid. (Red circles) The data are the model observed at just a few points. (Green curve) The estimated model is reconstructed from the data using the *a priori* information of smoothness in the interior of the (0, 100) interval and flatness at its ends. *MatLab* script gda03_09.

This equation is of the form

$$\begin{aligned} \Delta \mathbf{m} &= \mathbf{M} \Delta \mathbf{d} \\ \text{with } \Delta \mathbf{d} &= \mathbf{d} - \mathbf{G} \langle \mathbf{m} \rangle \quad \text{and} \quad \Delta \mathbf{m} = [\mathbf{m}^{\text{est}} - \langle \mathbf{m} \rangle] \\ \text{and } \mathbf{M} &= [\mathbf{G}^T \mathbf{W}_e \mathbf{G} + \varepsilon^2 \mathbf{W}_m]^{-1} \mathbf{G}^T \mathbf{W}_e \end{aligned} \quad (3.49)$$

This form emphasizes that the *deviation* $\Delta \mathbf{m}$ of the estimated solution from the *a priori* value is a linear function of the deviation $\Delta \mathbf{d}$ of the data from the value predicted by the *a priori* model.

Finally, we note that an alternative form of \mathbf{M} in Equation (3.49), reminiscent of the minimum-length solution, is

$$\mathbf{M} = \mathbf{W}_m^{-1} \mathbf{G}^T [\mathbf{G} \mathbf{W}_m^{-1} \mathbf{G}^T + \varepsilon^2 \mathbf{W}_e^{-1}]^{-1} \quad (3.50)$$

The equivalence can be demonstrated by equating the two forms of \mathbf{M} , premultiplying by $[\mathbf{G}^T \mathbf{W}_e \mathbf{G} + \varepsilon^2 \mathbf{W}_m]$ and postmultiplying by $[\mathbf{G} \mathbf{W}_m^{-1} \mathbf{G}^T + \varepsilon^2 \mathbf{W}_e^{-1}]$. In both instances, one must take care to ascertain whether the inverses actually exist. Depending on the choice of the weighting matrices, sufficient *a priori* information may or may not have been added to the problem to damp the underdeterminacy.

3.10 OTHER TYPES OF A *PRIORI* INFORMATION

One commonly encountered type of *a priori* information is the knowledge that some function of the model parameters equals a constant. Linear equality constraints of the form $\mathbf{H} \mathbf{m} = \mathbf{h}$ are particularly easy to implement. For example,

one such linear constraint requires that the mean of the model parameters must equal some value h_1 :

$$\mathbf{H}\mathbf{m} = \frac{1}{M} [1 \quad 1 \quad \cdots \quad 1] \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_M \end{bmatrix} = [h_1] = \mathbf{h} \quad (3.51)$$

Another such constraint requires that a particular model parameter, m_k , equals a given value

$$\mathbf{H}\mathbf{m} = [0 \quad \cdots \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0] \begin{bmatrix} m_1 \\ \vdots \\ m_k \\ \vdots \\ m_M \end{bmatrix} = [h_k] = \mathbf{h} \quad (3.52)$$

One problem that frequently arises is to solve an inverse problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ in the least squares sense with the *a priori* constraint that linear relationships between the model parameters of the form $\mathbf{H}\mathbf{m} = \mathbf{h}$ are satisfied exactly.

One way to implement this constraint is to use weighted damped least squares (Equation 3.46), with $\mathbf{D} = \mathbf{H}$ and $\mathbf{D}\langle\mathbf{m}\rangle = \mathbf{h}$, and with the weighting factor ε chosen to be very large, so that the *a priori* equations are given much more weight than the data equations (Lanczos, 1961). This method is well suited for computation, but it does require the value of ε to be chosen with some care—too big and the solution will suffer from numerical noise; too small and the constraints will be only very approximately satisfied.

Another method of implementing the constraints is through the use of Lagrange multipliers. One minimizes $E = \mathbf{e}^T \mathbf{e}$ with the constraint that $\mathbf{H}\mathbf{m} - \mathbf{h} = 0$ by forming the function

$$\Phi(m) = \sum_{i=1}^N \left[\sum_{j=1}^M G_{ij} m_j - d_i \right]^2 + 2 \sum_{i=1}^p \lambda_i \left[\sum_{j=1}^M H_{ij} m_j - h_i \right] \quad (3.53)$$

(where there are p constraints and $2\lambda_i$ are the Lagrange multipliers) and setting its derivatives with respect to the model parameters to zero as

$$\frac{\partial \Phi(\mathbf{m})}{\partial m_q} = 2 \sum_{i=1}^M m_i \sum_{j=1}^N G_{jq} G_{ji} - 2 \sum_{i=1}^N G_{iq} d_i + 2 \sum_{i=1}^p \lambda_i H_{iq} \quad (3.54)$$

This equation must be solved simultaneously with the constraint equations $\mathbf{H}\mathbf{m} = \mathbf{h}$ to yield the estimated solution. These equations, in matrix form, are

$$\begin{bmatrix} \mathbf{G}^T \mathbf{G} & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^T \mathbf{d} \\ \mathbf{h} \end{bmatrix} \quad (3.55)$$

Although these equations can be manipulated to yield an explicit formula for \mathbf{m}^{est} , it is often more convenient to solve directly this $M+p$ system of equations for M estimates of model parameters and p Lagrange multipliers by premultiplying by the inverse of the square matrix.

3.10.1 Example: Constrained Fitting of a Straight Line

Consider the problem of fitting the straight line $d_i = m_1 + m_2 z_i$ to data, where one has *a priori* information that the line must pass through the point (z', d') (Figure 3.11). There are two model parameters: intercept m_1 and slope m_2 . The $p=1$ constraint is that $d' = m_1 + m_2 z'$, or

$$\mathbf{Hm} = \begin{bmatrix} 1 & z' \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = [d'] = \mathbf{h} \quad (3.56)$$

Using the $\mathbf{G}^T \mathbf{G}$ and $\mathbf{G}^T \mathbf{d}$ computed in Section 3.5.1, the solution is

$$\begin{bmatrix} m_1^{\text{est}} \\ m_2^{\text{est}} \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} N & \sum_{i=1}^N z_i & 1 \\ \sum_{i=1}^N z_i & \sum_{i=1}^N z_i^2 & z' \\ 1 & z' & 0 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^N d_i \\ \sum_{i=1}^N z_i d_i \\ d' \end{bmatrix} \quad (3.57)$$

Another kind of *a priori* constraint is the *linear inequality constraint*, which we can write as $\mathbf{Hm} \geq \mathbf{h}$ (the inequality being interpreted component by component). Note that this form can also include \leq inequalities by multiplying the inequality relation by -1 . This kind of *a priori* constraint has application to problems in which the model parameters are inherently positive quantities,

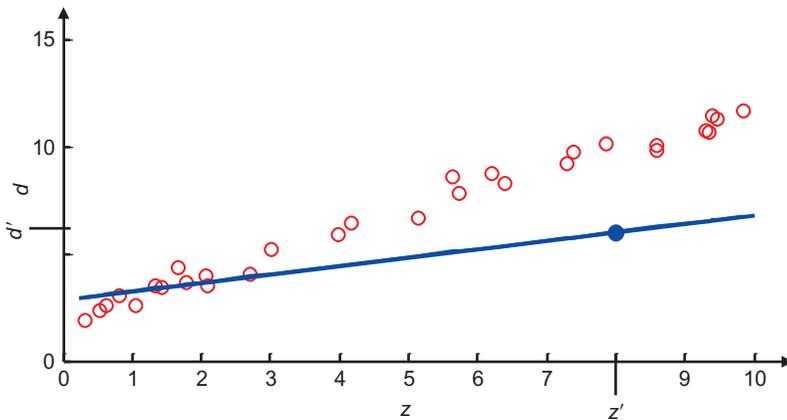


FIGURE 3.11 Least squares fitting of a straight line to (z, d) data, where the line is constrained to pass through the point $(z', d') = (8, 6)$. *MatLab* script gda03_10.

$m_i > 0$, and to other cases when the solution is known to possess some kind of bounds. One could therefore propose a new kind of constrained least squares solution of overdetermined problems, one that minimizes the error subject to the given inequality constraints. *A priori* inequality constraints also have application to underdetermined problems. One can find the smallest solution that solves both $\mathbf{Gm} = \mathbf{d}$ and $\mathbf{Hm} \geq \mathbf{h}$. These problems can be solved in a straightforward fashion, which will be discussed in Chapter 7.

3.11 THE VARIANCE OF THE MODEL PARAMETER ESTIMATES

The data invariably contain noise that causes errors in the estimates of the model parameters. We can calculate how this measurement error *maps* into errors in \mathbf{m}^{est} by noting that all of the formulas derived above for estimates of the model parameters are linear functions of the data, of the form $\mathbf{m}^{\text{est}} = \mathbf{M}\mathbf{d} + \mathbf{v}$, where \mathbf{M} is some matrix and \mathbf{v} some vector. Therefore, if we assume that the data have a distribution characterized by some covariance matrix $[\text{cov } \mathbf{d}]$, the estimates of the model parameters have a distribution characterized by a covariance matrix $[\text{cov } \mathbf{m}] = \mathbf{M}[\text{cov } \mathbf{d}]\mathbf{M}^T$. The covariance of the solution can therefore be calculated in a straightforward fashion. If the data are uncorrelated and of equal variance σ_d^2 , then very simple formulas are obtained for the covariance of some of the more simple inverse problem solutions.

The simple least squares solution $\mathbf{m}^{\text{est}} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{d}$ has covariance

$$[\text{cov } \mathbf{m}] = \left[[\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T \right] \sigma_d^2 \mathbf{I} \left[[\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T \right]^T = \sigma_d^2 [\mathbf{G}^T\mathbf{G}]^{-1} \quad (3.58)$$

and the simple minimum-length solution $\mathbf{m}^{\text{est}} = \mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1}\mathbf{d}$ has covariance

$$[\text{cov } \mathbf{m}] = \left[\mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1} \right] \sigma_d^2 \mathbf{I} \left[\mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1} \right]^T = \sigma_d^2 \mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-2}\mathbf{G} \quad (3.59)$$

An important issue is how to arrive at an estimate of the variance of the data σ_d^2 that can be used in these equations. One possibility is to base it upon knowledge about the inherent accuracy of the measurement process, in which case it is termed an *a priori variance*. For instance, if lengths are being measured with a ruler with 1 mm divisions, the estimate $\sigma_d \approx 1/2$ mm would be reasonable. Another possibility is to base the estimate upon the size distribution of prediction errors \mathbf{e} determined by fitting a model to the data, in which case it is termed an *a posteriori variance*. A reasonable estimate, whose theoretical justification will be discussed in Chapter 5, is

$$\sigma_d^2 \approx \frac{1}{N-M} \sum_{i=1}^N e_i^2 \quad (3.60)$$

This formula is essentially the mean-squared error $N^{-1} \sum_{i=1}^N e_i^2$, except that N has been replaced by $N-M$ to account for the ability of a model with M parameters

to exactly fit M data. A posteriori estimates are usually overestimates because inaccuracies in the model contribute to the size of the prediction error.

The least squares rule for error propagation, $[\text{cov } \mathbf{m}] = \sigma_d^2 [\mathbf{G}^T \mathbf{G}]^{-1}$, indicates that the model parameters can be correlated and can be of unequal variance even when the data are uncorrelated and are of equal variance. Whether observational error is attenuated or amplified by the inversion process is critically dependent upon the structure of the data kernel \mathbf{G} . In the problem for the mean of N data, discussed above, observational error is attenuated, but this desirable behavior is not common to all inverse problems (Figure 3.12).

3.12 VARIANCE AND PREDICTION ERROR OF THE LEAST SQUARES SOLUTION

If the prediction error $E(\mathbf{m}) = \mathbf{e}^T \mathbf{e}$ of an overdetermined problem has a very sharp minimum in the vicinity of the estimated solution \mathbf{m}^{est} , we would expect that the solution is well determined in the sense that it has small variance. Small errors in determining the shape of $E(\mathbf{m})$ due to random fluctuations in the data lead to only small errors in \mathbf{m}^{est} . Conversely, if $E(\mathbf{m})$ has a broad minimum, we expect that \mathbf{m}^{est} has a large variance. Since the curvature of a function is a measure of the sharpness of its minimum, we expect that the variance of the solution is related to the curvature of $E(\mathbf{m})$ at its minimum, which in turn depends on the structure of the data kernel \mathbf{G} (Figure 3.13).

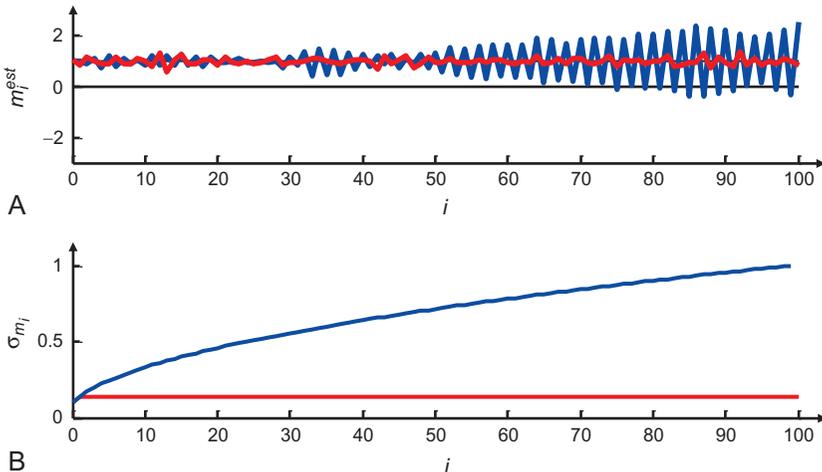


FIGURE 3.12 Two hypothetical experiments to measure the weight m_i of each of 100 bricks. In experiment 1 (red), the bricks are accumulated on a scale so that observation d_i is the sum of the weight of the first i bricks. In experiment 2 (blue), the first brick is weighed, and then subsequently, pairs or bricks (the first and the second, the second and the third, and so forth). (A) Least squares solution for weights m_i . (B) Corresponding error σ_{m_i} . Note that the first experiment has the lower error. *MatLab* script gda03_11.

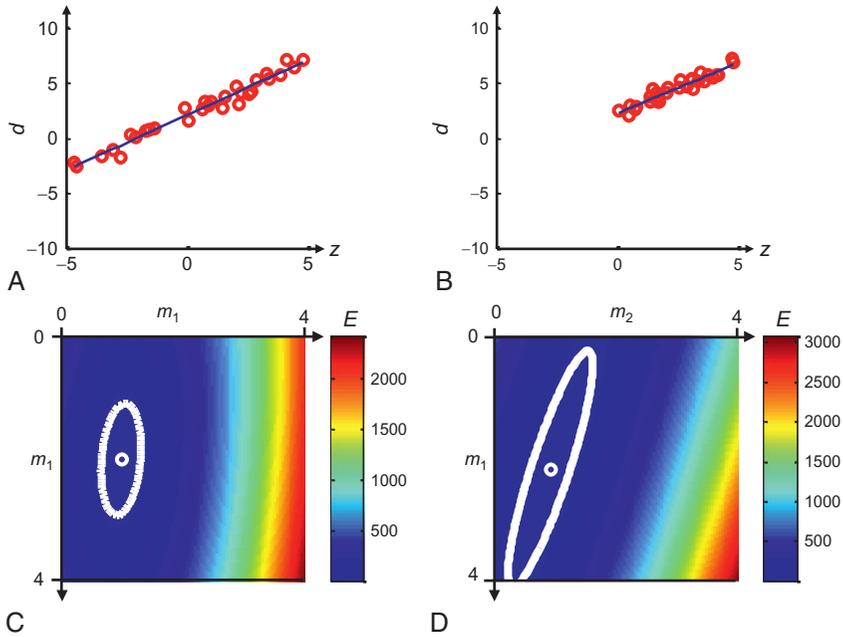


FIGURE 3.13 (A) Least squares fitting of a straight line (blue) to (z, d) data (red). (C) The best estimate of the model parameters $(m_1^{\text{est}}, m_2^{\text{est}})$ (white circle) occurs at the minimum of the error surface $E(m_1, m_2)$, which is a function of model parameters, intercept m_1 and slope m_2 . The minimum is surrounded by a region of low error (white ellipse) that corresponds to lines that fit “almost as well” as the best estimate. The variance of the estimate is related to the size of the ellipse. In this example, the ellipse is narrowest in the m_2 direction, indicating that the slope m_2 is determined more accurately than intercept m_1 . The geometry of the experiment, and not the overall level of observational error, determines the shape of the ellipse, as can be seen from the example in (B) and (D). The tilt of the ellipse indicates that the intercept and slope are negatively correlated. *MatLab* script `gda03_12`.

The curvature of the prediction error can be measured by its second derivative, as we can see by computing how small changes in the model parameters change the prediction error. Expanding the prediction error in a Taylor series about its minimum and keeping up to second-order terms give

$$\Delta E = E(\mathbf{m}) - E(\mathbf{m}^{\text{est}}) = [\mathbf{m} - \mathbf{m}^{\text{est}}]^T \left[\frac{1}{2} \frac{\partial^2 E}{\partial \mathbf{m}^2} \right]_{\mathbf{m}=\mathbf{m}^{\text{est}}} [\mathbf{m} - \mathbf{m}^{\text{est}}] \quad (3.61)$$

Here the matrix $\frac{\partial^2 E}{\partial \mathbf{m}^2}$ has elements $\frac{\partial^2 E}{\partial m_i \partial m_j}$. Note that the first-order term is zero, since the expansion is made at a minimum. The second derivative can also be computed directly from the expression

$$E(\mathbf{m}) = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 = \sum_{i=1}^N d_i^2 - 2 \sum_{i=1}^N d_i \sum_{j=1}^M G_{ij} m_j + \sum_{i=1}^N \sum_{j=1}^M G_{ij} m_j \sum_{k=1}^M G_{ik} m_k \quad (3.62)$$

which gives

$$\frac{\partial^2 E}{\partial m_p \partial m_q} = 2 \sum_{i=1}^N \sum_{j=1}^M G_{ij} \frac{\partial m_j}{\partial m_p} \sum_{k=1}^M G_{ik} \frac{\partial m_k}{\partial m_q} = 2 \sum_{i=1}^N G_{ip} G_{iq} \quad \text{or} \quad \left[\frac{1}{2} \frac{\partial^2 E}{\partial \mathbf{m}^2} \right] = \mathbf{G}^T \mathbf{G} \quad (3.63)$$

The covariance of the least squares solution (assuming uncorrelated data all with equal variance σ_d^2) is therefore

$$[\text{cov } \mathbf{m}] = \sigma_d^2 [\mathbf{G}^T \mathbf{G}]^{-1} = \sigma_d^2 \left[\frac{1}{2} \frac{\partial^2 E}{\partial \mathbf{m}^2} \right]_{\mathbf{m}=\mathbf{m}^{\text{est}}}^{-1} \quad (3.64)$$

The prediction error $E = \mathbf{e}^T \mathbf{e}$ is the sum of squares of Gaussian data minus a constant. It is, therefore, a random variable with a χ^2 distribution with $N - M$ degrees of freedom, which has mean $(N - M)\sigma_d^2$ and variance $2(N - M)\sigma_d^4$. (The degrees of freedom are reduced by M since the model can force M linear combinations of the e_i to zero.) We can use the standard deviation of E

$$\sigma_E = [2(N - M)]^{1/2} \sigma_d^2 \quad (3.65)$$

in the expression for variance as

$$[\text{cov } \mathbf{m}] = \sigma_d^2 [\mathbf{G}^T \mathbf{G}]^{-1} = \frac{\sigma_E}{[2(N - M)]^{1/2}} \left[\frac{1}{2} \frac{\partial^2 E}{\partial \mathbf{m}^2} \right]_{\mathbf{m}=\mathbf{m}^{\text{est}}}^{-1} \quad (3.66)$$

The covariance $[\text{cov } \mathbf{m}]$ can be interpreted as being controlled either by the variance of the data times a measure of how error in the data is mapped into error in the model parameters or by the standard deviation of the total prediction error times a measure of the curvature of the prediction error at its minimum.

The methods of solving inverse problems that have been discussed in this chapter emphasize the data and model parameters themselves. The method of least squares estimates the model parameters with smallest prediction length. The method of minimum-length estimates the simplest model parameters. The ideas of data and model parameters are very concrete and straightforward, and the methods based on them are simple and easily understood. Nevertheless, this viewpoint tends to obscure an important aspect of inverse problems: that the nature of the problems depends more on the *relationship* between the data and model parameters than on the data or model parameters themselves. It should, for instance, be possible to tell a well-designed experiment from a poor one without knowing what the numerical values of the data or model parameters are, or even the range in which they fall. In the next chapter, we will begin to explore this kind of problem.

3.13 PROBLEMS

- 3.1.** Show that the equations worked out for the straight line problem in Equation (3.5) have the solution given in Equation (3.17).
- 3.2.** This problem builds on Problem 1.1. Suppose that you determine the masses of 100 objects by weighing the first, then weighing the first and second together, and then weighing the rest in triplets: the first, second, and third; the second, third, and fourth; and so forth. Write a *MatLab* script that (A) randomly assigns masses m_i^{true} to the objects in the range of 0–1 kg; (B) builds the appropriate data kernel \mathbf{G} ; (C) creates synthetic observed data $\mathbf{d}^{\text{obs}} = \mathbf{G}\mathbf{m} + \mathbf{n}$, where \mathbf{n} is a vector of Gaussian random numbers with zero mean and $\sigma_d = 0.01$ kg; (D) solves the inverse problem by simple least squares; (E) estimates the variance of each of the estimated model parameters \mathbf{m}^{est} ; and (F) counts up the number of estimated model parameters that are within $\pm 2\sigma_m$ of their true value. (G) Make a plot of σ_m as a function of the index of the model parameter. Does it decline, remain constant, or grow?
- 3.3.** This problem builds on Problem 1.2. Suppose that you determine the height of 50 objects by measuring the first, and then stacking the second on top of the first and measuring their combined height, stacking the third on top of the first two and measuring their combined height, and so forth. Write a *MatLab* script that (A) randomly assigns heights m_i^{true} to the objects in the range of 0–1 m; (B) builds the appropriate data kernel \mathbf{G} ; (C) creates synthetic observed data $\mathbf{d}^{\text{obs}} = \mathbf{G}\mathbf{m} + \mathbf{n}$, where \mathbf{n} is a vector of Gaussian random numbers with zero mean and $\sigma_d = 0.01$ m; (D) solves the inverse problem by simple least squares; (E) estimates the variance of each of the estimated model parameters \mathbf{m}^{est} ; and (F) counts up the number of estimated model parameters that are within $\pm 2\sigma_m$ of their true value. (G) Make a plot of σ_m as a function of the index of the model parameter. Does it decline, remain constant, or grow?
- 3.4.** This problem builds on Problem 1.3, which considers the cubic equation, $d_i = m_1 + m_2 z_i + m_3 z_i^2 + m_4 z_i^3$. Write a *MatLab* script that (A) computes a vector \mathbf{z} with $N = 11$ elements equally spaced from 0 to 1; (B) randomly assigns the elements of \mathbf{m}^{true} in the range of -1 to 1 ; (C) builds the appropriate data kernel \mathbf{G} ; (D) creates synthetic observed data $\mathbf{d}^{\text{obs}} = \mathbf{G}\mathbf{m} + \mathbf{n}$, where \mathbf{n} is a vector of Gaussian random numbers with zero mean and $\sigma_d = 0.05$; (E) solves the inverse problem by simple least squares; (F) calculates the predicted data, $\mathbf{d}^{\text{pre}} = \mathbf{G}\mathbf{m}^{\text{est}}$; and (G) plots d_i^{obs} and d_i^{pre} . Comment upon the results.
- 3.5.** This problem builds on Problem 3.4. Modify your solution of Problem 3.4 by adding the constraint that the predicted data pass through the point $(z', d') = (5, 0)$. Comment upon the results.

REFERENCES

- Franklin, J.N., 1970. Well-posed stochastic extensions of ill-posed linear problems. *J. Math. Anal. Appl.* 31, 682–716.
- Jackson, D.D., 1972. Interpretation of inaccurate, insufficient and inconsistent data. *Geophys. J. R. Astron. Soc.* 28, 97–110.
- Jackson, D.D., 1979. The use of a priori data to resolve non-uniqueness in linear inversion. *Geophys. J. R. Astron. Soc.* 57, 137–157.
- Jordan, T.H., Franklin, J.N., 1971. Optimal solutions to a linear inverse problem in geophysics. *Proc. Natl. Acad. Sci. USA* 68, 291–293.
- Lanczos, C., 1961. *Linear Differential Operators*. Van Nostrand-Reinhold, Princeton, NJ.
- Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc, Oxford, UK 263pp.

Solution of the Linear, Gaussian Inverse Problem, Viewpoint 2: Generalized Inverses

4.1 SOLUTIONS VERSUS OPERATORS

In the previous chapter, we derived methods of solving the linear inverse problem $\mathbf{G}\mathbf{m}=\mathbf{d}$ that were based on examining two properties of its solution: prediction error and solution simplicity (or length). Most of these solutions had a form that was linear in the data, $\mathbf{m}^{\text{est}}=\mathbf{M}\mathbf{d}+\mathbf{v}$, where \mathbf{M} is some matrix and \mathbf{v} is some vector, both of which are independent of the data \mathbf{d} . This equation indicates that the estimate of the model parameters is controlled by some matrix \mathbf{M} operating on the data (that is, multiplying the data). We therefore shift our emphasis from the estimates \mathbf{m}^{est} to the operator matrix \mathbf{M} , with the expectation that by studying it we can learn more about the properties of inverse problems. Since the matrix \mathbf{M} solves, or “inverts,” the inverse problem $\mathbf{G}\mathbf{m}=\mathbf{d}$, it is often called the *generalized inverse* and given the symbol \mathbf{G}^{-g} . The exact form of the generalized inverse depends on the problem at hand. The generalized inverse of the overdetermined least squares problem is $\mathbf{G}^{-g}=[\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T$, and for the minimum length underdetermined solution it is $\mathbf{G}^{-g}=\mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1}$.

Note that in some ways the generalized inverse is analogous to the ordinary matrix inverse. The solution to the square (even-determined) matrix equation $\mathbf{A}\mathbf{x}=\mathbf{y}$ is $\mathbf{x}=\mathbf{A}^{-1}\mathbf{y}$, and the solution to the inverse problem $\mathbf{G}\mathbf{m}=\mathbf{d}$ is $\mathbf{m}^{\text{est}}=\mathbf{G}^{-g}\mathbf{d}$ (plus some vector, possibly). The analogy is very limited, however. The generalized inverse is not a matrix inverse in the usual sense. It is not square, and neither $\mathbf{G}^{-g}\mathbf{G}$ nor $\mathbf{G}\mathbf{G}^{-g}$ need equal an identity matrix.

4.2 THE DATA RESOLUTION MATRIX

Suppose we have found a generalized inverse that in some sense solves the inverse problem $\mathbf{G}\mathbf{m}=\mathbf{d}$, yielding an estimate of the model parameters $\mathbf{m}^{\text{est}}=\mathbf{G}^{-g}\mathbf{d}$ (for the sake of simplicity we assume that there is no additive vector). We can then retrospectively ask how well this estimate of the model

parameters fits the data. By plugging our estimate into the equation $\mathbf{Gm} = \mathbf{d}$ we conclude

$$\mathbf{d}^{\text{pre}} = \mathbf{Gm}^{\text{est}} = \mathbf{G}[\mathbf{G}^{-\text{g}}\mathbf{d}^{\text{obs}}] = [\mathbf{G}\mathbf{G}^{-\text{g}}]\mathbf{d}^{\text{obs}} = \mathbf{N}\mathbf{d}^{\text{obs}} \quad (4.1)$$

Here, the superscripts obs and pre mean observed and predicted, respectively. The $N \times N$ square matrix $\mathbf{N} = \mathbf{G}\mathbf{G}^{-\text{g}}$ is called the *data resolution matrix*. This matrix describes how well the predictions match the data. If $\mathbf{N} = \mathbf{I}$, then $\mathbf{d}^{\text{pre}} = \mathbf{d}^{\text{obs}}$ and the prediction error is zero. On the other hand, if the data resolution matrix is not an identity matrix, the prediction error is nonzero.

If the elements of the data vector \mathbf{d} possess a natural ordering, then the data resolution matrix has a simple interpretation. Consider, for example, the problem of fitting a curve to (z, d) points, where the data have been ordered according to the value of the auxiliary variable z . If \mathbf{N} is not an identity matrix but is close to an identity matrix (in the sense that its largest elements are near its main diagonal), then the configuration of the matrix signifies that averages of neighboring data can be predicted, whereas individual data cannot. Consider the i th row of \mathbf{N} . If this row contained all zeros except for a one in the i th column, then d_i would be predicted exactly. On the other hand, suppose that the row contained the elements

$$[\dots 0 \ 0 \ 0 \ 0.1 \ 0.8 \ 0.1 \ 0 \ 0 \ 0\dots] \quad (4.2)$$

where the 0.8 is in the i th column. Then the i th datum is given by

$$d_i^{\text{pre}} = \sum_{j=1}^N N_{ij}d_j^{\text{obs}} = 0.1d_{i-1}^{\text{obs}} + 0.8d_i^{\text{obs}} + 0.1d_{i+1}^{\text{obs}} \quad (4.3)$$

The predicted value is a weighted average of three neighboring observed data. If the true data vary slowly with the auxiliary variable, then such an average might produce an estimate reasonably close to the observed value.

The rows of the data resolution matrix \mathbf{N} describe how well neighboring data can be independently predicted, or *resolved*. If the data have a natural ordering, then a graph of the elements of the rows of \mathbf{N} against column indices illuminates the sharpness of the resolution (Figure 4.1A). If the graphs have a single sharp maximum centered about the main diagonal, then the data are well resolved. If the graphs are very broad, then the data are poorly resolved. Even in cases where there is no natural ordering of the data, the resolution matrix still shows how much weight each observation has in influencing the predicted value. There is then no special significance to whether large off-diagonal elements fall near to or far from the main diagonal.

A straight line has only two parameters and so cannot accurately product many independent data. Consequently, the data resolution matrix for the problem of fitting a straight line to data is not diagonal (Figure 4.1B). Its largest amplitudes are at its top-right and bottom-left corners, indicating that the points at the *ends* of the line are controlling the fit.

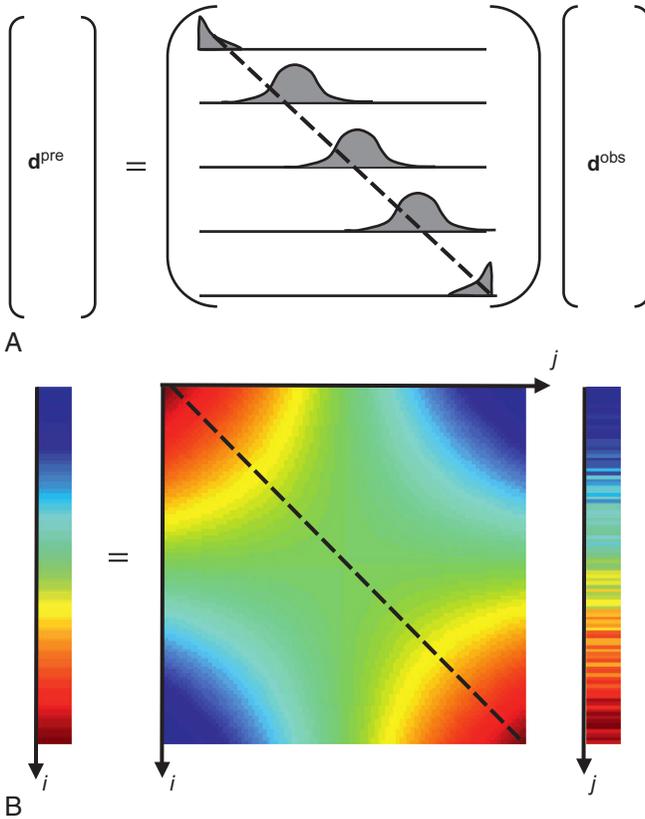


FIGURE 4.1 (A) Plots of selected rows of the data resolution matrix, \mathbf{N} , indicate how well the data can be predicted. Narrow peaks occurring near the main diagonal of the matrix (dashed line) indicate that the resolution is good. (B) Actual \mathbf{N} for the case of fitting a straight line to 100 data, equally spaced along the z -axis. Large values (red colors) occur only near the ends of the main diagonal (dashed line), indicating that the resolution is poor at intermediate values of z . *MatLab* script gda04_01.

Because the diagonal elements of the data resolution matrix indicate how much weight a datum has in its own prediction, these diagonal elements are often singled out and called the *importance* \mathbf{n} of the data (Minster et al., 1974)

$$\mathbf{n} = \text{diag}(\mathbf{N}) \tag{4.4}$$

The data resolution matrix is not a function of the data but only of the data kernel \mathbf{G} (which embodies the model and experimental geometry) and any *a priori* information applied to the problem. It can therefore be computed and studied without actually performing the experiment and can be a useful tool in experimental design.

4.3 THE MODEL RESOLUTION MATRIX

The data resolution matrix characterizes whether the data can be independently predicted, or resolved. The same question can be asked about the model parameters. To explore this question we imagine that there is a true, but unknown set of model parameters \mathbf{m}^{true} that solve $\mathbf{G}\mathbf{m}^{\text{true}} = \mathbf{d}^{\text{obs}}$. We then inquire how closely a particular estimate of the model parameters \mathbf{m}^{est} is to this true solution. Plugging the expression for the observed data $\mathbf{G}\mathbf{m}^{\text{true}} = \mathbf{d}^{\text{obs}}$ into the expression for the estimated model $\mathbf{m}^{\text{est}} = \mathbf{G}^{-\text{g}}\mathbf{d}^{\text{obs}}$ gives

$$\mathbf{m}^{\text{est}} = \mathbf{G}^{-\text{g}}\mathbf{d}^{\text{obs}} = \mathbf{G}^{-\text{g}}[\mathbf{G}\mathbf{m}^{\text{true}}] = [\mathbf{G}^{-\text{g}}\mathbf{G}]\mathbf{m}^{\text{true}} = \mathbf{R}\mathbf{m}^{\text{true}} \quad (4.5)$$

(Wiggins, 1972). Here \mathbf{R} is the $M \times M$ model resolution matrix. If $\mathbf{R} = \mathbf{I}$, then each model parameter is uniquely determined. If \mathbf{R} is not an identity matrix, then the estimates of the model parameters are really weighted averages of the true model parameters. If the model parameters have a natural ordering (as they would if they represented a discretized version of a continuous function), then plots of the rows of the resolution matrix can be useful in determining to what scale features in the model can actually be resolved (Figure 4.2A). Like the data resolution matrix, the model resolution is a function of only the data kernel and the *a priori* information added to the problem. It is therefore independent of the actual values of the data and can therefore be another important tool in experimental design.

As an example, we examine the resolution of the discrete version of the Laplace transform

$$d(c) = \int_0^{\infty} \exp(-cz)m(z)dz \rightarrow d_i = \sum_{j=1}^M \exp(-c_i z_j)m_j \quad (4.6)$$

Here, the datum d_i is a weighed average of the model parameters m_j , with weights that decline exponentially with depth z . The decay rate of the exponential is controlled by the constant, c_i , so that the smaller i s correspond to averages over a wider range of depths and the larger i s over a shallower range of depths. Not surprisingly, the shallow model parameters are better resolved (Figure 4.2B).

4.4 THE UNIT COVARIANCE MATRIX

The covariance of the model parameters depends on the covariance of the data and the way in which error is mapped from data to model parameters. This mapping is a function of only the data kernel and the generalized inverse, not of the data itself. A *unit covariance matrix* can be defined to characterize the degree of error amplification that occurs in the mapping. If the data are assumed to be uncorrelated and to have uniform variance σ^2 , the unit covariance matrix is given by

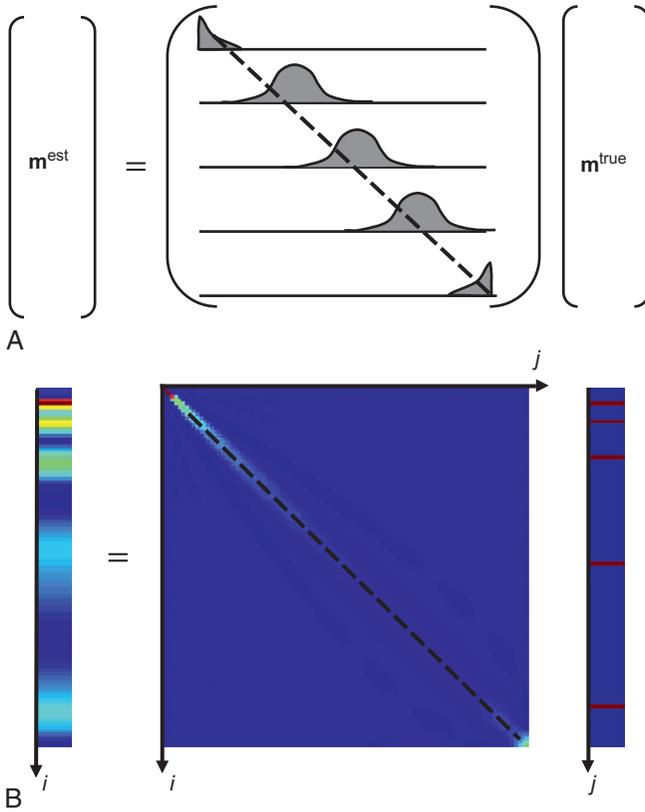


FIGURE 4.2 (A) Plots of selected rows of the model resolution matrix, \mathbf{R} , indicate how well the model parameters can be resolved. Narrow peaks occurring near the main diagonal of the matrix (dashed line) indicate that the resolution is good. (B) Actual \mathbf{R} for the case where the model parameters, $m_j(z)$, are related to the data through the kernel, $G_{ij} = \exp(-c_i z_j)$, where the c s are constants. Large values (red colors) occur only near the top (small z) of the main diagonal (dashed line), indicating that the resolution is poor at larger values of z . *MatLab* script gda04_02.

$$[\text{cov}_u \mathbf{m}] = \sigma^{-2} \mathbf{G}^{-g} [\text{cov} \mathbf{d}] \mathbf{G}^{-gT} = \mathbf{G}^{-g} \mathbf{G}^{-gT} \quad (4.7)$$

Even if the data are correlated, one can often find some normalization of the data covariance matrix, so that one can define a *unit data covariance matrix* $[\text{cov}_u \mathbf{d}]$, related to the model covariance matrix by

$$[\text{cov}_u \mathbf{m}] = \mathbf{G}^{-g} [\text{cov}_u \mathbf{d}] \mathbf{G}^{-gT} \quad (4.8)$$

The unit covariance matrix is a useful tool in experimental design, especially because it is independent of the actual values and variances of the data themselves.

As an example, reconsider the problem of fitting a straight line to (z, d) data. The unit covariance matrix for intercept m_1 and slope m_2 is given by

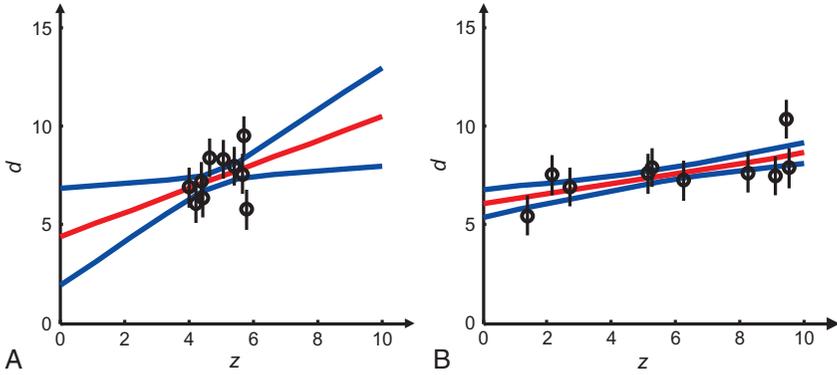


FIGURE 4.3 (A) The method of least squares is used to fit a straight line (red) to uncorrelated data (black circles) with uniform variance (vertical bars, 1σ confidence limits). Since the data are not well-separated in z , the variance of the slope and intercept is large, and consequently the variance of the predicted data is large as well (blue curves, 1σ confidence limits). (B) Same as (A) but with the data well-separated in z . Although the variance of the data is the same as in (A), the variance of the intercept and slope, and consequently the predicted data, is much smaller. *MatLab* script `gda04_03`.

$$[\text{cov}_u \mathbf{m}] = \frac{1}{N \sum z_i^2 - (\sum z_i)^2} \begin{bmatrix} \sum z_i^2 & -\sum z_i \\ -\sum z_i & N \end{bmatrix} \quad (4.9)$$

Note that the estimates of intercept and slope are uncorrelated only when the data are centered about $z=0$. The overall size of the variance is controlled by the denominator of the fraction. If all the z values are nearly equal, then the denominator of the fraction is small and the variance of the intercept and slope is large (Figure 4.3A). On the other hand, if the z values have a large spread, the denominator is large, and the variance is small (Figure 4.3B).

4.5 RESOLUTION AND COVARIANCE OF SOME GENERALIZED INVERSES

The data and model resolution and unit covariance matrices describe many interesting properties of the solutions to inverse problems. We therefore calculate these quantities for some of the simpler generalized inverses (with $[\text{cov}_u \mathbf{d}] = \mathbf{I}$).

4.5.1 Least Squares

$$\begin{aligned} \mathbf{G}^{-g} &= [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \\ \mathbf{N} &= \mathbf{G} \mathbf{G}^{-g} = \mathbf{G} [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \\ \mathbf{R} &= \mathbf{G}^{-g} \mathbf{G} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{G} = \mathbf{I} \\ [\text{cov}_u \mathbf{m}] &= \mathbf{G}^{-g} \mathbf{G}^{-gT} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{G} [\mathbf{G}^T \mathbf{G}]^{-1} = [\mathbf{G}^T \mathbf{G}]^{-1} \end{aligned} \quad (4.10)$$

4.5.2 Minimum Length

$$\begin{aligned}
 \mathbf{G}^{-g} &= \mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1} \\
 \mathbf{N} &= \mathbf{G}\mathbf{G}^{-g} = \mathbf{G}\mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1} = \mathbf{I} \\
 \mathbf{R} &= \mathbf{G}^{-g}\mathbf{G} = \mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1}\mathbf{G} \\
 [\text{cov}_u \mathbf{m}] &= \mathbf{G}^{-g}\mathbf{G}^{-gT} = \mathbf{G}^T[\mathbf{G}\mathbf{G}^T]^{-1}[\mathbf{G}\mathbf{G}^T]^{-1}\mathbf{G} = \mathbf{G}^T[\mathbf{G}^T\mathbf{G}]^{-2}\mathbf{G} \quad (4.11)
 \end{aligned}$$

Note that there is a great deal of symmetry between the least squares and minimum length solutions. Least squares solves the completely overdetermined problem and has perfect model resolution; minimum length solves the completely underdetermined problem and has perfect data resolution. As we shall see later, generalized inverses that solve the intermediate mixed-determined problems will have data and model resolution matrices that are intermediate between these two extremes.

4.6 MEASURES OF GOODNESS OF RESOLUTION AND COVARIANCE

Just as we were able to quantify the goodness of the model parameters by measuring their overall prediction error and simplicity, we shall develop techniques that quantify the goodness of data and model resolution matrices and unit covariance matrices. Because the resolution is best when the resolution matrices are identity matrices, one possible measure of resolution is based on the size, or *spread*, of the off-diagonal elements.

$$\begin{aligned}
 \text{spread}(\mathbf{N}) &= \|\mathbf{N} - \mathbf{I}\|_2^2 = \sum_{i=1}^N \sum_{j=1}^N [N_{ij} - \delta_{ij}]^2 \\
 \text{spread}(\mathbf{R}) &= \|\mathbf{R} - \mathbf{I}\|_2^2 = \sum_{i=1}^M \sum_{j=1}^M [R_{ij} - \delta_{ij}]^2
 \end{aligned} \quad (4.12)$$

Here δ_{ij} are the elements of the identity matrix \mathbf{I} . These measures of the goodness of the resolution spread are based on the L_2 norm of the difference between the resolution matrix and an identity matrix. They are sometimes called the *Dirichlet spread functions*. When $\mathbf{R} = \mathbf{I}$, $\text{spread}(\mathbf{R}) = 0$.

Since the unit standard deviation of the model parameters is a measure of the amount of error amplification mapped from data to model parameters, this quantity can be used to estimate the size of the unit covariance matrix as

$$\text{size}([\text{cov}_u \mathbf{m}]) = \left\| [\text{var}_u \mathbf{m}]^{1/2} \right\|_2^2 = \sum_{i=1}^M [\text{cov}_u \mathbf{m}]_{ii} \quad (4.13)$$

where the square root is interpreted element by element. Note that this measure of covariance size does not take into account the size of the off-diagonal elements in the unit covariance matrix.

4.7 GENERALIZED INVERSES WITH GOOD RESOLUTION AND COVARIANCE

Having found a way to measure quantitatively the goodness of the resolution and covariance of a generalized inverse, we now consider whether it is possible to use these measures as guiding principles for deriving generalized inverses. This procedure is analogous to that of [Chapter 3](#), which involves first defining measures of solution prediction error and simplicity and then using those measures to derive the least squares and minimum length estimates of the model parameters.

4.7.1 Overdetermined Case

We first consider a purely overdetermined problem of the form $\mathbf{G}\mathbf{m}=\mathbf{d}$. We postulate that this problem has a solution of the form $\mathbf{m}^{\text{est}}=\mathbf{G}^{-g}\mathbf{d}$ and try to determine \mathbf{G}^{-g} by minimizing some combination of the above measures of goodness. Since we previously noted that the overdetermined least squares solution had perfect model resolution, we shall try to determine \mathbf{G}^{-g} by minimizing only the spread of the data resolution. We begin by examining the spread of the k th row of \mathbf{N} , say, J_k :

$$J_k = \sum_{i=1}^N (N_{ki} - \delta_{ki})^2 = \sum_{i=1}^N N_{ki}^2 - 2 \sum_{i=1}^N N_{ki} \delta_{ki} + \sum_{i=1}^N \delta_{ki}^2 \quad (4.14)$$

Since each of the J_k s is positive, we can minimize the total spread $(\mathbf{N}) = \sum J_k$ by minimizing each individual J_k . We therefore insert the definition of the data resolution matrix $\mathbf{N}=\mathbf{G}\mathbf{G}^{-g}$ into the formula for J_k and minimize it with respect to the elements of the generalized inverse matrix:

$$\frac{\partial J_k}{\partial G_{qr}^{-g}} = 0 \quad (4.15)$$

We shall perform the differentiation separately for each of the three terms of J_k . The first term is given by

$$\begin{aligned} \frac{\partial}{\partial G_{qr}^{-g}} \left[\sum_{i=1}^N \left[\sum_{j=1}^M G_{kj} G_{ji}^{-g} \right] \left[\sum_{p=1}^M G_{kp} G_{pi}^{-g} \right] \right] &= \frac{\partial}{\partial G_{qr}^{-g}} \left[\sum_{i=1}^N \sum_{j=1}^M \sum_{p=1}^M G_{ji}^{-g} G_{pi}^{-g} G_{kj} G_{kp} \right] \\ &= 2 \sum_{i=1}^N \sum_{j=1}^M \sum_{p=1}^M \delta_{jq} \delta_{ir} G_{pi}^{-g} G_{kj} G_{kp} \\ &= 2 \sum_{p=1}^M G_{pr}^{-g} G_{kq} G_{kp} \end{aligned} \quad (4.16)$$

The second term is given by

$$-2 \frac{\partial}{\partial G_{qr}^{-g}} \sum_{i=1}^N \sum_{j=1}^M G_{kj} G_{ji}^{-g} \delta_{ki} = -2 \sum_{i=1}^N \sum_{j=1}^M G_{kj} \delta_{jq} \delta_{ir} \delta_{ki} = -2 G_{kq} \delta_{kr} \quad (4.17)$$

The third term is zero, since it is not a function of the generalized inverse. The complete equation is $\sum_{p=1}^M G_{kq} G_{kp} G_{pr}^{-g} = G_{kq} \delta_{kr}$. After summing over k and converting to matrix notation, we obtain

$$\mathbf{G}^T \mathbf{G} \mathbf{G}^{-g} = \mathbf{G}^T \quad (4.18)$$

Since $\mathbf{G}^T \mathbf{G}$ is square, we can premultiply by its inverse to solve for the generalized inverse, $\mathbf{G}^{-g} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T$, which is precisely the same as the formula for the least squares generalized inverse. The least squares generalized inverse can be interpreted either as the inverse that minimizes the L_2 norm of the prediction error or as the inverse that minimizes the Dirichlet spread of the data resolution.

4.7.2 Underdetermined Case

The data can be satisfied exactly in a purely underdetermined problem. The data resolution matrix is, therefore, precisely an identity matrix and its spread is zero. We might therefore try to derive a generalized inverse for this problem by minimizing the spread of the model resolution matrix with respect to the elements of the generalized inverse. It is perhaps not particularly surprising that the generalized inverse obtained by this method is exactly the minimum length generalized inverse $\mathbf{G}^{-g} = \mathbf{G}^T [\mathbf{G} \mathbf{G}^T]^{-1}$. The minimum length solution can be interpreted either as the inverse that minimizes the L_2 norm of the solution length or as the inverse that minimizes the Dirichlet spread of the model resolution. This is another aspect of the symmetrical relationship between the least squares and minimum length solutions.

4.7.3 The General Case with Dirichlet Spread Functions

We seek the generalized inverse \mathbf{G}^{-g} that minimizes the weighted sum of Dirichlet measures of resolution spread and covariance size.

$$\text{Minimize: } \alpha_1 \text{ spread}(\mathbf{N}) + \alpha_2 \text{ spread}(\mathbf{R}) + \alpha_3 \text{ size}([\text{cov}_u \mathbf{m}]) \quad (4.19)$$

where the α s are arbitrary weighting factors. This problem is done in exactly the same fashion as the one in [Section 4.7.1](#), except that there is now three times as much algebra. The result is an equation for the generalized inverse:

$$\alpha_1 [\mathbf{G}^T \mathbf{G}] \mathbf{G}^{-g} + \mathbf{G}^{-g} [\alpha_2 [\mathbf{G} \mathbf{G}^T] + \alpha_3 [\text{cov}_u \mathbf{d}]] = [\alpha_1 + \alpha_2] \mathbf{G}^T \quad (4.20)$$

An equation of this form is called a *Sylvester equation*. It is just a set of linear equations in the elements of the generalized inverse \mathbf{G}^{-g} and so could be solved by writing the elements of \mathbf{G}^{-g} as a vector in a huge $NM \times NM$ matrix equation, but it has no explicit solution in terms of algebraic functions of the component matrices. Explicit solutions can be written, however, for a variety of

special choices of the weighting factors. The least squares solution is recovered if $\alpha_1 = 1$ and $\alpha_2 = \alpha_3 = 0$, and the minimum length solution is recovered if $\alpha_1 = 0$, $\alpha_2 = 1$, and $\alpha_3 = 0$. Of more interest is the case in which $\alpha_1 = 1$, $\alpha_2 = 0$, α_3 equals some constant (say, ε^2) and $[\text{cov}_u \mathbf{d}] = \mathbf{I}$. The generalized inverse is then given by

$$\mathbf{G}^{-g} = [\mathbf{G}^T \mathbf{G} + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{G}^T \quad (4.21)$$

This formula is precisely the damped least squares inverse, which we derived in the previous chapter by minimizing a combination of prediction error and solution length. The damped least squares solution can also be interpreted as the inverse that minimizes a weighted combination of data resolution spread and covariance size.

Another interesting solution is obtained when a weighted combination of model resolution spread and covariance size is minimized. Setting $\alpha_1 = 0$, $\alpha_2 = 1$, $\alpha_3 = \varepsilon^2$, and $[\text{cov}_u \mathbf{d}] = \mathbf{I}$, we find

$$\mathbf{G}^{-g} = \mathbf{G}^T [\mathbf{G} \mathbf{G}^T + \varepsilon^2 \mathbf{I}]^{-1} \quad (4.22)$$

This solution might be termed *damped minimum length*. It will be important in the discussion later in this chapter, because it is the Dirichlet analog to the Backus-Gilbert generalized inverse that will be introduced there.

Note that it is quite possible for these generalized inverses to possess resolution matrices containing *negative* off-diagonal elements. Physically, an average makes most sense when it contained only positive weighting factors, so negative elements interfere with the interpretation of the rows of \mathbf{R} as localized averages. In principle, it is possible to include non-negativity as a constraint when choosing the generalized inverse by minimizing the spread functions. However, in practice, this constraint is never implemented because it makes the calculation of the generalized inverse very difficult. Furthermore, the more constraints that one places on \mathbf{R} , the less localized it tends to become.

4.8 SIDELOBES AND THE BACKUS-GILBERT SPREAD FUNCTION

The Dirichlet spread function is not a particularly appropriate measure of the goodness of resolution when the data or model parameters have a natural ordering because the off-diagonal elements of the resolution matrix are all weighted equally, regardless of whether they are close or far from the main diagonal. We would much prefer that any large elements be close to the main diagonal when there is a natural ordering (Figure 4.4) because the rows of the resolution matrix then represent *localized* averaging functions.

If one uses the Dirichlet spread function to compute a generalized inverse, it will often have *sidelobes*, that is, large amplitude regions in the resolution matrices far from the main diagonal. We would prefer to find a generalized inverse

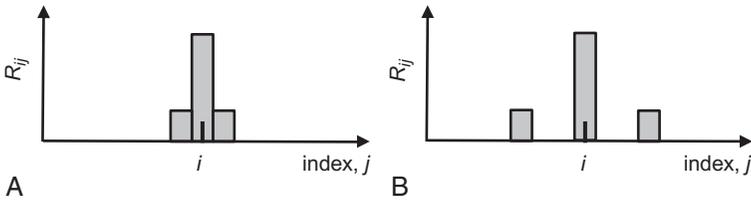


FIGURE 4.4 (A, B) Resolution matrices have the same spread when measured by the Dirichlet spread function. Nevertheless, if the model parameters possess a natural ordering, then (A) is better resolved. The Backus-Gilbert spread function is designed to measure (A) as having a smaller spread than (B).

without sidelobes, even at the expense of widening the band of nonzero elements near the main diagonal, since a solution with such a resolution matrix is then interpretable as a localized average of physically adjacent model parameters.

We therefore add a weighting factor $w(i, j)$ to the measure of spread that weights the (i, j) element of \mathbf{R} according to its physical distance from the diagonal element. This weighting preferentially selects resolution matrices that are “spiky,” or “deltalike.” If the natural ordering were a simple linear one, then the choice $w(i, j) = (i - j)^2$ would be reasonable. If the ordering is multidimensional, a more complicated weighting factor is needed. It is usually convenient to choose the spread function so that the diagonal elements have no weight, i.e., $w(i, i) = 0$, and so that $w(i, j)$ is always non-negative and symmetric in i and j . The new spread function, often called the Backus-Gilbert spread function (Backus and Gilbert, 1967, 1968), is then given by

$$\text{spread}(\mathbf{R}) = \sum_{i=1}^M \sum_{j=1}^M w(i, j) [R_{ij} - \delta_{ij}]^2 = \sum_{i=1}^M \sum_{j=1}^M w(i, j) R_{ij}^2 \quad (4.23)$$

A similar expression holds for the spread of the data resolution. One can now use this measure of spread to derive new generalized inverses. Their sidelobes will be smaller than those based on the Dirichlet spread functions. On the other hand, they are sometimes worse when judged by other criteria. As we shall see, the Backus-Gilbert generalized inverse for the completely underdetermined problem does not exactly satisfy the data, even though the analogous minimum length generalized inverse does. These facts demonstrate that there are unavoidable trade-offs inherent in finding solutions to inverse problems.

4.9 THE BACKUS-GILBERT GENERALIZED INVERSE FOR THE UNDERDETERMINED PROBLEM

This problem is analogous to deriving the minimum length solution by minimizing the Dirichlet spread of model resolution. Since it is very easy to satisfy the data when the problem is underdetermined (so that the data resolution has small

spread), we shall find a generalized inverse that minimizes the spread of the model resolution alone.

We seek the generalized inverse \mathbf{G}^{-g} that minimizes the Backus-Gilbert spread of model resolution. Since the diagonal elements of the model resolution matrix are given no weight, we also require that the resulting model resolution matrix satisfy the equation

$$\sum_{j=1}^M R_{ij} = [1]_i \quad (4.24)$$

This constraint ensures that the diagonal of the resolution matrix is finite and that the rows are unit averaging functions acting on the true model parameters. Writing the spread of one row of the resolution matrix as J_k and inserting the expression for the resolution matrix, we have

$$\begin{aligned} J_k &= \sum_{l=1}^M w(l, k) R_{kl} R_{kl} \\ &= \sum_{l=1}^M w(l, k) \left[\sum_{i=1}^N G_{ki}^{-g} G_{il} \right] \left[\sum_{j=1}^N G_{kj}^{-g} G_{jl} \right] \\ &= \sum_{i=1}^N \sum_{j=1}^N G_{ki}^{-g} G_{kj}^{-g} \sum_{l=1}^M w(l, k) G_{il} G_{jl} \\ &= \sum_{i=1}^N \sum_{j=1}^N G_{ki}^{-g} G_{kj}^{-g} S_{ij}^{(k)} \end{aligned} \quad (4.25)$$

where the quantity $S_{ij}^{(k)}$ is defined as

$$S_{ij}^{(k)} = \sum_{l=1}^M w(l, k) G_{il} G_{jl} \quad (4.26)$$

The left-hand side of the constraint equation $\sum_j R_{ij} = [1]_i$ can also be written in terms of the generalized inverse

$$\sum_{k=1}^M R_{ik} = \sum_{k=1}^M \left[\sum_{j=1}^N G_{ij}^{-g} G_{jk} \right] = \sum_{j=1}^N G_{ij}^{-g} \sum_{k=1}^M G_{jk} = \sum_{j=1}^N G_{ij}^{-g} u_j \quad (4.27)$$

Here the quantity u_j is defined as

$$u_j = \sum_{k=1}^M G_{jk} \quad (4.28)$$

The problem of minimizing J_k with respect to the elements of the generalized inverse (under the given constraints) can be solved through the use of Lagrange multipliers. We first define a Lagrange function Φ such that

$$\Phi = \sum_{i=1}^N \sum_{j=1}^M G_{ki}^{-g} G_{kj}^{-g} S_{ij}^{(k)} + 2\lambda \sum_{j=1}^N G_{kj}^{-g} u_j \quad (4.29)$$

where 2λ is the Lagrange multiplier. We then differentiate Φ with respect to the elements of the generalized inverse and set the result equal to zero as

$$\frac{\partial \Phi}{\partial G_{kp}^{-g}} = 2 \sum_{i=1}^N S_{pi}^{(k)} G_{ki}^{-g} + 2\lambda u_p = 0 \quad (4.30)$$

(Note that one can solve for each row of the generalized inverse separately, so that it is only necessary to take derivatives with respect to the elements in the k th row.) The above equation must be solved along with the original constraint equation. Treating the k th row of \mathbf{G}^{-g} as the transform of a column-vector $\mathbf{g}^{(k)}$ and the quantity $S_{ij}^{(k)}$ as a matrix $\mathbf{S}^{(k)}$, we can write these equations as the matrix equation

$$\begin{bmatrix} \mathbf{S}^{(k)} & \mathbf{u} \\ \mathbf{u}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{g}^{(k)} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \quad (4.31)$$

This is a square $(N+1) \times (N+1)$ system of linear equations that must be solved for the N elements of the k th row of the generalized inverse and for the one Lagrange multiplier λ .

The matrix equation can be solved explicitly using a variant of the *bordering method* of linear algebra, which is used to construct the inverse of a matrix by partitioning it into submatrices with simple properties. Suppose that the inverse of the symmetric matrix in Equation (4.31) exists and that we partition it into an $N \times N$ symmetric square matrix \mathbf{A} , vector \mathbf{b} , and scalar c . By assumption, pre-multiplication by the inverse yields the identity matrix

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} \begin{bmatrix} \mathbf{S}^{(k)} & \mathbf{u} \\ \mathbf{u}^T & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{S}^{(k)} + \mathbf{b}\mathbf{u}^T & \mathbf{A}\mathbf{u} \\ \mathbf{b}^T\mathbf{S}^{(k)} + c\mathbf{u}^T & \mathbf{b}^T\mathbf{u} \end{bmatrix} \quad (4.32)$$

The unknown submatrices \mathbf{A} , \mathbf{b} , and c can now be determined by equating the submatrices

$$\begin{aligned} \mathbf{A}\mathbf{S}^{(k)} + \mathbf{b}\mathbf{u}^T &= \mathbf{I} \quad \text{so that } \mathbf{A} = \left[\mathbf{S}^{(k)}\right]^{-1} [\mathbf{I} - \mathbf{b}\mathbf{u}^T] \\ \mathbf{A}\mathbf{u} &= \mathbf{0} \quad \text{so that } \left[\mathbf{S}^{(k)}\right]^{-1} \mathbf{u} = \mathbf{b}\mathbf{u}^T\mathbf{S}^{(k)}\mathbf{u} \quad \text{and} \quad \mathbf{b} = \frac{\left[\mathbf{S}^{(k)}\right]^{-1} \mathbf{u}}{\mathbf{u}^T \left[\mathbf{S}^{(k)}\right]^{-1} \mathbf{u}} \\ \mathbf{b}^T\mathbf{S}^{(k)} + c\mathbf{u}^T &= 0 \quad \text{so that } c = \frac{-1}{\mathbf{u}^T \left[\mathbf{S}^{(k)}\right]^{-1} \mathbf{u}} \end{aligned} \quad (4.33)$$

Multiplying Equation (4.31) by the inverse matrix yields $\mathbf{g}^{(k)} = \mathbf{b}$ and $\lambda = c$. The generalized inverse, written with summations, is

$$G_{kl}^{-g} = \frac{\sum_{i=1}^N u_i \left[(\mathbf{S}^{(k)})^{-1} \right]_{il}}{\sum_{i=1}^N \sum_{j=1}^N u_i \left[(\mathbf{S}^{(k)})^{-1} \right]_{ij} u_j} \quad (4.34)$$

This generalized inverse is the Backus-Gilbert analog to the minimum length solution.

As an example, we compare the Dirichlet and Backus-Gilbert solutions for the Laplace transform problem discussed in Section 4.3 (Figure 4.5). The Backus-Gilbert solution is the smoother of the two and has a corresponding model resolution matrix that consists of a single band along the main diagonal. The Dirichlet solution has more details but also more artifacts (such as negative values at $z \approx 3$). They are associated with the large-amplitude sidelobes in the corresponding model resolution matrix.

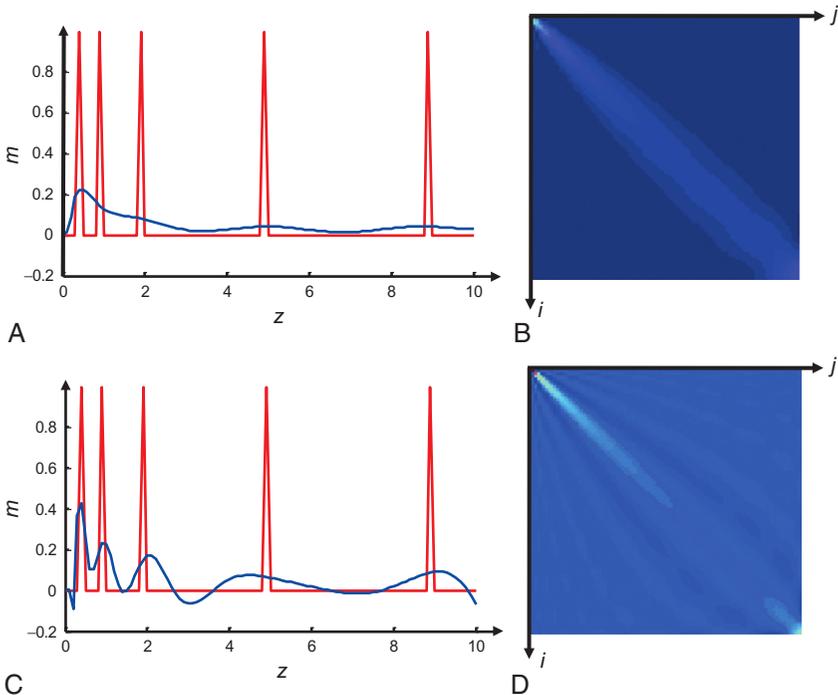


FIGURE 4.5 Comparison of the Backus-Gilbert and Dirichlet solutions of the inverse problem described in Figure 4.2. (A) The true model (red) contains a series of sharp spikes. The estimated model (blue) using the Backus-Gilbert spread function is much smoother, with the width of the smoothing increasing with z . (B) Corresponding model resolution matrix, \mathbf{R} . (C, D) Same, but for a Dirichlet spread function. Note that the Backus-Gilbert resolution matrix has the lower intensity sidelobes, but a wider central band. *MatLab* scripts gda04_02 and gda04_04.

4.10 INCLUDING THE COVARIANCE SIZE

The measure of goodness that was used to determine the Backus-Gilbert inverse can be modified to include a measure of the covariance size of the model parameters (Backus and Gilbert, 1970). We shall use the same measure as we did when considering the Dirichlet spread functions, so that goodness is measured by

$$\alpha \text{spread}(\mathbf{R}) + (1 - \alpha) \text{size}([\text{cov}_u \mathbf{m}]) = \alpha \sum_{i=1}^M \sum_{j=1}^M w(i, j) R_{ij}^2 + (1 - \alpha) \sum_{i=1}^M [\text{cov}_u \mathbf{m}]_{ii} \quad (4.35)$$

where $0 \leq \alpha \leq 1$ is a weighting factor that determines the relative contribution of model resolution and covariance to the measure of the goodness of the generalized inverse. The goodness J'_k of the k th row is then

$$\begin{aligned} J'_k &= \alpha \sum_{l=1}^M w(k, l) R_{kl}^2 + (1 - \alpha) [\text{cov}_u \mathbf{m}]_{kk} \\ &= \alpha \sum_{i=1}^N \sum_{j=1}^N G_{ki}^{-g} G_{kj}^{-g} [S_{ij}]_k + (1 - \alpha) \sum_{i=1}^N \sum_{j=1}^N G_{ki}^{-g} G_{kj}^{-g} [\text{cov}_u \mathbf{d}]_{ij} \\ &= \sum_{i=1}^N \sum_{j=1}^N G_{ki}^{-g} G_{kj}^{-g} S'_{ij}{}^{(k)} \end{aligned} \quad (4.36)$$

where the quantity $S'_{ij}{}^{(k)}$ is defined by the equation

$$S'_{ij}{}^{(k)} = \alpha S_{ij}^{(k)} + (1 - \alpha) [\text{cov}_u \mathbf{d}]_{ij} \quad (4.37)$$

Since the function J'_k has exactly the same form as J_k had in the previous section, the generalized inverse is just the previous result with $S_{ij}^{(k)}$ replaced by $S'_{ij}{}^{(k)}$:

$$G_{kl}^{-g} = \frac{\sum_{i=1}^N u_i \left[\left(\mathbf{S}'^{(k)} \right)^{-1} \right]_{il}}{\sum_{i=1}^N \sum_{j=1}^N u_i \left[\left(\mathbf{S}'^{(k)} \right)^{-1} \right]_{ij} u_j} \quad (4.38)$$

This generalized inverse is the Backus-Gilbert analog to the damped minimum length solution. In *MatLab*, the one-dimensional Backus-Gilbert generalized inverse GMG (that is, for the $w(i, j) = (i - j)^2$ weight function) is calculated as

```
GMG = zeros(M, N);
u = G*ones(M, 1);
for k = [1:M]
    S = G * diag([1:M]-k, 2) * G';
    Sp = alpha*S + (1-alpha)*eye(N, N);
    uSpinV = u' / Sp;
    GMG(k, :) = uSpinV / (uSpinV*u);
end
```

(*MatLab* script gda04_05)

In higher dimensions, the definition of S is more complicated, since the weight function must represent the physical distance between model parameters. In two dimensions, a reasonable choice is

$$S = G * \text{diag} ((\text{abs}(\text{ixofj}([1:M]) - \text{ixofj}(k))) .^2 + \dots \\ (\text{abs}(\text{iyofj}([1:M]) - \text{iyofj}(k))) .^2) * G' ;$$

Here the index vectors $\text{ixofj}(k)$ and $\text{iyofj}(k)$ give the x and y values of model parameter k .

4.11 THE TRADE-OFF OF RESOLUTION AND VARIANCE

Suppose that one is attempting to determine a set of model parameters that represents a discretized version of a continuous function, such as X-ray opacity in the medical tomography problem (Figure 4.6). If the discretization is made very fine, then the X-rays will not sample every box; the problem will be underdetermined. If we try to determine the opacity of each box individually, then estimates of opacity will tend to have rather large variance. Few boxes will have several X-rays passing through them, so that little averaging out of the errors will take place. On the other hand, the boxes are very small—and very small features can be detected (the resolution is very good). The large variance can be reduced by increasing the box size (or alternatively, averaging several neighboring boxes). Each of these larger regions will then contain several X-rays, and noise will tend to be averaged out. But because the regions are now larger, small features can no longer be detected and the resolution of the X-ray opacity has become poorer.

This scenario illustrates an important trade-off between model resolution spread and variance size. One can be decreased only at the expense of increasing

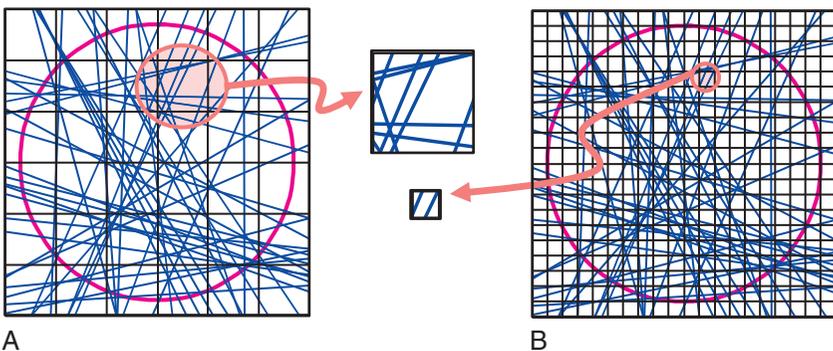


FIGURE 4.6 Hypothetical tomography experiment with (A) large voxels and (B) small voxels. *MatLab* Script. The small voxel case not only has better spatial resolution but also higher variance, as fewer rays pass through each voxel, leaving less opportunity for measurement error to average out. *MatLab* script gda04_06.

the other. We can study this trade-off by choosing a generalized inverse that minimizes a weighted sum of resolution spread and covariance size:

$$\alpha \text{ spread}(\mathbf{R}) + (1 - \alpha) \text{size}([\text{cov}_u \mathbf{m}]) \quad (4.39)$$

If the weighting parameter α is set near 1, then the model resolution matrix of the generalized inverse will have small spread, but the model parameters will have large variance. If α is set close to 0, then the model parameters will have a relatively small variance, but the resolution will have a large spread. A *trade-off curve* can be defined by varying α on the interval $(0, 1)$ (Figure 4.7). Such curves can be helpful in choosing a generalized inverse that has an optimum trade-off in model resolution and variance (judged by criteria appropriate to the problem at hand).

Trade-off curves play an important role in continuous inverse theory, where the discretization is (so to speak) infinitely fine, and all problems are underdetermined. It is known that in this continuous limit the curves are monotonic and possess asymptotes in resolution and variance (Figure 4.8). The process of approximating a continuous function by a finite set of discrete parameters somewhat complicates this picture. The resolution and variance, and indeed the solution itself, are dependent on the parameterization, so it is difficult to make any definitive statement regarding the properties of the trade-off curves. Nevertheless, if the discretization is sufficiently fine, the discrete trade-off curves are usually close to ones obtained with the use of continuous inverse theory. Therefore, discretizations should always be made as fine as computational considerations permit.

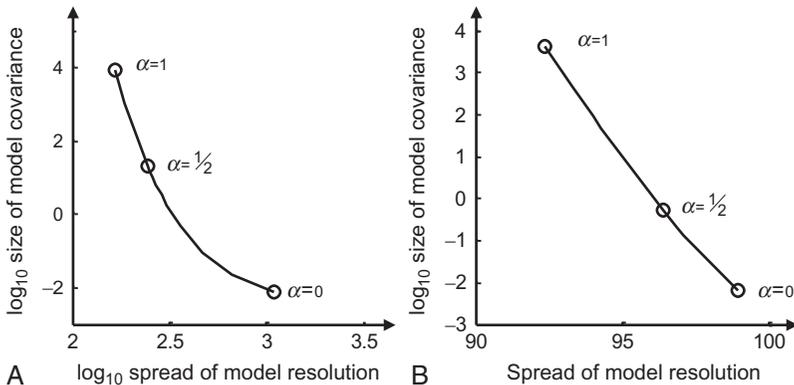


FIGURE 4.7 Trade-off curves of resolution and variance for the inverse problem shown in Figure 4.2. (A) Backus-Gilbert solution, (B) damped minimum length solution. The larger the parameter α , the more weight resolution is given (relative to variance) when forming the generalized inverse. The details of the trade-off curve depend upon the parameterization. The resolution can be no better than the smallest element in the parameterization and no worse than the sum of all the elements. *MatLab* script gda04_05.

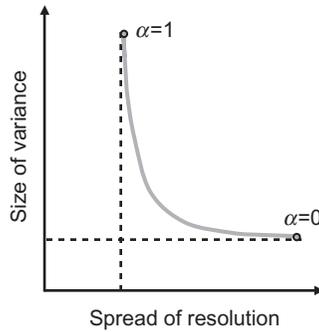


FIGURE 4.8 Trade-off curve of resolution and variance has two asymptotes in the case when the model parameter is a continuous function.

4.12 TECHNIQUES FOR COMPUTING RESOLUTION

In very large problems, the model resolution matrix \mathbf{R} can be cumbersome to compute, owing to its large $M \times M$ size and non-sparse character. Furthermore, time is rarely available for examining all of its rows in detail. Plots of just a few rows, corresponding to model parameters located at strategically chosen points within the model volume, are usually sufficient.

Suppose that we call the k th row of \mathbf{R} the vector $\mathbf{r}^{(k)T}$. Then the identity $\mathbf{R} = \mathbf{R}\mathbf{I}$ can be rewritten as

$$R_{ik} = \sum_{j=1}^M R_{ij} \delta_{jk} \rightarrow r_i^{(k)} = \sum_{j=1}^N R_{ij} m_j^{(k)} \quad \text{with} \quad m_j^{(k)} = \delta_{jk} \quad (4.40)$$

Here we have identified the k th column of \mathbf{I} as “model parameter” vector $m_j^{(k)}$ that is zero except for its k th element, which is unity. Recalling that $\mathbf{R} = \mathbf{G}^{-g} \mathbf{G}$, we can write

$$\mathbf{r}^{(k)} = \mathbf{R}\mathbf{m}^{(k)} = \mathbf{G}^{-g} \mathbf{G}\mathbf{m}^{(k)} = \mathbf{G}^{-g} \mathbf{d}^{(k)} \quad \text{with} \quad \mathbf{d}^{(k)} = \mathbf{G}\mathbf{m}^{(k)} \quad (4.41)$$

Thus, the k th row of the model resolution matrix solves the inverse problem for synthetic data $\mathbf{d}^{(k)}$ corresponding to a specific model parameter vector $\mathbf{m}^{(k)}$, one that is zero except for its k th element, which is unity (that is, a unit spike at row k). This suggests a procedure for calculating the resolution: construct the desired $\mathbf{m}^{(k)}$, solve the forward problem to generate $\mathbf{d}^{(k)}$, solve the inverse problem, and then interpret the result as the k th row of the resolution matrix (Figure 4.9A, B). The great advantage of this technique is that \mathbf{R} in its entirety need not be constructed. Furthermore, the technique will work when the inverse problem is solved by an iterative method, such as the biconjugate gradient method, that does not explicitly construct \mathbf{G}^{-g} .

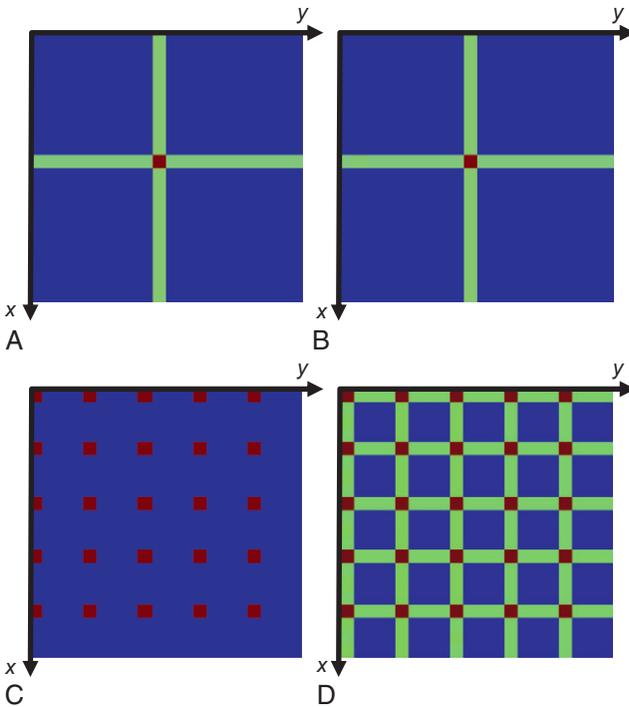


FIGURE 4.9 Resolution of an acoustic tomography problem solved with the minimum length method. The physical model space is a 20×20 grid of pixels on an (x,y) grid. Data are measured only along rows and columns, as in Figure 1.2. (Top row) One row of the resolution matrix, for a model parameter near the center of the (x,y) grid, calculated using two methods, (A) by computing the complete matrix \mathbf{R} and extracting one row and (B) by calculating the row separately. (Bottom row) Checkerboard resolution test showing (C) true checkerboard and (D) reconstructed checkerboard. *MatLab* scripts `gda04_07` and `gda04_08`.

If the resolution of a problem is sufficiently good that the pattern for two well-separated model parameters does not overlap, or overlaps only minimally, then the calculation of two rows of the resolution matrix can be combined into one. One merely solves the inverse problem for synthetic data corresponding to a model parameter vector containing two unit spikes. Nor need one stop with two; a model parameter vector corresponding to a grid of spikes (that is, a *checkerboard*) allows the resolution to be assessed throughout the model volume (Figure 4.9C, D). If the problem has perfect resolution, this checkerboard pattern will be perfectly reproduced. If not, portions of the model volume with poor resolution will contain fuzzy spikes. The main limitation of this technique is that it makes the detection of unlocalized side lobes very difficult, since an unlocalized sidelobe associated with a particular spike will tend to be confused with a localized sidelobe of another spike.

4.13 PROBLEMS

- 4.1.** Consider an underdetermined problem in which each datum is the sum of three neighboring model parameters, that is, $d_i = m_{i-1} + m_i + m_{i+1}$ for $2 \leq i \leq (M-1)$ with $M=100$. Compute and plot both the Dirichlet and Backus-Gilbert model resolution matrices. Use the standard Backus-Gilbert weight function $w(i, j) = (i-j)^2$. Interpret the results.
- 4.2.** This problem builds upon Problem 4.1. How does the Backus-Gilbert result change if you use the weight function $w(i, j) = |i-j|^{1/2}$, which gives less weight to distant sidelobes?
- 4.3.** This problem is especially difficult. Consider a two-dimensional acoustic tomography problem like the one discussed in [Section 1.3.3](#), consisting of a 20×20 rectangular array of pixels, with observations only along rows and columns. (A) Design an appropriate Backus-Gilbert weight function that quantifies the spread of resolution. (B) Write a *MatLab* script that calculates the model resolution matrix \mathbf{R} . (C) Plot a few representative rows of \mathbf{R} , but where each row is reorganized into a two-dimensional image, using the same scheme that was applied to the model parameters. Interpret the results. (Hint: You will need to switch back and forth between a 20×20 rectangular array of model parameters and a length $M=400$ vector of model parameters, as in [Figure 10.12](#).)

REFERENCES

- Backus, G.E., Gilbert, J.F., 1967. Numerical application of a formalism for geophysical inverse problems. *Geophys. J. Roy. Astron. Soc.* 13, 247–276.
- Backus, G.E., Gilbert, J.F., 1968. The resolving power of gross earth data. *Geophys. J. Roy. Astron. Soc.* 16, 169–205.
- Backus, G.E., Gilbert, J.F., 1970. Uniqueness in the inversion of gross Earth data. *Philos. Trans. R. Soc. Lond. A* 266, 123–192.
- Minster, J.F., Jordan, T.J., Molnar, P., Haines, E., 1974. Numerical modelling of instantaneous plate tectonics. *Geophys. J. Roy. Astron. Soc.* 36, 541–576.
- Wiggins, R.A., 1972. The general linear inverse problem: Implication of surface waves and free oscillations for Earth structure. *Rev. Geophys. Space Phys.* 10, 251–285.

Solution of the Linear, Gaussian Inverse Problem, Viewpoint 3: Maximum Likelihood Methods

5.1 THE MEAN OF A GROUP OF MEASUREMENTS

Suppose that an experiment is performed N times and that each time a single datum d_i is collected. Suppose further that these data are all noisy measurements of the same model parameter m_1 . In the view of probability theory, N realizations of random variables, all of which have the same probability density function, have been measured. If these random variables are Gaussian, their joint probability density function can be characterized in terms of a variance σ^2 and a mean m_1 (see Section 2.4) as

$$p(\mathbf{d}) = \sigma^{-N} (2\pi)^{-N/2} \exp \left[-\frac{1}{2} \sigma^{-2} \sum_{i=1}^N [d_i - m_1]^2 \right] \quad (5.1)$$

The data \mathbf{d}^{obs} can be represented graphically as a point in the N -dimensional space whose coordinate axes are d_1, d_2, \dots, d_N (Figure 5.1). The probability density function for the data can also be graphed (Figure 5.2). Note that the probability density function is centered about the line $d_1 = d_2 = \dots = d_N$, since all the d s are supposed to have the same mean, and that it is spherically symmetric, since all the d s have the same variance.

Suppose that we guess a value for the unknown mean and variance, thus fixing the center and diameter of the probability density function. We can then calculate its numerical value at the data $p(\mathbf{d}^{\text{obs}})$. If the guessed values of mean and variance are close to being correct, then $p(\mathbf{d}^{\text{obs}})$ should be a relatively large number. If the guessed values are incorrect, then the probability, or *likelihood*, of the observed data will be small. We can imagine sliding the cloud of probability in Figure 5.2 up along the line and adjusting its diameter until its probability at the point \mathbf{d}^{obs} is maximized.

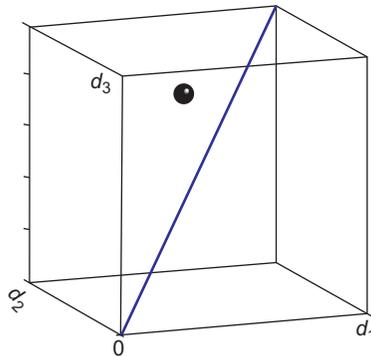


FIGURE 5.1 The data are represented by a single point (black) in a space whose dimensions equal the number of observations (in this case, 3). These data are realizations of random variables with the same mean and variance. Nevertheless, they do not necessarily fall on the line $d_1 = d_2 = d_3$ (blue). *MatLab* script gda05_01.

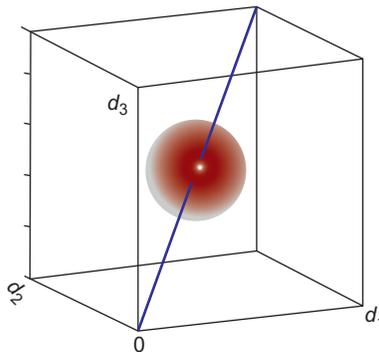


FIGURE 5.2 If the data d_i are assumed to be uncorrelated with equal mean and uniform variance, their probability density function $p(\mathbf{d})$ is a spherical cloud (red), centered on the line $d_1 = d_2 = d_3$ (blue). *MatLab* script gda05_02.

This procedure defines a method of estimating the unknown parameters in the distribution, the *method of maximum likelihood*. It asserts that the optimum values of the parameters maximize the probability that the observed data are in fact observed. In other words, the value of the probability density function at the point \mathbf{d}^{obs} is made as large as possible. The maximum is located by differentiating $p(\mathbf{d}^{\text{obs}})$ with respect to mean and variance and setting the result to zero as

$$\partial p / \partial m_1 = \partial p / \partial \sigma = 0 \quad (5.2)$$

Maximizing $\log p(\mathbf{d}^{\text{obs}})$ gives the same result as maximizing $p(\mathbf{d}^{\text{obs}})$, since $\log(p)$ is a monotonic function of p . We therefore compute derivatives of the *likelihood function*, $L = \log p(\mathbf{d}^{\text{obs}})$ (Figure 5.3). Ignoring the overall normalization of $(2\pi)^{-N/2}$ we have

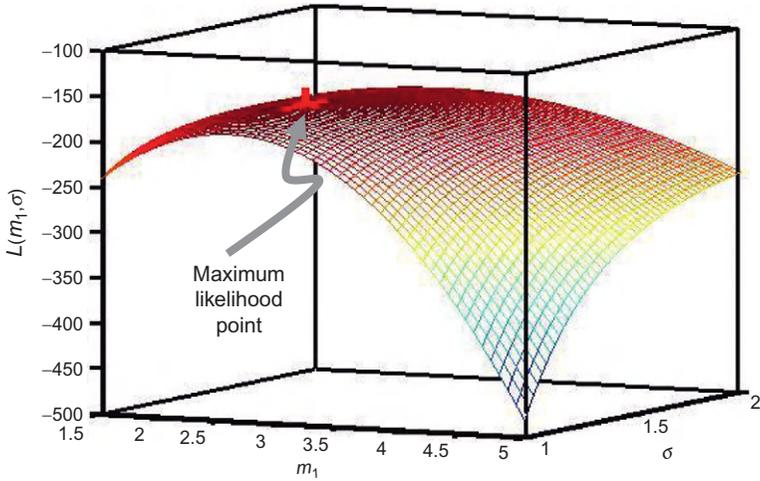


FIGURE 5.3 Likelihood surface for 100 realizations of random variables with equal mean $m_1=2.5$ and uniform variance $\sigma^2=(1.5)^2$. The curvature in the direction of m_1 is greater than the maximum in the direction of the σ , indicating that the former can be determined to greater certainty. *MatLab* script gda05_03.

$$L = \log(p(\mathbf{d}^{\text{obs}})) = -N \log(\sigma) - \frac{1}{2} \sigma^{-2} \sum_{i=1}^N (d_i^{\text{obs}} - m_1)^2$$

$$\frac{\partial L}{\partial m_1} = 0 = \frac{1}{2} \sigma^{-2} 2m_1 \sum_{i=1}^N (d_i^{\text{obs}} - m_1) \quad (5.3)$$

$$\frac{\partial L}{\partial \sigma} = 0 = -\frac{N}{\sigma} + \sigma^{-3} \sum_{i=1}^N (d_i^{\text{obs}} - m_1)^2$$

These equations can be solved for the estimated mean and variance as

$$m_1^{\text{est}} = \frac{1}{N} \sum_{i=1}^N d_i^{\text{obs}} \quad \text{and} \quad \sigma^{\text{est}} = \left[\frac{1}{N} \sum_{i=1}^N (d_i^{\text{obs}} - m_1^{\text{est}})^2 \right]^{1/2} \quad (5.4)$$

The estimate for m_1 is just the usual formulas for the sample mean. The estimate for σ is the root mean squared error and also is almost the formula for the sample standard deviation, except that it has a leading factor of $1/N$, instead of $1/(N-1)$. We note that these estimates arise as a direct consequence of the assumption that the data possess a Gaussian distribution. If the data distribution were not Gaussian, then the arithmetic mean might not be an appropriate estimate of the mean of the distribution. (As we shall see in [Section 8.2](#), the sample median is the maximum likelihood estimate of the mean of an exponential distribution.)

5.2 MAXIMUM LIKELIHOOD APPLIED TO INVERSE PROBLEM

5.2.1 The Simplest Case

Assume that the data in the linear inverse problem $\mathbf{Gm} = \mathbf{d}$ have a multivariate Gaussian probability density function, as given by

$$p(\mathbf{d}) \propto \exp\left[-\frac{1}{2}(\mathbf{d} - \mathbf{Gm})^T [\text{cov } \mathbf{d}]^{-1} (\mathbf{d} - \mathbf{Gm})\right] \quad (5.5)$$

We assume that the model parameters are unknown but (for the sake of simplicity) that the data covariance is known. We can then apply the method of maximum likelihood to estimate the model parameters. The optimum values for the model parameters are the ones that maximize the probability that the observed data are in fact observed. The maximum of $p(\mathbf{d}^{\text{obs}})$ occurs when the argument of the exponential is a maximum or when the quantity given by

$$(\mathbf{d}^{\text{obs}} - \mathbf{Gm})^T [\text{cov } \mathbf{d}]^{-1} (\mathbf{d}^{\text{obs}} - \mathbf{Gm}) \quad (5.6)$$

is a minimum. But this expression is just a weighted measure of prediction error. The maximum likelihood estimate of the model parameters is nothing but the weighted least squares solution, where the weighting matrix is the inverse of the covariance matrix of the data (in the notation of Chapter 3, $\mathbf{W}_e = [\text{cov } \mathbf{d}]^{-1}$). If the data happen to be uncorrelated and all have equal variance, then $[\text{cov } \mathbf{d}] = \sigma_d^2 \mathbf{I}$, and the maximum likelihood solution is the simple least squares solution. If the data are uncorrelated but their variances are all different (say, σ_{di}^2), then the prediction error is given by

$$E = \sum_{i=1}^N \sigma_{di}^{-2} e_i^2 \quad (5.7)$$

where $e_i = (d_i^{\text{obs}} - d_i^{\text{pre}})$ is the prediction error for each datum. Each individual error is weighted by the reciprocal of its standard deviation; the most certain data are weighted most.

We have justified the use of the L_2 norm through the application of probability theory. The least squares procedure for minimizing the L_2 norm of the prediction error makes sense if the data are uncorrelated, have equal variance, and obey Gaussian statistics. If the data are not Gaussian, then other measures of prediction error may be more appropriate.

5.2.2 *A Priori* Distributions

The least squares solution does not exist when the linear problem is underdetermined. From the standpoint of probability theory, the probability density function of the data $p(\mathbf{d}^{\text{obs}})$ has no well-defined maximum with respect to variations of the model parameters. At best, it has a ridge of maximum probability (Figure 5.4).

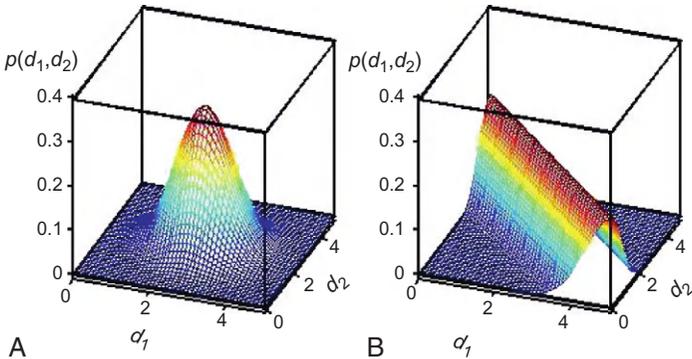


FIGURE 5.4 (A) Probability density function $p(d_1, d_2)$ with a well-defined peak. (B) Probability density function with a ridge. *MatLab* script gda05_04.

We must add *a priori* information that causes the distribution to have a well-defined peak in order to solve an underdetermined problem. One way to accomplish this goal is to write the *a priori* information about the model parameters as a probability density function $p_A(\mathbf{m})$, where the subscript A means “*a priori*.” The mean of this probability density function is then the value we expect the model parameter vector to have, and its shape reflects the certainty of this expectation.

A priori distributions for the model parameters can take a variety of forms. For instance, if we expected that the model parameters are close to $\langle \mathbf{m} \rangle$, we might use a Gaussian distribution with mean $\langle \mathbf{m} \rangle$ and variance that reflects the certainty of our knowledge (Figure 5.5). If the *a priori* value of one model parameter were more certain than another, we might use different variances for

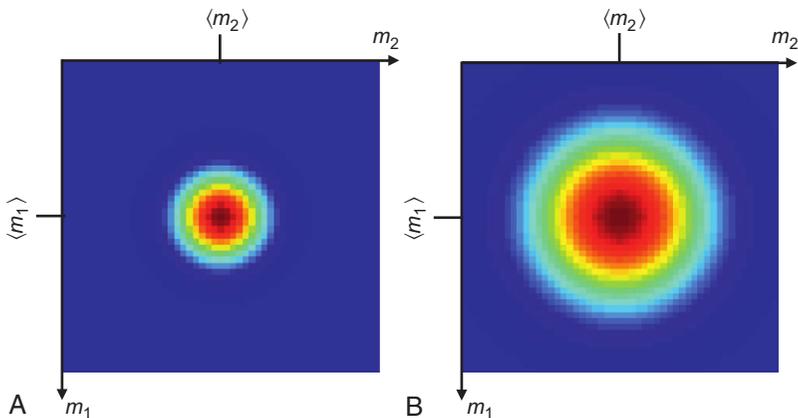


FIGURE 5.5 *A priori* information about model parameters m_1 and m_2 , represented with a probability density function $p(m_1, m_2)$. Most probable values are given by means $\langle m_1 \rangle$ and $\langle m_2 \rangle$. Width of the probability density function reflects certainty of knowledge: (A) certain; (B) uncertain. *MatLab* script gda05_05.

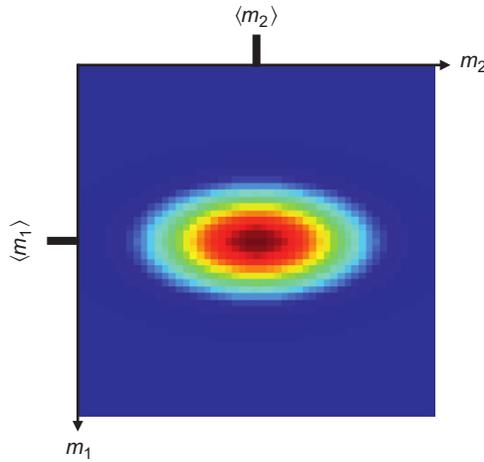


FIGURE 5.6 *A priori* information about model parameters m_1 and m_2 represented with a probability density function $p(m_1, m_2)$. The model parameters are thought to be near $\langle \mathbf{m} \rangle$, with the uncertainty in m_1 less than the uncertainty of m_2 . *MatLab* script `gda05_06`.

the different model parameters (Figure 5.6). The general Gaussian case, with covariance $[\text{cov } \mathbf{m}]_A$, is

$$p_A(\mathbf{m}) \propto \exp \left[-\frac{1}{2} (\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) \right] \quad (5.8)$$

Equality constraints can be implemented with a distribution that contains a ridge (Figure 5.7). This distribution is non-Gaussian but might be approximated by a Gaussian distribution with nonzero covariance if the expected range of the model parameters were small. Inequality constraints can also be represented by an *a priori* distribution but are inherently non-Gaussian (Figure 5.8).

Similarly, one can summarize the state of knowledge about the measurements with an *a priori* probability density function $p_A(\mathbf{d})$. It simply summarizes the observations, so its mean is \mathbf{d}^{obs} and its covariance is the *a priori* covariance $[\text{cov } \mathbf{d}]$ of the data

$$p_A(\mathbf{d}) \propto \exp \left[-\frac{1}{2} (\mathbf{d} - \mathbf{d}^{\text{obs}})^T [\text{cov } \mathbf{d}]^{-1} (\mathbf{d} - \mathbf{d}^{\text{obs}}) \right] \quad (5.9)$$

One important attribute of $p_A(\mathbf{m})$ and $p_A(\mathbf{d})$ is that they contain information about the model parameters \mathbf{m} and the data \mathbf{d} , respectively. The amount of information can be quantified by the *information gain*, a scalar number S defined as

$$\begin{aligned} S[p_A(\mathbf{m})] &= \int p_A(\mathbf{m}) \log \left[\frac{p_A(\mathbf{m})}{p_N(\mathbf{m})} \right] d^M \mathbf{m} \\ S[p_A(\mathbf{d})] &= \int p_A(\mathbf{d}) \log \left[\frac{p_A(\mathbf{d})}{p_N(\mathbf{d})} \right] d^N \mathbf{d} \end{aligned} \quad (5.10)$$

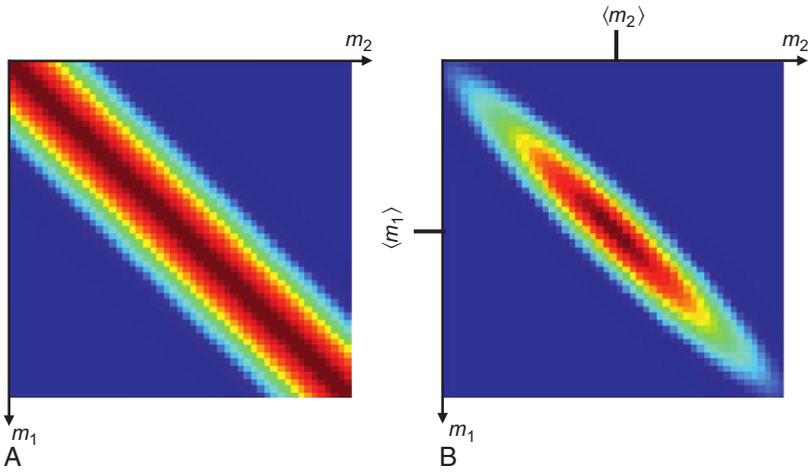


FIGURE 5.7 *A priori* information about model parameters m_1 and m_2 , represented with a probability density function $p(m_1, m_2)$. (A) Case when the values of m_1 and m_2 are unknown, but believed to be correlated. (B) Approximation of (A) with a Gaussian probability density function with finite variance. *MatLab* script gda05_07.

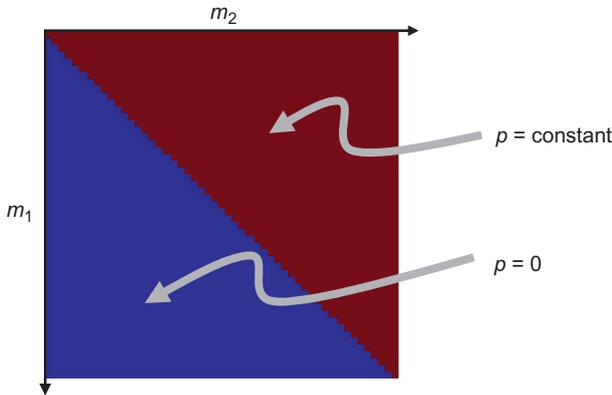


FIGURE 5.8 *A priori* information about model parameters m_1 and m_2 represented with a probability density function $p(m_1, m_2)$. The value of the model parameters are unknown, but the relationship $m_1 \leq m_2$ is believed to hold exactly. This is a non-Gaussian probability density function. *MatLab* script gda05_08.

Here, the *null* probability density functions $p_N(\mathbf{m})$ and $p_N(\mathbf{d})$ express the state of complete ignorance about the model parameters and data, respectively. When the range of \mathbf{m} and \mathbf{d} are bounded, the null probability density functions can be taken to be proportional to a constant; that is, $p_N(\mathbf{m}) \propto \text{constant}$ and $p_N(\mathbf{d}) \propto \text{constant}$, meaning \mathbf{m} and \mathbf{d} “could be anything.” However, when \mathbf{m} and \mathbf{d} are unbounded, the uniform distribution does not exist, and some other probability density function, such as a very wide Gaussian, must be used, instead.

The quantity $-S$ is sometimes called the *relative entropy* between the two probability density functions. A wide distribution is “more random” than a narrow one; it has more *entropy*.

The information gain is always a nonnegative number and is only zero when $p_A(\mathbf{m}) = p_N(\mathbf{m})$ and $p_A(\mathbf{d}) = p_N(\mathbf{d})$ (Figure 5.9). The information gain S has the following properties (Tarantola and Valette, 1982b): (1) the information gain of the null distribution is zero; (2) all distributions except the null distribution have positive information gain; (3) the more sharply peaked the probability density function becomes, the more its information gain increases; and (4) the information gain is invariant under reparameterizations.

We can summarize the state of knowledge about the inverse problem *before* it is solved by first defining an *a priori* probability density function for the data $p_A(\mathbf{d})$ and then combining it with the *a priori* probability density function for the model $p_A(\mathbf{m})$. The *a priori* data probability density function simply summarizes the observations, so its mean is \mathbf{d}^{obs} and its variance is equal to the *a priori* variance of the data. Since the *a priori* model probability density function is completely independent of the actual values of the data, we can form the joint *a priori* probability density function simply by multiplying the two as

$$p_A(\mathbf{m}, \mathbf{d}) = p_A(\mathbf{m})p_A(\mathbf{d}) \quad (5.11)$$

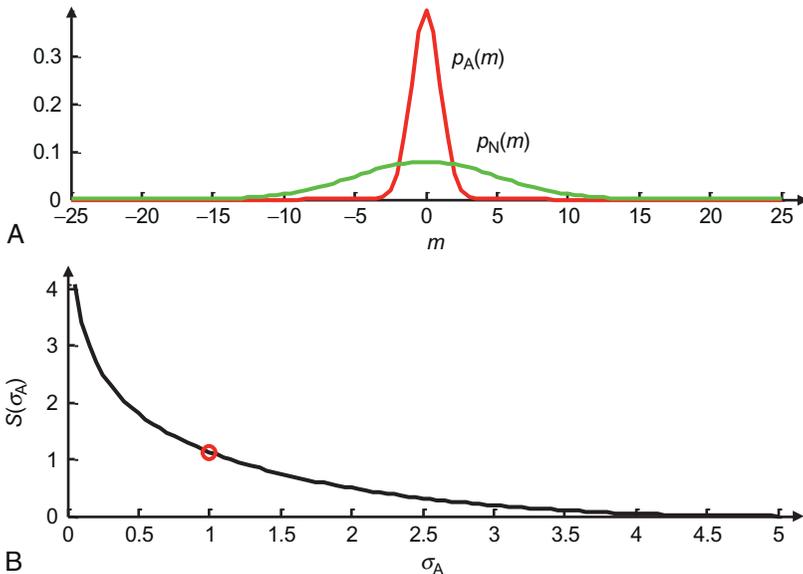


FIGURE 5.9 (A) In this example, a wide Gaussian (green, $\sigma_N=5$) is used for the null probability density function $p_N(\mathbf{m})$ and a narrow Gaussian (red, $\sigma_N=1$) is used for the *a priori* probability density function $p_A(\mathbf{m})$. (B) The information gain S decreases as the width of $p_A(\mathbf{m})$ is increased. The case in (A) is depicted with a red circle. *MatLab* script `gda05_09`.

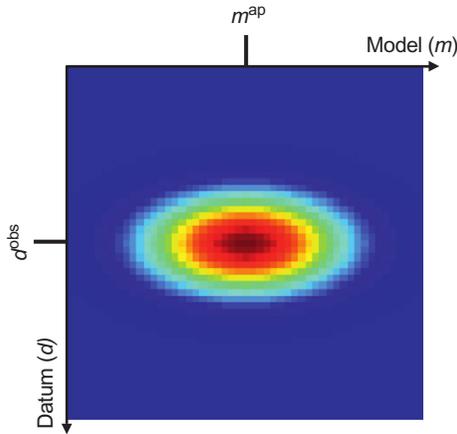


FIGURE 5.10 Joint probability density function $p_A(\mathbf{m}, \mathbf{d})$ for model parameter m and datum d . The distribution is peaked at mean values m^{ap} and d^{obs} . *MatLab* script gda05_10.

This probability density function can be depicted graphically as a “cloud” of probability centered on the observed data and *a priori* model parameters, with a width that reflects the certainty of these quantities (Figure 5.10). If we apply the maximum likelihood method to this distribution, we simply recover the data and *a priori* model. We have not yet applied our knowledge of the model (the relationship between data and model parameters).

5.2.3 Maximum Likelihood for an Exact Theory

Suppose that the model is the rather general equation $\mathbf{g}(\mathbf{m}) = \mathbf{d}$ (which may or may not be linear). This equation defines a surface in the space of model parameters and data along which the solution must lie (Figure 5.11). The maximum likelihood problem then translates into finding the maximum of the joint distribution $p_A(\mathbf{m}, \mathbf{d})$ (or, equivalently, its logarithm) on the surface $\mathbf{d} = \mathbf{g}(\mathbf{m})$ (Tarantola and Valette, B., 1982a):

$$\text{maximize } \log[p(\mathbf{m}, \mathbf{d})] \text{ with the constraint } \mathbf{g}(\mathbf{m}) - \mathbf{d} = \mathbf{0} \quad (5.12)$$

Note that if the *a priori* probability density function for the model parameters is much more certain than that of the observed data (that is, if $\sigma_m < \sigma_d$), then the estimate of the model parameters (the maximum likelihood point) tends to be close to the *a priori* model parameters (Figure 5.12). On the other hand, if the data are far more certain than the model parameters (that is, $\sigma_d < \sigma_m$), then the estimates of the model parameters primarily reflect information contained in the data (Figure 5.13).

In the case of Gaussian probability density function, and the linear theory $\mathbf{d} = \mathbf{Gm}$, we need only to substitute \mathbf{Gm} for \mathbf{d} in the expression for $p_A(\mathbf{d})$ to obtain

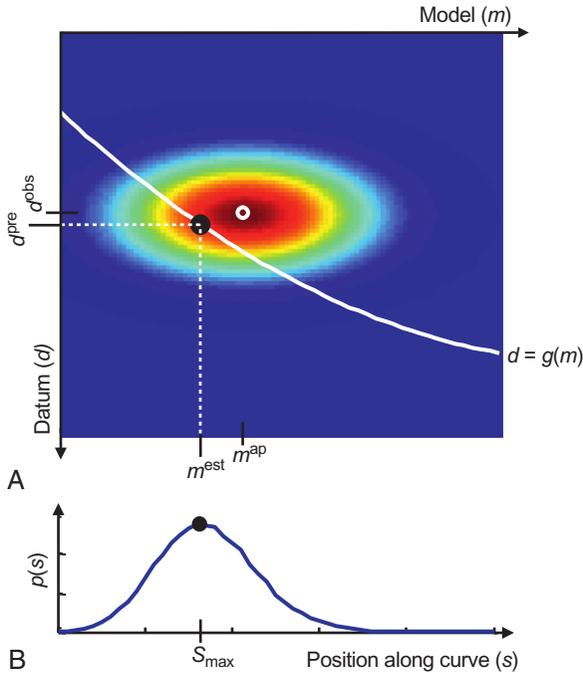


FIGURE 5.11 (A) *A priori* joint probability density function $p(m, d)$ for model parameter m and datum d represents the idea that the model parameter is near its *a priori* value m^{ap} and the datum is near its observed value d^{obs} (white circle). The data and model parameters are believed to be related by an exact theory $d = g(m)$ (white curve). The estimated model parameter m^{est} and predicted datum d^{pre} fall on this curve at the point of maximum probability (black dot). (B) Probability density p evaluated along the curve. The *MatLab* script `gda05_11`.

$$\begin{aligned} \text{minimize } \Phi(\mathbf{m}) &= L(\mathbf{m}) + E(\mathbf{m}) \text{ with respect to } \mathbf{m} \text{ with} \\ L(\mathbf{m}) &= (\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) \\ E(\mathbf{m}) &= (\mathbf{G}\mathbf{m} - \mathbf{d}^{\text{obs}})^T [\text{cov } \mathbf{d}]^{-1} (\mathbf{G}\mathbf{m} - \mathbf{d}^{\text{obs}}) \end{aligned} \quad (5.13)$$

Comparison with [Section 3.9.3](#) indicates that this is the weighted damped least squares problem, with

$$\varepsilon^2 \mathbf{W}_m = [\text{cov } \mathbf{m}]_A^{-1} \quad \text{and} \quad \mathbf{W}_e = [\text{cov } \mathbf{d}]^{-1} \quad (5.14)$$

so its solution is the least squares solution of $\mathbf{F}\mathbf{m} = \mathbf{f}$; that is, $\mathbf{F}^T \mathbf{F} \mathbf{m}^{\text{est}} = \mathbf{F}^T \mathbf{f}$ with

$$\mathbf{F} = \begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \mathbf{G} \\ [\text{cov } \mathbf{m}]_A^{-1/2} \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \mathbf{d}^{\text{obs}} \\ [\text{cov } \mathbf{m}]_A^{-1/2} \langle \mathbf{m} \rangle \end{bmatrix} \quad (5.15)$$

The matrices $[\text{cov } \mathbf{d}]^{-1/2}$ and $[\text{cov } \mathbf{m}]_A^{-1/2}$ can be interpreted as the *certainty* of \mathbf{d}^{obs} and $\langle \mathbf{m} \rangle$, respectively, since they are numerically large when the

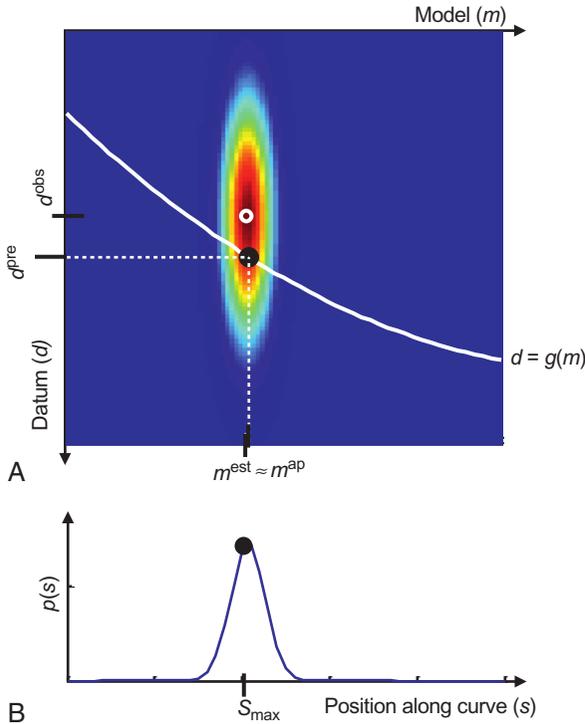


FIGURE 5.12 (A) If the *a priori* model parameter m^{ap} is much more certain than the observed datum d^{obs} , the solution is close to m^{ap} but may be far from d^{obs} . (B) The probability density function p evaluated along the curve. *MatLab* script gda05_12.

uncertainty of these quantities are small. Thus, the top part of the equation $\mathbf{Fm} = \mathbf{f}$ is the data equation $\mathbf{Gm} = \mathbf{d}^{est}$, weighted by its certainty, and the bottom part is the *prior* equation $\mathbf{m} = \langle \mathbf{m} \rangle$, weighted by its certainty. Thus, the matrices \mathbf{W}_m and \mathbf{W}_e of weighted least squares have an important probabilistic interpretation.

The vector \mathbf{f} in Equation (5.15) has unit covariance $[\text{cov } \mathbf{f}] = \mathbf{I}$, since its component quantities $[\text{cov } \mathbf{d}]^{-1/2} \mathbf{d}^{obs}$ and $[\text{cov } \mathbf{m}]_A^{-1/2} \langle \mathbf{m} \rangle$ each have unit covariance (for example, by the usual rules of error propagation, $[\text{cov } \mathbf{d}]^{-1/2T} [\text{cov } \mathbf{d}] [\text{cov } \mathbf{d}]^{-1/2} = \mathbf{I}$). Thus, the covariance of the estimated model parameters are

$$[\text{cov } \mathbf{m}^{est}] = [\mathbf{F}^T \mathbf{F}]^{-1} \tag{5.16}$$

Somewhat incidentally, we note that, had the *a priori* information involved a linear function $\mathbf{Hm} = \mathbf{h}$ of the model parameters, with covariance $[\text{cov } \mathbf{h}]_A$, in contrast to the model parameters themselves, the appropriate form of Equation (5.15) would be

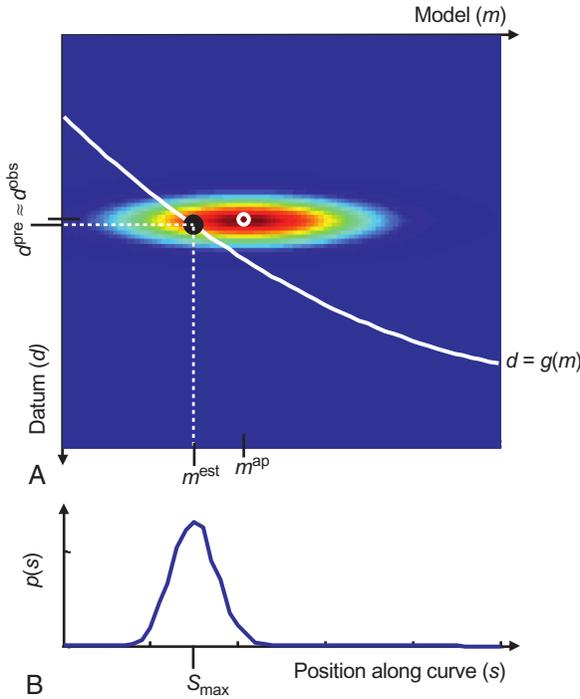


FIGURE 5.13 (A) If the *a priori* model parameter m^{ap} is much less certain than the observed datum d^{obs} , the solution is close to d^{obs} but may be far from m^{ap} . (B) The probability density function p evaluated along the curve. *MatLab* script gda05_13.

$$\mathbf{F} = \begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \mathbf{G} \\ [\text{cov } \mathbf{h}]_A^{-1/2} \mathbf{H} \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \mathbf{d}^{obs} \\ [\text{cov } \mathbf{h}]_A^{-1/2} \mathbf{h} \end{bmatrix} \quad (5.17)$$

This form of weighted damped least squares is especially well suited for computations, especially when $\mathbf{F}^T \mathbf{F} \mathbf{m}^{est} = \mathbf{F}^T \mathbf{f}$ is solved with the biconjugate gradient method.

5.2.4 Inexact Theories

Weighted damped least squares, as it is embodied in Equations (5.15)–(5.17), is extremely useful. Nevertheless, it is somewhat unsatisfying from the standpoint of a probabilistic analysis because the theory has been assumed to be exact. In many realistic problems, there are errors associated with the theory. Some of the assumptions that go into the theory may be unrealistic, or it may be an approximate form of a clumsier but exact theory.

In this case, the model equation $\mathbf{g}(\mathbf{m}) = \mathbf{d}$ can no longer be represented by a simple surface. It has become “fuzzy” because there are now errors associated

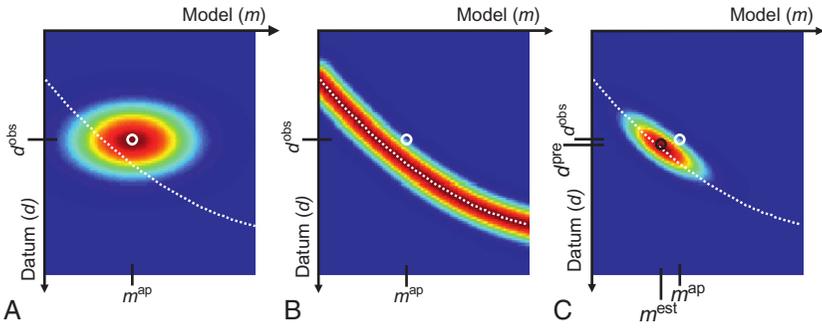


FIGURE 5.14 (A) The *a priori* probability density function $p_A(\mathbf{m}, \mathbf{d})$ represents the state of knowledge before the theory is applied. Its mean (white circle) is the *a priori* model parameter m^{ap} and observed data d^{obs} . (B) An inexact theory is represented by the conditional probability density function $p_g(\mathbf{m}, \mathbf{d})$, which is centered about the exact theory (dotted white curve). (C) The product $p_T(\mathbf{m}, \mathbf{d}) = p_A(\mathbf{m}, \mathbf{d})p_g(\mathbf{m}, \mathbf{d})$ combines the *a priori* information and theory. Its peak is at the estimated data m^{est} and predicted data d^{pre} . *MatLab* script `gda05_14`.

with it (Figure 5.14A; Tarantola and Valette, 1982b). Instead of a surface, one might envision a probability density function $p_g(\mathbf{m}, \mathbf{d})$ centered about $\mathbf{g}(\mathbf{m}) = \mathbf{d}$, with width proportional to the uncertainty of the theory. Rather than find the maximum likelihood point of $p_A(\mathbf{m}, \mathbf{d})$ on a surface, we should instead combine $p_A(\mathbf{m}, \mathbf{d})$ and $p_g(\mathbf{m}, \mathbf{d})$ into a single distribution and find the maximum likelihood point in the overall volume (Figure 5.13C). To proceed, we need a way of combining two probability density functions, each of which contains information about the data and model parameters.

We have already encountered one special case of a combination in our discussion of Bayesian inference (Section 2.7). After adjusting the variable names to match the current discussion, Bayes’ theorem takes the form

$$p(\mathbf{m}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{m})p(\mathbf{m})}{p(\mathbf{d})} \quad \text{or} \quad p(\mathbf{m}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{m}) p(\mathbf{m}) \quad (5.18)$$

Note that the second form omits the denominator, which is not a function of the model parameters and hence acts only as an overall normalization. This second form can be interpreted as updating the information in $p(\mathbf{m})$ (identified now as the *a priori* information) with $p(\mathbf{d}|\mathbf{m})$ (identified now with the data and quantitative model). Thus, in Bayesian inference, probability density functions are combined by multiplication.

In the general case, we denote the process of combining two probability density functions as $p_3 = C(p_1, p_2)$, meaning that functions 1 and 2 are combined into function 3. Then, clearly, the process of combining must have the following properties (adapted from Tarantola and Valette, 1982b):

- (a) The order in which two probability density functions are combined should not matter; that is, $C(p_1, p_2)$ should be commutative: $C(p_1, p_2) = C(p_2, p_1)$.

- (b) The order in which three probability density functions are combined should not matter; that is, $C(p_1, p_2)$ should be associative: $C(p_1, C(p_2, p_3)) = C(C(p_1, p_2), p_3)$.
- (c) Combining a distribution with the null distribution should return the same distribution; that is, $C(p_1, p_N) = p_1$.
- (d) The combination $C(p_1, p_2)$ should never be everywhere zero except if p_1 or p_2 is everywhere zero.
- (e) The combination $C(p_1, p_2)$ should be invariant under reparameterizations.

These conditions can be shown to be satisfied by the choice (Tarantola and Valette, 1982b):

$$p_3 = C(p_1, p_2) = \frac{p_1 p_2}{p_N} \quad (5.19)$$

(at least up to an overall normalization). Note that if the null distribution is constant (as we shall assume for the rest of this chapter), one combines distributions simply by multiplying them:

$$p_T(\mathbf{m}, \mathbf{d}) = p_A(\mathbf{m}, \mathbf{d}) p_g(\mathbf{m}, \mathbf{d}) \quad (5.20)$$

Here, the subscript T means the combined or total distribution. Note that as the error associated with the theory increases, the maximum likelihood point moves back toward the *a priori* values of model parameters and observed data (Figure 5.15). The limiting case in which the theory is infinitely accurate is equivalent to the case in which the distribution is replaced by a distinct surface.

The maximum likelihood point of $p_T(\mathbf{m}, \mathbf{d})$ is specified by both a set of model parameters \mathbf{m}^{est} and a set of data \mathbf{d}^{pre} . They are determined simultaneously. This approach is different from that of the least squares problem examined in Section 5.2. In that section, we maximized the probability density function with respect to the model parameters only to determine the most probable model parameters \mathbf{m}^{est} and afterward can calculate the predicted data via $\mathbf{d}^{\text{pre}} = \mathbf{G}\mathbf{m}^{\text{est}}$. The two methods do not necessarily yield the same estimates for the model parameters. To find the likelihood point of $p_T(\mathbf{m}, \mathbf{d})$ with respect to model parameters only, we must sum all the probabilities along lines of equal model parameter. This summation can be thought of as projecting the distribution onto the $\mathbf{d} = 0$ plane (Figure 5.16) and then finding the maximum. The projected distribution $p(\mathbf{m})$ is then

$$p(\mathbf{m}) = \int p_T(\mathbf{m}, \mathbf{d}) d^N \quad (5.21)$$

where the integration is performed over the entire range of the ds .

5.2.5 The Simple Gaussian Case with a Linear Theory

To illustrate this method, we rederive the weighted damped least squares solution, with *a priori* probability density function:

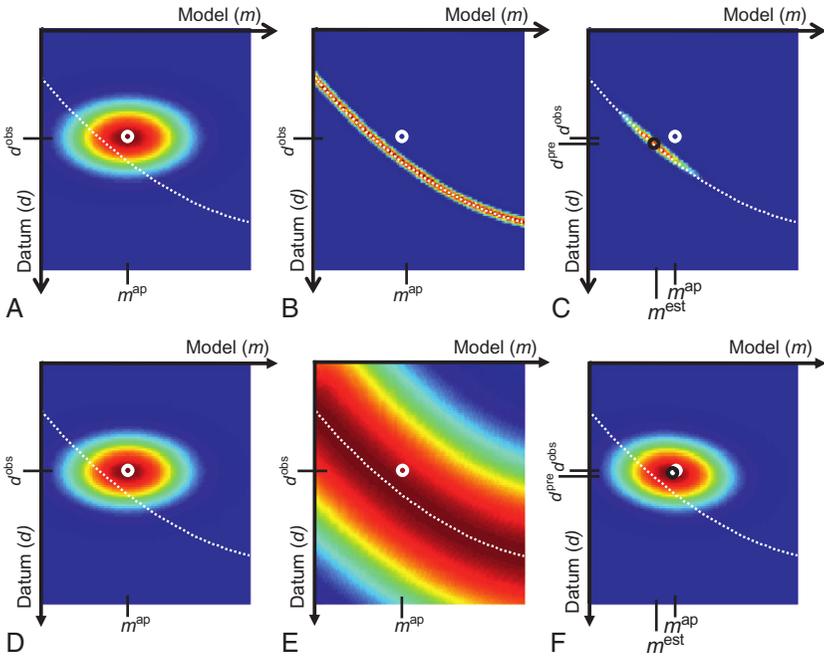


FIGURE 5.15 The rows of the figure have the same format as Figure 5.14. If the theory is made more and more inexact (compare (A–C) with (D–F)), the solution (black circle) moves toward the maximum likelihood point of the *a priori* distribution. *MatLab* script gda05_15.

$$p_A(\mathbf{m}, \mathbf{d}) \propto$$

$$\exp\left[-\frac{1}{2}(\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov} \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) - \frac{1}{2}(\mathbf{d} - \mathbf{d}^{\text{obs}})^T [\text{cov} \mathbf{d}]^{-1} (\mathbf{d} - \mathbf{d}^{\text{obs}})\right] \quad (5.22)$$

If there are no errors in the theory, then its probability density function is “infinitely narrow” and can be represented by a *Dirac delta function*

$$p_A(\mathbf{m}, \mathbf{d}) = \delta(\mathbf{Gm} - \mathbf{d}) \quad (5.23)$$

where we assume that we are dealing with the linear theory $\mathbf{Gm} = \mathbf{d}$. The total distribution is then given by

$$p_T(\mathbf{m}, \mathbf{d}) = p_A(\mathbf{m}, \mathbf{d})\delta(\mathbf{Gm} - \mathbf{d}) \quad (5.24)$$

Performing the projection “integrates away” the delta function

$$p(\mathbf{m}) = \int p_A(\mathbf{m}, \mathbf{d})\delta(\mathbf{Gm} - \mathbf{d})d^N \mathbf{d} \propto \exp\left[-\frac{1}{2}(\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov} \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) - \frac{1}{2}(\mathbf{Gm} - \mathbf{d}^{\text{obs}})^T [\text{cov} \mathbf{d}]^{-1} (\mathbf{Gm} - \mathbf{d}^{\text{obs}})\right] \quad (5.25)$$

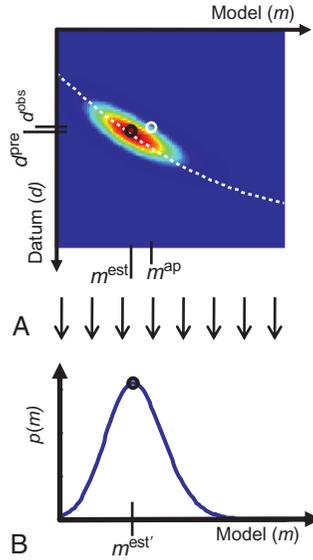


FIGURE 5.16 (A) The joint probability density function $p_T(\mathbf{m}, \mathbf{d})$ can be considered the solution to the inverse problem. Its maximum likelihood point (black circle) gives an estimate of the model parameter m^{est} and a prediction of the data d^{pre} . (B) The function $p_T(\mathbf{m}, \mathbf{d})$ is projected onto the m -axis, by integrating over d , to form the probability density function $p(m)$ of the model parameter irrespective of the datum. This function also has a maximum likelihood point $m^{\text{est}'}$ which in general can be different than m^{est} . The distinction points out the difficulty of defining a unique “solution” to an inverse problem. *MatLab* script gda05_16.

This projected distribution is exactly the one we encountered in the weighted damped least squares problem (the logarithm of which is shown in Equation (5.13)).

5.2.6 The General Linear, Gaussian Case

In the general linear, Gaussian case, we assume that all the component probability density functions are Gaussian and that the theory is the linear equation $\mathbf{Gm} = \mathbf{d}$, so that

$$\begin{aligned}
 p_A(\mathbf{m}) &\propto \exp \left[-\frac{1}{2} (\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) \right] \\
 p_A(\mathbf{d}) &\propto \exp \left[-\frac{1}{2} (\mathbf{d} - \mathbf{d}^{\text{obs}})^T [\text{cov } \mathbf{d}]^{-1} (\mathbf{d} - \mathbf{d}^{\text{obs}}) \right] \\
 p_g(\mathbf{m}, \mathbf{d}) &\propto \exp \left[-\frac{1}{2} (\mathbf{d} - \mathbf{Gm})^T [\text{cov } \mathbf{g}]^{-1} (\mathbf{d} - \mathbf{Gm}) \right]
 \end{aligned} \tag{5.26}$$

Here, the theory is represented by a Gaussian probability density function with covariance $[\text{cov } \mathbf{g}]$. The total distribution $p_T(\mathbf{m}, \mathbf{d})$ is the product of these three distributions. As we noted in Section 2.4, products of Gaussian probability density functions are themselves Gaussian, so $p_T(\mathbf{m}, \mathbf{d})$ is Gaussian. We need to determine the mean of this distribution, since it is also the maximum likelihood point that defines \mathbf{m}^{est} and \mathbf{d}^{pre} .

To simplify the algebra, we first define a vector $\mathbf{x} = [\mathbf{d}^T, \mathbf{m}^T]^T$ that contains the data and model parameters, a vector $\langle \mathbf{x} \rangle = [\mathbf{d}^{\text{obs}T}, \langle \mathbf{m} \rangle^T]^T$ that contains their *a priori* values and a covariance matrix

$$[\text{cov } \mathbf{x}] = \begin{bmatrix} [\text{cov } \mathbf{d}] & 0 \\ 0 & [\text{cov } \mathbf{m}]_A \end{bmatrix} \quad (5.27)$$

The first two products in the total distribution can then be combined into an exponential, with the argument given by

$$-\frac{1}{2}(\mathbf{x} - \langle \mathbf{x} \rangle)^T [\text{cov } \mathbf{x}]^{-1} (\mathbf{x} - \langle \mathbf{x} \rangle) \quad (5.28)$$

To express the third product in terms of \mathbf{x} , we define a matrix $\mathbf{F} = [\mathbf{I}, -\mathbf{G}]$ such that $\mathbf{F}\mathbf{x} = \mathbf{d} - \mathbf{G}\mathbf{m} = 0$. The argument of the third product's exponential is then given by

$$-\frac{1}{2}(\mathbf{F}\mathbf{x})^T [\text{cov } \mathbf{g}]^{-1} (\mathbf{F}\mathbf{x}) \quad (5.29)$$

The total distribution is proportional to an exponential with argument

$$\begin{aligned} & -\frac{1}{2}(\mathbf{x} - \langle \mathbf{x} \rangle)^T [\text{cov } \mathbf{x}]^{-1} (\mathbf{x} - \langle \mathbf{x} \rangle) - \frac{1}{2}(\mathbf{F}\mathbf{x})^T [\text{cov } \mathbf{g}]^{-1} (\mathbf{F}\mathbf{x}) = \\ & -\frac{1}{2}\mathbf{x}^T ([\text{cov } \mathbf{x}]^{-1} + \mathbf{F}^T [\text{cov } \mathbf{g}]^{-1} \mathbf{F}) \mathbf{x} - \mathbf{x}^T [\text{cov } \mathbf{x}]^{-1} \langle \mathbf{x} \rangle - \frac{1}{2} \langle \mathbf{x} \rangle^T [\text{cov } \mathbf{x}]^{-1} \langle \mathbf{x} \rangle \end{aligned} \quad (5.30)$$

We would like to manipulate this expression into the standard form of the argument of a Gaussian probability density function; that is, an expression involving just a single vector, say \mathbf{x}^* , and a single covariance matrix, say $[\text{cov } \mathbf{x}^*]$, related by

$$\begin{aligned} & -\frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T [\text{cov } \mathbf{x}^*]^{-1} (\mathbf{x} - \mathbf{x}^*) = \\ & -\frac{1}{2}\mathbf{x}^T [\text{cov } \mathbf{x}^*]^{-1} \mathbf{x} + \mathbf{x}^T [\text{cov } \mathbf{x}^*]^{-1} \mathbf{x}^* - \frac{1}{2}\mathbf{x}^{*T} [\text{cov } \mathbf{x}^*]^{-1} \mathbf{x}^* \end{aligned} \quad (5.31)$$

We can identify \mathbf{x}^* and $[\text{cov } \mathbf{x}^*]$ by matching terms of equal powers of \mathbf{x} with Equation (5.30). Matching the quadratic terms yields

$$[\text{cov } \mathbf{x}^*]^{-1} = [\text{cov } \mathbf{x}]^{-1} + \mathbf{F}^T [\text{cov } \mathbf{g}]^{-1} \mathbf{F} \quad (5.32)$$

and matching the linear terms yields

$$[\text{cov } \mathbf{x}^*]^{-1} \mathbf{x}^* = [\text{cov } \mathbf{x}]^{-1} \langle \mathbf{x} \rangle \quad \text{or} \quad \mathbf{x}^* = [\text{cov } \mathbf{x}^*][\text{cov } \mathbf{x}]^{-1} \langle \mathbf{x} \rangle \quad (5.33)$$

These choices do not match the constant term, but such a match is not necessary, because the constant term effects only the overall normalization of the probability density function $p_T(\mathbf{x})$. The vector \mathbf{x}^* corresponds to the maximum likelihood point of $p_T(\mathbf{x})$ and so can be considered the solution to the inverse problem. This solution has covariance $[\text{cov } \mathbf{x}^*]$.

Equation (5.32) for $[\text{cov } \mathbf{x}^*]^{-1}$ can be explicitly inverted to yield an expression for $[\text{cov } \mathbf{x}^*]$, using two matrix identities that we now derive (adapted from Tarantola and Valette, 1982a, with permission). Let \mathbf{C}_1 and \mathbf{C}_2 be two symmetric matrices whose inverses exist, and let \mathbf{M} be a third matrix. The expression $\mathbf{M}^T + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M} \mathbf{C}_2 \mathbf{M}^T$ can be written in two ways by grouping terms as $\mathbf{M}^T \mathbf{C}_1^{-1} [\mathbf{C}_1 + \mathbf{M} \mathbf{C}_2 \mathbf{M}^T]$ or $[\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}] \mathbf{C}_2 \mathbf{M}^T$. Multiplying by the matrix inverses gives

$$\mathbf{C}_2 \mathbf{M}^T [\mathbf{C}_1 + \mathbf{M} \mathbf{C}_2 \mathbf{M}^T]^{-1} = [\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}]^{-1} \mathbf{M}^T \mathbf{C}_1^{-1}. \quad (5.34)$$

Now consider the symmetric matrix expression $\mathbf{C}_2 - \mathbf{C}_2 \mathbf{M}^T [\mathbf{C}_1 + \mathbf{M} \mathbf{C}_2 \mathbf{M}^T]^{-1} \mathbf{M} \mathbf{C}_2$. By Equation (5.34), this expression equals $\mathbf{C}_2 - [\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}]^{-1} \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M} \mathbf{C}_2$. Factoring out the term in brackets gives $[\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}]^{-1} \{[\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}] \mathbf{C}_2 - \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M} \mathbf{C}_2\}$. Canceling terms gives $[\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}]^{-1}$ from which we conclude

$$\begin{aligned} [\mathbf{C}_2^{-1} + \mathbf{M}^T \mathbf{C}_1^{-1} \mathbf{M}]^{-1} &= \mathbf{C}_2 - \mathbf{C}_2 \mathbf{M}^T [\mathbf{C}_1 + \mathbf{M} \mathbf{C}_2 \mathbf{M}^T]^{-1} \mathbf{M} \mathbf{C}_2 \\ &= \{\mathbf{I} - \mathbf{C}_2 \mathbf{M}^T [\mathbf{C}_1 + \mathbf{M} \mathbf{C}_2 \mathbf{M}^T]^{-1} \mathbf{M}\} \mathbf{C}_2 \end{aligned} \quad (5.35)$$

Equating now $\mathbf{C}_2 = [\text{cov } \mathbf{x}]$, $\mathbf{C}_1 = [\text{cov } \mathbf{g}]$, and $\mathbf{M} = \mathbf{F}$, Equation (5.32) becomes

$$\begin{aligned} [\text{cov } \mathbf{x}^*] &= [[\text{cov } \mathbf{x}]^{-1} + \mathbf{F}^T [\text{cov } \mathbf{g}]^{-1} \mathbf{F}]^{-1} = \\ &= \{\mathbf{I} - [\text{cov } \mathbf{x}] \mathbf{F}^T [[\text{cov } \mathbf{g}] + \mathbf{F} [\text{cov } \mathbf{x}] \mathbf{F}^T]^{-1} \mathbf{F}\} [\text{cov } \mathbf{x}] \end{aligned} \quad (5.36)$$

and Equation (5.33) becomes

$$\mathbf{x}^* = [\text{cov } \mathbf{x}^*][\text{cov } \mathbf{x}]^{-1} \langle \mathbf{x} \rangle = \{\mathbf{I} - [\text{cov } \mathbf{x}] \mathbf{F}^T [[\text{cov } \mathbf{g}] + \mathbf{F} [\text{cov } \mathbf{x}] \mathbf{F}^T]^{-1} \mathbf{F}\} \langle \mathbf{x} \rangle \quad (5.37)$$

Nothing in this derivation requires the special forms of \mathbf{F} and $[\text{cov } \mathbf{x}]$ assumed above that made $\mathbf{F}\mathbf{x} = 0$ separable into an *explicit* linear inverse problem. Equation (5.37) is in fact the solution to the completely general, *implicit*, linear inverse problem.

When $\mathbf{F}\mathbf{x} = 0$ is an explicit equation, the formula for \mathbf{x}^* in Equation (5.21) can be decomposed into its component vectors \mathbf{d}^{pre} and \mathbf{m}^{est} by substituting the definition of \mathbf{F} and $[\text{cov } \mathbf{x}]$ (Equation (5.27)) into it and performing the matrix multiplications. An explicit formula for the estimated model parameters is then given by

$$\mathbf{m}^{\text{est}} = \langle \mathbf{m} \rangle + \mathbf{G}^{-\text{g}}(\mathbf{d}^{\text{obs}} - \mathbf{G}\langle \mathbf{m} \rangle) = \mathbf{G}^{-\text{g}}\mathbf{d}^{\text{obs}} + [\mathbf{I} - \mathbf{R}]\langle \mathbf{m} \rangle$$

with $\mathbf{G}^{-\text{g}} = [\text{cov } \mathbf{m}]_{\text{A}} \mathbf{G}^{\text{T}} \{ [\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}] + \mathbf{G}[\text{cov } \mathbf{m}]_{\text{A}} \mathbf{G}^{\text{T}} \}^{-1}$ (5.38)

where we have used the generalized inverse $\mathbf{G}^{-\text{g}}$ and resolution matrix $\mathbf{R} = \mathbf{G}^{-\text{g}}\mathbf{G}$ notation for convenience. This generalized inverse is reminiscent of minimum-length inverse $\mathbf{G}^{\text{T}}[\mathbf{G}\mathbf{G}^{\text{T}}]^{-1}$. However, by equating $\mathbf{C}_2 = [\text{cov } \mathbf{m}]_{\text{A}}$, $\mathbf{C}_1 = [\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}]$, and $\mathbf{M} = \mathbf{G}$ in matrix identity Equation (5.34), we can also write the generalized inverse as

$$\mathbf{G}^{-\text{g}} = \{ \mathbf{G}^{\text{T}}([\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}])^{-1} \mathbf{G} + [\text{cov } \mathbf{m}]_{\text{A}}^{-1} \}^{-1} \mathbf{G}^{\text{T}}([\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}])^{-1}$$

(5.39)

which is reminiscent of least squares generalized inverse $[\mathbf{G}^{\text{T}}\mathbf{G}]^{-1}\mathbf{G}^{\text{T}}$. Both forms of the generalized inverse depend only upon the sum of the covariance of the data $[\text{cov } \mathbf{d}]$ and the covariance of the theory $[\text{cov } \mathbf{g}]$; that is, they make only a combined contribution. This is the most important insight gained from this problem, for the exact-theory case (Equation (5.15)) can be made identical to the inexact-theory case (Equations (5.38) and (5.39)) with the substitution

$$[\text{cov } \mathbf{d}] \rightarrow [\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}]$$

(5.40)

Hence, from the point of view of computations, one continues to use Equation (5.15) but adjusts the covariance using Equation (5.40).

Since the estimated model parameters are a linear combination of observed data and *a priori* model parameters, we can therefore calculate its covariance as

$$[\text{cov } \mathbf{m}^{\text{est}}] = \mathbf{G}^{-\text{g}}[\text{cov } \mathbf{d}]\mathbf{G}^{-\text{gT}} + [\mathbf{I} - \mathbf{R}][\text{cov } \mathbf{m}]_{\text{A}}[\mathbf{I} - \mathbf{R}]^{\text{T}}$$

(5.41)

This expression differs from those derived in Chapters 3 and 4 in that it contains a term dependent on the *a priori* model parameter covariance $[\text{cov } \mathbf{m}]_{\text{A}}$.

We can examine a few interesting limiting cases of problems which have uncorrelated *a priori* model parameters ($[\text{cov } \mathbf{m}]_{\text{A}} = \sigma_m^2 \mathbf{I}$), data ($[\text{cov } \mathbf{d}]_{\text{A}} = \sigma_d^2 \mathbf{I}$), and theory ($[\text{cov } \mathbf{g}] = \sigma_g^2 \mathbf{I}$).

5.2.7 Exact Data and Theory

Suppose $\sigma_d^2 = \sigma_g^2 = 0$. The solution is then given by

$$\mathbf{m}^{\text{est}} = \mathbf{G}^{\text{T}}[\mathbf{G}\mathbf{G}^{\text{T}}]^{-1}\mathbf{d}^{\text{obs}} = [\mathbf{G}^{\text{T}}\mathbf{G}]^{-1}\mathbf{G}^{\text{T}}\mathbf{d}^{\text{obs}}$$

(5.42)

Note that the solution does not depend on the *a priori* model variance, since the data and theory are infinitely more accurate than the *a priori* model parameters. These solutions are just the minimum-length and least squares solutions, which (as we now see) are simply two different aspects of the same solution. The minimum-length form of the solution, however, exists only when the problem

is purely underdetermined; the least squares form exists only when the problem is purely overdetermined.

If the *a priori* model parameters are not equal to zero, then another term appears in the estimated solution:

$$\begin{aligned}\mathbf{m}^{\text{est}} &= \mathbf{G}^{-\mathbf{g}} \mathbf{d}^{\text{obs}} + (\mathbf{I} - \mathbf{R}) \langle \mathbf{m} \rangle \\ &= \mathbf{G}^{\text{T}} [\mathbf{G} \mathbf{G}^{\text{T}}]^{-1} \mathbf{d}^{\text{obs}} + \{ \mathbf{I} - \mathbf{G}^{\text{T}} [\mathbf{G} \mathbf{G}^{\text{T}}]^{-1} \mathbf{G} \} \langle \mathbf{m} \rangle \\ &= [\mathbf{G}^{\text{T}} \mathbf{G}]^{-1} \mathbf{G}^{\text{T}} \mathbf{d}^{\text{obs}}\end{aligned}\quad (5.43)$$

The minimum-length-type solution has been changed by adding a weighted amount of the *a priori* model vector, with the weighting factor being $\{ \mathbf{I} - \mathbf{G}^{\text{T}} [\mathbf{G} \mathbf{G}^{\text{T}}]^{-1} \mathbf{G} \}$. This term is not zero, since it can also be written as $\{ \mathbf{I} - \mathbf{R} \}$. The resolution matrix of the underdetermined problem never equals the identity matrix. On the other hand, the resolution matrix of the overdetermined least squares problem does equal the identity matrix, so the estimated model parameters of the overdetermined problem are not a function of the *a priori* model parameters. Adding *a priori* information with finite error to an inverse problem that features exact data and theory only affects the underdetermined part of the solution.

5.2.8 Infinitely Inexact Data and Theory

In the case of infinitely inexact data and theory, we take the $\sigma_d^2 \rightarrow \infty$ or $\sigma_g^2 \rightarrow \infty$ (or both). The solution becomes

$$\mathbf{m}^{\text{est}} = \langle \mathbf{m} \rangle \quad (5.44)$$

Since the data and theory contain no information, we simply recover the *a priori* model parameters.

5.2.9 No *A Priori* Knowledge of the Model Parameters

In this case, the limit is $\sigma_m^2 \rightarrow \infty$. The solutions are the same as in Section 5.2.7:

$$\mathbf{m}^{\text{est}} = \mathbf{G}^{\text{T}} [\mathbf{G} \mathbf{G}^{\text{T}}]^{-1} \mathbf{d}^{\text{obs}} + \{ \mathbf{I} - \mathbf{G}^{\text{T}} [\mathbf{G} \mathbf{G}^{\text{T}}]^{-1} \mathbf{G} \} \langle \mathbf{m} \rangle = [\mathbf{G}^{\text{T}} \mathbf{G}]^{-1} \mathbf{G}^{\text{T}} \mathbf{d}^{\text{obs}} \quad (5.45)$$

Infinitely weak *a priori* information and finite-error data and theory produce the same results as finite-error *a priori* information and error-free data and theory.

5.3 RELATIVE ENTROPY AS A GUIDING PRINCIPLE

In Section 5.2.2, we introduced the information gain, S (Equation (5.10)), as a way of quantifying the difference in information content between two probability density functions. It can be used as a guiding principle for constructing solutions to inverse problems. The idea is to find the probability density function

$p_T(\mathbf{m})$ —the solution to the inverse problem—that minimizes the information gain of $p_T(\mathbf{m})$ relative to the *a priori* probability density function $p_A(\mathbf{m})$. Thus, as little information as possible has been added to the *a priori* information to create the solution. The quantity $-S$ is the relative entropy of the two probability density functions, so this method is called the *Maximum Relative Entropy* method and is often abbreviated MRE (Kapur, 1989). Some authors define the entropy as $+S$, in which case it is called the *Minimum Relative Entropy* method (also abbreviated MRE).

Constraints need to be added to the minimization of S or else the solution would simply be $p_T(\mathbf{m})=p_A(\mathbf{m})$. One of these constraints must be that the area beneath $p_T(\mathbf{m})$ is unity. The choice of the other constraints depends on the particular type of inverse problem; that is, whether it is under- or overdetermined.

As an example, consider the underdetermined problem, where the equation $\mathbf{d}=\mathbf{Gm}$ can be assumed to hold in the mean. The minimization problem is

$$\begin{aligned} \text{minimize: } S = \int p_T(\mathbf{m}) \log \left(\frac{p_T(\mathbf{m})}{p_A(\mathbf{m})} \right) d^M m \quad \text{with constraints} \\ \int p_T(\mathbf{m}) d^M m = 1 \quad \text{and} \quad \int p_T(\mathbf{m})(\mathbf{d} - \mathbf{Gm}) d^M m = 0 \end{aligned} \quad (5.46)$$

Here, the final distribution $p_T(\mathbf{m})$ is unknown and the *a priori* distribution $p_A(\mathbf{m})$ is prescribed. Note that the second constraint indicates that the mean (expected) value of the error $\mathbf{e}=\mathbf{d}-\mathbf{Gm}$ is zero.

This minimization problem can be solved by using the *Euler-Lagrange method*. It states that the integral $\int F[f(\mathbf{m}), \mathbf{m}] d^M m$ is minimized subject to the integral constraint $\int G[f(\mathbf{m}), \mathbf{m}] d^M m$ when $\Phi=F+\lambda G$ is minimized with respect to f . Here, λ is a Lagrange multiplier. In our case, we introduce one Lagrange multiplier λ_0 associated with the first constraint and a vector $\boldsymbol{\lambda}$ of Lagrange multipliers associated with the second

$$\Phi(\mathbf{m}) = p_T \log(p_T) - p_T \log(p_A) + \lambda_0 p_T + \boldsymbol{\lambda}^T (\mathbf{d} - \mathbf{Gm}) p_T \quad (5.47)$$

Differentiating with respect to p_T yields

$$\frac{\partial \Phi}{\partial p_T} = 0 = \log(p_T) + 1 - \log(p_A) + \lambda_0 + \boldsymbol{\lambda}^T (\mathbf{d} - \mathbf{Gm}) \quad (5.48)$$

or

$$p_T(\mathbf{m}) = p_A(\mathbf{m}) \exp\{-(1 + \lambda_0) - \boldsymbol{\lambda}^T (\mathbf{d} - \mathbf{Gm})\}. \quad (5.49)$$

Now suppose that the *a priori* probability density function is Gaussian in form, with *a priori* value $\langle \mathbf{m} \rangle$ and *a priori* covariance $[\text{cov } \mathbf{m}]_A$

$$p_A(\mathbf{m}) \propto \exp\left\{-\frac{1}{2}(\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle)\right\} \quad (5.50)$$

Then

$$p_T(\mathbf{m}) \propto \exp\{-A(\mathbf{m})\} \quad \text{with} \\ A(\mathbf{m}) = \frac{1}{2}(\mathbf{m} - \langle \mathbf{m} \rangle)^T [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m} - \langle \mathbf{m} \rangle) - (1 + \lambda_0) - \boldsymbol{\lambda}^T (\mathbf{d} - \mathbf{G}\mathbf{m}) \quad (5.51)$$

We now assert that the best estimate of the model parameter \mathbf{m}^{est} is the mean of this distribution, which is also its maximum likelihood point. This point occurs where $A(\mathbf{m})$ is minimum:

$$\frac{\partial A}{\partial m_q} = 0 \quad \text{or} \quad 0 = [\text{cov } \mathbf{m}]_A^{-1} (\mathbf{m}^{\text{est}} - \langle \mathbf{m} \rangle) + \mathbf{G}^T \boldsymbol{\lambda} \quad (5.52)$$

Premultiplying by $\mathbf{G}[\text{cov } \mathbf{m}]_A$, substituting in the constraint equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ (which is assumed to hold in the mean) and rearranging yields

$$\boldsymbol{\lambda} = \{\mathbf{G}[\text{cov } \mathbf{m}]_A \mathbf{G}^T\}^{-1} \{\mathbf{d} - \mathbf{G}\langle \mathbf{m} \rangle\}. \quad (5.53)$$

Substituting this expression for $\boldsymbol{\lambda}$ into Equation (5.52) for $\partial A / \partial \mathbf{m}$ yields the solution

$$\mathbf{m}^{\text{est}} - \langle \mathbf{m} \rangle = [\text{cov } \mathbf{m}]_A \mathbf{G}^T \{\mathbf{G}[\text{cov } \mathbf{m}]_A \mathbf{G}^T\}^{-1} \{\mathbf{d} - \mathbf{G}\langle \mathbf{m} \rangle\}. \quad (5.54)$$

Thus, the principle of maximum relative entropy, when applied to the underdetermined problem, yields the weighted minimum-length solution (compare Equations (5.54) with (3.43) when $\mathbf{W}_m^{-1} = [\text{cov } \mathbf{m}]_A$). Many of the other inverse theory solutions that were developed in this chapter using maximum likelihood techniques can also be derived using the MRE principle (Woodbury, 2011).

5.4 EQUIVALENCE OF THE THREE VIEWPOINTS

We can arrive at the same general solution to the linear inverse problem by three distinct routes.

Viewpoint 1. The solution is obtained by minimizing a weighted sum of L_2 prediction error and L_2 solution simplicity

$$\text{Minimize:} \quad \mathbf{e}^T \mathbf{W}_e \mathbf{e} + \varepsilon^2 [\mathbf{m} - \langle \mathbf{m} \rangle]^T \mathbf{W}_m [\mathbf{m} - \langle \mathbf{m} \rangle] \quad (5.55)$$

where ε^2 is a weighting factor.

Viewpoint 2. The solution is obtained by minimizing a weighted sum of three terms: the Dirichlet spreads of model resolution and data resolution and the size of the model covariance.

$$\text{Minimize:} \quad \alpha_1 \text{ spread}(\mathbf{R}) + \alpha_2 \text{ spread}(\mathbf{N}) + \alpha_3 \text{ size}([\text{cov}_u \mathbf{m}]) \quad (5.56)$$

Viewpoint 3. The solution is obtained by maximizing the likelihood of the joint Gaussian distribution of data, *a priori* model parameters, and theory.

$$\text{Maximize: } L = \log p_T(\mathbf{m}, \mathbf{d}) \quad (5.57)$$

These derivations emphasize the close relationship among the L_2 norm, the Dirichlet spread function, and the Gaussian probability density function.

5.5 THE F-TEST OF ERROR IMPROVEMENT SIGNIFICANCE

We sometimes have *two* candidate models for describing an overdetermined inverse problem, one of which is more complicated than the other (in the sense that it possesses a greater number of model parameters). Suppose that Model B is more complicated than Model A and that the total prediction error for Model B is less than the total prediction error for Model A: $E_B < E_A$. Does Model B really fit the data better than Model A?

The answer to this question depends on the variance of the data. Almost any complicated model will fit data better than a less complicated one. The relevant question is whether the fit is *significantly* better, that is, whether the improvement is too large to be accounted for by random fluctuations in the data. For statistical reasons that will be cited, we pretend, in this case, that the two inverse problems are solved with two different realizations of the data.

Suppose that we estimate the variance of the data d_i from the prediction error e_i of each model as

$$(\sigma_d^{\text{est}})^2 = \frac{1}{v} \sum_{i=1}^N e_i^2 = \frac{E}{v} \quad \text{with } v = N - M \quad (5.58)$$

This estimate will usually be larger than the true variance of the data, since it also includes a contribution from the (possibly) poor fit of the model. If one model fits the data about as well as the other, then the variance $(\sigma_{dA}^{\text{est}})^2$ estimated from Model A should be about the same as the variance $(\sigma_{dB}^{\text{est}})^2$ estimated from Model B. On the other hand, if Model B gives a better fit than Model A, the estimated variances will differ in such a way that the ratio $(\sigma_{dA}^{\text{est}})^2 / (\sigma_{dB}^{\text{est}})^2$ will be greater than unity. If the ratio is only slightly greater than unity, the difference in fit may be entirely a result of random fluctuations in the data and therefore may not be significant. Nevertheless, there is clearly some value for the ratio that indicates a significant difference between the two fits.

To compute this critical value, we consider the theoretical distribution for the ratio of two variance estimates derived from two different realizations of the *same* data set. Of course, the ratio of the true variance with itself always has the value unity; but the ratio of two estimates of the true variance will fluctuate randomly about unity. We therefore determine whether or not ratios greater than or equal to the observed ratio occur less than, say, 5% of the time. If they do, then there is a 95% probability that the two estimates are derived from data sets with different true variances. We are justified in concluding that the second model is a significant improvement over the first.

To handle data with nonuniform variance, we form a ratio, not of estimated variances, but of the related quantity

$$\chi_v^2 = \frac{v(\sigma_d^{\text{est}})^2}{(\sigma_d^{\text{true}})^2} = \frac{\sum_{i=1}^N e_i^2}{(\sigma_{di}^{\text{true}})^2} \quad (5.59)$$

This quantity is chosen because it has a χ_v^2 distribution with v degrees of freedom. The ratio of the χ_v^2 for the two models is given by

$$F(v_A, v_B) = \frac{\chi_{v_A}^2/v_A}{\chi_{v_B}^2/v_B} = \frac{(\sigma_{dA}^{\text{est}})^2/(\sigma_{dA}^{\text{true}})^2}{(\sigma_{dB}^{\text{est}})^2/(\sigma_{dB}^{\text{true}})^2} \quad (5.60)$$

Note that the true variance cancels out of the equation, as long as it is the same for the two models. Thus, the F ratio is not a function of the overall amplitude of the total error but only of the relative error between the two models. It is, however, a function of the number of degrees of freedom of the two models.

The probability density function of the F ratio is known, but it cannot be written in terms of elementary functions, so we omit it here. It is a unimodal distribution with mean and variance given by

$$\langle F \rangle = \frac{v_B}{v_B - 2} \quad \sigma_F^2 = \frac{2v_B^2(v_A + v_B - 2)}{v_A(v_B - 2)^2(v_B - 4)} \quad (5.61)$$

Note that for large degrees of freedom, $\langle F \rangle \approx 1$. Note, also, that F^{est} and $1/F^{\text{est}}$ play symmetrical roles, in the sense that the first quantifies the improvement of fit of Model B with respect to Model A, and the latter, the improvement of fit of Model A with respect to Model B. Thus, in testing the *Null Hypothesis* that any difference between the two estimated variances is due to random variation, we should compute the probability that F is smaller than $1/F^{\text{est}}$ or larger than F^{est} :

$$P\left(F < \frac{1}{F^{\text{est}}} \quad \text{or} \quad F > F^{\text{est}}\right) \quad (5.62)$$

The Null Hypothesis can be rejected if this probability is less than 5%. The *MatLab* function `fcdf()` computes the cumulative probability of F :

```
Fobs = (EA/vA) / (EB/vB);
if ( Fobs<1 )
    Fobs=1/Fobs;
end
P = 1 - (fcdf(Fobs, vA, vB) - fcdf(1/Fobs, vA, vB));
(MatLab script gda05_17)
```

Here E_A and E_B are E_A and E_B , respectively, and v_A and v_B are v_A and v_B , respectively. An example is shown in [Figure 5.17](#).

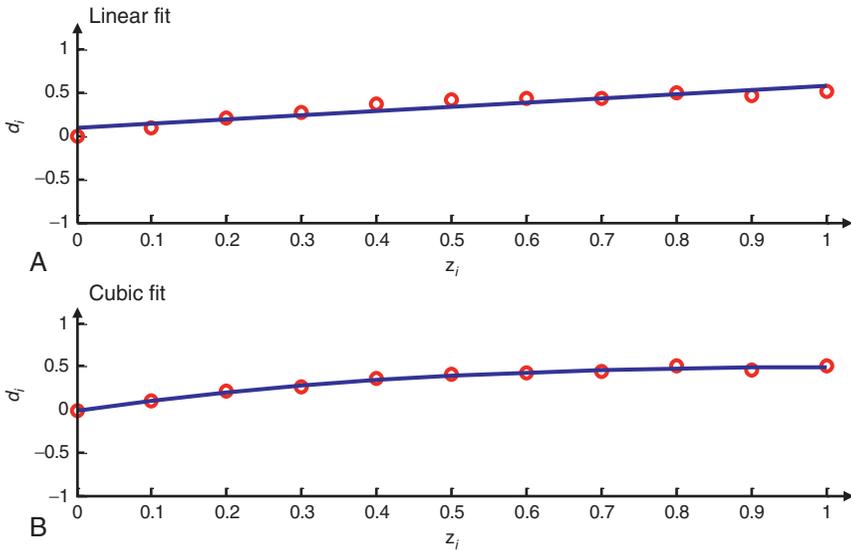


FIGURE 5.17 Hypothetical data set (red circles) fit (blue curve) with (A) a straight line and (B) a cubic polynomial. Although the cubic fit appears superior, an F -test reveals that this level of improvement of fit will be obtained 6.4% of the time under the Null Hypothesis that the improvement is due to random variation. The improvement of fit is not significant at the 95% level. *MatLab* script gda05_17.

5.6 PROBLEMS

- 5.1** Suppose that a random variable m is defined on the interval $[-1, 1]$. A reasonable choice for the null probability density function is $p_N(\mathbf{m}) = 1/2$, meaning that m can be anywhere within the interval with equal probability. (A) Calculate (either analytically or numerically) the information gain S of the *a priori* probability density function $p_A(\mathbf{m}) = 1/2 + cm$, where $0 < c < 1/2$. Note that the larger the constant c , the higher the probability that m will fall in the positive half of the interval. (B) Make and interpret a plot of $S(c)$.
- 5.2** Suppose that a Gaussian probability density function with mean $\langle m \rangle$ and variance σ_m^2 is used to represent *a priori* information about the following types of model parameters: (A) the density of sea water; (B) the shear velocity at 100-km depth in the earth; (C) the O^{18} to O^{16} ratio in glacial ice. Propose plausible values of $\langle m \rangle$ and σ_m^2 in each case, citing references that justify your values.
- 5.3** Suppose that you are fitting a cubic polynomial to data, $d_i = m_1 + m_2 z_i + m_3 z_i^2 + m_4 z_i^3$, but have *a priori* information that $m_1 = 2m_2 = 4m_3 = 8m_4$. Write a *MatLab* script to solve this problem using Equation (3.55). Set up the problem so that $N = 50$, $0 < z_i < 1$, and $m_1^{\text{true}} = 1$, and generate synthetic data with $\sigma_d^2 = (0.1)^2$. How well do the data alone (that is, without the *a priori* information) constrain the ratios between the m s?

- 5.4 This problem builds upon Problem 5.3. Suppose that you are fitting a cubic polynomial to data, $d_i = m_1 + m_2 z_i + m_3 z_i^2 + m_4 z_i^3$, but have *a priori* information that $m_1 = 2m_2 = 4m_3 = 8m_4$. Write a *MatLab* script to solve this problem using Equation (5.55). Use a range of values for the variance σ_m^2 of the *a priori* information, from very uncertain to very certain. Set up the problem so that $N = 50$, $0 < z_i < 1$, and $m_1^{\text{true}} = 1$ and generate synthetic data with $\sigma_d^2 = (0.1)^2$. How well do the data alone (that is, without the *a priori* information) constrain the ratios between the *ms*?
- 5.5 This problem builds upon Problems 5.3 and 5.4. Modify your script in Problem 5.4 in the case where the model of the data fitting a cubic polynomial is thought to be inexact, with a variance $\sigma_d^2 = (0.05)^2$. How does this modification change your results?
- 5.6 Write a *MatLab* function to empirically generate the $p(F)$ probability density function with $v_1 = v_2 = 20$ by (A) using the `random('Normal', ...)` function to generate batches of 20 Gaussian-distributed random numbers with zero mean and unit variance. (B) Calculating χ_{20}^2 by summing the squares of each batch. (C) Calculating F for pairs of batches. (D) Repeat many times, creating a histogram of the resulting F s, and normalize to unit area to produce an empirical estimate of $p(F)$. (E) Compare your result with *MatLab*'s `fpdf()` function.
- 5.7 This problem expands upon Problem 5.5. Suppose that the random numbers in step (A) are drawn from a uniform distribution with zero mean and unit variance, but that F is calculated the same as before (let us call it F'). (A) How different is $p(F')$ from $p(F)$? (B) How would treating data with uniformly distributed error as if they were Gaussian-distributed affect the results of an F -test? Hint: A distribution that is uniform between a and b has a variance of $(b - a)^2/12$.

REFERENCES

- Kapur, J.N., 1989. Maximum Entropy Models in Science and Engineering. Wiley, New York 636pp.
- Tarantola, A., Valette, B., 1982a. Generalized non-linear inverse problems solved using the least squares criterion. *Rev. Geophys. Space Phys.* 20, 219–232.
- Tarantola, A., Valette, B., 1982b. Inverse problems = quest for information. *J. Geophys.* 50, 159–170.
- Woodbury, A.D., 2011. Minimum relative entropy, Bayes and Kapur. *Geophys. J. Int.* 185, 181–189.

Nonuniqueness and Localized Averages

6.1 NULL VECTORS AND NONUNIQUENESS

In Chapters 3–5, we presented the basic method of finding estimates of the model parameters in a linear inverse problem. We showed that we could always obtain such estimates but that sometimes in order to do so we had to add *a priori* information to the problem. We shall now consider the meaning and consequences of nonuniqueness in linear inverse problems and show that it is possible to devise solutions that do not depend at all on *a priori* information. As we shall show, however, these solutions are not estimates of the model parameters themselves but estimates of *weighted averages* (linear combinations) of the model parameters.

When the linear inverse problem $\mathbf{G}\mathbf{m}=\mathbf{d}$ has nonunique solutions, there exist nontrivial solutions (that is, solutions with some nonzero m_i) to the homogeneous equation $\mathbf{G}\mathbf{m}=\mathbf{0}$. These solutions are called the *null vectors* of the inverse problem as premultiplying them by the data kernel yields zero. To see why nonuniqueness implies null vectors, suppose that the inverse problem has two distinct solutions $\mathbf{m}^{(1)}$ and $\mathbf{m}^{(2)}$ as

$$\begin{aligned}\mathbf{G}\mathbf{m}^{(1)} &= \mathbf{d} \\ \mathbf{G}\mathbf{m}^{(2)} &= \mathbf{d}\end{aligned}\quad (6.1)$$

Subtracting these two equations yields

$$\mathbf{G}(\mathbf{m}^{(1)} - \mathbf{m}^{(2)}) = \mathbf{0}\quad (6.2)$$

Since the two solutions are by assumption distinct, their difference $\mathbf{m}^{\text{null}} = \mathbf{m}^{(1)} - \mathbf{m}^{(2)}$ is nonzero. The converse is also true; any linear inverse problem that has null vectors is nonunique. Note that the equation $\mathbf{G}\mathbf{m}^{\text{null}} = \mathbf{0}$ can be interpreted to mean that \mathbf{m}^{null} is perpendicular to every row of \mathbf{G} (as its dot product with every row is zero). Consequently, no linear combination of the rows of \mathbf{G} can be a null vector. If \mathbf{m}^{par} (where par stands for “particular”) is any nonnull solution to $\mathbf{G}\mathbf{m}=\mathbf{d}$ (for instance, the minimum length solution), then $\mathbf{m}^{\text{par}} + \alpha\mathbf{m}^{\text{null}}$ is also a solution with the same error for any choice of α .

Note that as $\alpha \mathbf{m}^{\text{null}}$ is a null vector for any nonzero α , null vectors are only distinct if they are linearly independent. If a given inverse problem has q distinct null solutions, then the most general solution is

$$\mathbf{m}^{\text{gen}} = \mathbf{m}^{\text{par}} + \sum_{i=1}^q \alpha_i \mathbf{m}^{\text{null}(i)} \quad (6.3)$$

where gen stands for “general.” We shall show in [Section 7.6](#) that $0 \leq q \leq M$, that is, that there can be no more linearly independent null vectors than there are unknowns.

6.2 NULL VECTORS OF A SIMPLE INVERSE PROBLEM

As an example, consider the following very simple equations:

$$\mathbf{G}\mathbf{m} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = [d_1] \quad (6.4)$$

This equation implies that only the mean value of a set of four model parameters has been measured. One obvious solution to this equation is $\mathbf{m} = [d_1, d_1, d_1, d_1]^T$ (in fact, this is the minimum length solution).

Three linearly independent null solutions can be determined by inspection as

$$\mathbf{m}^{\text{null}(1)} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{m}^{\text{null}(2)} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \quad \mathbf{m}^{\text{null}(3)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \quad (6.5)$$

The most general solution is then

$$\mathbf{m}^{\text{gen}} = \begin{bmatrix} d_1 \\ d_1 \\ d_1 \\ d_1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \quad (6.6)$$

where the α s are arbitrary parameters.

Finding a particular solution to this problem now consists of choosing values for the parameters α_i . If one chooses these parameters so that $\|\mathbf{m}\|_2$ is minimized, one obtains the minimum length solution. Since the first vector is orthogonal to all the others, this minimum occurs when $\alpha_i = 0$, $i = 1, 2, 3$. We shall show in [Chapter 7](#) that this is a general result: the minimum length solution never contains any null vectors. Note, however, that if other definitions of solution simplicity are used (e.g., flatness or smoothness), those solutions will contain null vectors.

6.3 LOCALIZED AVERAGES OF MODEL PARAMETERS

In Chapters 3–5, we have sought to estimate the elements of the solution vector \mathbf{m} . Another approach is to estimate some average of the model parameter $\langle m \rangle = \mathbf{a}^T \mathbf{m}$, where \mathbf{a} is some averaging vector. The average is said to be localized if this averaging vector consists mostly of zeros (except for some group of nonzero elements that multiplies model parameters centered about one particular model parameter). This definition makes particular sense when the model parameters possess some natural ordering in space and time, such as acoustic velocity as a function of depth in the earth. For instance, if $M=8$, the averaging vector $\mathbf{a} = [0, 0, 1/4, 1/2, 1/4, 0, 0, 0]^T$ could be said to be localized about the fourth model parameter. The averaging vectors are usually normalized so that the sum of their elements is unity.

The advantage of estimating averages of the model parameters rather than the model parameters themselves is that quite often it is possible to identify unique averages even when the model parameters themselves are not unique. To examine when uniqueness can occur, we compute the average of the general solution as

$$\langle m \rangle = \mathbf{a}^T \mathbf{m}^{\text{gen}} = \mathbf{a}^T \mathbf{m}^{\text{par}} + \sum_{i=1}^q \alpha_i \mathbf{a}^T \mathbf{m}^{\text{null}(i)} \quad (6.7)$$

If $\mathbf{a}^T \mathbf{m}^{\text{null}(i)}$ is zero for all i , then $\langle m \rangle$ is unique. The process of averaging has completely removed the nonuniqueness of the problem. Since \mathbf{a} has M elements and there are $q \leq M$ constraints placed on \mathbf{a} , one can always find at least one vector that cancels (or “annihilates”) the null vectors. One cannot, however, always guarantee that the averaging vector is localized around some particular model parameter. But, if $q < M$, one has some freedom in choosing \mathbf{a} and there is some possibility of making the averaging vector at least somewhat localized. Whether this can be done depends on the structure of the null vectors, which, in turn, depends on the structure of the data kernel \mathbf{G} . Since the small-scale features of the model are unresolvable in many problems, unique localized averages can often be found.

6.4 RELATIONSHIP TO THE RESOLUTION MATRIX

During the discussion of the resolution matrix \mathbf{R} (Section 4.3), we encountered in a somewhat different form the problem of determining averaging vectors. We showed that any estimate \mathbf{m}^{est} computed from a generalized inverse \mathbf{G}^{-g} was related to the true model parameters by

$$\mathbf{m}^{\text{est}} = \mathbf{R} \mathbf{m}^{\text{true}} = \mathbf{G}^{-g} \mathbf{G} \mathbf{m}^{\text{true}} \quad \text{or} \quad m_i^{\text{est}} = \sum_{j=1}^M R_{ij} m_j^{\text{true}} = \sum_{j=1}^M \sum_{k=1}^N G_{ik}^{-g} G_{kj} m_j^{\text{true}} \quad (6.8)$$

The i th row of \mathbf{R} (or rather its transpose) can be interpreted as a unique averaging vector \mathbf{a} that is centered about m_i , with the averaging vector \mathbf{a} being built up from linear combinations of the rows of the data kernel \mathbf{G}

$$m_i^{\text{est}} = \langle m \rangle^{(i)} = \sum_{j=1}^M a_j^{(i)} m_j^{\text{true}} \quad \text{with} \quad a_j^{(i)} = \sum_{k=1}^N c_k^{(i)} G_{kj} \quad \text{and} \quad c_k^{(i)} = G_{ik}^{-g} \quad (6.9)$$

Note that the resolution matrix in Equation (6.8) is composed of the product of the generalized inverse and the data kernel. We can interpret this product as meaning that a row of the resolution matrix is composed of a weighted sum of the rows of the data kernel \mathbf{G} (where the elements of the generalized inverse are the weighting factors $c_k^{(i)} = G_{ik}^{-g}$) regardless of the generalized inverse's particular form. An averaging vector \mathbf{a} produces a unique average $\langle m \rangle$ if and only if \mathbf{a}^T can be represented as a linear combination of the rows of the data kernel \mathbf{G} , since the rows of \mathbf{G} are guaranteed to be perpendicular to every null vector.

Whether or not the average is truly localized depends on the structure of \mathbf{R} . The spread function discussed previously in Section 4.6 is a measure of the degree of localization.

The process of forming the generalized inverse is equivalent to “shuffling” the rows of the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ by forming linear combinations until the data kernel is as close as possible to an identity matrix. Each row of the data kernel can then be viewed as a localized averaging vector, and each corresponding row of the shuffled data vector is the estimated value of the average.

6.5 AVERAGES VERSUS ESTIMATES

We can, therefore, identify a type of dualism in inverse theory. Given a generalized inverse \mathbf{G}^{-g} that in some sense solves $\mathbf{G}\mathbf{m} = \mathbf{d}$, we can speak either of estimates of model parameters $\mathbf{m}^{\text{est}} = \mathbf{G}^{-g}\mathbf{d}$ or of localized averages $\langle \mathbf{m} \rangle = \mathbf{G}^{-g}\mathbf{d}$. The numerical values are the same but the interpretation is quite different. When the solution is interpreted as a localized average, it can be viewed as a unique quantity that exists independently of any *a priori* information applied to the inverse problem. Examination of the resolution matrix may reveal that the average is not especially localized and the solution may be difficult to interpret. When the solution is viewed as an estimate of a model parameter, the location of what is being solved for is clear. The estimate can be viewed as unique only if one accepts as appropriate whatever *a priori* information was used to remove the inverse problem's underdeterminacy. In most instances, the choice of *a priori* information is somewhat *ad hoc*, so the solution may still be difficult to interpret.

In the sample problem stated above, the data kernel has only one row. There is therefore only one averaging vector that will annihilate all the null vectors: one proportional to that row

$$\mathbf{a} = \left[\frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \right]^T \quad (6.10)$$

This averaging vector is clearly unlocalized. In this problem, the structure of \mathbf{G} is just too poor to form good averages. The generalized inverse to this problem is by inspection

$$\mathbf{G}^{-g} = [1 \ 1 \ 1 \ 1]^T \quad (6.11)$$

The resolution matrix is therefore

$$\mathbf{R} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (6.12)$$

which is very unlocalized and equivalent to Equation (6.8).

6.6 NONUNIQUE AVERAGING VECTORS AND A PRIORI INFORMATION

There are instances in which even nonunique averages of model parameters can be of value, especially when they are used in conjunction with other *a priori* knowledge of the nature of the solution (Wunsch and Minster, 1982). Suppose that one simply picks a localized averaging vector that does not necessarily annihilate all the null vectors and that, therefore, does not lead to a unique average. In the above problem, the vector $\mathbf{a} = (1/3)[1, 1, 1, 0]^T$ might be such a vector. It is somewhat localized, being centered about the second model parameter. Note that it does not lead to a unique average, since

$$\langle m \rangle = \mathbf{a}^T \mathbf{m}^{\text{gen}} = d_1 + 0 + 0 + \frac{1}{3} \alpha_3 \quad (6.13)$$

is still a function of one of the arbitrary parameters α_i . Suppose, however, that there is *a priori* knowledge that every m_i must satisfy $0 \leq m_i \leq 2d_1$. Then from the equation for \mathbf{m}^{gen} , α_3 must be no greater than d_1 and no less than $-d_1$. Since $-d_1 \leq \alpha_3 \leq d_1$, the average has bounds $(2/3)d_1 \leq \langle m \rangle \leq (4/3)d_1$. These constraints are considerably tighter than the *a priori* bounds on m_i , which demonstrates that this technique has indeed produced some useful information. This approach works because even though the averaging vector does not annihilate all the null vectors, $\mathbf{a}^T \mathbf{m}^{\text{null}(i)}$ is small compared with the elements of the null vector. Localized averaging vectors often lead to small products since the null vectors often fluctuate rapidly about zero, indicating that small-scale features of the model are the most poorly resolved. A slightly more complicated example of this type is solved in Figure 6.1.

This approach can be generalized as follows (Oldenberg, 1983):

$$\begin{aligned} &\text{maximize/minimize } \langle m \rangle = \mathbf{a}^T \mathbf{m} && \text{with respect to } \mathbf{m} \\ &\text{with the constraints } \mathbf{G}\mathbf{m} = \mathbf{d}^{\text{obs}} && \text{and } \mathbf{m}^{(l)} \leq \mathbf{m} \leq \mathbf{m}^{(u)} \end{aligned} \quad (6.14)$$

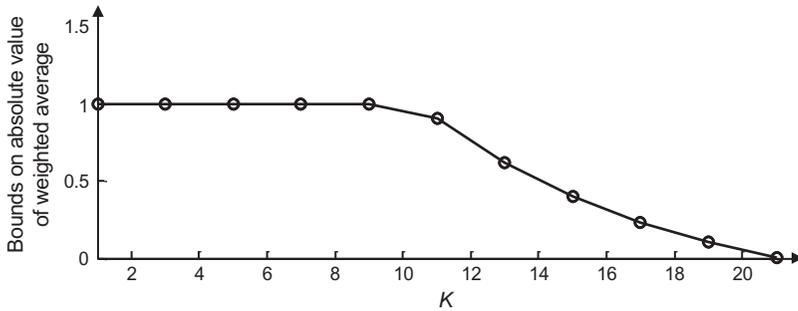


FIGURE 6.1 Bounds on weighted averages of model parameters, m_i , in a problem in which the only datum is that the sum of all model parameters is zero. When this observation is combined with the *a priori* information that each model parameter must satisfy $|m_i| \leq 1$, bounds can be placed on the weighted averages of the model parameter. The bounds shown here are for averages of K neighboring model parameters. Note that the bounds are tighter than the *a priori* bounds only when $K > 10$. *MatLab* script gda06_01.

Here, \mathbf{m}^l and \mathbf{m}^u are the *a priori* lower and upper bounds on \mathbf{m} , respectively. Note that this formulation does not explicitly include null vectors; the constraint $\mathbf{G}\mathbf{m} = \mathbf{d}^{\text{obs}}$ is sufficient to ensure that the model parameters satisfy the data. This problem is a special case of the *linear programming problem*

$$\begin{aligned} &\text{find } \mathbf{x} \text{ that minimizes } z = \mathbf{f}^T \mathbf{x} \\ &\text{with the constraints } \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{C}\mathbf{x} = \mathbf{d} \text{ and } \mathbf{x}^{(l)} \leq \mathbf{x} \leq \mathbf{x}^{(u)} \end{aligned} \quad (6.15)$$

Note that the minimization problem can be converted into a maximization problem by flipping the sign of \mathbf{f} . Furthermore, “ \geq type” inequality constraints can be converted to “ \leq type” by multiplication by -1 .

The linear programming problem was first studied by economists and business operations analysts. For example, z might represent the profit realized by a factory producing a product line, where the number of each product is given by \mathbf{x} and the profit on each item given by \mathbf{f} . The problem is to maximize the total profit $\mathbf{f}^T \mathbf{x}$ without violating the constraint that one can produce only a positive amount of each product, or any other linear inequality constraints that might represent labor laws, union regulations, physical limitation of machines, etc.

MatLab provides a `linprog()` function for solving the linear programming problem. Equation (6.13) is solved by calling it twice, once to minimize $\langle m \rangle$ and the other to maximize it

```
[mest1, amin]=linprog(a, [], [], G, dobs, mlb, mub);
[mest2, amax]=linprog(-a, [], [], G, dobs, mlb, mub);
amax = -amax;
```

(*MatLab* script gsa06_01)

Here, `amin` and `amax` are its lower and upper bounds on $\langle m \rangle$, respectively. While the corresponding model parameter vectors `mest1` and `mest2` are also

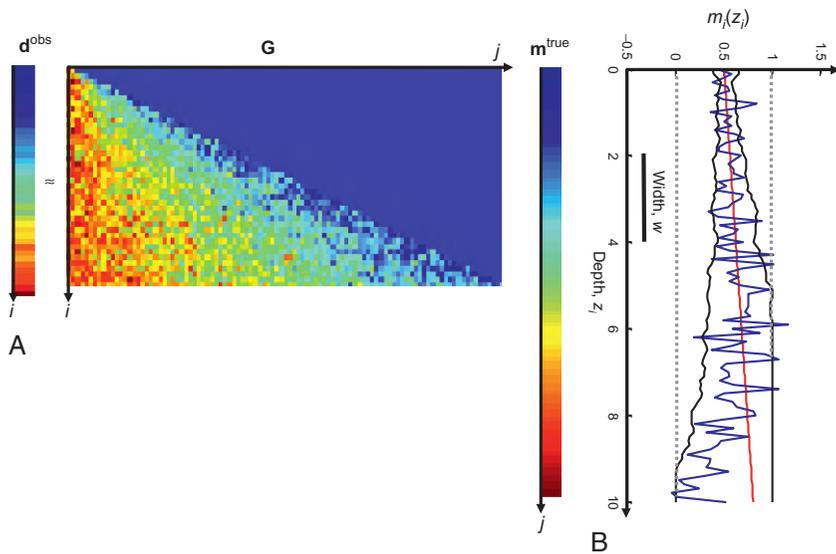


FIGURE 6.2 (A) This underdetermined inverse problem, $\mathbf{d} = \mathbf{G}\mathbf{m}$, has $M = 100$ model parameters m_i and $N = 40$ data d_i . The data are weighted averages of the model parameters, from the surface down to a depth, z , that increases with index, i . The observed data d_i^{est} include additive noise. (B) The true model parameters (red curve) increase linearly with depth z . The estimated model parameters (blue curve), computed using the minimum length method, scatter about the true model at shallow depths ($z < 6$) but decline toward zero at deeper depths due to poor resolution. Bounds on localized averages of the model parameters, with an averaging width, $w = 2$ (black curves), are for *a priori* information, $0 < m_i < 1$ (gray dotted lines). *MatLab* script `gda06_02`.

calculated, they are not usually of interest. The arguments of `linprog()` include the averaging vector \mathbf{a} , the data kernel \mathbf{G} , the observed data \mathbf{d}_{obs} , and the *a priori* upper and lower bounds m_{lb} and m_{ub} on the model parameters. An example using a Laplace transform-like data kernel is shown in Figure 6.2.

6.7 PROBLEMS

6.1. What is the general solution to the problem $\mathbf{G}\mathbf{m} = \mathbf{d}$, with

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

6.2. Give some examples of physical problems where the model parameters can be assumed, with reasonable certainty, to fall between lower and upper bounds (and identify the bounds).

6.3. Suppose that $M = 21$, model parameters are known to have bounds $-1 \leq m_i \leq 1$. Suppose that the unweighted average of each three adjacent model parameters are observed so that the data kernel has the form

$$\mathbf{G} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Can useful bounds be placed on the weighted average of the three adjacent model parameters, where the weights are $[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$? Adapt *MatLab* script `gda06_01` to explore this problem.

REFERENCES

- Oldenburg, D.W., 1983. Funnel functions in linear and nonlinear appraisal. *J. Geophys. Res.* 88, 7387–7398.
- Wunsch, C., Minster, J.F., 1982. Methods for box models and ocean circulation tracers: mathematical programming and non-linear inverse theory. *J. Geophys. Res.* 87, 5647–5662.

Applications of Vector Spaces

7.1 MODEL AND DATA SPACES

So far, we have used vector notation for the data \mathbf{d} and model parameters \mathbf{m} mainly because it facilitates the algebra. The interpretations of vectors as geometrical quantities can also be used to gain an insight into the properties of inverse problems. We therefore introduce the idea of vector spaces containing \mathbf{d} and \mathbf{m} , which we shall denote $S(\mathbf{d})$ and $S(\mathbf{m})$. Any particular choice of \mathbf{m} and \mathbf{d} is then represented as a vector in these spaces (Figure 7.1).

The linear equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ can be interpreted as a mapping of vectors from $S(\mathbf{m})$ to $S(\mathbf{d})$ and its solution $\mathbf{m}^{\text{est}} = \mathbf{G}^{-\text{g}}\mathbf{d}$ as a mapping of vectors from $S(\mathbf{d})$ to $S(\mathbf{m})$.

One important property of a vector space, such as $S(\mathbf{m})$, is that its coordinate axes are arbitrary. Thus far we have been using axes parallel to the individual model parameters, but we recognize that we are by no means required to do so. Any set of vectors that spans the space will serve as coordinate axes. The M th dimensional space $S(\mathbf{m})$ is spanned by any M vectors, say, $\mathbf{m}^{(i)}$, as long as these vectors are linearly independent. An arbitrary vector lying in $S(\mathbf{m})$, say \mathbf{m}^* , can be expressed as a sum of these M basis vectors, written as

$$\mathbf{m}^* = \sum_{i=1}^M \alpha_i \mathbf{m}^{(i)} \quad (7.1)$$

where the α s are the components of the vector \mathbf{m}^* in the new coordinate system. If the $\mathbf{m}^{(i)}$ s are linearly dependent, then the vectors $\mathbf{m}^{(i)}$ lie in a subspace, or *hyperplane*, of $S(\mathbf{m})$ and the expansion cannot be made (Figure 7.2).

We shall consider, therefore, transformations of the coordinate systems of the two spaces $S(\mathbf{m})$ and $S(\mathbf{d})$. Using $S(\mathbf{m})$ as an example, if \mathbf{m} is the representation of a vector in one coordinate system and \mathbf{m}' its representation in another, we can write the transformation as

$$\mathbf{m}' = \mathbf{T}\mathbf{m} \quad \text{and} \quad \mathbf{m} = \mathbf{T}^{-1}\mathbf{m}' \quad (7.2)$$

where \mathbf{T} is the transformation matrix. If the new basis vectors are still mutually orthogonal unit vectors, then \mathbf{T} represents simple rotations or reflections of the coordinate axes. As we shall show below, however, it is sometimes convenient to choose a new set of basis vectors that are not unit vectors.

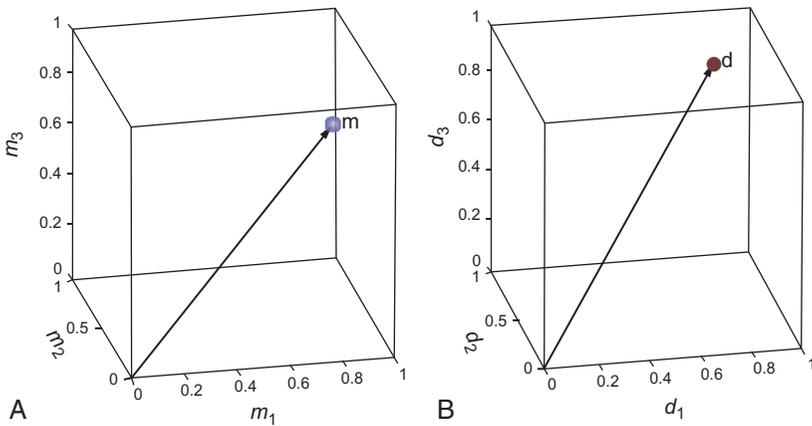


FIGURE 7.1 (A) The model parameters represented as a vector \mathbf{m} in the M -dimensional space $S(\mathbf{m})$ of all possible model parameters. (B) The data represented as a vector \mathbf{d} in the N -dimensional space $S(\mathbf{d})$ of all possible data. *MatLab* script gda07_01.

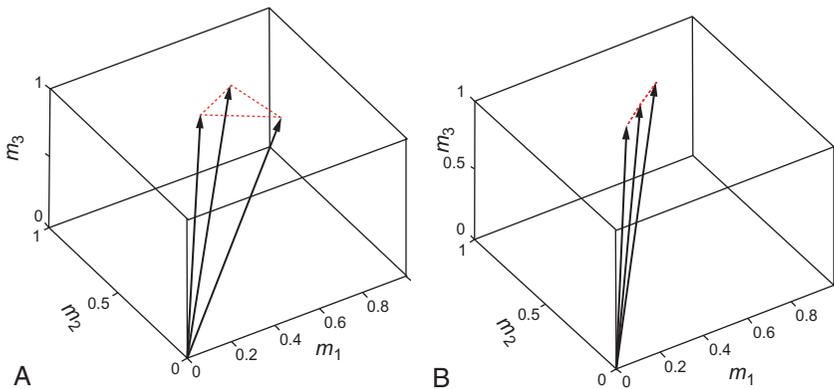


FIGURE 7.2 (A) These three vectors span the three-dimensional space $S(\mathbf{m})$. (B) These three vectors do not span the space as they all lie on the same plane. *MatLab* script gda07_02.

7.2 HOUSEHOLDER TRANSFORMATIONS

In this section, we shall show that the minimum length, least squares, and constrained least squares solutions can be found through simple transformations of the equation $\mathbf{G}\mathbf{m} = \mathbf{d}$. While the results are identical to the formulas derived in Chapter 3, the approach and emphasis are quite different and will enable us to visualize these solutions in a new way. We shall begin by considering a purely underdetermined linear problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ with $M > N$. Suppose we want to find the minimum-length solution (the one that minimizes $L = \mathbf{m}^T \mathbf{m}$). We shall show that it is easy to find this solution by transforming the model parameters into a new coordinate system $\mathbf{m}' = \mathbf{T}\mathbf{m}$. The inverse problem becomes

$$\mathbf{d} = \mathbf{G}\mathbf{m} = \mathbf{G}\mathbf{I}\mathbf{m} = \{\mathbf{G}\mathbf{T}^{-1}\}\{\mathbf{T}\mathbf{m}\} = \mathbf{G}'\mathbf{m}' \quad (7.3)$$

where $\mathbf{G}' = \mathbf{G}\mathbf{T}^{-1}$ is the data kernel in the new coordinate system. The solution length becomes

$$L = \mathbf{m}^T\mathbf{m} = \{\mathbf{T}^{-1}\mathbf{m}'\}^T\{\mathbf{T}^{-1}\mathbf{m}'\} = \mathbf{m}'^T\{\mathbf{T}^{-1T}\mathbf{T}^{-1}\}\mathbf{m}' \quad (7.4)$$

Suppose that we could choose \mathbf{T} so that $\{\mathbf{T}^{-1T}\mathbf{T}^{-1}\} = \mathbf{I}$. The solution length would then have the same form in both coordinate systems, namely, the sum of squares of the vector elements. Minimizing $\mathbf{m}'^T\mathbf{m}'$ would be equivalent to minimizing $\mathbf{m}^T\mathbf{m}$. Transformations of this type that do not change the length of the vector components are called *unitary transformations*. They may be interpreted as rotations and reflections of the coordinate axes. We can see from Equation (7.4) that unitary transformations satisfy $\mathbf{T}^T = \mathbf{T}^{-1}$.

Now suppose that we could also choose the transformation so that \mathbf{G}' is the lower triangular matrix

$$\begin{bmatrix} G'_{11} & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ G'_{21} & G'_{22} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ G'_{31} & G'_{32} & G'_{33} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & & & & \\ G'_{N1} & G'_{N2} & G'_{N3} & G'_{N4} & \cdots & G'_{NN} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} m'_1 \\ m'_2 \\ m'_3 \\ \vdots \\ m'_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_N \end{bmatrix} \quad (7.5)$$

Notice that no matter what values we pick for $\{m_i^{\text{est}}, i = N + 1, M\}$, we cannot change the value of $\mathbf{G}'\mathbf{m}'$ as the last $M - N$ columns of \mathbf{G}' are all zero. On the other hand, we can solve for the first N elements of \mathbf{m}^{est} uniquely as

$$\begin{aligned} m_1^{\text{est}} &= [d_1]/G'_{11} \\ m_2^{\text{est}} &= [d_2 - G'_{21}m_1^{\text{est}}]/G'_{22} \\ m_3^{\text{est}} &= [d_3 - G'_{31}m_1^{\text{est}} - G'_{32}m_2^{\text{est}}]/G'_{33} \\ &\vdots \end{aligned} \quad (7.6)$$

This process is known as *back-solving*. As the first N elements of \mathbf{m}^{est} are thereby determined, $\mathbf{m}'^T\mathbf{m}'$ can be minimized by setting the remaining m_i^{est} equal to zero. The solution in the original coordinate system is then $\mathbf{m}^{\text{est}} = \mathbf{T}^{-1}\mathbf{m}'^{\text{est}}$. As this solution satisfies the data exactly and minimizes the solution length, it is equal to the minimum-length solution (Section 3.7).

We have employed a transformation process that separates the determined and undetermined linear combinations of model parameters into two distinct groups so that we can deal with them separately. It is interesting to note that we can now easily determine the null vectors for the inverse problem. In the transformed coordinates they are the set of vectors whose first N elements are zero and whose last $M - N$ elements are zero except for one element. There are clearly $M - N$ such vectors, so we have established that there are never more

than M null vectors in a purely underdetermined problem. The null vectors can easily be transformed back into the original coordinate system by premultiplication by \mathbf{T}^{-1} . As all but one element of the transformed null vectors are zero, this operation just selects a column of \mathbf{T}^{-1} (or, equivalently, a row of \mathbf{T}).

One transformation that can triangularize a matrix is called a *Householder transformation*. We shall discuss in Section 7.3 how it can be constructed. As we shall see below, Householder transformations have application to a wide variety of methods that employ the L_2 norm as a measure of size. These transformations provide an alternative method of solving such problems and additional insight into their structure.

The overdetermined linear inverse problem $\mathbf{Gm} = \mathbf{d}$ with $N > M$ can also be solved through the use of Householder transformations. In this case, we seek a solution that minimizes the prediction error $E = \mathbf{e}^T \mathbf{e}$. We seek a transformation with two properties: it must operate on the transformed prediction error $\mathbf{e}' = \mathbf{T}\mathbf{e}$ in such a way that minimizing $\mathbf{e}'^T \mathbf{e}'$ is the same as minimizing $\mathbf{e}^T \mathbf{e}$, and it must transform the data kernel into the upper-triangularized form. The transformed prediction error is

$$\mathbf{e}' = \mathbf{T}\mathbf{e} = \mathbf{T}\{\mathbf{d} - \mathbf{Gm}\} = \mathbf{Td} - \mathbf{TGm} = \mathbf{d}' - \mathbf{G}'\mathbf{m} \quad (7.7)$$

where \mathbf{d}' is the transformed data and \mathbf{G}' is the transformed and triangularized data kernel

$$\begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \\ \vdots \\ e'_M \\ e'_{M+1} \\ \vdots \\ e'_N \end{bmatrix} = - \begin{bmatrix} G'_{11} & G'_{12} & G'_{13} & G'_{14} & \cdots & G'_{1M} \\ 0 & G'_{22} & G'_{23} & G'_{24} & \cdots & G'_{2M} \\ 0 & 0 & G'_{33} & G'_{34} & \cdots & G'_{3M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & G'_{MM} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_M \end{bmatrix} + \begin{bmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d'_M \\ d'_{M+1} \\ \vdots \\ d'_N \end{bmatrix} \quad (7.8)$$

We note that no matter what values we choose for \mathbf{m}^{est} , we cannot alter the last $N - M$ elements of \mathbf{e}' as the last $N - M$ rows of the transformed data kernel are zero. We can, however, set the first M elements of \mathbf{e}' equal to zero by satisfying the first M equations $\mathbf{e}' = \mathbf{d}' - \mathbf{G}'\mathbf{m} = 0$ exactly. As the top part of \mathbf{G}' is triangular, we can use the back-solving technique described above. The total error is then the length of the last $N - M$ elements of \mathbf{e}' , written as

$$E = \sum_{i=M+1}^N \mathbf{e}_i'^2 \quad (7.9)$$

Again we use the Householder transformations to separate the problem into two parts: data that can be satisfied exactly and data that cannot be satisfied at all. The solution is chosen so that it minimizes the length of the prediction error, and the least square solution is thereby obtained.

Finally, we note that the constrained least squares problem can also be solved with Householder transformations. Suppose that we want to solve $\mathbf{G}\mathbf{m} = \mathbf{d}$ in the least squares sense except that we want the solution to obey p linear equality constraints of the form $\mathbf{H}\mathbf{m} = \mathbf{h}$. Because of the constraints, we do not have complete freedom in choosing the model parameters. We therefore employ Householder transformations to separate those linear combinations of \mathbf{m} that are completely determined by the constraints from those that are completely undetermined. This process is precisely the same as the one used in the underdetermined problem and consists of finding a transformation, say, \mathbf{T} , that triangularizes $\mathbf{H}\mathbf{m} = \mathbf{h}$ as

$$\mathbf{h} = \mathbf{H}\mathbf{m} = \{\mathbf{H}\mathbf{T}^{-1}\}\{\mathbf{T}\mathbf{m}\} = \mathbf{H}'\mathbf{m}' \quad (7.10)$$

The first p elements of \mathbf{m}'^{est} are now completely determined and can be computed by back-solving the triangular system. The same transformation can be applied to $\mathbf{G}\mathbf{m} = \mathbf{d}$ to yield the transformed inverse problem $\mathbf{d} = \mathbf{G}\mathbf{m} = \{\mathbf{G}\mathbf{T}^{-1}\}\{\mathbf{T}\mathbf{m}\} = \mathbf{G}'\mathbf{m}'$. But \mathbf{G}' will not be triangular as the transformation was designed to triangularize \mathbf{H} , not \mathbf{G} . As the first p elements of \mathbf{m}'^{est} have been determined by the constraints, we can partition \mathbf{G}' into two submatrices $\mathbf{G}' = [\mathbf{G}'_1, \mathbf{G}'_2]$, where \mathbf{G}'_1 multiplies the p -determined model parameters and \mathbf{G}'_2 multiplies the as yet unknown $M - p$ model parameters

$$[\mathbf{G}'_1, \mathbf{G}'_2] \left[\left[m'_1{}^{\text{est}} \cdots m'_p{}^{\text{est}} \right], \left[m'_{p+1}{}^{\text{est}} \cdots m'_M{}^{\text{est}} \right] \right]^T = \mathbf{d} \quad (7.11)$$

The equation can be rearranged into standard form by subtracting the part involving the already-determined model parameters:

$$\mathbf{G}'_2 \left[m'_{p+1}{}^{\text{est}} \cdots m'_M{}^{\text{est}} \right]^T = \mathbf{d} - \mathbf{G}'_1 \left[m'_1{}^{\text{est}} \cdots m'_p{}^{\text{est}} \right]^T \quad (7.12)$$

The equation is now a completely overdetermined one in the $M - p$ unknown model parameters and can be solved as described above. Finally, the solution is transformed back into the original coordinate system by $\mathbf{m}^{\text{est}} = \mathbf{T}^{-1}\mathbf{m}'^{\text{est}}$.

7.3 DESIGNING HOUSEHOLDER TRANSFORMATIONS

For a transformation to preserve length, it must be a unitary transformation (i.e., it must satisfy $\mathbf{T}^T = \mathbf{T}^{-1}$). Any transformation of the form

$$\mathbf{T} = \mathbf{I} - \frac{2\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} \quad (7.13)$$

(where \mathbf{v} is any vector) is a unitary transformation as

$$\mathbf{T}^{-1}\mathbf{T} = \mathbf{T}^T\mathbf{T} = \left[\mathbf{I} - \frac{2\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} \right]^2 = \mathbf{I} - \frac{4\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} + \frac{4\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} = \mathbf{I} \quad (7.14)$$

This can be shown to be the most general form of a unitary transformation. The problem is to find the vector \mathbf{v} such that the transformation triangularizes a given matrix. To do so, we shall begin by finding a sequence of transformations, each of which converts to zeros either the elements beneath the main diagonal of one *column* of the matrix (for premultiplication of the transformation) or the elements to the right of the main diagonal of one *row* of the matrix (for postmultiplication by the transformation). The first i columns are converted to zeros by

$$\mathbf{T} = \mathbf{T}^{(i)}\mathbf{T}^{(i-1)}\mathbf{T}^{(i-2)}\dots\mathbf{T}^{(1)} \quad (7.15)$$

In the overdetermined problem, applying M of these transformations produces an upper-triangularized matrix. In the $M=3, N=4$ case, the transformations proceed as

$$\begin{array}{ccccccc} \mathbf{G} & \rightarrow & \mathbf{T}^{(1)}\mathbf{G} & \rightarrow & \mathbf{T}^{(2)}\mathbf{T}^{(1)}\mathbf{G} & \rightarrow & \mathbf{T}^{(3)}\mathbf{T}^{(2)}\mathbf{T}^{(1)}\mathbf{G} \\ \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \rightarrow & \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} & \rightarrow & \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} & \rightarrow & \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{bmatrix} \end{array} \quad (7.16)$$

where the x s symbolize nonzero matrix elements. Consider the i th column of \mathbf{G} , denoted by the vector $\mathbf{g} = [G_{1i}, G_{2i}, \dots, G_{Ni}]^T$. We want to construct a transformation such that

$$\mathbf{g}' = \mathbf{T}^{(i)}\mathbf{g} = [G'_{1i}, G'_{2i}, \dots, G'_{ii}, 0, 0, \dots, 0]^T \quad (7.17)$$

Substituting the expression for the transformation yields

$$\mathbf{I}\mathbf{g} - \mathbf{I} - \left(\frac{2\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} \right) \mathbf{g} = \begin{bmatrix} G_{1i} \\ G_{2i} \\ \vdots \\ G_{Ni} \end{bmatrix} - \frac{2\mathbf{v}^T\mathbf{g}}{\mathbf{v}^T\mathbf{v}} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} G'_{1i} \\ G'_{2i} \\ \vdots \\ G'_{ii} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (7.18)$$

As the term $2\mathbf{v}^T\mathbf{g}/\mathbf{v}^T\mathbf{v}$ is a scalar, we can only zero the last $N-i$ elements of \mathbf{g}' if $[2\mathbf{v}^T\mathbf{g}/\mathbf{v}^T\mathbf{v}][v_{i+1}, \dots, v_N] = [G_{i+1,i}, \dots, G_{Ni}]$. We therefore set $[v_{i+1}, \dots, v_N] = [G_{i+1,i}, \dots, G_{Ni}]$ and choose the other i elements of \mathbf{v} so that the normalization is correct. As this is but a single constraint on i elements of \mathbf{v} , we have considerable flexibility in making the choice. It is convenient to choose the first $i-1$ elements of \mathbf{v} as zero (this choice is both simple and causes the transformation to leave the first $i-1$ elements of \mathbf{g} unchanged). This leaves the i th element of \mathbf{v} to be determined by the constraint. It is easy to show that $v_i = G_{ii} - \alpha_i$, where

$$\alpha_i^2 = \sum_{j=1}^N G_{ji}^2 \quad (7.19)$$

(Although the sign of α_i is arbitrary, the choice is usually better than the other in terms of the numerical stability of computer algorithms.) The Householder transformation is then defined by the vector as

$$\mathbf{v} = [0 \ 0 \ \cdots \ 0 \ (G_{ii} - \alpha_i) \ G_{i+1,i} \ G_{i+2,i} \ \cdots \ G_{Ni}]^T \quad (7.20)$$

Finally, we note that the $(i + 1)$ st Householder transformation does not destroy any of the zeros created by the i th, as long as we apply them in order of decreasing number of zeros. We can thus apply a succession of these transformations to triangularize an arbitrary matrix. The inverse transformations are also trivial to construct as they are just the transforms of the forward transformations.

7.4 TRANSFORMATIONS THAT DO NOT PRESERVE LENGTH

Suppose we want to solve the linear inverse problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ in the sense of finding a solution \mathbf{m}^{est} that minimizes a weighted combination of prediction error and solution simplicity as

$$\text{minimize: } E + L = \mathbf{e}^T \mathbf{W}_e \mathbf{e} + \mathbf{m}^T \mathbf{W}_m \mathbf{m} \quad (7.21)$$

It is possible to find transformations $\mathbf{m}' = \mathbf{T}_m \mathbf{m}$ and $\mathbf{e}' = \mathbf{T}_e \mathbf{e}$ which, although they do not preserve the length, nevertheless change it in precisely such a way that $E + L = \mathbf{e}'^T \mathbf{e}' + \mathbf{m}'^T \mathbf{m}'$ (Wiggins, 1972). The weighting factors are identity matrices in the new coordinate system.

Consider the weighted measure of length $L = \mathbf{m}^T \mathbf{W}_m \mathbf{m}$. If we could factor the weighting matrix into the product $\mathbf{W}_m = \mathbf{T}_m^T \mathbf{T}_m$, where \mathbf{T}_m is some matrix, then we could identify \mathbf{T}_m as a transformation of coordinates

$$L = \mathbf{m}^T \mathbf{W}_m \mathbf{m} = \mathbf{m}^T \{ \mathbf{T}_m^T \mathbf{T}_m \} \mathbf{m} = \{ \mathbf{T}_m \mathbf{m} \}^T \{ \mathbf{T}_m \mathbf{m} \} = \mathbf{m}'^T \mathbf{m}' \quad (7.22)$$

This factorization can be accomplished in several ways. If we have built \mathbf{W}_m up from the \mathbf{D} matrix, then $\mathbf{W}_m = \mathbf{D}^T \mathbf{D}$ and $\mathbf{T}_m = \mathbf{D}$. If not, then we can rely upon the fact that \mathbf{W}_m , being symmetric, has a symmetric square root, so that $\mathbf{W}_m = \mathbf{W}_m^{1/2} \mathbf{W}_m^{1/2} = \mathbf{W}_m^{1/2T} \mathbf{W}_m^{1/2}$, in which case $\mathbf{T} = \mathbf{W}_m^{1/2}$. Then

$$\begin{aligned} \mathbf{m}' &= \mathbf{W}_m^{1/2} \mathbf{m} \text{ or } \mathbf{m}' = \mathbf{D} \mathbf{m} & \mathbf{m} &= \mathbf{W}_m^{-1/2} \mathbf{m}' \text{ or } \mathbf{m} = \mathbf{D}^{-1} \mathbf{m}' \\ \mathbf{d}' &= \mathbf{W}_e^{1/2} \mathbf{d} & \text{and} & & \mathbf{d} &= \mathbf{W}_e^{-1/2} \mathbf{d}' \\ \mathbf{G}' &= \mathbf{W}_e^{1/2} \mathbf{G} \mathbf{W}_m^{-1/2} \text{ or } \mathbf{G}' = \mathbf{W}_e^{1/2} \mathbf{G} \mathbf{D}^{-1} & \mathbf{G} &= \mathbf{W}_e^{-1/2} \mathbf{G}' \mathbf{W}_m^{1/2} \text{ or } \mathbf{G} = \mathbf{W}_e^{-1/2} \mathbf{G}' \mathbf{D} \end{aligned} \quad (7.23)$$

We have encountered this weighting before, in Equations (3.46) and (5.15). In fact, the damped least squares solution (Equation (3.35))

$$\mathbf{m}^{\text{est}} = [\mathbf{G}'^T \mathbf{G}' + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{G}'^T \mathbf{d}' \quad (7.24)$$

transforms into the weighted damped least squares solution (Equation (3.45) with $\langle \mathbf{m} \rangle$ set to zero) under this transformation:

$$\begin{aligned} \mathbf{W}_m^{1/2} \mathbf{m}^{\text{est}} &= \left[\mathbf{W}_m^{-1/2} \mathbf{G}^T \mathbf{W}_e^{-1/2} \mathbf{W}_e^{1/2} \mathbf{G} \mathbf{W}_m^{-1/2} + \varepsilon^2 \mathbf{I} \right]^{-1} \mathbf{W}_m^{-1/2} \mathbf{G}^T \mathbf{W}_e^{1/2} \mathbf{W}_e^{1/2} \mathbf{d} \\ &\text{or} \\ \mathbf{m}^{\text{est}} &= [\mathbf{G}^T \mathbf{W}_e \mathbf{G} + \varepsilon^2 \mathbf{W}_m]^{-1} \mathbf{G}^T \mathbf{W}_e \mathbf{d} \end{aligned} \quad (7.25)$$

The square root of a symmetric matrix \mathbf{W} is defined via its eigenvalue decomposition. Let $\mathbf{\Lambda}$ and \mathbf{U} be the eigenvalues and eigenvectors, respectively, of \mathbf{W} so that $\mathbf{W} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$. Then

$$\begin{aligned} \mathbf{T} &= \mathbf{W}^{1/2} = \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{U}^T \quad \text{so that} \quad \mathbf{W}^{1/2T} \mathbf{W}^{1/2} = \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{U}^T \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{U}^T \\ &= \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{\Lambda}^{1/2} \mathbf{U}^T = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T = \mathbf{W} \end{aligned} \quad (7.26)$$

Here we have used the fact that $\mathbf{U}^T \mathbf{U} = \mathbf{I}$. The *MatLab* command `sqrt(wm)` correctly computes the square root of a square matrix, so the eigenvalue decomposition need not be used explicitly. Yet another possible definition of the transformation \mathbf{T} is

$$\mathbf{T} = \mathbf{\Lambda}^{1/2} \mathbf{U}^T \quad (7.27)$$

as then $\mathbf{T}^T \mathbf{T} = \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{\Lambda}^{1/2} \mathbf{U}^T = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T = \mathbf{W}$. The transformed inverse problem is then $\mathbf{G}' \mathbf{m}' = \mathbf{d}'$, where

$$\begin{aligned} \mathbf{m}' &= \left\{ \mathbf{\Lambda}_m^{1/2} \mathbf{U}_m^T \right\} \mathbf{m} & \mathbf{m} &= \left\{ \mathbf{U}_m \mathbf{\Lambda}_m^{-1/2} \right\} \mathbf{m}' \\ \mathbf{d}' &= \left\{ \mathbf{\Lambda}_e^{1/2} \mathbf{U}_e^T \right\} \mathbf{d} & \text{and} & \quad \mathbf{d} = \left\{ \mathbf{U}_e \mathbf{\Lambda}_e^{-1/2} \right\} \mathbf{d}' \\ \mathbf{G}' &= \left\{ \mathbf{\Lambda}_e^{1/2} \mathbf{U}_e^T \right\} \mathbf{G} \left\{ \mathbf{U}_m^T \mathbf{\Lambda}_m^{-1/2} \right\} & \mathbf{G} &= \left\{ \mathbf{U}_e \mathbf{\Lambda}_e^{-1/2} \right\} \mathbf{G}' \left\{ \mathbf{\Lambda}_m^{1/2} \mathbf{U}_m^T \right\} \end{aligned} \quad (7.28)$$

where $\mathbf{W}_e = \mathbf{U}_e \mathbf{\Lambda}_e \mathbf{U}_e^T$ and $\mathbf{W}_m = \mathbf{U}_m \mathbf{\Lambda}_m \mathbf{U}_m^T$. It is sometimes convenient to transform weighted L_2 problems into one or another of these two forms before proceeding with their solution.

7.5 THE SOLUTION OF THE MIXED-DETERMINED PROBLEM

The concept of vector spaces is particularly helpful in understanding the mixed-determined problem, in which some linear combinations of the model parameters are overdetermined and some are underdetermined.

If the problem is to some degree underdetermined, then the equation $\mathbf{G} \mathbf{m} = \mathbf{d}$ contains information about only some of the model parameters. We can think of these combinations as lying in a subspace $S_p(\mathbf{m})$ of the model parameters space. No information is provided about the part of the solution that lies in the rest of the space, which we shall call the *null space* $S_0(\mathbf{m})$.

The part of the \mathbf{m} that lies in the null space is completely “unilluminated” by $\mathbf{G}\mathbf{m} = \mathbf{d}$, as the equation contains no information about these linear combinations of the model parameters.

On the other hand, if the problem is to some degree overdetermined, the product $\mathbf{G}\mathbf{m}$ may not be able to span $S(\mathbf{d})$ no matter what one chooses for \mathbf{m} . At best $\mathbf{G}\mathbf{m}$ may span a subspace $S_p(\mathbf{d})$ of the data space. Then no part of the data lying outside of this space, say, in $S_0(\mathbf{d})$, can be satisfied for any choice of the model parameters.

If the model parameters and data are divided into parts with subscript p that lie in the p spaces and parts with subscript 0 that lie in the null spaces, we can write $\mathbf{G}\mathbf{m} = \mathbf{d}$ as

$$\mathbf{G}[\mathbf{m}_p + \mathbf{m}_0] = [\mathbf{d}_p + \mathbf{d}_0] \quad (7.29)$$

The solution length is then

$$L = \mathbf{m}^T \mathbf{m} = [\mathbf{m}_p + \mathbf{m}_0]^T [\mathbf{m}_p + \mathbf{m}_0] = \mathbf{m}_p^T \mathbf{m}_p + \mathbf{m}_0^T \mathbf{m}_0 \quad (7.30)$$

(The cross terms $\mathbf{m}_p^T \mathbf{m}_0$ and $\mathbf{m}_0^T \mathbf{m}_p$ are zero as the vectors lie in different spaces.) Similarly, the prediction error is

$$E = [\mathbf{d}_p + \mathbf{d}_0 - \mathbf{G}\mathbf{m}_p]^T [\mathbf{d}_p + \mathbf{d}_0 - \mathbf{G}\mathbf{m}_p] = [\mathbf{d}_p - \mathbf{G}\mathbf{m}_p]^T [\mathbf{d}_p - \mathbf{G}\mathbf{m}_p] + \mathbf{d}_0^T \mathbf{d}_0 \quad (7.31)$$

(as $\mathbf{G}\mathbf{m}_0 = 0$ and \mathbf{d}_p and \mathbf{d}_0 lie in different spaces). We can now define precisely what we mean by a solution to the mixed-determined problem that minimizes prediction error while adding a minimum of *a priori* information: *a priori* information is added to specify only those linear combinations of the model parameters that reside in the null space $S_0(\mathbf{m})$, and the prediction error is reduced to only the portion in the null space $S_0(\mathbf{d})$ by satisfying $\mathbf{e}_p = [\mathbf{d}_p - \mathbf{G}\mathbf{m}_p] = 0$ exactly. One possible choice of *a priori* information is $\mathbf{m}_0^{\text{est}} = 0$, which is sometimes called the “natural solution” of the mixed-determined problem. We note that when $\mathbf{G}\mathbf{m} = \mathbf{d}$ is purely underdetermined the natural solution is just the minimum-length solution, and when $\mathbf{G}\mathbf{m} = \mathbf{d}$ is purely overdetermined it is just the least squares solution.

One might be tempted to view the natural solution as *better* than, say, the damped least squares solution, as the prior information is applied only to the part of the solution that is in the null space and does not increase the prediction error E . However, such an assessment is not clear-cut. If the prior information is indeed accurate, it should be fully utilized, even if its use leads to a slightly larger prediction error. Anyway, given noisy measurements, two slightly different prediction errors will not be statistically distinguishable. This analysis emphasizes that the choice of the inversion method must be tailored carefully to the problem at hand.

7.6 SINGULAR-VALUE DECOMPOSITION AND THE NATURAL GENERALIZED INVERSE

The p and null subspaces of the linear problem can be easily identified through a type of eigenvalue decomposition of the data kernel that is called the *singular-value decomposition*. We shall derive this decomposition in Section 7.7, but first we shall state the result and demonstrate its usefulness.

Any $N \times M$ square matrix can be written as the product of three matrices (Penrose, 1955; Lanczos, 1961):

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \quad (7.32)$$

The matrix \mathbf{U} is an $N \times N$ matrix of eigenvectors that span the data space $S(\mathbf{d})$:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \mathbf{u}^{(3)} & \dots & \mathbf{u}^{(N)} \end{bmatrix} \quad (7.33)$$

where the $\mathbf{u}^{(i)}$ s are the individual vectors. The vectors are orthogonal to one another and can be chosen to be of unit length so that $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$ (where the identity matrix is $N \times N$). Similarly, \mathbf{V} is an $M \times M$ matrix of eigenvectors that span the model parameter space $S(\mathbf{m})$ as

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & \mathbf{v}^{(3)} & \dots & \mathbf{v}^{(M)} \end{bmatrix} \quad (7.34)$$

Here the $\mathbf{v}^{(i)}$ s are the individual orthonormal vectors so that $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ (the identity matrix being $M \times M$). The matrix $\mathbf{\Lambda}$ is an $N \times M$ diagonal eigenvalue matrix whose diagonal elements are nonnegative and are called *singular values*. In the $N=4, M=3$ case

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \\ 0 & 0 & 0 \end{bmatrix} \quad (7.35)$$

The singular values are usually arranged in order of decreasing size. Some of the singular values may be zero. We therefore partition $\mathbf{\Lambda}$ into a submatrix $\mathbf{\Lambda}_p$ of p nonzero singular values and several zero matrices as

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (7.36)$$

where $\mathbf{\Lambda}_p$ is a $p \times p$ diagonal matrix. The decomposition then becomes $\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T$, where \mathbf{U}_p and \mathbf{V}_p consist of the first p columns of \mathbf{U} and \mathbf{V} , respectively. The other portions of the eigenvector matrices are canceled by the zeros in $\mathbf{\Lambda}$. The matrix \mathbf{G} contains no information about the subspaces spanned by these portions of the data and model eigenvectors, which we shall call \mathbf{V}_0 and \mathbf{U}_0 , respectively. As we shall soon prove, these are precisely the same spaces as the p and null spaces defined in the previous section.

The data kernel is not a function of the null eigenvectors \mathbf{V}_0 and \mathbf{U}_0 . The equation $\mathbf{d} = \mathbf{G}\mathbf{m} = \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{m}$ contains no information about the part of the model parameters in the space spanned by \mathbf{V}_0 as the model parameters \mathbf{m} are multiplied by \mathbf{V}_p (which is orthogonal to everything in \mathbf{V}_0). The eigenvector \mathbf{V}_p , therefore, lies completely in $S_p(\mathbf{m})$, and \mathbf{V}_0 lies completely in $S_0(\mathbf{m})$. Similarly, no matter what value $\{\mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{m}\}$ attains, it can have no component in the space spanned by \mathbf{U}_0 as it is multiplied by \mathbf{U}_p (and \mathbf{U}_0 and \mathbf{U}_p are orthogonal). Therefore, \mathbf{U}_p lies completely in $S_p(\mathbf{d})$ and \mathbf{U}_0 lies completely in $S_0(\mathbf{d})$.

We have demonstrated that the p and null spaces can be identified through the singular-value decomposition of the data kernel. The full spaces $S(\mathbf{m})$ and $S(\mathbf{d})$ are spanned by \mathbf{V} and \mathbf{U} , respectively. The p spaces are spanned by the parts of the eigenvector matrices that have nonzero eigenvalues: $S_p(\mathbf{m})$ is spanned by \mathbf{V}_p and $S_p(\mathbf{d})$ is spanned by \mathbf{U}_p . The remaining eigenvectors \mathbf{V}_0 and \mathbf{U}_0 span the null spaces $S_0(\mathbf{m})$ and $S_0(\mathbf{d})$. The p and null matrices are orthogonal and are normalized in the sense that $\mathbf{V}_p^T\mathbf{V}_p = \mathbf{U}_p^T\mathbf{U}_p = \mathbf{I}$, where \mathbf{I} is $p \times p$ in size. However, as these matrices do not in general span the complete data and model spaces, $\mathbf{V}_p\mathbf{V}_p^T$ and $\mathbf{U}_p\mathbf{U}_p^T$ are not in general identity matrices.

The natural solution to the inverse problem can be constructed from the singular-value decomposition. This solution must have an \mathbf{m}^{est} that has no component in $S_0(\mathbf{m})$ and a prediction error \mathbf{e} that has no component in $S_p(\mathbf{d})$. We therefore consider the solution

$$\mathbf{m}^{\text{est}} = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{d} \quad (7.37)$$

which is picked in analogy to the square matrix case. To show that \mathbf{m}^{est} has no component in $S_0(\mathbf{m})$, we take the dot product of the equation with \mathbf{V}_0 , which lies completely in $S_0(\mathbf{m})$, as

$$\mathbf{V}_0^T\mathbf{m}^{\text{est}} = \mathbf{V}_0^T\mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{d} = \mathbf{0} \quad (7.38)$$

as \mathbf{V}_0^T and \mathbf{V}_p are orthogonal. To show that \mathbf{e} has no component in $S_p(\mathbf{d})$, we take the dot product with \mathbf{U}_p as

$$\begin{aligned} \mathbf{U}_p^T\mathbf{e} &= \mathbf{U}_p^T[\mathbf{d} - \mathbf{G}\mathbf{m}^{\text{est}}] = \mathbf{U}_p^T[\mathbf{d} - \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{d}] \\ &= \mathbf{U}_p^T[\mathbf{d} - \mathbf{U}_p\mathbf{U}_p^T\mathbf{d}] = \mathbf{U}_p^T\mathbf{d} - \mathbf{U}_p^T\mathbf{d} = \mathbf{0} \end{aligned} \quad (7.39)$$

as $\mathbf{V}_p^T\mathbf{V}_p = \mathbf{U}_p^T\mathbf{U}_p = \mathbf{\Lambda}_p\mathbf{\Lambda}_p^{-1} = \mathbf{I}$. The natural solution of the inverse problem is therefore shown to be

$$\mathbf{m}^{\text{est}} = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{d} \quad (7.40)$$

We note that we can define a generalized inverse operator for the mixed-determined problem, the *natural generalized inverse* $\mathbf{G}^{-g} = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T$. (This generalized inverse is so useful that it is sometimes referred to as *the* generalized inverse, although of course there are other generalized inverses that can be designed for the mixed-determined problem that embody other kinds of *a priori* information.)

In *MatLab*, the singular-value decomposition is computed as

```
[U, L, V] = svd(G);
lambda = diag(L);
```

(*MatLab* script gda07_03)

Here we have copied the diagonal elements of matrix \mathbf{L} of singular values into the vector `lambda`. The submatrices \mathbf{U}_p and \mathbf{V}_p (with p an integer) are extracted via:

```
Up = U(:, 1:p);
Vp = V(:, 1:p);
lambdap = lambda(1:p);
```

(*MatLab* script gda07_03)

and the model parameters are estimated as

```
mest = Vp * ((Up' * dobs) ./ lambdap);
```

(*MatLab* script gda07_03)

Note that the calculation is organized so that the matrices \mathbf{U}_p^T and \mathbf{V}_p multiply vectors (as contrasted to matrices); the generalized inverse is never explicitly formed. The value of the integer p must be chosen so that zero eigenvalues are excluded from the estimate; else a division-by-zero error will occur. As discussed below, one often excludes near-zero eigenvalues as well, as the resulting solution (while not quite the natural solution) is often better behaved.

The natural generalized inverse has model resolution

$$\mathbf{R} = \mathbf{G}^{-g} \mathbf{G} = \left\{ \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T \right\} \left\{ \mathbf{U}_p \mathbf{\Lambda}_p \mathbf{V}_p^T \right\} = \mathbf{V}_p \mathbf{V}_p^T \quad (7.41)$$

The model parameters will be perfectly resolved only if \mathbf{V}_p spans the complete space of model parameters, that is, if there are no zero eigenvalues and $p \geq M$. The data resolution matrix is

$$\mathbf{N} = \mathbf{G} \mathbf{G}^{-g} = \left\{ \mathbf{U}_p \mathbf{\Lambda}_p \mathbf{V}_p^T \right\} \left\{ \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T \right\} = \mathbf{U}_p \mathbf{U}_p^T \quad (7.42)$$

The data are only perfectly resolved if \mathbf{U}_p spans the complete space of data and $p = N$. Finally, we note that if the data are uncorrelated with uniform variance σ_d^2 , the model covariance is

$$\begin{aligned} [\text{cov } \mathbf{m}^{\text{est}}] &= \mathbf{G}^{-g} [\text{cov } \mathbf{d}] \mathbf{G}^{-gT} = \sigma_d^2 \left\{ \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T \right\} \left\{ \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T \right\}^T \\ &= \sigma_d^2 \mathbf{V}_p \mathbf{\Lambda}_p^{-2} \mathbf{V}_p^T \end{aligned} \quad (7.43)$$

The covariance of the estimated model parameters is very sensitive to the smallest nonzero eigenvalue. (Note that forming the natural inverse corresponds to assuming that linear combinations of the *a priori* model parameters in the p space have infinite variance and that combinations in the null space have zero

variance and zero mean.) The covariance of the estimated model parameters, therefore, does not explicitly contain $[\text{cov } \mathbf{m}]$. If one prefers a solution based on the natural inverse (but with the null vectors chosen to minimize the distance to a set of *a priori* model parameters with mean $\langle \mathbf{m} \rangle$ and covariance $[\text{cov } \mathbf{m}]_A$), it is appropriate to use the formula $\mathbf{m}^{\text{est}} = \mathbf{G}^{-g} \mathbf{d} + [\mathbf{I} - \mathbf{R}] \langle \mathbf{m} \rangle$, where \mathbf{G}^{-g} is the natural inverse. The covariance of this estimate is now

$$[\text{cov } \mathbf{m}^{\text{est}}] = \mathbf{G}^{-g} [\text{cov } \mathbf{d}] \mathbf{G}^{-gT} + [\mathbf{I} - \mathbf{R}] [\text{cov } \mathbf{m}]_A [\mathbf{I} - \mathbf{R}]^T \quad (7.44)$$

which is based on the usual rule for computing covariances.

To use the natural inverse one must be able to identify the number p , that is, to count the number of nonzero singular values. Plots of the sizes of the singular values against their index numbers (the *spectrum* of the data kernel) can be useful in this process. The value of p can be easily determined if the singular values fall into two clearly distinguishable groups, one nonzero and one zero (Figure 7.3A and B). In realistic inverse problems, however, the singular values often smoothly decline in size (Figure 7.3C and D), making it difficult to distinguish ones that are actually nonzero from ones that are zero but computed somewhat inaccurately owing to round-off error by the computer. Furthermore, if one chooses p so as to include these very small singular values, the solution variance will be very large, as it contains the factor Λ_p^{-2} . One solution

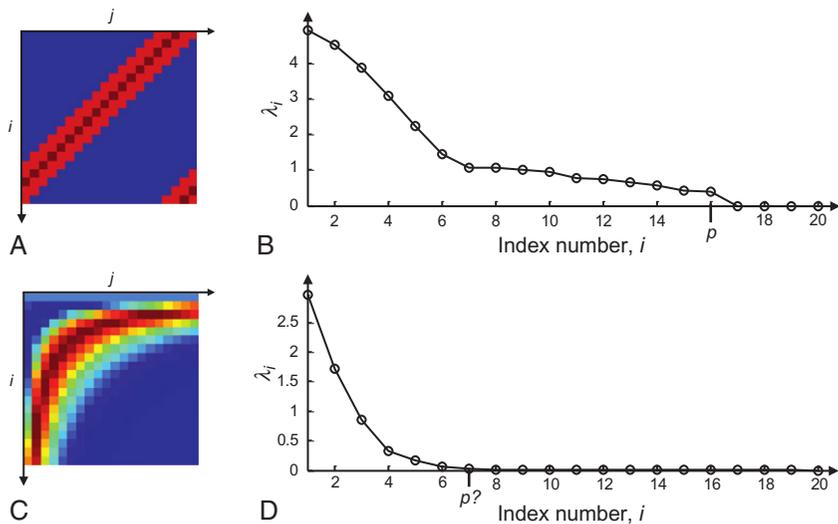


FIGURE 7.3 (A) Hypothetical data kernel G_{ij} for an inverse problem with $M=20$ model parameters and $N=20$ observations. (B) Corresponding singular values λ_i have a clear cutoff at $p=16$. (C) Another hypothetical data kernel, also with $N=20$ and $M=20$. (D) Corresponding singular values do not have a clear cutoff, so the parameter, p , must be chosen in a more arbitrary fashion near $p \approx 7$. *MatLab* Script gda07_04.

to this problem is to pick some cutoff size for singular values and then consider any values smaller than this as equal to zero. This process artificially reduces the dimensions of \mathbf{V}_p and \mathbf{U}_p that are included in the generalized inverse. The resulting estimates of the model parameters are no longer exactly the natural solution. But, if only small singular values are excluded, the solution is generally close to the natural solution and possesses better variance. On the other hand, its model and data resolution are worse. We recognize that this trade-off is just another manifestation of the trade-off between resolution and variance discussed in Chapter 4.

As an example, we consider the problem $\mathbf{Gm} = \mathbf{d}$ illustrated in Figure 7.4A, which has the same data kernel as in Figure 7.3A. Although \mathbf{G} is 20×20 , the problem is mixed-determined, as $p = 16$ and four of the singular values are zero. In this case, the natural solution is very close to the true solution, presumably because, by coincidence, the true solution did not have much of its power in the null space. The natural solution is also close to the damped minimum-length solution (Equation (4.22)), which is not surprising, for both arise from similar minimization principles. The damped minimum-length solution minimizes a weighted sum of prediction error and solution length; the natural solution minimizes the prediction error and the component of the solution length in the null space.

Instead of choosing a sharp cutoff for the singular values, it is possible to include all the singular values while damping the smaller ones. We let $p = M$, but replace the reciprocals of all the singular values by $\lambda_i / (\varepsilon^2 + \lambda_i^2)$, where ε is some small number. This change has little effect on the larger singular values but prevents the smaller ones from leading to large variances. Of course, the solution is no longer the natural solution. While its variance is improved, its

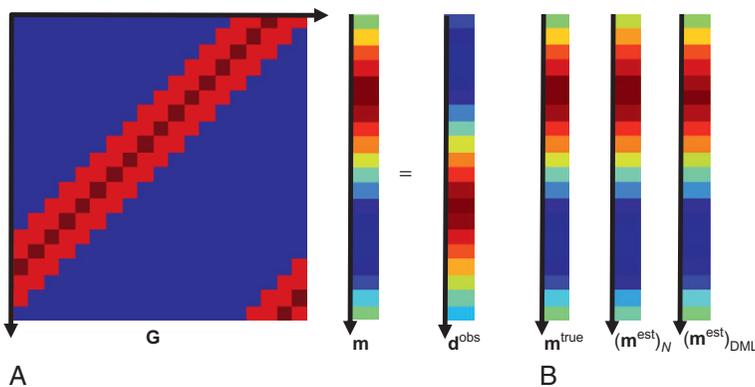


FIGURE 7.4 (A) Same linear problem as in Figure 7.3A, where $\mathbf{d}^{\text{obs}} = \mathbf{Gm}^{\text{true}} + \mathbf{n}$, with \mathbf{n} uncorrelated Gaussian noise. (B) Corresponding solutions, true solutions \mathbf{m}^{true} , natural solution $(\mathbf{m}^{\text{est}})_N$, and damped minimum-length solution $(\mathbf{m}^{\text{est}})_{\text{DML}}$. *MatLab* Script gda07_03.

model and data resolution are degraded. In fact, this solution is precisely the damped least squares solution discussed in [Chapter 3](#):

$$\begin{aligned}
 [\mathbf{G}^T \mathbf{G} + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{G}^T &= [\mathbf{V} \mathbf{\Lambda} \mathbf{U}^T \mathbf{U} \mathbf{A} \mathbf{V}^T + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{V} \mathbf{\Lambda} \mathbf{U}^T \\
 &= [\mathbf{V} \mathbf{\Lambda}^2 \mathbf{V}^T + \varepsilon^2 \mathbf{V} \mathbf{I} \mathbf{V}^T]^{-1} \mathbf{V} \mathbf{\Lambda} \mathbf{U}^T \\
 &= \mathbf{V} [\mathbf{\Lambda}^2 + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{V}^T \mathbf{V} \mathbf{\Lambda} \mathbf{U}^T \\
 &= \mathbf{V} \{ [\mathbf{\Lambda}^2 + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{\Lambda} \} \mathbf{U}^T
 \end{aligned} \tag{7.45}$$

Here we rely upon the fact that if $\mathbf{M} = \mathbf{V} \mathbf{D} \mathbf{V}^T$ is the eigenvalue decomposition of \mathbf{M} , then $\mathbf{M}^{-1} = \mathbf{V} \mathbf{D}^{-1} \mathbf{V}^T$.

The damping of the singular values corresponds to the addition of *a priori* information that the model parameters are small. The precise value of the number used as the cutoff or damping parameter must be chosen by a trial-and-error process which weighs the relative merits of having a solution with small variance against those of having one that fits the data and is well resolved.

In [Section 6.6](#), we discussed the problem of bounding nonunique averages of model parameters by incorporating *a priori* inequality constraints into the solution of the inverse problem. We see that the singular-value decomposition provides a simple way of identifying the null vectors of $\mathbf{G} \mathbf{m} = \mathbf{d}$. The general solution to the inverse problem ([Wunsch and Minster, 1982](#))

$$\mathbf{m}^{\text{gen}} = \mathbf{m}^{\text{par}} + \sum_{i=1}^p \alpha_i \mathbf{m}^{\text{null}(i)} \tag{7.46}$$

can be thought of as having the natural solution as its particular solution and a sum over the null eigenvectors as its null solution:

$$\mathbf{m}^{\text{gen}} = \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T \mathbf{d} + \mathbf{V}_0 \boldsymbol{\alpha} \tag{7.47}$$

There are $q = M - p$ null vectors in the general solution, with the coefficients given by $\boldsymbol{\alpha}$. In [Section 6.6](#), upper and lower bounds on localized averages of the solution were found by determining the α that maximizes (for the upper bound) or minimizes (for the lower bound) the localized average $\langle m \rangle = \mathbf{a}^T \mathbf{m}$ with the constraint that $\mathbf{m}^l \leq \mathbf{m} \leq \mathbf{m}^u$, where \mathbf{m}^l and \mathbf{m}^u are *a priori* bounds. The use of the natural solution guarantees that the prediction error is minimized in the L_2 sense.

The bounds on the localized average $\langle m \rangle = \mathbf{a}^T \mathbf{m}$ should be treated with some skepticism, as they depend on a particular choice of the solution (in this case one that minimizes the L_2 prediction error). If the total error E increases only slightly as this solution is perturbed and if this additional error can be considered negligible, then the true bounds of the localized average will be larger than those given above. In principle, one can handle this problem by forsaking the eigenvalue decomposition and simply determining the \mathbf{m} that extremizes $\langle m \rangle$ with the constraints that $\mathbf{m}^l \leq \mathbf{m} \leq \mathbf{m}^u$ and that the total prediction error is less than some tolerable amount, say, E_M . For L_2 measures of the

prediction error, this is a very difficult nonlinear problem. However, if the error is measured under the L_1 norm, it can be transformed into a linear programming problem.

7.7 DERIVATION OF THE SINGULAR-VALUE DECOMPOSITION

The singular-value decomposition can be derived in many ways. We follow here the treatment of [Lanczos \(1961\)](#). For an alternate derivation, see [Menke and Menke \(2011, Section 8.2\)](#). We first form an $(N+M) \times (N+M)$ square symmetric matrix \mathbf{S} from \mathbf{G} and \mathbf{G}^T as

$$\mathbf{S} = \begin{bmatrix} 0 & \mathbf{G} \\ \mathbf{G}^T & 0 \end{bmatrix} \quad (7.48)$$

From elementary linear algebra, we know that this matrix has $N+M$ real eigenvalues λ_i and a complete set of eigenvectors $\mathbf{w}^{(i)}$ which solve $\mathbf{S}\mathbf{w}^{(i)} = \lambda_i \mathbf{w}^{(i)}$. Partitioning \mathbf{w} into a part \mathbf{u} of length N and a part \mathbf{v} of length M , we obtain

$$\mathbf{S}\mathbf{w}^{(i)} = \lambda_i \mathbf{w}^{(i)} \rightarrow \begin{bmatrix} 0 & \mathbf{G} \\ \mathbf{G}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(i)} \\ \mathbf{v}^{(i)} \end{bmatrix} = \lambda_i \begin{bmatrix} \mathbf{u}^{(i)} \\ \mathbf{v}^{(i)} \end{bmatrix} \quad (7.49)$$

We shall now show that $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ are the same vectors as those defined in the previous section. We first note that the above equation implies that $\mathbf{G}\mathbf{v}^{(i)} = \lambda_i \mathbf{u}^{(i)}$ and $\mathbf{G}^T \mathbf{u}^{(i)} = \lambda_i \mathbf{v}^{(i)}$. Suppose that there is a positive eigenvalue λ_i with eigenvector $[\mathbf{u}^{(i)}, \mathbf{v}^{(i)}]^T$. Then we note that $-\lambda_i$ is also an eigenvalue with eigenvector $[-\mathbf{u}^{(i)}, \mathbf{v}^{(i)}]^T$. If there are p positive eigenvalues, then there are $N+M-2p$ zero eigenvalues. Now by manipulating the above equations we obtain

$$\mathbf{G}^T \mathbf{G} \mathbf{v}^{(i)} = \lambda_i^2 \mathbf{v}^{(i)} \quad \text{and} \quad \mathbf{G} \mathbf{G}^T \mathbf{u}^{(i)} = \lambda_i^2 \mathbf{u}^{(i)} \quad (7.50)$$

As a symmetric matrix can have no more distinct eigenvectors than its dimension, we note that $p \leq \min(N, M)$. As both matrices are square and symmetric, there are M vectors $\mathbf{v}^{(i)}$ that form a complete orthogonal set \mathbf{V} spanning $S(\mathbf{m})$ and N vectors $\mathbf{u}^{(i)}$ that form a complete orthogonal set \mathbf{U} spanning $S(\mathbf{d})$. These include p of the \mathbf{w} eigenvectors with distinct nonzero eigenvalues and remaining ones chosen from the eigenvectors with zero eigenvalues. The equation $\mathbf{G}\mathbf{v}^{(i)} = \lambda_i \mathbf{u}^{(i)}$ can be written in matrix form as $\mathbf{G}\mathbf{V} = \mathbf{U}\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues. Postmultiplying by \mathbf{V}^T gives the singular-value decomposition $\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$.

7.8 SIMPLIFYING LINEAR EQUALITY AND INEQUALITY CONSTRAINTS

The singular-value decomposition can be used to simplify linear constraints.

7.8.1 Linear Equality Constraints

Consider the problem of solving $\mathbf{G}\mathbf{m} = \mathbf{d}$ in the sense of finding a solution that minimizes the L_2 prediction error subject to the $K < M$ constraints that $\mathbf{H}\mathbf{m} = \mathbf{h}$. This problem can be reduced to the unconstrained problem $\mathbf{G}'\mathbf{m}' = \mathbf{d}'$ in $M' \leq M$ new model parameters. We first find the singular-value decomposition of the constraint matrix $\mathbf{H} = \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T$. If $p=K$, the constraints are consistent and determine p linear combinations of the unknowns. The general solution is then $\mathbf{m} = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{h} + \mathbf{V}_0\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is an arbitrary vector of length $M-p$ and is to be determined by minimizing the prediction error. Substituting this equation for \mathbf{m} into $\mathbf{G}\mathbf{m} = \mathbf{d}$ and rearranging terms yields

$$\mathbf{G}\mathbf{V}_0\boldsymbol{\alpha} = \mathbf{d} - \mathbf{G}\mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{h} \quad (7.51)$$

This equation in the unknown coefficients $\boldsymbol{\alpha}$ now can be solved as an unconstrained least squares problem. We note that we have encountered this problem in a somewhat different form during the discussion of Householder transformations (Section 7.2). The main advantage of using the singular-value decomposition is that it provides a test of the constraint's consistency.

7.8.2 Linear Inequality Constraints

Consider the L_2 problem

$$\text{minimize } \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2 \quad \text{subject to } \mathbf{H}\mathbf{m} \geq \mathbf{h} \quad (7.52)$$

We shall show that as long as $\mathbf{G}\mathbf{m} = \mathbf{d}$ is in fact overdetermined, this problem can be reduced to the simpler problem (Lawson and Hanson, 1974):

$$\text{minimize } \|\mathbf{m}'\|_2 \quad \text{subject to } \mathbf{H}'\mathbf{m}' \geq \mathbf{h}' \quad (7.53)$$

To demonstrate this transformation, we form the singular-value decomposition of the data kernel $\mathbf{G} = \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T$. We first divide the data \mathbf{d} into two parts $\mathbf{d} = [\mathbf{d}_p^T, \mathbf{d}_0^T]^T$ where $\mathbf{d}_p = \mathbf{U}_p^T\mathbf{d}$ is in the $S_p(\mathbf{m})$ subspace and $\mathbf{d}_0 = \mathbf{U}_0^T\mathbf{d}$ is in the $S_0(\mathbf{m})$ subspace. The prediction error is then $E = \|\mathbf{d}^{\text{obs}} - \mathbf{d}^{\text{pre}}\|_2$ or

$$\begin{aligned} E &= \left\| \begin{bmatrix} \mathbf{U}_p^T\mathbf{d} \\ \mathbf{U}_0^T\mathbf{d} \end{bmatrix} - \begin{bmatrix} \mathbf{U}_p^T\mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{m} \\ 0 \end{bmatrix} \right\|_2 = \|\mathbf{d}_p - \mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{m}\|_2 + \|\mathbf{d}_0\|_2 \\ &= \|\mathbf{m}'\|_2 + \|\mathbf{d}_0\|_2 \end{aligned} \quad (7.54)$$

where $\mathbf{m}' = \mathbf{d}_p - \mathbf{\Lambda}_p\mathbf{V}_p^T\mathbf{m}$. We note that minimizing $\|\mathbf{m}'\|_2$ is the same as minimizing E as the other term is a constant. Inverting this expression for the unprimed model parameters gives $\mathbf{m} = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}[\mathbf{d}_p - \mathbf{m}'] = \mathbf{V}_p\mathbf{\Lambda}_p^{-1}[\mathbf{U}_p^T\mathbf{d} - \mathbf{m}']$. Substituting this expression into the constraint equation $\mathbf{H}\mathbf{m} \geq \mathbf{h}$ and rearranging terms yields

$$\left\{ -\mathbf{H}\mathbf{V}_p\mathbf{\Lambda}_p^{-1} \right\} \mathbf{m}' \geq \left\{ \mathbf{h} - \mathbf{H}\mathbf{V}_p\mathbf{\Lambda}_p^{-1}\mathbf{U}_p^T\mathbf{d} \right\} \quad \text{or} \quad \mathbf{H}'\mathbf{m}' \geq \mathbf{h}' \quad (7.55)$$

which is in the desired form.

7.9 INEQUALITY CONSTRAINTS

We shall now consider the solution of L_2 minimization problems with inequality constraints of the form

$$\text{minimize} \quad \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2 \quad \text{subject to} \quad \mathbf{H}\mathbf{m} \geq \mathbf{h} \quad (7.56)$$

We first note that problems involving $=$ and \leq constraints can be reduced to this form. Equality constraints can be removed by the transformation described in Section 7.8.1, and \leq constraints can be removed by multiplication by -1 to change them into \geq constraints. For this minimization problem to have any solution, the constraints $\mathbf{H}\mathbf{m} \geq \mathbf{h}$ must be consistent; there must be at least one \mathbf{m} that satisfies all the constraints. We can view these constraints as defining a volume in $S(\mathbf{m})$. Each constraint defines a hyperplane that divides $S(\mathbf{m})$ into two half-spaces, one in which that constraint is satisfied (the *feasible half-space*) and the other in which it is violated (the *infeasible half-space*). The set of inequality constraints, therefore, defines a volume in $S(\mathbf{m})$ which might be zero, finite, or infinite in extent. If the region has zero volume, then no feasible solution exists (Figure 7.5B). If it has nonzero volume, then there is at least one solution that minimizes the prediction error (Figure 7.5A). This volume has the shape of a polyhedron as its boundary surfaces are planes. It can be shown that the polyhedron must be convex: it can have no reentrant angles or grooves.

The starting point for solving the L_2 minimization problem with inequality constraints is the Kuhn-Tucker theorem, which describes the properties that any solution to this problem must possess. For any \mathbf{m} that minimizes

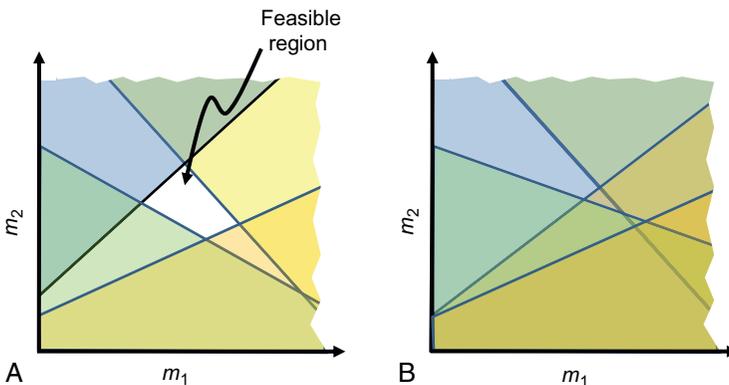


FIGURE 7.5 (A) Each linear inequality constraint divides $S(\mathbf{m})$ into two half-spaces, one infeasible (shaded) and the other feasible (white). Consistent constraints combine to form a convex, polygonal region within $S(\mathbf{m})$ (white) of feasible \mathbf{m} . (B) Inconsistent constraints do not form a feasible region.

$\|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2$ subject to p constraints $\mathbf{H}\mathbf{m} \geq \mathbf{h}$, it is possible to find a vector \mathbf{y} of length p such that

$$\frac{1}{2}\nabla E = \nabla[\mathbf{H}\mathbf{m}] \cdot \mathbf{y} \quad \text{or} \quad -\mathbf{G}^T[\mathbf{d} - \mathbf{G}\mathbf{m}] = \mathbf{H}^T \mathbf{y} \quad (7.57)$$

This equation states that the gradient of the error ∇E can be written as a linear combination of hyperplane normals \mathbf{H}^T , with the combination specified by the vector \mathbf{y} . The Kuhn-Tucker theorem goes on to state that the elements of \mathbf{y} are nonnegative; that \mathbf{y} can be partitioned into two parts \mathbf{y}_E and \mathbf{y}_S (possibly requiring reordering of the constraints) that satisfy

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_E > 0 \\ \mathbf{y}_S = 0 \end{bmatrix} \quad \text{and} \quad \begin{matrix} \mathbf{H}_E \mathbf{m} - \mathbf{h}_E = 0 \\ \mathbf{H}_S \mathbf{m} - \mathbf{h}_S > 0 \end{matrix} \quad (7.58)$$

The first group of constraints is satisfied in the equality sense (thus the subscript E for “equality”). The rest are satisfied more loosely in the inequality sense (thus the subscript S for “slack”).

The theorem indicates that any feasible solution \mathbf{m} is the minimum solution only if the direction in which one would have to perturb \mathbf{m} to decrease the total error E causes the solution to cross some constraint hyperplane and become infeasible. The direction of decreasing error is $-1/2\nabla E = \mathbf{G}^T[\mathbf{d} - \mathbf{G}\mathbf{m}]$. The constraint hyperplanes have normals $+\nabla[\mathbf{H}\mathbf{m}] = \mathbf{H}^T$ which point into the feasible side. As $\mathbf{H}_E \mathbf{m} - \mathbf{h}_E = 0$, the solution lies exactly on the bounding hyperplanes of the \mathbf{H}_E constraints. As $\mathbf{H}_S \mathbf{m} - \mathbf{h}_S > 0$, it lies within the feasible volume of the \mathbf{H}_S constraints. An infinitesimal perturbation $\delta\mathbf{m}$ of the solution can, therefore, only violate the \mathbf{H}_E constraints. If it is not to violate these constraints, the perturbation must be made in the direction of feasibility so that it must be expressible as a nonnegative combination of hyperplane normals, that is, $\delta\mathbf{m} \cdot \nabla[\mathbf{H}\mathbf{m}] \geq 0$. On the other hand, if it is to decrease the total prediction error it must satisfy $\delta\mathbf{m} \cdot \nabla E \leq 0$. These two conditions are incompatible with the Kuhn-Tucker theorem, as dotting Equation (7.57) with $\delta\mathbf{m}$ yields $\delta\mathbf{m} \cdot 1/2\nabla E = \delta\mathbf{m} \cdot \nabla[\mathbf{H}\mathbf{m}] \cdot \mathbf{y} \geq 0$ as both $\delta\mathbf{m} \cdot \nabla[\mathbf{H}\mathbf{m}]$ and \mathbf{y} are positive. These solutions are indeed minimum solutions to the constrained problem (Figure 7.6).

To demonstrate how the Kuhn-Tucker theorem can be used, we consider the simplified problem

$$\text{minimize } E = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2 \quad \text{subject to } \mathbf{m} \geq \mathbf{0} \quad (7.59)$$

which is called *nonnegative least squares*. We find the solution using an iterative scheme of several steps (Lawson and Hanson, 1974):

Step 1. Start with an initial guess for \mathbf{m} . As $\mathbf{H} = \mathbf{I}$ each model parameter is associated with exactly one constraint. These model parameters can be separated into a set \mathbf{m}_E that satisfies the constraints in the equality sense and a set \mathbf{m}_S that satisfies the constraints in the inequality sense. The particular initial guess $\mathbf{m} = \mathbf{0}$ is clearly feasible and has all its elements in \mathbf{m}_E .

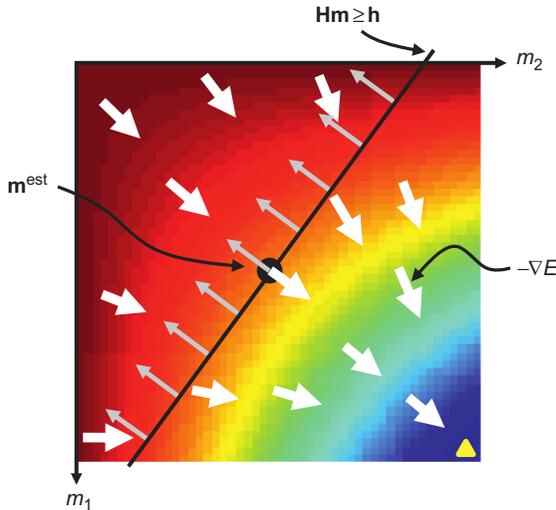


FIGURE 7.6 The error $E(\mathbf{m})$ (colors) has a single minimum (yellow triangle). The linear equality constraint, $\mathbf{H}\mathbf{m} \geq \mathbf{h}$, divides $S(\mathbf{m})$ into two half-spaces (black line, with gray arrows pointing into the feasible half-space). Solution (circle) lies on the boundary between the two half-spaces and therefore satisfies the constraint in the equality sense. At this point, the normal of the constraint hyperplane (gray arrow) is antiparallel to $-\nabla E$ (white arrows). *MatLab* script gda07_05.

Step 2. Any model parameter m_i in \mathbf{m}_E that has associated with it a negative gradient $[\nabla E]_i$ can be changed both to decrease the error and to remain feasible. Therefore, if there is no such model parameter in \mathbf{m}_E , the Kuhn-Tucker theorem indicates that this \mathbf{m} is the solution to the problem.

Step 3. If some model parameter m_i in \mathbf{m}_E has a corresponding negative gradient, then the solution can be changed to decrease the prediction error. To change the solution, we select the model parameter corresponding to the most negative gradient and move it to the set \mathbf{m}_S . All the model parameters in \mathbf{m}_S are now recomputed by solving the system $\mathbf{G}_S \mathbf{m}'_S = \mathbf{d}_S$ in the least squares sense. The subscript S on the matrix indicates that only the columns multiplying the model parameters in \mathbf{m}_S have been included in the calculation. All the $\mathbf{m}_{E,S}$ are still zero. If the new model parameters are all feasible, then we set $\mathbf{m} = \mathbf{m}'$ and return to Step 2.

Step 4. If some of the elements of \mathbf{m}'_S are infeasible, however, we cannot use this vector as a new guess for the solution. Instead, we compute the change in the solution $\delta \mathbf{m} = \mathbf{m}'_S - \mathbf{m}_S$ and add as much of this vector as possible to the solution \mathbf{m}_S without causing the solution to become infeasible. We therefore replace \mathbf{m}_S with the new guess $\mathbf{m}_S + \alpha \delta \mathbf{m}$, where $\alpha = \min_i \{m_{Si} / [m_{Si} - m'_{Si}]\}$ is the largest choice that can be made without some \mathbf{m}_S becoming infeasible. At least one of the m_{Si} s has its constraint satisfied in the equality sense and must be moved back to \mathbf{m}_E . The process then returns to Step 3.

This algorithm contains two loops, one nested within the other. The outer loop successively moves model parameters from the group that is constrained to the group that minimizes the prediction error. The inner loop ensures that the addition of a variable to this latter group has not caused any of the constraints to be violated. Discussion of the convergence properties of this algorithm can be found in [Lawson and Hanson \(1974\)](#).

MatLab provides a function that implements nonnegative least squares

```
mest = lsqnonneg(G, dobs);
```

(*MatLab* script gda07_06)

As an example of its use, we analyze the problem of determining the mass distribution of an object from observations of its gravitational force ([Figure 7.7](#)). Mass is an inherently positive quantity, so the positivity constraint constitutes very accurate *a priori* information. The gravitational force d_i is measured at $N=600$ points (x_i, y_i) above the object. The object ([Figure 7.7A](#)) is subdivided into a grid of 20×20 pixels, each of mass m_j and position (x_j, y_j) . According to Newton's inverse-square law, the data kernel is $G_{ij} = \gamma \cos(\theta_{ij})/R_{ij}^2$, where γ is the gravitational constant, θ_{ij} is angle with respect to the vertical and R_{ij} is distance. Singular-value decomposition of \mathbf{G} indicates that this mixed-determined problem is extremely nonunique, with only about 20 nonzero eigenvalues.

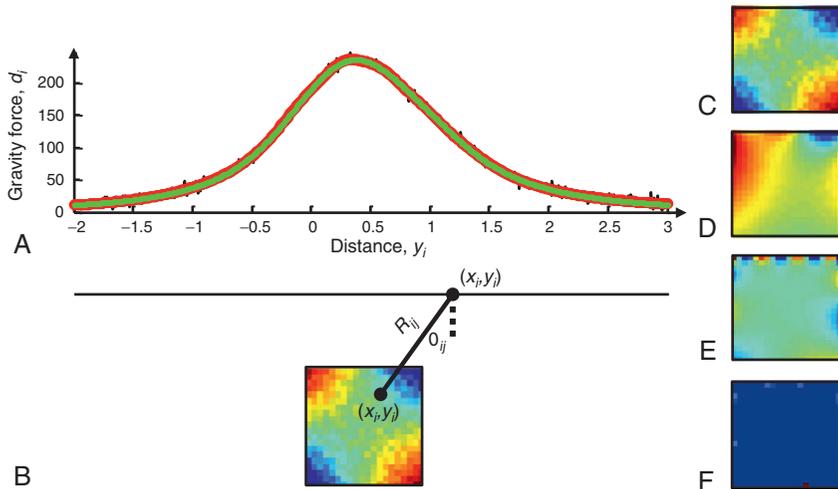


FIGURE 7.7 (A) The vertical force of gravity is measured on a horizontal line above a massive cube-shaped object. This cube contains a grid of 20×20 model parameters representing spatially varying density, \mathbf{m} (colors). The equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ embodies Newton's inverse-square law of gravity. (B) The $N=600$ gravitational force observations \mathbf{d}^{obs} (black curve) and the gravitational force predicted by the natural solution (red curve, $p=15$) and nonnegative least squares (green curve). (C) True model. (D) Natural estimate of model, with $p=4$. (E) Natural estimate of model, with $p=15$. (F) Nonnegative estimate of model. *MatLab* script gda07_06.

Thus, the solution that one obtains depends critically on the type of *a priori* information that one employs. In this case, the natural solution (Figure 7.7E), which contains 136 negative masses, is completely different from the nonnegative solution (Figure 7.7F), which contains none. Nevertheless, both satisfy the data almost exactly (Figure 7.7A), with the prediction error E of the nonnegative solution about 1% larger than that of the natural solution. Ironically, neither the nonnegative solution nor the natural solution looks at all like the true solution (Figure 7.7C), but a heavily damped solution (Figure 7.7D) does.

The nonnegative least squares algorithm can also be used to solve the problem (Lawson and Hanson, 1974)

$$\text{minimize } E = \|\mathbf{m}\|_2 \quad \text{subject to } \mathbf{H}\mathbf{m} \geq \mathbf{h} \quad (7.60)$$

and, by virtue of the transformation described in Section 7.9.1, the completely general problem. The method consists of forming the $(M+1) \times p$ equation

$$\mathbf{G}'\mathbf{m}' = \mathbf{d}' = \begin{bmatrix} \mathbf{H}^T \\ \mathbf{h}^T \end{bmatrix} \mathbf{m}' = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \quad (7.61)$$

and finding the \mathbf{m}' that minimizes $\|\mathbf{d}' - \mathbf{G}'\mathbf{m}'\|_2$ subject to $\mathbf{m}' \geq \mathbf{0}$ by the nonnegative least squares algorithm described above. If the prediction error $\mathbf{e}' = \mathbf{d}' - \mathbf{G}'\mathbf{m}'$ is identically zero, then the constraints $\mathbf{H}\mathbf{m} \geq \mathbf{h}$ are inconsistent. Otherwise, the solution is $m_i = -e'_i/e'_{M+1}$ (which can also be written as $\mathbf{e}' = [-\mathbf{m}, 1]^T e'_{M+1}$).

In *MatLab*, the solution is computed as

```
Gp = [H, h]';
dp = [zeros(1, length(H(1, :))), 1]';
mp = lsqnonneg(Gp, dp);
ep = dp - Gp*mp;
m = -ep(1:end-1)/ep(end);
```

(*MatLab* script gda07_07)

An example with $M=2$ model parameters and $N=3$ constraints is shown in Figure 7.8. The *MatLab* code for converting a *least squares with inequality constraints* problem into a *nonnegative least squares problem* using the procedure in Section 7.8.2 is:

```
[Up, Lp, Vp] = svd(G, 0);
lambda = diag(Lp);
rlambda = 1./lambda;
Lpi = diag(rlambda);

% transformation 1
Hp = -H*Vp*Lpi;
hp = h + Hp*Up'*dobs;
```

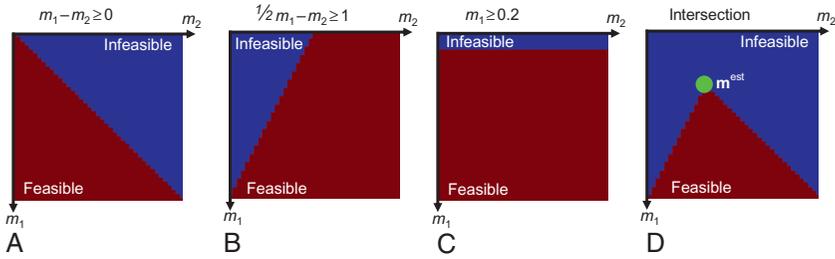


FIGURE 7.8 Exemplary solution of the problem of minimizing $\mathbf{m}^T \mathbf{m}$ with $N=3$ inequality constraints $\mathbf{H}\mathbf{m} \geq \mathbf{h}$. (A–C) Each constraint divides the (m_1, m_2) plane into two half-planes, one feasible and the other infeasible. (D) The intersection of the three feasible half-planes is polygonal in shape. The solution \mathbf{m}^{est} (green circle) is the point in feasible area that is closest to the origin. Note that two of the three constraints are satisfied in the equality sense. *MatLab* script `gda07_07`.

```
% transformation 2
Gp = [Hp, hp]';
dp = [zeros(1, length(Hp(1, :))), 1]';
mpp = lsqnonneg(Gp, dp);
ep = dp - Gp*mpp;
mp = -ep(1:end-1)/ep(end);

% take mp back to m
mest = Vp*Lpi*(Up'*dobs-mp);
dpre = G*mest;
```

(*MatLab* script `gda07_07`)

Note that `svd()` is called a second argument, set to zero, which causes it to compute \mathbf{U}_p rather than the default, the \mathbf{U} . An exemplary problem is illustrated in [Figure 7.9](#).

We now demonstrate that this method does indeed solve the indicated problem (adapted from [Lawson and Hanson, 1974](#)). Step 1 is to show that e'_{M+1} is nonnegative. We first note that the gradient of the error satisfies $1/2 \nabla E' = -\mathbf{G}'^T [\mathbf{d}' - \mathbf{G}' \mathbf{m}'] = -\mathbf{G}'^T \mathbf{e}'$, and that because of the Kuhn-Tucker theorem, \mathbf{m}' and $\nabla E'$ satisfy

$$\begin{aligned} \mathbf{m}'_E &= 0 & [\nabla E']_E &< 0 \\ \mathbf{m}'_S &> 0 & [\nabla E']_S &= 0 \end{aligned} \tag{7.62}$$

Note that these conditions imply $\mathbf{m}'^T \nabla E' = 0$. The length of the error E' is therefore

$$\begin{aligned} E' &= \mathbf{e}'^T \mathbf{e}' = [\mathbf{d}' - \mathbf{G}' \mathbf{m}']^T \mathbf{e}' = \mathbf{d}'^T \mathbf{e}' - \mathbf{m}'^T \mathbf{G}'^T \mathbf{e}' \\ &= e'_{M+1} + 1/2 \mathbf{m}'^T \nabla E' = e'_{M+1} \end{aligned} \tag{7.63}$$

as $\mathbf{d}'^T \mathbf{e}' = [\mathbf{0}, 1] \mathbf{e}' = e'_{M+1}$. The error E' is necessarily a nonnegative quantity, so if it is not identically zero, then e'_{M+1} must be greater than zero.

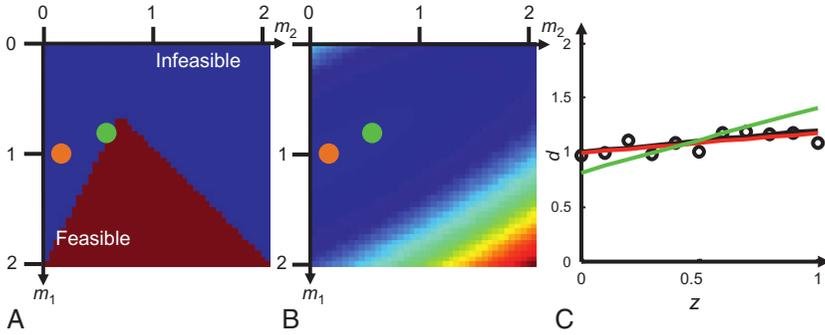


FIGURE 7.9 Problem of minimizing the prediction error E subject to inequality constraints, applied to the straight line problem. (A) Feasible region of the model parameters (the intercept m_1 and slope m_2 of the straight line). The unconstrained solution (orange circle) is outside the feasible region, but the constrained solution (green circle) is on its boundary. (B) The unconstrained solution (orange circle) is at the global minimum of the prediction error E , while the constrained solution is not. (C) Plot of the data $d(z)$ showing true data (black line), observed data (black circles), unconstrained prediction (red line), and constrained prediction (green line). *MatLab* script gda07_08.

Step 2 is to show that the solution satisfies the inequality constraints, $\mathbf{H}\mathbf{m} - \mathbf{h} \geq 0$. We start with the gradient of the error $\nabla E'$, which must have all nonnegative elements (or else the solution \mathbf{m}' could be further minimized without violating the constraints $\mathbf{m}' \geq 0$):

$$0 \leq \frac{1}{2} \nabla E' = -\mathbf{G}'^T \mathbf{e}' = -[\mathbf{H}, \mathbf{h}] [-\mathbf{m}, 1]^T e'_{M+1} = [\mathbf{H}\mathbf{m} - \mathbf{h}] e'_{M+1} \quad (7.64)$$

As $e'_{M+1} > 0$, we have $\mathbf{H}\mathbf{m} - \mathbf{h} \geq 0$.

Step 3 is to show that the solution minimizes $\|\mathbf{m}\|_2$. This follows from the Kuhn-Tucker condition that, at a valid solution, the gradient of the error ∇E be represented as a nonnegative combination of the rows of \mathbf{H} :

$$\nabla E = \nabla \|\mathbf{m}\|_2^2 = 2\mathbf{m} = -2[e'_1 \cdots e'_M]^T / e'_{M+1} = 2\mathbf{H}^T [m'_1 \cdots m'_M]^T / e'_{M+1} \quad (7.65)$$

Here we have used the fact that $\mathbf{e}' = \mathbf{d}' - \mathbf{G}'\mathbf{m}'$, together with the fact that the first M elements of \mathbf{d}' are zero. Note that \mathbf{m}' and e'_{M+1} are nonnegative, so the Kuhn-Tucker condition is satisfied.

Finally, we can also show that a feasible solution exists only when the error is not identically zero. Consider the contradiction, that the error is identically zero but that a feasible solution exists. Then

$$0 = \mathbf{e}'^T \mathbf{e}' / e'_{M+1} = [-\mathbf{m}'^T, 1][\mathbf{d}' - \mathbf{G}'\mathbf{m}'] = 1 + [\mathbf{H}\mathbf{m} - \mathbf{h}]^T \mathbf{m}' \quad (7.66)$$

As $\mathbf{m}' \geq 0$, the relationship $\mathbf{H}\mathbf{m} < \mathbf{h}$ is implied. This contradicts the constraint equations $\mathbf{H}\mathbf{m} \geq \mathbf{h}$ so that an identically zero error implies that no feasible solution exists and that the constraints are inconsistent.

7.10 PROBLEMS

- 7.1** This problem builds upon the discussion in Section 6.2. Use *MatLab*'s `svd()` function to compute the null vectors associated with the data kernel $\mathbf{G} = [1, 1, 1, 1]$. (A) The null vectors are different from given in Equation (6.5). Why? Show that the two sets are equivalent.
- 7.2** Modify the weighted damped least squares “gap-filling” problem of Figure 3.10 (*MatLab* script `gda03_09`) so that the *a priori* information is applied only to the part of the solution in the null space. First, transform the weighted problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ into an unweighted one $\mathbf{G}'\mathbf{m}' = \mathbf{d}'$ using the transformation given by Equation (7.23). Then, find the minimum-length solution (or the natural solution) \mathbf{m}'^{est} . Finally, transform back to obtain \mathbf{m}^{est} . How different is this solution from the one given in the figure? Compare the two prediction errors. In order to insure that $\mathbf{W}_e = \mathbf{D}^T\mathbf{D}$ has an inverse, which is required by the transformation, you should make \mathbf{D} square by adding two rows, one at the top and the other at the bottom, that constrain the first and last model parameters to known values.
- 7.3** (A) Compute and plot the null vectors for the data kernel shown in Figure 7.4A. (B) Suppose that the elements of \mathbf{m}^{true} are drawn from a uniform distribution between 0 and 1. How large a contribution do the null vectors make to the true solution \mathbf{m}^{true} ? (Hint: Write the model parameters as a linear combination of all the eigenvectors \mathbf{V} by writing $\mathbf{m}^{\text{true}} = \mathbf{V}\mathbf{a}$, where \mathbf{a} are coefficients, and then solving for \mathbf{a} . Then examine the size of the elements of \mathbf{a} . You may wish to refer to *MatLab* script `gda07_03` for the definition of \mathbf{G}).
- 7.4** Consider fitting a cubic polynomial $d_i = m_1 + m_2 z_i + m_3 z_i^2 + m_4 z_i^3$ to $N = 20$ data d_i^{obs} , where the z s are evenly spaced on the interval (0,1), where $m_2 = m_3 = m_4 = 1$, and where m_1 is varied from -1 to 1 , as described below. (A) Write a *MatLab* script that generates synthetic observed data (including Gaussian-distributed noise) for a particular choice of m_1 , estimates the model parameters using both simple least squares and nonnegative least squares, plots the observed and predicted data, and outputs the total error. (B) Run a series of test cases for a range of values of m_1 and comment upon the results.
- 7.5** (A) Modify the *MatLab* `gda07_07` script so that the last constraint is $m_1 \geq 1.2$. How do the feasible region and the solution change? (B) Modify the script to add a constraint $m_2 \geq 0.2$. How do the feasible region and the solution change?

REFERENCES

- Lanczos, C., 1961. *Linear Differential Operators*. Van Nostrand-Reinhold, Princeton, New Jersey.
- Lawson, C.L., Hanson, D.J., 1974. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, New Jersey.
- Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc., Oxford, UK 263pp.
- Penrose, R.A., 1955. A generalized inverse for matrices. *Proc. Cambridge Phil. Soc.* 51, 406–413.
- Wiggins, R.A., 1972. The general linear inverse problem: implication of surface waves and free oscillations for Earth structure. *Rev. Geophys. Space Phys.* 10, 251–285.
- Wunsch, C., Minster, J.F., 1982. Methods for box models and ocean circulation tracers: mathematical programming and non-linear inverse theory. *J. Geophys. Res.* 87, 5647–5662.

Linear Inverse Problems and Non-Gaussian Statistics

8.1 L_1 NORMS AND EXPONENTIAL PROBABILITY DENSITY FUNCTIONS

In [Chapter 5](#), we showed that the method of least squares and the more general use of L_2 norms could be rationalized through the assumption that the data and *a priori* model parameters followed Gaussian statistics. This assumption is not always appropriate; however, some data sets follow other distributions. The *two-sided exponential probability density function* is one simple alternative. Here “two-sided” means that the function is defined on the interval $(-\infty, +\infty)$ and has tails in both directions. When Gaussian and exponential distributions of the same mean $\langle d \rangle$ and variance σ^2 are compared, the exponential distribution is found to be much longer tailed ([Figure 8.1](#), [Table 8.1](#)):

$$p(d) = \frac{1}{(2\pi)^{1/2} \sigma} \exp \left\{ -\frac{1}{2} \frac{(d - \langle d \rangle)^2}{\sigma^2} \right\} \quad \text{Gaussian} \quad p(d) = \frac{1}{(2)^{1/2} \sigma} \exp \left\{ -(2)^{1/2} \frac{|d - \langle d \rangle|}{\sigma} \right\} \quad \text{exponential} \quad (8.1)$$

Note that the probability of realizing data far from $\langle d \rangle$ is much higher for the exponential probability density function than for the Gaussian probability density function. A few data, say, 4 standard deviations from the mean, are reasonably probable in a data set of, say, 1000 samples drawn from an exponential probability density function, but very improbable for data drawn from a Gaussian probability density function. We therefore expect that methods based on the exponential probability density function will be able to handle a data set with a few “bad” data (outliers) better than Gaussian methods. Methods that can tolerate a few outliers are said to be *robust* ([Claerbout and Muir, 1973](#)).

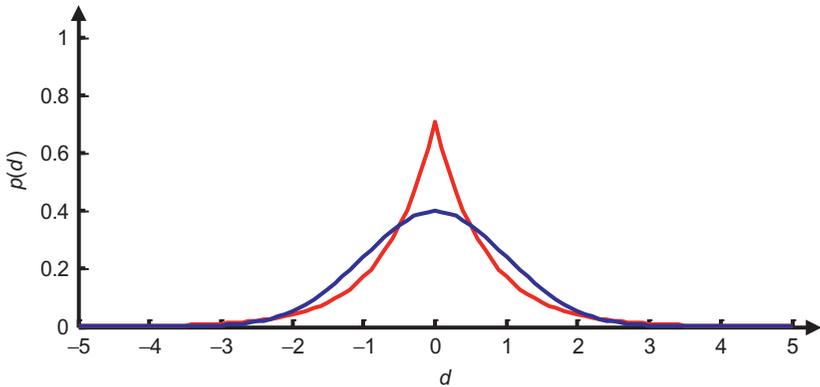


FIGURE 8.1 Comparison of exponential probability density distribution (red) with a Gaussian probability density distribution (blue) of equal variance, $\sigma^2=1$. The exponential probability density distribution is longer tailed. *MatLab* script gda08_01.

TABLE 8.1 The area beneath $\langle d \rangle \pm n\sigma$ for the Gaussian and exponential distributions

	Gaussian	Exponential
n	<i>Area (%)</i>	<i>Area (%)</i>
1	68.2	76
2	95.4	94
3	99.7	98.6
4	99.999+	99.7

In *MatLab*, the exponential probability density function with mean \bar{d} and variance sd^2 is calculated as

$$pE = (1/\sqrt{2}) * (1/\text{sd}) * \exp(-\sqrt{2} * \text{abs}((d-\bar{d})/\text{sd}));$$

(MatLab script gda08_01)

MatLab's `random()` function provides only a one-sided version of the exponential probability density function, defined on the interval $(0, +\infty)$, but a two-sided version can be created by multiplication of its realizations by a random sign, created here from realizations of the discrete uniform probability density function

```
mu = sd/sqrt(2);
rsign = (2*(random('unid', 2, Nr, 1)-1)-1);
dr = dbar + rsign .* random('exponential', mu, Nr, 1);
```

(MatLab script gda08_01)

8.2 MAXIMUM LIKELIHOOD ESTIMATE OF THE MEAN OF AN EXPONENTIAL PROBABILITY DENSITY FUNCTION

Exponential probability density functions bear the same relationship to L_1 norms as Gaussian probability density function bear to L_2 norms. To illustrate this relationship, we consider the joint distribution for N independent data, each with the same mean m_1 and variance σ^2 . Since the data are independent, the joint probability density function is just the product of N univariate functions:

$$p(d) = (2)^{-N/2} \sigma^{-N} \exp \left[-\frac{(2)^{1/2}}{\sigma} \sum_{i=1}^N |d_i - m_1| \right] \quad (8.2)$$

To maximize the likelihood of $p(d)$, we must maximize the argument of the exponential, which involves minimizing the sum of absolute residuals as

$$\text{minimize } E = \sum_{i=1}^N |d_i - m_1| \quad (8.3)$$

This is the L_1 norm of the prediction error in a linear inverse problem of the form $\mathbf{G}\mathbf{m} = \mathbf{d}$, where $M = 1$, $\mathbf{G} = [1, 1, \dots, 1]^T$, and $\mathbf{m} = [m_1]$. Applying the principle of maximum likelihood, we obtain

$$\text{maximize } L = \log P = -\frac{N}{2} \log(2) - N \log(\sigma) - \frac{(2)^{1/2}}{\sigma} \sum_{i=1}^N |d_i - m_1| \quad (8.4)$$

Setting the derivatives to zero yields

$$\begin{aligned} \frac{\partial L}{\partial m_1} &= 0 = \frac{(2)^{1/2}}{\sigma} \sum_{i=1}^N \text{sign}(d_i - m_1) \\ \frac{\partial L}{\partial \sigma} &= 0 = \frac{N}{\sigma} - \frac{(2)^{1/2}}{\sigma^2} \sum_{i=1}^N |d_i - m_1| \end{aligned} \quad (8.5)$$

The sign function $\text{sign}(x)$ equals $+1$ if $x > 0$, -1 if $x < 0$ and 0 if $x = 0$. Note that the derivative $d|x|/dx = \text{sign}(x)$, since if x is positive, then it is just equal to $dx/dx = 1$ and if it is negative, then to -1 . The first equation yields the implicit expression for $m_1 = \langle d \rangle^{\text{est}}$ for which $\sum_i \text{sign}(d_i - m_1) = 0$. The second equation can then be solved for an estimate of the variance as

$$\sigma^{\text{est}} = \frac{(2)^{1/2}}{N} \sum_{i=1}^N |d_i - m_1| \quad (8.6)$$

The equation for m_1^{est} is exactly the sample median; one finds an m_1 such that half the d s are less than m_1 and half are greater than m_1 . There are then an equal number of negative and positive signs and the sum of the signs is zero. The median is a robust property of a set of data. Adding one outlier can at worst move

the median from one central datum to another nearby central datum. While the maximum likelihood estimate of a Gaussian distribution's true mean is the sample arithmetic mean, the maximum likelihood estimate of an exponential distribution's true mean is the sample median.

The estimate of the variance also differs between the two distributions: in the Gaussian case, it is the square of the sample standard deviation, but in the exponential case, it is not. If there are an odd number of samples, then m_1^{est} equals the middle d_i . If there is an even number of samples, any m_1^{est} between the two middle data maximizes the likelihood. In the odd case, the error E attains a minimum only at the middle sample, but in the even case, it is flat between the two middle samples (Figure 8.2). We see, therefore, that L_1 problems of minimizing the prediction error of $\mathbf{Gm} = \mathbf{d}$ can possess nonunique solutions that are distinct from the type of nonuniqueness encountered in the L_2 problems. The L_1 problems can still possess nonuniqueness owing to the existence of null solutions since the null solutions cannot change the prediction error under any norm. That kind of nonuniqueness leads to a completely unbounded range of estimates. The new type of nonuniqueness, on the other hand, permits the solution to take on any values between *finite bounds*.

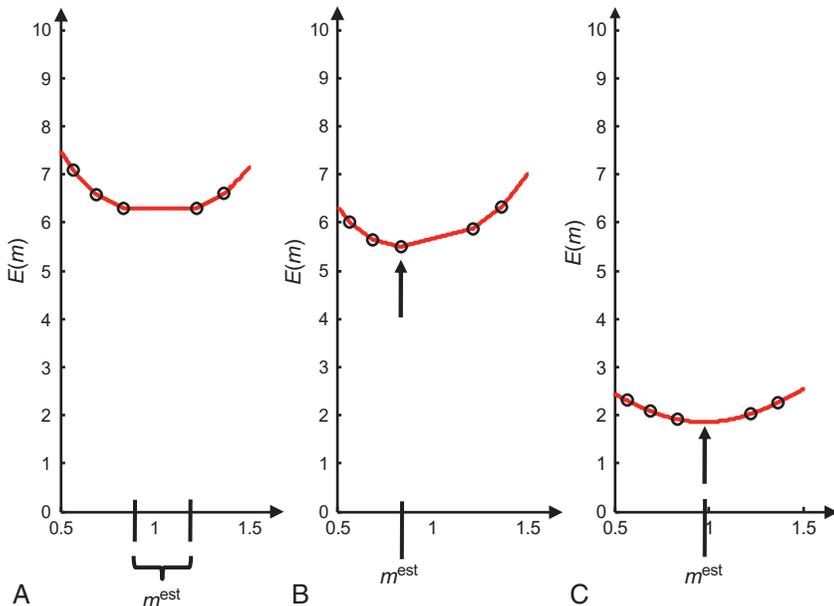


FIGURE 8.2 Inverse problem to estimate the mean m^{est} of N observations. (A) L_1 error $E(m)$ (red curve) with even N . The error has a flat minimum, bounded by two observations (circles), and the solution is nonunique. (B) L_1 error with odd N . The error is minimum at one of the observation points, and the solution is unique. (C) The L_2 error, both odd and even N . The error is minimum at a single point, which may not correspond to an observation point, and the solution is unique. *MatLab* script gda08_02.

We also note that regardless of whether N is even or odd, we can choose m_1^{est} so that one of the equations $\mathbf{G}\mathbf{m} = \mathbf{d}$ is satisfied exactly (in this case, $m_1^{\text{est}} = d_k$, where k is the index of the “middle” datum). This can be shown to be a general property of L_1 norm problems. Given N equations and M unknowns related by $\mathbf{G}\mathbf{m} = \mathbf{d}$, it is possible to choose \mathbf{m} so that the L_1 prediction error is minimized and so that M of the equations are satisfied exactly.

In *MatLab*, the L_1 estimate of the mean $\langle d \rangle$ and square root of the variance σ are computed as

```
dbarest = median(dr);
sdest = (sqrt(2)/Nr) * sum(abs(dr - dbarest));
```

(*MatLab* script gda08_01)

Here dr is a vector of the data of length N_{r} .

8.3 THE GENERAL LINEAR PROBLEM

Consider the linear inverse problem $\mathbf{G}\mathbf{m} = \mathbf{d}$ in which the data and *a priori* model parameters are uncorrelated with known means \mathbf{d}^{obs} and $\langle \mathbf{m} \rangle$ and known variances σ_d^2 and σ_m^2 , respectively. Their joint distribution is then

$$p(\mathbf{d}, \mathbf{m}) = 2^{-(M+N)/2} \prod_{i=1}^N \sigma_{d_i}^{-1} \exp \left[-2^{1/2} \frac{|e_i|}{\sigma_{d_i}} \right] \prod_{j=1}^M \sigma_{m_j}^{-1} \exp \left[-2^{1/2} \frac{|l_j|}{\sigma_{m_j}} \right] \quad (8.7)$$

where the prediction error is given by $\mathbf{e} = \mathbf{d} - \mathbf{G}\mathbf{m}$ and the solution length by $\mathbf{l} = \mathbf{m} - \langle \mathbf{m} \rangle$. The maximum likelihood estimate of the model parameters occurs when the exponential is a minimum, that is, when the sum of the weighted L_1 prediction error and the weighted L_1 solution length is minimized:

$$\text{minimize} \quad E + L = \sum_{i=1}^N \frac{|e_i|}{\sigma_{d_i}} + \sum_{j=1}^M \frac{|l_j|}{\sigma_{m_j}} \quad (8.8)$$

In this case, the weighting factors are the reciprocals of the square root of the variance, in contrast to the Gaussian case, in which they are the reciprocals of the variances. Note that linear combinations of exponentially distributed random variables are not themselves exponential (unlike Gaussian variables, which give rise to Gaussian combinations). The covariance matrix of the estimated model parameters is, therefore, difficult both to calculate and to interpret since the manner in which it is related to confidence intervals varies from case to case. We shall focus our discussion on estimating only the model parameters themselves.

8.4 SOLVING L_1 NORM PROBLEMS

We shall show that this problem can be transformed into a “linear programming” problem (see [Section 6.6](#))

$$\begin{aligned} &\text{find } \mathbf{x} \text{ that minimizes } z = \mathbf{f}^T \mathbf{x} \\ &\text{with the constraints } \mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \text{and} \quad \mathbf{C}\mathbf{x} = \mathbf{d} \quad \text{and} \quad \mathbf{x}^{(l)} \leq \mathbf{x} \leq \mathbf{x}^{(u)} \end{aligned} \quad (8.9)$$

We first define the L_1 versions of the weighted solution length L and the weighted prediction error E

$$L = \sum_{i=1}^M \frac{|m_i - \langle m_i \rangle|}{\sigma_{m_i}} \quad \text{and} \quad E = \sum_{i=1}^N \frac{|d_i^{\text{obs}} - d_i^{\text{pre}}|}{\sigma_{d_i}} \quad \text{with} \quad \mathbf{d}^{\text{pre}} = \mathbf{G}\mathbf{m} \quad (8.10)$$

Note that these formulas are weighted by the square root of the variances of the *a priori* model parameters and measurement error (σ_m and σ_d , respectively).

First, we shall consider the completely underdetermined linear problem with *a priori* model parameters, mean $\langle \mathbf{m} \rangle$, and variance σ_m^2 . The problem is to minimize the weighted length L subject to the constraint $\mathbf{G}\mathbf{m} = \mathbf{d}$. We first introduce $5M$ new variables $m'_i, m''_i, \alpha_i, x_i$, and x'_i , where $i = 1, \dots, M$. The linear programming problem may be stated as follows (Cuer and Bayer, 1980)

$$\text{minimize } z = \sum_{i=1}^M \frac{\alpha_i}{\sigma_{m_i}} \quad \text{subject to the constraints}$$

$$\mathbf{G}(\mathbf{m}' - \mathbf{m}'') = \mathbf{d} \quad \text{and} \quad \mathbf{m}' - \mathbf{m}'' + \mathbf{x}_i - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \quad \mathbf{m}' - \mathbf{m}'' - \mathbf{x}' - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle$$

and

$$\mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0 \quad (8.11)$$

This linear programming problem has $5M$ unknowns, $N + 2M$ equality constraints and $5M$ inequality constraints. If one makes the identification $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$, the first equality constraint is equivalent to $\mathbf{G}\mathbf{m} = \mathbf{d}$. The signs of the elements of \mathbf{m} are not constrained even though those of \mathbf{m}' and \mathbf{m}'' are. The remaining equality constraints can be rewritten as

$$\boldsymbol{\alpha} - \mathbf{x} = [\mathbf{m} - \langle \mathbf{m} \rangle] \quad \text{and} \quad \boldsymbol{\alpha} - \mathbf{x}' = -[\mathbf{m} - \langle \mathbf{m} \rangle] \quad (8.12)$$

where the α_i, x_i , and x'_i are nonnegative. Now if $[m_i - \langle m_i \rangle]$ is positive, the first equation requires $\alpha_i \geq [m_i - \langle m_i \rangle]$ since x_i cannot be negative. The second constraint can always be satisfied by choosing some appropriate x'_i . On the other hand, if $[m_i - \langle m_i \rangle]$ is negative, then the first constraint can always be satisfied by choosing some appropriate x_i , but the second constraint requires that $\alpha_i \geq -[m_i - \langle m_i \rangle]$. Taken together, these two constraints imply that $\alpha_i \geq |[m_i - \langle m_i \rangle]|$. Minimizing $\sum \alpha_i / \sigma_{m_i}$ is therefore equivalent to minimizing the weighted solution length L . The L_1 minimization problem has been converted to a linear programming problem.

The completely overdetermined problem of minimizing E with no *a priori* information can be converted into a linear programming problem in a similar

manner. We introduce $2M + 3N$ new variables, $m'_i, m''_i, i = 1, M$ and $\alpha_i, x_i, x'_i, i = 1, N, 2N$ equality constraints and $2M + 3N$ inequality constraints. The equivalent linear programming problem is (Cuer and Bayer, 1980):

$$\text{minimize } E = \sum_{i=1}^N \frac{\alpha_i}{\sigma_{d_i}} \text{ subject to the constraints}$$

$$\mathbf{G}[\mathbf{m}' - \mathbf{m}''] + \mathbf{x} - \boldsymbol{\alpha} = \mathbf{d} \quad \text{and} \quad \mathbf{G}[\mathbf{m}' - \mathbf{m}''] - \mathbf{x}' + \boldsymbol{\alpha} = \mathbf{d} \quad (8.13)$$

and

$$\mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0$$

In *MatLab*, the solution is computed as

```
% L1 solution to overdetermined problem
% linear programming problem is
% min f*x subject to A x <= b, Aeq x = beq

% variables
% m=mp - mpp
% x = [mp', mpp', alpha', x', xp']'
% with mp, mpp length M and alpha, x, xp, length N
L = 2*M + 3*N;
x = zeros(L, 1);

f = zeros(L, 1);
f(2*M + 1 : 2*M + N) = 1./sd;

% equality constraints
Aeq = zeros(2*N, L);
beq = zeros(2*N, 1);

% first equation G(mp-mpp)+x-alpha = d
Aeq(1:N, 1:M) = G;
Aeq(1:N, M+1:2*M) = -G;
Aeq(1:N, 2*M+1:2*M+N) = -eye(N, N);
Aeq(1:N, 2*M+N+1:2*M+2*N) = eye(N, N);
beq(1:N) = dobs;

% second equation G(mp-mpp)-xp+alpha=d
Aeq(N+1:2*N, 1:M) = G;
Aeq(N+1:2*N, M+1:2*M) = -G;
Aeq(N+1:2*N, 2*M+1:2*M+N) = eye(N, N);
Aeq(N+1:2*N, 2*M+2*N+1:2*M+3*N) = -eye(N, N);
beq(N+1:2*N) = dobs;

% inequality constraints A x <= b
% part 1: everything positive
```

```

A = zeros(L+2*M,L);
b = zeros(L+2*M,1);
A(1:L, :) = -eye(L,L);
b(1:L) = zeros(L,1);

% part 2; mp and mpp have an upper bound.
A(L+1:L+2*M, :) = eye(2*M,L);
mls = (G'*G)\(G'*dobs); % L2 solution - sure easier!
mupperbound=10*max(abs(mls));
b(L+1:L+2*M) = mupperbound;

% solve linear programming problem
[x, fmin] = linprog(f,A,b,Aeq,beq);
fmin=-fmin;
mest = x(1:M) - x(M+1:2*M);

```

(*MatLab* script gda08_03)

Note that the *MatLab* implementation adds upper bounds for \mathbf{m}' and \mathbf{m}'' , 10 times the largest element of the least-squares solution (an amount that might need to be adjusted for the problem at hand). Without this constraint, the algorithm could possibly return very large values for these variables, leading to a solution $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$ that is inaccurate because of round-off error. An example of the L_1 problem applied to curve fitting is shown in [Figure 8.3](#).

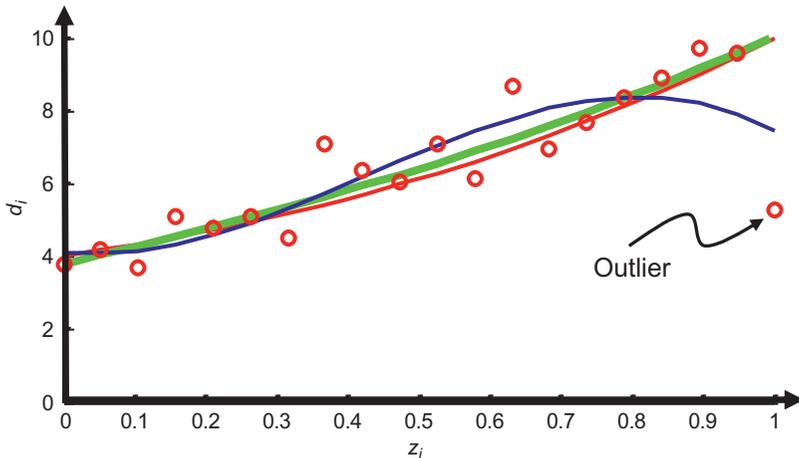


FIGURE 8.3 Curve fitting using the L_1 norm. The true data (red curve) follow a cubic polynomial in an auxiliary variable, z . Observations (red circles) have additive noise with zero mean and variance $\sigma^2 = 1$ drawn from an exponential probability density function. Note the outlier at $z \approx 1$. The L_1 fit (green curve) is not as affected by the outlier as a standard L_2 (least-squares) fit (blue curve). *MatLab* script gda08_03.

The mixed-determined problem can be solved by any of several methods. By analogy to the L_2 methods described in Chapters 3 and 7, we could either pick some *a priori* model parameters and minimize $E + L$ or separate the overdetermined model parameters from the underdetermined ones and apply *a priori* information to the underdetermined ones only. The first method leads to a linear programming problem similar to the two cases stated above but with even more variables ($5M + 3N$), equality constraints ($2M + 2N$) and inequality constraints ($5M + 2N$):

$$\begin{aligned} & \text{minimize} \quad \sum_{i=1}^M \frac{\alpha_i}{\sigma_{m_i}} + \sum_{i=1}^N \frac{\alpha'_i}{\sigma_{d_i}} \quad \text{subject to the constraints} \\ & [\mathbf{m}' - \mathbf{m}''] + \mathbf{x} - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \quad [\mathbf{m}' - \mathbf{m}''] - \mathbf{x}' + \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \\ & \mathbf{G}[\mathbf{m}' - \mathbf{m}''] + \mathbf{x}'' - \boldsymbol{\alpha}' = \mathbf{d} \quad \text{and} \quad \mathbf{G}[\mathbf{m}' - \mathbf{m}''] - \mathbf{x}'' + \boldsymbol{\alpha}' = \mathbf{d} \quad \text{and} \\ & \mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \boldsymbol{\alpha}' \geq 0 \\ & \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0 \quad \text{and} \quad \mathbf{x}'' \geq 0 \quad \text{and} \quad \mathbf{x}''' \geq 0 \end{aligned} \quad (8.14)$$

The second method is more interesting. First, we use the singular-value decomposition to identify the null space of \mathbf{G} . The solution then has the form

$$\mathbf{m}^{\text{est}} = \sum_{i=1}^p a_i \mathbf{v}_p^{(i)} + \sum_{i=p+1}^M b_i \mathbf{v}_0^{(i)} = \mathbf{V}_p \mathbf{a} + \mathbf{V}_0 \mathbf{b} \quad (8.15)$$

where the \mathbf{v} s are eigenvectors and \mathbf{a} and \mathbf{b} are vectors of unknown coefficients. Only the vector \mathbf{a} can affect the prediction error, so one uses the overdetermined algorithm to determine it

$$\text{find } \mathbf{a} \text{ that minimizes} \quad E = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_1 = \sum_{i=1}^N \frac{|d_i - [\mathbf{U}_p \boldsymbol{\Lambda}_p \mathbf{a}]_i|}{\sigma_{d_i}} \quad (8.16)$$

Next, one uses the underdetermined algorithm to determine \mathbf{b} :

$$\text{find } \mathbf{b} \text{ that minimizes} \quad E = \|\mathbf{m} - \langle \mathbf{m} \rangle\|_1 = \sum_{i=1}^N \frac{|[\mathbf{V}_0 \mathbf{b}]_i - (\langle m_i \rangle - [\mathbf{V}_p \mathbf{a}]_i)|}{\sigma_{m_i}} \quad (8.17)$$

It is also possible to implement the basic underdetermined and overdetermined L_1 algorithms in such a manner that the many extra variables are never explicitly calculated (Cuer and Bayer, 1980). This procedure vastly decreases the storage and computation time required, making these algorithms practical for solving moderately large ($M = 1000$) inverse problems.

8.5 THE L_∞ NORM

While the L_1 norm weights “bad” data less than the L_2 norm, the L_∞ norm weights it more:

$$\text{minimize } L + E = \|\mathbf{e}\|_\infty + \|\mathbf{l}\|_\infty = \max_i \frac{|e_i|}{\sigma_{d_i}} + \max_i \frac{|l_i|}{\sigma_{m_i}} \quad (8.18)$$

The prediction error and solution length are weighted by the reciprocal of their *a priori* standard deviations. Normally, one does not want to emphasize outliers, so the L_∞ form is useful mainly in that it can provide a “worst-case” estimate of the model parameters for comparison with estimates derived on the basis of other norms. If the estimates are in fact close to one another, one can be confident that the data are highly consistent. Since the L_∞ estimate is controlled only by the worst error or length, it is usually nonunique.

The general linear equation $\mathbf{G}\mathbf{m} = \mathbf{d}$ can be solved in the L_∞ sense by transformation into a linear programming problem using a variant of the method used to solve the L_1 problem. In the underdetermined problem, we again introduce new variables, m'_i , m''_i , x_i , and x'_i , each of length M , and a single parameter α ($4M + 1$ variables, total). The linear programming problem is

$$\begin{aligned} &\text{minimize } \alpha \quad \text{subject to the constraints} \\ &\mathbf{G}[\mathbf{m}' - \mathbf{m}''] = \mathbf{d} \quad \text{and} \\ &[m'_i - m''_i]_i + x_i - \alpha\sigma_{m_i} = \langle m_i \rangle \quad \text{and} \quad [m'_i - m''_i]_i - x'_i + \alpha\sigma_{m_i} = \langle m_i \rangle \\ &\text{and } \mathbf{m}' \geq 0 \quad \text{and } \mathbf{m}'' \geq 0 \quad \text{and } \alpha \geq 0 \quad \text{and } \mathbf{x} \geq 0 \quad \text{and } \mathbf{x}' \geq 0 \end{aligned} \quad (8.19)$$

where $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$. We note that the new constraints can be written as

$$\alpha - \frac{x_i}{\sigma_{m_i}} = \frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \quad \text{and} \quad \alpha - \frac{x'_i}{\sigma_{m_i}} = -\frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \quad (8.20)$$

where α , x_i , and x'_i are nonnegative. Using the same argument as was applied in the L_1 case, we conclude that these constraints require that $\alpha \geq |m_i - \langle m_i \rangle|/\sigma_{m_i}$ for all i . Since this problem has but a single parameter α , it must therefore satisfy

$$\alpha \geq \max_i \left| \frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \right| \quad (8.21)$$

Minimizing α yields the L_∞ solution.

The linear programming problem for the overdetermined case is

$$\begin{aligned} &\text{minimize } \alpha \quad \text{subject to the constraints} \\ &\sum_{j=1}^M G_{ij} [m'_j - m''_j] + x_i - \alpha\sigma_{d_i} = d_i \quad \text{and} \quad \sum_{j=1}^M G_{ij} [m'_j - m''_j] - x'_i + \alpha\sigma_{d_i} = d_i \\ &\text{and } \mathbf{m}' \geq 0 \quad \text{and } \mathbf{m}'' \geq 0 \quad \text{and } \alpha \geq 0 \quad \text{and } \mathbf{x} \geq 0 \quad \text{and } \mathbf{x}' \geq 0 \end{aligned} \quad (8.22)$$

Here $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$, where \mathbf{m}' and \mathbf{m}'' are two unknown vectors of length M , and \mathbf{x} and \mathbf{x}' are two unknown vectors of length N . The *MatLab* implementation is

```
% Linf solution to overdetermined problem
% linear programming problem is
% min f*x subject to A x <= b, Aeq x = beq

% variables
% m = mp - mpp
% x = [mp', mpp', alpha', x', xp']'
% with mp, mpp length M; alpha length 1, x, xp, length N
L = 2*M+1+2*N;
x = zeros(L,1);

% f is length L
% minimize alpha
f = zeros(L,1);
f(2*M+1:2*M+1)=1;

% equality constraints
Aeq = zeros(2*N,L);
beq = zeros(2*N,1);

% first equation G(mp-mpp)+x-alpha=d
Aeq(1:N,1:M) = G;
Aeq(1:N,M+1:2*M) = -G;
Aeq(1:N,2*M+1:2*M+1) = -1./sd;
Aeq(1:N,2*M+1+1:2*M+1+N) = eye(N,N);
beq(1:N) = dobs;

% second equation G(mp-mpp)-xp+alpha=d
Aeq(N+1:2*N,1:M) = G;
Aeq(N+1:2*N,M+1:2*M) = -G;
Aeq(N+1:2*N,2*M+1:2*M+1) = 1./sd;
Aeq(N+1:2*N,2*M+1+N+1:2*M+1+2*N) = -eye(N,N);
beq(N+1:2*N) = dobs;

% inequality constraints A x <= b
% part 1: everything positive
A = zeros(L+2*M,L);
b = zeros(L+2*M,1);
A(1:L,:) = -eye(L,L);
b(1:L) = zeros(L,1);

% part 2; mp and mpp have an upper bound
A(L+1:L+2*M,:) = eye(2*M,L);
mls = (G'*G)\(G'*dobs); % L2 solution - sure easier!
```

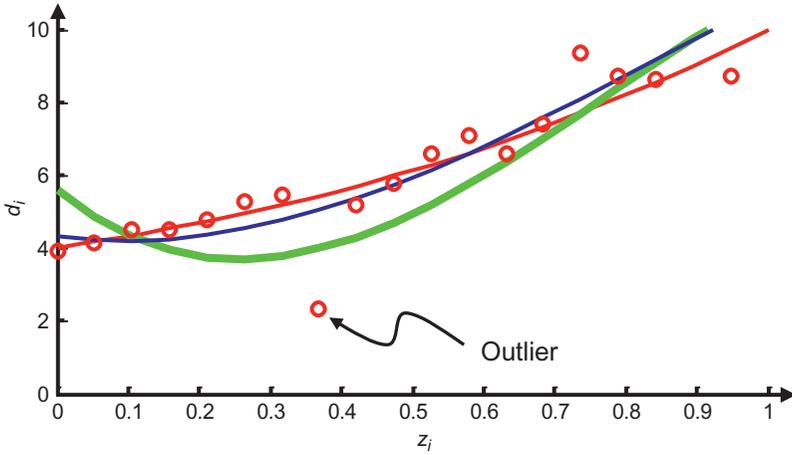


FIGURE 8.4 Curve fitting using the L_∞ norm. The true data (red curve) follow a cubic polynomial in an auxiliary variable z . Observations (red circles) have additive noise with zero mean and variance $\sigma^2 = 1$ drawn from an exponential probability density function. Note the outlier at $z \approx 0.37$. The L_∞ fit (green curve) is more affected by the outlier than is a standard L_2 (least-squares) fit (blue curve). *MatLab* script gda08_04.

```

mupperbound=10*max(abs(mls));
b(L+1:L+2*M) = mupperbound;

% solve linear programming problem
[x, fmin] = linprog(f,A,b,Aeq,beq);
fmin = -fmin;
mest = x(1:M) - x(M+1:2*M);

```

(*MatLab* script gda08_04)

As in the L_1 case, this implementation adds upper bounds on the variables \mathbf{m}' and \mathbf{m}'' . An example of the L_1 problem applied to curve fitting is shown in Figure 8.4.

The mixed-determined problem can be solved by applying these algorithms and either of the two methods described for the L_1 problem.

8.6 PROBLEMS

- 8.1. Solve the best-fitting plane problem of Figure 3.6 under the L_1 norm and compare the estimated model parameters with those determined under the L_2 norm. How much does the estimated *strike and dip* of the plane change?
- 8.2. Solve the constrained best-fitting line problem of Figure 3.11 under the L_1 norm, by these two methods: (A) by considering the point (z', d') as normal data with very small variance and (B) by explicitly including the constraint as a

linear equality constraint within the linear programming problem. Compare the estimated model parameters with those determined under the L_2 norm.

- 8.3.** This problem builds upon Problem 7.4. Consider fitting a cubic polynomial $d_i = m_1 + m_2 z_i + m_3 z_i^2 + m_4 z_i^3$ to $N=20$ data d_i^{obs} , where the z s are evenly spaced on the interval $(0,1)$, where $m_2 = m_3 = m_4 = 1$, and where m_1 is varied from -1 to 1 , as described below. (A) Write a *MatLab* script that generates synthetic observed data (including exponential-distributed noise) for a particular choice of m_1 , estimates the model parameters under the L_1 norm, with the extra inequality constraint that $\mathbf{m} \geq 0$. (B) Run a series of test cases for a range of values of m_1 and comment upon the results.

REFERENCES

- Claerbout, J.F., Muir, F., 1973. Robust modelling with erratic data. *Geophysics* 38, 826–844.
- Cuer, M., Bayer, R., 1980. FORTRAN routines for linear inverse problems. *Geophysics* 45, 1706–1719.

Nonlinear Inverse Problems

9.1 PARAMETERIZATIONS

Variables representing the data and model parameters—the *parameterization*—must be selected at the start of any inverse problem. In many instances, this selection is rather *ad hoc*; there might be no strong reasons for selecting one parameterization over another. This can become a substantial issue in nonlinear inverse problems because the answer obtained by solving it is dependent on the parameterization. In other words, the solution is not invariant under nonlinear transformations of the variables. This is in contrast to the linear inverse problem with Gaussian statistics, in which solutions are invariant for any linear reparameterization of the data and model parameters.

As an illustration of this difficulty, consider the problem of fitting a straight line to the data pairs (1, 1), (2, 2), (3, 3), (4, 5). Suppose that we regard these data as (z, d) pairs, where z is an auxiliary variable. The least-squares fit is $d = -0.500 + 1.300z$. On the other hand, we might regard them as (d', z') pairs, where z' is the auxiliary variable. Least squares then gives $d' = 0.457 + 0.743z'$, which can be rearranged as $z' = -0.615 + 1.346d'$. These two straight lines have intercepts that differ by 20% and slopes that differ by 4% (see *MatLab* script `gda09_01`).

This discrepancy arises from two sources. The first is an inconsistent application of probability theory. In the example above, we alternately assumed that z was exactly known but d contained Gaussian noise and that d was exactly known but z contained Gaussian noise. These are two radically different assumptions about the distribution of errors, so it is no wonder that the solutions are different.

This first source of discrepancy can in theory be avoided completely by recognizing and taking into account the fact that reparameterizing a problem changes the associated probability density functions. For instance, consider an inverse problem in which there is a model parameter m that is known to possess a uniform probability density function $p(m)$ on the interval $[0, 1]$ (Figure 9.1A). If the inverse problem is reparameterized in terms of a new model parameter $m' = m^2$, then the transformed probability density function $p(m')$ can be calculated as

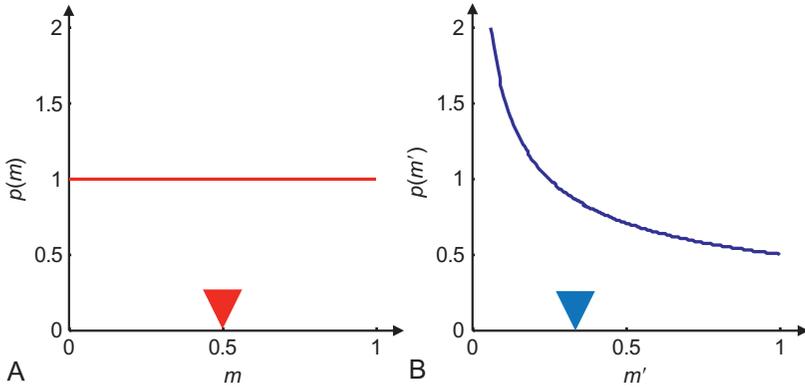


FIGURE 9.1 (A) A probability density function, $p(m)$, that is uniform on the interval $0 < m < 1$. (B) The corresponding probability distribution, $p(m')$, for the transformation $m' = m^2$. The mean (expectation) of each probability density function is indicated by a triangle. *MatLab* script gda09_02.

$$p(m)dm = p[m(m')] \left| \frac{dm}{dm'} \right| dm' = p(m')dm' \quad \text{so} \quad p(m') = \frac{1}{2}m'^{-1/2} \quad (9.1)$$

The distribution of m' is not uniform (Figure 9.1B), and any inverse method developed to solve the problem under this parameterization must account for this fact.

The second source of discrepancy is more serious. Suppose that we could use some inverse theory to calculate the distribution of the model parameters under a particular parameterization. We could then use Equation (9.1) to find their distribution under any arbitrary parameterization. Insofar as the distribution of the model parameters is the *answer* to the inverse problem, we would have the correct answer in the new parameterization. Probability distributions are invariant under changes of parameterization. However, a distribution is not always the answer for which we are looking. More typically, we need an estimate (a single number) based on a probability distribution (for instance, its maximum likelihood point or mean value).

Estimates are *not* invariant under changes in the parameterization. For example, suppose $p(m)$ has a uniform distribution as above. Then, if $m' = m^2$, $p(m') = \frac{1}{2}m'^{-1/2}$. The distribution in m has no maximum likelihood point, whereas the distribution in m' has one at $m' = m = 0$. The distributions also have different means (expectations):

$$\begin{aligned} \langle m \rangle &= \int_0^1 m p(m) dm = \int_0^1 m dm = 1/2 \\ \langle m' \rangle &= \int_0^1 m' p(m') dm' = \int_0^1 \frac{1}{2} m'^{1/2} dm' = 1/3 \end{aligned} \quad (9.2)$$

Even though m' equals the square of the model parameter m , the mean (expectation) of m' is not equal to the square of the expectation of m , that is, $(1/3) \neq (1/2)^2$.

There is some advantage, therefore, in working explicitly with probability distributions as long as possible, forming estimates only at the last step. If \mathbf{m} and \mathbf{m}' are two different parameterizations of model parameters, we want to avoid as much as possible sequences like

$$\text{p.d.f. for } \mathbf{m} \rightarrow \text{estimate of } \mathbf{m} \rightarrow \text{estimate of } \mathbf{m}' \quad (9.3)$$

in favor of sequences like

$$\text{pdf for } \mathbf{m} \rightarrow \text{pdf for } \mathbf{m}' \rightarrow \text{estimate of } \mathbf{m}' \quad (9.4)$$

Note, however, that the mathematics for this second sequence is typically much more difficult than that for the first.

There are objective criteria for the “goodness” of a particular estimate of a model parameter. Suppose that we are interested in the value of a model parameter \mathbf{m} . Suppose further that this parameter is either deterministic with a true value or (if it is a random variable) has a well-defined distribution from which the true expectation could be calculated if the distribution were known. Of course, we cannot know the true value; we can only perform experiments and then apply inverse theory to derive an estimate of the model parameter. As any one experiment contains noise, the estimate we derive will not coincide with the true value of the model parameter. But we can at least expect that if we perform the experiment enough times, the estimated values will scatter about the true value. If they do, then the method of estimating is said to be *unbiased*. Estimating model parameters by taking nonlinear combinations of estimates of other model parameters almost always leads to bias.

9.2 LINEARIZING TRANSFORMATIONS

One of the reasons for changing parameterizations is that an inverse problem can sometimes be transformed into a simpler form that can be solved by a known method. The problems that most commonly benefit from such transformations involve fitting exponential and power functions to data. Consider a set of (z, d) data pairs that are thought to obey the model $d_i = m_1 \exp(m_2 z_i)$. By making the transformation

$$m'_1 = \log(m_1) \quad \text{and} \quad m'_2 = m_2 \quad \text{and} \quad d'_i = \log(d_i) \quad (9.5)$$

we can write the model as the linear equation $d'_i = m'_1 + m'_2 z_i$, which can be solved by simple least-squares techniques. However, we must assume that the d'_i are independent random variables with a Gaussian probability density function with uniform variance in order to justify rigorously the application of least-squares techniques to this problem. The distribution of the data in their original parameterization must therefore be non-Gaussian. (It must have a *log-normal* probability density function.)

Notice that, if the exponential decays with increasing z for all $m_2 < 0$, the process of taking a logarithm amplifies the scattering of the near-zero points

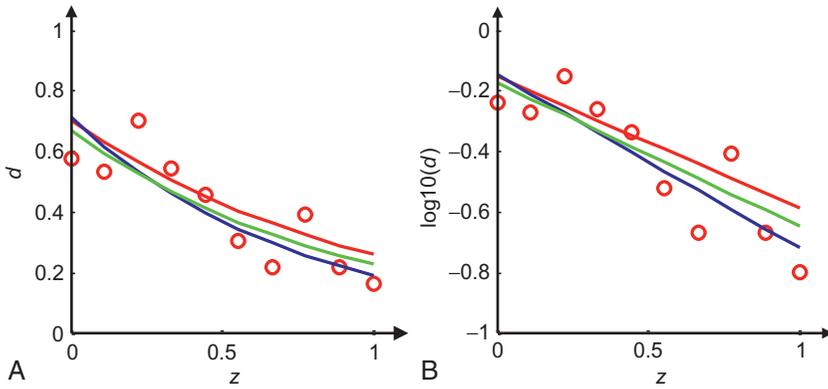


FIGURE 9.2 (A) Least-squares fit to the exponential function, $d_i = m_1 \exp(m_2 z_i)$. (Red curve) The true function, $d(z)$, for $(m_1, m_2) = (0.7, 1.0)$. (Red circles) Data that include Normally distributed random noise with zero mean and variance, $\sigma_d^2 = (0.1)^2$. (Green curve) Nonlinear least-squares solution using Newton's method. (Blue curve) Least-squares solution using the linearizing transformation, $\ln(d_i) = \ln(m_1) + m_2 z_i$. Note that the two solutions are different. (B) Log-linear version of the graph in (A). Note that the scatter of the data increases with z , and that the solution based on the linearizing transformation is strongly affected by outliers associated with it. *MatLab* script gda09_03.

that occurs at large z . The assumption that the d'_i have uniform variance, therefore, implies that the data \mathbf{d} were measured with an accuracy that increases with z (Figure 9.2). This assumption may well be inconsistent with the facts of the experiment. Linearizing transformations must be used with some caution.

9.3 ERROR AND LIKELIHOOD IN NONLINEAR INVERSE PROBLEMS

Suppose that the data \mathbf{d} in an inverse problem has a possibly non-Gaussian probability density function $p(\mathbf{d}; \langle \mathbf{d} \rangle)$, where $\langle \mathbf{d} \rangle$ is the mean and the semicolon is used to indicate that $\langle \mathbf{d} \rangle$ is just a parameter in the probability density function for \mathbf{d} . The principle of maximum likelihood—that the observed data are the most probable data—holds regardless of the form of $p(\mathbf{d}; \langle \mathbf{d} \rangle)$. As long as the theory is explicit, we can assume that the theory predicts the mean of the data, that is, $\langle \mathbf{d} \rangle = \mathbf{g}(\mathbf{m})$ and write the likelihood $L(\mathbf{m})$ as

$$L(\mathbf{m}) = \log p(\mathbf{d}^{\text{obs}}, \mathbf{m}) = c - \frac{1}{2} E(\mathbf{m}) \quad (9.6)$$

Here c is some constant, $E(\mathbf{m})$ is some function, and the factor of 1/2 has been introduced so that $E(\mathbf{m})$ equals the weighted prediction error in the Gaussian case; that is, $E(\mathbf{m}) = [\mathbf{d}^{\text{obs}} - \mathbf{g}(\mathbf{m})]^T [\text{cov } \mathbf{d}]^{-1} [\mathbf{d}^{\text{obs}} - \mathbf{g}(\mathbf{m})]$. Thus, it is always possible to define an “error” $E(\mathbf{m})$ such that minimizing $E(\mathbf{m})$ is equivalent to maximizing the likelihood $L(\mathbf{m})$. However, although $E(\mathbf{m})$ is the total

prediction error in the Gaussian case, it does not necessarily have all the properties of the total prediction error in other cases. For instance, it may not be zero when $\mathbf{d}^{\text{obs}} = \mathbf{g}(\mathbf{m})$.

There is no simple means for deciding whether a nonlinear inverse problem has a unique solution. Consider the very simple nonlinear model $d_i = m_1^2 + m_1 m_2 z_i$ with Gaussian data. This problem can be linearized by the transformation of variables $m'_1 = m_1^2, m'_2 = m_1 m_2$ and can therefore be solved by the least-squares method if $N \geq 2$. Nevertheless, even if the primed parameters are unique, the unprimed ones are not: if \mathbf{m}^{est} is a solution that minimizes the prediction error, then $-\mathbf{m}^{\text{est}}$ is also a solution with the same error. In this instance, the error $E(\mathbf{m})$ has two minima of equal depth.

We must examine the global properties of the prediction error $E(\mathbf{m})$ in order to determine whether the inverse problem is unique. If the function has but a single minimum point \mathbf{m}^{est} , then the solution is unique. If it has more than one minimum point, the solution is nonunique, and *a priori* information must be added to resolve the indeterminacy. The error surface of a linear problem is always a paraboloid (Figure 9.3), which can have only a simple range of shapes. An arbitrarily complex nonlinear inverse problem can have an arbitrarily complicated error. If $M=2$ or 3, it may be possible to investigate the shape of the surface by graphical techniques (Figure 9.4). For most realistic problems this is infeasible.

9.4 THE GRID SEARCH

One strategy for solving a nonlinear inverse problem is to exhaustively consider “every possible” solution and pick the one with the smallest error $E(\mathbf{m})$. Of course, it is impossible to examine “every possible” solution; but it is possible to examine a large set of trial solutions. When the trial solutions are drawn from

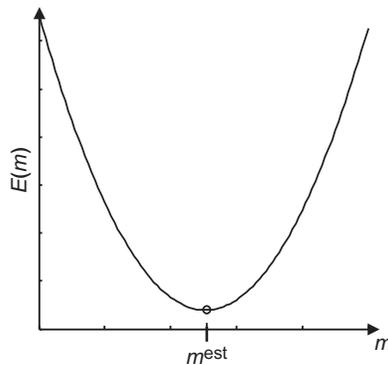


FIGURE 9.3 L_2 prediction error $E(m)$, as a function of model parameter m , for a typical linear inverse problem. The solution m^{est} minimizes the error. In the linear case, $E(m)$ is always a paraboloid. *MatLab* script gda09_04.

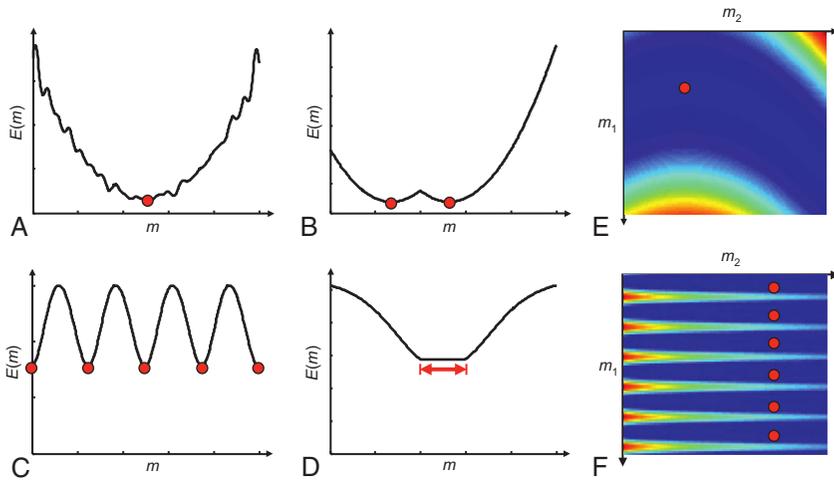


FIGURE 9.4 (A–D) Prediction error, E , as a function of a single model parameter, m . (A) A single minimum (red dot) corresponds to an inverse problem with a unique solution. (B) Two solutions. (C) Many well-separated solutions. (D) Finite range of solutions (red arrow). (E and F) Error (colors) as a function of two model parameters, m_1 and m_2 . (E) A single solution, with the minimum occurring within a nearly flat valley. (F) Many well-separated solutions. *MatLab* scripts `gda09_05` and `gda09_06`.

a regular grid in model space, this procedure is called a *grid search*. Grid searches are most practical when

1. The total number of model parameters is small, say $M < 7$. The grid is M -dimensional, so the number of trial solutions is proportional to L^M , where L is the number of trial solutions along each dimension of the grid.
2. The solution is known to lie within a specific range of values, which can be used to define the limits of the grid.
3. The forward problem $\mathbf{d} = \mathbf{g}(\mathbf{m})$ can be computed rapidly enough that the time needed to compute L^M of them is not prohibitive.
4. The error function $E(\mathbf{m})$ is smooth over the scale of the grid spacing, Δm , so that the minimum is not missed through the grid spacing being too coarse.

As an example, we use *MatLab* to solve the nonlinear problem $d(x_i) = \sin(\omega_0 m_1 x_i) + m_1 m_2$. The data are assumed to be Gaussian and uncorrelated with uniform variance so that the error is $E(m_1, m_2) = \|\mathbf{d}^{\text{obs}} - \mathbf{d}^{\text{pre}}(\mathbf{m})\|_2$. The script below has three sections: the first section defines the grid of trial \mathbf{m} values; the second evaluates $E(\mathbf{m})$ for every trial \mathbf{m} and stores the results in a matrix \mathbf{E} ; and the third searches the matrix for the smallest value of E and calculates \mathbf{m}^{est} on the basis of its row and column indices. A useful by-product of the grid search is a tabulation of E on the grid, which can be turned into an informative plot (Figure 9.5).

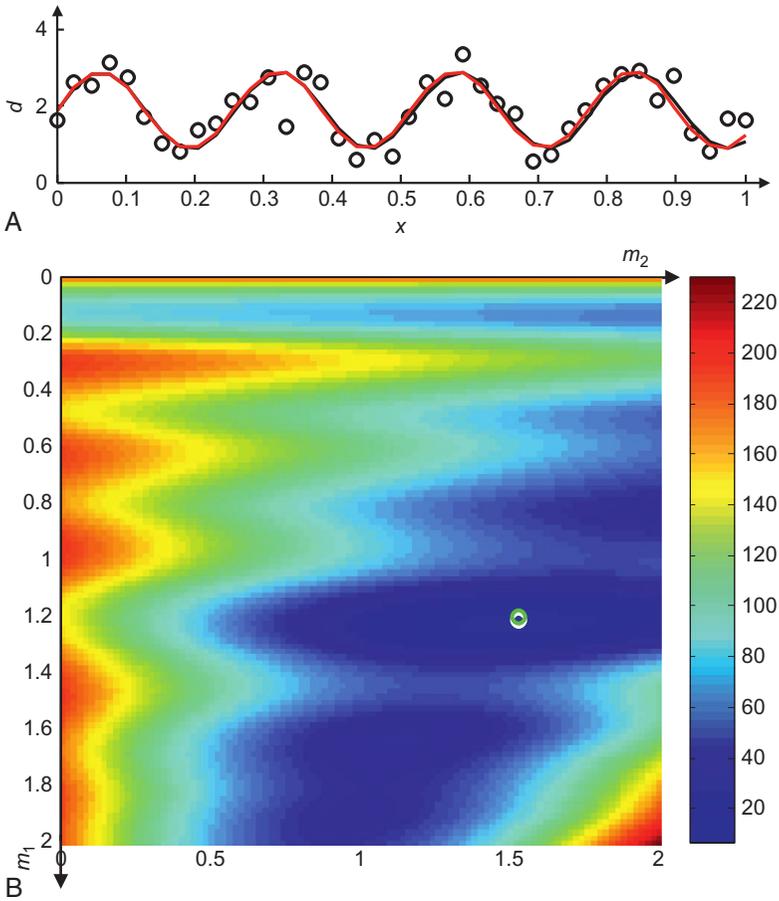


FIGURE 9.5 A grid search is used to solve the nonlinear curve-fitting problem, $d_i(x_i) = \sin(\omega_0 m_1 x_i) + m_1 m_2$. (A) The true data (black curve) are for $m_1 = 1.21$, $m_2 = 1.54$. The observed data (black circles) have additive noise with variance, $\sigma_d^2 = (0.4)^2$. The predicted data (red curve) are based on results of the grid search. (B) Error surface (colors), showing true solution (green circle) and estimated solution (white circle). *MatLab* script gda09_07.

```
% 2D grid of m's
L = 101;
Dm = 0.02;
m1min=0;
m2min=0;
m1a = m1min+Dm*[0:L-1]';
m2a = m2min+Dm*[0:L-1]';
m1max = m1a(L);
m2max = m2a(L);
```

```

% grid search, compute error, E
E = zeros(L,L);
for j = [1:L]
for k = [1:L]
    dpre = sin(w0*m1a(j)*x) + m1a(j)*m2a(k);
    E(j,k) = (dobs-dpre)'*(dobs-dpre);
end
end

% find the minimum value of E
[Erowmins, rowindices] = min(E);
[Emin, colindex] = min(Erowmins);
rowindex = rowindices(colindex);
m1est = m1min+Dm*(rowindex-1);
m2est = m2min+Dm*(colindex-1);

```

(*MatLab* script gda09_07)

9.5 THE MONTE CARLO SEARCH

The Monte Carlo search is a modification of the grid search in which the trial solutions are randomly generated, in contrast to being drawn from a regular grid. In its pure form, where each trial solution is generated independently of previous ones, it has only minor advantages over the grid search. In a later section, we will show that it can be improved by the introduction of correlation between successive trial solutions.

The accompanying *MatLab* script, which implements the algorithm, has two sections: the first section generates an initial trial solution and its corresponding error; the second is a loop that randomly generates a trial solution, calculates its corresponding error, and accepts it if the error is less than the previously accepted solution.

```

% initial guess and corresponding error
mg=[1,1]';
dg = sin(w0*mg(1)*x) + mg(1)*mg(2);
Eg = (dobs-dg)'*(dobs-dg);

% randomly generate pairs of model parameters and check
% if they further minimize the error
ma = zeros(2,1);
for k = [1:Niter]

    % randomly generate a solution
    ma(1) = random('unif',m1min,m1max);
    ma(2) = random('unif',m2min,m2max);

```

```

% compute its error
da = sin(w0*ma(1)*x) + ma(1)*ma(2);
Ea = (dobs-da) *(dobs-da);

% adopt it if it is better
if( Ea < Eg )
    mg=ma;
    Eg=Ea;
end
end
end

```

(MatLab script gda09_08)

An example is shown in Figure 9.6.

9.6 NEWTON'S METHOD

A completely different approach to solving the nonlinear inverse problem is to use information about the shape of the error $E(\mathbf{m})$ in the vicinity of a trial solution $\mathbf{m}^{(p)}$ to devise a better solution $\mathbf{m}^{(p+1)}$. One source of shape information is

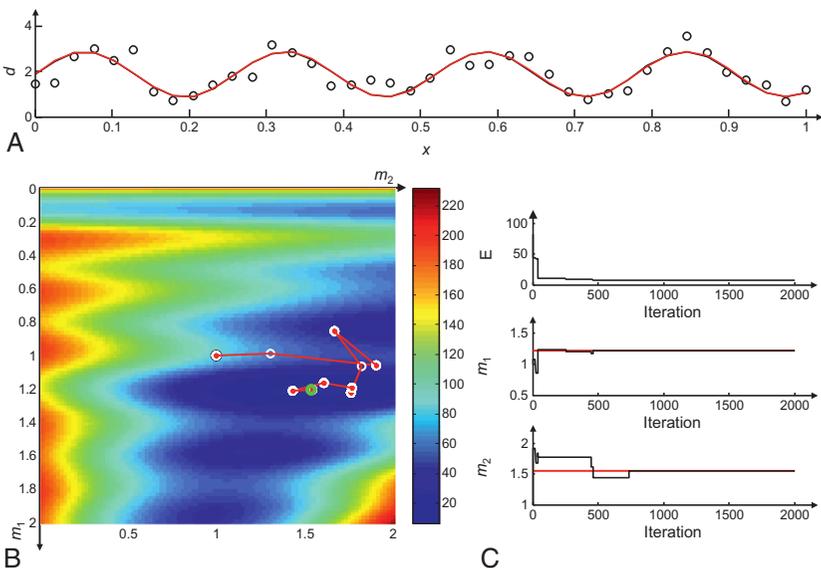


FIGURE 9.6 A Monte Carlo search is used to solve the same nonlinear curve-fitting problem as in Figure 9.5. (A) The observed data (black circles) are computed from the true data (black curve) by adding random noise. The predicted data (red curve) are based on the results of the method. (B) Error surface (colors), showing true solution (green circle), and a series of improved solutions (white circles connected by red lines) determined by the method. (C) Plot of error E and model parameters m_1 and m_2 as a function of iteration number. *MatLab* script gda09_08.

the derivatives of $E(\mathbf{m})$ at a trial solution $\mathbf{m}^{(p)}$, as Taylor's theorem indicates that the entire function can be built up from them. Expanding $E(\mathbf{m})$ in a Taylor series about the trial solution and keeping the first three terms, we obtain the parabolic approximation

$$E(\mathbf{m}) \approx E(\mathbf{m}^{(p)}) + \sum_{i=1}^M b_i (m_i - m_i^{(p)}) + \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M B_{ij} (m_i - m_i^{(p)}) (m_j - m_j^{(p)})$$

with $b_i = \left. \frac{\partial E}{\partial m_i} \right|_{\mathbf{m}^{(p)}}$ and $B_{ij} = \left. \frac{\partial^2 E}{\partial m_i \partial m_j} \right|_{\mathbf{m}^{(p)}}$

(9.7)

Here \mathbf{b} is a vector of first derivatives and \mathbf{B} is a matrix of second derivatives of $E(\mathbf{m})$, evaluated at the trial solution $\mathbf{m}^{(p)}$. These derivatives can be calculated either by analytically differentiating $E(\mathbf{m})$, if its functional form is known, or by approximating it with finite differences

$$\left. \frac{\partial E}{\partial m_i} \right|_{\mathbf{m}^{(p)}} \approx \frac{1}{\Delta m} \{E(\mathbf{m} + \Delta \mathbf{m}^{(i)}) - E(\mathbf{m})\}$$

$$\left. \frac{\partial^2 E}{\partial m_i \partial m_j} \right|_{\mathbf{m}^{(p)}} \approx \begin{cases} \frac{1}{(\Delta m)^2} \{E(\mathbf{m} + \Delta \mathbf{m}^{(i)}) - 2E(\mathbf{m}) + E(\mathbf{m} - \Delta \mathbf{m}^{(i)})\} & (i=j) \\ \frac{1}{4(\Delta m)^2} \{E(\mathbf{m} + \Delta \mathbf{m}^{(i)} + \Delta \mathbf{m}^{(j)}) - E(\mathbf{m} + \Delta \mathbf{m}^{(i)} - \Delta \mathbf{m}^{(j)}) \\ - E(\mathbf{m} - \Delta \mathbf{m}^{(i)} + \Delta \mathbf{m}^{(j)}) + E(\mathbf{m} - \Delta \mathbf{m}^{(i)} - \Delta \mathbf{m}^{(j)})\} & (i \neq j) \end{cases}$$

(9.8)

Here $\Delta \mathbf{m}^{(i)}$ is a small increment Δm in the i th direction; that is, $\Delta \mathbf{m}^{(i)} = \Delta m [0, \dots, 0, 1, 0, \dots, 0]^T$, where the i th element is unity and the rest are zero. Note that these approximations are computationally expensive, in the sense that E must be evaluated many times for each instance of \mathbf{b} and \mathbf{G} .

We can now find the minimum by differentiating this approximate form of $E(\mathbf{m})$ with respect to m_q and setting the result to zero:

$$\frac{\partial E(\mathbf{m})}{\partial m_q} = 0 = b_q + \sum_{j=1}^M B_{qj} (m_j - m_j^{(p)}) \quad \text{or} \quad \mathbf{m} - \mathbf{m}^{(p)} = -\mathbf{B}^{-1} \mathbf{b} \quad (9.9)$$

In the case of uncorrelated Gaussian data with uniform variance and the *linear* theory $\mathbf{d} = \mathbf{G}\mathbf{m}$, the error is $E(\mathbf{m}) = [\mathbf{d} - \mathbf{G}\mathbf{m}]^T [\mathbf{d} - \mathbf{G}\mathbf{m}]$, from whence we find that $\mathbf{b} = -2\mathbf{G}^T(\mathbf{d} - \mathbf{G}\mathbf{m}^{(p)})$, $\mathbf{B} = 2\mathbf{G}^T\mathbf{G}$ and $\mathbf{m} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{d}$, which is the familiar least-squares solution. In this case, the result is independent of the trial solution and is exact.

In the case of uncorrelated Gaussian data with uniform variance and the *non-linear* theory $\mathbf{d} - \mathbf{g}(\mathbf{m}) = 0$, the error is $E(\mathbf{m}) = [\mathbf{d} - \mathbf{g}(\mathbf{m})]^T [\mathbf{d} - \mathbf{g}(\mathbf{m})]$, from which we conclude

$$\mathbf{b} = -2\mathbf{G}^{(p)T}[\mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)})] \quad \text{and} \quad \mathbf{B} \approx 2[\mathbf{G}^{(p)T}\mathbf{G}^{(p)}] \quad \text{with} \quad G_{ij}^{(p)} = \left. \frac{\partial g_i}{\partial m_j} \right|_{\mathbf{m}^{(p)}}$$

$$\text{so} \quad \mathbf{m} - \mathbf{m}^{(p)} \approx [\mathbf{G}^{(p)T}\mathbf{G}^{(p)}]^{-1}\mathbf{G}^{(p)T}[\mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)})] \quad (9.10)$$

Note that we have omitted the term involving the gradient of \mathbf{G} in the formula for \mathbf{B} . In a linear theory, \mathbf{G} is constant and its gradient is zero. We assume that the theory is sufficiently close to linear that here too it is insignificant.

The form of Equation (9.10) is very similar to simple least squares. The matrix $\mathbf{G}^{(p)}$, which is the gradient of the model $\mathbf{g}(\mathbf{m})$ at the trial solution, acts as a data kernel. It can be calculated analytically, if its functional form is known, or approximated by finite differences (see Equation (9.8)). The generalized inverse $\mathbf{G}^{-g} = [\mathbf{G}^{(p)T}\mathbf{G}^{(p)}]^{-1}\mathbf{G}^{(p)T}$ relates the deviation of the data $\Delta\mathbf{d} = [\mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)})]$ from what is predicted by the trial solution to the deviation of the solution $\Delta\mathbf{m} = \mathbf{m} - \mathbf{m}^{(p)}$ from the trial solution, that is, $\Delta\mathbf{m} = \mathbf{G}^{-g}\Delta\mathbf{d}$. However, because it is based on a truncated Taylor Series, the solution is only approximate; it yields a solution that is improved over the trial solution, but not the exact solution. However, it can be iterated to yield a succession of improvements

$$\mathbf{m}^{(p+1)} = \mathbf{m}^{(p)} + [\mathbf{G}^{(p)T}\mathbf{G}^{(p)}]^{-1}\mathbf{G}^{(p)T}[\mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)})] \quad (9.11)$$

until the error declines to an acceptably low level (Figure 9.7). Unfortunately, while convergence of $\mathbf{m}^{(p)}$ to the value that globally minimizes $E(\mathbf{m})$ is often

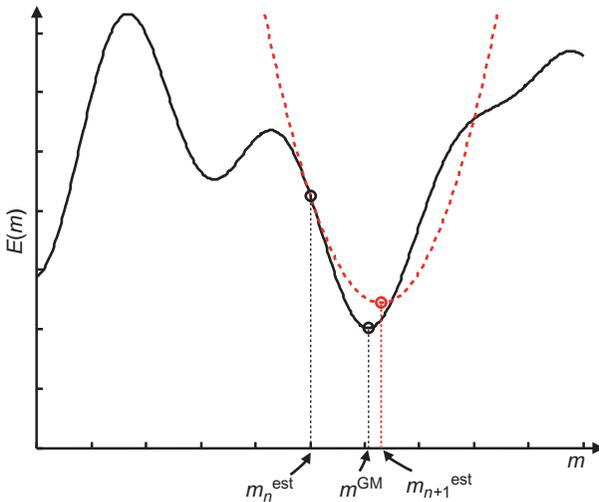


FIGURE 9.7 The iterative method locates the global minimum m^{GM} of the error $E(m)$ (black curve) by determining the paraboloid (red curve) that is tangent to E at the trial solution m_n^{est} . The improved solution m_{n+1}^{est} is at the minimum (red circle) of this paraboloid and, under favorable conditions, can be closer to the solution corresponding to the global minimum than is the trial solution. *MatLab* script gda09_09.

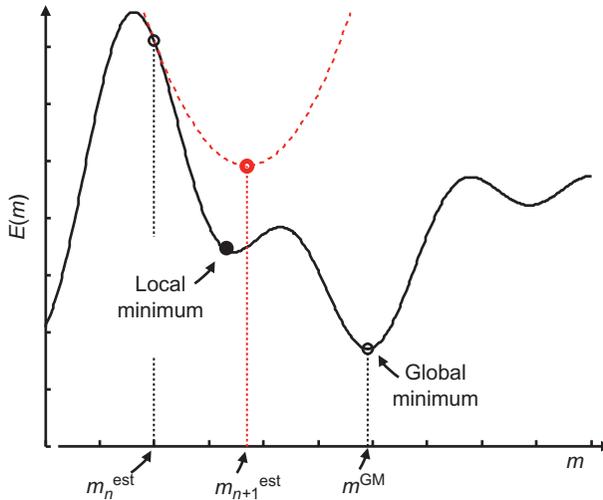


FIGURE 9.8 If the trial solution, m_n^{est} , is too far from the global minimum m^{GM} , the method may converge to a local minimum. *MatLab* script gda09_10.

very rapid, it is not guaranteed. One common problem is the solution converging to a local minimum instead of the global minimum (Figure 9.8). A critical part of the algorithm is picking the initial trial solution $\mathbf{m}^{(1)}$. The method may not find the global minimum if the trial solution is too far from it.

As an example, we consider the problem from Section 9.4, in which $g_i(\mathbf{m}) = \sin(\omega_0 m_1 x_i) + m_1 m_2$. The matrix of partial derivatives is

$$\mathbf{G}^{(p)} = \begin{bmatrix} \omega_0 x_1 \cos(\omega_0 x_1 m_1^{(p)}) + m_2^{(p)} & m_1^{(p)} \\ \omega_0 x_2 \cos(\omega_0 x_2 m_1^{(p)}) + m_2^{(p)} & m_1^{(p)} \\ \vdots & \vdots \\ \omega_0 x_N \cos(\omega_0 x_N m_1^{(p)}) + m_2^{(p)} & m_1^{(p)} \end{bmatrix} \quad (9.12)$$

In *MatLab*, the main part of the script is a loop that iteratively updates the trial solution. First, the deviation $\Delta \mathbf{d}$ (`dd` in the script) and the data kernel \mathbf{G} are calculated, using the trial solution $\mathbf{m}^{(p)}$. Then the deviation $\Delta \mathbf{m}$ is calculated using least squares. Finally, the trial solution is updated as $\mathbf{m}^{(p+1)} = \mathbf{m}^{(p)} + \Delta \mathbf{m}$. In *MatLab*

```
% initial guess and corresponding error
mg=[1,1]';
dg = sin(w0*mg(1)*x) + mg(1)*mg(2);
Eg = (dobs-dg)'*(dobs-dg);

% iterate to improve initial guess
Niter = 20;
```

```

G = zeros(N,M);
for k = [1:Niter]

    dg = sin(w0*mg(1)*x) + mg(1)*mg(2);
    dd = dobs-dg;
    Eg=dd'*dd;

    G = zeros(N,2);
    G(:,1) = w0 * x .* cos( w0 * mg(1) * x ) + mg(2);
    G(:,2) = mg(2)*ones(N,1);

    % least squares solution
    dm = (G'*G)\(G'*dd);

    % update
    mg = mg+dm;
end

```

(MatLab script gda09_11)

The results of this script are shown in [Figure 9.9](#). This exemplary script iterates a fixed number of times, as specified by the variable `Niter`. A more elegant approach would be to perform a test that terminates the iterations when the error declines to an acceptable level or when the solution no longer changes significantly from iteration to iteration.

9.7 THE IMPLICIT NONLINEAR INVERSE PROBLEM WITH GAUSSIAN DATA

We now generalize the iterative method of the previous section to the general case of the implicit theory $\mathbf{f}(\mathbf{d}, \mathbf{m})=0$, where \mathbf{f} is of length $L \leq M+N$. We assume that the data \mathbf{d} and *a priori* model parameters $\langle \mathbf{m} \rangle$ have Gaussian distributions with covariance $[\text{cov } \mathbf{d}]$ and $[\text{cov } \mathbf{m}]_A$, respectively. If we let $\mathbf{x} = [\mathbf{d}^T, \mathbf{m}^T]^T$, we can think of the *a priori* distribution of the data and model as a cloud in the space $S(\mathbf{x})$ centered about the observed data and mean *a priori* model parameters, with a shape determined by the covariance matrix $[\text{cov } \mathbf{x}]$ ([Figure 9.10](#)). The matrix $[\text{cov } \mathbf{x}]$ contains $[\text{cov } \mathbf{d}]$ and $[\text{cov } \mathbf{m}]$ on diagonal blocks, as in [Equation \(5.27\)](#). In principle, the off-diagonal blocks could be made nonzero, indicating correlation between observed data and *a priori* model parameters. However, specifying *a priori* constraints is typically an *ad hoc* procedure that one can seldom find motivation for introducing such a correlation. The *a priori* distribution is therefore

$$p_A(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}[\mathbf{x} - \langle \mathbf{x} \rangle]^T [\text{cov } \mathbf{x}]^{-1} [\mathbf{x} - \langle \mathbf{x} \rangle]\right\} \quad (9.13)$$

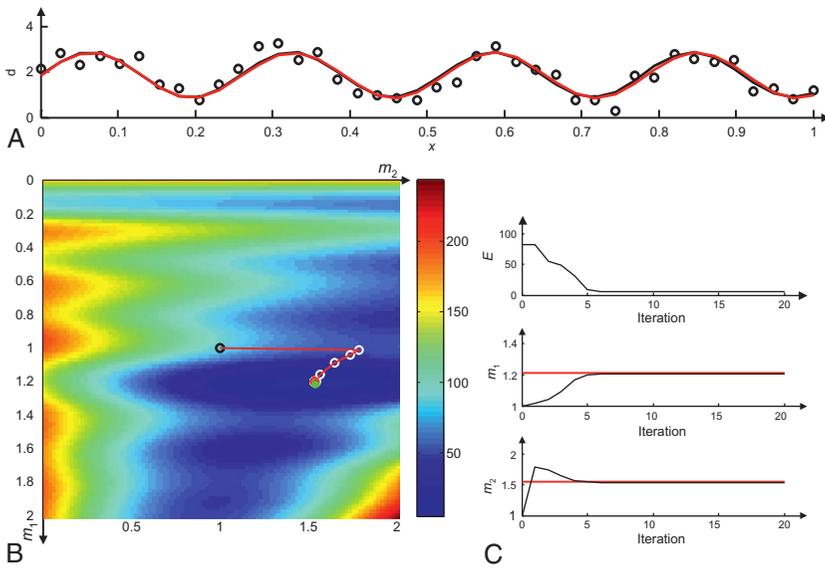


FIGURE 9.9 Newton's method (linearized least squares) is used to solve the same nonlinear curve-fitting problem as in Figure 9.5. (A) The observed data (black circles) are computed from the true data (black curve) by adding random noise. The predicted data (red curve) are based on the results of the method. (B) Error surface (colors), showing true solution (green dot), and a series of improved solutions (white circles connected by red lines) determined by the method. (C) Plot of error E and model parameters, m_1 and m_2 as a function of iteration number. *MatLab* script gda09_11.

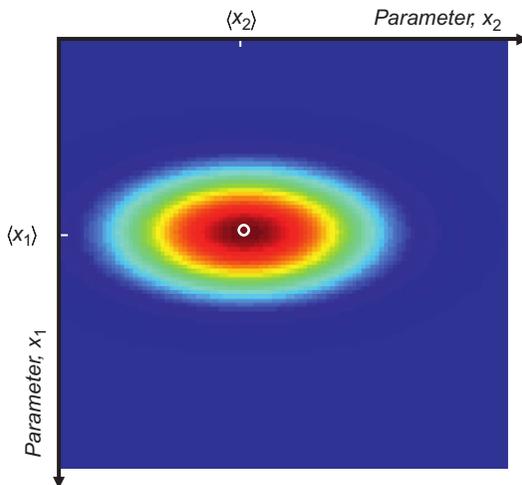


FIGURE 9.10 The data and model parameters are grouped together in a vector, \mathbf{x} . The prior information for \mathbf{x} is then represented as a probability density function (colors) in the $(M+N)$ -dimensional space, $S(\mathbf{x})$. *MatLab* script gda09_12.

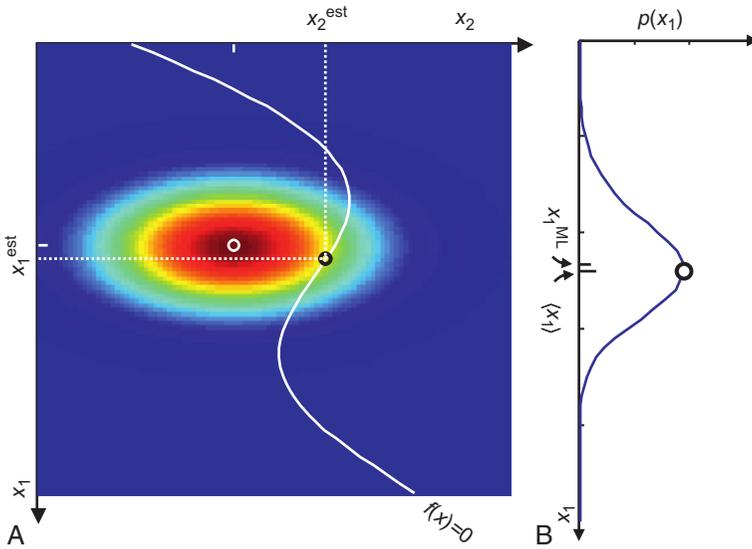


FIGURE 9.11 (A) The estimated solution \mathbf{x}^{est} (black circle) is at the point on the surface $\mathbf{f}(\mathbf{x})=0$ (white curve) where the *a priori* probability density function (colors) attains its largest value. (B) The probability density function $p(x_1)$ evaluated along the surface, as a function of position x_1 . As the function is nonnormal, its mean $\langle x_1 \rangle$ may be distinct from its mode x_1^{ML} (although in this case they are similar). *MatLab* script gda09_13.

where $\langle \mathbf{x} \rangle = [(\mathbf{d}^{\text{obs}})^T, \langle \mathbf{m} \rangle^T]^T$ is a vector containing the observed data and *a priori* model parameters.

The theory $\mathbf{f}(\mathbf{x})=0$ defines a surface in $S(\mathbf{x})$ on which the predicted data and estimated model parameters $\mathbf{x}^{\text{est}} = [\mathbf{d}^{\text{pre } T}, \mathbf{m}^{\text{est } T}]^T$ must lie. The probability distribution for \mathbf{x}^{est} is, therefore, $p_A(\mathbf{x})$, evaluated on this surface (Figure 9.11). If the surface is plane, this is just the linear case described in Chapter 5 and the final distribution is Gaussian. On the other hand, if the surface is very “bumpy,” the distribution on the surface will be very non-Gaussian and may even possess several maxima (Figure 9.12).

One approach to estimating the solution is to find the maximum likelihood point of $p_A(\mathbf{x})$ on the surface $\mathbf{f}(\mathbf{x})=0$ (Figure 9.11). This point can be found without explicitly determining the distribution on the surface. One just maximizes $p_A(\mathbf{x})$ with the constraint that $\mathbf{f}(\mathbf{x})=0$. One should keep in mind, however, that the maximum likelihood point of a non-Gaussian distribution may not be the most sensible estimate that can be made from that distribution. Gaussian distributions are symmetric, so their maximum likelihood point always coincides with their mean value. In contrast, the maximum likelihood point can be arbitrarily far from the mean of a non-Gaussian distribution (Figure 9.12). Computing the mean, however, requires one to compute explicitly the distribution on the surface and then take its expectation (a much more difficult procedure).

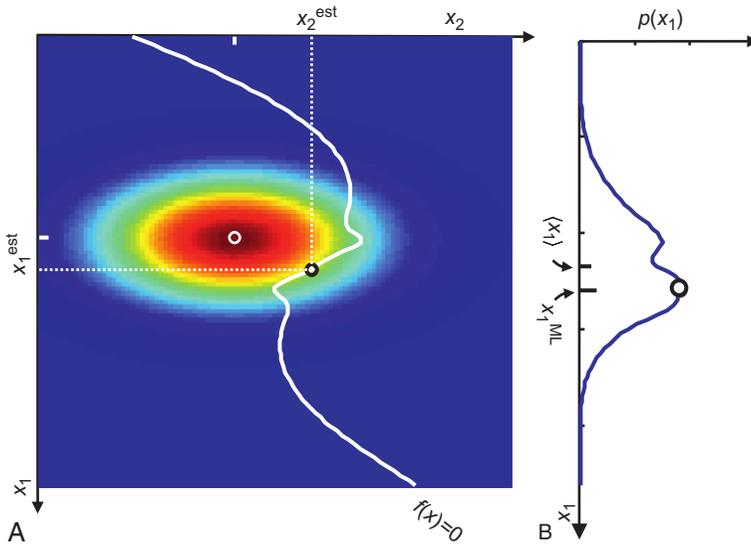


FIGURE 9.12 (A) A highly nonlinear inverse problem corresponds to a complicated surface $\mathbf{f}(\mathbf{x})=0$ (white curve). (B) The probability density function $p(x_1)$ evaluated along the surface, as a function of position x_1 . It may have several peaks, and as it is nonnormal, its mean $\langle x_1 \rangle$ may be distinct from its mode x_1^{ML} . *MatLab* script gda09_14.

These caveats aside, we proceed with the calculation of the maximum likelihood point by minimizing the argument of the exponential in $p_{\Lambda}(\mathbf{x})$ with the constraint that $\mathbf{f}(\mathbf{x})=0$ (adapted from [Tarantola and Valette, 1982](#)):

$$\text{minimize } \Phi = [\mathbf{x} - \langle \mathbf{x} \rangle]^T [\text{cov } \mathbf{x}]^{-1} [\mathbf{x} - \langle \mathbf{x} \rangle] \quad \text{subject to } \mathbf{f}(\mathbf{x}) = 0 \quad (9.14)$$

The Lagrange multiplier equations are

$$\frac{\partial \Phi}{\partial x_i} - \sum_{j=1}^L 2\lambda_j \frac{\partial f_j}{\partial x_i} = 0 \quad \text{or} \quad [\mathbf{x} - \langle \mathbf{x} \rangle]^T [\text{cov } \mathbf{x}]^{-1} = \mathbf{F}^T \boldsymbol{\lambda} \quad (9.15)$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers and \mathbf{F} is the matrix of derivatives $F_{ij} = \frac{\partial f_i}{\partial x_j}$. The Lagrange multipliers can be determined by premultiplying the transformed equation by \mathbf{F} as

$$\mathbf{F}[\mathbf{x} - \langle \mathbf{x} \rangle] = \mathbf{F}[\text{cov } \mathbf{x}] \mathbf{F}^T \boldsymbol{\lambda} \quad (9.16)$$

and then premultiplying $\{\mathbf{F}[\text{cov } \mathbf{x}] \mathbf{F}^T\}^{-1}$ as

$$\boldsymbol{\lambda} = \{\mathbf{F}[\text{cov } \mathbf{x}] \mathbf{F}^T\}^{-1} \mathbf{F}[\mathbf{x} - \langle \mathbf{x} \rangle] \quad (9.17)$$

Substitution into the transpose of the original equation yields

$$[\mathbf{x} - \langle \mathbf{x} \rangle] = [\text{cov } \mathbf{x}] \mathbf{F}^T \{\mathbf{F}[\text{cov } \mathbf{x}] \mathbf{F}^T\}^{-1} \mathbf{F}[\mathbf{x} - \langle \mathbf{x} \rangle] \quad (9.18)$$

which must be solved simultaneously with the constraint equation $\mathbf{f}(\mathbf{x})=0$. These two equations are equivalent to the single equation

$$[\mathbf{x} - \langle \mathbf{x} \rangle] = [\text{cov } \mathbf{x}] \mathbf{F}^T \{ \mathbf{F} [\text{cov } \mathbf{x}] \mathbf{F}^T \}^{-1} \{ \mathbf{F} [\mathbf{x} - \langle \mathbf{x} \rangle] - \mathbf{f}(\mathbf{x}) \} \quad (9.19)$$

as the original two equations can be recovered by premultiplying this equation by \mathbf{F} . The form of this equation is very similar to the linear solution of Equation (5.37); in fact, it reduces to it in the case of the exact linear theory $\mathbf{f}(\mathbf{x})=\mathbf{F}\mathbf{x}$. As the unknown \mathbf{x} appears on both sides of the equation and \mathbf{f} and \mathbf{F} are functions of \mathbf{x} , this equation may be difficult to solve explicitly. We now examine an iterative method of solving it. This method consists of starting with some initial trial solution, say, $\mathbf{x}^{(p)}$, where $p = 1$, and then generating successive approximations as

$$\mathbf{x}^{(p+1)} = \langle \mathbf{x} \rangle + [\text{cov } \mathbf{x}] \mathbf{F}^{(p)T} \left\{ \mathbf{F}^{(p)} [\text{cov } \mathbf{x}] \mathbf{F}^{(p)T} \right\}^{-1} \left\{ \mathbf{F}^{(p)} [\mathbf{x}^{(p)} - \langle \mathbf{x} \rangle] - \mathbf{f}(\mathbf{x}^{(p)}) \right\} \quad (9.20)$$

The superscript on $\mathbf{F}^{(p)}$ implies that it is evaluated at $\mathbf{x}^{(p)}$. If the initial guess is close enough to the maximum likelihood point, the successive approximations will converge to the true solution \mathbf{x}^{est} ; else it may converge to a local minimum.

If the theory is explicit (i.e., if $\mathbf{f}(\mathbf{x})=\mathbf{d}-\mathbf{g}(\mathbf{m})=0$) and if the data and *a priori* model parameters are uncorrelated, the iterative formula can be rewritten as

$$\mathbf{m}^{(p+1)} = \langle \mathbf{m} \rangle + \mathbf{G}_{(p)}^{-g} \{ \mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)}) + \mathbf{G}^{(p)} [\mathbf{m}^{(p)} - \langle \mathbf{m} \rangle] \} \quad (9.21a)$$

$$\begin{aligned} \mathbf{G}_{(p)}^{-g} &= [\text{cov } \mathbf{m}]_A \mathbf{G}^{(p)T} \left\{ \mathbf{G}^{(p)} [\text{cov } \mathbf{m}]_A \mathbf{G}^{(p)T} + [\text{cov } \mathbf{d}] \right\}^{-1} \\ &= \left\{ \mathbf{G}^{(p)T} [\text{cov } \mathbf{d}]^{-1} \mathbf{G}^{(p)} + [\text{cov } \mathbf{m}]_A^{-1} \right\}^{-1} \mathbf{G}^{(p)T} [\text{cov } \mathbf{d}]^{-1} \end{aligned} \quad (9.21b)$$

or solved using simple least squares

$$\begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \mathbf{G}^{(p)} \\ [\text{cov } \mathbf{m}]_A^{-1/2} \mathbf{I} \end{bmatrix} \mathbf{m}^{(p+1)} = \begin{bmatrix} [\text{cov } \mathbf{d}]^{-1/2} \left\{ \mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)}) + \mathbf{G}^{(p)} \mathbf{m}^{(p)} \right\} \\ [\text{cov } \mathbf{m}]_A^{-1/2} \langle \mathbf{m} \rangle \end{bmatrix} \quad (9.21c)$$

Here $[\mathbf{G}^{(p)}]_{ij} = \partial g_i / \partial m_j$ is evaluated at $\mathbf{m}^{(p)}$ and the generalized inverse notation has been used for convenience. The two versions of the generalized inverse in Equation (9.21b)—one with the form of the minimum length solution and the other with the form of the least-squares solution—are equivalent, as was shown in Equation (5.39). Equation (9.21a–9.21c) is the nonlinear, iterative analog to the linear, noniterative formulas stated for the linear inverse problem in Equation (5.37), except that in this case the theory

has been assumed to be exact. One obtains formulas for an inexact theory with covariance $[\text{cov } \mathbf{g}]$ with the substitution $[\text{cov } \mathbf{d}] \rightarrow [\text{cov } \mathbf{d}] + [\text{cov } \mathbf{g}]$ and for *a priori* information of the form $\mathbf{H}\mathbf{m} = \mathbf{h}$ with the substitutions $\mathbf{I} \rightarrow \mathbf{H}$, $\langle \mathbf{m} \rangle \rightarrow \mathbf{h}$, and $[\text{cov } \mathbf{m}]_{\mathbf{A}} \rightarrow [\text{cov } \mathbf{h}]_{\mathbf{A}}$.

Provided the problem is overdetermined, Equation (9.21a–9.21c) reduces to Newton's method in the limit where $\langle \mathbf{m} \rangle \rightarrow 0$, $[\text{cov } \mathbf{m}]_{\mathbf{A}}^{-1} \rightarrow \mathbf{0}$, and $[\text{cov } \mathbf{d}]^{-1} \rightarrow \mathbf{I}$

$$\Delta \mathbf{m} = \mathbf{G}^{-g} \Delta \mathbf{d} + (\mathbf{I} - \mathbf{R}^{(p)}) \mathbf{m}^{(p)} \rightarrow \mathbf{G}^{-g} \Delta \mathbf{d} \quad (9.22)$$

as the resolution matrix $\mathbf{R}^{(p)} = \mathbf{G}_{(p)}^{-g} \mathbf{G}^{(p)} \rightarrow \mathbf{I}$ in the overdetermined case. However, Equation (9.21a–9.21c) indicates that Newton's method cannot be *patched* for underdetermined problems merely by using a damped least-squares version of $\mathbf{G}_{(p)}^{-g}$. The problem is that damped least squares drives $\Delta \mathbf{m}$ toward zero, rather than (as is more sensible) driving $\mathbf{m}^{(p+1)}$ toward zero. Instead, *MatLab* scripts should solve Equation (9.21c), using the `bicg()` solver together with the `weightedleastquaresfcn()` function.

In a linear problem, the error $E(\mathbf{m})$ is a paraboloid and the estimated model parameters \mathbf{m}^{est} are at its minimum. While a nonlinear problem has an error with a more complicated shape, it may still be approximately paraboloid in the vicinity of its minimum. This is the region of the space of model parameters where the inverse problem behaves linearly and the probability density function $p(\mathbf{m})$ is approximately Gaussian in shape. If the patch is big enough to encompass a large percentage of the total probability, then one might use the linear formula

$$[\text{cov } \mathbf{m}^{\text{est}}] = \mathbf{G}_{(p)}^{-g} [\text{cov } \mathbf{d}] \mathbf{G}_{(p)}^{-gT} + [\mathbf{I} - \mathbf{R}^{(p)}] [\text{cov } \mathbf{m}]_{\mathbf{A}} [\mathbf{I} - \mathbf{R}^{(p)}]^T \quad (9.23)$$

where p is the final iteration, to calculate approximate variances of the model parameters. Whether 95% confidence intervals inferred from these variances are correct will depend upon the size of the patch because only the central part of the probability density function, and not its tails, is approximately Gaussian. For this reason, estimates of confidence intervals based on the Bootstrap method (Section 9.11) are usually preferred.

The same caveat applies to interpretations of the resolution matrices $\mathbf{N}^{(p)}$ and $\mathbf{R}^{(p)}$. As the problem is nonlinear, they do not describe the true resolution of the problem. On the other hand, they give the resolution of a linear problem that is in some sense close to the nonlinear one.

9.8 GRADIENT METHOD

Occasionally, one encounters an inverse problem in which the error $E(\mathbf{m})$ and its gradient $[\nabla E]_i = dE/dm_i$ are especially easy to calculate. It is possible to solve the inverse problem using this information alone, as the unit vector

$$\mathbf{v} = - \frac{\nabla E}{|\nabla E|} \quad (9.24)$$

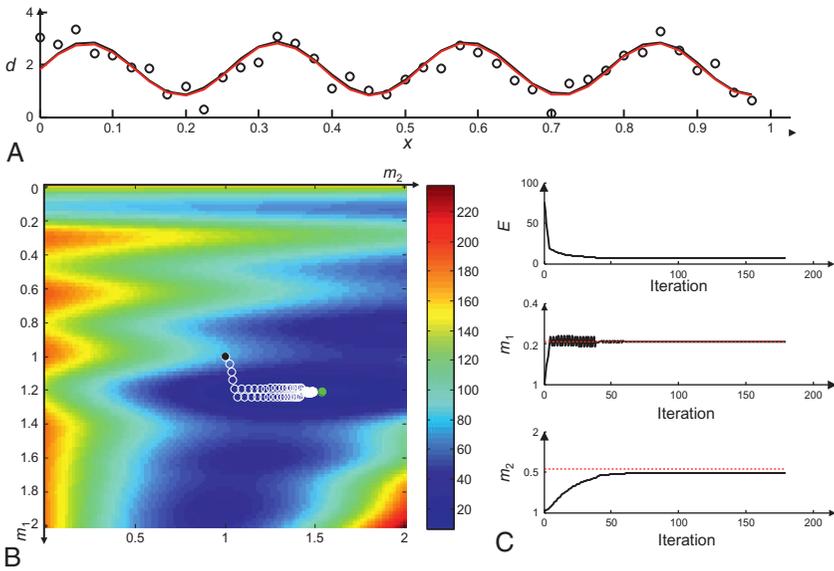


FIGURE 9.13 Gradient method is used to solve the same nonlinear curve-fitting problem as in Figure 9.5. (A) The observed data (black circles) are computed from the true data (black curve) by adding random noise. The predicted data (red curve) are based on the results of the method. (B) Error surface (colors), showing true solution (green dot), and a series of improved solutions (white circles) determined by the method. (C) Plot of error E and model parameters m_1 and m_2 as a function of iteration number. *MatLab* script gda09_15.

points in the direction in which the error E is minimized. Thus a trial solution $\mathbf{m}^{(j)}$ can be improved to $\mathbf{m}^{(j+1)} = \mathbf{m}^{(j)} + \alpha \mathbf{v}$, where α is a positive number. The only problem is that one does not immediately know how large α should be. Too large and the minimum may be skipped over; too small and the convergence will be very slow. *Armijo's rule* provides an acceptance criterion for α :

$$E(\mathbf{m}^{(k+1)}) \leq E(\mathbf{m}^{(k)}) + c\alpha \mathbf{v}^T \nabla E|_{\mathbf{m}^{(k)}} \quad (9.25)$$

Here c is an empirical constant in the range $(0, 1)$ that is usually chosen to be about 10^{-4} . One strategy is to start the iteration with a “largish” value of α and to use it as long as it passes the test, but to decrease it whenever it fails, say using the rule $\alpha \rightarrow \alpha/2$. An example is shown in Figure 9.13.

9.9 SIMULATED ANNEALING

The Monte Carlo method (Section 9.5) is completely *undirected*. The whole model space is sampled randomly so that the global minimum of the error *eventually* is found, provided that enough trial solutions are examined. Unfortunately, the number of trial solutions that need be computed may be very large—perhaps millions.

In contrast, Newton's method (Section 9.6) is completely directed. Local properties of the error—its slope and curvature—are used to determine the optimal improvement to a trial solution. Convergence to the global minimum of the error, when it occurs, is rapid and just a few trial solutions need be computed—perhaps just ten. Unfortunately, the method may only find a local minimum of the error that corresponds to an incorrect estimate of the model parameters.

The *simulated annealing* method combines the best features of these two methods. Like the Monte Carlo method, it samples the whole model space and so can avoid getting stuck in local minima. And like Newton's method, it uses local information to direct the sequence of trial solutions. Its development was inspired by the physical annealing of metals (Kirkpatrick et al., 1983), where an orderly minimum-energy crystal structure develops within the metal as it is slowly cooled from a red hot state. Initially, when the metal is hot, atomic motions are completely dominated by random thermal fluctuations, but as the temperature is slowly lowered, interatomic forces become more and more important. In the end, the atoms become a crystal lattice that represents a minimum-energy configuration.

In the simulated annealing algorithm, a parameter T is the analog to temperature and the error E is the analog to energy. Large values of T cause the algorithm to behave like a Monte Carlo search; small values cause it to act in a more directed way. As in physical annealing, one starts with a large T and then slowly decreases it as more and more trial solutions are examined. Initially, a very large volume of model space is randomly sampled, but the search becomes increasingly directed as it progresses.

The simulated annealing algorithm starts with a trial solution $\mathbf{m}^{(p)}$ with corresponding error $E(\mathbf{m}^{(p)})$. A test solution \mathbf{m}^* with corresponding error $E(\mathbf{m}^*)$ is then generated that is in the neighborhood of $\mathbf{m}^{(p)}$, say by adding to $\mathbf{m}^{(p)}$ an increment $\Delta\mathbf{m}$ drawn from a Gaussian distribution. The test solution is always accepted as the new trial solution $\mathbf{m}^{(p+1)}$ when $E(\mathbf{m}^*) \leq E(\mathbf{m}^{(p)})$, but it is also sometimes accepted even when $E(\mathbf{m}^*) > E(\mathbf{m}^{(p)})$. To decide the later case, a test parameter

$$t = \frac{\exp\{-E(\mathbf{m}^*)/T\}}{\exp\{-E(\mathbf{m}^{(p)})/T\}} = \exp\left\{-\frac{[E(\mathbf{m}^*) - E(\mathbf{m}^{(p)})]}{T}\right\} \quad (9.26)$$

is computed. Then, a random number r that is uniformly distributed on the interval $[0, 1]$ is generated and the solution \mathbf{m}^* is accepted if $t > r$. When T is large, the parameter t is close to unity and \mathbf{m}^* is almost always accepted, regardless of the value of the error. This corresponds to the “thermal motion” case where the space of model parameters is explored in an undirected way. When T is small, the parameter t is close to zero and \mathbf{m}^* is almost never accepted. This corresponds to the directed search case, as then the only solutions that decrease the error are accepted. An astute reader will recognize that this is just the Metropolis-Hastings algorithm (Section 2.8) applied to the Boltzmann probability density function

$$p(\mathbf{m}) \propto \exp\left\{\frac{-E(\mathbf{m})}{T}\right\} \quad (9.27)$$

The maximum likelihood point of this probability density function corresponds to the point of minimum error, regardless of the value of the parameter T . However, T controls the width of the probability density function, with a larger T corresponding to a wider function. At first, T is high and the distribution is wide. The sequence of realizations samples a broad region of model space that includes the global minimum and, possibly, local minima as well. As T is decreased, the probability density function becomes increasingly peaked at the global minimum. In the limit of $T=0$, all the realizations are at the maximum likelihood point; that is, the sought-after point that minimizes the error.

Note that when $T=2$, we recover the probability density function $p(\mathbf{d}^{\text{obs}}; \mathbf{m})$, as defined in Section 9.3. An alternate strategy for “solving” the inverse problem is to stop the cooling at this temperature and then to produce a large set of realizations of this distribution (see Section 1.4.4). Either the entire set of solutions, itself, or a single parameter derived from it, such as the mean, can be considered the “solution” of the inverse problem.

In *MatLab*, the main part of the Metropolis-Hastings algorithm is a loop that computes the current value of T , randomly computes a $\Delta\mathbf{m}$ to produce the solution \mathbf{m}^* and a corresponding error $E(\mathbf{m}^*)$, and applies the Metropolis rules to either accept or reject \mathbf{m}^* .

```
Dm = 0.2;
Niter=400;
for k = [1:Niter]

    % temperature falls off with iteration number
    T = 0.1 * Eg0 * ((Niter-k+1)/Niter)^2;

    % randomly pick model parameters and evaluate error
    ma(1) = random('Normal', mg(1), Dm);
    ma(2) = random('Normal', mg(2), Dm);
    da = sin(w0*ma(1)*x) + ma(1)*ma(2);
    Ea = (dobs-da)'*(dobs-da);

    % accept according to Metropolis rules
    if ( Ea < Eg )
        mg=ma;
        Eg=Ea;
        plhis(k+1)=1;
    else
        p1 = exp( -(Ea-Eg)/T );
        p2 = random('unif', 0, 1);
        if ( p1 > p2 )
```

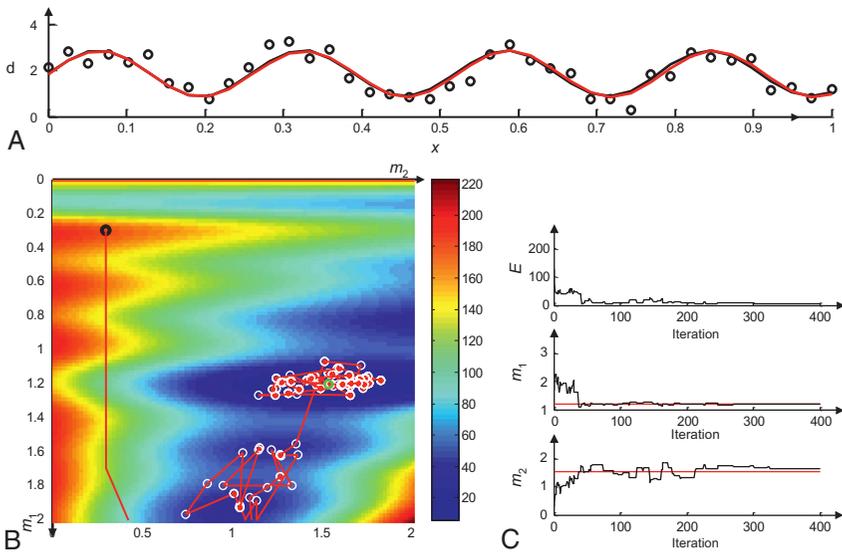


FIGURE 9.14 Simulated annealing is used to solve the same nonlinear curve-fitting problem as in Figure 9.5. (A) The observed data (black circles) are computed from the true data (black curve) by adding random noise. The predicted data (red curve) are based on the results of the method. (B) Error surface (colors), showing true solution (green circle), and a series of solutions (white circles connected by red lines) determined by the method. (C) Plot of error E and model parameters m_1 and m_2 as a function of iteration number. *MatLab* script gda09_16.

```

mg=ma ;
Eg=Ea ;
end
end
end
end

```

(*MatLab* script gda09_16)

An example is shown in Figure 9.14.

9.10 CHOOSING THE NULL DISTRIBUTION FOR INEXACT NON-GAUSSIAN NONLINEAR THEORIES

As we found in Section 5.2.4, the starting place for analyzing inexact theories is the rule for combining probability density functions to yield the total probability density function p_T . It is built up by combining three probability density functions p_A (*a priori*), p_f or p_g (theory), and p_N (null)

$$p_T(\mathbf{d}, \mathbf{m}) = \frac{p_A(\mathbf{d})p_A(\mathbf{m})p_g(\mathbf{d}, \mathbf{m})}{p_N(\mathbf{d})p_N(\mathbf{m})} \quad \text{or} \quad p_T(\mathbf{x}) = \frac{p_A(\mathbf{x})p_f(\mathbf{x})}{p_N(\mathbf{x})} \quad (9.28)$$

In the Gaussian case, we have been assuming $p_N \propto \text{constant}$. However, this definition is sometimes inadequate. For instance, if \mathbf{x} is the Cartesian coordinate of an object in three-dimensional space, then $P_N \propto \text{constant}$ means that the object could be anywhere with equal probability. This is an adequate definition of the null distribution. On the other hand, if the position of the object is specified by the spherical coordinates $\mathbf{x} = [r, \theta, \phi]^T$, then the statement $P_N \propto \text{constant}$ actually implies that the object is near the origin. The statement that the object could be anywhere is $P_N(\mathbf{x}) \propto r^2 \sin(\theta)$. The null distribution must be chosen with the physical significance of the vector \mathbf{x} in mind.

Unfortunately, it is sometimes difficult to find a guiding principle with which to choose the null distribution. Consider the case of an acoustics problem in which a model parameter is the acoustic velocity v . At first sight it may seem that a reasonable choice for the null distribution is $p_N(v) \propto \text{constant}$. Acousticians, however, often work with the acoustic slowness $s = 1/v$, and the distribution $p_N(v) \propto \text{constant}$ implies $p_N(s) \propto s^2$. This is somewhat unsatisfactory, as one could, with equal plausibility, argue that $p_N(s) \propto \text{constant}$, in which case $p_N(v) \propto v^2$. One possible solution to this dilemma is to choose a null solution whose *form* is invariant under the reparameterization. The distribution that works in this case is $p_N(v) \propto 1/v$, as this leads to $p_N(s) \propto 1/s$.

Thus, while a non-Gaussian inexact implicit theory can be handled with the same machinery as was applied to the Gaussian case of [Section 9.7](#), more care must be taken when choosing the parameterization and defining the null distribution.

9.11 BOOTSTRAP CONFIDENCE INTERVALS

The probability density function of a nonlinear problem is non-Gaussian, even when the data have Gaussian-distributed error. The simple formulas that we developed for error propagation are not accurate in such cases, except perhaps when the nonlinearity is very weak. We describe here an alternative method of computing confidence intervals for the model parameters that performs the error propagation in an alternative way.

If many *repeat* data sets were available, the problem of estimating confidence intervals could be approached empirically. If we had repeated the experiment 1000 times, each time with the exact same experimental conditions, we would have a group of 1000 data sets, all similar to one another, but each containing a different pattern of observational noise. We could then solve the inverse problem 1000 times, once for each repeat data set, make histograms of the resulting estimates of the model parameters and infer confidence intervals from them.

Repeat data sets are rarely available. However, it is possible to construct an approximate repeat data set by the *random resampling with duplication* of a single data set. The idea is to treat a set of N data as a pool of hypothetical observations and randomly draw N *realized* observations, which together constitute one repeat data set, from them. Even though the pool and the repeat data set are

both of length N , they are not the same because the latter contains duplications. It can be shown that this process leads to a group of data sets that approximately have the probability density function of the original data.

As an example, suppose that a data set consists of N pairs of (x_i, d_i) observations, where x_i is an auxiliary variable. In *MatLab*, the resampling is performed as

```
rowindex = unidrnd(N,N,1);
xresampled = x( rowindex );
dresampled = dobs( rowindex );
```

(*MatLab* script gda09_17)

Here, the function `unidrnd(N,N,1)` returns a vector of N uniformly distributed integers in the range 1 through N . The inverse problem is then solved for many such repeat data sets, and the resulting estimates of the model parameters are saved. Confidence intervals are estimates as

```
Nbins=50;
mlhmin=min(mlsave);
mlhmax=max(mlsave);
Dmlbins = (mlhmax-mlhmin) / (Nbins-1);
mlbins=mlhmin+Dmlbins*[0:Nbins-1]';
mlhist = hist(mlsave,mlbins);
pm1 = mlhist / (Dmlbins*sum(mlhist));
Pm1 = Dmlbins*cumsum(pm1);
mllow=mlbins(find(Pm1>0.025,1));
mlhigh=mlbins(find(Pm1>0.975,1));
```

(*MatLab* script gda09_17)

Here, estimates of the model parameter m_1 for all the repeat data sets have been saved in the vector `mlsave`. A histogram `mlhist` is computed from them using the `hist()` function, after determining a reasonable range of m_1 values for its bins. The histogram is converted to an empirical probability density function `pm1`, by scaling it so that its integral is unity, and the cumulative sum function `cumsum()` is used to integrate it to a cumulative probability distribution `Pm1`. The `find()` function is then used to determine the values of m_1 that enclose 95% of the area, defining 95% confidence limits for m_1 . An example is shown in [Figure 9.15](#).

9.12 PROBLEMS

9.1. Use the principle of maximum likelihood to estimate the mean μ of N uncorrelated data, each of which is drawn from the same one-sided exponential probability density function $p(d_i) = \mu^{-1} \exp(-d_i/\mu)$ on the interval $(0, \infty)$. Suppose all the N data are equal to unity. What is the variance?

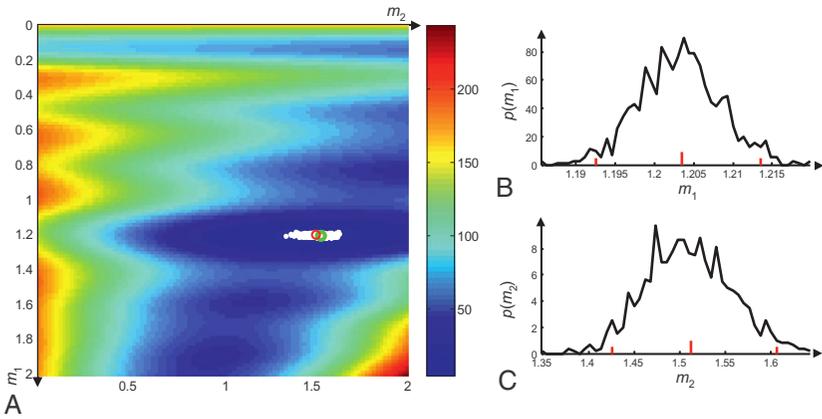


FIGURE 9.15 Bootstrap confidence intervals for model parameters estimated using Newton's method for the same problem as in Figure 9.5. (A) Error surface (colors), showing true solution (red circle), estimated solution (green circle) and bootstrap solutions (white dots). (B) Empirically derived probability density function $p(m_1)$, with m_1^{est} (large red tick) and 95% confidence limits (small red ticks). (C) Same as (B), but for $p(m_2)$. *MatLab* script `gda09_17`.

- 9.2.** Experiment with the Newton's method example of Figure 9.9, by changing the initial guess m_{G} of the solution. Map out the region of the (m_1, m_2) plane for which the solution converges to the global minimum.
- 9.3.** Solve the nonlinear inverse problem $d_i = m_1 z_i^{m_2}$, with z_i an auxiliary variable on the interval $(1, 2)$ and with $\mathbf{m}^{\text{true}} = [3, 2]^T$, using both a linearizing transformation based on taking the logarithm of the equation and by Newton's method and compare the result. Use noisy synthetic data to test your scripts.
- 9.4.** Suppose that the z s in the straight line problem $d_i = m_1 + m_2 z_i$ are considered data, not auxiliary variables. (A) How should the inverse problem be classified? (B) Write a *MatLab* script that solves a test case using an appropriate method.
- 9.5.** A vector \mathbf{d} is constructed by adding together scaled and shifted versions of vectors \mathbf{a} , \mathbf{b} , and \mathbf{c} . The vector \mathbf{d} , of length N , is observed. The vectors, \mathbf{a} , \mathbf{b} , and \mathbf{c} , also of length N , are auxiliary variables. They are related by $d_i = Aa_{i+p} + Bb_{i+q} + Cc_{i+r}$, where A , B , and C are unknown constants and p , q , and r are unknown positive integers (assume $a_{i+p} = 0$ if $i+p > N$ and similarly for \mathbf{b} and \mathbf{c}). This problem can be solved using a three-dimensional grid search over p , q , and r , only, as A , B , and C can be found using least squares once p , q , and r are specified. Write a *MatLab* script that solves a test case for $N=50$. Assume that the error is $E = \mathbf{e}^T \mathbf{e}$ with $e_i = d_i - (Aa_{i+p} + Bb_{i+q} + Cc_{i+r})$.

REFERENCES

- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *Science* 220 (4598), 671–680.
- Tarantola, A., Valette, B., 1982. Generalized non-linear inverse problems solved using the least squares criterion. *Rev. Geophys. Space Phys.* 20, 219–232.

Factor Analysis

10.1 THE FACTOR ANALYSIS PROBLEM

Consider an ocean whose sediment is derived from the simple mixing of continental source rocks A and B (Figure 10.1). Suppose that the concentrations of three elements are determined for many samples of sediment and then plotted on a graph whose axes are percentages of those elements. Since all the sediments are derived from only two source rocks, the sample compositions lie on the triangular portion of a plane bounded by the compositions of A and B (Figure 10.2).

The factor analysis problem is to deduce the number of the source rocks (called *factors*) and their composition from observations of the composition of the sediments (called *samples*). It is therefore a problem in inverse theory. We shall discuss it separately, since it provides an interesting example of the use of some of the vector space analysis techniques developed in Chapter 7.

The model proposes that the samples are simple mixtures (linear combinations) of the factors. If there are N samples containing M elements and if there are p factors, we can state this model algebraically with the equation

$$\mathbf{S} = \mathbf{CF} \quad (10.1)$$

where S_{ij} is the fraction of element j in sample i :

$$\mathbf{S} = \begin{bmatrix} \text{element 1 in sample 1} & \cdots & \text{element } M \text{ in sample 1} \\ \text{element 1 in sample 2} & \cdots & \text{element } M \text{ in sample 2} \\ \vdots & \ddots & \vdots \\ \text{element 1 in sample } N & \cdots & \text{element } M \text{ in sample } N \end{bmatrix} \quad (10.2)$$

(The word “element” is used in the generic sense, since most factor analysis problems will not involve *chemical* elements.) We will refer to individual samples as $\mathbf{s}^{(i)}$, with $\mathbf{s}^{(i)T}$ a row of \mathbf{S} . Similarly, F_{ij} is the fraction of element j in factor i :

$$\mathbf{F} = \begin{bmatrix} \text{element 1 in factor 1} & \cdots & \text{element } M \text{ in factor 1} \\ \text{element 1 in factor 2} & \cdots & \text{element } M \text{ in factor 2} \\ \vdots & \ddots & \vdots \\ \text{element 1 in factor } p & \cdots & \text{element } M \text{ in factor } p \end{bmatrix} \quad (10.3)$$

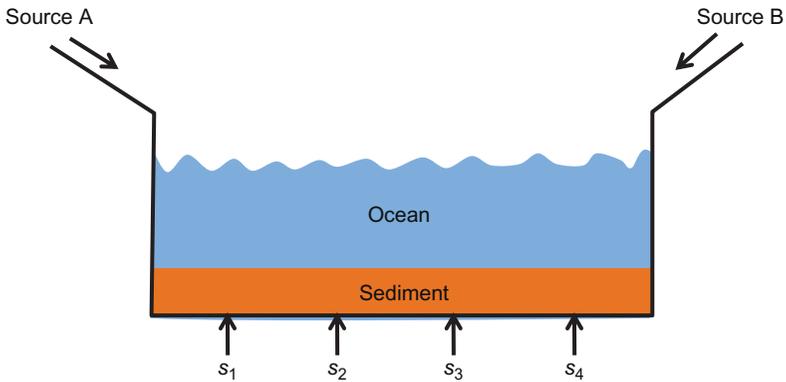


FIGURE 10.1 Material from sources A and B is eroded into the ocean and deposited to form sediment. Samples s_i of the sediment are collected and their chemical composition is determined. The data are used to infer the composition of the sources.

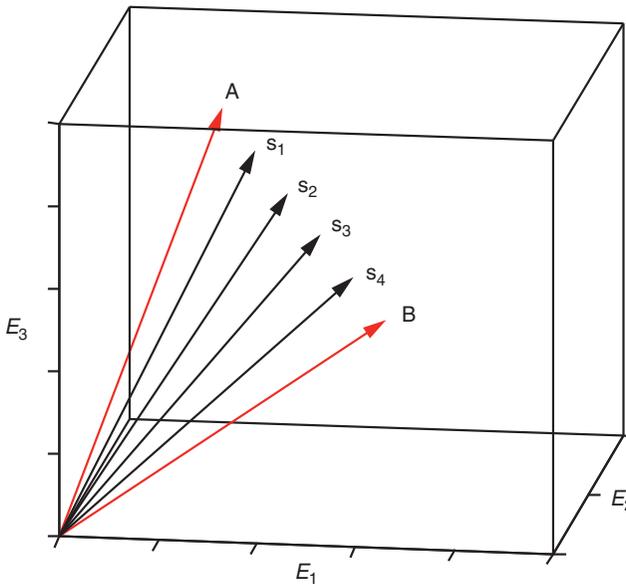


FIGURE 10.2 The composition of the samples s_i (black arrows) lies on a triangular sector of a plane bounded by the composition of the sources A and B (red arrows). *MatLab* script gda10_01.

We will refer to individual factors as $\mathbf{f}^{(i)}$, with $\mathbf{f}^{(i)T}$ a row of \mathbf{F} . C_{ij} is the fraction of factor i in sample j :

$$\mathbf{C} = \begin{bmatrix} \text{factor 1 in sample 1} & \cdots & \text{factor } p \text{ in sample 1} \\ \text{factor 1 in sample 2} & \cdots & \text{factor } p \text{ in sample 2} \\ \vdots & \ddots & \vdots \\ \text{factor 1 in sample } N & \cdots & \text{factor } p \text{ in sample } N \end{bmatrix} \quad (10.4)$$

The elements of the matrix \mathbf{C} are referred to as the *factor loadings* (or sometimes just the *loadings*).

The inverse problem is to factor the matrix \mathbf{S} into \mathbf{C} and \mathbf{F} . Each sample (each row of \mathbf{S}) is represented as a linear combination of factors (rows of \mathbf{F}), with the elements of \mathbf{C} giving the coefficients of the combination. As long as we pick an \mathbf{F} whose rows span the space spanned by the rows of \mathbf{S} , we can perform the factorization. For $p \geq M$, any linearly independent set of factors will do, so in this sense the factor analysis problem is completely nonunique. It is much more interesting to ask what the minimum number of factors is that can be used to represent the samples. Then the factor analysis problem is equivalent to examining the space spanned by \mathbf{S} and determining its dimension. This problem can easily be solved by representing the sample matrix with its singular-value decomposition as

$$\mathbf{S} = \mathbf{U}_p \mathbf{\Lambda}_p \mathbf{V}_p^T = (\mathbf{U}_p \mathbf{\Lambda}_p)(\mathbf{V}_p^T) = \mathbf{C}\mathbf{F} \quad (10.5)$$

Only the eigenvectors with nonzero singular values appear in the decomposition. The number of factors is given by the number of nonzero singular values. One possible set of factors is the p eigenvectors. This set of factors is not unique (Figure 10.3). Any set of factors that spans the p space will do. Mathematically, we transform the factors with any matrix \mathbf{T} that possesses an inverse, in which case $\mathbf{S} = \mathbf{C}\mathbf{F} = \mathbf{C}\mathbf{I}\mathbf{F} = (\mathbf{C}\mathbf{T}^{-1})(\mathbf{T}\mathbf{F}) = \mathbf{C}'\mathbf{F}'$ with the transformed factors being $\mathbf{F}' = (\mathbf{T}\mathbf{F})$ and the new loadings being $\mathbf{C}' = \mathbf{C}\mathbf{T}^{-1}$.

If we write out Equation (10.5), we find that the composition of the i th sample $\mathbf{s}^{(i)}$ is related to the eigenvectors $\mathbf{v}^{(i)}$ and singular values λ_i by

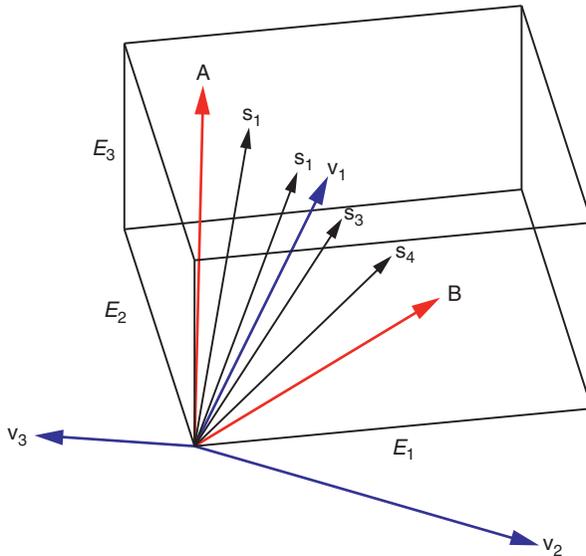


FIGURE 10.3 Eigenvectors \mathbf{v}_1 and \mathbf{v}_2 lie in the plane of the samples (\mathbf{v}_1 is closest to the mean sample). Eigenvector \mathbf{v}_3 is normal to the plane. *MatLab* script `gda10_02`.

$$\begin{aligned}
 \mathbf{s}^{(1)} &= [\mathbf{U}_p]_{11}\lambda_1\mathbf{v}^{(1)} + [\mathbf{U}_p]_{12}\lambda_2\mathbf{v}^{(2)} + \cdots + [\mathbf{U}_p]_{1p}\lambda_p\mathbf{v}^{(p)} \\
 &\quad \cdots \\
 \mathbf{s}^{(N)} &= [\mathbf{U}_p]_{N1}\lambda_1\mathbf{v}^{(1)} + [\mathbf{U}_p]_{N2}\lambda_2\mathbf{v}^{(2)} + \cdots + [\mathbf{U}_p]_{Np}\lambda_p\mathbf{v}^{(p)}
 \end{aligned}
 \tag{10.6}$$

If the singular values are arranged in descending order, then most of each sample is composed of factor 1, with a smaller contribution from factor 2, etc. Because \mathbf{U}_p and $\mathbf{v}^{(p)}$ are composed of unit vectors, on average their elements are of equal size. We have identified the most “important” factors (Figure 10.4). Even if $p=M$, it might be possible to neglect some of the smaller singular values and still achieve a reasonably good prediction of the sample compositions; that is, $\mathbf{S} \approx \mathbf{C}\mathbf{F} = (\mathbf{U}_q\mathbf{\Lambda}_q)(\mathbf{V}_q^T)$ with $q < p$.

The eigenvector with the largest singular value is near the mean of the sample vectors. It is easy to show that the sample mean $\langle \mathbf{s} \rangle$ maximizes the sum of dot products with the data $\sum_i [\mathbf{s}^{(i)} \cdot \langle \mathbf{s} \rangle]$, while the eigenvector \mathbf{v} with largest singular value maximizes the sum of squared dot products $\sum_i [\mathbf{s}^{(i)} \cdot \mathbf{v}]^2$. (To show this, maximize the given functions using Lagrange multipliers, with the constraint that $\langle \mathbf{s} \rangle$ and \mathbf{v} are unit vectors.) As long as most of the samples are in the same quadrant, these two functions have roughly the same maximum.

The *MatLab* code for computing the singular-value decomposition is

```

[U, LAMBDA, V] = svd(S, 0);
lambda = diag(LAMBDA);
F = V';
C = U*LAMBDA;

```

(*MatLab* gda10_04)

Here we use the “economy” version of `svd()`, which has a second argument of zero. In the $N > M$ case, it returns \mathbf{U} as $N \times M$ and $\mathbf{\Lambda}$ as $M \times M$ (since the bottom $N - M$ rows of $\mathbf{\Lambda}$ are zero and hence the right $N - M$ columns of \mathbf{U} do not

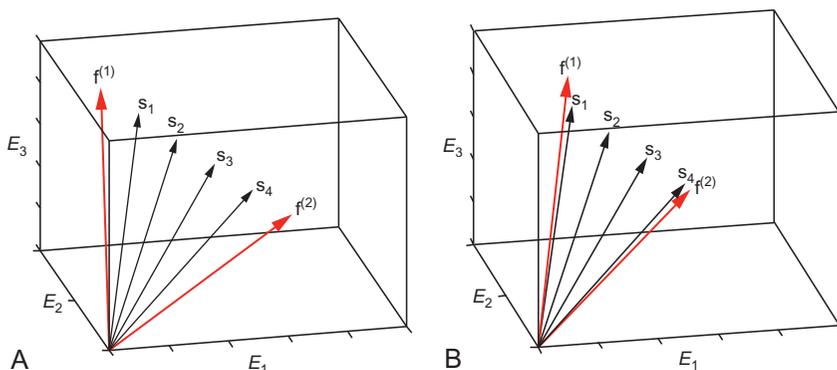


FIGURE 10.4 Any two factors $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$ (red arrows) that lie in the plane of the samples and that bound the range of sample compositions (black arrows) are acceptable, such as those shown in (A) and (B). *MatLab* script gda10_03.

contribute to \mathbf{G}). The `svd()` function does not throw out any of the zero (or near-zero) eigenvalues; this is left to the user. The diagonal of `LAMBDA` has been copied into the column-vector, `lambda`, for convenience.

We apply factor analysis to rock chemistry data taken from a petrologic database (PetDB at www.petdb.org). This database contains chemical information on igneous and metamorphic rocks collected from the floor of all the world's oceans, but we analyze here $N=6356$ samples from the Atlantic Ocean that have the following chemical species: SiO_2 , TiO_2 , Al_2O_3 , $\text{FeO}_{\text{total}}$, MgO , CaO , Na_2O , and K_2O (units of weight percent).

A plot of the singular values of the Atlantic Rock data set (Figure 10.5) reveals that the first value is by far the largest, values 2 through 5 are intermediate in size, and values 6 through 8 are near-zero. The fact that the first singular value λ_1 is much larger than all the others reflects the composition of the rock samples having only a small range of variability. Thus, all rock samples contain a large amount of the first factor, $\mathbf{f}^{(1)}$ —the typical sample. Only four additional factors, $\mathbf{f}^{(2)}$, $\mathbf{f}^{(3)}$, $\mathbf{f}^{(4)}$, and $\mathbf{f}^{(5)}$, out of a total of eight are needed to describe the variability about the typical sample:

Element	$\mathbf{f}^{(1)}$	$\mathbf{f}^{(2)}$	$\mathbf{f}^{(3)}$	$\mathbf{f}^{(4)}$	$\mathbf{f}^{(5)}$
SiO_2	+0.908	+0.007	-0.161	+0.209	+0.309
TiO_2	+0.024	-0.037	-0.126	+0.151	-0.100
Al_2O_3	+0.275	-0.301	+0.567	+0.176	-0.670
FeO-total	+0.177	-0.018	-0.659	-0.427	-0.585
MgO	+0.141	+0.923	+0.255	-0.118	-0.195
CaO	+0.209	-0.226	+0.365	-0.780	+0.207
Na_2O	+0.044	-0.058	-0.0417	+0.302	-0.145
K_2O	+0.003	-0.007	-0.006	+0.073	+0.015

Each role of each of the factors can be understood by examining its elements. Factor 2, for instance, increases the amount of MgO while decreasing mostly Al_2O_3 and CaO , with respect to the typical sample.

The *factor analysis* has reduced the dimensions of variability of the rock data set from eight elements to four factors, improving the effectiveness of scatter plots. *MatLab*'s three-dimensional plotting capabilities are useful in this case, since any three of the four factors can be used as axes and the resulting

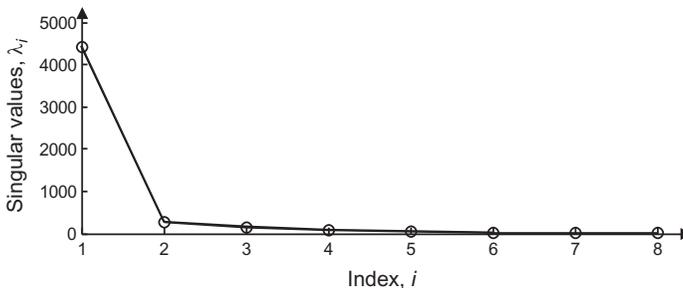


FIGURE 10.5 Singular values λ_i of the Atlantic Ocean Rock dataset. *MatLab* script `gda10_04`.

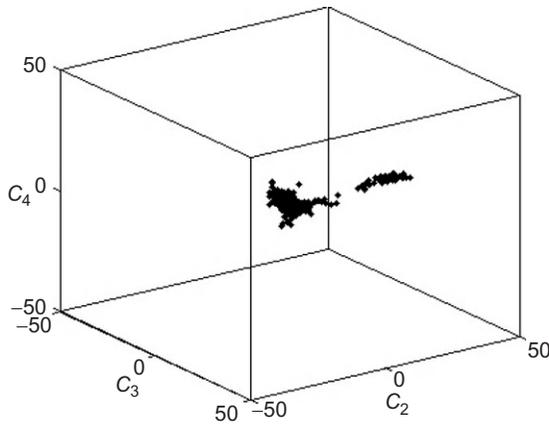


FIGURE 10.6 Three-dimensional perspective view of the coefficients C_i of factors 2, 3, and 4 in each of the rock samples (dots) of the Atlantic Ocean Rock data set. *MatLab* script `gda10_04`.

three-dimensional scatter plot viewed from a variety of perspectives. The following *MatLab* command plots the coefficients of factors 2 through 4 for each sample:

```
plot3(C(:,2), C(:,3), C(:,4), 'k.');
```

(*MatLab* script `gda10_04`)

The plot can then be viewed from different perspectives by using the rotation controls of the Figure Window (Figure 10.6). Note that the samples appear to form two populations, one in which the variability is due to $\mathbf{f}^{(2)}$ and another due to $\mathbf{f}^{(3)}$.

10.2 NORMALIZATION AND PHYSICALITY CONSTRAINTS

In many instances, an element can be important even though it occurs only in trace quantities. In such cases, one cannot neglect factors simply because they have small singular values. They may contain an important amount of the trace elements. It is therefore appropriate to normalize the matrix \mathbf{S} so that there is a direct correspondence between singular-value size and importance. This is usually done by defining a diagonal matrix of weights \mathbf{W} (usually proportional to the reciprocal of the standard deviations of measurement of each of the elements) and then forming a new weighted sample matrix $\mathbf{S}' = \mathbf{S}\mathbf{W}$.

The singular-value decomposition enables one to determine a set of factors that span, or approximately span, the space of samples. These factors, however, are not unique in the sense that one can form linear combinations of factors that also span the space. This transformation is typically a useful thing to do since, ordinarily, the singular-value decomposition eigenvectors violate *a priori* constraints on what “good” factors should be like. One such constraint is that the factors should have a unit L_1 norm, that is, their elements should sum to one.

If the components of a factor represent fractions of chemical elements, for example, it is reasonable that the elements should sum to 100%. Another constraint is that the elements of both the factors and the factor loadings should be nonnegative. Ordinarily a material is composed of a positive combination of components. Given an initial representation of the samples $\mathbf{S} = \mathbf{C}\mathbf{F}\mathbf{W}^{-1}$, we could imagine finding a new representation consisting of linear combinations of the old factors, defined by $\mathbf{F}' = \mathbf{T}\mathbf{F}$, where \mathbf{T} is an arbitrary $p \times p$ transformation matrix. The problem can then be stated.

Find \mathbf{T} subject to the following constraints:

$$\sum_{j=1}^M [\mathbf{F}'\mathbf{W}^{-1}]_{ij} = 1 \quad \text{for all } i$$

$$[\mathbf{C}\mathbf{T}^{-1}]_{ij} \geq 0 \quad \text{and} \quad [\mathbf{F}'\mathbf{W}^{-1}]_{ij} \geq 0 \quad \text{for all } i \text{ and } j \quad (10.7)$$

These conditions do not uniquely determine \mathbf{T} , as can be seen from Figure 10.4. Note that the second constraint is nonlinear in the elements of \mathbf{T} . This is a very difficult constraint to implement and in practice is often ignored.

To find a unique solution, one must add some *a priori* information. One possibility is to find a set of factors that maximize some measure of simplicity. One such measure is *spikiness*: the notion that a factor should have only a few large elements, with the other elements being near-zero. Minerals, for example, obey this principle. While a rock can contain upward of 20 chemical elements, typically it will be composed of minerals such as fosterite (Mg_2SiO_4), anorthite ($\text{CaAl}_2\text{Si}_2\text{O}_8$), rutile (TiO_2), etc., each of which contains just a few elements. Spikiness is more or less equivalent to the idea that the elements of the factors should have *high variance*. The usual formula for estimated variance, σ_d^2 , of a data set, \mathbf{d} , is

$$\sigma_d^2 = \frac{1}{N} \left(\sum_{i=1}^N (d_i - \bar{d})^2 \right) = \frac{1}{N^2} \left(N \sum_{i=1}^N d_i^2 - \left(\sum_{i=1}^N d_i \right)^2 \right) \quad (10.8)$$

Its generalization to a factor, f_i , is

$$\sigma_f^2 = \frac{1}{M^2} \left(M \sum_{i=1}^M f_i^4 - \left(\sum_{i=1}^M f_i^2 \right)^2 \right) \quad (10.9)$$

Note that this is the variance of the *squares* of the elements of the factors. Thus, a factor, \mathbf{f} , has a large variance, σ_f^2 , if the absolute values of its elements have high variation. The signs of the elements are irrelevant.

The *varimax* procedure is a way of constructing a matrix, \mathbf{T} , that increases the variance of the factors while preserving their orthogonality (Kaiser, 1958). It is an iterative procedure, with each iteration operating on only one pair of factors, with other pairs being operated upon in subsequent iterations. The idea is to view the

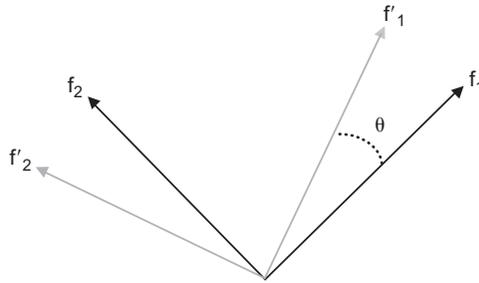


FIGURE 10.7 Two mutually perpendicular factors \mathbf{f}_1 and \mathbf{f}_2 are rotated in their plane by an angle, θ , creating two new mutually orthogonal vectors, \mathbf{f}'_1 and \mathbf{f}'_2 .

factors as vectors and to rotate them in their plane (Figure 10.7) by an angle, θ , chosen to maximize the sum of their variances. The rotation changes only the two factors, leaving the other $p - 2$ factors unchanged, as in the following example:

$$\begin{bmatrix} \mathbf{f}^{(1)\text{T}} \\ \mathbf{f}^{(2)\text{T}} \\ \cos(\theta)\mathbf{f}^{(3)\text{T}} + \sin(\theta)\mathbf{f}^{(5)\text{T}} \\ \mathbf{f}^{(4)\text{T}} \\ -\sin(\theta)\mathbf{f}^{(3)\text{T}} + \cos(\theta)\mathbf{f}^{(5)\text{T}} \\ \mathbf{f}^{(6)\text{T}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{f}^{(1)\text{T}} \\ \mathbf{f}^{(2)\text{T}} \\ \mathbf{f}^{(3)\text{T}} \\ \mathbf{f}^{(4)\text{T}} \\ \mathbf{f}^{(5)\text{T}} \\ \mathbf{f}^{(6)\text{T}} \end{bmatrix} \quad (10.10)$$

or $\mathbf{F}' = \mathbf{T}\mathbf{F}$. Here, only the pair, $\mathbf{f}^{(3)}$ and $\mathbf{f}^{(5)}$, is changed.

In Equation (10.10), the matrix, \mathbf{T} , represents a rotation of *one pair* of vectors. The rotation matrix for many such rotations is just the product of a series of pair-wise rotations. Note that the matrix, \mathbf{T} , obeys the rule, $\mathbf{T}^{-1} = \mathbf{T}^{\text{T}}$ (that is, \mathbf{T} is a *unary* matrix). For a given pair of factors, \mathbf{f}^A and \mathbf{f}^B , the rotation angle θ is determined by minimizing $\Phi(\theta) = M^2(\sigma_{fA}^2 + \sigma_{fB}^2)$ with respect to θ (that is, by solving $d\Phi/d\theta = 0$).

The minimization requires a substantial amount of algebraic and trigonometric manipulation, so we omit it here. The result is (Kaiser, 1958)

$$\theta = \frac{1}{4} \tan^{-1} \frac{2M \sum_i u_i v_i - \sum_i u_i \sum_i v_i}{M \sum_i (u_i^2 - v_i^2) - \left(\left(\sum_i u_i \right)^2 - \left(\sum_i v_i \right)^2 \right)} \quad \text{with} \quad (10.11)$$

$$u_i = (f_i^A)^2 - (f_i^B)^2 \quad \text{and} \quad v_i = 2f_i^A f_i^B$$

By way of example, we note that the two vectors

$$\mathbf{f}^A = \frac{1}{2} [1 \ 1 \ 1 \ 1]^{\text{T}} \quad \text{and} \quad \mathbf{f}^B = \frac{1}{2} [1 \ -1 \ 1 \ -1]^{\text{T}} \quad (10.12)$$

are extreme examples of two *nonspiky* orthogonal vectors because all their elements have the same absolute value. When applied to them, the varimax procedure returns

$$\mathbf{f}^{A'} = \frac{1}{\sqrt{2}}[1 \ 0 \ 1 \ 0]^T \quad \text{and} \quad \mathbf{f}^{B'} = \frac{1}{\sqrt{2}}[0 \ -1 \ 0 \ -1]^T \quad (10.13)$$

which are significantly spikier than the originals. The *MatLab* code is

```
u = fA.^2 - fB.^2;
v = 2 * fA .* fB;
```

```
A = 2 * M * u' * v;
B = sum(u) * sum(v);
top = A - B;
```

```
C = M * (u' * u - v' * v);
D = (sum(u)^2) - (sum(v)^2);
bot = C - D;
q = 0.25 * atan2(top, bot);
```

```
cq = cos(q);
sq = sin(q);
```

```
fAp = cq * fA + sq * fB;
fBp = -sq * fA + cq * fB;
```

(*MatLab* gda10_05)

Here, the original pair of factors are f_A and f_B , and the rotated pair are f_{Ap} and f_{Bp} .

We now apply this procedure to factors f_2 through f_5 of the Atlantic Rock data set (that is, the factors related to deviations about the typical rock). The varimax procedure is applied to all pairs of these factors and achieves convergence after several such iterations. The *MatLab* code for the loops are

```
FP=F;

% spike these factors using the varimax procedure
k = [2, 3, 4, 5]';
Nk = length(k);

for iter = [1:3]
    for ii = [1:Nk]
        for jj = [ii+1:Nk]

            % spike factors i and j
            i=k(ii);
            j=k(jj);
```

```

% copy factors from matrix to vectors
fA = FP(i, :)';
fB = FP(j, :)';

% standard varimax procedure to determine rotation angle  $\alpha$ 
- - -

% copy rotated factors back to matrix
FP(i, :) = fAp';
FP(j, :) = fBp';

end
end
end

```

(MatLab gda10_05)

Here the rotated matrix of factors \mathbf{FP} is initialized to the original matrix of factors \mathbf{F} and then modified by the varimax procedure (omitted and replaced with a “- - -”), with each pass through the inner loop rotating one pair of factors. The procedure converges very rapidly, with three iterations of the outside loop being sufficient. The resulting factors (Figure 10.8) are much spikier than the original ones. Each now involves mainly variations in one chemical element. For example, \mathbf{f}'_2 mostly represents variations in MgO, and \mathbf{f}'_5 mostly represents variations in Al_2O_3 .

Another possible way of adding *a priori* information is to find factors that are in some sense close to a set of *a priori* factors. If closeness is measured by the L_1 or L_2 norm and if the constraint on the positivity of the factor loadings is

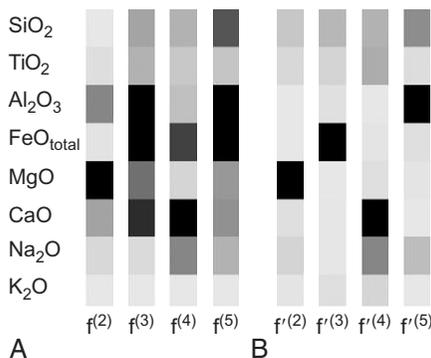


FIGURE 10.8 (A) Factors $\mathbf{f}^{(2)}$ through $\mathbf{f}^{(5)}$ of the Atlantic Rock data set, as calculated by singular-value decomposition. (B) Factors $\mathbf{f}'^{(2)}$ through $\mathbf{f}'^{(5)}$, after application of the varimax procedure. MatLab script gda10_05.

omitted, then this problem can be solved using the techniques of Chapters 7 and 12. One advantage of this latter approach is that it permits one to test whether a particular set of *a priori* factors can be factors of the problem (that is, whether or not the distance between *a priori* factors and actual factors can be reduced to an insignificant amount).

10.3 Q-MODE AND R-MODE FACTOR ANALYSIS

The eigenvectors \mathbf{U} and \mathbf{V} play completely symmetric roles in the singular-value decomposition of the sample matrix $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$. We introduced an asymmetry when we grouped them as $\mathbf{S} = (\mathbf{U}\mathbf{\Lambda})(\mathbf{V}^T) = \mathbf{C}\mathbf{F}$ to define the loadings \mathbf{C} and factors \mathbf{F} .

This grouping is associated with the term *R-mode factor analysis*. It is appropriate when the focus is on patterns among the elements, which is to say, reducing a large number of elements to a smaller number of factors. Thus, for example, we might note that the elements in the Atlantic Rock data set contain a pattern, associated with factor $\mathbf{f}^{(2)}$, in which Al_2O_3 and MgO are strongly and negatively correlated and another pattern, associated with factor $\mathbf{f}^{(3)}$, in which Al_2O_3 and $\text{FeO}_{\text{total}}$ are strongly and negatively correlated. The effect of these correlations is to reduce the effective number of elements, that is, to allow us to substitute a small number of factors for a large number of elements.

Alternately, we could have grouped the singular-value decomposition as $\mathbf{S} = (\mathbf{U})(\mathbf{\Lambda}\mathbf{V}^T)$, an approach associated with the term *Q-mode factor analysis*. The equivalent transposed form $\mathbf{S}^T = (\mathbf{V}\mathbf{\Lambda})(\mathbf{U}^T)$ is more frequently encountered in the literature and is also more easily understood, since it can be interpreted as “normal factor analysis” applied to the matrix \mathbf{S}^T . The transposition has reversed the sense of samples and elements, so the factor matrix \mathbf{U}^T quantifies patterns of variability among samples, in the same way that \mathbf{V}^T quantifies patterns of variability among elements. To pursue this comparison further, consider an R-mode problem in which there is only one factor, $\mathbf{v}^{(1)} = [1, 1, \dots]^T$. This factor implies that all of the samples in the data table contain a 1:1 ratio of elements 1 and 2. Similarly, a Q-mode problem in which there is only one factor, $\mathbf{u}^{(1)} = [1, 1, \dots]^T$ implies that all of the elements in the transposed data table have a 1:1 ratio of samples 1 and 2. This approach is especially useful in detecting clustering among the samples; indeed, Q-mode factor analysis is often referred to as a form of *cluster analysis*.

10.4 EMPIRICAL ORTHOGONAL FUNCTION ANALYSIS

Factor analysis need not be limited to data that contain actual mixtures of components. Given any set of vectors $\mathbf{s}^{(i)}$, one can perform the singular-value decomposition and represent $\mathbf{s}^{(i)}$ as a linear combination of a set of orthogonal factors. Even when the factors have no obvious physical interpretation, the decomposition can be useful as a tool for quantifying the similarities between the

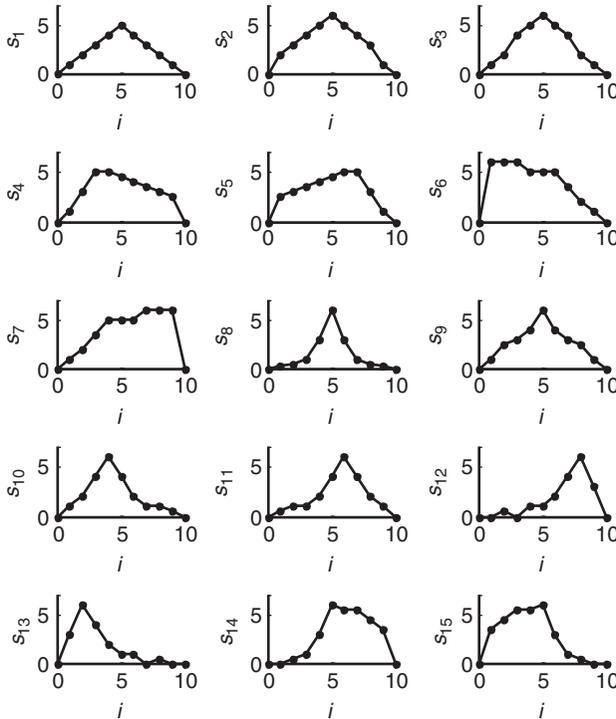


FIGURE 10.9 A set of hypothetical mountain profiles. The variability of shape will be determined using factor analysis. *MatLab* script `gda10_06`

$\mathbf{s}^{(i)}$ vectors. This kind of factor analysis is often called *empirical orthogonal function (EOF)* analysis.

As an example of this application of factor analysis, consider the set of $N = 14$ shapes shown in Figure 10.9. These shapes might represent profiles of mountains or other subjects of interest. The problem we shall consider is how these profiles might be ordered to bring out the similarities and differences between the shapes. A geomorphologist might desire such an ordering because, when combined with other kinds of geological information, it might reveal the kinds of erosional processes that cause the shape of mountains to evolve with time.

We begin by discretizing each profile and representing it as a unit vector (in this case of length $M = 11$). These unit vectors make up the matrix \mathbf{S} , on which we perform factor analysis. Since the factors do not represent any particular physical object, there is no need to impose any positivity constraints on them, and we use the untransformed singular-value decomposition factors. The three most important EOFs (factors) (that is, the ones with the three largest singular values) are shown in Figure 10.10. The first EOF, as expected, appears to be simply an “average” mountain; the second seems to control the skewness, or degree of asymmetry, of the mountain; and the third, the sharpness of the mountain’s summit.

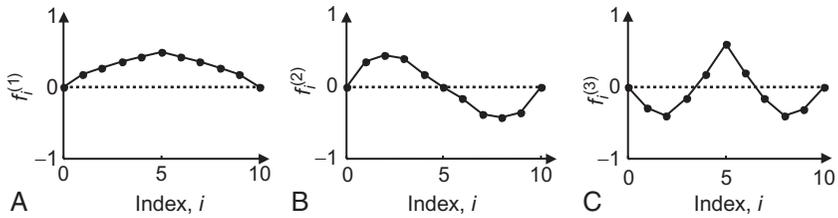


FIGURE 10.10 The three largest factors $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ in the representation of the mountain profiles in Figure 10.9. (A) The factor with the largest singular value $\lambda_1=38.4$ has the shape of the average profile. (B) The factor with the second largest singular value $\lambda_2=12.7$ quantifies the asymmetry of the profiles. (C) The factor with the third largest singular value $\lambda_3=7.4$ quantifies the sharpness of the mountain summits. *MatLab* script gda10_06.

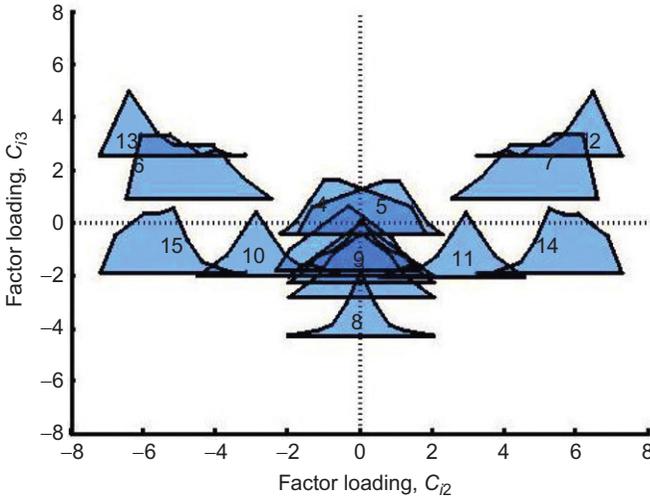


FIGURE 10.11 The mountain profiles of Figure 10.9, arranged according to the relative amounts of factors 2 and 3 contained in each profile's orthogonal decomposition; that is, by the size of the factor loadings C_{12} and C_{13} . *MatLab* script gda10_06.

We emphasize, however, that this interpretation was made after the EOF analysis and was not based on any *a priori* notions of how mountains might differ. We can then use the loadings as a measure of the similarities between the mountains. Since the amount of the first factor does not vary much between mountains, we use a two-dimensional ordering based on the relative amounts of the second and third factors in each of the mountain profiles (Figure 10.11).

EOF analysis is especially useful when the data have an ordering in space, time, or some other sequential variable. For instance, suppose that profiles in Figure 10.9 are measured at sequential times. Then we can understand the model equation $\mathbf{S} = \mathbf{C}\mathbf{F}$ to mean

$$S(t_i, x_j) = \sum_{k=1}^p C_k(t_i) F_k(x_j) \quad (10.14)$$

Here the quantity $S(t_i, x_j)$, which varies with both time and space, has been broken up into the sum of two sets of function, the EOFs $F_k(x_j)$, which vary only with space, and the corresponding loadings $C_k(t_i)$, which vary only with time. A plot of the k th loading $C_k(t_i)$ as a function of time t_i reveals how the importance of the k th EOF varies with time.

This analysis can be extended to functions of two or more spatial dimensions by writing Equation (10.14) as

$$S(t_i, \mathbf{x}^{(j)}) = \sum_{k=1}^p C_k(t_i) F_k(\mathbf{x}^{(j)}) \quad (10.15)$$

Here the spatial observation points \mathbf{x} are multidimensional, but they have been given a linear ordering through the index k . As an example, suppose that the data are a sequence of two-dimensional images, where each image represents a physical parameter observed on the (x, y) plane. This image can be *unfolded* (reordered) into a vector (Figure 10.12), which then becomes a row of the sample matrix \mathbf{S} . The resulting EOFs have this same ordering and must be folded back into two-dimensional images before being interpreted.

As an example, we consider a sequence of $N = 25$ images, each of which contains a grid of $20 \times 20 = 400$ pixels (Figure 10.13). Each image represents the spatial variation of a physical parameter such as pressure or temperature at a fixed time, with the overall sequence being time-sequential. These data are synthetic and are constructed by summing three spatial patterns (EOFs) with coefficients (loadings) that vary systematically with time, and then adding random noise. As expected, only $p = 3$ singular values are found to be significant (Figure 10.14).

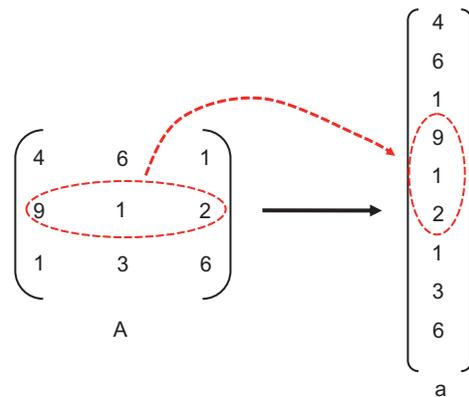


FIGURE 10.12 A matrix \mathbf{A} representing a discrete version of a two-dimensional function $A_{ij} = a(x_i, y_j)$ is unfolded row-wise into a vector \mathbf{a} using the rule $a_k = A_{ij}$ with $k = (i - 1)M + j$.

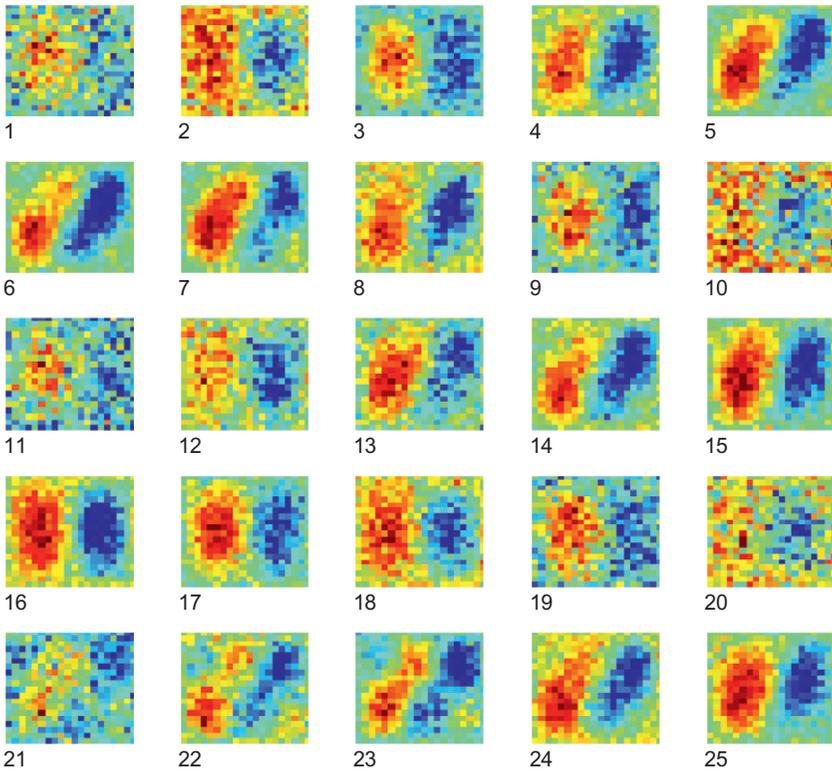


FIGURE 10.13 Time sequence of $N=25$ images. Each image represents a parameter, such as pressure or temperature, that varies spatially in the (x, y) plane. *MatLab* script gda10_07.

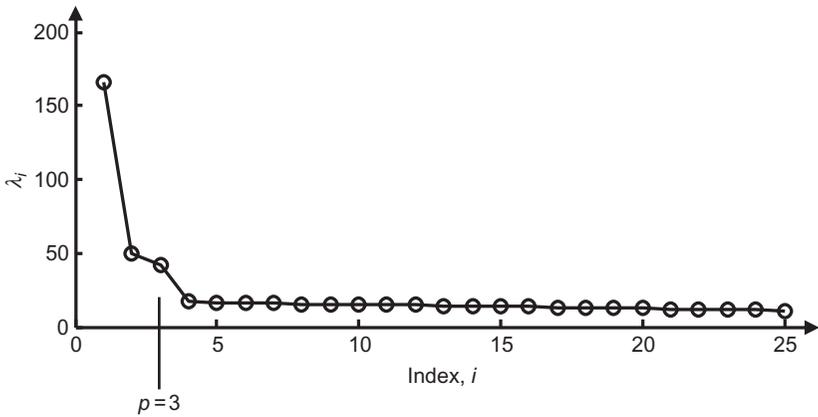


FIGURE 10.14 Singular values λ_i of the image sequence shown in Figure 10.12. Only $p=3$ singular values have significant amplitudes. *MatLab* script gda10_07.

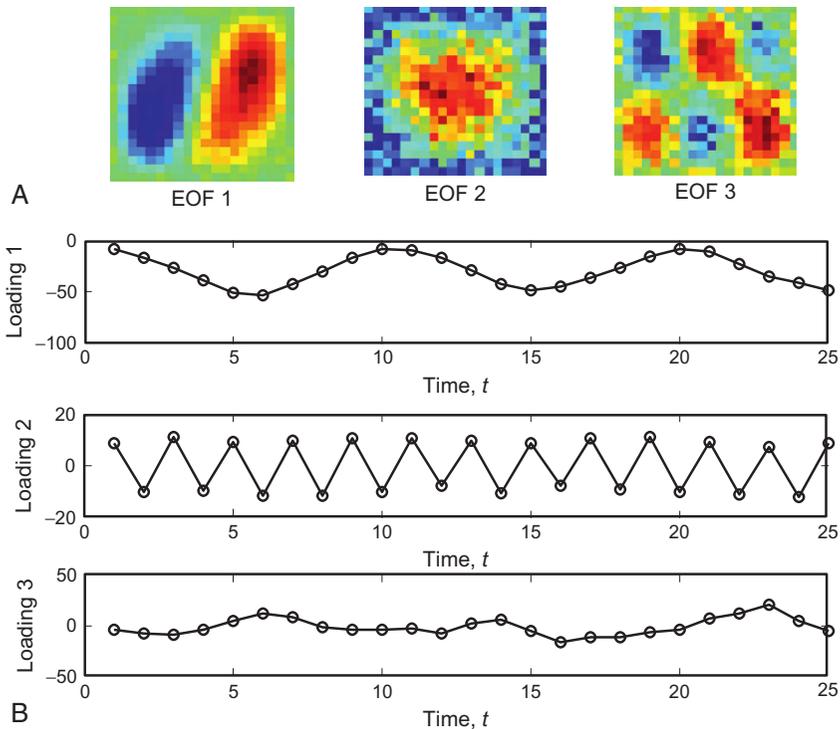


FIGURE 10.15 (A) First three empirical orthogonal functions (EOFs) of the image sequence shown in Figure 10.12. (B) Corresponding loadings as a function of time, t . *MatLab* script `gda10_07`.

The corresponding EOFs and loadings are shown in Figure 10.15. Had this analysis been based upon actual data, the time variation of each of the loadings, which have different periodicities, would be of special interest and might possibly provide insight into the physical processes associated with each of the EOFs. A similar example that uses actual ocean temperature data to examine the El Niño-Southern Oscillation climate instability is given by Menke and Menke (2011, their Section 8.5).

10.5 PROBLEMS

- 10.1** Suppose that a set of $N > M$ samples are represented as $\mathbf{S} \approx \mathbf{C}_p \mathbf{F}_p$ where the matrix \mathbf{F}_p contains $p < M$ factors whose values are prescribed (that is, known *a priori*). (A) How can the loadings \mathbf{C}_p be determined? (B) Write a *MatLab* script that implements your procedure for the case $M = 3$, $N = 10$, $p = 2$. (C) Make a three-dimensional plot of your results.

- 10.2** Write a *MatLab* script that verifies the varimax result given in Equation (10.12). Use the following steps. (A) Compute the factors \mathbf{f}'^A and \mathbf{f}'^B for a complete suite of angles θ using the rotation

$$\begin{aligned}\mathbf{f}'^A &= \cos(\theta)\mathbf{f}^A + \sin(\theta)\mathbf{f}^B \\ \mathbf{f}'^B &= -\sin(\theta)\mathbf{f}^A + \cos(\theta)\mathbf{f}^B\end{aligned}$$

(B) Compute and plot the variance $\sigma_{f'_A}^2 + \sigma_{f'_B}^2$ as a function of angle θ . (C) Note the angle of the minimum variance and verify that the angle is the one predicted by the varimax formula. (D) Verify that the factors corresponding to the angle are as stated in Equation (10.12).

- 10.3** Suppose that a data set represents a function of three spatial dimensions; that is, with samples $S(t, x, y, z)$ on an evenly spaced three-dimensional grid. How can these samples be unfolded into a matrix, \mathbf{S} ?

REFERENCES

- Kaiser, H.F., 1958. The varimax criterion for analytic rotation in factor analysis. *Psychometrika* 23, 187–200.
- Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc, Oxford, UK 263pp.

Continuous Inverse Theory and Tomography

11.1 THE BACKUS-GILBERT INVERSE PROBLEM

While continuous inverse problems are not the main subject of this book, we will cover them briefly to illustrate their relationship to discrete problems. Discrete and continuous inverse problems differ in their assumptions about the model parameters. Whereas the model parameters are treated as a finite-length vector in discrete inverse theory, they are treated as a continuous function in continuous inverse theory. The standard form of the continuous inverse problem is

$$d_i = \int_a^b G_i(z)m(z)dz \quad (11.1)$$

when the model function $m(z)$ varies only with one parameter, such as depth z . When the model function depends on several—say L —variables, then Equation (11.1) must be generalized to

$$d_i = \int_V G_i(\mathbf{x})m(\mathbf{x})d^Lx \quad (11.2)$$

where d^Lx is the volume element in the space of \mathbf{x} .

The “solution” of a discrete problem can be viewed as either an estimate of the model parameter vector \mathbf{m}^{est} or a series of weighted averages of the model parameters, $\mathbf{m}^{\text{avg}} = \mathbf{R}\mathbf{m}^{\text{true}}$, where \mathbf{R} is the resolution matrix (see Section 4.3). If the discrete inverse problem is very underdetermined, then the interpretation of the solution in terms of weighted averages is most sensible, since a single model parameter is very poorly resolved. Continuous inverse problems can be viewed as the limit of discrete inverse problems as the number of model parameters becomes infinite, and they are inherently underdetermined. Attempts to estimate the model function $m(\mathbf{x})$ at a specific point $\mathbf{x} = \mathbf{x}'$ are futile. All determinations

of the model function must be made in terms of local averages, which are simple generalizations of the discrete case, $m_i^{\text{avg}} = \sum_j G_{ij}^{-g} d_j = \sum_j R_{ij} m_j^{\text{true}}$ where $R_{ij} = \sum_k G_{ik}^{-g} G_{kj}$

$$m^{\text{avg}}(\mathbf{x}') = \sum_{i=1}^N G_i^{-g}(\mathbf{x}') d_i = \int R(\mathbf{x}', \mathbf{x}) m^{\text{true}}(\mathbf{x}) d^L x \quad (11.3)$$

where

$$R(\mathbf{x}', \mathbf{x}) = \sum_{i=1}^N G_i^{-g}(\mathbf{x}') G_i(\mathbf{x}) \quad (11.4)$$

Here, $G_i^{-g}(\mathbf{x}')$ is the continuous analogy to the generalized inverse G_{ij}^{-g} and the averaging function $R(\mathbf{x}', \mathbf{x})$ (often called the *resolving kernel*) is the analogy to the model resolution matrix R_{ij} . The average is localized near the target point \mathbf{x}' if the resolving kernel is peaked near \mathbf{x}' . The solution of the continuous inverse problem involves constructing the most peaked resolving kernel possible with a given set of measurements, that is, with a given set of data kernels, $G_i(\mathbf{x})$. The spread of the resolution function is quantified by (compare with [Equation \(4.23\)](#))

$$J(\mathbf{x}') = \int w(\mathbf{x}', \mathbf{x}) R^2(\mathbf{x}', \mathbf{x}) d^L x \quad (11.5)$$

Here, $w(\mathbf{x}', \mathbf{x})$ is a nonnegative function that is zero at the point \mathbf{x}' and that grows monotonically away from that point. One commonly used choice is the quadratic function $w(\mathbf{x}', \mathbf{x}) = |\mathbf{x}' - \mathbf{x}|^2$. Other, more complicated functions can be meaningful if the elements of \mathbf{x} have an interpretation other than spatial position. After inserting the definition of the resolving kernel ([Equation \(11.3\)](#)) into the definition of the spread ([Equation \(11.5\)](#)), we find

$$\begin{aligned} J(\mathbf{x}') &= \int w(\mathbf{x}', \mathbf{x}) R(\mathbf{x}', \mathbf{x}) R(\mathbf{x}', \mathbf{x}) d^L x \\ &= \int w(\mathbf{x}', \mathbf{x}) \sum_{i=1}^N G_i^{-g}(\mathbf{x}') G_i(\mathbf{x}) \sum_{j=1}^N G_j^{-g}(\mathbf{x}') G_j(\mathbf{x}) d^L x \\ &= \sum_{i=1}^N \sum_{j=1}^N G_i^{-g}(\mathbf{x}') G_j^{-g}(\mathbf{x}') \int w(\mathbf{x}', \mathbf{x}) G_i(\mathbf{x}) G_j(\mathbf{x}) d^L x \\ &= \sum_{i=1}^N \sum_{j=1}^N G_i^{-g}(\mathbf{x}') G_j^{-g}(\mathbf{x}') [\mathbf{S}(\mathbf{x}')]_{ij} \end{aligned} \quad (11.6)$$

where

$$[\mathbf{S}(\mathbf{x}')]_{ij} = \int w(\mathbf{x}', \mathbf{x}) G_i(\mathbf{x}) G_j(\mathbf{x}) d^L x \quad (11.7)$$

[Equation \(11.7\)](#) might be termed an *overlap integral*, since it is large only when the two data kernels overlap, that is, when they are simultaneously large in the same region of space. The continuous spread function has now been

manipulated into a form completely analogous to the discrete spread function in Equation (4.26). The generalized inverse that minimizes the spread of the resolution is the precise analogy of Equation (4.34)

$$G_l^{-g}(\mathbf{x}') = \frac{\sum_{i=1}^N u_i [\mathbf{S}^{-1}(\mathbf{x}')]_{il}}{\sum_{i=1}^N \sum_{j=1}^N u_i [\mathbf{S}^{-1}(\mathbf{x}')]_{ij} u_j} \quad \text{where} \quad u_i = \int G_i(\mathbf{x}) d^L x \quad (11.8)$$

11.2 RESOLUTION AND VARIANCE TRADE-OFF

Since the data \mathbf{d} are determined only up to some error quantified by the covariance matrix $[\text{cov } \mathbf{d}]$, the localized average $m^{\text{avg}}(\mathbf{x}')$ is determined up to some corresponding error

$$\text{var}[m^{\text{avg}}(\mathbf{x}')] = \sum_{i=1}^N \sum_{j=1}^N G_i^{-g}(\mathbf{x}') [\text{cov } \mathbf{d}]_{ij} G_j^{-g}(\mathbf{x}') \quad (11.9)$$

As in the discrete case, the generalized inverse that minimizes the spread of resolution may lead to a localized average with large error bounds. A slightly less localized average may be desirable because it may have much less error. This generalized inverse may be found by minimizing a weighted average of the spread of resolution and size of variance

$$\text{minimize } J'(\mathbf{x}') = \alpha \int w(\mathbf{x}', \mathbf{x}) R^2(\mathbf{x}', \mathbf{x}) d^L x + (1 - \alpha) \text{var}[m^{\text{avg}}(\mathbf{x}')] \quad (11.10)$$

The parameter α , which varies between 0 and 1, quantifies the relative weight given to spread of resolution and size of variance. As in the discrete case (see Section 4.10), the corresponding generalized inverse can be found by using Equation (11.7), where all instances of $\mathbf{S}(\mathbf{x}')$ are replaced by

$$[\mathbf{S}'(\mathbf{x}')]_{ij} = \alpha \int w(\mathbf{x}', \mathbf{x}) G_i(\mathbf{x}) G_j(\mathbf{x}) d^L x + (1 - \alpha) [\text{cov } \mathbf{d}]_{ij} \quad (11.11)$$

Backus and Gilbert (1968) prove a number of important properties of the trade-off curve (Figure 11.1), which is a plot of size of variance against spread of resolution, including the fact that the variance decreases monotonically with spread.

11.3 APPROXIMATING CONTINUOUS INVERSE PROBLEMS AS DISCRETE PROBLEMS

A continuous inverse problem can be converted into a discrete one with the assumption that the model function can be represented by a finite number M of coefficients, that is

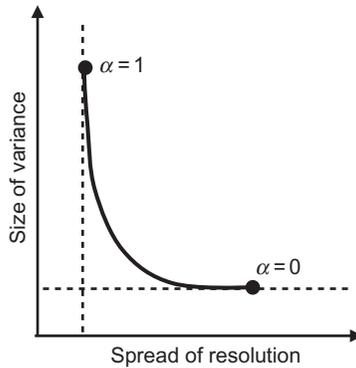


FIGURE 11.1 Typical trade-off of resolution and variance for a linear continuous inverse problem. Note that the size(spread) function decreases monotonically with spread and that it is tangent to the two asymptotes at the endpoints $\alpha=1$ and $\alpha=0$.

$$m(\mathbf{x}) \approx \sum_{j=1}^N m_j f_j(\mathbf{x}) \quad (11.12)$$

The particular choice of the functions $f_j(\mathbf{x})$ implies particular *a priori* information about the behavior of $m(\mathbf{x})$, so the solution one obtains is sensitive to the choice. One commonly used set of functions assumes that the model is constant within certain subregions V_j of the space of model parameters (e.g., voxels). In this case, $f_j(\mathbf{x})$ is unity inside V_j and zero outside it, and m_j is the value of the model in each voxel. In one dimension, where the x -axis is divided into intervals of width Δx

$$m(x) \approx \sum_{j=1}^M m_j f_j(x) \quad \text{with} \quad f_j(x) = H(x - j\Delta x) - H(x - (j+1)\Delta x) \quad (11.13)$$

Here $H(x - \xi)$ is the Heaviside step function, which is zero for $x < \xi$ and unity for $x > \xi$. Many other choices of $f_j(\mathbf{x})$ are encountered, based on polynomial approximations, splines, and truncated Fourier series representations of $m(\mathbf{x})$.

Inserting Equation (11.12) into the standard form of the continuous problem (Equation 11.2) leads to a discrete problem

$$d_i = \int G_i(\mathbf{x}) \sum_{j=1}^m m_j f_j(\mathbf{x}) d^L x = \sum_{j=1}^M \left\{ \int G_i(\mathbf{x}) f_j(\mathbf{x}) d^L x \right\} m_j \quad (11.14)$$

So the discrete form of the equation is $\mathbf{d} = \mathbf{Gm}$

$$d_i = \sum_{j=1}^M G_{ij} m_j \quad \text{with} \quad G_{ij} = \int G_i(\mathbf{x}) f_j(\mathbf{x}) d^L x \quad (11.15)$$

In the special case of voxels of volume V_j centered on the points $\mathbf{x}^{(j)}$, Equation (11.15) becomes

$$G_{ij} = \int_{V_j} G_i(\mathbf{x}) d^L x \quad (11.16)$$

If the data kernel varies slowly with position so that it is approximately constant in the voxel, then we may approximate the integral as just its integrand evaluated at $\mathbf{x}^{(j)}$ times the volume V_j :

$$G_{ij} = \int_{V_j} G_i(\mathbf{x}) d^L x \approx G_i(\mathbf{x}^{(j)}) V_j \quad (11.17)$$

In one dimension, this is equivalent to the Reimann summation approximation to the integral

$$d_i = \int G_i(x) m(x) dx \approx \sum_{j=1}^M \{G_i(j\Delta x) \Delta x\} m_j = \sum_{j=1}^M G_{ij} m_j \quad (11.18)$$

Two different factors control the choice of the size of the voxels. The first is dictated by an *a priori* assumption of the smoothness of the model. The second becomes important only when one uses Equation (11.17) in preference to Equation (11.16). Then the subregion must be small enough that the data kernels $G_i(\mathbf{x})$ are approximately constant in the voxel. This second requirement often forces the voxels to be much smaller than dictated by the first requirement, so Equation (11.16) should be used whenever the data kernels can be integrated analytically. Equation (11.17) fails completely whenever a data kernel has an integrable singularity within the subregion. This case commonly arises in problems involving the use of seismic rays to determine acoustic velocity structure.

11.4 TOMOGRAPHY AND CONTINUOUS INVERSE THEORY

The term “tomography” has come to be used in geophysics almost synonymously with the term “inverse theory.” Tomography is derived from the Greek word *tomos*, that is, slice, and denotes forming an image of an object from measurements made from slices (or rays) through it. We consider tomography a subset of inverse theory, distinguished by a special form of the data kernel that involves measurements made along rays. The model function in tomography is a function of two or more variables and is related to the data by

$$d_i = \int_{C_i} m[x(s), y(s)] ds \quad (11.19)$$

Here, the model function is integrated along a curved ray C_i having arc length s . This integral is equivalent to the one in a standard continuous problem

(Equation (11.2)) when the data kernel is $G_i(x, y) = \delta\{x(s) - x_i[y(s)]\} ds/dy$, where $\delta(x)$ is the Dirac delta function:

$$d_i = \iint m(x, y) \delta\{x(s) - x_i[y(s)]\} \frac{ds}{dy} dx dy = \int_{C_i} m[x(s), y(s)] ds \quad (11.20)$$

Here x is supposed to vary with y along the curve C_i and y is supposed to vary with arc length s .

While the tomography problem is a special case of a continuous inverse problem, several factors limit the applicability of the formulas of the previous sections. First, the Dirac delta functions in the data kernel are not square integrable so that the overlap integrals S_{ij} (Equation (11.7)) have nonintegrable singularities at points where rays intersect. Further, in three-dimensional cases, the rays may not intersect at all so that all the S_{ij} may be identically zero. Neither of these problems is insurmountable, and they can be overcome by replacing the rays with tubes of finite cross-sectional width. (Rays are often an idealization of a finite-width process anyway, as in acoustic wave propagation, where they are an infinitesimal wavelength approximation.) Since this approximation is equivalent to some statement about the smoothness of the model function $m(x, y)$, it often suffices to discretize the continuous problem by dividing it into constant m subregions, where the subregions are large enough to guarantee a reasonable number containing more than one ray. The discrete inverse problem is then of the form $d_i = \sum_j G_{ij} m_j$, where the data kernel G_{ij} gives the arc length of the i th ray in the j th subregion. The concepts of resolution and variance, now interpreted in the discrete fashion of Chapter 4, are still applicable and of considerable importance.

11.5 TOMOGRAPHY AND THE RADON TRANSFORM

The simplest tomography problem involves straight-line rays and a two-dimensional model function $m(x, y)$ and is called Radon's problem. By historical convention, the straight-line rays C_i in Equation (11.19) are parameterized by their perpendicular distance u from the origin and the angle θ (Figure 11.2) that the perpendicular makes with the x -axis. Position (x, y) and ray coordinates (u, s) , where s is arc length, are related by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} u \\ s \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} u \\ s \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (11.21)$$

The tomography problem is then

$$d(u, \theta) = \int_{-\infty}^{+\infty} m(x = u \cos \theta - s \sin \theta, y = u \sin \theta + s \cos \theta) ds \quad (11.22)$$

In realistic experiments, $d(u, \theta)$ is sampled only at discrete points $d_i = d(u_i, \theta_i)$. Nevertheless, much insight can be gained into the behavior of Equation (11.22) by regarding (u, θ) as continuous variables. Equation (11.22) is then an integral

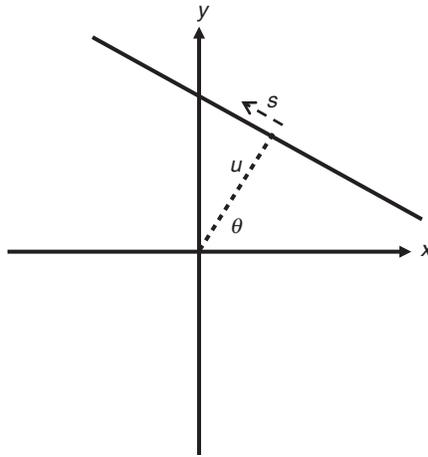


FIGURE 11.2 The Radon transform is performed by integrating a function of (x,y) along straight lines (bold) parameterized by the arc length, s ; perpendicular distance, u ; and angle, θ .

transform that transforms variables (x,y) into two new variables (u,θ) and is called a *Radon transform*.

11.6 THE FOURIER SLICE THEOREM

The Radon transform is similar to another integral transform, the Fourier transform, which transforms spatial position x into spatial wave number k_x

$$\hat{f}(k_x) = \int_{-\infty}^{+\infty} f(x)\exp(ik_x x)dx \quad \text{and} \quad f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(k_x)\exp(-ik_x x)dk_x \tag{11.23}$$

In fact, the two are quite closely related, as can be seen by Fourier transforming Equation (11.22) with respect to $u \rightarrow k_u$:

$$\hat{d}(k_u, \theta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} m(u \cos \theta - s \sin \theta, u \sin \theta + s \cos \theta) ds \exp(ik_u u) du \tag{11.24}$$

We now transform the double integral from $ds du$ to $dx dy$, using the fact that the Jacobian determinant $|\det[\partial(x, y)/\partial(u, s)]|$ is unity (see Equation (11.21)):

$$\begin{aligned} \hat{d}(k_u, \theta) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} m(x, y)\exp(ik_u x \cos \theta + ik_u y \sin \theta) dx dy \\ &= \hat{m}(k_x = k_u \cos \theta, k_y = k_u \sin \theta) \end{aligned} \tag{11.25}$$

This result, called the *Fourier slice theorem*, provides a method of inverting the Radon transform. The Fourier-transformed quantity $\hat{d}(k_u, \theta)$ is simply the

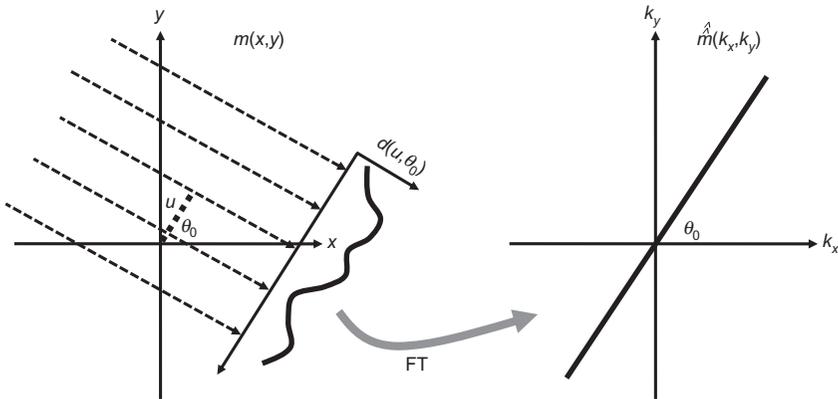


FIGURE 11.3 (Left) The function $m(x,y)$ is integrated along a set of parallel lines (dashed) in a Radon transform to form the function $d(u,\theta_0)$. This function is called the *projection* of $m(x,y)$ at the angle θ_0 . (Right) The Fourier slice theorem states that the Fourier transform (FT) of the projection is equal to the Fourier-transformed image evaluated along a line (bold) of angle θ_0 in the (k_x,k_y) plane.

Fourier-transformed image $\hat{m}(k_x, k_y)$ evaluated along radial lines in the (k_x,k_y) plane (Figure 11.3). If the Radon transform is known for all values of (u,θ) , then the Fourier-transformed image is known for all (k_x,k_y) , and since the Fourier transform can be inverted uniquely, the image itself is known for all (x,y) .

Since the Fourier transform and its inverse are unique, the Radon transform can be uniquely inverted if it is known for all possible (u,θ) . Further, the Fourier slice theorem can be used to invert the Radon transform in practice by using discrete Fourier transforms in place of integral Fourier transforms. However, u must be sampled sufficiently evenly that the $u \rightarrow k_u$ transform can be performed and θ must be sampled sufficiently finely that $\hat{m}(k_x, k_y)$ can be sensibly interpolated onto a rectangular grid of (k_x,k_y) to allow the $k_x \rightarrow x$ and $k_y \rightarrow y$ transforms to be performed (see the discussion in *MatLab* script `gda11_01`). An example of the use of the Fourier slice theorem to invert tomography data is shown in Figure 11.4.

11.7 CORRESPONDENCE BETWEEN MATRICES AND LINEAR OPERATORS

The continuous function $m(x)$ is the continuous analog to the vector \mathbf{m} . What is the continuous analog to \mathbf{Lm} , where \mathbf{L} is a matrix? Let us call it $\mathcal{L}m$, that is, \mathcal{L} operating on the function $m(x)$. In the discrete case, \mathbf{Lm} is another vector, so by analogy $\mathcal{L}m$ is another function. Just as \mathbf{L} is linear in the sense that $\mathbf{L}(\mathbf{m}^A + \mathbf{m}^B) = \mathbf{Lm}^A + \mathbf{Lm}^B$, we would like to choose \mathcal{L} so that it is linear in the sense that $\mathcal{L}(m^A + m^B) = \mathcal{L}m^A + \mathcal{L}m^B$. A hint to the identity of \mathcal{L} can be drawn from the fact that a matrix can be used to approximate derivatives and integrals. Consider, for example,

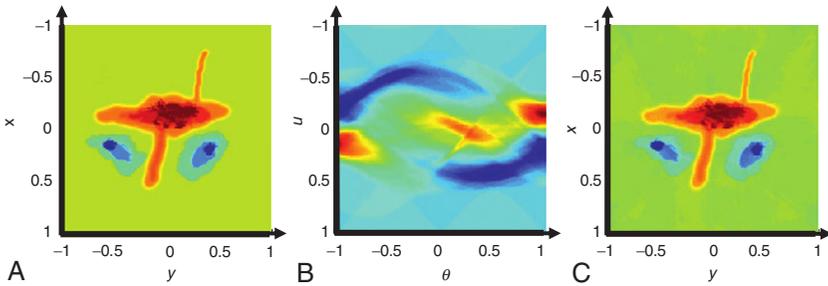


FIGURE 11.4 (A) A test image $m^{\text{true}}(x,y)$ of 256×256 discrete values, or pixels. This synthetic image depicts a hypothetical magma chamber beneath a volcano. (B) The Radon transform $d(u,\theta)$ of the image in (A), also evaluated at 256×256 points. (C) The image $m^{\text{est}}(x,y)$ reconstructed from its Radon transform by direct application of the Fourier slice theorem. Small errors in the reconstruction arise from the interpolation of the Fourier-transformed image onto a rectangular grid. *MatLab* script gda11_02.

$$\mathbf{L}^A = \frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{L}^B = \Delta x \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ & & \ddots & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \tag{11.26}$$

Here, $\mathbf{L}^A \mathbf{m}$ is the finite difference approximation to the derivative dm/dx and $\mathbf{L}^B \mathbf{m}$ is the Riemann sum approximation to the indefinite integral

$$\int_0^x m(x') dx' \tag{11.27}$$

Thus, \mathcal{L} can be any linear combination of integrals and derivatives (a *linear operator*, for short) (Lanczos, 1961). In the general multidimensional case, where $m(\mathbf{x})$ is a function of a N -dimensional position vector \mathbf{x} , \mathcal{L} is built up of partial derivatives and volume integrals. However, for simplicity, we restrict ourselves to the one-dimensional case here.

Now let us consider whether we can define an inverse operator \mathcal{L}^{-1} that is the continuous analog to the matrix inverse \mathbf{L}^{-1} . By analogy to $\mathbf{L}^{-1} \mathbf{Lm} = \mathbf{m}$, we want to choose \mathcal{L}^{-1} so that $\mathcal{L}^{-1} \mathcal{L}m(x) = m(x)$. Actually, the derivative matrix \mathbf{L}^A (Equation (11.26)) is problematical in this regard, for it is one row short of being square, and thus has no inverse. This corresponds to a function being determined only up to an integration constant when its derivative is known. This problem can be patched by adding to it a top row

$$\mathbf{L}^C = \frac{1}{\Delta x} \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{11.28}$$

that fixes the value of m_1 , that is, by specifying the integration constant. Thus, \mathcal{L} is not just a linear operator, but rather a linear operator plus one or more associated *boundary conditions*. It is easy to verify that $\mathbf{L}^C \mathbf{L}^B = \mathbf{I}$. The analogous continuous result that the derivative is the inverse operator of the derivative

$$m(x) = \frac{d}{dx} \int_0^x m(x') dx' \quad (11.29)$$

is well known and is called the *fundamental theorem of calculus*. Note that a linear differential equation can be written $\mathcal{L}m(x) = f(x)$. Its solution in terms of the inverse operator is $m(x) = \mathcal{L}^{-1}f(x)$. But its solution can also be written as a Green function integral

$$m(x) = \int_{-\infty}^{+\infty} F(x, \xi) f(\xi) d\xi = \mathcal{L}^{-1}f(x) \quad (11.30)$$

where $F(x, \xi)$ solves $\mathcal{L}F(x, \xi) = \delta(x - \xi)$. Hence, the inverse operator to a differential operator is the Green function integral.

Another important quantity ubiquitous in the formulas of inverse theory is the dot product between two vectors; written here generically as $s = \mathbf{a}^T \mathbf{b} = \sum_i a_i b_i$ where s is a scalar. The continuous analog is the integral

$$s = \int a(\mathbf{x}) b(\mathbf{x}) d^N x = (a, b) \quad (11.31)$$

where s is a scalar and $d^N x$ is the volume element, and the integration is over the whole space of \mathbf{x} . This integral is called the *inner product* of the functions $a(x)$ and $b(x)$ and is abbreviated $s = (a, b)$. As before, we restrict the discussion to the one-dimensional case:

$$s = \int_{-\infty}^{+\infty} a(x) b(x) dx = (a, b) \quad (11.32)$$

Many of the dot products that we encountered earlier in this book contained matrices, for example, $[\mathbf{Aa}]^T \mathbf{b}$ where \mathbf{A} is a matrix. The continuous analog is an inner product containing a linear operator, that is $(\mathcal{L}a, b)$.

An extremely important property of the dot product $[\mathbf{Aa}]^T \mathbf{b}$ is that it can also be written as $\mathbf{a}^T [\mathbf{Bb}]$ with $\mathbf{B} = \mathbf{A}^T$. We can propose the analogous relationship for linear operators:

$$(\mathcal{L}^A a, b) = (a, \mathcal{L}^B b) \quad (11.33)$$

The question then is what is the relationship between the two linear operators \mathcal{L}^A and \mathcal{L}^B , or put another way, what is the continuous analog of the transpose of matrix? As before, we still start off by merely giving the answer a name, the *adjoint*, and a symbol, \mathcal{L}^\dagger

$$(\mathcal{L}a, b) = (a, \mathcal{L}^\dagger b) \quad (11.34)$$

Several approaches are available for determining the \mathcal{L}^\dagger corresponding to a particular \mathcal{L} . The most straightforward is to start with the definition of the inner

product. For instance, if $\mathcal{L} = c(x)$, where $c(x)$ is an ordinary function, then $\mathcal{L}^\dagger = c(x)$, too, since

$$\int_{-\infty}^{+\infty} (ca)bdx = \int_{-\infty}^{+\infty} a(cb)dx \tag{11.35}$$

When $\mathcal{L} = d/dx$, we can use integration by parts to find \mathcal{L}^\dagger :

$$\int_{-\infty}^{+\infty} \frac{da}{dx} bdx = ab|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} a \frac{db}{dx} dx \tag{11.36}$$

So $\mathcal{L}^\dagger = -d/dx$, as long as the functions approach zero as x approaches $\pm\infty$. This same procedure, applied twice, can be used to show that $\mathcal{L} = d^2/dx^2$ is *self-adjoint*, that is, it is its own adjoint. As another example, we derive the adjoint of the indefinite integral $\int_{-\infty}^x d\xi$ by first writing it as

$$\int_{-\infty}^x a(\xi)d\xi = \int_{-\infty}^{+\infty} H(x - \xi)a(\xi)d\xi \tag{11.37}$$

where $H(x - \xi)$ is the Heaviside step function, which is unity if $x > \xi$ and zero if $x < \xi$. Then

$$\begin{aligned} (\mathcal{L}a, b) &= \int_{-\infty}^{+\infty} \left\{ \int_{-\infty}^x a(\xi)d\xi \right\} b(x)dx \\ &= \int_{-\infty}^{+\infty} \left\{ \int_{-\infty}^{+\infty} H(x - \xi)a(\xi)d\xi \right\} b(x)dx \\ &= \int_{-\infty}^{+\infty} a(\xi) \left\{ \int_{-\infty}^{+\infty} H(x - \xi)b(x)dx \right\} d\xi \\ &= \int_{-\infty}^{+\infty} a(\xi) \left\{ \int_{\xi}^{+\infty} b(x)dx \right\} d\xi = (a, \mathcal{L}^\dagger b) \end{aligned} \tag{11.38}$$

Hence, the adjoint of $\int_{-\infty}^x d\xi$ is $\int_x^{+\infty} d\xi$.

Another technique for computing an adjoint is to approximate the operator \mathcal{L} as a matrix, transpose the matrix, and then to “read” the adjoint operator back by examining the matrix. In the cases of the integral and first derivative, the transposes are

$$\mathbf{L}^{\text{AT}} = \Delta x \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 1 & 1 & 1 & \cdots & 1 \\ & & \ddots & & & \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \text{ and } \mathbf{L}^{\text{CT}} = \frac{1}{\Delta x} \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 \\ & & \ddots & & & \\ 0 & 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.39}$$

\mathbf{L}^{AT} represents the integral from progressively larger values of x to infinity and is equivalent to $\int_x^{+\infty} d\xi$. All but the last row of the \mathbf{L}^{CT} represents $-d/dx$, and the

last row is the boundary condition (which has moved from the top of \mathbf{L}^C to the bottom of \mathbf{L}^{CT}). Hence, these results agree with those previously derived.

Adjoint operators have many of the properties of matrix transposes, including

$$\begin{aligned} (\mathcal{L}^\dagger)^\dagger &= \mathcal{L} \quad \text{and} \quad (\overline{\mathcal{L}^{-1}})^\dagger = (\mathcal{L}^\dagger)^{-1} \\ (\mathcal{L}^A + \mathcal{L}^B)^\dagger &= (\mathcal{L}^B)^\dagger + (\mathcal{L}^A)^\dagger \quad \text{and} \quad (\mathcal{L}^A \mathcal{L}^B)^\dagger = (\mathcal{L}^B)^\dagger (\mathcal{L}^A)^\dagger \end{aligned} \quad (11.40)$$

11.8 THE FRÉCHET DERIVATIVE

Previously, we wrote the relationship between the model $m(x)$ and the data d_i as

$$d_i = \int G_i(x)m(x)dx = (G_i, m) \quad (11.41)$$

But now, we redefine it in terms of perturbations around some reference model $m^{(0)}(x)$. We define $m(x) = m^{(0)}(x) + \delta m(x)$ where $m^{(0)}(x)$ is a reference function and $\delta m(x)$ is a perturbation. If we write $d_i = d_i^{(0)} + \delta d_i$ where $d_i^{(0)} = (G_i, m^{(0)})$ is the data predicted by the reference model, then

$$\delta d_i = \int G_i(x)\delta m(x)dx = (G_i, \delta m) \quad (11.42)$$

This equation says that a perturbation $\delta m(x)$ in the model causes a perturbation δd_i in the data. This formulation is especially useful in linearized problems, since then the data kernel can be approximate, that is, giving results valid only when $\delta m(x)$ is small.

Equation (11.42) is reminiscent of the standard derivative formula that we have used in linearized discrete problems:

$$\Delta d_i = \sum_{j=1}^M G_{ij}^{(0)} \Delta m_j \quad \text{with} \quad G_{ij}^{(0)} = \left. \frac{\partial d_i}{\partial m_j} \right|_{\mathbf{m}^{(0)}} \quad (11.43)$$

The only difference is that $\Delta \mathbf{m}$ is a vector, whereas $\delta m(x)$ is a continuous function. Thus, we can understand $G_i(x)$ in Equation (11.42) an analog to a derivative. We might use the notation

$$G_i(x) = \left. \frac{\delta d_i}{\delta m} \right|_{\mathbf{m}^{(0)}} \quad (11.44)$$

in which case it is called the *Fréchet derivative* of the datum d_i with respect to the model $m(x)$.

11.9 THE FRÉCHET DERIVATIVE OF ERROR

Fréchet derivatives of quantities other than the data are possible. One that is of particular usefulness is the Fréchet derivative of the error E with respect to the model, where the data $d(x)$ is taken to be a continuous variable.

$$E = (d^{\text{obs}} - d, d^{\text{obs}} - d) \quad \text{and} \quad \delta E = E - E^{(0)} = \left(\frac{\delta E}{\delta m} \Big|_{\mathbf{m}^{(0)}}, \delta m \right) \quad (11.45)$$

Here the error is the continuous analog to the L_2 norm discrete error $E = (\mathbf{d}^{\text{obs}} - \mathbf{d})^T (\mathbf{d}^{\text{obs}} - \mathbf{d})$. Suppose now that the data are related to the model via a linear operator, $d = \mathcal{L}m$. We compute the perturbation δE due to the perturbation δm as

$$\begin{aligned} \delta E &= E - E^{(0)} = (d^{\text{obs}} - d, d^{\text{obs}} - d) - (d^{\text{obs}} - d^{(0)}, d^{\text{obs}} - d^{(0)}) \\ &= -2(d, d^{\text{obs}}) + (d, d) + 2(d^{(0)}, d^{\text{obs}}) - (d^{(0)}, d^{(0)}) \\ &= -2(d^{\text{obs}} - d^{(0)}, d - d^{(0)}) + (d - d^{(0)}, d - d^{(0)}) \\ &= -2(d^{\text{obs}} - d^{(0)}, \delta d) + (\delta d, \delta d) \approx -2(d^{\text{obs}} - d^{(0)}, \delta d) \\ &= -2(d^{\text{obs}} - d^{(0)}, \mathcal{L}\delta m) \end{aligned} \quad (11.46)$$

Note that we have ignored a term involving the second-order quantity $(\delta d)^2$. Using the adjoint operator \mathcal{L}^\dagger , we find

$$\delta E = (-2\mathcal{L}^\dagger(d^{\text{obs}} - d^{(0)}), \delta m) \quad (11.47)$$

which implies that the Fréchet derivative of the error E is

$$\frac{\delta E}{\delta m} \Big|_{\mathbf{m}^{(0)}} = -2\mathcal{L}^\dagger(d^{\text{obs}} - d^{(0)}) \quad (11.48)$$

This result can be useful when solving the inverse problem using a gradient method (see Section 9.8). As an example, consider the problem $d = \mathcal{L}m$ where

$$\mathcal{L}m(x) = a \frac{d}{dx} m(x) + b \int_{-\infty}^x m(x') dx' \quad (11.49)$$

where a and b are constants. Using the adjoint relationships derived in Section 11.7, we find

$$\mathcal{L}^\dagger d(x) = -a \frac{d}{dx} d(x) + b \int_x^{-\infty} d(x') dx' \quad (11.50)$$

and hence the Fréchet derivative of the error E is

$$\frac{\delta E}{\delta m} \Big|_{\mathbf{m}^{(0)}} = 2a \frac{d}{dx} [d^{\text{obs}}(x) - d^{(0)}(x)] - 2b \int_x^{-\infty} [d^{\text{obs}}(x') - d^{(0)}(x')] dx' \quad (11.51)$$

In order to use this result in a numerical scheme, one must discretize it; say by using voxels of width Δx and amplitude \mathbf{m} for the model and \mathbf{d} for the data, both of length $M=N$. Equation (11.51) then yields the gradient $\partial E/m_i$, which can be used to minimize E via the gradient method (Figure 11.5).

11.10 BACKPROJECTION

The formula for the Fréchet derivative of the error E (Equation (11.48)) is the continuous analog of the discrete gradient $\nabla E = -2\mathbf{G}^T(\mathbf{d}^{\text{obs}} - \mathbf{d}^{\text{pre}}) = -2\mathbf{G}^T(\mathbf{d}^{\text{obs}} - \mathbf{G}\mathbf{m})$. In Section 3.4, the discrete gradient was set to zero, leading

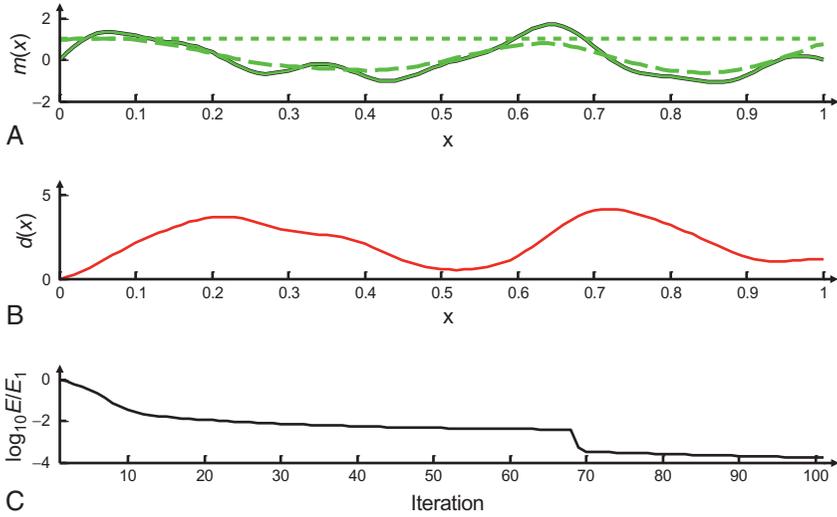


FIGURE 11.5 Example of the solution of a continuous inverse problem using a gradient method to minimize the error E , where an adjoint method is used to compute ∇E . (A) A test function, $m^{\text{true}}(x)$ (green), trial function (dotted green) reconstructed function after 40 iterations (dashed green) and final reconstructed function after 15,890 iterations (green). (B) The data, $d(t)$, satisfies $d(t) = \mathcal{L}m(t)$, where \mathcal{L} is the linear operator discussed in the text. (C) Error E as a function of iteration number, for the first 100 iterations. *MatLab* script `gda11_03`.

to the least-squares equation for the model parameters, $\mathbf{G}^T \mathbf{G} \mathbf{m} = \mathbf{G}^T \mathbf{d}^{\text{obs}}$. A similar result is achieved in the continuous case:

$$\begin{aligned} \left. \frac{\delta E}{\delta m} \right|_{\mathbf{m}^{(0)}} = 0 &= -2\mathcal{L}^\dagger (d^{\text{obs}} - d) = -2\mathcal{L}^\dagger (d^{\text{obs}} - \mathcal{L}m) \quad \text{or} \\ \mathcal{L}^\dagger \mathcal{L}m &= \mathcal{L}^\dagger d^{\text{obs}} \end{aligned} \quad (11.52)$$

Now suppose that \mathcal{J} represents the identity operator, that is, satisfying $m = \mathcal{J}m$ (in one dimension, this operator is $m(x) = \int_{-\infty}^{\infty} \delta(x - x')m(x')dx'$). Then we can write

$$\begin{aligned} (\mathcal{L}^\dagger \mathcal{L} + \mathcal{J} - \mathcal{J})m &= \mathcal{L}^\dagger d^{\text{obs}} \quad \text{or} \\ m = \mathcal{J}m &= \mathcal{L}^\dagger d^{\text{obs}} - (\mathcal{L}^\dagger \mathcal{L} - \mathcal{J})m \end{aligned} \quad (11.53)$$

In the special case that $\mathcal{L}^\dagger \mathcal{L} = \mathcal{J}$ (that is, $\mathcal{L}^\dagger = \mathcal{L}^{-1}$), the solution is very simple, $m = \mathcal{J}m = \mathcal{L}^\dagger d^{\text{obs}}$. However, even in other cases the equation is still useful, since it can be viewed as a recursion relating an old estimate of the model parameters $m^{(i)}$ to a new one, $m^{(i+1)}$

$$m^{(i+1)} = \mathcal{L}^\dagger d^{\text{obs}} - (\mathcal{L}^\dagger \mathcal{L} - \mathcal{J})m^{(i)} \quad (11.54)$$

If the recursion is started with $m^{(0)} = 0$, the new estimate is

$$m^{(1)} = \mathcal{L}^\dagger d^{\text{obs}} \quad (11.55)$$

As an example, suppose that \mathcal{L} is the indefinite integral, so that

$$d^{\text{obs}}(x) = \mathcal{L} m(x) = \int_{-\infty}^x m(x') dx' \quad (11.56a)$$

and

$$m^{(1)}(x) = \mathcal{L}^\dagger d^{\text{obs}}(x) = \int_x^\infty d^{\text{obs}}(x') dx' \quad (11.56b)$$

Equation (11.56b) may seem crazy, since an indefinite integral is inverted by taking a derivative, not another integral. Yet this result, while approximate, is nevertheless quite good in some cases, at least up to an overall multiplicative factor (Figure 11.6).

Equation (11.56a) might be considered an ultrasimplified one-dimensional tomography problem, relating acoustic slowness $m(x)$ to traveltime $d^{\text{obs}}(x)$. Note that the ray associated with traveltime $d^{\text{obs}}(x)$ starts at $-\infty$ and ends at x ; that is, the traveltime at a point x depends only upon the slowness to the left of x . The formula for $m^{(1)}$ has a simple interpretation: the slowness at x is estimated by summing up all the traveltimes for rays ending at points $x' > x$; that is, summing up traveltimes only for those rays that sample the slowness at x . This process, which carries over to multidimensional tomography, is called *backprojection*. The inclusion, in a model parameter's own estimate, of just those data that are affected by it, is intuitively appealing.

Note that the accuracy of the backprojection depends upon the degree to which $\mathcal{L}^\dagger \mathcal{L} - \mathcal{I} = 0$ in Equation (11.54). Some insight into this issue can be

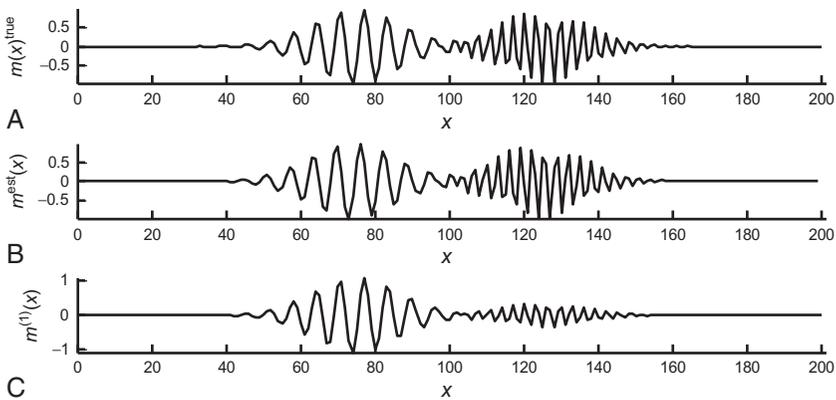


FIGURE 11.6 (A) True one-dimensional model $m^{\text{true}}(x)$. The data satisfy $d^{\text{obs}} = \mathcal{L} m^{\text{true}}$, where \mathcal{L} is the indefinite integral. (B) Estimated model, using $m^{\text{est}} = \mathcal{L}^{-1} d^{\text{obs}}$ where \mathcal{L}^{-1} is the first derivative. Note that $m^{\text{est}} = m^{\text{true}}$. (C) Backprojected model $m^{(1)} = \mathcal{L}^\dagger d^{\text{obs}}$, where \mathcal{L}^\dagger is the adjoint of \mathcal{L} . Note that, up to an overall multiplicative factor, $m^{(1)} \approx m^{\text{true}}$. *MatLab* script `gda11_04`.

gained by examining the singular value decomposition of a discrete data kernel $\mathbf{G} = \mathbf{U}_p \mathbf{\Lambda}_p \mathbf{V}_p^T$, which has transpose $\mathbf{G}^T = \mathbf{V}_p \mathbf{\Lambda}_p \mathbf{U}_p^T$ and generalized inverse $\mathbf{G}^{-g} = \mathbf{V}_p \mathbf{\Lambda}_p^{-1} \mathbf{U}_p^T$. Clearly, the accuracy of the approximation $\mathbf{G}^{-g} = \mathbf{G}^T$ will depend upon the degree to which $\mathbf{\Lambda}_p = \mathbf{\Lambda}_p^{-1}$, that is, to the degree to which the singular values have unit amplitude. The correspondence might be improved by scaling the rows of \mathbf{G} and corresponding elements of \mathbf{d}^{obs} by carefully chosen constants c_i : $G_{ij} \rightarrow c_i G_{ij}$ and $d_i \rightarrow c_i d_i$ so that the singular values are of order unity. This analysis suggests a similar scaling of the continuous problem, $\mathcal{L} \rightarrow c(x)\mathcal{L}$ and $d^{\text{obs}}(x) \rightarrow c(x)d^{\text{obs}}(x)$, where $c(x)$ is some function. In tomography, a commonly used scaling is $c(x) = 1/L(x)$, where $L(x)$ is the length of ray x . The transformed data d^{obs}/L represents the average slowness along the ray. The backprojection process sums up the average slowness of all rays that interact with the model parameter at point x . This is a bit counterintuitive; one might expect that averaging the averages, as contrasted to summing the averages, would be more appropriate. Remarkably, the use of the summation introduces only long-wavelength errors into the image. An example of two-dimensional back projection is shown in Figure 11.7.

11.11 FRÉCHET DERIVATIVES INVOLVING A DIFFERENTIAL EQUATION

Equation (11.42) links model parameters directly to the data. Some inverse problems are better analyzed when the link is indirect, through a field, $u(x)$. The model parameters are linked to the field, and the field to the data. Consider, for example, a problem in ocean circulation, where the model parameters $m(x)$ represent the force of the wind, the field $u(x)$ represents the velocity of the ocean water and the data d_i represent the volume of water transported from one ocean to another. Wind forcing is linked to water velocity through a fluid-mechanical differential equation, and water transport is related to water velocity through an

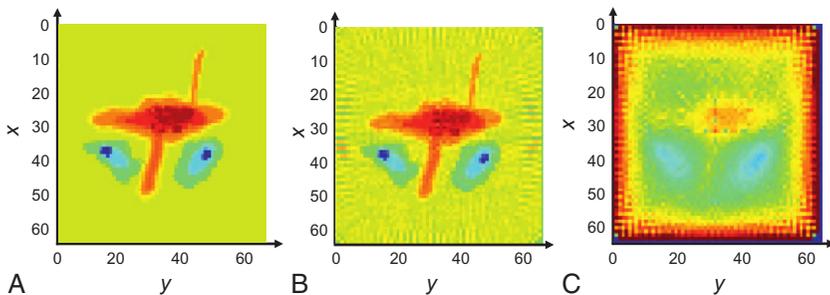


FIGURE 11.7 Example of backprojection. (A) True two-dimensional model, for which travel-times associated with a dense and well-distributed set of rays are measured. (B) Estimated model, using damped least squares. (C) Estimated model, using backprojection. *MatLab* script gda11_05.

integral of the velocity across the ocean-ocean boundary. Such a relationship has the form

$$\mathcal{L}u(x) = m(x) \quad \text{and} \quad \mathcal{L}\delta u(x) = \delta m(x) \quad (11.57)$$

$$d_i = (h_i(x), u(x)) \quad \text{and} \quad \delta d_i = (h_i(x), \delta u(x)) \quad (11.58)$$

Here, Equation (11.57) is a differential equation with known boundary conditions and Equation (11.58) is an inner product involving a known function $h_i(t)$. Symbolically, we can write the solution of the differential equation as

$$\begin{aligned} u(x) &= \int F(x, \xi) m(\xi) d\xi = \mathcal{L}^{-1} m(x) \\ \delta u(x) &= \int F(x, \xi) \delta m(\xi) d\xi = \mathcal{L}^{-1} \delta m(x) \end{aligned} \quad (11.59)$$

where $F(x, \xi)$ is the Green function. Note that \mathcal{L}^{-1} is a linear integral operator, whereas \mathcal{L} is a linear differential operator. We now combine Equations (11.57) and (11.58):

$$\delta d_i = (h_i, \delta u) = (h_i, \mathcal{L}^{-1} \delta m) = ((\mathcal{L}^{-1})^\dagger h_i, \delta m) = ((\mathcal{L}^\dagger)^{-1} h_i, \delta m) \quad (11.60)$$

Comparing to Equation (11.42), we find

$$G_i(x) = (\mathcal{L}^\dagger)^{-1} h_i(x) \quad \text{or} \quad \mathcal{L}^\dagger G_i(x) = h_i(x) \quad (11.61)$$

Thus, the scalar field satisfies the equation $\mathcal{L}u(x) = m(x)$ and the data kernel satisfies the adjoint equation $\mathcal{L}^\dagger G_i(x) = h_i(x)$. In most applications, the scalar field $u(t)$ must be computed by numerical solution of its differential equation. Thus, the computational machinery is typically already in place to solve the adjoint differential equation and thus to construct the data kernel.

An important special case is when the data is the field $u(x)$ itself, that is, $d_i = u(x_i)$. This choice implies that the weighting function in Equation (11.58) is a Dirac delta function, $h_i(x) = \delta(x - x_i)$ and that the data kernel is the Green function, say $Q(x, x_i)$, to the adjoint equation. Then

$$d_i = (Q(x, x_i), m(x)) \quad \text{with} \quad \mathcal{L}^\dagger Q(x, x_i) = \delta(x - x_i) \quad (11.62)$$

As an example, we consider an object that is being heated by a flame. We will use time t instead of position x in this example; the object is presumed to be of a spatially uniform temperature $u(t)$ that varies with time, t . Suppose that temperature obeys the simple Newtonian heat flow equation

$$\mathcal{L}u(t) = \left\{ \frac{d}{dt} + c \right\} u(t) = m(t) \quad (11.63)$$

where the function $m(t)$ represents the heating and c is a thermal constant. This equation implies that in the absence of heating, the temperature of the object decays away toward zero at a rate determined by c . We assume that the heating is restricted to some finite time interval, so that temperature satisfies the boundary condition $u(t \rightarrow -\infty) = 0$.

The solution to the heat flow equation can be constructed from its Green function $F(t, \tau)$, which represents the temperature at time t due to an impulse of heat at time τ . It solves the equation

$$\left\{ \frac{d}{dt} + c \right\} F(t, \tau) = \delta(t - \tau) \quad (11.64)$$

The solution to this equation can be shown to be

$$F(t, \tau) = H(t - \tau) \exp\{-c(t - \tau)\} \quad (11.65)$$

Here the function $H(t - \tau)$ is the Heaviside step function, which is zero when $t < \tau$ and unity when $t > \tau$. This result can be verified by first noting that it satisfies the boundary condition and then by checking, through direct differentiation, that it solves the differential equation. The temperature of the object is zero before the heat pulse is applied, jumps to a maximum at the moment of application, and then exponentially decays back to zero at a rate determined by the constant, c (Figure 11.8A). A hypothetical heating function $m(x)$ and resulting temperature $u(t)$ are shown in Figure 11.8B and C, respectively.

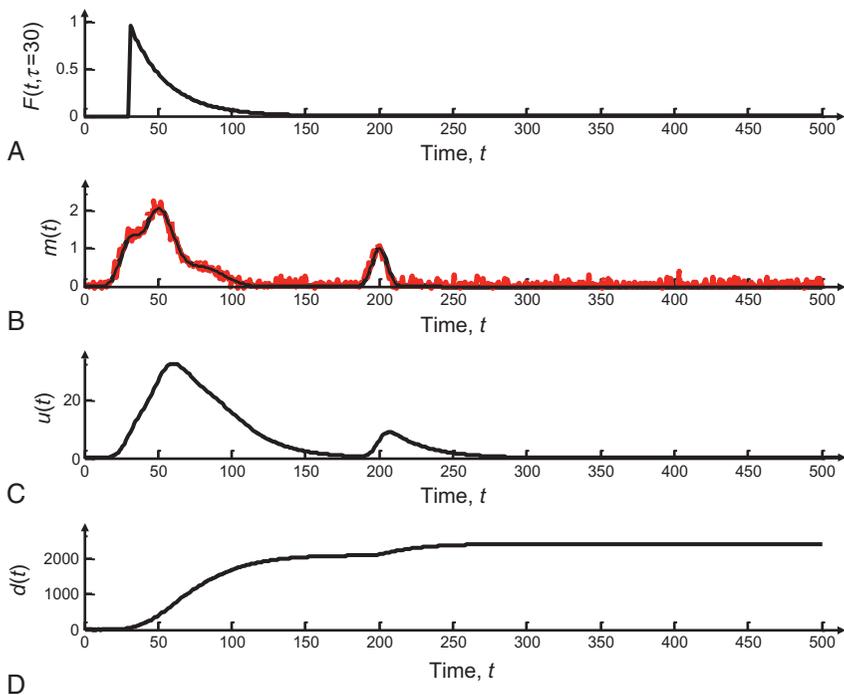


FIGURE 11.8 Example of the solution of a continuous inverse problem involving a differential equation. (A) Green function $F(t, \tau)$ for $\tau = 30$. (B) True (black) and estimated (red) heat production function $m(t)$. (C) Temperature $u(t)$, which solves $\mathcal{L}u = m$. (D) Observed data $d(t)$, which is proportional to the integral of $u(t)$. *MatLab* script gda11_06.

We now need to couple the temperature function $u(t)$ to observations, in order to build a complete forward problem. Suppose that a chemical reaction occurs within the object, at a rate that is proportional to temperature. We observe the amount of chemical product $d_i = P(t_i)$, which is given by the integral

$$d_i = P(t_i) = b \int_0^{t_i} u(t) dt = b \int_0^\infty H(t_i - t) u(t) dt = (bH(t_i - t), u(t)) \quad (11.66)$$

where b is the proportionality factor. Thus, $h_i(t) = bH(t_i - t)$, where H is the Heaviside step function. The adjoint equation is

$$\mathcal{L}^\dagger u(t) = \left\{ -\frac{d}{dt} + c \right\} g_i(t) = h_i(t) \quad (11.67)$$

since, as noted in Section 11.7, the adjoint of d/dt is $-d/dt$ and the adjunct of a constant c is itself. The Green function of the adjoint equation can be shown to be

$$Q(t, \tau) = H(\tau - t) \exp\{+c(t - \tau)\} \quad (11.68)$$

Note that $Q(t, \tau)$ is just a *time-reversed* version of $F(t, \tau)$, a relationship that arises because the two differential equations differ only by the sign of t . The data kernel $G_i(t)$ is given by the Green function integral

$$\begin{aligned} G_i(t) &= \int_0^\infty Q(t, \tau) h_i(\tau) d\tau \\ &= \int_0^\infty H(\tau - t) \exp\{c(t - \tau)\} bH(t_i - \tau) d\tau \\ &= b \int_0^\infty H(\tau - t) H(t_i - \tau) \exp\{c(t - \tau)\} d\tau \\ &= b \int_t^{t_i} \exp\{-c(\tau - t)\} d\tau \end{aligned} \quad (11.69)$$

The integral is nonzero only in the case where $t_i > t$:

$$G_i(t) = \begin{cases} 0 & t_i \leq t \\ -\frac{b}{c} [\exp\{-c(t_i - t)\} - 1] & t_i > t \end{cases} \quad (11.70)$$

The data kernel $G_i(t)$ (Figure 11.9) quantifies the effect of heat applied at time t on an observation made at time t_i . It being zero for times $t > t_i$ is a manifestation of causality; heat applied in the future of a given observation cannot affect it.

The problem that we have been discussing is completely linear; neither the differential equation relating $m(t)$ to $u(t)$ nor the equation relating $u(t)$ to d_i contains any approximations. The data kernel is therefore accurate for perturbations

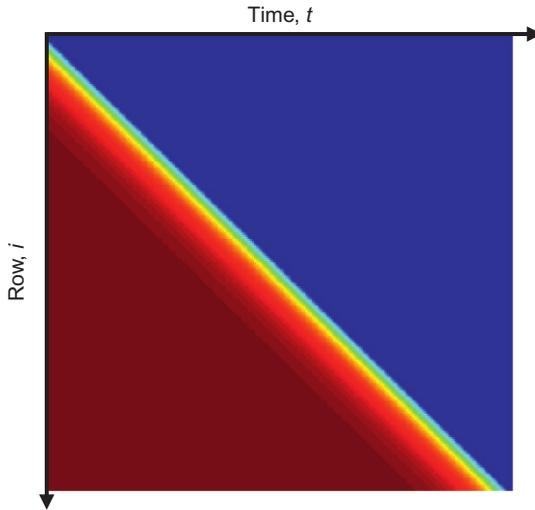


FIGURE 11.9 Data kernel $G_i(t)$ for the continuous inverse problem involving a differential equation. See text for further discussion. *MatLab* script gda11_06.

of any size and hence holds for $m(t)$ and d_i , as well as for the perturbations $\delta m(t)$ and δd_i :

$$d_i = (G_i(t), m(t)) \quad (11.71)$$

We solve the inverse problem—determining the heating through observations of the chemical product—by first discretizing all quantities with a time increment Δt

$$m(t) \rightarrow m_i = m(i\Delta t) \quad \text{and} \quad G_i(t) \rightarrow G_{ij} = \Delta t G_i(j\Delta t) \quad (11.72)$$

so that

$$d_i = (G_i(t), m(t)) \rightarrow \mathbf{d} = \mathbf{Gm} \quad (11.73)$$

Note that a factor of Δt has been included in the definition of \mathbf{G} to account for the one that arises when the inner product is approximated by its Riemann sum. We assume that data is measured at all times, and then solve [Equation \(11.73\)](#) by damped least squares ([Figure 11.8B](#)). Some noise amplification occurs, owing to the very smooth data kernel. Rapid fluctuations in heating $m(t)$ have very little effect on the temperature $u(t)$, owing to the diffusive nature of heat, and thus are poorly constrained by the data. Adding smoothness constraints would lead to a better solution.

In the example above, we were concerned with the effect of a perturbation in *forcing* (meaning the function on the right-hand side of the differential equation) on the data. Another common problem is to understand the effect of a

perturbation of a *parameter* in the differential operator, \mathcal{L} , on the data. The parameter $c(t)$ in the differential equation

$$\left\{ \frac{d}{dt} + c(t) \right\} u(t) = f(t) \quad (11.74)$$

would be an example, where the forcing $f(t)$ is now considered a known function. We now write $c(t) = c^{(0)} + \delta c(t)$ and $u(t) = u^{(0)}(t) + \delta u(t)$, where $u^{(0)}(t)$ solves the unperturbed equation

$$\left\{ \frac{d}{dt} + c^{(0)} \right\} u^{(0)}(t) = f(t) \quad (11.75)$$

The perturbed equation then becomes

$$\begin{aligned} \left\{ \frac{d}{dt} + c^0 + \delta c(t) \right\} \{ u^{(0)}(t) + \delta u(t) \} &= f(t) \\ \left\{ \frac{d}{dt} + c^0 \right\} u^{(0)}(t) + \left\{ \frac{d}{dt} + c^0 \right\} \delta u(t) + \delta c(t) u^{(0)}(t) + \delta c(t) \delta u(t) &= f(t) \end{aligned} \quad (11.76)$$

After subtracting out the unperturbed equation, ignoring the second-order term and rearranging, we find

$$\left\{ \frac{d}{dt} + c^0 \right\} \delta u(t) = -\delta c(t) u^{(0)}(t) \quad (11.77)$$

Thus, the perturbation δc in the parameter c is equivalent to a perturbation in an *unknown* forcing:

$$\delta m(t) = -\delta c(t) u^{(0)}(t) \quad (11.78)$$

We can now use the formalism that we developed above to determine the resulting perturbation in the data.

Adjoint techniques have had extensive application in seismology (Dahlen et al., 2000; Tromp et al., 2005) and in atmospheric and ocean science, where they are associated with the term *data assimilation* (Hall and Cacuci, 1983; Moore et al., 2004).

11.12 PROBLEMS

11.1. This inverse problem is due to Robert Parker. Consider a radially stratified sphere of radius $R=1$ with unknown density $\rho(r)$. Its mass $M = 4\pi \int \rho(r) r^2 dr$ and its moment of inertia $I = \left(\frac{8\pi}{3}\right) \int \rho(r) r^4 dr$ are measured. What is the best averaging kernel that can be designed that is

centered about $\frac{1}{2}R$? Since all the integrals involve only rational functions and all the matrices are 2×2 , the problem can be solved analytically. Do as much of the problem as you can, analytically; do the parts that you cannot, numerically.

- 11.2.** Suppose that a function $m(x)$ is discretized in each of two ways: (A) by dividing it into M rectangles, each of height m_i and width Δx ; and (B) by representing it as a sum of M overlapping Gaussians, spaced at regular intervals Δx , and with coefficients

$$m(x) = \sum_{i=1}^M m_i' \frac{\Delta x}{(2\pi)\sigma} \exp\left\{-\frac{(x-i\Delta x)^2}{2\sigma^2}\right\}$$

Here, σ^2 is a prescribed variance. Discuss the advantages and disadvantages of each representation. Include the respective effect of damping in the two cases.

- 11.3.** How sensitive are the results of tomography to noise in the data (u, θ) ? Run experiments with the Fourier slice script. Try two different types of noise: (A) uncorrelated random noise drawn from a Gaussian distribution and (B) a few large outliers. Comment upon the results.
- 11.4.** Modify the example in Section 11.9 to use a truncated Fourier sine series representation of the model:

$$m^{(0)}(x) = \sum_{n=1}^K m_n^{(0)} \sin\left(\frac{n\pi x}{L}\right) \quad \text{and} \quad \delta m(x) = \sum_{n=1}^K \delta m_n^{(0)} \sin\left(\frac{n\pi x}{L}\right)$$

where m_n are scalar coefficients, $K=15$ and $0 < x < L$. You will need to use the chain rule

$$\left. \frac{\delta E}{\delta m_n} \right|_{\mathbf{m}^{(0)}} = \left(\left. \frac{\delta E}{\delta m} \right|_{\mathbf{m}^{(0)}}, \left. \frac{\delta m}{\delta m_n} \right|_{\mathbf{m}^{(0)}} \right)$$

- 11.5.** Suppose that the Green function $F(t, \tau)$ of a particular differential equation $\mathcal{L}^F F(t, \tau) = \delta(t - \tau)$ depends only upon time differences; that is, $F(t, \tau) = F(t - \tau)$. Denote the Green function integral as the linear operator $(\mathcal{L}^F)^{-1}$. (A) Use the formula for the inner product to derive $(\mathcal{L}^F)^{-1\dagger}$. Call the function within this operator F' . (B) Suppose the differential equation $\mathcal{L}^{Q\dagger} Q(t, \tau) = \delta(t - \tau)$ has Green function $Q(t, \tau)$. Denote the Green function integral as the linear operator $(\mathcal{L}^{Q\dagger})^{-1}$. By comparing $(\mathcal{L}^F)^{-1\dagger}$ and $(\mathcal{L}^{Q\dagger})^{-1}$, establish the relationship between F' and Q . Explain your results in words.

REFERENCES

- Backus, G.E., Gilbert, J.F., 1968. The resolving power of gross earth data, *Geophys. J. Roy. Astron. Soc.* 16, 169–205.
- Dahlen, F.A., Hung, S.-H., Nolet, G., 2000. Fréchet kernels for finite frequency traveltimes—I. Theory, *Geophys. J. Int.* 141, 157–174.

- Hall, M.C.G., Cacuci, D.G., 1983. Physical interpretation of the adjoint functions for sensitivity analysis of atmospheric models. *J. Atmos. Sci.* 40, 2537–2546.
- Lanczos, C., 1961. *Linear Differential Operators*. Van Nostrand-Reinhold, Princeton, NJ.
- Moore, A.M., Arango, H.G., Di Lorenzo, E., Cornuelle, B.D., Miller, A.J., Neilson, D.J., 2004. A comprehensive ocean prediction and analysis system based on the tangent linear and adjoint of a regional ocean model. *Ocean Modelling* 7, 227–258.
- Tromp, J., Tape, C., Liu, Q., 2005. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels, *Geophys. J. Int.* 160, 195–216.

Sample Inverse Problems

12.1 AN IMAGE ENHANCEMENT PROBLEM

Suppose that a camera moves slightly during the exposure of an image, so that the picture is blurred. Also suppose that the amount and direction of motion are known. Can the image be “unblurred?”

The optical sensor in the camera consists of rows and columns of light-sensitive elements that measure the total amount of light received during the exposure. For simplicity, we shall consider that the camera moves parallel to a row of pixels. The data d_i are a set of numbers that represent the amount of light recorded at each of N pixels. Because the scene’s brightness varies continuously, this is properly a problem in continuous inverse theory. We shall discretize it, however, by assuming that the scene can be adequately approximated by a row of small square elements, each with a constant brightness. These elements form M model parameters m_i . Since the camera’s motion is known, it is possible to calculate each scene element’s relative contribution to the light recorded at a given camera pixel. For instance, if the camera moves through three scene elements during the exposure, then each camera element records the average of three neighboring scene brightnesses

$$d_i = \frac{1}{3}[m_{i-1} + m_i + m_{i+1}] \quad (12.1)$$

Note that this is a linear equation and can be written in the form $\mathbf{Gm} = \mathbf{d}$, where

$$\mathbf{G} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ & & & \ddots & & & \\ 0 & \cdots & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad (12.2)$$

In general, there will be several more model parameters than data, so the problem will be underdetermined. In this case $M = N + 2$, so there will be at least two null vectors. In fact, the problem is purely underdetermined, and there are only two null vectors, which can be identified by inspection as

$$\begin{aligned} \mathbf{m}^{(1)\text{null}} &= [1 \ 0 \ -1 \ 1 \ 0 \ -1 \ \cdots \ 1 \ 0 \ -1]^T \\ \mathbf{m}^{(2)\text{null}} &= [0 \ 1 \ -1 \ 0 \ 1 \ -1 \ \cdots \ 0 \ 1 \ -1]^T \end{aligned} \quad (12.3)$$

These null vectors have rapidly fluctuating elements, which indicates that at best only the longer wavelength features can be recovered. To find a solution to the inverse problem, we must add *a priori* information. We shall use a simplicity constraint and find the scene of shortest length. If the data are normalized so that zero represents gray (with white negative and black positive), then this procedure in effect finds the scene of least contrast that fits the data. We therefore estimate the solution with the minimum length generalized inverse as

$$\mathbf{m}^{\text{est}} = \mathbf{G}^T [\mathbf{G}\mathbf{G}^T]^{-1} \mathbf{d}^{\text{obs}} \quad (12.4)$$

As an example, we shall solve the problem for the data kernel given above, for a blur width of 100 and with an image $M = 2000$ pixels wide. We first select a true scene (Figure 12.1A) and blur it by multiplication with the data kernel to produce synthetic data (Figure 12.1B). Note that the blurred image is much smoother than the true scene. We now try to invert back for the true scene by premultiplying by the generalized inverse. The result (Figure 12.1C) has not only correctly captured some of the sharp features in the original scene but also contains a short wavelength oscillation not present in the true scene. This error results from creating a reconstructed scene containing an incorrect combination of null vectors.

In *MatLab*, the inverse problem for each row of the image is solved separately, with Equation (12.5) evaluated in a loop over rows:

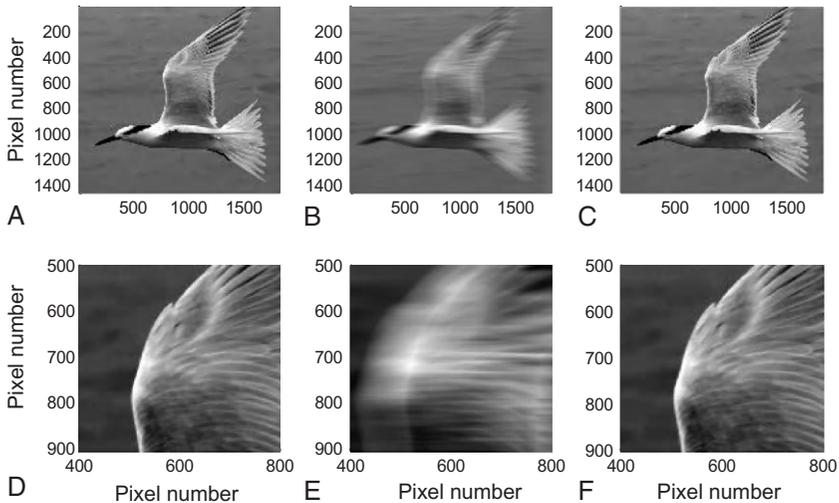


FIGURE 12.1 Example of removing blur from image. (A) True image. (B) Image blurred with 100 pixel-wide boxcar filter. (C) Estimated image, unblurred using the minimum length generalized inverse. (D–F) Enlargement of portion of images (A–C). *MatLab* script gda12_01.

```

epsilon=1.0e-6; % damping, just in case GGT is singular
GGT=G*G'+epsilon*speye(J,J);
for i=[1:I]
    dobsrow=dobs(i,:);
    mestrow=G'*(GGT\dobsrow);
    mest(i,:)=mestrow';
end

```

(*MatLab* script gda12_01)

The matrix \mathbf{G} is defined as sparse, and the matrix product $[\mathbf{G}\mathbf{G}^T]$ is calculated by simple matrix multiplication.

We note that the matrix \mathbf{G} is simple enough for the matrix product $[\mathbf{G}\mathbf{G}^T]$ to be computed analytically

$$\mathbf{G}\mathbf{G}^T = \frac{1}{9} \begin{bmatrix} 3 & 2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 2 & 3 & 2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 2 & 3 & 2 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix} \quad (12.5)$$

In this problem, which deals with moderate-sized matrices, the effort saved in using the analytic version over a numerical computation is negligible. In larger inverse problems, however, considerable saving can result from a careful analytical treatment of the structures of the various matrices.

Each row of the generalized inverse states how a particular model parameter is constructed from the data. We might expect that this blurred image problem would have a localized generalized inverse, meaning that each model parameter would be constructed mainly from a few neighboring data. By examining \mathbf{G}^{-g} , we see that this is not the case (Figure 12.2A). We also note that the process of

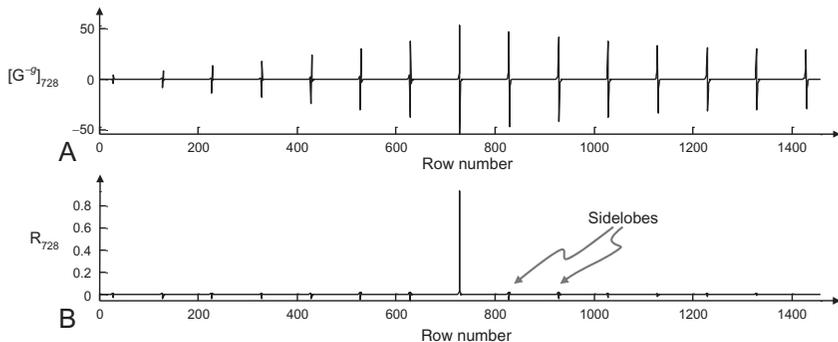


FIGURE 12.2 (A) Central row (row 728) of the generalized inverse of the image deblurring problem. (B) Central row (row 728) of the corresponding resolution matrix. The sidelobes are about 6% of the amplitude of the central peak. *MatLab* script gda12_01.

blurring is an integrating process; it sums neighboring data. The process of unblurring should be a sort of differentiating process, subtracting neighboring data. The generalized inverse is, in fact, just that.

The resolution matrix for this problem is seen to be quite “spiky” (Figure 12.2B); the diagonal elements are several orders of magnitude larger than the off-diagonal elements. On the other hand, the off-diagonal elements are all of uniform size, indicating that if the estimated model parameters are interpreted as localized averages, they are in fact not completely localized. It is interesting to note that the Backus-Gilbert inverse (not shown), which generally gives very localized resolution kernels, returns only the blurred image itself. The solution to this problem contains an unavoidable trade-off between the width of resolution and the presence of sidelobes.

12.2 DIGITAL FILTER DESIGN

Suppose that two signals $d(t)$ and $g(t)$ are known to be related by *convolution* with a filter $m(t)$:

$$d(t) = m(t) * g(t) = \int g(t - \tau)m(\tau) d\tau \quad (12.6)$$

where τ is a dummy integration variable. Can $m(t)$ be found if $g(t)$ and $d(t)$ are known?

Since the signals and filter are continuous functions, this is a problem in continuous inverse theory. We shall analyze it, however, by approximating the functions as *time series*. Each function will be represented by its value at a set of points spaced equally in time (with interval Δt). We shall assume that the signals are transient (have a definite beginning and end) so that $d(t)$ and $g(t)$ can be represented by time series of length N . Typically, the advantage of relating two signals by a filter is realized only when the filter length M is shorter than either signal, so $M < N$ is presumed. The convolution integral can then be approximated by the sum

$$d_i = \Delta t \sum_{j=1}^M g_{i-j+1} m_j \quad (12.7)$$

where $g_i = 0$, if $i < 1$ or $i > N$. This equation is linear in the unknown filter coefficients and can be written in the form $\mathbf{Gm} = \mathbf{d}$, where

$$\mathbf{G} = \Delta t \begin{bmatrix} g_1 & 0 & 0 & \cdots & 0 \\ g_2 & g_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_N & g_{N-1} & g_{N-2} & \cdots & g_{N-M+1} \end{bmatrix} \quad (12.8)$$

The time series d_i is identified with the data and the filter m_i with the model parameters. The equation is therefore an overdetermined linear system for

$M < N$ filter coefficients. For this problem to be consistent with the tenets of probability theory, however, \mathbf{g} must be known exactly, while \mathbf{d} must contain uncorrelated Gaussian noise of uniform variance.

Many approaches are available for solving this inverse problem. The simplest is to use the least squares equation $[\mathbf{G}^T \mathbf{G}] \mathbf{m}^{\text{est}} = \mathbf{G}^T \mathbf{d}$. This formulation is especially attractive because the matrices $[\mathbf{G}^T \mathbf{G}]$ and $\mathbf{G}^T \mathbf{d}$ can be computed analytically as

$$\mathbf{G}^T \mathbf{G} = (\Delta t)^2 \begin{bmatrix} \sum_{i=1}^N g_i^2 & \sum_{i=2}^{N-1} g_i g_{i-1} & \cdots \\ \sum_{i=2}^{N-1} g_i g_{i-1} & \sum_{i=1}^{N-1} g_i^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad \text{and} \quad \mathbf{G}^T \mathbf{d} = \Delta t \begin{bmatrix} \sum_{i=1}^N d_i g_i \\ \sum_{i=2}^{N-1} d_i g_{i-1} \\ \vdots \end{bmatrix} \quad (12.9)$$

Furthermore, the summations in $\mathbf{G}^T \mathbf{G}$ usually can be approximated adequately as all having the upper limit of N , so that the resulting matrix is *Toeplitz* (meaning that it has constant diagonals). Then $\mathbf{G}^T \mathbf{G}$ matrix contains the *autocorrelation* of \mathbf{g} (denoted $\mathbf{g} \star \mathbf{g}$) and $\mathbf{G}^T \mathbf{d}$ the *cross-correlation* of \mathbf{g} with \mathbf{d} (denoted $\mathbf{g} \star \mathbf{d}$)

$$[\mathbf{G}^T \mathbf{G}]_{ij} = (\Delta t)^2 [\mathbf{g} \star \mathbf{g}]_{|i-j|+1} \quad \text{and} \quad [\mathbf{G}^T \mathbf{d}]_i = \Delta t [\mathbf{g} \star \mathbf{d}]_i, \quad (12.10)$$

$$\text{where } [\mathbf{a} \star \mathbf{b}]_i = \sum_{j=1}^N a_j b_{i+j-1}$$

However, one often finds that the least squares solution is very rough, because the high frequency components of $m(t)$ are poorly constrained (that is, the problem is really mixed-determined).

An alternative approach is to incorporate *a priori* information of smoothness using weighted damped least squares $\mathbf{F}^T \mathbf{F} \mathbf{m} = \mathbf{F}^T \mathbf{d}$, where the matrix \mathbf{F} contains both \mathbf{G} and a smoothness matrix \mathbf{H} , scaled by a damping parameter ε^2 (see Equation (3.46)). If the biconjugate gradient method is used to solve this equation (see Section 3.9.3), then the only quantity that need be computed is the product $\mathbf{F}^T (\mathbf{F} \mathbf{v}) = \mathbf{G}^T (\mathbf{G} \mathbf{v}) + \varepsilon^2 \mathbf{H}^T (\mathbf{H} \mathbf{v})$, where \mathbf{v} is an arbitrary vector. The $\mathbf{G}^T (\mathbf{G} \mathbf{v})$ product can be computed extremely efficiently starting with \mathbf{g} and using *MatLab* `xcorr()` cross-correlation function (see the accompanying `filterfun()` function for details).

As an example of filter construction, we shall consider a time series $g(t)$, which represents a recording of the sound emitted by a seismic exploration airgun (Figure 12.3A). Signals of this sort are used to detect layering at depth in the earth through echo sounding. Ideally, a very spiky sound from the airgun is best because it allows echoes from layers at depth to be most easily detected. Engineering constraints, however, limit the airgun signal to a series of pulses.

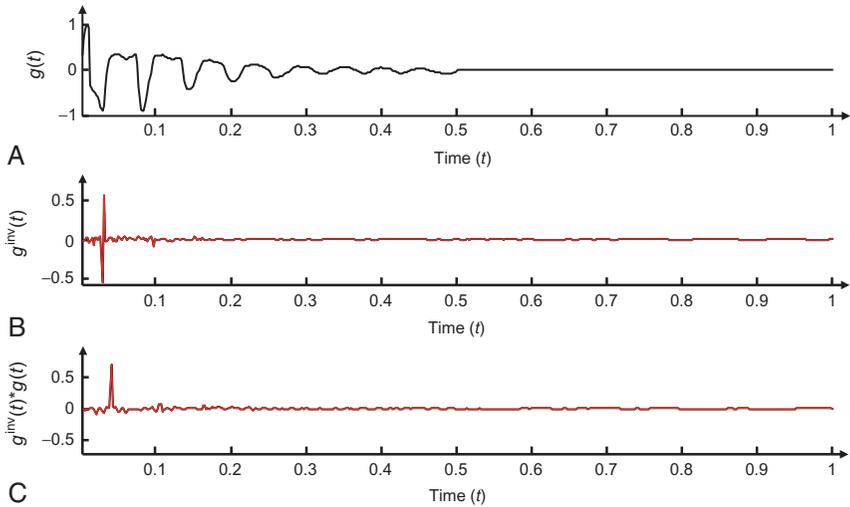


FIGURE 12.3 (A) An airgun signal $g(t)$, after Smith (1975). Ideally, the inverse filter $g^{\text{inv}}(t)$ when convolved with $g(t)$ should produce the spike $\delta(t - t_0)$, centered at time t_0 . (B) Estimate of the inverse filter $g^{\text{inv}}(t)$ for $t_0 = 0.04$, computed via generalized least squares with *a priori* information on solution size and smoothness. (C) The convolution of $g(t)$ with the estimated $g^{\text{inv}}(t)$. While not a perfect spike, the result is significantly spikier than the airgun signal, $g(t)$. *MatLab* script gda12_02.

We shall attempt, therefore, to find a filter that, when applied to the airgun pulse, produces a signal spike or delta function $\mathbf{d} = [0, 0, 0, \dots, 0, 1, 0, \dots, 0]^T$ centered on the largest pulse in the original signal. This filter can then be applied to the recorded echo soundings to remove the reverberation of the airgun and reveal the layering of the earth. The least squares filter $m(t)$ (computed for this example using weighted damped least squares with both smoothness and length constraints and solved with the biconjugate gradient method) is shown in Figure 12.3B and the resulting signal $d^{\text{pre}}(t) = m(t) * g(t)$ in Figure 12.3C. Note that, although the reverberations are reduced in amplitude, they are by no means completely removed.

12.3 ADJUSTMENT OF CROSSOVER ERRORS

Consider a set of radar altimetry data from a remote-sensing satellite. These data consist of measurements of the distance from the satellite to the surface of the earth directly below the satellite. If the altitude of the satellite with respect to the earth's center were known, then these data could be used to measure the elevation of the surface of the earth. Unfortunately, while the height of the satellite during each orbit is approximately constant, its exact value is unknown. Since the orbits crisscross the earth, one can try to solve for the satellite height in each orbit by minimizing the overall crossover error (Kaula, 1966).

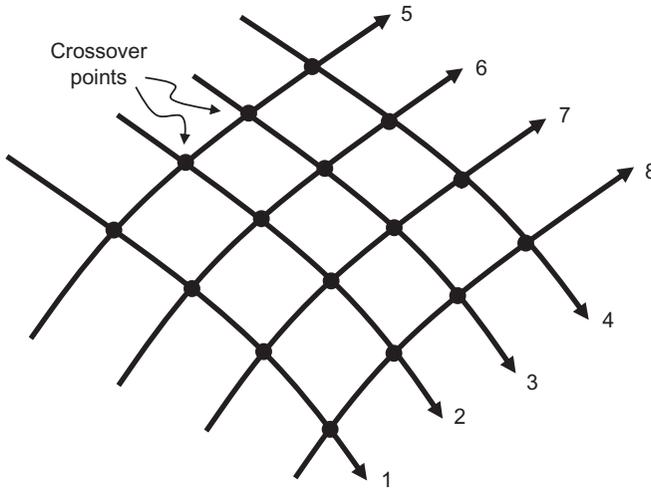


FIGURE 12.4 Descending tracks 1–4 intersect ascending tracks 5–8 at 16 points. The height of the satellite along each track is determined by minimizing the crossover error at the intersections.

Suppose that there are M orbits and that the unknown altitude of the satellite during the i th orbit is m_i . We shall divide these orbits into two groups (Figure 12.4), the ascending orbits (when the satellite is traveling north) and the descending orbits (when the satellite is traveling south). The ascending and descending orbits intersect at N points. At one such point ascending orbit number A_i intersects with descending orbit D_i (where the numbering refers to the ordering in \mathbf{m}). At this point, the two orbits have measured a satellite-to-earth distance of, say, s_{A_i} and s_{D_i} , respectively. The elevation of the ground is $m_{A_i} - s_{A_i}$ according to the data collected on the ascending orbit and $m_{D_i} - s_{D_i}$, according to the data from the descending orbit. The crossover error at the i th intersection is $e_i = (m_{A_i} - s_{A_i}) - (m_{D_i} - s_{D_i}) = (m_{A_i} - m_{D_i}) - (s_{A_i} - s_{D_i})$. The assertion that the crossover error should be zero leads to a linear equation of the form $\mathbf{G}\mathbf{m} = \mathbf{d}$, where

$$G_{ij} = \delta_{jA_i} - \delta_{jD_i} \quad \text{and} \quad d_i = s_{A_i} - s_{D_i} \quad (12.11)$$

Each row of the data kernel contains one 1, one -1 , and $M - 2$ zeros. Note that the matrix \mathbf{G} is extremely sparse; we would be well advised to declare it as such in a *MatLab* script.

Initially, we might assume that we can use simple least squares to solve this problem. We note, however, that the solution is always to a degree underdetermined. Any constant can be added to all the m s without changing the crossover error since the error depends only on the difference between the elevations of the satellite during the different orbits. This problem is therefore mixed-determined. We should therefore impose the *a priori* constraint that $\sum_i m_i = 0$. While this constraint is not physically realistic (implying as it does

that the satellite has on average zero altitude), it serves to remove the underdeterminacy. Any desired constant can subsequently be added to the solution.

This constraint can be approximately implemented with damped least squares

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G} + \varepsilon^2 \mathbf{I}]^{-1} \mathbf{G}^T \mathbf{d} \quad (12.12)$$

As long as damping parameter ε^2 is chosen carefully, the solution will very closely approximate the exact one. An example is shown in Figure 12.5.

As with the previous cases that we have studied, the matrices $[\mathbf{G}^T \mathbf{G}]$ and $[\mathbf{G}^T \mathbf{d}]$ can be computed analytically; furthermore, there is good reason for doing so. A realistic problem may have thousands of orbits that intersect at millions of points. The data kernel will therefore be very large, with dimensions on the order of $1,000,000 \times 1000$. We find

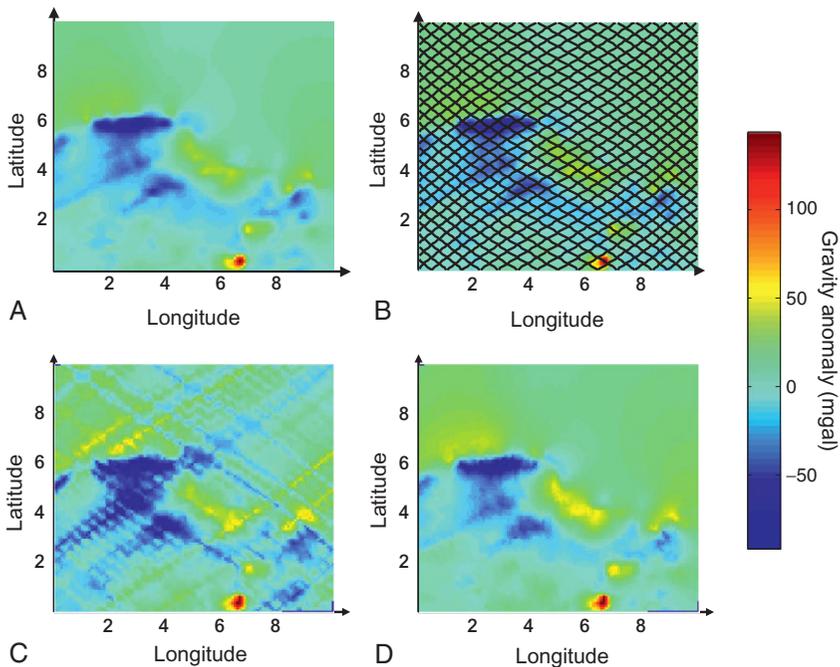


FIGURE 12.5 Example of crossover error adjustment of satellite gravity data. (A) True gravity anomaly data for the equatorial Atlantic Ocean. It reflects variations in the depth of the seafloor and density variations within the oceanic crust. (B) Hypothetical satellite tracks, along which the gravity is measured. The measurements along each track have a constant offset reflecting errors in the assumed altitude of the satellite. (C) Reconstructed gravity anomaly without crossover correction. Artifacts parallel to the tracks are clearly visible. (D) Reconstructed gravity anomaly with crossover correction. The artifacts are eliminated. Data courtesy of Bill Haxby, Lamont-Doherty Earth Observatory. *MatLab* script `gda12_03`.

$$\begin{aligned}
[\mathbf{G}^T \mathbf{G}]_{rs} &= \sum_{i=1}^N G_{ir} G_{is} = \sum_{i=1}^N (\delta_{rA_i} - \delta_{rD_i})(\delta_{sA_i} - \delta_{sD_i}) \\
&= \sum_{i=1}^N (\delta_{rA_i} \delta_{sA_i} - \delta_{rA_i} \delta_{sD_i} - \delta_{rD_i} \delta_{sA_i} + \delta_{rD_i} \delta_{sD_i})
\end{aligned} \tag{12.13}$$

The diagonal elements of $[\mathbf{G}^T \mathbf{G}]$ are

$$[\mathbf{G}^T \mathbf{G}]_{rr} = \sum_{i=1}^N (\delta_{rA_i} \delta_{rA_i} - 2\delta_{rA_i} \delta_{rD_i} + \delta_{rD_i} \delta_{rD_i}) \tag{12.14}$$

The first term contributes to the sum whenever the ascending orbit is r , and the third term contributes whenever the descending orbit is r . The second term is zero since an orbit never intersects itself. The r th element of the diagonal is the number of times the r th orbit is intersected by other orbits.

Only the two middle terms of the sum in the expression for $[\mathbf{G}^T \mathbf{G}]_{rs}$ contribute to the off-diagonal elements. The second term contributes whenever $A_i = r$ and $D_i = s$, and the third when $A_i = s$ and $D_i = r$. The (r, s) off-diagonal element is the number of times the r th and s th orbits intersect, multiplied by -1 .

The other matrix product is

$$[\mathbf{G}^T \mathbf{d}]_r = \sum_{i=1}^N G_{ir} d_i = \sum_{i=1}^N (\delta_{rA_i} - \delta_{rD_i}) d_i \tag{12.15}$$

We note that the delta functions can never both equal 1 since an orbit can never intersect itself. Therefore $[\mathbf{G}^T \mathbf{d}]_r$ is the sum of all the d_s that have ascending orbit number $A_i = r$ minus the sum of all the d_s that have descending orbit number $D_i = r$.

We can then compute the matrix products. We first prepare a table that gives the ascending orbit number A_i , descending orbit number D_i , and elevation difference d_i for each of the N orbital intersections. We then start with $[\mathbf{G}^T \mathbf{G}]$ and $[\mathbf{G}^T \mathbf{d}]$ initialized to zero and, for each i th row of the table, execute the following steps:

- (1) Add 1 to the $r = A_i$, $s = A_i$ element of $[\mathbf{G}^T \mathbf{G}]_{rs}$.
- (2) Add 1 to the $r = D_i$, $s = D_i$ element of $[\mathbf{G}^T \mathbf{G}]_{rs}$.
- (3) Subtract 1 from the $r = A_i$, $s = D_i$ element of $[\mathbf{G}^T \mathbf{G}]_{rs}$.
- (4) Subtract 1 from the $r = D_i$, $s = A_i$ element of $[\mathbf{G}^T \mathbf{G}]_{rs}$.
- (5) Add d_i to the $r = A_i$ element of $[\mathbf{G}^T \mathbf{d}]_r$.
- (6) Subtract d_i from the $r = D_i$ element of $[\mathbf{G}^T \mathbf{d}]_r$.

The final form for $\mathbf{G}^T \mathbf{G}$ is relatively simple. If two orbits intersect at most once, then it will contain only zeros and ones on its off-diagonal elements. As an alternative to damped least squares, the $\sum_i m_i = 0$ constraint can be implemented exactly using the Lagrange multiplier method (see [Section 3.10](#)).

$$\begin{bmatrix} [\mathbf{G}^T \mathbf{G}] & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ \lambda \end{bmatrix} = \begin{bmatrix} [\mathbf{G}^T \mathbf{d}] \\ 0 \end{bmatrix} \quad (12.16)$$

Here, $\mathbf{1}$ is a length M column vector of ones, λ is a Lagrange multiplier, and the matrix on the left-hand side of the equation is $M+1 \times M+1$.

12.4 AN ACOUSTIC TOMOGRAPHY PROBLEM

An acoustic tomography problem was discussed previously in [Section 1.1.3](#). In that simple case, travel times d_i of sound rays are measured through the rows and columns of a square 4×4 grid of bricks, each having height and width h and acoustic slowness m_i . The data kernel G_{ij} represents the length of ray i in brick j . Since the sound raypaths are either exactly horizontal or exactly vertical, each of the lengths is equal to h and the data kernel is

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_8 \end{bmatrix} = h \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ d_{16} \end{bmatrix} \quad (12.17)$$

The matrix \mathbf{G} is sparse, since the typical ray crosses just a small fraction of the total number of bricks. In the general case, where the image is divided up into rectangular pixels and where the raypaths are slanted and/or curved, the data kernel G_{ij} still represents the length of ray i in pixel j , but now that length is no longer constant. These variable lengths may be difficult to calculate analytically; instead, one must resort to numerical approximations.

One commonly used technique begins by initializing the data kernel to zero and then stepping along each ray i in arc length increments of Δs , chosen to be much smaller than the size of the pixels. The pixel index j of the center of each increment is determined, and the whole increment is added to the corresponding element of the data kernel; that is, $G_{ij} \rightarrow G_{ij} + \Delta s$.

We examine a test case where a square object is divided into a 256×256 grid of pixels ([Figure 12.6A](#)), with the model parameter m_i representing the acoustic slowness within the pixels. A set of evenly distributed source and receiver points are placed on the four edges of the square and connected with straight-line rays ([Figure 12.6A](#)). The data kernel is constructed using the approximate ray-steeping method described above and a data set of synthetic travel times is constructed via $\mathbf{d} = \mathbf{G}\mathbf{m}^{\text{true}}$. Having an effective way of plotting travel time data is important, for instance, in detecting outliers. We parameterize each ray by its perpendicular distance r from the center of the object and by its angle θ from the horizontal and form an (r, θ) image of the data ([Figure 12.6C](#)) for this purpose. The estimated image ([Figure 12.6D](#)), computed using damped least squares solved with the biconjugate gradient method,

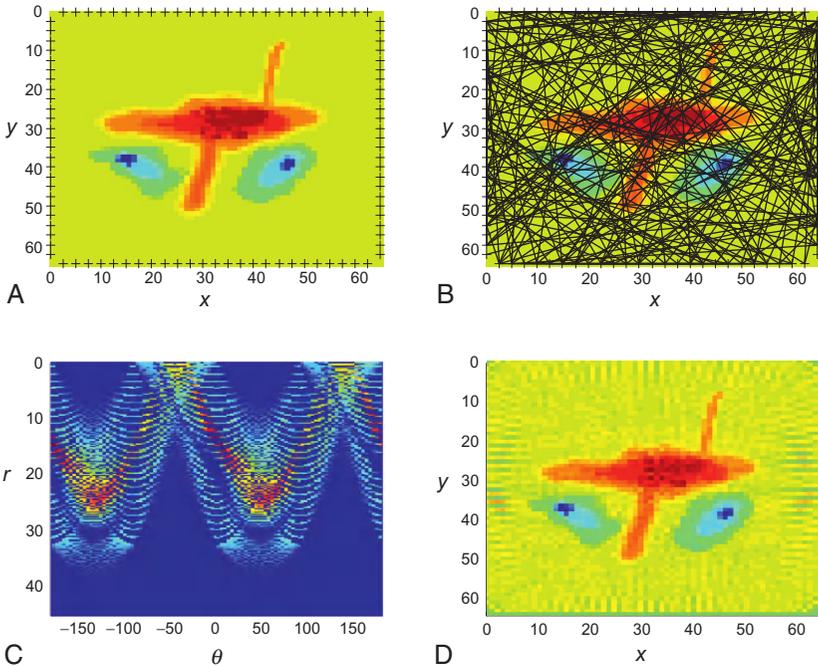


FIGURE 12.6 Acoustic tomography problem. (A) True image. (B) Ray paths (only a few percent of rays are shown, else the image would be black). (C) Travel time data, organized by the distance r and angle θ of the ray from the midpoint of the image. (D) Reconstructed image. See text for further discussion. *MatLab* script gda12_04.

recovers most of the long-wavelength features of the image, but contains faint streaks along ray paths. These streaks can be eliminated by increasing the number of rays (not shown).

The success of tomography is critically dependent upon the ray coverage, which must not only be spatially dense but must—at every point—cover a 90° suite of angles (from horizontal to vertical). Tomographic reconstructions based on poorer ray coverage (Figure 12.7) generally have poor resolution.

12.5 ONE-DIMENSIONAL TEMPERATURE DISTRIBUTION

As a hot slab within a uniform whole space cools, heat flows from the slab to the surrounding material, slowly warming it. The width of the warm zone slowly grows with time and the boundary between the slab and the whole space, initially distinct, slowly fades away. If several hot slabs are present, the temperatures from each blends together as time increases, making them hard to distinguish. The question that we pose is how well the initial pattern of temperatures can be reconstructed, given the temperature profile measured at some later time.

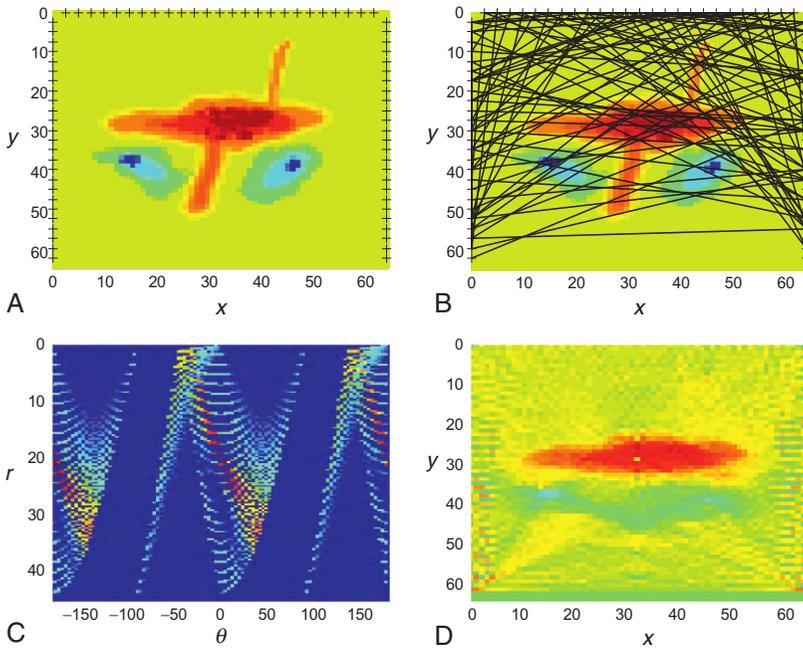


FIGURE 12.7 Acoustic tomography problem with deficient distribution of rays. (A) True image. (B) Ray paths (only a few percent of rays are shown, else the image would be black). (C) Travel time data, organized by the distance r and angle θ of the ray from the midpoint of the image. (D) Reconstructed image. See text for further discussion. *MatLab* script gda12_05.

For a single slab, the temperature $T(x, t)$ at position x and time t can be shown to be (Abbott and Menke, 1990, their [Section 6.3.3](#))

$$T(x, t) = \frac{1}{2} T_0 \left\{ \operatorname{erf} \left[\frac{x - (\xi - 1/2h)}{\sqrt{t}} \right] - \operatorname{erf} \left[\frac{x - (\xi + 1/2h)}{\sqrt{t}} \right] \right\} = T_0 g(x, t, \xi) \quad (12.18)$$

Here x is distance measured perpendicular to the face of the slab, h is the thickness of the slab, and ξ is its position ([Figure 12.8A](#)). The slab has initial temperature T_0 while the whole space is initially at zero temperature. The thermal diffusivity of both materials is taken, for simplicity, to be unity. The special function $\operatorname{erf}()$ is called the *error function*; *MatLab*'s implementation of it has the same name.

If several slabs of different temperature m_i are placed face-to-face within the whole space, the temperature at position x_i and time t is

$$T(x_i, t) = \sum_{j=1}^M g(x_i, t, \xi_j) m_j \quad (12.19)$$

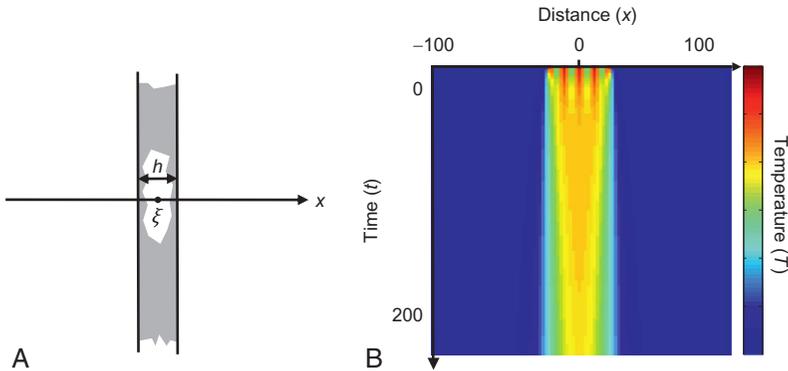


FIGURE 12.8 (A) Single hot slab of thickness, h , located at position, $x = \zeta$. (B) Temporal evolution of the temperature $T(x, t)$ of 100 adjacent slabs. The initial temperature distribution of the slabs, $T(x, t = 0)$, is taken to be the model parameter vector, \mathbf{m} . It is nonzero only for slabs near $|x| \leq 20$. The temperature, $T(x, t = 0)$, at subsequent times, t , can be computed from the initial temperature distribution, since the data kernel can be calculated from the physics of heat transport. Note that the band of hot temperatures widens with increasing time, and that fine scale temperature fluctuations are preferentially attenuated. *MatLab* script `gda12_06`.

The inverse problem that we consider is how well the temperature distribution m_i can be reconstructed by making measurements of the temperature at all positions, but at a fixed time, t . We might anticipate that the reconstruction will be well resolved at short times, since heat will not have much time to flow and the initial pattern of temperature will still be partly preserved. On the other hand, it will be poorly resolved at long times, since the initial pattern of temperature will have faded away.

Our test scenario has $M = 100$ slabs located in the distance interval $|x| \leq 20$, which have a sinusoidal temperature pattern with five oscillations, overall (Figure 12.8B). The details of this pattern fade with time, so that after about $t \approx 20$ only a broad warm zone is present. The width of this warm zone slowly grows with time, roughly doubling by $t \approx 250$.

We reconstruct the initial temperature using $N = 100$ observations of temperature $T(x_i, t)$ from equally spaced between $-100 < x < 100$ all made at a fixed time t . Two different inversion methods are used, minimum length and Backus-Gilbert (Figure 12.9). As hypothesized, their quality falls off rapidly with observation time t , with little detail being present after $t \approx 50$. The minimum length inversion does better at recovering the sinusoidal pattern, but it also contains artifacts, especially at long time, where two instead of five oscillations appear to be present. In contrast, the Backus-Gilbert inversion provides a poorer reconstruction, but one that is artifact free. These differences can be understood by examining the model resolution matrices of the two methods (Figure 12.10). The minimum length inversion has the narrower resolution, but also the stronger sidelobes.

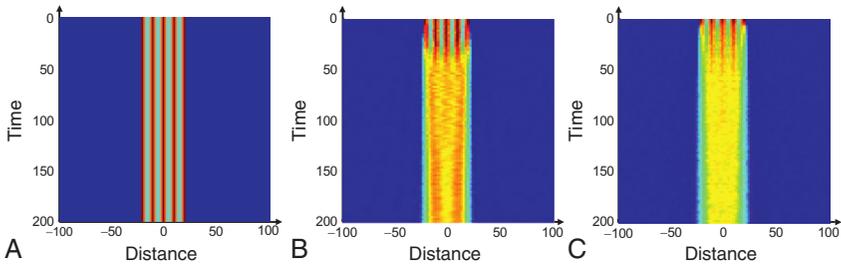


FIGURE 12.9 Model for the temperature distribution problem. (A) The true model is the initial temperature distribution, $T(x, t=0)$. While this function is not a function of time, it is displayed on the (x, t) image for comparison purposes. (B) Minimum length (ML) estimate of the model, $T(x, t=0)$, for a data set consisting of observations at all distances at a single time $t > 0$. (C) Corresponding Backus-Gilbert (BG) estimate. In both the ML and BG cases, the ability of the data to resolve fine details declines with time, with the BG case declining fastest. *MatLab* script gda12_06.

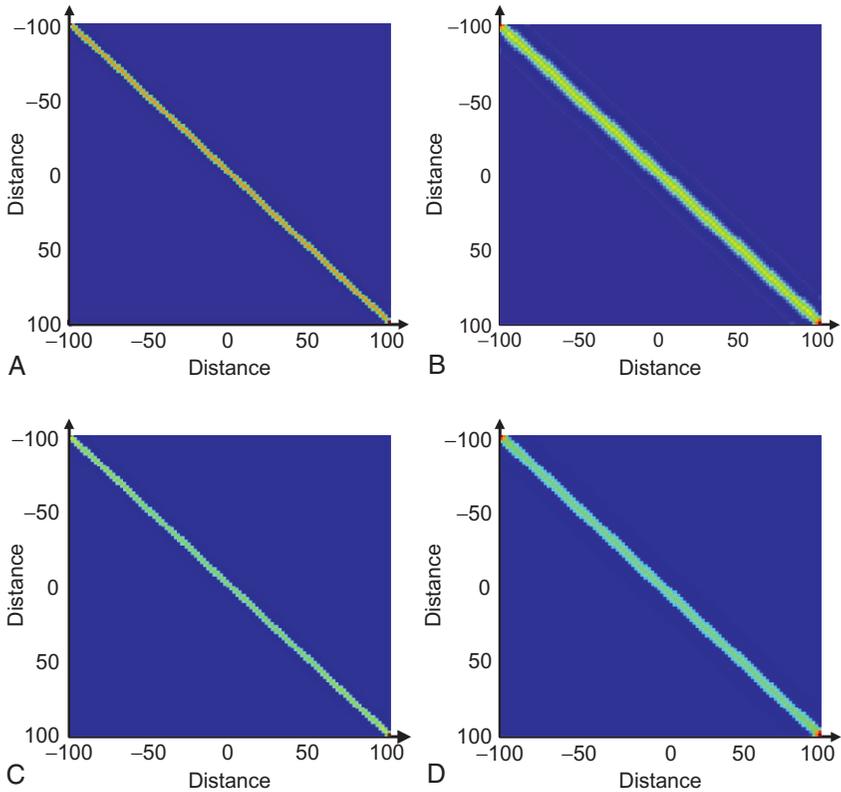


FIGURE 12.10 Model resolution matrices for the temperature distribution problem. (A) Minimum length (ML) solution for data at time $t = 10$. (B) ML solution for data at time $t = 40$. (C) Backus-Gilbert (BG) solution for data at time $t = 10$. (D) BG solution for data at time $t = 40$. Note that the BG resolution matrix has smaller sidelobes than the corresponding ML case. *MatLab* script gda12_06.

12.6 L_1 , L_2 , AND L_∞ FITTING OF A STRAIGHT LINE

The L_1 , L_2 , and L_∞ problem is to fit the straight line $d_i = m_1 + m_2 z_i$ to a set of (z, d) pairs by minimizing the prediction error under a variety of norms. This is a linear problem with an $N \times 2$ data kernel

$$\mathbf{G} = \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} \quad (12.20)$$

The L_2 norm is the simplest to implement. It implies that the error follows a Gaussian probability density function with $\text{cov } \mathbf{d} = \sigma_d^2 \mathbf{I}$. The simple least squares solution $\mathbf{m}^{\text{est}} = [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T \mathbf{d}$ is adequate since the problem typically is very overdetermined. Since the L_2 problem has been discussed, we shall not treat it in detail here. However, it is interesting to compute the data resolution matrix $\mathbf{N} = \mathbf{G} \mathbf{G}^{-\text{g}}$

$$\mathbf{N} = \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} \frac{1}{N \sum z_i^2 - (\sum z_i)^2} \begin{bmatrix} \sum_{k=1}^N z_k^2 & -\sum_{k=1}^N z_k \\ -\sum_{k=1}^N z_k & N \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ z_1 & z_2 & z_3 & \cdots & z_N \end{bmatrix} \quad (12.21)$$

$$N_{ij} = \frac{\sum z_k^2 - (z_i + z_j) \sum z_k + z_i z_j N}{N \sum z_i^2 - (\sum z_i)^2} = A_i + B_i z_j$$

Each row of the resolution matrix N_{ij} is a linear function of z_j , so that the elements with the largest absolute value are at an edge of the matrix and not along its main diagonal. The resolution is not at all localized; instead the points with most extreme z_i control the fit of the straight line.

The L_1 and L_∞ estimates can be determined by using the transformation to a linear programming problem described in [Chapter 8](#). Although more efficient algorithms exist, we shall set up the problems so that they can be solved with a standard linear programming algorithm. This algorithm determines a vector \mathbf{y} that minimizes $\mathbf{c}^T \mathbf{y}$ subject to $\mathbf{A} \mathbf{y} = \mathbf{b}$ and $\mathbf{y} \geq 0$. The first step is to define two new variables \mathbf{m}' and \mathbf{m}'' such that $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$. This definition relaxes the positivity constraints on the model parameters. For the L_1 problem, we define three additional vectors $\boldsymbol{\alpha}$, \mathbf{x} , and \mathbf{x}' and then arrange them in the form of a linear programming problem in $2M + 3N$ variables and $2N$ constraints as

$$\begin{aligned} \mathbf{y}^T &= [[m'_1, \dots, m'_M], [m''_1, \dots, m''_M], [\alpha_1, \dots, \alpha_N], [x_1, \dots, x_N], [x'_1, \dots, x'_N]] \\ \mathbf{c}^T &= [[0, \dots, 0], [0, \dots, 0], [1, \dots, 1], [0, \dots, 0], [0, \dots, 0]] \\ \mathbf{A} &= \begin{bmatrix} \mathbf{G}_{N \times M} & -\mathbf{G}_{N \times M} & -\mathbf{I}_{N \times N} & \mathbf{I}_{N \times N} & \mathbf{O}_{N \times N} \\ \mathbf{G}_{N \times M} & -\mathbf{G}_{N \times M} & \mathbf{I}_{N \times N} & \mathbf{O}_{N \times N} & -\mathbf{I}_{N \times N} \end{bmatrix} \\ \mathbf{b}^T &= [[d_1, \dots, d_N], [d_1, \dots, d_N]] \end{aligned} \quad (12.22)$$

The L_∞ problem is transformed into a linear programming problem with additional variables \mathbf{x} and \mathbf{x}' and a scalar parameter α . The transformed problem in $2M + 2N + 1$ unknowns and $2N$ constraints is

$$\begin{aligned} \mathbf{y}^T &= [[m'_1, \dots, m'_M], [m''_1, \dots, m''_M], [\alpha], [x_1, \dots, x_N], [x'_1, \dots, x'_N]] \\ \mathbf{c}^T &= [[0, \dots, 0], [0, \dots, 0], [1], [0, \dots, 0], [0, \dots, 0]] \\ \mathbf{A} &= \begin{bmatrix} \mathbf{G}_{N \times M} & -\mathbf{G}_{N \times M} & -\mathbf{1} & \mathbf{I}_{N \times N} & \mathbf{0}_{N \times N} \\ \mathbf{G}_{N \times M} & -\mathbf{G}_{N \times M} & \mathbf{1} & \mathbf{0}_{N \times N} & -\mathbf{I}_{N \times N} \end{bmatrix} \\ \mathbf{b}^T &= [[d_1, \dots, d_N], [d_1, \dots, d_N]] \end{aligned} \quad (12.23)$$

We illustrate the results of using these three different norms to data with exponentially distributed error (Figure 12.11). Note that the L_1 line is by far the best; it is designed to give little weight to outliers, which are common in data with exponentially distributed noise. Both the L_1 and L_∞ fits may be nonunique. Most versions of the linear programming algorithm will find only one solution, so the process of identifying the complete range of minimum solutions may be difficult.

12.7 FINDING THE MEAN OF A SET OF UNIT VECTORS

Suppose that a set of measurements of direction (defined by unit vectors in a three-dimensional Cartesian space) are thought to scatter randomly about a mean direction (Figure 12.12). How can the mean vector be determined?

This problem is similar to that of determining the mean of a group of scalar quantities (Sections 5.1 and 8.2) and is solved by direct application of the

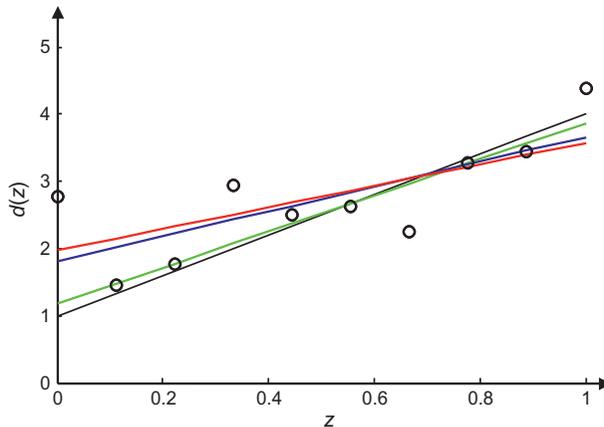


FIGURE 12.11 Fitting a straight line to data $d(z)$. The true model (black line) is the straight line, $d^{\text{true}}(z) = 1 + 3z$. The $N = 10$ observations d_i^{obs} are the true data perturbed with exponentially distributed noise with variance $\sigma_d^2 = (0.4)^2$. Three different fits have been computed, by minimizing the L_1 , L_2 , and L_∞ norms of the error. The corresponding predicted data are shown in green, blue, and red, respectively. *MatLab* script gda12_07.

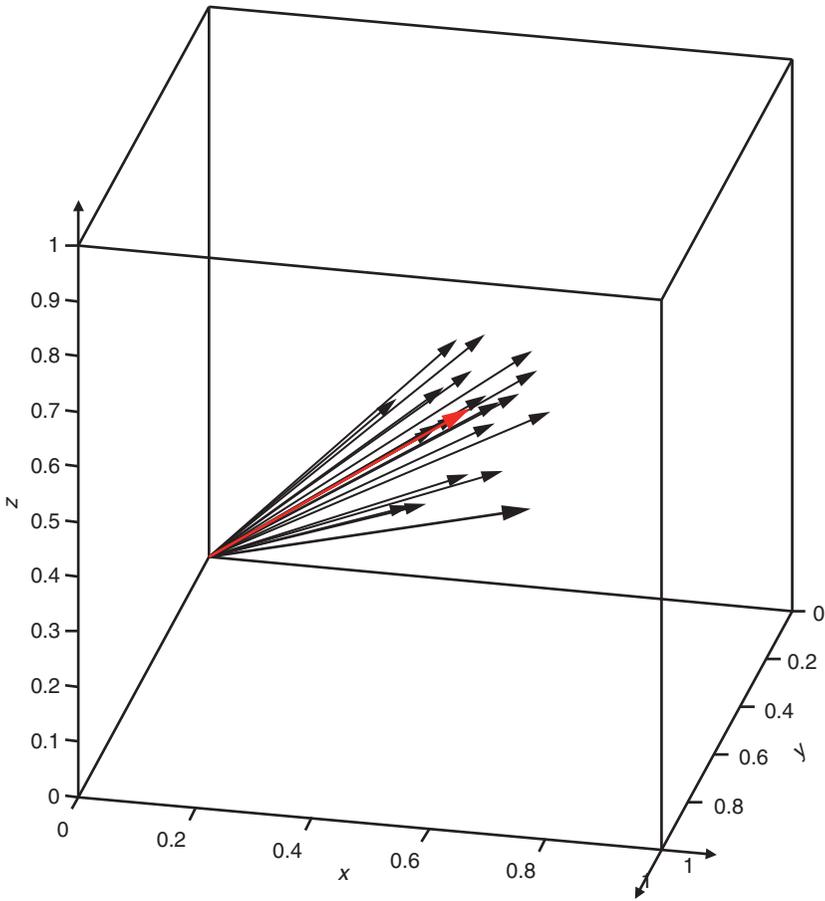


FIGURE 12.12 Several unit vectors (black) scattering about a central direction (red). *MatLab* script gda12_08.

principle of maximum likelihood. In the scalar mean problems, we assume that the data possess a Gaussian or exponential probability density function and then apply the principle of maximum likelihood to estimate a single model parameter, the mean. Neither of these probability density functions is applicable to directional data because they are defined on the wrong interval $(-\infty, +\infty]$, instead of $[0, \pi]$. A better choice is the Fisher probability density function (Fisher, 1953). Its vectors are clumped near the mean direction with no preferred azimuthal direction. It proposes that the probability of finding a vector in an increment of solid angle $d\Omega = \sin(\theta)d\theta d\phi$, located at an angle with inclination θ and azimuth ϕ from the mean direction (Figure 12.13), is

$$p(\theta, \phi) = \frac{\kappa}{4\pi \sinh(\kappa)} \exp[\kappa \cos(\theta)] \quad (12.24)$$

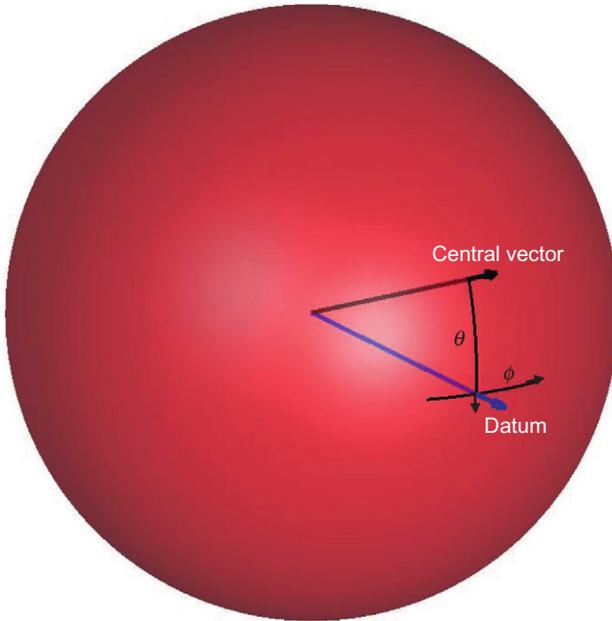


FIGURE 12.13 Unit sphere, showing coordinate system used in Fisher distribution. *MatLab* script gda12_09.

This distribution is peaked near the mean direction $\theta=0$, and its width depends on the value of the *precision parameter* κ (Figure 12.14). The reciprocal of this parameter serves a role similar to the variance in the Gaussian distribution. When $\kappa=0$ the distribution is completely white or random on the sphere, but when $\kappa \gg 1$ it becomes very peaked near the mean direction.

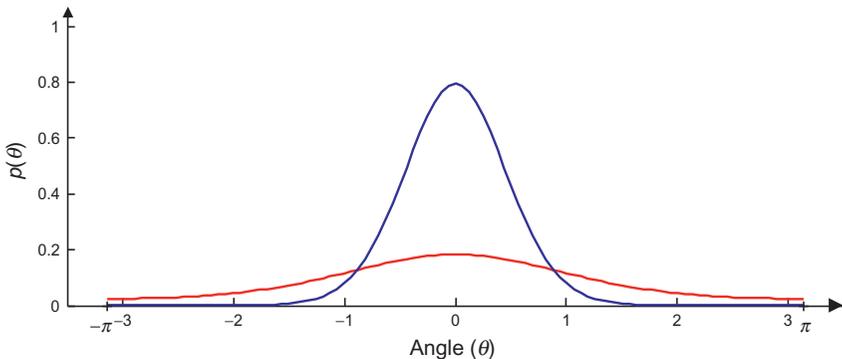


FIGURE 12.14 Fisher distribution, $p(\theta, \phi)$, for a large precision parameter ($\kappa=5$, blue curve) and a small precision parameter ($\kappa=1$, red curve). *MatLab* script gda12_10.

If the data are specified by N Cartesian unit vectors (x_i, y_i, z_i) and the mean by its unit vector (m_1, m_2, m_3) , then the cosine of the inclination for any data is just the dot product $\cos(\theta_i) = x_i m_1 + y_i m_2 + z_i m_3$. The joint probability density function for the data $p(\theta, \phi)$ is then the product of N distributions of the form $p(\theta_i, \phi_i) \sin(\theta_i)$. The $\sin(\theta_i)$ term must be included since we are using Cartesian coordinates, as contrasted to polar coordinates, to represent directions. The joint distribution is then

$$p(\theta, \phi) = \left[\frac{\kappa}{4\pi \sinh(\kappa)} \right]^N \exp \left[\kappa \sum_{i=1}^N \cos(\theta_i) \right] \prod_{i=1}^N \sin(\theta_i) \quad (12.25)$$

where $\cos(\theta_i) = \mathbf{x}_i^T \mathbf{m} = [x_i m_1 + y_i m_2 + z_i m_3]$. The likelihood function is

$$\begin{aligned} L = \log(p) &= N \log(\kappa) - N \log(4\pi) - N \log[\sinh(\kappa)] \\ &+ \kappa \sum_{i=1}^N [x_i m_1 + y_i m_2 + z_i m_3] + \sum_{i=1}^N \log[\sin(\theta_i)] \end{aligned} \quad (12.26)$$

The best estimate of model parameters occurs when the likelihood is maximized. However, since the solution is assumed to be a unit vector, L must be maximized under the constraint that \mathbf{m} represents a unit vector; that is, $\sum_i m_i^2 = 1$. We first simplify this maximization by making an approximation. As long as κ is reasonably large (say, $\kappa > 5$), any variations in the magnitude of the joint probability that are caused by varying the m_i will come mainly from the exponential, since it varies much faster than the sine. We therefore ignore the last term in the likelihood function when computing the derivatives $\partial L / \partial m_q$. The Lagrange multiplier equations for the problem are then approximately

$$\begin{aligned} \kappa \sum_i x_i - 2\lambda m_1 &= 0 \\ \kappa \sum_i y_i - 2\lambda m_2 &= 0 \\ \kappa \sum_i z_i - 2\lambda m_3 &= 0 \\ \frac{N}{\kappa} - N \frac{\cosh(\kappa)}{\sinh(\kappa)} + \sum_{i=1}^N [x_i m_1 + y_i m_2 + z_i m_3] &= 0 \end{aligned} \quad (12.27)$$

where λ is the Lagrange multiplier. The first three equations can be solved simultaneously along with the constraint equation for the model parameters as

$$[m_1, m_2, m_3]^T = \frac{\left[\sum_i x_i, \sum_i y_i, \sum_i z_i \right]^T}{\left\{ \left(\sum_i x_i \right)^2 + \left(\sum_i y_i \right)^2 + \left(\sum_i z_i \right)^2 \right\}^{1/2}} \quad (12.28)$$

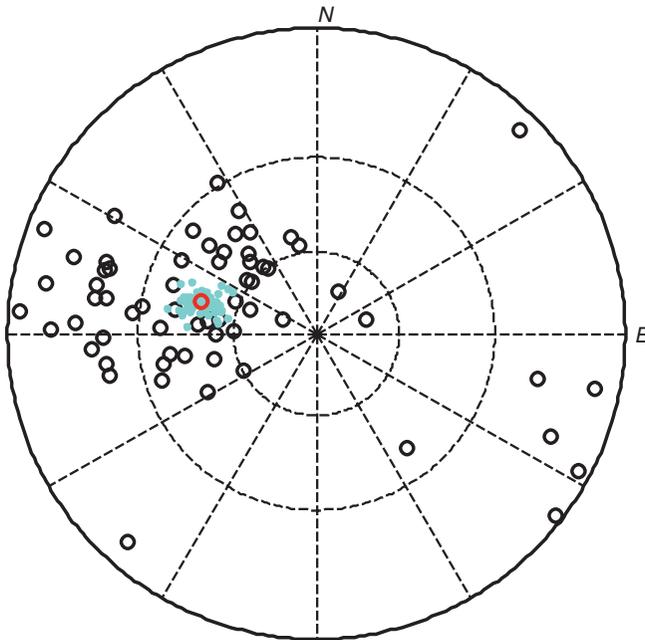


FIGURE 12.15 Lower hemisphere stereonet showing P -axes of deep (300–600 km) earthquakes in the Kurile-Kamchatka subduction zone. The axes mostly dip to the west, parallel to the direction of subduction, indicating that the subducting slab is in down-dip compression. (Black circles) Axes of individual earthquakes. (Red circle) central axis, computed by maximum-likelihood technique. (Blue dots) Scatter of central axis determined by bootstrapping. Data courtesy of the Global CMT Project. *MatLab* script gda12_11.

Note that the mean vector is the normalized vector sum of the individual observed unit vectors. The fourth equation is an implicit transcendental equation for κ . Since we have assumed that $\kappa > 5$, we can use the approximation $\cosh(\kappa)/\sinh(\kappa) \approx 1$, and the fourth equation yields

$$\kappa \approx \frac{N}{N - \sum_i \cos(\theta_i)} \quad (12.29)$$

An example is shown in [Figure 12.15](#).

12.8 GAUSSIAN AND LORENTZIAN CURVE FITTING

Many types of spectral data consist of several overlapping peaks, each of which has either Gaussian or *Lorentzian* shape. (Both the Gaussian and the Lorentzian have a single maximum, but the Lorentzian is much longer tailed.) The problem is to determine the location, area, and width of each peak through least squares curve fitting.

Suppose that the data consist of $N(z, d)$ pairs, where the auxiliary variable z represents spectral frequency. Each of, say, q peaks is parameterized by its

center frequency f_i , area A_i , and width c_i . There are then $M = 3q$ model parameters $\mathbf{m} = [A_1, f_1, c_1, \dots, A_q, f_q, c_q]^T$. The model is nonlinear and of the form $\mathbf{d} = \mathbf{g}(\mathbf{m})$

$$\begin{aligned} \text{Gaussian: } d_i &= \sum_{j=1}^q \frac{A_j}{(2\pi)^{1/2} c_j} \exp \left[-\frac{(z_i - f_j)^2}{2c_j^2} \right] \\ \text{Lorentzian: } d_i &= \sum_{j=1}^q \frac{A_j c_j^2}{(z_i - f_j)^2 + c_j^2} \end{aligned} \quad (12.30)$$

Note that in the case of the Gaussian, the quantity c_i^2 has the interpretation of variance, but in the case of the Lorentzian (which has infinite variance), it does not. If the data have Gaussian error, it is appropriate to use Newton's method to solve this problem. Furthermore, the problem will typically be overdetermined, at least if $N > M$ and if the peaks do not overlap completely. The equation is linearized around an initial guess using Taylor's theorem, as in Section 9.3. This linearization involves computing a matrix \mathbf{G} of derivatives with rows

$$[\partial g_i / \partial A_1 \quad \partial g_i / \partial f_1 \quad \partial g_i / \partial c_1 \quad \cdots \quad \partial g_i / \partial A_p \quad \partial g_i / \partial f_p \quad \partial g_i / \partial c_p] \quad (12.31)$$

In this problem the Gaussian and Lorentzian models are simple enough for the derivatives to be computed analytically as

Gaussian:

$$\begin{aligned} \partial g_i / \partial A_j &= [1 / (2\pi)^{1/2} c_j] \exp[-(z_i - f_j)^2 / 2c_j^2] \\ \partial g_i / \partial f_j &= \left[\frac{A_j}{(2\pi)^{1/2} c_j} \right] [(z_i - f_j) / c_j^2] \exp[-(z_i - f_j)^2 / 2c_j^2] \\ \partial g_i / \partial c_j &= \left[\frac{A_j}{(2\pi)^{1/2} c_j^2} \right] [((z_i - f_j)^2 / c_j^2) - 1] \exp[-(z_i - f_j)^2 / 2c_j^2] \end{aligned} \quad (12.32)$$

Lorentzian:

$$\begin{aligned} \partial g_i / \partial A_j &= c_j^2 / [(z_i - f_j)^2 + c_j^2] \\ \partial g_i / \partial f_j &= 2A_j c_j^2 (z_i - f_j) / [(z_i - f_j)^2 + c_j^2]^2 \\ \partial g_i / \partial c_j &= 2A_j c_j / [(z_i - f_j)^2 + c_j^2] - 2A_j c_j^3 / [(z_i - f_j)^2 + c_j^2]^2 \end{aligned} \quad (12.33)$$

These derivatives are evaluated at the trial solution $\mathbf{m}^{(p)}$ with the starting value (when $p = 1$) chosen from visual inspection of the data. Improved estimates of the model parameters are found using the recursions $\mathbf{G}\Delta\mathbf{m} = \mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)})$ and $\mathbf{m}^{(p+1)} = \mathbf{m}^{(p)} + \Delta\mathbf{m}$, where the matrix equation is solved with simple least squares. The iterations are terminated when the correction factor $\Delta\mathbf{m}$ becomes negligibly small (for instance, when the absolute value of each component becomes less than some given tolerance). An example is shown in Figure 12.16.

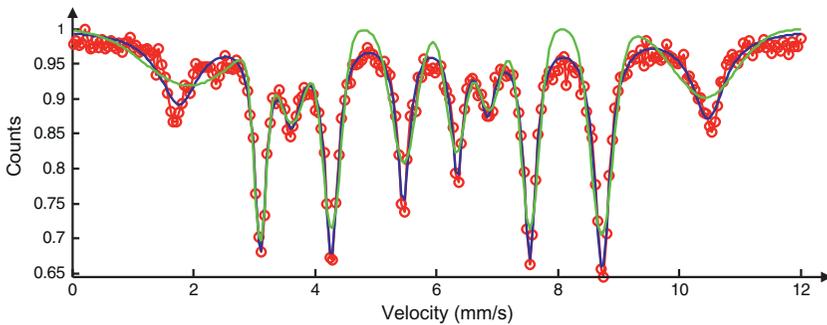


FIGURE 12.16 Example of fitting the sum of 10 Lorentzian (blue) or Normal (green) curves to Mossbauer spectroscopic data (red). The Lorentzian curves are better able to fit the shape of the curve, with the ratio of estimated variances being about 4. An F -test indicates that a null hypothesis that any difference between the two fits can be ascribed to random variation can be rejected to better than 99.99%. Data courtesy of NASA and the University of Mainz. *MatLab* script `gda12_12`.

Occasionally *a priori* information requires that the separation between two peaks be equal to a known value $f_i - f_j = \langle s_{ij} \rangle$. This constraint can be implemented with the linearized version of weighted damped least squares (Equation 9.21c), with a very certain constraint $\mathbf{H}\mathbf{m} = \mathbf{h}$

$$\begin{bmatrix} \mathbf{G}^{(p)} \\ \varepsilon \mathbf{H} \end{bmatrix} \mathbf{m}^{(p+1)} = \begin{bmatrix} \left\{ \mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)}) + \mathbf{G}^{(p)} \mathbf{m}^{(p)} \right\} \\ \varepsilon \mathbf{h} \end{bmatrix} \quad (12.34)$$

$$\mathbf{H} = [\cdots \ 0 \ 1 \ 0 \ \cdots \ 0 \ -1 \ 0 \ \cdots], \quad \text{and} \quad \mathbf{h} = [\langle s_{ij} \rangle]$$

Here, ε is set to a very large number.

One of the drawbacks to these iterative methods is that if the initial guess is too far off, the solution may oscillate wildly or diverge from one iteration to the next. There are several possible remedies for this difficulty. One is to force the perturbation $\Delta \mathbf{m}$ to be less than a given length. This result can be achieved by examining the perturbation on each iteration and, if it is longer than some empirically derived limit, decreasing its length (but not changing its direction). This procedure will prevent the method from wildly “overshooting” the true minimum but will also slow the convergence. Another possibility is to constrain some of the model parameters to equal *a priori* values for the first few iterations, thus allowing only some of the model parameters to vary. The constraints are relaxed after convergence, and the iteration is continued until the unconstrained solution converges.

12.9 EARTHQUAKE LOCATION

When a fault ruptures within the earth, seismic compressional P and shear S waves are emitted. These waves propagate through the earth and are recorded by instruments on the earth’s surface. The earthquake location problem is to

determine the *hypo*center (location, $\mathbf{x}^{(0)} = [x_0, y_0, z_0]^T$) and *origin time* (time of occurrence, t_0) of the rupture on the basis of measurements of the arrival time of the P and S waves at the instruments.

The data are the arrival times t_i^P of P waves and the arrival time t_i^S of S waves, at a set of $i = 1, \dots, N$ receivers (seismometers) located at $\mathbf{x}^{(i)} = [x_i, y_i, z_i]^T$. The model is the arrival time equals the origin time plus the travel time of the wave from the source (geologic fault) to the receiver. In a constant velocity medium, the waves travel along straight-line *rays* connecting source and receiver, so the travel time is the length of the line divided by the P or S wave velocity. In heterogeneous media, the rays are curved (Figure 12.17) and the ray paths and travel times are much more difficult to calculate. We shall not discuss the process of *ray tracing* here, but merely assume that the travel times t_i^P and t_i^S can be calculated for arbitrary source and receiver locations. Then

$$t_i^P = T_i^P(\mathbf{x}^{(0)}, \mathbf{x}^{(i)}) + t_0 \quad \text{and} \quad t_i^S = T_i^S(\mathbf{x}^{(0)}, \mathbf{x}^{(i)}) + t_0 \quad (12.35)$$

These equations are nonlinear and of the form $\mathbf{d} = \mathbf{g}(\mathbf{m})$. If many observations are made so that the equations are overdetermined, an iterative least squares approach may be tried. This method requires that the derivatives $\nabla T = [\partial T_i / \partial x_0, \partial T_i / \partial y_0, \partial T_i / \partial z_0]^T$ be computed for various locations of the source. Unfortunately, there is no simple, differentiable analytic formula for travel time. One possible solution is to calculate this derivative numerically using the finite difference formula. For the P wave, we find

$$\begin{aligned} \frac{\partial T_i^P}{\partial x_0} &\approx \frac{T_i^P(x_0 + \Delta x, y_0, z_0, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta x} \\ \frac{\partial T_i^P}{\partial y_0} &\approx \frac{T_i^P(x_0, y_0 + \Delta y, z_0, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta y} \\ \frac{\partial T_i^P}{\partial z_0} &\approx \frac{T_i^P(x_0, y_0, z_0 + \Delta z, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta z} \end{aligned} \quad (12.36)$$

and similarly for the S wave. Here, Δx , Δy , and Δz are small distance increments. These equations represent moving the location of the earthquake a small

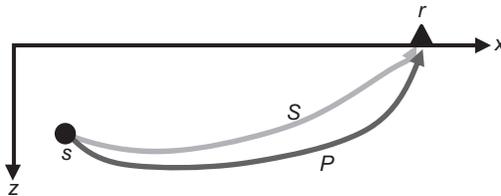


FIGURE 12.17 Compressional P and shear S waves travel along rays from earthquake source s (circle) to receiver r (triangle).

distance Δx along the directions of the coordinate axes and then computing the change in travel time. This approach has two disadvantages. First, if Δx is made very small so that the finite difference approximates a derivative very closely, the terms in the numerator become nearly equal and computer round-off error can become very significant. Second, this method requires that the travel time be computed for three additional earthquake locations and, therefore, is $4 \times$ as expensive as computing travel time alone.

In some inverse problems, there is no alternative but to use finite element derivatives. Fortunately, it is possible in this problem to deduce the gradient of travel time by examining the geometry of a ray as it leaves the source (Figure 12.18). If the earthquake is moved a small distance s parallel to the ray in the direction of the receiver, then the travel time is simply decreased by an amount s/v_0 , where $v_0 = v(\mathbf{x}^{(0)})$ is the P or S wave velocity at the source. If it is moved a small distance perpendicular to the ray, then the change in travel time is negligible since the new ray path will have nearly the same length as the old. The gradient is therefore $\nabla T = -\mathbf{s}/v_0$, where \mathbf{s} is a unit vector tangent to the ray at the source that points toward the receiver. This insight is due to Geiger (1912) and linearized earthquake location is often referred to as *Geiger's method*. Since we must calculate the ray path to find the travel time, no extra computational effort is required to find the gradient. If $\mathbf{s}^{(i)P}$ is the direction of the P wave ray to the i th receiver (of a total of N receivers), and similarly $\mathbf{s}^{(i)S}$ is for the S wave, the linearized problem is

$$\begin{bmatrix} -s_1^{(1)P}/v_0^P & -s_2^{(1)P}/v_0^P & -s_3^{(1)P}/v_0^P & 1 \\ \dots & \dots & \dots & \dots \\ -s_1^{(N)P}/v_0^P & -s_2^{(N)P}/v_0^P & -s_3^{(N)P}/v_0^P & 1 \\ -s_1^{(1)S}/v_0^S & -s_2^{(1)S}/v_0^S & -s_3^{(1)S}/v_0^S & 1 \\ \dots & \dots & \dots & \dots \\ -s_1^{(N)S}/v_0^S & -s_2^{(N)S}/v_0^S & -s_3^{(N)S}/v_0^S & 1 \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \\ \Delta z_0 \\ t_0 \end{bmatrix} = \begin{bmatrix} t_1^P - T_1^P \\ \dots \\ t_N^P - T_N^P \\ t_1^S - T_1^S \\ \dots \\ t_N^S - T_N^S \end{bmatrix} \quad (12.37)$$

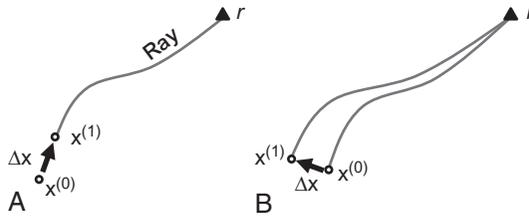


FIGURE 12.18 (A) Moving the source a distance Δx from $\mathbf{x}^{(0)}$ to $\mathbf{x}^{(1)}$ parallel to the ray path leads to a large change in travel time. (B) Moving the source perpendicular to the ray path leads to no (first order) change in travel time. The partial derivative of travel time with respect to distance can therefore be determined with minimal extra effort.

This equation is then solved iteratively using Newton's method. A sample problem is shown in Figure 12.19.

There are some instances in which the matrix can become underdetermined and the least squares method will fail. This possibility is especially likely if the problem contains only P wave arrival times. The matrix equation consists of only the top half of Equation (12.37). If all the rays leave the source in such a manner that one or more components of their unit vectors are all equal, then the corresponding column of the matrix will be proportional to the vector $[1, 1, 1, \dots, 1]^T$ and will therefore be linearly dependent on the fourth column of the matrix. The earthquake location is then nonunique and can be traded off with origin time (Figure 12.20). This problem occurs when the earthquake is far from all of the stations. The addition of S wave arrival times resolves the underdeterminacy (the columns are then proportional to $[1, 1, 1, \dots, 1, v_S/v_P, \dots]^T$ and are not linearly dependent on the fourth column). It is therefore wise to use singular-value decomposition and the natural inverse to solve earthquake location problems, permitting easy identification of underdetermined cases.

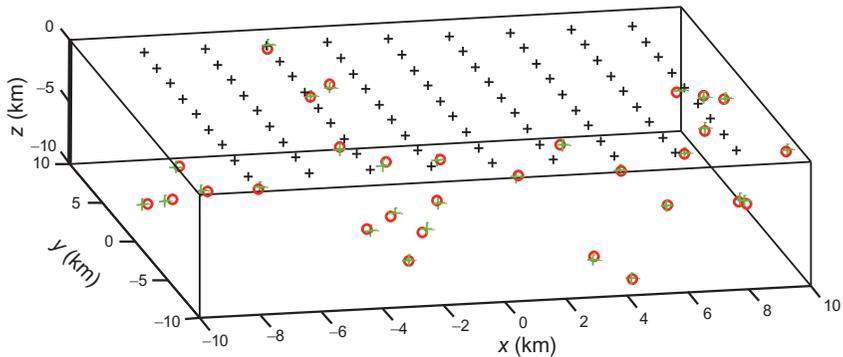


FIGURE 12.19 Earthquake location example. Arrival times of P and S waves from earthquakes (red circles) are recorded on an array of 81 stations (black crosses). The observed arrival times include random noise with variance $\sigma_d^2 = (0.1)^2$ s. The estimated locations (green crosses) are computed using Geiger's method. *MatLab* script gda12_13.

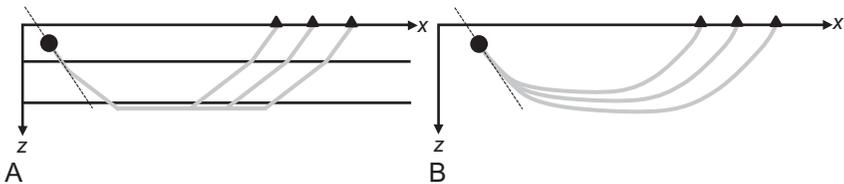


FIGURE 12.20 (A) In layered media, rays follow "refracted" paths, such that the rays to all receivers (triangles) leave the source (circle) at the same angle. The position and travel time of the source on the dashed line therefore trade off. (B) This phenomenon also occurs in nonlayered media if the source is far from the receivers.

Another problem can occur if all the stations are on the horizontal plane. Then the solution is nonunique, with the solution $[x, y, z, t_0]^T$ and its mirror image $[x, y, -z, t_0]^T$ having exactly the same error. This problem can be avoided by including *a priori* information that the earthquake occur beneath the ground and not in the air.

The earthquake location problem can also be extended to locate the earthquake and determine the velocity structure of the medium simultaneously. It then becomes similar to the tomography problem (Section 12.4) except that the orientations of the ray paths are unknown. To ensure that the medium is crossed by a sufficient number of rays to resolve the velocity structure, one must locate simultaneously a large set of earthquakes—on the order of 100 when the velocity structure is assumed to be one dimensional (Crosson, 1976) but thousands to millions when it is fully three dimensional.

12.10 VIBRATIONAL PROBLEMS

There are many inverse problems that involve determining the structure of an object from measurements of its *characteristic frequencies* of vibration (*eigenfrequencies*). For instance, the solar and terrestrial vibrational frequencies can be inverted for the density and elastic structure of those two bodies.

The forward problem of calculating the *modes* (spatial patterns) and eigenfrequencies of vibration of a body of a given structure is quite formidable. A commonly used approximate method is based on *perturbation theory*. The modes and eigenfrequencies are first computed for a simple *unperturbed* structure and then they are *perturbed* to match a more complicated structure. Consider, for instance, an acoustic problem in which the pressure modes $p_n(\mathbf{x})$ and corresponding eigenfrequencies ω_n satisfy the differential equation

$$-\omega_n^2 p_n(\mathbf{x}) = v^2(\mathbf{x}) \nabla^2 p_n(\mathbf{x}) \quad (12.38)$$

with homogeneous boundary conditions. Many properties of the solutions to this equation can be deduced without actually solving it. For instance, two modes p_n and p_m can be shown to obey the orthogonality relationship

$$\int p_n(\mathbf{x}) p_m(\mathbf{x}) v^{-2}(\mathbf{x}) d^3x = \delta_{nm} \quad (12.39)$$

(where δ_{nm} is the Kronecker delta) as long as their eigenfrequencies are different; that is, $\omega_n \neq \omega_m$. Actually determining the modes and characteristic frequencies is a difficult problem for an arbitrary complicated velocity $v(\mathbf{x})$. Suppose, however, that velocity can be written as

$$v(\mathbf{x}) = v^{(0)}(\mathbf{x}) + \varepsilon v^{(1)}(\mathbf{x}) \quad (12.40)$$

where ε is a small number. Furthermore, suppose that the modes $p_n^{(0)}(\mathbf{x})$ and eigenfrequencies $\omega_n^{(0)}$ of the *unperturbed problem* (that is, with $\varepsilon=0$) are

known. The idea of perturbation theory is to determine approximate versions of the *perturbed* modes $p_n(\mathbf{x})$ and eigenfrequencies ω_n that are valid for small ε .

The special case where all the eigenfrequencies are distinct (that is, with numerically different values) is easiest (and the only one we solve here). We first assume that the eigenfrequencies and perturbed modes can be written as series in powers in ε

$$\begin{aligned}\omega_n &= \omega_n^{(0)} + \varepsilon\omega_n^{(1)} + \varepsilon^2\omega_n^{(2)} + \dots \quad \text{and} \\ p_n(\mathbf{x}) &= p_n^{(0)}(\mathbf{x}) + \varepsilon p_n^{(1)}(\mathbf{x}) + \varepsilon^2 p_n^{(2)}(\mathbf{x}) + \dots\end{aligned}\quad (12.41)$$

Here, $\omega_n^{(i)}$ (for $i > 0$) are unknown constants and $p_n^{(i)}$ (for $i > 0$) are unknown functions. When ε is small, it suffices to solve merely for $\omega_n^{(1)}$ and $p_n^{(1)}$. We first represent $p_n^{(1)}$ as a sum of the other unperturbed modes

$$p_m^{(1)} = \sum_{\substack{m \\ \omega_m \neq \omega_n}}^{\infty} b_{nm} p_m^{(0)} \quad (12.42)$$

where b_{nm} are unknown coefficients. After substituting the Equations (12.41) and (12.42) into Equation (12.38), equating terms of equal power in ε , and applying the orthogonality relationship, the first order quantities can be shown to be (see, for example, Menke and Abbott, 1990, their Section 8.6.7)

$$\begin{aligned}\omega_n^{(1)} &= \omega_n^{(0)} \int [p_n^{(0)}(\mathbf{x})]^2 [v^{(0)}(\mathbf{x})]^{-3} v^{(1)}(\mathbf{x}) d^3x \\ b_{nm} &= \frac{2(\omega_m^{(0)})^2}{(\omega_m^{(0)})^2 - (\omega_n^{(0)})^2} \int p_n^{(0)}(\mathbf{x}) p_m^{(0)}(\mathbf{x}) [v^{(0)}(\mathbf{x})]^{-3} v^{(1)}(\mathbf{x}) d^3x\end{aligned}\quad (12.43)$$

The equation for $\omega_n^{(1)}$ is the relevant one for the discussion of the inverse problem. Notice that it is a linear function of the velocity perturbation $v^{(1)}(\mathbf{x})$,

$$\omega_n^{(1)} = \int G_n(\mathbf{x}) v^{(1)}(\mathbf{x}) d^3x \quad \text{with} \quad G_n(\mathbf{x}) = \omega_n^{(0)} [p_n^{(0)}(\mathbf{x})]^2 [v^{(0)}(\mathbf{x})]^{-3} \quad (12.44)$$

and so the problem of determining $v^{(1)}(\mathbf{x})$ from measurements of $\omega_n^{(1)}$ is a problem of linearized continuous inverse theory.

As an example, we will consider a one-dimensional organ pipe of length h , open at one end and closed at the other (corresponding to the boundary conditions $p|_{x=0} = dp/dx|_{x=h} = 0$). We consider the unperturbed problem of a constant velocity structure, which has modes and eigenfrequencies given by

$$\begin{aligned}p_n^{(0)}(x) &= \frac{2[v^{(0)}]^2}{h} \sin\left\{\frac{(n-1/2)\pi}{h}x\right\} \quad \text{and} \\ \omega_n^{(0)} &= \frac{\pi(n-1/2)v^{(0)}}{h} \quad \text{with} \quad n = 1, 2, 3, \dots\end{aligned}\quad (12.45)$$

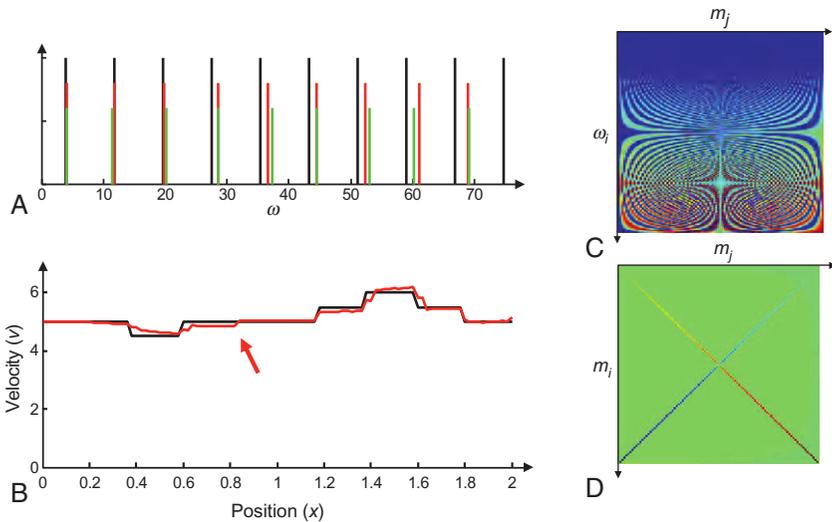


FIGURE 12.21 Organ pipe example. (A) Ladder diagram for unperturbed (black), true (red), and observed (green) eigenfrequencies of the organ pipe. (B) True (black) and estimated (red) velocity structure. Arrow points to an artifact in the estimated solution. (C) Data kernel G . (D) Model resolution matrix R . *MatLab* script gda12_14.

Note that the eigenfrequencies are evenly spaced in frequency (Figure 12.21A). We now imagine that we measure the first N eigenfrequencies ω_n of an organ pipe whose velocity structure $v(x)$ is unknown (Figure 12.21A). The data are the deviations of these frequencies from those of the unperturbed problem, $d_n = \omega_n^{(1)} - \omega_n^{(0)}$. The model parameters are the continuous function $m(x) = v^{(1)}(x) - v^{(0)}$, which we will discretize into a vector \mathbf{m} by sampling it at M evenly spaced values, with spacing Δx . The data kernel in the discrete equation $\mathbf{d} = \mathbf{G}\mathbf{m}$ is the discrete version of Equation (12.44) (Figure 12.21C)

$$G_{nm} = G_n(x_m)\Delta x = \omega_n^{(0)}[p_n^{(0)}(x_m)]^2[v^{(0)}]^{-3}\Delta x \quad (12.46)$$

The equation can be solved using damped least squares. Superficially, the estimated solution (red curve in Figure 12.21B) looks reasonably good, except that it seems to have small steps (arrow in Figure 12.21B) in locations where the true solution is smooth. However, a close inspection of the model resolution matrix (Figure 12.21D), which is X-shaped, indicates a serious problem: two widely separated model parameters are trading off. The steps in the estimated solution are artifacts, steps from elsewhere in the model that are mapped into the wrong position. This is an inherent nonuniqueness of this and many other eigenfrequency-type problems. In seismology, it is addressed by combining eigenfrequency data with other data types (for example, travel time measurements along rays) that do not suffer from this nonuniqueness.

12.11 PROBLEMS

- 12.1** Modify the *MatLab* script for the airgun problem to compute a shorter filter $m(t)$. How short can it be and still do a reasonable job at spiking the airgun pulse?
- 12.2** Modify the *MatLab* script for the tomography problem so that it retains receivers on the top and bottom of the model but omits them on the sides. Interpret the results.
- 12.3** Modify the *MatLab* script for the L_1 straight line problem to allow for data of different accuracy. Test it against synthetic data, where the first $N/2$ data have variance $(\sigma^L)^2$ and the last $N/2$ have variance $(\sigma^R)^2$, both for the case where $\sigma^L = 10 \sigma^R$ and $\sigma^R = 10 \sigma^L$.
- 12.4** Modify the *MatLab* script for the earthquake location problem to include *a priori* information that all the earthquakes occur near depth z_A . Adjust the true locations of the earthquakes to reflect this information. [Hint: You will need to use Equation (9.21c) to implement the *a priori* information. It can be solved using `big()` with the weighted least squares function `weightedleast_squaresfcn()`.]
- 12.5** Modify the *MatLab* script for the earthquake location problem to use *differential P wave travel time data*, but no absolute travel time data. Each of these new data is the time difference between the arrival times of two earthquakes observed at a common station. Compare the locations to the results of the absolute arrival time script, in the case where the variance of the differential data is four orders of magnitude smaller than the variance of the absolute data. [Hint: You may have to use a better starting guess than was used in the absolute-arrival time script.]

REFERENCES

- Crosson, R.S., 1976. Crustal structure modeling of earthquake data: 1. Simultaneous least squares estimation of hypocenter and velocity parameters. *J. Geophys. Res.* 81, 3036–3046.
- Fisher, R.A., 1953. Dispersion on a sphere. *Phil. Trans. R. Soc. Lond., A* 217, 295–305.
- Geiger, L., 1912. Probability method for the determination of earthquake epicenters from the arrival time only (translated from Geiger's 1910 German article). *Bull. St. Louis University* 8 (1), 56–71.
- Kaula, W.M., 1966. *Theory of Satellite Geodesy*. Ginn (Blaisdel), Boston, Massachusetts.
- Menke, W., Abbott, D., 1990. *Geophysical Theory*. Columbia University Press, New York, 457pp.
- Smith, S.G., 1975. Measurement of airgun waveforms. *Geophys. J. R. Astr. Soc.* 42, 273–280.

Applications of Inverse Theory to Solid Earth Geophysics

13.1 EARTHQUAKE LOCATION AND DETERMINATION OF THE VELOCITY STRUCTURE OF THE EARTH FROM TRAVEL TIME DATA

The problem of determining the *hypocentral* parameters of an earthquake (that is, its location and its origin time) and the problem of determining the velocity structure of the earth from travel time data are very closely coupled in geophysics, because earthquakes, which occur naturally at initially unknown locations, are a primary source of structural information.

When the wavelength of the seismic waves is smaller than the scale of the velocity heterogeneities, the propagation of seismic waves can be described with *ray theory*, an approximation in which energy is assumed to propagate from source to observer along a curved path called a *ray*. In this approximation, the wave field is completely determined by the pattern of rays and the travel time along them.

In areas where the velocity structure is already known, earthquakes can be located using Geiger's (1912) method (see Section 12.9). The travel time $T(\mathbf{x})$ of an earthquake with location \mathbf{x} and its ray tangent $\mathbf{t}(\mathbf{x})$ (pointing from earthquake to receiver) is calculated using ray theory. The arrival time is $t = \tau + T(\mathbf{x})$, where τ is the origin time of the earthquake. The perturbation in arrival time due to a change in the earthquake location from \mathbf{x}_0 to $\mathbf{x}_0 + \delta\mathbf{x}$ and a change in the origin time from τ_0 to $\tau_0 + \delta\tau$ is

$$\delta t = t - [\tau_0 + T(\mathbf{x}_0)] \approx \delta\tau - s(\mathbf{x}_0)\mathbf{t}(\mathbf{x}_0) \cdot \delta\mathbf{x} \quad (13.1)$$

where s is the slowness (reciprocal velocity). Note that this linearized equation has four model parameters, τ , and the three components of \mathbf{x} .

The forward problem, that of finding the ray path connecting earthquake and station and its traveltime, is computationally intensive. Two alternative strategies have been put forward, one based on first finding the ray path using a shooting or bending strategy and then calculating the travel time by an integral of slowness

along the ray (Cerveny, 2001), or conversely, by first finding the traveltime by solving its partial differential equation (the *Eikonal* equation) and then calculating the ray path by taking the gradient of the travel time (Vidale, 1990).

The velocity structure can be represented with varying degrees of complexity. The simplest is a vertically stratified structure consisting of a stack of homogeneous layers or, alternatively, a continuously varying function of depth. Analytic formula for the travel time and its derivative with source parameters can be derived in some cases (Aki and Richards, 2002, their Section 9.3). The most complicated cases are fully three-dimensional velocity models represented with voxels or splines. Compromises between these extremes are also popular and include two-dimensional models (in cases where the structure is assumed to be uniform along the strike of a linear tectonic feature) and regionalized models (which assign a different vertically stratified structure to each tectonically distinct region but average them when computing travel times of rays that cross from one region to another).

In the simplest formulation, each earthquake is located separately. The data kernel of the linearized problem is an $N \times 4$ matrix, where N is the number of travel time observations, and is typically solved by singular-value decomposition. A very commonly used computer program that uses a layered velocity model is HYPOINVERSE (Klein, 1985). However, locations can often be improved by simultaneously solving for at least some aspect of the velocity model, which requires the earthquakes to be located simultaneously. The simplest modification is to solve for a *station correction* for each station; that is, a time increment that can be added to predicted travel times to account for near-station velocity heterogeneity not captured by the velocity model. More complicated formulations solve for a vertically stratified structure (Crosson, 1976) and for fully three-dimensional models (Menke, 2005; Zelt, 1998; Zelt and Barton, 1998). In the three-dimensional case, the inverse problem is properly one of tomographic inversion, as the perturbation in arrival time δt includes a line integral

$$\delta t \approx \delta \tau - s(\mathbf{x}_0)\mathbf{t}(\mathbf{x}_0) \cdot \delta \mathbf{x} + \int_{\text{unperturbed ray}} \delta s \, d\ell \quad (13.2)$$

Here, δs is the perturbation in slowness with respect to the reference model and $d\ell$ is the arc-length along the unperturbed ray. This problem is $N \times (4K + L)$, where N is the number of arrival time observations, K is the number of earthquakes, and L is the number of unknown parameters in the velocity model. One of the limitations of earthquake data is that earthquakes tend to be spatially clustered. Thus, the ray paths often poorly sample the model volume, and *a priori* information, usually in the form of smoothness information, is required to achieve useful solutions. Even so, earthquake location (and especially earthquake depth) will often trade off strongly with velocity structure.

A very useful algorithm due to Pavlis and Booker (1980) simplifies the inversion of the very large matrices that result from simultaneously inverting for the source parameter of many thousands of earthquakes and an even larger

number of velocity model parameters. Suppose that the linearized problem has been converted into a standard discrete linear problem $\mathbf{d} = \mathbf{G}\mathbf{m}$, where the model parameters can be arranged into two groups $\mathbf{m} = [\mathbf{m}_1, \mathbf{m}_2]^T$, where \mathbf{m}_1 is a vector of the earthquake source parameters and \mathbf{m}_2 is a vector of the velocity parameters. Then the inverse problem can be written

$$\mathbf{d} = \mathbf{G}\mathbf{m} = [\mathbf{G}_1 \ \mathbf{G}_2] \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \end{bmatrix} = \mathbf{G}_1\mathbf{m}_1 + \mathbf{G}_2\mathbf{m}_2 \quad (13.3)$$

Now suppose that \mathbf{G}_1 has singular-value decomposition $\mathbf{G}_1 = \mathbf{U}_{1p} \Lambda_{1p} \mathbf{V}_{1p}^T$, with p nonzero singular values, so that $\mathbf{U}_1 = [\mathbf{U}_{1p}, \mathbf{U}_{10}]$ where $\mathbf{U}_{10}^T \mathbf{U}_{1p} = 0$. Premultiplying the inverse problem (Equation 13.3) by \mathbf{U}_{10}^T yields

$$\begin{aligned} \mathbf{U}_{10}^T \mathbf{d} &= \mathbf{U}_{10}^T \mathbf{G}_1 \mathbf{m}_1 + \mathbf{U}_{10}^T \mathbf{G}_2 \mathbf{m}_2 = \mathbf{U}_{10}^T \mathbf{G}_2 \mathbf{m}_2 \\ \text{or } \mathbf{d}' &= \mathbf{G}' \mathbf{m}_2 \quad \text{with } \mathbf{d}' = \mathbf{U}_{10}^T \mathbf{d} \quad \text{and } \mathbf{G}' = \mathbf{U}_{10}^T \mathbf{G}_2 \end{aligned} \quad (13.4)$$

Here, we have used the fact that $\mathbf{U}_{10}^T \mathbf{G}_1 \mathbf{m}_1 = \mathbf{U}_{10}^T \mathbf{U}_{1p} \Lambda_{1p} \mathbf{V}_{1p}^T \mathbf{m}_1 = 0$. Premultiplying the inverse problem (Equation 13.3) by \mathbf{U}_{1p}^T yields

$$\begin{aligned} \mathbf{U}_{1p}^T \mathbf{d} &= \mathbf{U}_{1p}^T \mathbf{G}_1 \mathbf{m}_1 + \mathbf{U}_{1p}^T \mathbf{G}_2 \mathbf{m}_2 = \Lambda_{1p} \mathbf{V}_{1p}^T \mathbf{m}_1 + \mathbf{U}_{1p}^T \mathbf{G}_2 \mathbf{m}_2 \\ \text{or } \mathbf{d}'' &= \mathbf{G}'' \mathbf{m}_1 \quad \text{with } \mathbf{d}'' = \mathbf{U}_{1p}^T \mathbf{d} - \mathbf{U}_{1p}^T \mathbf{G}_2 \mathbf{m}_2 \quad \text{and } \mathbf{G}'' = \Lambda_{1p} \mathbf{V}_{1p}^T \end{aligned} \quad (13.5)$$

The inverse problem has been partitioned into two equations, an equation for \mathbf{m}_2 (the velocity parameters) that can be solved first, and an equation for \mathbf{m}_1 (the source parameters) that can be solved once \mathbf{m}_2 is known (see *MatLab* script `gda13_01`). When the data consist of only absolute travel times (as contrasted to differential travel times, described below), the source parameter data kernel \mathbf{G}_1 is block diagonal, since the source parameters of a given earthquake affect only the travel times associated with that earthquake. The problem of computing the singular-value decomposition of \mathbf{G}_1 can be broken down into the problem of computing the singular-value decomposition of the sub-matrices.

Signal processing techniques based on waveform cross-correlation are able to determine the difference $\Delta t^{A,B} = t^A - t^B$ in arrival times of two earthquakes (say, labeled A and B), observed at a common station, to several orders of magnitude better precision than the absolute arrival times t^A and t^B can be determined. Thus, earthquake locations can be vastly improved, either by supplementing absolute arrival time data t with differential arrival time data $\Delta t^{A,B}$ or by relying on differential data alone. The perturbation in differential arrival time for two earthquakes, A and B, is formed by differencing two versions of Equation (13.1)

$$\delta \Delta t^{A,B} \approx \delta \tau^A - s(\mathbf{x}_0^A) \mathbf{t}(\mathbf{x}_0^A) \cdot \delta \mathbf{x}^A - \delta \tau^B + s(\mathbf{x}_0^B) \mathbf{t}(\mathbf{x}_0^B) \cdot \delta \mathbf{x}^B \quad (13.6)$$

Each equation involves eight unknowns. Early versions of this method focused on using differential data to improve the *relative* location between earthquakes (e.g., [Spence and Alexander, 1968](#)). The modern formulation, known as the

double-difference method, acknowledges that the relative locations are better determined than the absolute locations but solves for both (e.g., [Menke and Schaff, 2004](#); [Slunga et al., 1995](#); [Waldhauser and Ellsworth, 2000](#)). Double-difference methods have also been extended to include the tomographic estimation of velocity structure ([Zhang and Thurber, 2003](#)).

At the other extreme is the case where the origin times and locations of the seismic sources are known and the travel time T of elastic waves through the earth is used to determine the slowness $s(\mathbf{x})$ in the earth

$$\delta T = \int_{\text{perturbed ray}} \delta s \, d\ell \approx \int_{\text{unperturbed ray}} \delta s \, d\ell \quad (13.7)$$

The approximation relies on *Fermat's Principle*, which states that perturbations in ray path cause only second-order changes in traveltimes. Tomographic inversion based on [Equation \(13.7\)](#) is now a standard tool in seismology and has been used to image the earth at many scales, from kilometer-scale geologic structures to the whole earth. An alternative to model estimation is presented by [Vasco \(1986\)](#) who uses extremal inversion to determine quantities such as the maximum difference between the velocities in two different parts of the model, assuming that the velocity perturbation is everywhere within a given set of bounds (see [Chapter 6](#)).

Tomographic inversions are often performed separately from earthquake location but then are degraded by whatever error is present in the earthquake locations and origin times. In the case where the earthquakes are all very distant from the volume being imaged, the largest error arises from the uncertainty in origin time, since errors in location do not change the part of the ray path in the imaged volume by very much. Suppose that the slowness in the imaged volume is represented as $s(x,y,z) = s_1(z) + s_2(x,y,z)$, with (x,y) horizontal and z vertical position, and with the mean of $s_2(x,y,z)$ zero at every depth, so that it represents horizontal variations in velocity, only. Poor knowledge of origin times affects only the estimates of $s_1(z)$ and not $s_2(x,y,z)$ ([Aki et al., 1976](#); [Menke et al., 2006](#)). Estimates of the horizontal variations in velocity contain important information about the geologic structures that are being imaged, such as their (x,y) location and dip direction. Consequently, the method, called *teleseismic tomography*, has been applied in many areas of the world.

13.2 MOMENT TENSORS OF EARTHQUAKES

Seismometers measure the motion or *displacement* of the ground, a three-component vector $\mathbf{u}(\mathbf{x},t)$ that is a function of position \mathbf{x} and time t . At wavelengths longer than the spatial scale of a seismic source, such as a geologic fault, an earthquake can be approximated by a system of nine *force-couples* acting at a point \mathbf{x}_0 , the *centroid* of the source. The amplitude of these force-couples is described by a 3×3 symmetric matrix $\mathbf{M}(t)$, called the *moment tensor*, which is a function of time, t ([Aki and Richards, 2002](#), their [Section 3.3](#)). Ground displacement depends on the time derivative $d\mathbf{M}/dt$, called the *moment-rate tensor*.

Its six independent elements constitute six continuous model parameters $m_i(t)$. The ground displacement at (\mathbf{x}, t) due to a set of force-couples at \mathbf{x}_0 is linearly proportional to the elements of the moment-rate tensor via

$$u_i^{\text{pre}}(\mathbf{x}, t) = \sum_{j=1}^6 \int G_{ij}(\mathbf{x}, \mathbf{x}_0, t - t_0) m_j(t_0) dt_0 \quad (13.8)$$

Here, $G_{ij}(\mathbf{x}, \mathbf{x}_0, t - t_0)$ are data kernels that describe the i th component displacement at (\mathbf{x}, t) due to the j th force couple at (\mathbf{x}_0, t_0) . The data kernels depend upon earth structure and, though challenging to calculate, are usually assumed to be known. If the location \mathbf{x}_0 of the source is also known, then the problem of estimating the moment-rate tensor from observations of ground displacement is completely linear (Dziewonski and Woodhouse, 1983; Dziewonski et al., 1981). The time variability of the moment-rate tensor is usually parameterized, with a triangular function of fixed width (or several overlapping such functions) being popular, so that the total number of unknowns is $M = 6L$, where L is the number of triangles. The error $e_i^{(j)}(t_k)$ associated with component i , observation location j , and observation time k is then

$$e_i^{(j)}(t_k) = u_i^{\text{obs}}(\mathbf{x}^{(j)}, t_k) - u_i^{\text{pre}}(\mathbf{x}^{(j)}, t_k) \quad (13.9)$$

When the total error E is defined as the usual L_2 prediction error

$$E = \sum_i \sum_j \sum_k [e_i^{(j)}(t_k)]^2 \quad (13.10)$$

the problem can be solved with simple least squares. Most natural sources, such as earthquakes, are thought to occur without causing volume changes, so some authors add the linear constraint $\dot{M}_{11} + \dot{M}_{22} + \dot{M}_{33} = 0$ (at all times).

Sometimes, the location $\mathbf{x}_0 = [x_0, y_0, z_0]^T$ of the source (or just its depth z_0) is also assumed to be unknown. The inverse problem is then nonlinear. This problem can be solved using a grid search over source location, that is, by solving the linear inverse problem for a grid of fixed source locations and then choosing the one with smallest total error E . The Fréchet derivatives of E with respect to hypocentral parameters are known; so alternately, Newton's method can be used. Adjoint methods are also applicable to this problem (Kim et al., 2011).

Earthquake locations are now routinely calculated by the U.S. Geological Survey (<http://earthquake.usgs.gov/earthquakes/recenteqsww>) and seismic moment tensors by the Global Centroid Moment Tensor (CMT) Project (<http://www.globalcmt.org>). They are standard data sets used in earthquake and tectonic research.

13.3 WAVEFORM "TOMOGRAPHY"

The ray approximation used in Section 13.1 is valid only when the wavelength of the elastic waves is much smaller than the spatial scale of the velocity heterogeneities. This condition is violated for "long-period" seismic waves, which

have wavelengths of hundreds of kilometers—comparable with the spatial scales of many of the earth’s most significant heterogeneities, including the lithosphere and asthenosphere. In this case, the displacement field must be computed by solving its partial differential equation, a much more challenging computation than those of ray theory. The inverse problem is to find the set of the earth’s material constants (Lamé parameter λ , shear modulus μ , and density ρ) that minimizes some measure of the misfit between the observed and predicted displacement, as observed at set of N locations $\mathbf{x}^{(j)}$. The simplest misfit is simply the L_2 difference between the observed and predicted wave field

$$E = \sum_{i=1}^3 \sum_{j=1}^N \int_{-\infty}^{+\infty} [u_i^{\text{obs}}(\mathbf{x}^{(j)}, t) - u_i^{\text{pre}}(\mathbf{x}^{(j)}, t)]^2 dt \quad (13.11)$$

The Fréchet derivatives $\delta u/\delta \lambda$, $\delta u/\delta \mu$, and $\delta u/\delta \rho$ (and also $\delta E/\delta \lambda$, $\delta E/\delta \mu$, and $\delta E/\delta \rho$) can be computed via adjoint methods, as described in [Section 11.10 \(Dahlen et al., 2000; Tromp et al., 2005\)](#). These derivatives are fully three-dimensional, in the sense that a perturbation in material constants anywhere in the model will perturb the wave field at every observer. This is in contrast to the ray-theoretical approximation, where the effect is limited to perturbations located along the ray path connecting source and receiver. Thus, waveform “tomography” is not tomography in the strict sense; nevertheless, the term is commonly used.

At intermediate periods, seismic wave fields often appear to contain pulse-like arrivals (or *phases*), which are reminiscent of ray-like behavior. Ray theory correctly predicts that energy has traveled along a volumetrically restricted path but does not correctly predict the details. The ray-theoretical notion that a phase has a travel time is still very powerful, because it captures the underlying physics that the main effect of changing the slowness along the path is to advance or delay the arrival of the wave. A differential travel time ΔT can be defined by comparing time windows of the observed and predicted wave fields, centered about a particular phase, and determining whether one is advanced or delayed with respect to the other. These differential travel times constitute a new type of data. While they contain less information than the wave field itself, they are more robust.

Cross-correlation is typically used to define the differential travel time between an observed and predicted phase

$$\Delta T = \arg \max_{\tau} c(\tau) \quad \text{where} \quad c(\tau) = \int_{t_1}^{t_2} u^{\text{pre}}(t - \tau) u^{\text{obs}}(t) dt \quad (13.12)$$

Here, the phase is assumed to arrive in the time window $t_1 < t < t_2$. The travel time difference ΔT (positive when the observed signal is delayed with respect to the predicted signal) is the time lag τ at which the cross-correlation $c(\tau)$ attains its maximum, that is, the time lag at which the two signals best align. Suppose now that the predicted wave field is perturbed by a small amount $\delta u(t)$, so that

$u^{\text{pre}}(t) = u_0(t) + \delta u(t)$. Marquering et al. (1999) show that the resulting change in differential travel time $\Delta T = \Delta T_0 + \delta T$ is

$$\delta T = \frac{1}{N} \int_{t_1}^{t_2} \dot{u}_0(t) \delta u(t) dt \quad \text{with} \quad N = \int_{t_1}^{t_2} \ddot{u}_0(t) u_0(t) dt \quad (13.13)$$

Here, the dot denotes differentiation in time. If we define a *window function* $W(t)$ which is unity within a time interval $t_1 < t < t_2$ that encloses a particular seismic arrival, such as the P wave, and zero outside it, then

$$\delta T = (h(t), \delta u(t))$$

with $h(t) = \frac{1}{N} W(t) \dot{u}_0(t)$ and $N = \int_{-\infty}^{+\infty} W(t) \ddot{u}_0(t) u_0(t) dt$ (13.14)

Hence, the travel time data are related to the perturbed displacement field by an equation of the form 11.58, for which we have already derived a method for calculating the Fréchet derivative.

As the period of the seismic waves is decreased, the high-amplitude part of the travel time derivative becomes more and more concentrated along the ray-theoretical path, taking on a curved shape reminiscent of a banana. Counter-intuitively, although the derivative's amplitude is large *near* the ray path, it is exactly zero *on* the ray path. This behavior is due to the Fréchet derivative lacking an intrinsic length scale. A heterogeneity with an infinitesimal diameter can perturb the travel time only to the extent that the travel time from source to heterogeneity to observer is different than the travel time from source to receiver. For heterogeneities located along the ray path, the two travel times are exactly the same and the derivative is zero. In cross-section, the banana has a zero at its center; that is, it looks like a doughnut—hence the term *banana-doughnut* kernel. Another counter-intuitive property of these data kernels is that they contain sign reversals. A positive velocity perturbation near the ray path causes (as expected) a travel time advance but can cause a delay at other locations. This is a wave interference effect (Figure 13.1).

13.4 VELOCITY STRUCTURE FROM FREE OSCILLATIONS AND SEISMIC SURFACE WAVES

The earth, being a finite body, can oscillate only with certain characteristic patterns of motion (eigenfunctions) at certain corresponding discrete frequencies ω_{knm} (eigenfrequencies), where the indices (k, n, m) increase with the complexity of the motion. Most commonly, the frequencies of vibration are measured from the spectra of seismograms of very large earthquakes, and the corresponding indices are estimated by comparing the observed frequencies of vibration to those predicted using a reference model of the earth's structure. The inverse problem is to use small differences between the measured and predicted frequencies to improve the reference model.

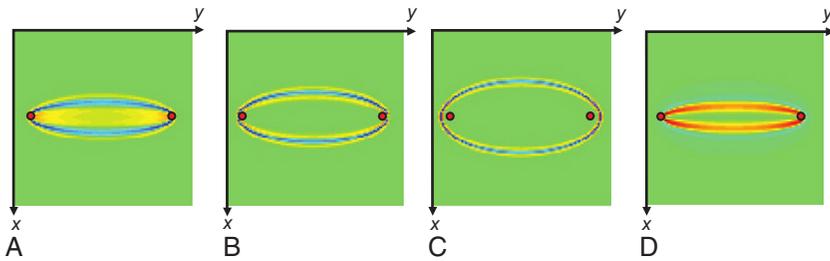


FIGURE 13.1 (A–C) Fréchet derivative $\delta p/\delta m$ of pressure p with respect to $\delta m = 2v_0\delta v$, for an acoustic wave propagation problem with constant background velocity v_0 and velocity perturbation δv , as a function of spatial position (x,y) , and at three different times. Source and receiver, separated by a distance R , are shown as red circles. The time shown in (A) is shortly after the ray-theoretical time $t = R/v_0$, with (B) and (C) corresponding to subsequent times. Note that the region of space containing heterogeneities that can affect the pressure at a given time t expands with time. (D) Fréchet derivative $\delta T/\delta m$ of travel time, called a banana-doughnut kernel. Note that the derivative is zero on the line connecting source and receiver, which corresponds to the geometrical ray path. Note also the sign reversal (blue). *MatLab* script `gda13_01`.

The most fundamental part of this inverse problem is the Fréchet derivatives relating small changes in the earth's material properties to the resultant changes in the frequencies of vibration. These derivatives are computed through the perturbative techniques described in Section 12.10 and are three-dimensional analogs of Equation (12.44) (Takeuchi and Saito, 1972). All involve integrals of the perturbation in earth structure with the eigenfunctions of the reference model, so the problem is one in linearized continuous inverse theory. The complexity of this problem is determined by the assumptions made about the earth model—whether it is isotropic or anisotropic, radially stratified or laterally heterogeneous—and how it is parameterized. A radially stratified inversion that incorporates both body wave travel times and eigenfrequencies and that has become a standard in the seismological community is the Preliminary Reference Earth Model of Dziewonski and Anderson (1981).

Seismic *surface waves*, such as the horizontally polarized *Love* wave and the vertically polarized *Rayleigh* wave, occur when seismic energy is trapped near the earth's surface by the rapid increase in seismic velocities with depth. The velocity of these waves is very strongly frequency dependent, since the lower-frequency waves extend deeper into the earth than the higher-frequency waves and are affected by correspondingly higher velocities. The surface wave caused by an impulsive source, such as an earthquake, is not itself impulsive, but rather is dispersed, since its component frequencies propagate at different phase velocities.

The phase velocity c of the surface wave is defined as $c = \omega(k)/k$, where ω is angular frequency and k is wavenumber (wavenumber is 2π divided by wavelength). The function $c(\omega)$ (or alternatively, $c(k)$) is called the *dispersion function* (or, sometimes, the *dispersion curve*) of the surface wave. It is easily measured from observations of earthquake wave fields and easily predicted for vertically stratified earth models. The inverse problem is to infer the earth's material properties from observations of the dispersion function.

As in the case of free oscillations, Fréchet derivatives can be calculated using a perturbative approach. For Love waves, the perturbation in phase velocity c due to a perturbation in shear modulus μ and density ρ is (Aki and Richards, 2002, their Equation 7.71):

$$\left(\frac{\delta c}{c}\right)_\omega = \frac{\int_0^\infty \left[k^2 l_1^2 + \left(\frac{dl_1}{dz}\right)^2 \right] \delta\mu dz - \int_0^\infty [\omega^2 l_1^2] \delta\rho dz}{2k^2 \int_0^\infty \mu_0 l_1^2 dz} \quad (13.15)$$

Here, shear modulus $\mu(z)$ and density $\rho(z)$ are assumed to be vertically stratified in depth z , and $u_y = l_1(\omega, k, z) \exp(ikx - i\omega t)$ is the horizontal displacement of the Love wave. The corresponding formula for the Rayleigh wave (Aki and Richards, 2002, their Equation 7.78) is similar in form but involves perturbations in three material parameters, Lamé parameter λ , shear modulus μ , and density ρ .

When the earth model contains three-dimensional heterogeneities, the inverse problem of determining earth structure from surface waves is just a type of waveform tomography (see Section 13.3). The dispersion function, now understood to encode the frequency-dependent time shift between an observed and predicted surface wave field, is still a useful quantity. Adjoint methods can be used to compute its Fréchet derivatives (Sieminski et al., 2007).

Ray-theoretical ideas can also be applied to surface waves, which at sufficiently short periods can be approximated as propagating along a horizontal ray path connecting source and receiver (Wang and Dahlen, 1995). Each point on the earth's surface is presumed to have its own dispersion function $c(\omega, x, y)$. The dispersion function $c^{(i)}(\omega)$ observed for a particular source-observer path i is then calculated using the *pure path approximation*

$$\frac{1}{c^{(i)}(\omega)} = \frac{1}{S_i} \int_{\text{path } i} \frac{d\ell}{c(\omega, x, y)} \quad (13.16)$$

Here, S_i is the length of path i . The path is often approximated as the great circle connecting source and receiver. This is a two-dimensional tomography problem relating line integrals of $c(\omega, x, y)$ to observations of $c^{(i)}(\omega)$ —hence the term *surface wave tomography*. The $c(\omega, x, y)$ then can be inverted for estimates of the vertical structure using the vertically stratified method described above (e.g., Boschi and Ekström, 2002). This two-step process is not as accurate as full waveform tomography but is computationally much faster.

13.5 SEISMIC ATTENUATION

Internal friction within the earth causes seismic waves to lose amplitude as they propagate. The amplitude loss can be characterized by the *attenuation factor* $\alpha(\omega, \mathbf{x})$, which can vary with angular frequency ω and position \mathbf{x} . In

a regime where ray theory is accurate, the fractional loss of amplitude A of the seismic wave is

$$\frac{A}{A_0} = \exp \left\{ - \int_{(\text{ray})} \alpha(\omega, x) d\ell \right\} \quad (13.17)$$

Here, A_0 is the initial amplitude. After linearization through taking its logarithm, this equation is a standard tomography problem (identical in form to the X-ray problem described in [Section 1.3.4](#))—hence the term *attenuation tomography*. [Jacobson et al. \(1981\)](#) discuss its solution when the attenuation is assumed to vary only with depth. [Dalton and Ekström \(2006\)](#) perform a global two-dimensional tomographic inversion using surface waves. Many authors have performed three-dimensional inversions (see review by [Romanowicz, 1998](#)).

A common problem in attenuation tomography is that the initial amplitude A_0 (or equivalently, the CMT of the seismic source) is unknown (or poorly known). [Menke et al. \(2006\)](#) show that including A_0 as an unknown parameter in the inversion introduces a nonuniqueness that is mathematically identical to the one encountered when an unknown origin time is introduced into the velocity tomography problem.

Attenuation also has an effect on the free oscillations of the earth, causing a widening in the spectral peaks associated with the eigenfrequencies ω_{kmm} . The amount of widening depends on the relationship between the spatial variation of the attenuation and the spatial dependence of the eigenfunction and can be characterized by a quality factor Q_{kmm} (that is, a fractional loss of amplitude per oscillation) for that mode. Perturbative methods are used to calculate the Fréchet derivatives of Q_{kmm} with the attenuation factor. One example of such an inversion is that of [Masters and Gilbert \(1983\)](#).

13.6 SIGNAL CORRELATION

Geologic processes record signals only imperfectly. For example, while variations in oxygen isotopic ratios $r(t)$ through geologic time t are recorded in oxygen-bearing sediments as they are deposited, the sedimentation rate is itself a function of time. Measurements of isotopic ratio $r(z)$ as a function of depth z cannot be converted to the variation of $r(t)$ without knowledge of the sedimentation function $z(t)$ (or equivalently $t(z)$).

Under certain circumstances, the function $r(t)$ is known *a priori* (for instance, oxygen isotopic ratio correlates with temperature, which can be estimated independently). In these instances, it is possible to use the observed $r^{\text{obs}}(z)$ and the predicted $r^{\text{pre}}(t)$ to invert for the function $t(z)$. This is essentially a problem in signal correlation: distinctive features that can be correlated between $r^{\text{obs}}(z)$ and $r^{\text{pre}}(t)$ establish the function $t(z)$. The inverse problem

$$r^{\text{obs}}(z) = r^{\text{pre}}[t(z)] \quad (13.18)$$

is therefore a problem in nonlinear continuous inverse theory. The unknown function $t(z)$ —often called the *mapping function*—must increase monotonically with z . The solution of this problem is discussed by [Martinson et al. \(1982\)](#) and [Shure and Chave \(1984\)](#).

13.7 TECTONIC PLATE MOTIONS

The motion of the earth's rigid tectonic plates can be described by an *Euler vector* ω , whose orientation gives the pole of rotation and whose magnitude gives its rate. Euler vectors can be used to represent relative rotations, that is, the rotation of one plate relative to another, or absolute motion, that is, motion relative to the earth's mantle. If we denote the relative rotation of plate A with respect to plate B as ω_{AB} , then the Euler vectors of three plates A–C satisfy the relationship $\omega_{AB} + \omega_{BC} + \omega_{CA} = 0$. Once the Euler vectors for plate motion are known, the relative velocity between two plates at any point on their boundary can easily be calculated from trigonometric formulas.

Several geologic features provide information on the relative rotation between plates, including the faulting directions of earthquakes at plate boundaries and the orientation of transform faults, which constrain the direction of the relative velocity vectors, and spreading rate estimates based on magnetic lineations at ridges, which constrain the magnitude of the relative velocity vectors.

These data can be used in an inverse problem to determine the Euler vectors ([Chase and Stuart, 1972](#); [DeMets et al., 2010](#); [Minster et al., 1974](#)). The main differences between various authors' approaches are in the manner in which the Euler vectors are parameterized and the types of data that are used. Some authors use Cartesian components of the Euler vectors; others use magnitude, azimuth, and inclination. The two inversions behave somewhat differently in the presence of noise; Cartesian parameterizations seem to be more stable. Data include seafloor spreading rates, azimuths of strike-slip plate boundaries, earthquake moment tensors, and Global Positioning System strain measurements.

13.8 GRAVITY AND GEOMAGNETISM

Inverse theory plays an important role in creating representations of the earth's gravity and magnetic fields. Field measurements made at many points about the earth need to be combined into a smooth representation of the field, a problem which is mainly one of interpolation in the presence of noise and incomplete data (see [Section 3.9.3](#)). Both spherical harmonic expansions and various kinds of spline functions ([Sandwell, 1987](#); [Shure et al., 1982, 1985](#); [Wessel and Becker, 2008](#)) have been used in the representations. In either case, the trade-off of resolution and variance is very important.

Studies of the earth's core and geodynamo require that measurements of the magnetic field at the earth's surface be extrapolated to the core-mantle

boundary. This is an inverse problem of considerable complexity, since the short-wave-length components of the field that are most important at the core-mantle boundary are very poorly measured at the earth's surface. Furthermore, the rate of change of the magnetic field with time (called "secular variation") is of critical interest. This quantity must be determined from fragmentary historical measurements, including measurements of compass deviation recorded *a priori* constraints on the behavior of the field near the core, including the assumption that the root-mean-square time rate of change of the field is minimized and the total energy dissipation in the core is minimized (Bloxam, 1987). Once the magnetic field and its time derivative at the core-mantle boundary have been determined, they can be used in inversions for the fluid velocity near the surface of the outer core (Bloxam, 1992).

The earth's gravity field $\mathbf{g}(\mathbf{x})$ is determined by its density structure $\rho(\mathbf{x})$. In parts of the earth where electric currents and magnetic induction are unimportant, the magnetic field $\mathbf{H}(\mathbf{x})$ is caused by the magnetization $\mathbf{M}(\mathbf{x})$ of the rocks. These quantities are related by

$$\begin{aligned}
 g_i(\mathbf{x}) &= \int_V G_i^g(\mathbf{x}, \mathbf{x}_0) \rho(\mathbf{x}_0) dV_0 \quad \text{and} \quad H_i(\mathbf{x}) = \int_V G_{ij}^H(\mathbf{x}, \mathbf{x}_0) M_j(\mathbf{x}_0) dV_0 \\
 G_i^g(\mathbf{x}) &= -\frac{\gamma(x_i - x_{0i})}{|\mathbf{x} - \mathbf{x}_0|^3} \\
 G_{ij}^H(\mathbf{x}) &= \frac{1}{4\pi\mu_0} \left[\frac{\delta_{ij}}{|\mathbf{x} - \mathbf{x}_0|^3} - \frac{3(x_i - x_{0i})(x_j - x_{0j})}{|\mathbf{x} - \mathbf{x}_0|^5} \right]
 \end{aligned} \tag{13.19}$$

Here, γ is the gravitational constant and μ_0 is the magnetic permeability of the vacuum. Note that in both cases, the fields are linearly related to the sources (density and magnetization), so these are linear, continuous inverse problems. Nevertheless, inverse theory has proved to have little application to these problems, owing to their inherent underdetermined nature. In both cases, it is possible to show (e.g., Menke and Abbott, 1989, their Section 5.14) that the field outside a finite body can be generated by an infinitesimally thick spherical shell of mass or magnetization surrounding the body and below the level of the measurements. The null space of these problems is so large that it is generally impossible to formulate any useful solution (as we encountered in Figure 7.7), except when an enormous amount of *a priori* information is added to the problem.

Some progress has been made in special cases where *a priori* constraints can be sensibly stated. For instance, the magnetization of sea mounts can be computed from their magnetic anomaly, assuming that their magnetization vector is everywhere parallel (or nearly parallel) to some known direction and that the shape of the magnetized region closely corresponds to the observed bathymetric expression of the sea mount (Grossling, 1970; Parker, 1988; Parker et al., 1987).

13.9 ELECTROMAGNETIC INDUCTION AND THE MAGNETOTELLURIC METHOD

An electromagnetic field is *induced* within a conducting medium when it is illuminated by a plane electromagnetic wave. In a homogeneous medium, the field decays exponentially with depth, since energy is dissipated by electric currents induced in the conductive material. In a heterogeneous medium, the behavior of the field is more complicated and depends on the details of the electrical conductivity $\sigma(\mathbf{x})$.

Consider the special case of a vertically stratified earth illuminated by a vertically incident electromagnetic wave (e.g., from space). The plane wave consists of a horizontal electric field E_x and a horizontal magnetic field H_y , with their ratio on the earth's surface (at $z=0$) being determined by the conductivity profile $\sigma(z)$. The quantity

$$Z(\omega) = \frac{E_x(\omega, z=0)}{i\omega H_y(\omega, z=0)} \quad (13.20)$$

that relates the two fields is called the *impedance*. It is frequency dependent, because the depth of penetration of an electromagnetic wave into a conductive medium depends upon its frequency, so different frequencies sense different parts of the conductivity structure. The *magnetotelluric* problem is to determine the conductivity $\sigma(z)$ from measurements of the impedance $Z(\omega)$ at a suite of angular frequencies ω . This one-dimensional nonlinear inverse problem is relatively well understood. For instance, Fréchet derivatives are relatively straightforward to compute (Parker, 1970, 1977). Oldenburg (1983) discusses the discretization of the problem and the application of extremal inversion methods (see Chapter 6).

In three-dimensional problems, all the components of magnetic \mathbf{H} and electric \mathbf{E} fields are now relevant, and the impedance matrix $\mathbf{Z}(\omega, \mathbf{x})$ that connects them via $\mathbf{E} = \mathbf{Z}\mathbf{H}$ constitutes the data. The inverse problem of determining $\sigma(\mathbf{x})$ from observations of $\mathbf{Z}(\omega, \mathbf{x})$ at a variety of frequencies and positions is much more difficult than the one-dimensional version (Chave et al., 2012), especially in the presence of large heterogeneities in electrical resistivity caused, for instance, by topography (e.g., Baba and Chave, 2005).

REFERENCES

- Aki, K., Richards, P.G., 2002. Quantitative Seismology, second ed. University Science Books, Sausalito, California 700pp.
- Aki, K., Christofferson, A., Husebye, E., 1976. Three-dimensional seismic structure under the Montana LASA. Bull. Seism. Soc. Am. 66, 501–524.
- Baba, K., Chave, A., 2005. Correction of seafloor magnetotelluric data for topographic effects during inversion. J. Geophys. Res. 110, B12105, doi:10.1029/2004JB003463, 16pp.
- Bloxam, J., 1987. Simultaneous inversion for geomagnetic main field and secular variation, 1. A large scale inversion problem. J. Geophys. Res. 92, 11597–11608.

- Bloxam, J., 1992. The steady part of the secular variation of the earth's magnetic field. *J. Geophys. Res.* 97, 19565–19579.
- Boschi, J., Ekström, G., 2002. New images of the Earth's upper mantle from measurements of surface wave phase velocity anomalies. *J. Geophys. Res.* 107, 2059.
- Cervený, V., 2001. *Seismic Ray Theory*. Cambridge University Press, New York 607pp.
- Chase, E.P., Stuart, G.S., 1972. The N plate problem of plate tectonics. *Geophys. J. R. Astronom. Soc.* 29, 117–122.
- Chave, A., Jones, A., Mackie, R., Rodi, W., 2012. *The Magnetotelluric Method, Theory and Practice*. Cambridge University Press, New York, 590pp.
- Crosson, R.S., 1976. Crustal structure modeling of earthquake data: 1. Simultaneous least squares estimation of hypocenter and velocity parameters. *J. Geophys. Res.* 81, 3036–3046.
- Dahlen, F.A., Hung, S.-H., Nolet, G., 2000. Fréchet kernels for finite frequency traveltimes—i. Theory. *Geophys. J. Int.* 141, 157–174.
- Dalton, C.A., Ekström, G., 2006. Global models of surface wave attenuation. *J. Geophys. Res.* 111, B05317.
- DeMets, C., Gordon, R.G., Argus, D.F., 2010. Geologically current plate motions. *Geophys. J. Int.* 181, 1–80.
- Dziewonski, A.M., Anderson, D.L., 1981. Preliminary reference earth model. *Phys. Earth Planet. Interiors* 25, 297–358.
- Dziewonski, A.M., Woodhouse, J.H., 1983. An experiment in systematic study of global seismicity: centroid-moment tensor solutions for 201 moderate and large earthquakes of 1981. *J. Geophys. Res.* 88, 3247–3271.
- Dziewonski, A.M., Chou, T.-A., Woodhouse, J.H., 1981. Determination of earthquake source parameters from waveform data for studies of global and regional seismicity. *J. Geophys. Res.* 86, 2825–2852.
- Geiger, L., 1912. Probability method for the determination of earthquake epicenters from the arrival time only (translated from Geiger's 1910 German article). *Bull. St. Louis Univ.* 8, 56–71.
- Grossling, B.F., 1970. Seamount Magnetism. In: Maxwell, A.E. (Ed.), *The Sea*, vol. 4. Wiley, New York, pp. 129–156.
- Jacobson, R.S., Shor, G.G., Dorman, L.M., 1981. Linear inversion of body wave data—part 2: attenuation vs. depth using spectral ratios. *Geophysics* 46, 152–162.
- Kim, Y., Liu, Q., Tromp, J., 2011. Adjoint centroid-moment tensor inversions. *Geophys. J. Int.* 186, 264–278.
- Klein, F.W., 1985. User's guide to HYPOINVERSE, a program for VAX and PC350 computers to solve for earthquake locations. U.S. Geologic Survey Open File Report 85–515, 24pp.
- Marquering, H., Dahlen, F.A., Nolet, G., 1999. Three-dimensional sensitivity kernels for finite frequency traveltimes: the banana-doughnut paradox. *Geophys. J. Int.* 137, 805–815.
- Martinson, D.G., Menke, W., Stoffa, P., 1982. An inverse approach to signal correlation. *J. Geophys. Res.* 87, 4807–4818.
- Masters, G., Gilbert, F., 1983. Attenuation in the earth at low frequencies. *Philos. Trans. R. Soc. Lond. Ser. A* 388, 479–522.
- Menke, W., 2005. Case studies of seismic tomography and earthquake location in a regional context. In: Levander, A., Nolet, G. (Eds.), *Seismic Earth: Array Analysis of Broadband Seismograms*. Geophysical Monograph Series 157, American Geophysical Union, Washington, D.C., pp. 7–36.
- Menke, W., Abbott, D., 1989. *Geophysical Theory (Textbook)*. Columbia University Press, New York, 458pp.

- Menke, W., Schaff, D., 2004. Absolute earthquake location with differential data. *Bull. Seism. Soc. Am.* 94, 2254–2264.
- Menke, W., Holmes, R.C., Xie, J., 2006. On the nonuniqueness of the coupled origin time-velocity tomography problem. *Bull. Seism. Soc. Am.* 96, 1131–1139.
- Minster, J.B., Jordan, T.H., Molnar, P., Haines, E., 1974. Numerical modeling of instantaneous plate tectonics. *Geophys. J. R. Astronom. Soc.* 36, 541–576.
- Oldenburg, D.W., 1983. Funnel functions in linear and nonlinear appraisal. *J. Geophys. Res.* 88, 7387–7398.
- Parker, R.L., 1970. The inverse problem of the electrical conductivity of the mantle. *Geophys. J. R. Astronom. Soc.* 22, 121–138.
- Parker, R.L., 1977. The Fréchet derivative for the one dimensional electromagnetic induction problem. *Geophys. J. R. Astronom. Soc.* 39, 543–547.
- Parker, R.L., 1988. A statistical theory of seamount magnetism. *J. Geophys. Res.* 93, 3105–3115.
- Parker, R.L., Shure, L., Hildebrand, J., 1987. An application of inverse theory to seamount magnetism. *Rev. Geophys.* 25, 17–40.
- Pavlis, G.L., Booker, J.R., 1980. The mixed discrete-continuous inverse problem: application to the simultaneous determination of earthquake hypocenters and velocity structure. *J. Geophys. Res.* 85, 4801–4809.
- Romanowicz, B., 1998. Attenuation tomography of the earth's mantle: a review of current status. *Pure Appl. Geophys.* 153, 257–272.
- Sandwell, D.T., 1987. Biharmonic spline interpolation of GEOS-3 and SEASAT altimeter data. *Geophys. Res. Lett.* 14, 139–142.
- Shure, L., Chave, A.D., 1984. Comments on “An inverse approach to signal correlation.” *J. Geophys. Res.* 89, 2497–2500.
- Shure, L., Parker, R.L., Backus, G.E., 1982. Harmonic splines for geomagnetic modeling. *Phys. Earth Planet. Interiors* 28, 215–229.
- Shure, L., Parker, R.L., Langel, R.A., 1985. A preliminary harmonic spline model for MAGSAT data. *J. Geophys. Res.* 90, 11505–11512.
- Sieminski, A., Liu, Q., Trampert, J., Tromp, J., 2007. Finite-frequency sensitivity of surface waves to anisotropy based upon adjoint methods. *Geophys. J. Int.* 168, 1153–1174.
- Slunga, R., Rognvaldsson, S.T., Bodvarsson, B., 1995. Absolute and relative locations of similar events with application to microearthquakes in southern Iceland. *Geophys. J. Int.* 123, 409–419.
- Spence, W., Alexander, S., 1968. A method of determining the relative location of seismic events. *Earthquake Notes* 39, 13.
- Takeuchi, H., Saito, M., 1972. Seismic surface waves. In: Bolt, B.A. (Ed.), *Methods of Computational Physics* 11, Academic Press, New York, 217–295.
- Tromp, J., Tape, C., Liu, Q., 2005. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophys. J. Int.* 160, 195–216.
- Vasco, D.W., 1986. Extremal inversion of travel time residuals. *Bull. Seismol. Soc. Am.* 76, 1323–1345.
- Vidale, J., 1990. Finite difference calculation of travel times in 3-D. *Geophysics* 55, 521–526.
- Waldhauser, F., Ellsworth, W., 2000. A double-difference earthquake location algorithm; method and application to the northern Hayward Fault, California. *Bull. Seism. Soc. Am.* 90, 1353–1368.
- Wang, Z., Dahlen, F.A., 1995. Validity of surface-wave ray theory on a laterally heterogeneous earth. *Geophys. J. Int.* 123, 757–773.

- Wessel, P., Becker, J., 2008. Gridding of spherical data using a Green's function for splines in tension. *Geophys. J. Int.* 174, 21–28.
- Zelt, C.A., 1998. FAST: 3-D First Arrival Seismic Tomography programs. <http://terra.rice.edu/departement/faculty/zelt/fast.html>.
- Zelt, C.A., Barton, P.J., 1998. 3D seismic refraction tomography: a comparison of two methods applied to data from the Faeroe Basin. *J. Geophys. Res.* 103, 7187–7210.
- Zhang, H., Thurber, C.H., 2003. Double-difference tomography; the method and its application to the Hayward Fault, California. *Bull. Seismol. Soc. Am.* 93, 1875–1889.

Appendices

14.1 IMPLEMENTING CONSTRAINTS WITH LAGRANGE MULTIPLIERS

Consider the problem of minimizing a function of two variables, say, $E(x, y)$, with respect to x and y , subject to the constraint that $C(x, y)=0$. One way to solve this problem is to first use $C(x, y)=0$ to write y as a function of x and then substitute this function into $E(x, y)$. The resulting function of a single variable $E(x, y(x))$ can now be minimized by setting $dE/dx=0$. The constraint equation is used to explicitly reduce the number of independent variables.

One problem with this method is that it is rarely possible to solve $C(x, y)$ explicitly for either $y(x)$ or $x(y)$. The method of Lagrange multipliers provides a method of dealing with the constraints in their implicit form.

The constraint equation $C(x, y)=0$ defines a curve in (x, y) on which the solution must lie (Figure 14.1). As a point is moved along this curve, the value of E changes, achieving a minimum at (x_0, y_0) . This point is not the global minimum of E , which lies, in general, off the curve. However, it must be at a point at which ∇E is perpendicular to the tangent \hat{t} to the curve, or else the point could be moved along the curve to further minimize E . The tangent is perpendicular to ∇C , so the condition that $\nabla E \perp \hat{t}$ is equivalent to $\nabla E \propto \nabla C$. Writing the proportionality factor as $-\lambda$, we have $\nabla E + \lambda \nabla C = 0$ or $\nabla(E + \lambda C) = 0$. Thus, the constrained minimization of E subject to $C=0$ is equivalent to the unconstrained minimization of $\Phi = E + \lambda C$, except that a new variable λ is introduced that needs to be determined as part of the problem. The two equations $\nabla \Phi = 0$ and $C=0$ must be solved simultaneously to yield (x_0, y_0) and λ .

In N -dimensions and with $L < N$ constraint equations, each constraint $C_i=0$ describes an $N-1$ dimensional surface. Their intersection is an $N-L$ dimensional surface. We treat the $L=N-1$ case here, where the intersection is a curve (the argument for the other cases being similar). As in the two-dimensional case, the condition that a point (x_0, y_0) is at a minimum of E on this curve is that ∇E is perpendicular to the tangent \hat{t} to the curve. The curve necessarily lies within all of the intersecting surfaces, and so \hat{t} is perpendicular to any surface normal ∇C_i . Thus, the condition that $\nabla E \perp \hat{t}$ is equivalent to the requirement that ∇E be

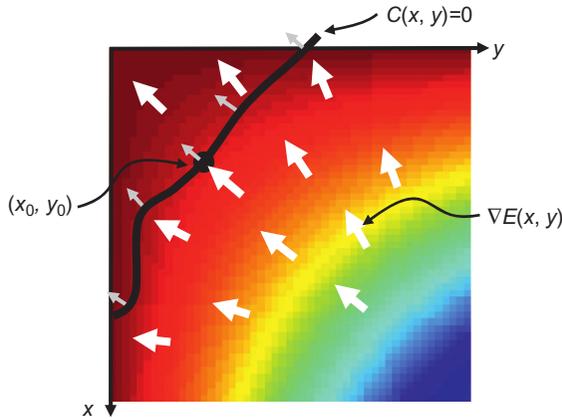


FIGURE 14.1 Graphical interpretation of the method of Lagrange multipliers, in which the function $E(x, y)$ is minimized subject to the constraint that $C(x, y) = 0$. The solution (bold dot) occurs at the point (x_0, y_0) on the curve $C(x, y) = 0$, where the perpendicular direction (gray arrows) is parallel to the gradient $\nabla E(x, y)$ (white arrows). At this point, E can only be further minimized by moving the point (x_0, y_0) off of the curve, which is disallowed by the constraint. *MatLab* script `gda14_01`.

constructed from a linear combination of surface normals, ∇C_i . Calling the coefficients of the linear combination $-\lambda_i$, we have

$$\nabla E = -\sum_{i=1}^L \lambda_i \nabla C_i \quad \text{or} \quad \nabla E + \sum_{i=1}^L \lambda_i \nabla C_i = 0 \quad \text{or} \quad \nabla \Phi = 0 \quad (14.1)$$

with $\Phi = E + \sum_{i=1}^L \lambda_i C_i$

As before, the constrained minimization of E subject to $C_i = 0$ is equivalent to the unconstrained minimization of Φ , but now Φ contains one Lagrange multiplier λ_i for each of the L constraints.

14.2 L_2 INVERSE THEORY WITH COMPLEX QUANTITIES

Some inverse problems (especially those that deal with data that have been operated on by Fourier or other transforms) involve complex quantities that are considered random variables. A complex random variable, say $z = x + iy$, has real and imaginary parts, x and y , that are themselves real-valued random variables. The probability density function $p(z)$ gives the probability that the real part of z is between x and $x + \Delta x$ and an imaginary part is in between y and $y + \Delta y$. The probability density function is normalized so that its integral over the complex plane is unity:

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(z) dx dy = 1 \quad (14.2)$$

The mean or expected value of the z is then the obvious generalization of the real-valued case

$$\langle z \rangle = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} zp(z) dx dy \quad (14.3)$$

In general, the real and imaginary parts of z can be of unequal variance and be correlated. However, the uncorrelated, equal variance case, which is called a *circular random variable*, suffices to describe the behavior of many systems. Its variance is defined as

$$\sigma = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (z - \langle z \rangle)^* (z - \langle z \rangle) p(z) dx dy = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |z - \langle z \rangle|^2 p(z) dx dy \quad (14.4)$$

Here, the asterisk signifies complex conjugation. A vector \mathbf{z} of N circular random variables, with probability density function $p(\mathbf{z})$, has mean and covariance given by

$$\begin{aligned} \langle \mathbf{z} \rangle_i &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} z_i p(\mathbf{z}) d^N x d^N y \\ [\text{cov} \mathbf{z}]_{ij} &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (z_i - \langle z_i \rangle)^* (z_j - \langle z_j \rangle) p(\mathbf{z}) d^N x d^N y \end{aligned} \quad (14.5)$$

Note that $[\text{cov} \mathbf{z}]$ is a Hermetian matrix, that is, a matrix whose transform is its complex conjugate.

The definition of the L_2 norm must be changed to accommodate the complex nature of quantities. The appropriate change is to define the squared length of a vector \mathbf{v} to be

$$\|\mathbf{v}\|_2^2 = \mathbf{v}^H \mathbf{v} \quad (14.6)$$

where \mathbf{v}^H is the *Hermetian transpose*, that is, the transpose of the complex conjugate of the vector \mathbf{v} . This choice ensures that the norm is a nonnegative real number. When the results of L_2 inverse theory are rederived for circular complex vectors, the results are very similar to those derived previously; the only difference is that all the ordinary transposes are replaced by Hermetian transposes. For instance, the least squares solution is

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^H \mathbf{G}]^{-1} \mathbf{G}^H \mathbf{d} \quad (14.7)$$

Note that all the square symmetric matrices of real inverse theory now become square Hermetian matrices:

$$[\text{cov} \mathbf{d}], [\text{cov} \mathbf{m}], [\mathbf{G}^H \mathbf{G}], [\mathbf{G} \mathbf{G}^H], \text{etc.} \quad (14.8)$$

Hermetian matrices have real eigenvalues, so no special problems arise when deriving eigenvalues or singular-value decompositions.

The modification made to Householder transformations is also very simple. The requirement that the Hermetian length of a vector be invariant under transformation implies that a unitary transformation must satisfy $\mathbf{T}^H\mathbf{T}=\mathbf{I}$. The most general unitary transformation is therefore $\mathbf{T}=\mathbf{I}-2\mathbf{v}\mathbf{v}^H/\mathbf{v}^H\mathbf{v}$, where \mathbf{v} is any complex vector. The Householder transformation that annihilates the j th column of \mathbf{G} below its main diagonal then becomes

$$\mathbf{T}_j = \mathbf{I} - \frac{1}{|\alpha|(|\alpha| - |G_{j,j}|)} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (G_{j,j} - \alpha) \\ G_{j+1,j} \\ \vdots \\ G_{N,j} \end{bmatrix} \begin{bmatrix} 0 \cdots 0 & (G_{j,j}^* - \alpha^*) & G_{j+1,j}^* \cdots G_{N,j}^* \end{bmatrix}$$

where $|\alpha| = \sqrt{\sum_{i=j}^N |G_{i,j}|^2}$

(14.9)

The phase of α is chosen to be π away from the phase of $G_{j,j}$. This choice guarantees that the transformation is in fact a unitary transformation and that the denominator is not zero.



Dedication

**To my parents
William and Lee Menke
who taught me how to learn**

1.1 FORWARD AND INVERSE THEORIES

Inverse theory is an organized set of mathematical techniques for reducing data to obtain *knowledge* about the physical world on the basis of inferences drawn from observations. Inverse theory, as we shall consider it in this book, is limited to observations and questions that can be represented numerically. The observations of the world will consist of a tabulation of measurements, or *data*. The questions we want to answer will be stated in terms of the numerical values (and statistics) of specific (but not necessarily directly measurable) properties of the world. These properties will be called model *parameters* for reasons that will become apparent. We shall assume that there is some specific method (usually a mathematical theory or model) for relating the model parameters to the data.

The question, what causes the motion of the planets? for example, is not one to which inverse theory can be applied. Even though it is perfectly scientific and historically important, its answer is not numerical in nature. On the other hand, inverse theory can be applied to the question, assuming that Newtonian mechanics applies, determine the number and orbits of the planets on the basis of the observed orbit of Halley's comet. The number of planets and their orbital ephemerides are numerical in nature. Another important difference between these two problems is that the first asks us to determine the reason for the orbital motions, and the second presupposes the reason and asks us only to determine certain details. Inverse theory rarely supplies the kind of insight demanded by the first question; it always requires that the physical model or theory be specified beforehand.

The term *inverse theory* is used in contrast to *forward theory*, which is defined as the process of predicting the results of measurements (predicting data) on the basis of some general principle or model and a set of specific conditions relevant to the problem at hand. Inverse theory, roughly speaking, addresses the reverse problem: starting with data and a general principle, theory, or quantitative model, it determines estimates of the model parameters. In the above example, predicting the orbit of Halley's comet from the presumably well-known orbital ephemerides of the planets is a problem for forward theory.

Another comparison of forward and inverse problems is provided by the phenomenon of temperature variation as a function of depth beneath the earth's surface. Let us assume that the temperature increases linearly with depth in the

earth, that is, temperature T is related to depth z by the rule $T(z) = az + b$, where a and b are numerical constants that we will refer to as *model parameters*. If one knows that $a = 0.1$ and $b = 25$, then one can solve the forward problem simply by evaluating the formula for any desired depth. The inverse problem would be to determine a and b on the basis of a suite of temperature measurements made at different depths in, say, a bore hole. One may recognize that this is the problem of fitting a straight line to data, which is a substantially harder problem than the forward problem of evaluating a first-degree polynomial. This brings out a property of most inverse problems: that they are substantially harder to solve than their corresponding forward problems.

Forward problem:

estimates of model parameters \rightarrow quantitative model \rightarrow predictions of data

Inverse problem:

observations of data \rightarrow quantitative model \rightarrow estimates of model parameters

Note that the role of inverse theory is to provide information about unknown numerical parameters that go into the model, not to provide the model itself. Nevertheless, inverse theory can often provide a means for assessing the correctness of a given model or of discriminating between several possible models.

The model parameters one encounters in inverse theory vary from discrete numerical quantities to continuous functions of one or more variables. The intercept and slope of the straight line mentioned above are examples of discrete parameters. Temperature, which varies continuously with position, is an example of a continuous function. This book deals mainly with discrete inverse theory, in which the model parameters are represented as a set of a finite number of numerical values. This limitation does not, in practice, exclude the study of continuous functions, since they can usually be adequately approximated by a finite number of discrete parameters. Temperature, for example, might be represented by its value at a finite number of closely spaced points or by a set of splines with a finite number of coefficients. This approach does, however, limit the rigor with which continuous functions can be studied. Parameterizations of continuous functions are always both approximate and, to some degree, arbitrary properties, which cast a certain amount of imprecision into the theory. Nevertheless, discrete inverse theory is a good starting place for the study of inverse theory, in general, since it relies mainly on the theory of vectors and matrices rather than on the somewhat more complicated theory of continuous functions and operators. Furthermore, careful application of discrete inverse theory can often yield considerable insight, even when applied to problems involving continuous parameters.

Although the main purpose of inverse theory is to provide estimates of model parameters, the theory has a considerably larger scope. Even in cases in which the model parameters are the only desired results, there is a plethora

of related information that can be extracted to help determine the “goodness” of the solution to the inverse problem. The actual values of the model parameters are indeed irrelevant in cases when we are mainly interested in using inverse theory as a tool in experimental design or in summarizing the data. Some of the questions inverse theory can help answer are the following:

- (a) What are the underlying similarities among inverse problems?
- (b) How are estimates of model parameters made?
- (c) How much of the error in the measurements shows up as error in the estimates of the model parameters?
- (d) Given a particular experimental design, can a certain set of model parameters really be determined?

These questions emphasize that there are many different kinds of answers to inverse problems and many different criteria by which the goodness of those answers can be judged. Much of the subject of inverse theory is concerned with recognizing when certain criteria are more applicable than others, as well as detecting and avoiding (if possible) the various pitfalls that can arise.

Inverse problems arise in many branches of the physical sciences. An incomplete list might include such entries as

- (a) medical and seismic tomography,
- (b) image enhancement,
- (c) curve fitting,
- (d) earthquake location,
- (e) oceanographic and meteorological data assimilation,
- (f) factor analysis,
- (g) determination of earth structure from geophysical data,
- (h) satellite navigation,
- (i) mapping of celestial radio sources with interferometry, and
- (j) analysis of molecular structure by X-ray diffraction.

Inverse theory was developed by scientists and mathematicians having various backgrounds and goals. Thus, although the resulting versions of the theory possess strong and fundamental similarities, they have tended to look, superficially, very different. One of the goals of this book is to present the various aspects of discrete inverse theory in such a way that both the individual viewpoints and the “big picture” can be clearly understood.

There are perhaps three major viewpoints from which inverse theory can be approached. The first and oldest sprang from probability theory—a natural starting place for such “noisy” quantities as observations of the real world. In this version of inverse theory, the data and model parameters are treated as random variables, and a great deal of emphasis is placed on determining the probability density functions that they follow. This viewpoint leads very naturally to the analysis of error and to tests of the significance of answers.

The second viewpoint developed from that part of the physical sciences that retains a deterministic stance and avoids the explicit use of probability theory. This approach has tended to deal only with estimates of model parameters (and perhaps with their error bars) rather than with probability density functions per se. Yet what one means by an estimate is often nothing more than the expected value of a probability density function; the difference is only one of emphasis.

The third viewpoint arose from a consideration of model parameters that are inherently continuous functions. Whereas the other two viewpoints handled this problem by approximating continuous functions with a finite number of discrete parameters, the third developed methods for handling continuous functions explicitly. Although continuous inverse theory is not the primary focus of this book, many of the concepts originally developed for it have application to discrete inverse theory, especially when it is used with discretized continuous functions.

1.2 *MATLAB* AS A TOOL FOR LEARNING INVERSE THEORY

The practice of inverse theory requires computer-based computation. A person can learn many of the *concepts* of inverse theory by working through short pencil-and-paper examples and by examining precomputed figures and graphs. But he or she cannot become proficient in the *practice* of inverse theory that way because it requires skills that can only be obtained through the experience of working with large data sets. Three goals are paramount: to develop the judgment needed to select the best solution method among many alternatives; to build confidence that the solution can be obtained even though it requires many steps; and to strengthen the critical faculties needed to assess the quality of the results. This book devotes considerable space to case studies and homework problems that provide the practical problem-solving experience needed to gain proficiency in inverse theory.

Computer-based computation requires software. Many different software environments are available for the type of scientific computation that underpins data analysis. Some are more applicable and others less applicable to inverse theory problems, but among the applicable ones, none has a decisive advantage. Nevertheless, we have chosen *MatLab*, a commercial software product of *The MathWorks, Inc.* as the book's software environment for several reasons, some having to do with its designs and other more practical. The most persuasive design reason is that its syntax fully supports linear algebra, which is needed by almost every inverse theory method. Furthermore, it supports *scripts*, that is, sequences of data analysis commands that are communicated in written form and which serve to document the data analysis process. Practical considerations include the following: it is a long-lived and stable product, available since the mid-1980s; implementations are available for most commonly used types of computers; its price, especially for students, is fairly modest; and it is widely used, at least, in university settings.

In *MatLab*'s scripting language, data are presented as one or more *named variables* (in the same sense that c and d in the formula, $c = \pi d$, are named variables). Data are manipulated by typing formula that create new variables from old ones and by running *scripts*, that is, sequences of formulas stored in a file. Much of inverse theory is simply the application of well-known formulas to novel data, so the great advantage of this approach is that the formulas that are typed usually have a strong similarity to those printed in a textbook. Furthermore, scripts provide both a way of documenting the sequence of a formula used to analyze a particular data set and a way to transfer the overall data analysis procedure from one data set to another. However, one disadvantage is that the parallel between the syntax of the scripting language and the syntax of standard mathematical notation is nowhere near perfect. A person needs to learn to translate one into the other.

I.3 A VERY QUICK *MATLAB* TUTORIAL

Unfortunately, this book must avoid discussion of the installation of *MatLab* and the appearance of *MatLab* on your computer screen, for procedures and appearances vary from computer to computer and quickly become outdated, anyway. We will assume that you have successfully installed it and that you can identify the Command Window, the place where *MatLab* formula and commands are typed. Once you have identified the Command Window, try typing:

```
date
```

MatLab should respond by displaying today's date. All the *MatLab* commands that are used in this book are in freely available *MatLab* scripts. This one is named `gda01_01` and is in a *MatLab* script file (*m-file*, for short) named `gda01_01.m` (conventionally, m-files have filenames that end with ".m"). In this case, the script is very short, since it just contains this one command, `date`, together with a couple of comment lines (which start with the character "%"):

```
% gda00_01
% displays the current date
date
```

(*MatLab* script `gda00_01`)

All the scripts are in a folder named `gda`. You should copy it to a convenient and easy-to-remember place in your computer's file system. The *MatLab* command window supports a number of commands that enable you to navigate from folder to folder, list the contents of folders, etc. For example, when you type:

```
pwd
```

(for "print working directory") in the Command Window, *MatLab* responds by displaying the name of the current folder. Initially, this is almost invariably the wrong folder, so you will need to `cd` (for "change directory") to the folder where you want

to be—the `ch00` folder in this case. The pathname will, of course, depend upon where you copied the `gda` folder but will end in `gda/ch00`. On the author’s computer, typing:

```
cd c:/menke/docs/gda/ch00
```

does the trick. If you have spaces in your pathname, just surround it with single quotes:

```
cd 'c:/menke/my docs/gda/ch00'
```

You can check if you are in the right folder by typing `pwd` again. Once in the `ch00` folder, typing:

```
gda00_01
```

will run the `gda00_01` m-script, which displays the current date. You can move to the folder above the current one by typing:

```
cd ..
```

and to one below it by giving just the folder name. For example, if you are in the `gda` folder you can move to the `ch00` folder by typing:

```
cd ch00
```

Finally, the command `dir` (for “directory”) lists the files and subfolders in the current directory.

```
dir
```

(*MatLab* script `gda00_02`)

The *MatLab* commands for simple arithmetic and algebra closely parallel standard mathematical notation. For instance, the command sequence

```
a=4.5;
b=5.1;
c=a+b;
c
```

(*MatLab* script `gda00_03`)

evaluates the formula $c = a + b$ for the case $a = 4.5$ and $b = 5.1$ to obtain $c = 9.6$. Only the semicolons require explanation. By default, *MatLab* displays the value of every formula typed into the Command Window. A semicolon at the end of the formula suppresses the display. Hence, the first three lines, which end with semicolons, are evaluated but not displayed. Only the final line, which lacks the semicolon, causes *MatLab* to print the final result, `c`.

Note that *MatLab* variables are *static*, meaning that they persist in *MatLab*’s *Workspace* until you explicitly delete them or exit the program. Variables

created by one script can be used by subsequent scripts. At any time, the value of a variable can be examined, either by displaying it in the Command Window (as we have done above) or by using the spreadsheet-like display tools available through *MatLab*'s Workspace Window. The persistence of *MatLab* variables can sometimes lead to scripting errors, such as when the definition of a variable in a script is inadvertently omitted, but *MatLab* used the value defined in a previous script. The command `clear all` deletes all previously defined variables in the Workspace. Placing this command at the beginning of a script causes it to delete any previously defined variables every time that it is run, ensuring that it cannot be affected by them.

The four commands discussed above can be run as a unit by typing `gda00_03`. Now open the m-file `gda01_03` in *MatLab*, using the File/Open menu. *MatLab* will bring up a text-editor type window. First, save it as a new file, say, `mygda01_03`; edit it in some simple way, say, by changing the 3.5 to 4.5; save the edited file; and run it by typing `mygda01_03` in the Command Window. The value of `c` that is displayed will have changed appropriately.

A somewhat more complicated formula is

$$c = \sqrt{a^2 + b^2} \quad \text{with } a = 6 \quad \text{and } b = 8$$

```
a=6;
b=8;
c = sqrt(a^2 + b^2);
c
```

(*MatLab* `gda00_04`)

Note that the *MatLab* syntax for a^2 is `a^2` and that the square root is computed using the function, `sqrt()`. This is an example of *MatLab*'s syntax differing from standard mathematical notation.

A final example is

$$c = \sin \frac{n\pi(x - x_0)}{L} \quad \text{with } n = 3, x = 4, x_0 = 1, L = 6$$

```
n=3; x=4; x0=1; L=6;
c = sin(n*pi*(x-x0)/L);
c
```

(*MatLab* `gda00_05`)

Note that several formulas, separated by semicolons, can be typed on the same line. Variables, such as `x0` and `pi`, above, can have names consisting of more than one character and can contain numerals as well as letters (though they must start with a letter). *MatLab* has a variety of predefined mathematical constants, including `pi`, which is the usual mathematical constant, π .

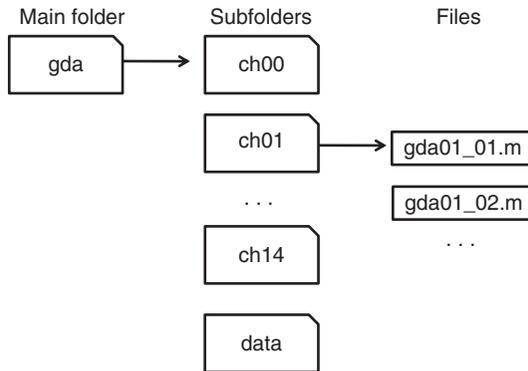


FIGURE I.1 Folder (directory) structure used for the files accompanying this book.

Files proliferate at an astonishing rate, even in the most trivial data analysis project. Data, notes, m-scripts, intermediate and final results, and graphics will all be contained in files, and their numbers will grow during the project. These files need to be organized through a system of folders (directories), subfolders (subdirectories), and filenames that are sufficiently systematic that files can be located easily and so that they are not confused with one another. Predictability both in the pattern of filenames and in the arrangement of folders and subfolders is an extremely important part of the design.

The files associated with this book are in a two-tiered folder/subfolder structure modeled on the chapter format of the book itself (Figure I.1). Folder and filenames are systematic. The chapter folder names are always of the form `chNN`, where `NN` is the chapter number. The m-script filenames are always of the form `gdaNN_MM.m`, where `NN` is the chapter number and `MM` are sequential integers. We have chosen to use leading zeros in the naming scheme (for example, `ch00`) so that filenames appear in the correct order when they are sorted alphabetically (as when listing the contents of a folder).

Many *MatLab* manuals, guides, and tutorials are available, both in printed form (e.g., Menke and Menke, 2011; Part-Enander *et al.*, 1996; Pratap, 2009) and on the Web (e.g., at www.mathworks.com). The reader may find that they complement this book by providing more detailed information about *MatLab* as a scientific computing environment.

I.4 REVIEW OF VECTORS AND MATRICES AND THEIR REPRESENTATION IN *MATLAB*

Vectors and matrices are fundamental to inverse theory both because they provide a convenient way to organize data and because many important operations on data can be very succinctly expressed using *linear algebra* (that is, the algebra of vectors and matrices).

In the simplest interpretation, a *vector* is just a list of numbers that is treated as unit and given a symbolic name. The list can be organized horizontally, as a row, in which case it is called a *row vector*. Alternately, the list can be organized vertically, as a column, in which case it is called a *column vector*. We will use lower-case bold letters to represent both kinds of vector. An exemplary 1×3 row vector \mathbf{r} and a 3×1 column vector \mathbf{c} are

$$\mathbf{r} = [2 \ 4 \ 6] \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$$

A row vector can be turned into a column vector, and vice versa, by the *transpose* operation, denoted by the superscript “T.” Thus,

$$\mathbf{r}^T = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \quad \text{and} \quad \mathbf{c}^T = [1 \ 3 \ 5]$$

In *MatLab*, row vector and column vectors are defined by

```
r = [2, 4, 6];
c = [1, 3, 5]';
```

(*MatLab* gda00_06)

In *MatLab*, the transpose is denoted by the single quote, so the column vector \mathbf{c} in the script above is created by transposing a row vector. Although both column vectors and row vectors are useful, our experience is that defining both in the same script creates serious opportunities for error. A formula that requires a column vector will usually yield incorrect results if a row vector is substituted into it, and vice versa. Consequently, we will adhere to a protocol where all vectors defined in this book are column vectors. Row vectors will be created when needed—and as close as possible to where they are used in the script—by transposing the equivalent column vector.

An individual number within a vector is called an *element* (or, sometimes, *component*) and is denoted with an integer index, written as a subscript, with the index 1 in the left of the row vector and top of the column vector. Thus, $r_2=4$ and $c_3=5$ in the example above. In *MatLab*, the index is written inside parentheses, as in

```
a=r(2);
b=c(3);
```

(*MatLab* gda00_06)

Sometimes, we will wish to indicate a generic element of the vector, in which case we will give it a variable index, as in r_i and c_j , with the understanding that i and j are integer variables.

In the simplest interpretation, a *matrix* is just a rectangular table of numbers. We will use bold uppercase names to denote matrices, as in

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

In the above example, the number of rows and number of columns are equal, but this property is not required; matrices can also be rectangular. Thus, row vectors and column vectors are just special cases of rectangular matrices. The transposition operation can also be performed on a matrix, in which case its rows and columns are interchanged.

$$\mathbf{M}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

A square matrix \mathbf{M} is said to be *symmetric* if $\mathbf{M} = \mathbf{M}^T$. In *MatLab*, a matrix is defined by

$$\mathbf{M} = [[1, 4, 7]', [2, 5, 8]', [3, 6, 9]'];$$

(*MatLab* gda00_06)

or, alternately, by

$$\begin{aligned} \mathbf{M} &= [1, 2, 3; \\ &4, 5, 6; \\ &7, 8, 9]; \end{aligned}$$

(*MatLab* gda00_06)

The first case, the matrix, \mathbf{M} , is constructed from a “row vector of column vectors” and in the second case, as a “column vector of row vectors.” The individual elements of a matrix are denoted with two integer indices, the first indicating the row and the second the column, starting in the upper left. Thus, in the example above, $M_{31} = 3$. Note that transposition swaps indices; that is, M_{ji} is the transpose of M_{ij} . In *MatLab*, the indices are written inside parentheses, as in

$$c = \mathbf{M}(1, 3);$$

(*MatLab* gda01_06)

One of the key properties of vectors and matrices is that they can be manipulated symbolically—as entities—according to specific rules that are similar to normal arithmetic. This allows tremendous simplification of data processing formulas, since all the details of what happens to individual elements within those entities are hidden from view and automatically performed.

In order to be added, two matrices (or vectors, viewing them as a special case of a rectangular matrix) must have the same number of rows and columns. Their sum is then just the matrix that results from summing corresponding elements. Thus, if

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{N} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix} \quad \text{then}$$

$$\mathbf{S} = \mathbf{M} + \mathbf{N} = \begin{bmatrix} 1+1 & 0+0 & 2-1 \\ 0+0 & 1+2 & 0+0 \\ 2-1 & 0+0 & 1+3 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 3 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

Subtraction is performed in an analogous manner. In terms of the components, addition and subtraction are written as

$$S_{ij} = M_{ij} + N_{ij} \quad \text{and} \quad D_{ij} = M_{ij} - N_{ij}$$

Note that addition is commutative (that is, $\mathbf{M} + \mathbf{N} = \mathbf{N} + \mathbf{M}$) and associative (that is, $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$). In *MatLab*, addition and subtraction are written as

$$S = M+N;$$

$$D = M-N;$$

(*MatLab* gda00_07)

Multiplication of two matrices is a more complicated operation and requires that the number of columns of the left-hand matrix equal the number of rows of the right-hand matrix. Thus, if the matrix \mathbf{M} is $N \times K$ and the matrix \mathbf{N} is $K \times M$, the product $\mathbf{P} = \mathbf{N}\mathbf{M}$ is an $N \times M$ matrix defined according to the rule (Figure I.2):

$$P_{ij} = \sum_{k=1}^K M_{ik}N_{kj}$$

The order of the indices is important. Matrix multiplication is in its standard form when all summations involve neighboring indices of adjacent quantities. Thus, for instance, the two instances of the summed variable k are *not* neighboring in the equation

$$Q_{ij} = \sum_{k=1}^K M_{ki}N_{kj}$$

and so the equation corresponds to $\mathbf{Q} = \mathbf{M}^T\mathbf{N}$ and not $\mathbf{Q} = \mathbf{M}\mathbf{N}$. Matrix multiplication is not commutative (that is, $\mathbf{M}\mathbf{N} \neq \mathbf{N}\mathbf{M}$) but is associative (that is, $(\mathbf{A}\mathbf{B})$

$$\begin{bmatrix} 1 & 1 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \times 1 + 1 \times 2 + 2 \times 1 + 2 \times 2 & 1 \times 2 + 1 \times 1 + 2 \times 2 + 2 \times 1 \\ 2 \times 1 + 2 \times 2 + 3 \times 1 + 3 \times 2 & 2 \times 2 + 2 \times 1 + 3 \times 2 + 3 \times 1 \end{bmatrix} = \begin{bmatrix} 9 & 9 \\ 15 & 15 \end{bmatrix}$$

FIGURE I.2 Graphical depiction of matrix multiplication. A row of the first matrix is paired up with a column of the second matrix. The pairs are multiplied together and the results are summed, producing one element of resultant matrix.

$\mathbf{C} = \mathbf{A}(\mathbf{BC})$) and distributive (that is, $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$). An important rule involving the matrix transpose is $(\mathbf{MN})^T = \mathbf{N}^T \mathbf{M}^T$ (note the reversal of the order).

Several special cases of multiplication involving vectors are noteworthy. Suppose that \mathbf{a} and \mathbf{b} are length- N column vectors. The combination $s = \mathbf{a}^T \mathbf{b}$ is a scalar number s and is called the *dot product* (or sometimes, *inner product*) of the vectors. It obeys the rule $\mathbf{a}^T \mathbf{b} = \mathbf{b}^T \mathbf{a}$. The dot product of a vector with itself is the square of its *Euclidian length*; that is, $\mathbf{a}^T \mathbf{a}$ is the sum of its squared elements of \mathbf{a} . The vector \mathbf{a} is said to be a unit vector when $\mathbf{a}^T \mathbf{a} = 1$. The combination $\mathbf{T} = \mathbf{ab}^T$ is an $N \times N$ matrix; it is called the *outer product*. The product of a matrix and a vector is another vector, as in $\mathbf{c} = \mathbf{Ma}$. One interpretation of this relationship is that the matrix \mathbf{M} “turns one vector into another.” Note that the vectors \mathbf{a} and \mathbf{c} can be of different length. An $M \times N$ matrix \mathbf{M} turns the length- N vector \mathbf{a} into a length- M vector \mathbf{c} . The combination $s = \mathbf{a}^T \mathbf{Ma}$ is a scalar and is called a *quadratic form*, since it contains terms quadratic in the elements of \mathbf{a} . Matrix multiplication, $\mathbf{P} = \mathbf{NM}$, has a useful interpretation in terms of dot products: P_{ij} is the dot product of the i th row of \mathbf{M} with the j th column of \mathbf{N} .

Any matrix is unchanged when multiplied by the *identity matrix*, conventionally denoted \mathbf{I} . Thus, $\mathbf{a} = \mathbf{Ia}$, $\mathbf{M} = \mathbf{IM} = \mathbf{MI}$, etc. This matrix has ones along its main diagonal, and zeroes elsewhere, as in

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The elements of the identity matrix are usually written δ_{ij} and not I_{ij} , and the symbol δ_{ij} is usually called the *Kronecker delta* symbol, not the *elements of the identity matrix* (though that is exactly what it is). The equation $\mathbf{M} = \mathbf{IM}$, for an $N \times N$ matrix \mathbf{M} , is written component-wise as

$$M_{ij} = \sum_{k=1}^N \delta_{ik} M_{kj}$$

This equation indicates that any summation containing a Kronecker delta symbol can be performed trivially. To obtain the result, one first identifies the variable that is being summed over (k in this case) and the variable that the summed variable is paired with in the Kronecker delta symbol (i in this case). The summation and the Kronecker delta symbol then are deleted from the equation, and all occurrences of the summed variable are replaced with the paired variable (all k s are replaced by i s in this case). In *MatLab*, an $N \times N$ identity matrix can be created with the command

```
I = eye(N);
```

(*MatLab* gda00_07)

MatLab performs all multiplicative operations with ease. For example, suppose column vectors, \mathbf{a} and \mathbf{b} , and matrices, \mathbf{M} and \mathbf{N} , are defined as

$$\mathbf{a} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{N} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix}$$

Then

$$s = \mathbf{a}^T \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}^T \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = [1 \ 3 \ 5] \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = 2 \times 1 + 3 \times 4 + 5 \times 6 = 44$$

$$\mathbf{T} = \mathbf{a} \mathbf{b}^T = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}^T = \begin{bmatrix} 2 \times 1 & 4 \times 1 & 6 \times 1 \\ 2 \times 3 & 4 \times 3 & 6 \times 3 \\ 2 \times 5 & 4 \times 5 & 6 \times 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 6 & 12 & 18 \\ 10 & 20 & 30 \end{bmatrix}$$

$$\mathbf{c} = \mathbf{M} \mathbf{a} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \times 1 & + & 0 \times 3 & + & 2 \times 5 \\ 0 \times 1 & + & 1 \times 3 & + & 0 \times 5 \\ 2 \times 1 & + & 0 \times 3 & + & 1 \times 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \\ 7 \end{bmatrix}$$

$$\mathbf{P} = \mathbf{M} \mathbf{N} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 5 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

corresponds to

$$s = \mathbf{a}' * \mathbf{b};$$

$$\mathbf{T} = \mathbf{a} * \mathbf{b}';$$

$$\mathbf{c} = \mathbf{M} * \mathbf{a};$$

$$\mathbf{P} = \mathbf{M} * \mathbf{N};$$

(*MatLab* gda00_07)

In *MatLab*, matrix multiplication is signified using the multiplication sign, * (the asterisk). There are cases, however, where one needs to violate the rules and multiply the quantities element-wise (for example, create a vector, \mathbf{d} , with elements $d_i = a_i b_i$). *MatLab* provides a special element-wise version of the multiplication sign, denoted .* (a period followed by an asterisk)

$$\mathbf{d} = \mathbf{a} .* \mathbf{b};$$

(*MatLab* gda00_07)

As described above, individual elements of vectors and matrices can be accessed by specifying the relevant row and column indices, in parentheses, e.g., $\mathbf{a}(2)$ is the second element of the column vector \mathbf{a} , and $\mathbf{M}(2,3)$ is the second row, third column element of the matrix, \mathbf{M} . Ranges of rows and columns can be specified using the : (colon) operator, e.g., $\mathbf{M}(:,2)$ is the second column of matrix, \mathbf{M} ; $\mathbf{M}(2,:)$ is the second row of matrix, \mathbf{M} ; and $\mathbf{M}(2:3,2:3)$ is the 2×2 submatrix

in the lower right-hand corner of the 3×3 matrix, \mathbf{M} (the expression, $\mathbf{M}(2:\text{end}, 2:\text{end})$, would work as well). These operations are further illustrated below:

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$s = a_2 = 2 \quad \text{and} \quad t = M_{23} = 6 \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} M_{12} \\ M_{22} \\ M_{32} \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$$

$$\mathbf{c} = [M_{21} \quad M_{22} \quad M_{23}]^T = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} M_{22} & M_{23} \\ M_{32} & M_{33} \end{bmatrix} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$$

correspond to

$$\begin{aligned} s &= \mathbf{a}(2); \\ t &= \mathbf{M}(2, 3); \\ \mathbf{b} &= \mathbf{M}(:, 2); \\ \mathbf{c} &= \mathbf{M}(2, :)' ; \\ \mathbf{T} &= \mathbf{M}(2:3, 2:3); \end{aligned}$$

(*MatLab* gda00_08)

The colon notation can be used in other contexts in *MatLab* as well. For instance, $[1:4]$ is the row vector $[1, 2, 3, 4]$. The syntax, $1:4$, which omits the square brackets, works fine in *MatLab*. However, we will usually use square brackets, since they draw attention to the presence of a vector. Finally, we note that two colons can be used in sequence to indicate the spacing of elements in the resulting vector. For example, the expression $[1:2:9]$ is the row vector $[1, 3, 5, 7, 9]$ and that the expression $[10:-1:1]$ is a row vector whose elements are in the reverse order from $[1:10]$.

Matrix division is defined in analogy to reciprocals. If s is a scalar number, then multiplication by the reciprocal s^{-1} is equivalent to division by s . Here, the reciprocal obeys $s^{-1}s = ss^{-1} = 1$. The matrix analog to the reciprocal is called the *matrix inverse* and obeys

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

It is defined only for square matrices. The calculation of the inverse of a matrix is complicated, and we will not describe it here, except to mention the 2×2 case

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Just as the reciprocal s^{-1} is defined only when $s \neq 0$, the matrix inverse \mathbf{A}^{-1} is defined only when a quantity called the *determinant* of \mathbf{A} , denoted $\det(\mathbf{A})$, is not equal to zero. The determinant of a square $N \times N$ matrix \mathbf{M} is defined as

$$\det(\mathbf{M}) = \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \dots \sum_{q=1}^N \varepsilon^{ijk\dots q} M_{1i} M_{2j} M_{3k} \dots M_{Nq}$$

Here the quantity $\varepsilon^{ijk\dots q}$ is +1 when (i, j, k, \dots, q) is an even permutation of $(1, 2, 3, \dots, N)$, -1 when it is an odd permutation and zero otherwise. Note that the determinant of an $N \times N$ is the sum of products of N elements of the matrix. In the case of a 2×2 matrix, the determinant contains products of two elements and is given by

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

Note that the reciprocal of the 2×2 determinant appears in the formula for the 2×2 matrix inverse, implying that this matrix inverse does not exist when the determinant is zero. This is a general property of matrix inverses; they exist only when the matrix has nonzero determinant. In *MatLab*, the matrix inverse and determinant of a square matrix \mathbf{A} are computed as

```
B = inv(A);
d = det(A);
```

(*MatLab* gda00_09)

In many of the formulas of inverse theory, the matrix inverse either premultiplies or postmultiplies other quantities, for instance:

$$\mathbf{c} = \mathbf{A}^{-1}\mathbf{b} \quad \text{and} \quad \mathbf{D} = \mathbf{B}\mathbf{A}^{-1}$$

These cases do not actually require the explicit calculation of \mathbf{A}^{-1} , just the combinations $\mathbf{A}^{-1}\mathbf{b}$ and $\mathbf{B}\mathbf{A}^{-1}$, which are computationally simpler. *MatLab* provides generalizations of the division operator that implement these two cases:

```
c = A\b;
D = B/A;
```

(*MatLab* gda00_09)

A surprising amount of information on the structure of a matrix can be gained by studying how it affects a column vector that it multiplies. Suppose that \mathbf{M} is an $N \times N$ square matrix and that it multiplies an *input* column vector, \mathbf{v} , producing an *output* column vector, $\mathbf{w} = \mathbf{M}\mathbf{v}$. We can examine how the output \mathbf{w} compares to the input \mathbf{v} as \mathbf{v} is varied. One question of particular importance is

When is the output parallel to the input?

This question is called the *algebraic eigenvalue problem*. If \mathbf{w} is parallel to \mathbf{v} , then $\mathbf{w} = \lambda\mathbf{v}$, where λ is a scalar proportionality factor. The parallel vectors satisfy the following equation:

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v} \quad \text{or} \quad (\mathbf{M} - \lambda\mathbf{I})\mathbf{v} = 0$$

The trivial solution $\mathbf{v} = (\mathbf{M} - \lambda\mathbf{I})^{-1}\mathbf{0} = \mathbf{0}$ is not very interesting. A nontrivial solution is only possible when the matrix inverse $(\mathbf{M} - \lambda\mathbf{I})^{-1}$ does not exist. This is the case where the parameter λ is specifically chosen to make the determinant $\det(\mathbf{M} - \lambda\mathbf{I})$ exactly zero, since a matrix with zero determinant has no inverse. The determinant is calculated by adding together terms, each of which contains the product of N elements of the matrix. Since each element of the matrix contains, at most, one instance of λ , the product will contain powers of λ up to λ^N . Thus, the equation, $\det(\mathbf{M} - \lambda\mathbf{I}) = 0$, is an N th order polynomial equation for λ . An N th order polynomial equation has N roots, so we conclude that there must be N different proportionality factors, say λ_i , and N corresponding column vectors, say $\mathbf{v}^{(i)}$, that solve $\mathbf{M}\mathbf{v}^{(i)} = \lambda_i\mathbf{v}^{(i)}$. The column vectors, $\mathbf{v}^{(i)}$, are called the *characteristic vectors* (or *eigenvectors*) of the matrix, \mathbf{M} , and the proportionality factors, λ_i , are called the *characteristic values* (or *eigenvalues*). Eigenvectors are determined only up to an arbitrary multiplicative factor s , since if $\mathbf{v}^{(i)}$ is an eigenvector, so is $s\mathbf{v}^{(i)}$. Consequently, they are conventionally chosen to be unit vectors.

In the special case where \mathbf{M} is symmetric, it can be shown that the eigenvalues, λ_i , are real and the eigenvectors are mutually perpendicular, $\mathbf{v}^{(i)\text{T}}\mathbf{v}^{(j)} = 0$ for $i \neq j$. The N eigenvalues can be arranged into a diagonal matrix, $\mathbf{\Lambda}$, whose elements are $[\mathbf{\Lambda}]_{ij} = \lambda_i\delta_{ij}$, where δ_{ij} is the Kronecker delta. The corresponding N eigenvectors $\mathbf{v}^{(i)}$ can be arranged as the columns of an $N \times N$ matrix \mathbf{V} , which satisfies, $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$. The eigenvalue equation $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ can then be succinctly written as

$$\mathbf{M}\mathbf{V} = \mathbf{V}\mathbf{\Lambda} \quad \text{or} \quad \mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

(The second equation is derived from the first by postmultiplying it by \mathbf{V}^T .) Thus, the matrix \mathbf{M} can be reconstructed from its eigenvalues and eigenvectors. In *MatLab*, the matrix of eigenvalues $\mathbf{\Lambda}$ and matrix of eigenvectors \mathbf{V} of a matrix \mathbf{M} are computed as

$$[\mathbf{V}, \text{LAMBDA}] = \text{eig}(\mathbf{M});$$

(*MatLab* gda00_09)

Here the eigenvector matrix is called `LAMBDA`.

Many of the derivations of inverse theory require that a column vector \mathbf{v} be considered a function of an independent variable, say x , and then differentiated with respect to that variable to yield the derivative $d\mathbf{v}/dx$. Such a derivative represents the fact that the vector changes from \mathbf{v} to $\mathbf{v} + d\mathbf{v}$ as the independent variable changes from x to $x + dx$. Note that the resulting change $d\mathbf{v}$ is itself a vector. Derivatives are performed element-wise; that is,

$$\left[\frac{d\mathbf{v}}{dx} \right]_i = \frac{dv_i}{dx}$$

A somewhat more complicated situation is where the column vector \mathbf{v} is a function of another column vector, say \mathbf{y} . The partial derivative

$$\frac{\partial v_i}{\partial y_j}$$

represents the change in the i th component of \mathbf{v} caused by a change in the j th component of \mathbf{y} . Frequently, we will need to differentiate the linear function, $\mathbf{v} = \mathbf{M}\mathbf{y}$, where \mathbf{M} is a matrix, with respect to \mathbf{y} :

$$\frac{\partial v_i}{\partial y_j} = \frac{\partial}{\partial y_j} \left[\sum_k M_{ik} y_k \right] = \sum_k M_{ik} \frac{\partial y_k}{\partial y_j}$$

Since the components of \mathbf{y} are assumed to be independent, the derivative $\frac{\partial y_k}{\partial y_j}$ is zero except when $j=k$, in which case it is unity, which is to say $\frac{\partial y_k}{\partial y_j} = \delta_{kj}$. The expression for the derivative then simplifies to

$$\frac{\partial v_i}{\partial y_j} = \frac{\partial}{\partial y_j} \left[\sum_k M_{ik} y_k \right] = \sum_k M_{ik} \frac{\partial y_k}{\partial y_j} = \sum_k M_{ik} \delta_{kj} = M_{ij}$$

Thus the derivative of the linear function $\mathbf{v} = \mathbf{M}\mathbf{y}$ is the matrix \mathbf{M} . This relationship is the vector analogue to the scalar case, where the linear function $v = my$ has the derivative $\frac{dv}{dy} = m$.

1.5 USEFUL MATLAB OPERATIONS

1.5.1 Loops

MatLab provides a looping mechanism, the `for` command, which can be useful when the need arises to sequentially access the elements of vectors and matrices. Thus, for example,

```
M = [ [1, 4, 7]', [2, 5, 8]', [3, 6, 9]' ];
for i = [1:3]
    a(i) = M(i,i);
end
```

(*MatLab* gda00_10)

executes the `a(i)=M(i,i)` formula three times, each time with a different value of i (in this case, $i=1$, $i=2$, and $i=3$). The net effect is to copy the diagonal elements of the matrix \mathbf{M} to the vector, \mathbf{a} , that is, $a_i = M_{ii}$. Note that the `end` statement indicates the position of the bottom of the loop. Subsequent commands are not part of the loop and are executed only once.

Loops can be nested; that is, one loop can be inside another. Such an arrangement is necessary for accessing all the elements of a matrix in sequence. For example,

```
M = [ [1, 4, 7]', [2, 5, 8]', [3, 6, 9]' ];
for i = [1:3]
```

```

for j = [1:3]
    N(i,4-j) = M(i,j);
end
end

```

(MatLab gda00_11)

copies the elements of the matrix, \mathbf{M} , to the matrix, \mathbf{N} , but reverses the order of the elements in each row, that is, $N_{i,4-j} = M_{i,j}$. Loops are especially useful in conjunction with *conditional* commands. For example,

```

a = [ 1, 2, 1, 4, 3, 2, 6, 4, 9, 2, 1, 4 ]';
for i = [1:12]
    if ( a(i) >= 6 )
        b(i) = 6;
    else
        b(i) = a(i);
    end
end
end

```

(MatLab gda00_12)

sets $b_i = a_i$ if $a_i < 6$ and sets $b_i = 6$, otherwise (a process called *clipping* a vector, for it lops off parts of the vector that are larger than 6).

A purist might point out that *MatLab* syntax is so flexible that `for` loops are almost never really necessary. In fact, all three examples, above, can be computed with one-line formulas that omit `for` loops:

```

a = diag(M);
N = fliplr(M);
b=a; b(find(a>6))=6;

```

(MatLab gda00_13)

The first two formulas are quite simple, but rely upon the *MatLab* functions `diag()` (for “diagonal”) and `fliplr()` (for “flip left-right”), whose existence we have not hitherto mentioned. The third formula, which used the `find()` function, requires further explanation. The first part just copies the column vector \mathbf{a} to \mathbf{b} . In the second part, the $(a > 6)$ operation returns a vector of zeros and ones, depending upon whether the elements of the column vector \mathbf{a} satisfy the inequality or not. The `find()` function uses this result and returns a list of the *indices* of the ones, that is, of the indices of the column vector \mathbf{a} that match the condition. This list is then used to reset just those elements of \mathbf{b} to 6, leaving the other elements unchanged.

One of the problems of a script-based environment is that learning the complete syntax of the scripting language can be pretty daunting. Writing a long script, such as one containing a `for` loop, will often be faster than searching through *MatLab* help files for a predefined function that implements the desired functionality in a single line of the script. When deciding between alternative

ways of implementing a given functionality, you should always choose the one which *you* find clearest. Scripts that are terse or even computationally efficient are not necessarily a virtue, especially if they are difficult to debug. You should avoid creating formulas that are so inscrutable that you are not sure whether they will function correctly. Of course, the degree of inscrutability of any given formula will depend upon your level of familiarity with *MatLab*. Your repertoire of techniques will grow as you become more practiced.

1.5.2 Loading Data from a File

MatLab can read and write files with a variety of formats, but we start here with the simplest and most common, the text file. As an example, we load a global temperature dataset compiled by the National Aeronautics and Space Administration. The author's recommendation is that you always keep a file of notes about any data set that you work with, and that these notes include information on where you obtained the data set and any modifications that you subsequently made to it:

The text file `global_temp.txt` contains global temperature change data from NASA's web site <http://data.giss.nasa.gov/gistemp>. It has two columns of data, time (in calendar years) and temperature anomaly (in degrees C) and is 46 lines long. Information about the data is in the file `global_temp_notes.txt`. The citation for this data is [Hansen et al. \(2010\)](#).

We reproduce the first few lines of `global_temp.txt`, here:

```
1965 -0.11
1966 -0.03
1967 -0.01
...   ...
```

The data are read into to *MatLab* as follows:

```
D = load('../data/global_temp.txt');
t = D(:,1);
d = D(:,2);
```

(*MatLab* gda00_14)

The `load()` function reads the data into a 46×2 matrix, **D**. Note that the filename is given as `../data/global_temp.txt`, as contrasted to just `global_temp.txt`, since the script is run from the `ch00` folder while the data are in the `data` folder. The filename is surrounded by single quotes to indicate that it is a *character string* and not a variable name. The subsequent two lines break out **D** into two separate column vectors, **t** of time and **d** of temperature data. This step is not strictly speaking necessary, but fewer mistakes will be made if the different variables in the dataset have each their own name.

The `figure(1)` command directs *MatLab* to create a figure window and to label it Figure 1. The `clf` (for clear figure) command erases any previous plots in the figure window, ensuring that it is blank. The default line width of plot axes in *MatLab* is one point, too thin to show up clearly in printed documents. We increase it to three points using the `set()` command. The `hold on` command instructs *MatLab* to overlay multiple plots in the same window, overriding the default that a second plot erases the first. The `axis([1965, 2010, -0.5, 1.0])` command manually sets the axes to an appropriate range. If this command is omitted, *MatLab* chooses the scaling of the axes based on the range of the data. The two `plot(...)` commands plot the time t and temperature d data in two different ways: first as a red line (indicated by the `'r-'`) and the second with black circles (the `'ko'`). We also increase the width of the lines with the `'LineWidth', 3` directive. Finally, the horizontal and vertical axes are labeled using the `xlabel()` and `ylabel()` commands, and the plot is titled with the `title()` command. Note that the text is surrounded with single quotes, to indicate that it is a character string.

1.5.4 Creating Character Strings Containing the Values of Variables

The results of computations are most understandable if described in a combination of words and numbers. The `sprintf()` function (for “string print formatted”) creates a character string that includes both text and the value of a variable. The string can then be used as a plot label, a filename, as text displayed in the Command Window, or for a wide variety of similar purposes. Thus, for instance,

```
title(sprintf('temperature data from %d to %d', t(1), t(N)));
(MatLab gda00_15)
```

creates a plot title `'temperature data from 1965 to 2010.'` The `sprintf()` command can be used in any function that expects a character string, including `title()`, `xlabel()`, `ylabel()`, `load()`, and the function `disp()`, not mentioned until now, which writes a character string to the Command Window. The `sprintf()` function is fairly inscrutable, and we refer readers to the *MatLab* help pages for a detailed description. Briefly, the function uses *placeholders* that start with the character `%` to indicate where in the character string the value of the variable should be placed. Thus,

```
disp(sprintf('Number of data: %d', N));
(MatLab gda00_15)
```

writes the character string `'Number of data: 46'` (since the value of `N` was 46) to the Command Window. The `%d` is the *placeholder for an integer*. It is replaced

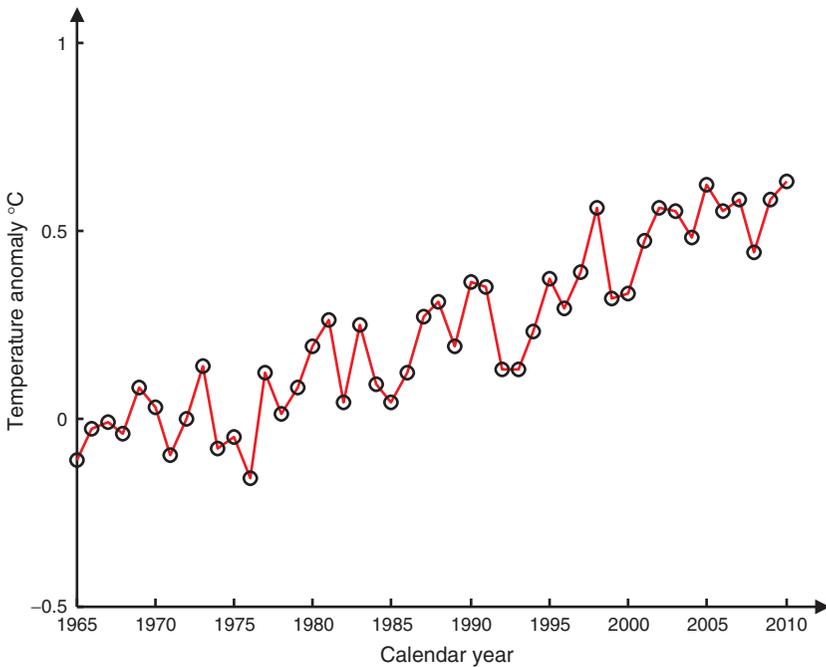


FIGURE I.3 Global temperature data for the time period 1965–2010. See text for further discussion. *MatLab* script gda00_14.

1.5.3 Plotting Data

MatLab's plotting commands are very powerful, but they are also very complicated. We present here a set of commands for making a simple x - y plot that is intermediate between a very crude, unlabeled plot, and an extremely artistic one. The reader may wish to adopt either a simpler or a more complicated version of this set, depending upon need and personal preference. The plot of the global temperature data shown in [Figure I.3](#) was created with the commands:

```
figure(1);
clf;
set(gca, 'LineWidth', 3);
hold on;
axis( [1965, 2010, -0.5, 1.0] );
plot(t,d, 'r-', 'LineWidth', 3);
plot(t,d, 'ko', 'LineWidth', 3);
xlabel('calendar year');
ylabel('temperature anomaly, deg C');
ylabel('temperature anomaly, deg C');
title('global temperature data');
```

(*MatLab* gda00_14)

with '46,' the value of N . If the variable is fractional, as contrasted to integer, the *floating-point placeholder*, %f, is used instead. For example,

```
% display first few lines
disp('first few lines of data');
for i=[1:4]
    disp(sprintf('%f %f',t(i),d(i)));
end
```

(MatLab gda00_15)

writes the first four lines of data to the Command Window. Note that two placeholders have been used in the same *format string*. When displaying text to the Command Window, an alternative function is available that combines the functionality of `sprintf()` and `disp()`

```
fprintf('%f %f\n',t(i),d(i));
```

which is equivalent to

```
disp(sprintf('%f %f',t(i),d(i)));
```

The `\n` (for *newline*) at the end of the format string '`a=%f\n`' indicates that subsequent characters should be written to a new line in the command window, rather than being appended onto the end of the current line.

REFERENCES

- Hansen, J., Ruedy, R., Sato, Mki., Lo, K., 2010. Global surface temperature change. *Rev. Geophys.* 48, RG4004.
- Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc, Oxford, UK, 263pp.
- Part-Enander, E., Sjoberg, A., Melin, B., Isaksson, P., 1996. *The Matlab Handbook*. Addison-Wesley, New York, 436pp.
- Pratap, R., 2009. *Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers*. Oxford University Press, Oxford, 256pp.

Preface

For now we see through a glass, darkly, but then . . .

Paul of Tarsus

Every researcher in the applied sciences who has analyzed data has practiced inverse theory. Inverse theory is simply the set of methods used to extract useful inferences about the world from physical measurements. The fitting of a straight line to data involves a simple application of inverse theory. Tomography, popularized by the physician's CT and MRI scanners, uses it on a more sophisticated level.

The study of inverse theory, however, is more than the cataloging of methods of data analysis. It is an attempt to organize these techniques, to bring out their underlying similarities and pin down their differences, and to deal with the fundamental question of the limits of information that can be gleaned from any given data set.

Physical properties fall into two general classes: those that can be described by discrete parameters (e.g., the mass of the earth or the position of the atoms in a protein molecule) and those that must be described by continuous functions (e.g., temperature over the face of the earth or electric field intensity in a capacitor). Inverse theory employs different mathematical techniques for these two classes of parameters: the theory of matrix equations for discrete parameters and the theory of integral equations for continuous functions.

Being introductory in nature, this book deals mainly with "discrete inverse theory," that is, the part of the theory concerned with parameters that either are truly discrete or can be adequately approximated as discrete. By adhering to these limitations, inverse theory can be presented on a level that is accessible to most first-year graduate students and many college seniors in the applied sciences. The only mathematics that is presumed is a working knowledge of the calculus and linear algebra and some familiarity with general concepts from probability theory and statistics.

Nevertheless, the treatment in this book is in no sense simplified. Realistic examples, drawn from the scientific literature, are used to illustrate the various techniques. Since in practice the solutions to most inverse problems require substantial computational effort, attention is given to how realistic problems can be solved.

The treatment of inverse theory in this book is divided into four parts. [Chapters 1 and 2](#) provide a general background, explaining what inverse problems are and what constitutes their solution as well as reviewing some of the basic concepts from linear algebra and probability theory that will be applied throughout the text. [Chapters 3–7](#) discuss the solution of the canonical inverse problem: the linear problem with Gaussian statistics. This is the best understood of all inverse problems, and it is here that the fundamental notions of uncertainty, uniqueness, and resolution can be most clearly developed. [Chapters 8–11](#) extend the discussion to problems that are non-Gaussian, non-linear, and continuous. [Chapters 12–13](#) provide examples of the use of inverse theory and a discussion of the steps that must be taken to solve inverse problems on a computer.

MatLab scripts are used throughout the book as a means of communicating how the formulas of inverse theory can be used in computer-based data processing scenarios. *MatLab* is a commercial software product of *The MathWorks, Inc.* and is widely used in university settings as an environment for scientific computing. All of the book's examples, its recommended homework problems, and the case studies of [Chapter 12](#) use *MatLab* extensively. Further, all the *MatLab* scripts used in the book are made available to readers through the book's Web site. The book is self-contained; it can be read straight through, and profitably, even by someone with no access to *MatLab*. But it is meant to be used in a setting where students are actively using *MatLab* both as an aid to studying (that is, by reproducing the examples and case studies described in the book) and as a tool for completing the recommended homework problems.

Many people helped me write this book. I am very grateful to my students at Columbia University and at Oregon State University for the helpful comments they gave me during the courses I have taught on inverse theory. Mike West, of the Alaska Volcano Observatory, did much to inspire this revision of the book, by inviting me to teach a mini-course on the subject in the fall of 2009. The use of *MatLab* in this book parallels the usage in *Environmental Data Analysis with MatLab* (Menke and Menke, 2011), a data analysis textbook that I wrote with my son Joshua Menke in 2011. The many hours we spent working together on its tutorials taught us both a tremendous amount about how to use that software in a pedagogical setting. Finally, I thank the many hundreds of scientists and mathematicians whose ideas I drew upon in writing this book.

REFERENCE

Menke, W., Menke, J., 2011. *Environmental Data Analysis with MatLab*. Academic Press, Elsevier Inc, Oxford UK, 263pp.

COMPANION WEB SITE

<http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9780123971609>.

Academic Press is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands

Third edition **2012**

Copyright © Elsevier Inc. 2012, 1989, 1984. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of the publisher. Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; email: permissions@elsevier.com. Alternatively you can submit your request online by visiting the Elsevier web site at <http://elsevier.com/locate/permissions>, and selecting Obtaining permission to use Elsevier material.

Notice

No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

Library of Congress Cataloging-in-Publication Data

Menke, William.

Geophysical data analysis : discrete inverse theory / William Menke. – MatLab ed., 3rd ed.
p. cm.

Includes bibliographical references and index.

ISBN 978-0-12-397160-9 (hardback)

1. Geophysics—Measurement. 2. Oceanography—Measurement. 3. Inverse problems (Differential equations)—Numerical solutions. I. Title.

QC802.A1M46 2012

551—dc23

2012000457

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

For information on all Academic Press publications
visit our web site at store.elsevier.com

Printed and bound in China

12 13 14 15 16 10 9 8 7 6 5 4 3 2 1

ISBN: 978-0-12-397160-9

Working together to grow
libraries in developing countries

www.elsevier.com | www.bookaid.org | www.sabre.org

ELSEVIER

BOOK AID
International

Sabre Foundation

Geophysical Data Analysis: Discrete Inverse Theory

Geophysical Data Analysis: Discrete Inverse Theory

MATLAB Edition

William Menke

Lamont-Doherty Earth Observatory and
Department of Earth and Environmental Sciences
Columbia University
Palisades, New York



ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO • SYDNEY • TOKYO

Academic Press is an imprint of Elsevier



Note: Page numbers followed by *f* indicate figures and *t* indicate tables.

A

- A posteriori* variance, 63
- A priori* constraint, 61, 237–238, 272
- A priori* distributions, 175–177
 - maximum likelihood methods, 92–97
- A priori* equations, 61
- A priori* information, 52–53, 55–56
 - length as, 56–60
 - model parameters, 93*f*, 94*f*, 95*f*, 99*f*, 100*f*, 134–135
 - variance, 107–108
- model vector, 57
- nonunique averaging vectors, 119–121
- smoothness, 59
- types of, 60–63
- A priori* probability density function, 94, 96
 - joint, 98*f*
 - value of, 177*f*
- A priori* variance, 63
- Absolute value, 41*f*
- Absorption peaks, 9*f*
- Acoustic tomography, 6–7
 - with deficient distribution of rays, 242*f*
- problems, 240–241
- resolution of, 87*f*
- success of, 241
- true image, 241*f*
- Acoustic travel time experiments, 51*f*
- Acoustic waves, 6*f*
- Adjoint, 216
 - operator, 219
- Airgun signal, 236*f*
- Anderson, D. L., 268
- Anorthite, 195
- Approximate ray-steeping method, 240–241
- Armijo's rule, 180–181
- Asymptotes, 86*f*
- Atlantic Rock dataset, 193
 - coefficients in, 194*f*

- singular value decomposition of, 198*f*
 - singular values of, 193*f*
- Attenuation factor, 269–270
- Attenuation tomography, 270
- Autocorrelation, 235
- Auxiliary variable, 186
- Averages
 - estimates and, 118–119
 - global temperature, 4*f*

B

- Backprojection, 219–222
 - example, 222*f*
- Back-solving, 125
- Backus-Gilbert generalized inverse, 243
 - for temperature distribution problem, 244*f*
 - for underdetermined problem, 79–82
- Backus-Gilbert inverse problem, 207–209, 234
- Backus-Gilbert solutions, 82
 - Dirichlet solution compared with, 82*f*
 - trade-off curves in, 85*f*
- Backus-Gilbert spread function, 78–79
- Banana-doughnut kernel, 267
- Bayes Theorem, 29, 31–32
- Bayesian Inference, 33, 101
- bigg () function, 45
- Biconjugate gradient algorithm, 45, 59, 235–236, 240–241
- Blur, removing, 232*f*, 233*f*
- Boltzmann probability density function, 182–183
- Booker, J. R., 262–263
- Bootstrap confidence intervals, 185–186
 - for model parameters, 187*f*
- Bootstrap method, 180
- Bordering method, 81–82
- Boundary conditions, 215–216
- Bounding values, 12
 - nonunique averages of model parameters, 137

- Bounds, 119
 - finite, 152
 - on weighted averages of model parameters, 120f
- Boxcar filters, 232f

C

- Calculus, fundamental theorem of, 215–216
- Cameras, 231
- Cartesian unit vectors, 249
- Central limit theorem, 26–27
- Centroid, 264–265
- Certainty, 97–99
- Characteristic frequencies, 256
- Chave, A. D., 270–271
- Checkerboard test, 87
- Chemical elements, 189
- Chi-squared probability density function, 29
- Circular random variable, 279
- Cluster analysis, 199
- CMT. *See* Global Centroid Moment Tensor
- Coefficients, 194f
- Collinearity, 43
- Columns, 128
- Combining, 100–101
 - probability density functions, 101
- Completely overdetermined problem, 154–155
- Completely underdetermined problems, 154
- Complex numbers, 278–280
- Computed tomography (CT), 7f
- Conditional distributions, 32
- Conditional probability density functions, 30–33
 - example of, 32f
 - inexact theory represented by, 101f
- Confidence intervals, 33–34
 - estimates, 186
- Constant velocity structure, 257
- Constrained least squares problem, 127
- Constraints
 - a priori*, 61, 237–238, 272
 - inequality, 120, 140–147
 - least squares with, 144–145
 - simplifying, 138–140
 - Lagrange multiplier implementation of, 277–278
 - linear equality, 139
 - simplifying, 138–140
 - linear inequality, 62–63, 139–140
 - half-spaces in, 140f, 142f
 - physicality, 194–199
- Continuous inverse problems
 - differential equation and, 224f

- discrete problems as, 209–211
 - linear, 210f
 - solution of, 220f
- Continuous inverse theory, 3
 - tomography and, 211–212
- Convolution, 234
- Correlated data, 19–21
- Covariance, 19–20. *See also* Variance
 - of estimated model parameters, 134–135
 - of generalized inverses, 74–75, 76–78
 - overdetermined case, 76–77
 - underdetermined case, 77
 - goodness of resolution measures and, 75
 - of least squares solution, 66
 - matrix, 21, 105
 - sample, 21
 - size, 83–84
- Cross-correlation, 235, 266
- Crossover errors, 237
 - adjustment of, 236–240
 - satellite gravity data, 238f
- CT. *See* Computed tomography
- Cumulative distribution function, 16
- Curve fitting
 - Gaussian, 250–252
 - L_∞ norm, 160f
 - $L1$ norms, 156f
 - nonlinear inverse problems
 - gradient method for, 181f
 - grid search for, 169f
 - Monte Carlo search for, 171f
 - simulated annealing for, 184f
 - spectral, 9–10

D

- Dalton, C. A., 270
- Damped least squares, 56, 238
 - weighted, 58–60
 - true model, 60f
- Damped minimum length, 78
 - trade-off curves in, 85f
- Damping parameters, 238
- Data assimilation, 227
- Data kernels, 135f, 211, 225–226, 226f, 232
 - spectrum of, 135–136
- Data spaces, 123
- Density function, 90f
- Derivatives
 - approximations, 56
 - computing, 44
 - finite element, 254
 - Fréchet, 218, 219–220, 268f, 269
 - differential equations and, 222–227

- of error, 218–219
 - partial, 174
 - singular-value decomposition, 138
 - Determinants, Jacobian, 23
 - Deviation, 60
 - Diagonal elements, 238–239
 - Differential equations, 225–226
 - continuous inverse problems, 224*f*
 - Fréchet derivatives and, 222–227
 - Digital filter design, 234–236
 - Dirac delta function, 103, 211–212, 223
 - Dirichlet spread functions, 75
 - Backus-Gilbert solution compared with, 82*f*
 - generalized inverse and, 77–78
 - Discrete Fourier transforms, 214
 - Discrete inverse theory, 3
 - Discrete problems, 210
 - continuous inverse problems as, 209–211
 - Dispersion function, 268
 - Displacement, 264–265
 - Distributions
 - a priori*, 175–177
 - maximum likelihood methods, 92–97
 - conditional, 32
 - cumulative, function, 16
 - exponential, 150*r*
 - Fisher, 248*f*
 - Gaussian, 91, 151–152
 - area beneath, 150*r*
 - long-tailed, 42
 - normal, with zero mean, 27*f*
 - null, 102
 - for inexact non-Gaussian nonlinear theories, 184–185
 - of rays, 242*f*
 - temperature
 - Backus-Gilbert for, 244*f*
 - minimum length for, 244*f*
 - model for, 244*f*
 - one-dimensional, 241–244
 - univariate, 24*f*, 32
 - Dot products, 216
 - Double-difference method, 263–264
 - Dziewonski, A. M., 268
- E**
- Earthquakes, 50*f*
 - location, 252–256
 - example, 255*f*
 - velocity structure, 261–264
 - moment tensors of, 264–265
 - Eigenfrequencies, 256, 257, 267
 - Eigenfunctions, 267
 - Eigenvalues, 130, 191–192
 - decomposition, 137–138
 - variance and, 136–137
 - Eigenvectors, 130
 - with large singular values, 192
 - matrix of, 132
 - with nonzero singular values, 191
 - null, 137
 - singular-value decomposition, 194–195
 - Eikonal equation, 261–262
 - Ekström, G., 270
 - El Niño-Southern Oscillation, 202–204
 - Electromagnetic induction, 273
 - Ellipse, 65*f*
 - Empirical orthogonal function analysis, 199–204
 - image sequence, 204*f*
 - Entropy, 94–96
 - relative, 94–96
 - maximum, 108–109
 - minimum, 108–109
 - erf(), 242
 - Erosion, 190*f*
 - Error
 - crossover, 237
 - adjustment of, 236–240
 - satellite gravity data, 238*f*
 - Fréchet derivative of, 218–219
 - function, 242
 - gradient method and, 220*f*
 - maps, 63
 - in nonlinear inverse problems, 166–167
 - prediction, 41*f*, 131, 137–138, 139–140, 157
 - as function of model parameters, 168*f*
 - L2*, 167*f*
 - of least squares solution, 64–66
 - for lines through points, 50*f*
 - minimizing, 55–56, 146*f*
 - propagation, 25–26, 64
 - Estimates, 11
 - averages and, 118–119
 - confidence intervals, 186
 - lengths of, 39
 - of model parameters, 11, 21–22, 65*f*, 134, 249
 - covariance of, 134–135
 - variance of, 63–64
 - unbiased, 165
 - Euler vector, 271
 - Euler-Lagrange method, 109–110
 - Even-determined problems, least square, 52
 - Exact data, 107–108
 - Exact linear theory, 178–179
 - Exact theory, 107–108
 - maximum likelihood for, 97–100

Expected measurements, 17
 Expected value, 18
 Explicit form, 2
 Explicit linear inverse problem, 106
 Exponential distribution, 150r
 Exponential functions, 166f
 Exponential probability density functions,
 149–150, 246–248
 comparison of, 150f
 in *MatLab*, 150
 maximum likelihood estimate of mean of,
 151–153
 two-sided, 149

F

F ratio, 112
 Factors, 189
 analysis, 10–11, 189–194
 Q-mode, 199
 R-mode, 199
 loadings, 191, 194–195, 198–199, 200–201,
 201f
 matrix, 199
 mutually perpendicular, 196f
 Faults, 252–253
 fcdf () function, 112
 Feasible half-space, 140
 Fermat's Principle, 264
 filterfun () function, 235
 find () function, 186
 Finite bounds, 152
 Finite element derivatives, 254
 First order terms, 65–66
 Fisher distribution, 248f
 Fisher probability density function, 246–248
 Fitting
 curve
 Gaussian, 250–252
 L_∞ norm, 160f
 $L1$ norms, 156f
 nonlinear inverse problems, 169f, 171f,
 181f, 184f
 spectral, 9–10
 least squares
 to exponential functions, 166f
 parabola, 47–48
 plane surface, 48–49
 of straight lines, 40f, 43–44, 62f, 65f, 74f
 Lorentzian curves, 250–252
 parabolas, 5
 least squares solution, 47–48
 straight lines, 4–5, 41f, 62–63
 constrained, 62–63
 L_∞ norm, 245–246
 $L1$ norm, 245–246
 $L2$ norm, 245–246
 least square, 40f, 43–44, 62f, 65f, 74f
 Flatness, 56–57
 Force-couples, 264–265
 Forcing, 226–227
 unknown, 227
 Forward transformations, 129
 Fosterite, 195
 Fourier slice theorem, 213–214
 Fourier transforms
 discrete, 214
 integral, 214
 Fréchet derivatives, 218, 219–220,
 268f, 269
 differential equations and, 222–227
 of error, 218–219
 Free oscillations, 267–269, 270
 F-test, 111–112
 Functions. *See also specific functions*
 analysis, 199–204, 204f
 error, 242
 of random variables, 21–26
 two-dimensional, 202f
 Fundamental theorem of calculus, 215–216

G

Gap filling problem, 147
 Gaussian curve fitting, 250–252
 Gaussian data
 implicit nonlinear inverse problem with,
 175–180
 linear theory and, 172
 nonlinear theory and, 172–173
 Gaussian distribution, 91, 151–152
 area beneath, 150r
 Gaussian joint probability density
 function, 32f
 Gaussian multivariate probability density
 functions, 34
 Gaussian probability density functions, 26–29,
 36f, 97–99
 Gaussian random variables, 29
 Gaussian statistics
 assumption of, 29–30
 least squares and, 42
 Geiger, L., 254
 Geiger's method, 254, 261
 General linear Gaussian case, 104–107
 General linear problem, 153

- Generalized inverse, 69
 Backus-Gilbert, 243
 for temperature distribution problem, 244f
 for underdetermined problem, 79–82
 covariance of, 74–75
 Dirichlet spread function and, 77–78
 with good resolution and covariance,
 76–78
 overdetermined case, 76–77
 underdetermined case, 77
 least squares, 74
 minimum length, 75
 natural, 133
 resolution of, 74–75
 singular-value decomposition, 132–138
- Geological Survey, U.S., 265
- Geomagnetism, 271–272
- Gilbert, F., 270
- Global Centroid Moment Tensor (CMT), 265
- Global minimum, 173f, 174f
- Global Positioning System, 271
- Gradient method, 180–181, 181f
 error and, 220f
- Gravity
 geomagnetism and, 271–272
 Newton's inverse square law of, 143f
 satellite, 238f
 vertical force of, 143f
- Green function integral, 215–216,
 223, 224
- Grid search, 167–170, 169f
- ## H
- Half-space
 feasible, 140
 infeasible, 140
 in linear inequality constraints,
 140f, 142f
- Handles, 45–46
- Hanson, D. J., 143
- Haxby, Bill, 238f
- Heating functions, 224
- Heaviside step function, 210, 217, 225
- Hermitian matrices, 279
- Hermitian transpose, 279
- High variance, 195
- hist () function, 186
- Histograms, 16f
 of random variables, 37f
- Householder transformation, 124–127, 280
 designing, 127–129
- Hyperplane, 123
- Hypocenter, 252–253, 261
- Hypoinverse, 262
- ## I
- Identity matrix, 72
- Identity operators, 220
- Image enhancement problem, 231–234
 blur removal, 232f, 233f
- Image sequence, 203f
 empirical orthogonal functions, 204f
- Impedance, 273
- Implicit linear form, 2
- Implicit linear inverse problem, 106
- Implicit nonlinear inverse problem, with
 Gaussian data, 175–180
- Importance, 71
- Indefinite integrals, 221
- Index vectors, 84
- Inequality constraints, 120, 140–147
 least squares with, 144–145
 simplifying, 138–140
- Inexact non-Gaussian nonlinear theories,
 184–185
- Inexact theories
 conditional probability density function for,
 101f
 infinitely, 108
 maximum likelihood for, 100–102, 103f
- Infeasible half-space, 140
- Infinitely inexact theory, 108
- Information gain, 94–96
- Inner product, 216
- Integral equation theory, 3
- Integral Fourier transforms, 214
- Inverse problems, 150f
 Backus-Gilbert, 207–209, 234
 continuous
 differential equations, 224f
 discrete problems as, 209–211
 linear, 210f
 even-determined, 52
 explicit form, 2
 explicit linear, 3, 106
 formulating, 1–3
 acoustic tomography, 6–7
 examples, 4–5
 factor analysis, 10–11
 fitting parabola, 5
 fitting straight line, 4–5
 spectral curve fitting, 12
 X-ray imaging, 7–9
 implicit linear, 2, 106

Inverse problems (*Continued*)

- linear, 3–4
 - general, 153
 - L_2 prediction error in, 167*f*
 - least squares solution for, 44–46
 - least squares solution of, 44–46
 - MatLab*, 232–233
 - maximum likelihood methods, 92–108
 - a priori* distributions, 92–97
 - simplest case, 92
 - mixed-determined, 52, 54–56
 - natural solution to, 133
 - nonlinear
 - curve-fitting, 169*f*, 171*f*, 181*f*, 184*f*
 - error in, 166–167
 - implicit, 175–180
 - likelihood in, 166–167
 - MatLab* for, 168–170
 - Newton's method, 171–175, 176*f*
 - nonunique, 53
 - overdetermination of, 43, 52
 - completely, 154–155
 - linear programming problem and, 158
 - Newton's method for, 180
 - overdetermined, 126
 - bricks, 51–52
 - completely, 154–155
 - of inverse problems, 43
 - least square, 52
 - linear programming problems, 158
 - n generalized inverse, 76–77
 - Newton's method for, 180
 - partitioning, 263
 - premultiplying, 263
 - purely underdetermined, 52–54
 - simple, 116
 - simple solutions, 53–54
 - solutions to, 11–13
 - underdetermined, 51–52
 - Backus-Gilbert generalized inverse and, 79–82
 - completely, 154
 - in generalized inverse, 77
 - model parameters, 121*f*
 - purely, 52–54
- Inverse transformations, 129
- Iterative formula, 179
- Iterative method, 173*f*

J

- Jacobian determinant, 23, 213
- Jacobson, R. S., 270
- Joint probability function, 32, 104*f*, 249

K

- Kepler's third law, 48
 - tests of, 48*f*
- Knowledge, 1
- Kronecker delta, 44, 256–257
- Kuhn-Tucker theorem, 140–141, 142, 145
- Kurile subduction zone, 50*f*, 250*f*

L

- L_∞ norms, 158–160
 - curve fitting using, 160*f*
 - straight line fitting, 245–246
- L_1 norms, 149–150
 - curve fitting using, 156*f*
 - solving, 153–157
 - straight line fitting, 245–246
- L_2 inverse theory, 278–280
- L_2 norms, 40–42, 53–54, 92, 111, 126, 137–138
 - straight line fitting, 245–246
- L_2 prediction error, 167*f*
- Lagrange function, 80–81
- Lagrange multipliers, 54, 61, 80–81, 239–240, 249
 - equations, 178
 - graphical interpretation of, 278*f*
 - implementing constraints with, 277–278
- Lamond-Doherty Earth Observatory, 238*f*
- Lanczos, C., 138
- Laplace transforms, 72, 82
- Lawson, C. L., 143
- Least squares, 156
 - constrained, Householder transformations and, 127
 - covariance of, 66
 - damped, 56, 238
 - weighted, 58–60, 60*f*
 - even-determined problems, 52
 - existence of, 49–52
 - filter, 235–236
 - fitting
 - to exponential functions, 166*f*
 - parabola, 47–48
 - plane surface, 48–49
 - of straight lines, 40*f*, 43–44, 62*f*, 65*f*, 74*f*
 - Gaussian statistics and, 42
 - generalized inverse and, 74
 - with inequality constraints, 144–145
 - linear inverse problem and, 44–46
 - matrix products required by, 46, 47–48
 - nonnegative, 141–143, 144–145
 - prediction error of, 64–66
 - simple, 179
 - for straight lines, 43–44

- undetermined problems, 51–52
 - variance of, 64–66
 - leastsquaresfcn () function, 45–46
 - Length
 - as *a priori* information, 56–60
 - of estimates, 39
 - measures, 39–43
 - minimum, 243
 - damped, 78, 85*f*
 - trade-off curves in, 85*f*
 - generalized inverse, 75
 - for temperature distribution problem, 244*f*
 - preservation of, 124–127, 129–130
 - weighted minimum, 58
 - Likelihood, 166–167
 - function, 90–91
 - surface, 91*f*
 - Linear algebra, 81–82
 - Linear continuous inverse problem, 210*f*
 - Linear equality constraints, 139
 - simplifying, 138–140
 - Linear equations, 231
 - Linear inequality constraint, 62–63, 139–140
 - half-spaces in, 140*f*, 142*f*
 - Linear inverse problem, 3–4
 - general, 153
 - L_2 prediction error in, 167*f*
 - least squares solution of, 44–46
 - Linear operators, 220*f*
 - matrices and, 214–218
 - Linear programming problem, 120, 154–155
 - for overdetermined case, 158
 - Linear theory
 - exact, 178–179
 - Gaussian data and, 172
 - simple Gaussian case with, 102–104
 - Linearizing transformations, 165–166
 - Lines. *See also* Straight lines
 - through a single point, 50*f*
 - parallel, 214*f*
 - linprog () function, 120–121
 - Loading matrix, 191
 - Local minimum, 174*f*
 - Localized averages, 13, 78, 209
 - with large error bounds, 209
 - of model parameters, 117
 - Log-normal probability density function, 165
 - Long-tailed distributions, 42
 - Long-tailed probability density function, 42*f*
 - Loops, 197–198
 - Lorentzian curves, 9*f*
 - fitting, 250–252
 - Lower hemisphere stereonet, 250*f*
- M**
- Magma chambers, 215*f*
 - Magnetotelluric method, 273
 - Mapping function, 270–271
 - Martinson, D. G., 270–271
 - Masters, G., 270
 - MatLab*, 9, 16–17, 159–160, 174–175
 - command plots, 193–194
 - expected value in, 18
 - exponential probability density function in, 150
 - inverse problem in, 232–233
 - least squares with inequality constraints in, 144–145
 - for loops, 197–198
 - Metropolis-Hastings algorithm for, 183–184
 - nonlinear problems, 168–170
 - nonnegative least squares implemented in, 143
 - probability computation in, 30
 - resampling in, 186
 - singular-value decomposition in, 134, 192
 - variance in, 18
 - Matrices
 - columns of, 128
 - covariance, 21, 105
 - equation, bordering method for, 81–82
 - factor, 199
 - Hermetian, 279
 - linear operators and, 214–218
 - loading, 191
 - model resolution, 72, 258
 - in resolution computation, 86
 - selected rows of, 73*f*
 - multiplication, 233
 - norms, 43
 - notation, 45
 - null, 133
 - partial derivatives, 174
 - products, 46, 47–48
 - resolution, 79*f*
 - data, 69–71, 71*f*
 - deltalike, 79
 - relationships to, 117–118
 - spikiness of, 234
 - roughness, 59
 - rows of, 128
 - sample, 191, 194, 199
 - steepness, 56–57
 - Toeplitz, 235
 - transformation, 123
 - unary, 195–196
 - unit covariance, 72–74
 - unit data covariance, 72–73

- Maximum likelihood methods, 17, 183
 - for exact theories, 97–100
 - for inexact theories, 100–102, 103*f*
 - inverse problem, 92–108
 - a priori* distributions, 92–97
 - simplest case, 92
 - of mean of exponential probability density function, 151–153
 - mean of measurement groups, 89–91
 - of probability density functions, 17*f*
 - Maximum Relative Entropy, 108–109
 - M -dimensional space, 124*f*
 - Mean, 17
 - of exponential probability density function, 151–153
 - of measurement groups, 89–91
 - sample, 18
 - of unit vector sets, 246–250
 - zero, 27*f*
 - Meaningful, 1
 - Measures of length, 39–43
 - Median, 151–152
 - Menke, J., 138
 - Menke, W., 138
 - Metropolis-Hastings algorithm, 35–36, 182–183
 - in *MatLab*, 183–184
 - Minimum length, 243
 - damped, 78
 - trade-off curves in, 85*f*
 - generalized inverse, 75
 - solution, 60
 - for temperature distribution problem, 244*f*
 - Minimum Relative Entropy, 108–109
 - Mixed-determined problems, 52, 54–56
 - vector spaces and, 130–131
 - Mixtures, 189, 199–200
 - Model, 1
 - one-dimensional, 221*f*
 - regionalized, 262
 - three-dimensional, 262
 - two-dimensional, 262
 - vector spaces, 123
 - Model parameters, 1
 - a priori* information, 93*f*, 94*f*, 95*f*, 99*f*, 100*f*, 134–135
 - bootstrap confidence intervals, 187*f*
 - estimates of, 11, 21–22, 65*f*, 134, 249
 - covariance of, 134–135
 - variance of, 63–64
 - localized averages of, 117
 - Newton's method, 187*f*
 - parameterization and, 150–151
 - prediction error as function of, 168*f*
 - probability density functions, 93*f*, 94*f*
 - representation of, 124*f*
 - resolution of, 73*f*
 - sets of realizations of, 13
 - sorting, 54–55
 - true, 21–22
 - underdetermined problems, 121*f*
 - unpartitioned, 55–56
 - vector, 86
 - weighted averages, 13
 - bounds on, 120*f*
 - Model resolution
 - matrix, 72, 258
 - in resolution computation, 86
 - selected rows of, 73*f*
 - natural generalized inverse, 134
 - Model spaces, 123
 - Model vector, *a priori*, 57
 - Modes, 256–257
 - Moment tensors of earthquakes, 264–265
 - Monte Carlo search, 170–171
 - for nonlinear curve-fitting problem, 171*f*
 - undirected, 181
 - Mossbauer spectroscopy, 9*f*, 252*f*
 - Mountain profiles, 200*f*, 201*f*
 - Mutually perpendicular factors, 196*f*
- ## N
- Natural generalized inverse, 133
 - model resolution of, 134
 - Negative off-diagonal elements, 78
 - Neighborhoods, 13
 - Newtonian heat flow equation, 223
 - Newton's inverse square law of gravity, 143*f*
 - Newton's method
 - model parameters using, 187*f*
 - for nonlinear inverse problem, 171–175
 - curve-fitting, 176*f*
 - for overdetermined problems, 180
 - Nonlinear explicit equations, 10
 - Nonlinear inverse problems
 - complicated surfaces and, 178*f*
 - curve-fitting
 - gradient method for, 181*f*
 - grid search for, 169*f*
 - Monte Carlo search for, 171*f*
 - simulated annealing for, 184*f*
 - error in, 166–167
 - implicit, with Gaussian data, 175–180

- likelihood in, 166–167
 - MatLab* for, 168–170
 - Newton's method, 171–175
 - curve-fitting, 176*f*
 - Nonlinear theory, 172–173
 - Nonnegative least squares, 141–143, 144–145
 - in *MatLab*, 143
 - Nonspiky orthogonal vectors, 196–197
 - Nonuniform variance, 112
 - Nonunique averaging vectors
 - a priori* information, 119–121
 - bounding, 137
 - Nonunique inverse problem, 53
 - Nonuniqueness, null vectors and, 115–116
 - Normal distribution with zero mean, 27*f*
 - Normal probability density function, 28*f*
 - Normalization, 194–199
 - norminv () function, 32
 - Norms, 40
 - L_∞ , 158–160
 - curve fitting using, 160*f*
 - straight line fitting, 245–246
 - L_1 , 149–150
 - curve fitting using, 156*f*
 - solving, 153–157
 - straight line fitting, 245–246
 - L_2 , 40–42, 53–54, 92, 111, 126, 137–138
 - straight line fitting, 245–246
 - matrix, 43
 - vector, 42
 - Null distribution, 102
 - for inexact non-Gaussian nonlinear theories, 184–185
 - Null eigenvectors, 137
 - Null Hypothesis, 112
 - Null matrix, 133
 - Null probability density functions, 94–96
 - Null solutions, 115–116
 - Null space, 130–131
 - identifying, 133
 - Null vectors
 - fluctuating elements, 232
 - nonuniqueness and, 115–116
 - of simple inverse problem, 116
 - Numbers, complex, 278–280
- O**
- Oldenburg, D. W., 273
 - One-dimensional models, 221*f*
 - One-dimensional temperature distribution, 241–244
 - One-dimensional tomography, 221
- Operators
 - adjoint, 219
 - identity, 220
 - linear, 220*f*
 - matrices and, 214–218
 - solutions and, 69
 - Optical sensors, 231
 - Organ pipe example, 258*f*
 - Orthogonality, 195–196
 - Outliers, 40–42, 149, 151–152, 240–241
 - Overdetermined algorithm, 157
 - Overdetermined problems, 126
 - bricks, 51–52
 - completely, 154–155
 - in generalized inverse with good resolution
 - and covariance, 76–77
 - of inverse problems, 43
 - least square, 52
 - linear programming problems, 158
 - Newton's method for, 180
 - Overshooting, 252
- P**
- P waves, 253*f*, 255
 - Parabolas, 18*f*
 - fitting, 5
 - least squares solution, 47–48
 - Paraboloids, 173*f*
 - Parallel lines, 214*f*
 - Parameterization, 163–165
 - model parameters under, 150–151
 - Parameters, 226–227
 - Parker, Robert, 227
 - Partial derivatives, matrix of, 174
 - Patching, 180
 - Pavlis, G. L., 262–263
 - P*-axes, 250*f*
 - Pearson's chi-squared test, 30
 - example of, 20*f*
 - Perturbation theory, 256–257
 - Perturbed equation, 226–227, 256–257
 - Perturbed modes, 256–257
 - Petrologic database, 193
 - Phases, 266
 - Physicality constraints, 194–199
 - Pixels, 240, 241
 - Plane surface, 48–49
 - Points
 - balancing, 17
 - lines through, 50*f*
 - Precision parameter, 246–248, 248*f*
 - Predicted wave fields, 266–267

- Prediction errors, 41*f*, 131, 137–138, 139–140, 157
 - as function of model parameters, 168*f*
 - L2*, 167*f*
 - of least squares solution, 64–66
 - for lines through points, 50*f*
 - minimizing, 55–56
 - problem of, 146*f*
 - Preliminary Reference Earth Model, 268
 - Probability, 30
 - Probability density functions, 12, 15
 - a priori*, 94, 96
 - joint, 98*f*
 - value of, 177*f*
 - Boltzmann, 182–183
 - chi-squared, 29
 - combining, 101
 - conditional, 30–33
 - example of, 32*f*
 - for inexact theories, 101*f*
 - displaying, 19*f*, 20*f*, 21*f*
 - evaluation of, 99*f*
 - along surface, 178*f*
 - exponential, 149–150, 246–248
 - comparison of, 150*f*
 - in *MatLab*, 150
 - maximum likelihood estimate of mean of, 151–153
 - two-sided, 149
 - Fisher, 246–248
 - Gaussian, 26–29, 36*f*, 97–99
 - joint, 32*f*
 - multivariate, 34
 - log-normal probability density function, 165
 - long-tailed, 42*f*
 - maximum likelihood point of, 17*f*
 - model parameters and, 93*f*, 94*f*
 - normal, 28*f*
 - null, 94–96
 - with ridge, 93*f*
 - shaded area, 16*f*
 - short-tailed, 42*f*
 - transformed, 22*f*, 23*f*, 24*f*
 - uniform, 22*f*, 23*f*, 24*f*
 - with well-defined peak, 93*f*
 - Probability theory
 - conditional probability density functions, 30–33
 - confidence intervals, 33–34
 - correlated data, 19–21
 - functions of random variables, 21–26
 - Gaussian probability density functions, 26–29
 - noise and, 15–18
 - random variables and, 15–18
 - Proportionality factor, 225
 - Proposed successors, 35–36
 - Pure path approximation, 269
 - Purely underdetermined problems, 52–54
- ## Q
- Q-mode factor analysis, 199
 - Quantitative model, 1
- ## R
- Radon transform
 - performing, 213*f*
 - tomography and, 212–213
 - random () function, 34, 150–151
 - Random resampling, 185–186
 - in *MatLab*, 186
 - Random variables
 - circular, 279
 - computing realizations of, 34–36
 - functions of, 21–26
 - Gaussian, 29
 - histograms of, 37*f*
 - realizations of, 90*f*
 - Rayleigh wave, 269
 - Rays, 212
 - deficient distribution of, 242*f*
 - refracted paths of, 255*f*
 - straight line, 253
 - theory, 261
 - tracing, 253
 - Realizations, 1
 - Reciprocal velocity, 261
 - Reconstructed gravity anomalies, 238*f*
 - Refracted paths, 255*f*
 - Regionalized models, 262
 - Relationships, 66
 - to resolution matrices, 117–118
 - Relative entropy, 94–96
 - Resolution
 - of acoustic tomography problem, 87*f*
 - checkerboard test, 87
 - computing, 86–87
 - model resolution matrix in, 86
 - covariance and, 75
 - generalized inverse with, 76–78
 - overdetermined case, 76–77
 - underdetermined case, 77
 - of generalized inverses, 74–75
 - matrix, 79*f*

- data, 69–71, 71*f*
 - deltalike, 79
 - relationships to, 117–118
 - spikiness of, 234
 - model, 134
 - trade-off curves, 85*f*
 - variance and, 79–82, 209
 - Resolving kernel, 208
 - Riemann summation, 211, 226
 - R-mode factor analysis, 199
 - Robust methods, 42, 149
 - Romanowicz, B., 270
 - Rotations, 123, 125
 - Roughness, 56–57
 - matrix, 59
 - Rows, 128
 - Rutile, 195
- S**
- S wave, 255
 - Sample covariance, 21
 - Sample matrix, 191, 194, 199
 - Sample mean, 18
 - Sample standard deviation, 18
 - Samples, 189
 - composition of, 190*f*
 - Satellite gravity data, 238*f*
 - Scalar quantities, 246–248
 - Second-order terms, 65
 - Sediment, 10*f*, 190*f*
 - Seismic attenuation, 269–270
 - Seismic surface waves, 267–269
 - Seismic wave fields, 266
 - Seismology, 227, 258
 - Self-adjoint, 217
 - Sets
 - of realizations of model parameters, 13
 - of unit vectors, 246–250
 - Shear waves, 253*f*
 - Short-tailed probability density function, 42*f*
 - Shuffling, 118
 - Shure, L., 270–271
 - Sidelobes, 78–79
 - Signal correlation, 270–271
 - Simple Gaussian case, 102–104
 - Simple inverse problem, 116
 - Simple least squares, 179
 - Simplicity, 53–54
 - Simulated annealing, 181–184
 - for nonlinear curve-fitting problem, 184*f*
 - parameters in, 182
 - Singular-value decomposition, 55, 139–140, 157, 191, 263
 - of Atlantic Rock dataset, 198*f*
 - derivation of, 138
 - eigenvectors, 194–195
 - generalized inverse, 132–138
 - in *MatLab*, 134, 192
 - Sinusoidal patterns, 243
 - Sinusoids, 60*f*
 - Size, covariance, 83–84
 - Slabs, 241, 242, 243*f*
 - Smoothness, 59
 - Sparse matrices, 8, 59
 - Spectral curve fitting, 9–10
 - Spectrum, 135–136
 - Spikiness, 195
 - of resolution matrix, 234
 - Spirit* rover, 9*f*
 - Spread, 75
 - sqrt () function, 130
 - Standard deviation, sample, 18
 - Station correction, 262
 - Steepness, 56–57
 - matrix, 56–57
 - Straight lines
 - fitting, 4–5, 41*f*
 - constrained, 62–63
 - L_∞ norm, 245–246
 - $L1$ norm, 245–246
 - $L2$ norm, 245–246, 248*f*
 - least squares, 40*f*, 43–44, 62*f*, 65*f*, 74*f*
 - problem, 46–47
 - rays, 253
 - Stretching factor, 22
 - Strike and dip, 160
 - Surface wave tomography, 269
 - svd (), 192–193
 - Sylvester equation, 77–78
- T**
- Taylor series, 65, 173–174
 - Taylor's theorem, 171–172
 - Tectonic plate motions, 271
 - Teleseismic tomography, 264
 - Temperature distribution
 - Backus-Gilbert for, 244*f*
 - minimum length for, 244*f*
 - model for, 244*f*
 - one-dimensional, 241–244
 - Temperature function, 225
 - Thermal motion, 182–183
 - Three-dimensional models, 262

- Time reversed operators, 225
 - Time sequence, 203*f*
 - Time series, 234
 - Toeplitz matrix, 235
 - Tomography
 - acoustic, 6–7
 - with deficient distribution of rays, 242*f*
 - success of, 241
 - true image, 241*f*
 - computed, 7*f*
 - continuous inverse theory and, 211–212
 - hypothetical experiments, 84*f*
 - inversions, 264
 - one-dimensional, 221
 - radon transform, 212–213
 - teleseismic, 264
 - waveform, 265–267
 - Tracks
 - ascending, 237*f*
 - descending, 237*f*
 - Trade-off curves, 85, 209
 - asymptotes in, 86*f*
 - Backus-Gilbert solutions, 85*f*
 - damped minimum length solution, 85*f*
 - resolution, 85*f*
 - variance, 85*f*
 - Transformations
 - forward, 129
 - householder, 124–127
 - designing, 124–127
 - inverse, 129
 - length preservation and, 124–127
 - linearizing, 165–166
 - matrix, 123
 - Transformed probability density function, 22*f*, 23*f*
 - Travel time data, 261–264
 - Triangle inequalities, 43
 - True model parameters, 21–22
 - Two-dimensional models, 262
 - Two-sided exponential probability density functions, 149
- U**
- Unary matrix, 195–196
 - Unbiased estimation, 165
 - Uncorrelated Gaussian noise, 234–235
 - Underdetermined algorithm, 157
 - Underdetermined problems, 51–52
 - Backus-Gilbert generalized inverse and, 79–82
 - completely, 154
 - in generalized inverse with good resolution and covariance, 77
 - model parameters, 121*f*
 - purely, 52–54
 - Undirected Monte Carlo method, 181
 - Unfolding, 202
 - unidrnd (), 186
 - Uniform probability density function, 22*f*, 23*f*, 24*f*
 - Uniform variance, 234–235
 - Unit covariance matrix, 72–74, 99
 - Unit data covariance matrix, 72–73
 - Unit sphere, 248*f*
 - Unit vectors
 - Cartesian, 249
 - mean of sets of, 246–250
 - scattering, 247*f*
 - Unitary transformations, 125, 127
 - Univariate distribution, 24*f*, 32
 - Unknown forcing, 227
 - Unpartitioned model parameters, 55–56
 - Unperturbed equation, 227
 - Unperturbed problem, 256–257
 - Unperturbed structures, 256–257
- V**
- Variables
 - auxiliary, 186
 - random
 - circular, 279
 - computing realizations of, 34–36
 - functions of, 21–26
 - Gaussian, 29
 - histograms of, 37*f*
 - realizations of, 90*f*
 - Variance, 17–18. *See also* Covariance
 - a posteriori*, 63
 - a priori*, 63
 - model parameters, 107–108
 - a priori* model variance, 107–108
 - eigenvalues and, 136–137
 - expression for, 66
 - high, 195
 - in MatLab*, 18
 - of model parameter estimates, 63–64
 - nonuniform, 112
 - prediction error of least squares solution, 64–66
 - resolution and, 79–82, 209
 - trade-off curves, 85*f*
 - uniform, 234–235
 - Varimax procedure, 195–196, 198

Vasco, D. W., 264

Vectors

arbitrary, 235

lambda, 134

N -dimensional position, 215

nonspiky orthogonal, 196–197

nonunique averaging

a priori information, 119–121

 bounding, 137

norms, 42

rotation, 195–196

spaces

 applications of, 123–148

 data spaces, 123

 mixed-determined problem solution,
 130–131

 model, 123

Velocity structure

 earthquake location and, 261–264

 from free oscillations, 267–269

 from seismic surface waves, 267–269

Vertically stratified structures, 262

Vibrational problems, 256–258

Volcano, 215*f*

Voxels, 84*f*

W

Waveform cross-correlation, 263

Waveform tomography, 265–267

Weight, 64*f*

Weighted averages, 115

 of model parameters, 13

 bounds on, 120*f*

Weighted damped least squares, 58–60

 true model, 60*f*

Weighted least squares, 58

Weighted measures, 56–60

Weighted minimum length, 58

weightedleastsquaresfcn (), 59, 180

Wind forcing, 222–223

X

xcorr (), 235

X-ray imaging, 7–9

X-ray opacity, 55*f*, 84

X-ray tomography, 54

Z

Zero mean, 27*f*