Chapter 12

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

$$d_i = \frac{1}{3}[m_{i-1} + m_i + m_{i+1}] \tag{12.1}$$

Note that this is a linear equation and can be written in the form $\mathbf{Gm} = \mathbf{d}$, where

$$\mathbf{G} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ & & & \ddots & & & \\ 0 & \cdots & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{12.2}$$

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

$$\begin{aligned} \mathbf{m}^{(1)\text{null}} &= [1 \quad 0 \quad -1 \quad 1 \quad 0 \quad -1 \quad \cdots \quad 1 \quad 0 \quad -1]^{\mathrm{T}} \\ \mathbf{m}^{(2)\text{null}} &= [0 \quad 1 \quad -1 \quad 0 \quad 1 \quad -1 \quad \cdots \quad 0 \quad 1 \quad -1]^{\mathrm{T}} \end{aligned} \tag{12.3}$$

Geophysical Data Analysis: Discrete Inverse Theory. DOI: 10.1016/B978-0-12-397160-9.00012-6

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

$$\mathbf{m}^{\text{est}} = \mathbf{G}^{\text{T}}\left[\mathbf{G}\mathbf{G}^{\text{T}}\right]^{-1}\mathbf{d}^{\text{obs}} \tag{12.4}$$

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:

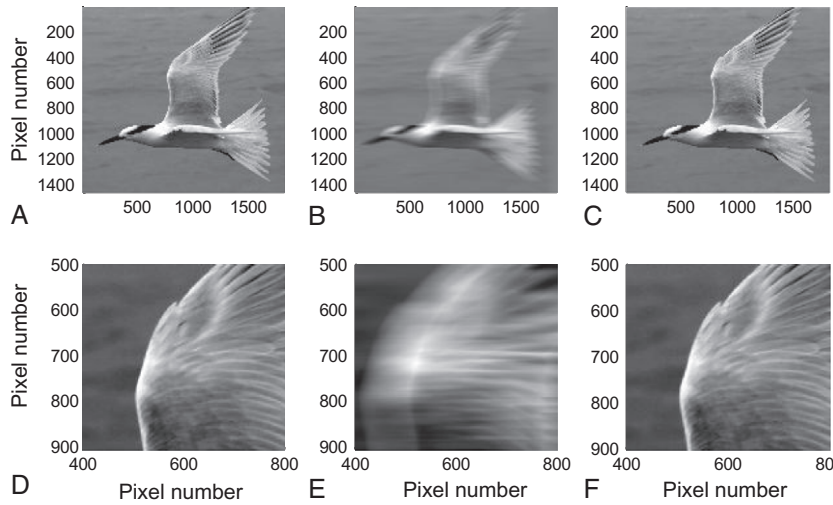

FIGURE 12.1 Example of removing blur from image. (A) True image. (B) Image blurred with 100 pixel-wide boxcar filter. (C) Estimated image, unblurred using the minimum length generalized inverse. (D–F) Enlargement of portion of images (A–C). *MatLab* script gda12_01.

```
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 $[\mathbf{GG}^{\mathbf{T}}]$ is calculated by simple matrix multiplication.

We note that the matrix $\mathbf{G}$ is simple enough for the matrix product $[\mathbf{GG}^{\mathbf{T}}]$ to be computed analytically

$$\mathbf{GG}^{\mathrm{T}} = \frac{1}{9}\begin{bmatrix} 3 & 2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 2 & 3 & 2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 2 & 3 & 2 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix} \tag{12.5}$$

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

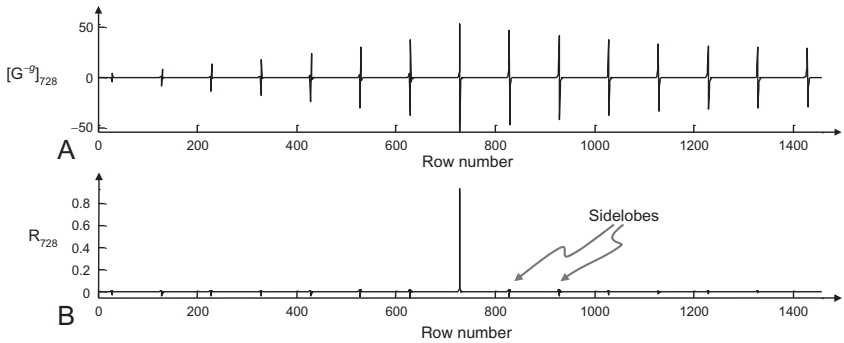

FIGURE 12.2 (A) Central row (row 728) of the generalized inverse of the image deblurring problem. (B) Central row (row 728) of the corresponding resolution matrix. The sidelobes are about 6% of the amplitude of the central peak. *MatLab* script gda12_01.

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 $m(t)$:

$$d(t) = m(t) * g(t) = \int g(t-\tau)m(\tau)\ \mathrm{d}\tau \tag{12.6}$$

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

$$d_i = \Delta t \sum_{j=1}^{M} g_{i-j+1}\, m_j \tag{12.7}$$

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

$$\mathbf{G} = \Delta t \begin{bmatrix} g_1 & 0 & 0 & \cdots & 0 \\ g_2 & g_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_N & g_{N-1} & g_{N-2} & \cdots & g_{N-M+1} \end{bmatrix} \tag{12.8}$$

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, **g** must be known exactly, while **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 $[\mathbf{G}^{\mathrm{T}}\mathbf{G}]\mathbf{m}^{\mathrm{est}}=\mathbf{G}^{\mathrm{T}}\mathbf{d}$. This formulation is especially attractive because the matrices $[\mathbf{G}^{\mathrm{T}}\mathbf{G}]$ and $\mathbf{G}^{\mathrm{T}}\mathbf{d}$ can be computed analytically as

$$\mathbf{G}^{\mathrm{T}}\mathbf{G}=(\Delta t)^2\begin{bmatrix}\sum_{i=1}^{N}g_i^2 & \sum_{i=2}^{N-1}g_ig_{i-1} & \cdots\\ \sum_{i=2}^{N-1}g_ig_{i-1} & \sum_{i=1}^{N-1}g_i^2 & \cdots\\ \vdots & \vdots & \ddots\end{bmatrix}\quad\text{and}\quad\mathbf{G}^{\mathrm{T}}\mathbf{d}=\Delta t\begin{bmatrix}\sum_{i=1}^{N}d_ig_i\\ \sum_{i=2}^{N-1}d_ig_{i-1}\\ \vdots\end{bmatrix}\tag{12.9}$$

Furthermore, the summations in $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ 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 $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ matrix contains the *autocorrelation* of **g** (denoted $\mathbf{g}\star\mathbf{g}$) and $\mathbf{G}^{\mathrm{T}}\mathbf{d}$ the *cross-correlation* of **g** with **d** (denoted $\mathbf{g}\star\mathbf{d}$)

$$[\mathbf{G}^{\mathrm{T}}\mathbf{G}]_{ij}=(\Delta t)^2[\mathbf{g}\star\mathbf{g}]_{|i-j|+1}\text{ and }[\mathbf{G}^{\mathrm{T}}\mathbf{d}]_i=\Delta t[\mathbf{g}\star\mathbf{d}]_i,$$
$$\text{where }[\mathbf{a}\star\mathbf{b}]_i=\sum_{j=1}^{N}a_jb_{i+j-1}\tag{12.10}$$

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 $\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{m}=\mathbf{F}^{\mathrm{T}}\mathbf{d}$, where the matrix **F** contains both **G** and a smoothness matrix **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 $\mathbf{F}^{\mathrm{T}}(\mathbf{F}\mathbf{v})=\mathbf{G}^{\mathrm{T}}(\mathbf{G}\mathbf{v})+\varepsilon^2\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{v})$, where **v** is an arbitrary vector. The $\mathbf{G}^{\mathrm{T}}(\mathbf{G}\mathbf{v})$ product can be computed extremely efficiently starting with **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.

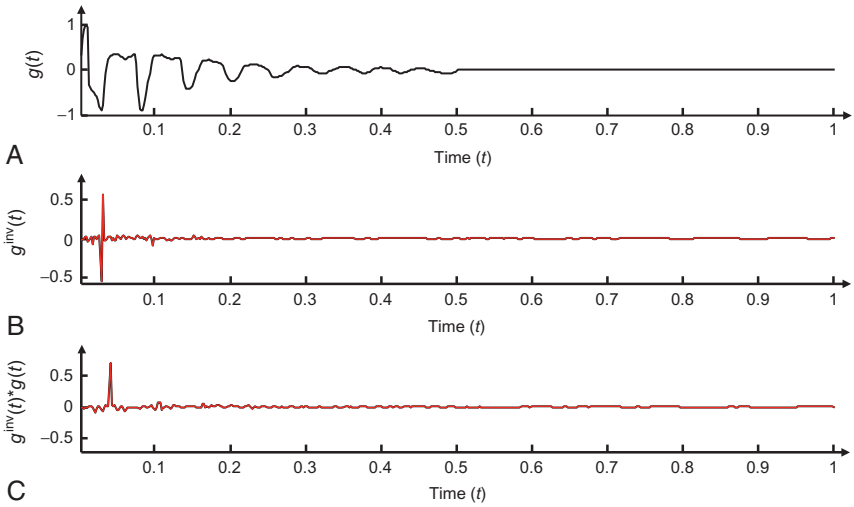

FIGURE 12.3 (A) An airgun signal $g(t)$, after Smith (1975). Ideally, the inverse filter $g^{\text{inv}}(t)$ when convolved with $g(t)$ should produce the spike $\delta(t-t_0)$, centered at time t_0. (B) Estimate of the inverse filter $g^{\text{inv}}(t)$ for $t_0=0.04$, computed via generalized least 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 $\mathbf{d}=[0,0,0,\ldots,0,1,0,\ldots,0]^{\text{T}}$ 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).

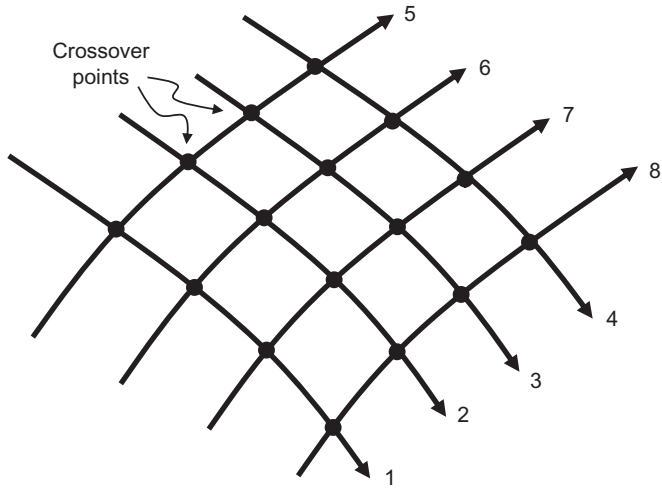

FIGURE 12.4 Descending tracks 1–4 intersect ascending tracks 5–8 at 16 points. The height of the satellite along each track is determined by minimizing the crossover error at the intersections.

Suppose that there are M orbits and that the unknown altitude of the satellite during the ith 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 ith 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 $\mathbf{Gm} = \mathbf{d}$, where

$$G_{ij} = \delta_{jA_i} - \delta_{jD_i} \quad \text{and} \quad d_i = s_{A_i} - s_{D_i} \tag{12.11}$$

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 ms 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

$$\mathbf{m}^{\text{est}} = [\mathbf{G}^{\text{T}}\mathbf{G} + \varepsilon^2\mathbf{I}]^{-1}\mathbf{G}^{\text{T}}\mathbf{d} \tag{12.12}$$

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 $[\mathbf{G}^{\text{T}}\mathbf{G}]$ and $[\mathbf{G}^{\text{T}}\mathbf{d}]$ 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

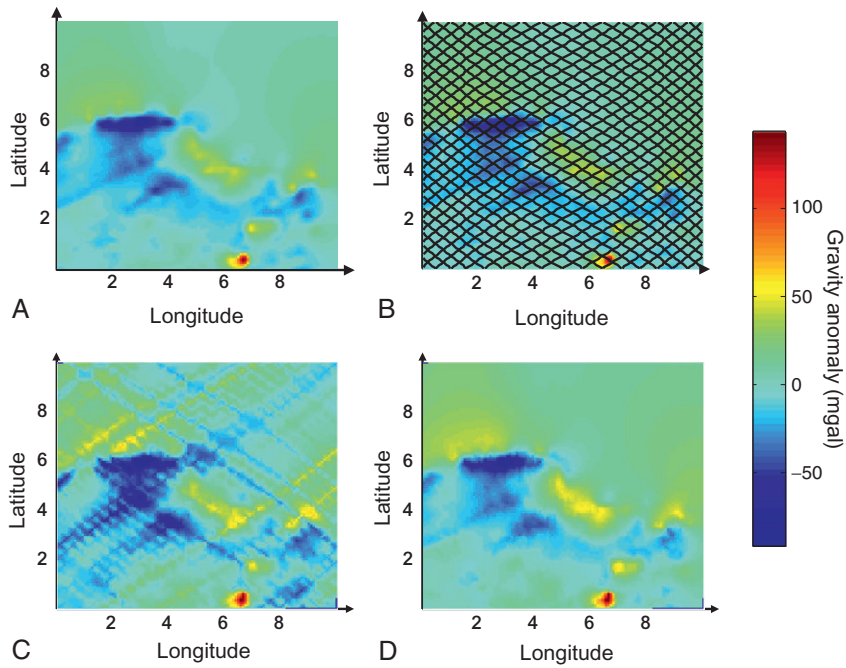

FIGURE 12.5 Example of crossover error adjustment of satellite gravity data. (A) True gravity anomaly data for the equatorial Atlantic Ocean. It reflects variations in the depth of the seafloor and density variations within the oceanic crust. (B) Hypothetical satellite tracks, along which the gravity 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.

$$[\mathbf{G}^{\mathrm{T}}\mathbf{G}]_{rs} = \sum_{i=1}^{N} G_{ir}G_{is} = \sum_{i=1}^{N} (\delta_{rA_i} - \delta_{rD_i})(\delta_{sA_i} - \delta_{sD_i})$$
$$= \sum_{i=1}^{N} (\delta_{rA_i}\delta_{sA_i} - \delta_{rA_i}\delta_{sD_i} - \delta_{rD_i}\delta_{sA_i} + \delta_{rD_i}\delta_{sD_i}) \tag{12.13}$$

The diagonal elements of $[\mathbf{G}^{\mathrm{T}}\mathbf{G}]$ are

$$[\mathbf{G}^{\mathrm{T}}\mathbf{G}]_{rr} = \sum_{i=1}^{N} (\delta_{rA_i}\delta_{rA_i} - 2\delta_{rA_i}\delta_{rD_i} + \delta_{rD_i}\delta_{rD_i}) \tag{12.14}$$

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 rth element of the diagonal is the number of times the rth orbit is intersected by other orbits.

Only the two middle terms of the sum in the expression for $[\mathbf{G}^{\mathrm{T}}\mathbf{G}]_{rs}$ 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 rth and sth orbits intersect, multiplied by -1.

The other matrix product is

$$[\mathbf{G}^{\mathrm{T}}\mathbf{d}]_r = \sum_{i=1}^{N} G_{ir}d_i = \sum_{i=1}^{N} (\delta_{rA_i} - \delta_{rD_i})d_i \tag{12.15}$$

We note that the delta functions can never both equal 1 since an orbit can never intersect itself. Therefore $[\mathbf{G}^{\mathrm{T}}\mathbf{d}]_r$ is the sum of all the ds that have ascending orbit number $A_i = r$ minus the sum of all the ds 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 $[\mathbf{G}^{\mathrm{T}}\mathbf{G}]$ and $[\mathbf{G}^{\mathrm{T}}\mathbf{d}]$ initialized to zero and, for each ith row of the table, execute the following steps:

(1) Add 1 to the $r = A_i$, $s = A_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{G}]_{rs}$.
(2) Add 1 to the $r = D_i$, $s = D_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{G}]_{rs}$.
(3) Subtract 1 from the $r = A_i$, $s = D_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{G}]_{rs}$.
(4) Subtract 1 from the $r = D_i$, $s = A_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{G}]_{rs}$.
(5) Add d_i to the $r = A_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{d}]_r$.
(6) Subtract d_i from the $r = D_i$ element of $[\mathrm{G}^{\mathrm{T}}\mathrm{d}]_r$.

The final form for $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ 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).

$$\begin{bmatrix} [\mathbf{G}^{\mathrm{T}}\mathbf{G}] & \mathbf{1} \\ \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ \lambda \end{bmatrix} = \begin{bmatrix} [\mathbf{G}^{\mathrm{T}}\mathbf{d}] \\ 0 \end{bmatrix} \tag{12.16}$$

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

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_8 \end{bmatrix} = h \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ d_{16} \end{bmatrix} \tag{12.17}$$

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 $\mathbf{d} = \mathbf{G}\mathbf{m}^{\mathrm{true}}$. 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,

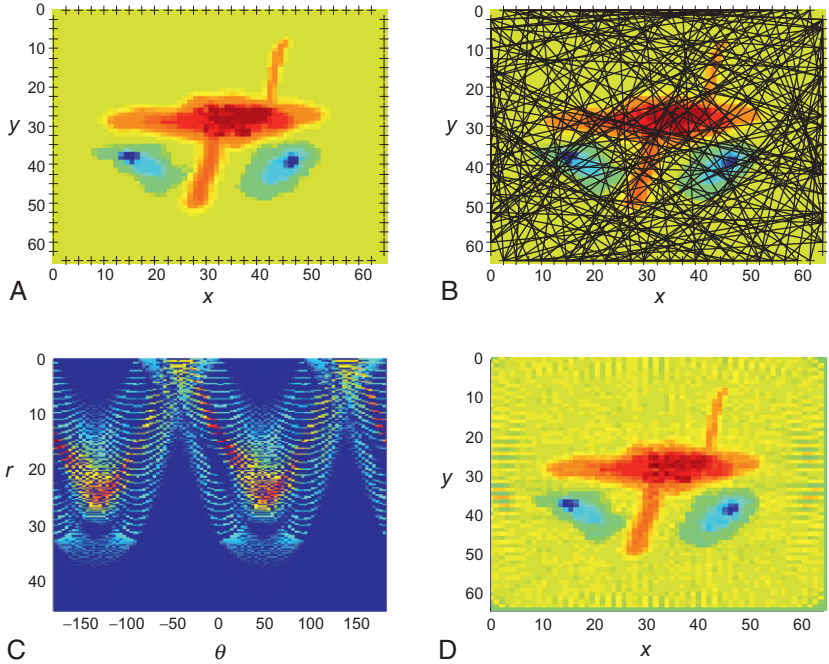

FIGURE 12.6 Acoustic tomography problem. (A) True image. (B) Ray paths (only a few percent of rays are shown, else the image would be black). (C) Travel time data, organized by the distance r and angle θ of the ray from the midpoint of the image. (D) Reconstructed image. See text for further discussion. *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.

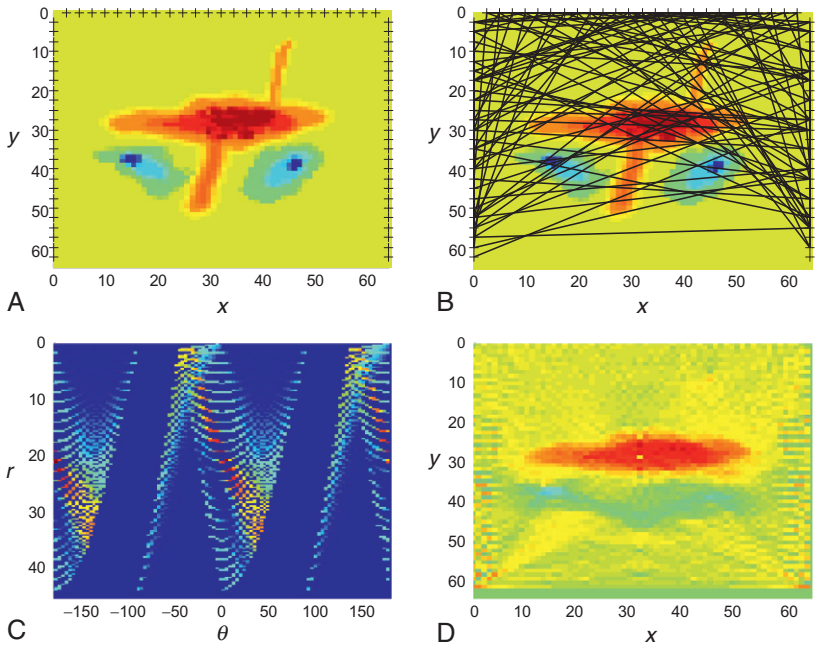

FIGURE 12.7 Acoustic tomography problem with deficient distribution of rays. (A) True image. (B) Ray paths (only a few percent of rays are shown, else the image would be black). (C) Travel time data, organized by the distance r and angle θ of the ray from the midpoint of the image. (D) Reconstructed 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)

$$T(x,t) = \frac{1}{2} T_0 \left\{ \text{erf}\left[\frac{x - (\xi - {}^1\!/_2 h)}{\sqrt{t}}\right] - \text{erf}\left[\frac{x - (\xi + {}^1\!/_2 h)}{\sqrt{t}}\right] \right\} = T_0\, g(x,t,\xi) \tag{12.18}$$

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 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

$$T(x_i,t) = \sum_{j=1}^{M} g(x_i, t, \xi_j) m_j \tag{12.19}$$

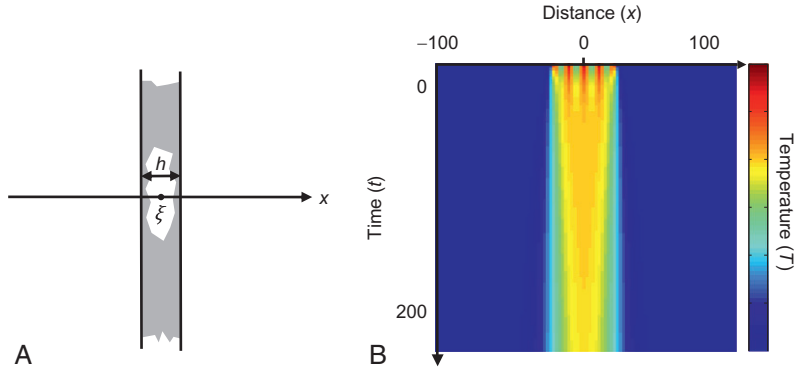

FIGURE 12.8 (A) Single hot slab of thickness, h, located at position, $x=\xi$. (B) Temporal evolution of the temperature $T(x, t)$ of 100 adjacent slabs. The initial temperature distribution of the slabs, $T(x, t=0)$, is taken to be the model parameter vector, $\mathbf{m}$. It is nonzero only for slabs 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.

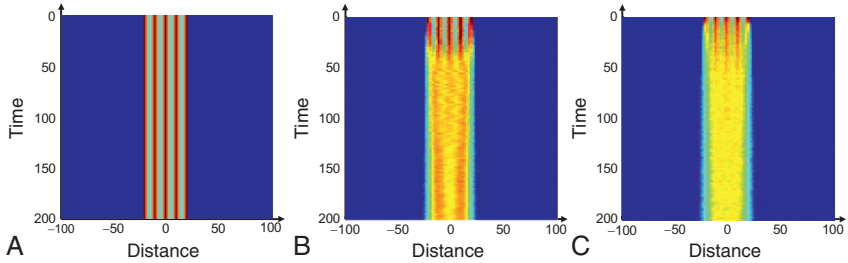

FIGURE 12.9 Model for the temperature distribution problem. (A) The true model is the initial temperature distribution, $T(x, t=0)$. While this function is not a function of time, it is displayed on the (x, t) image for comparison purposes. (B) Minimum length (ML) estimate of the model, $T(x, t=0)$, 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.

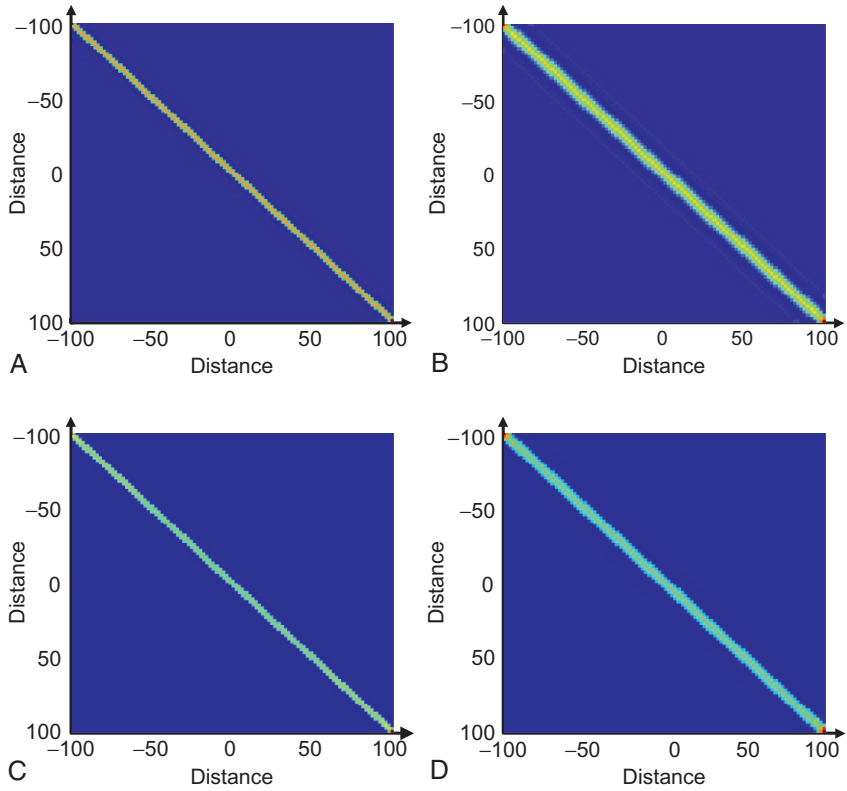

FIGURE 12.10 Model resolution matrices for the temperature distribution problem. (A) Minimum length (ML) solution for data at time $t=10$. (B) ML solution for data at time $t=40$. (C) Backus-Gilbert (BG) solution for data at time $t=10$. (D) BG solution for data at time $t=40$. Note that the BG resolution 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

$$\mathbf{G} = \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} \tag{12.20}$$

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

$$\mathbf{N} = \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_N \end{bmatrix} \frac{1}{N\sum z_i^2 - \left(\sum z_i\right)^2} \begin{bmatrix} \sum_{k=1}^{N} z_k^2 & -\sum_{k=1}^{N} z_k \\ -\sum_{k=1}^{N} z_k & N \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ z_1 & z_2 & z_3 & \cdots & z_N \end{bmatrix}$$

$$N_{ij} = \frac{\sum z_k^2 - (z_i + z_j)\sum z_k + z_i z_j N}{N\sum z_i^2 - \left(\sum z_i\right)^2} = A_i + B_i z_j \tag{12.21}$$

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 $\mathbf{c}^{\text{T}}\mathbf{y}$ 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

$$\begin{aligned} \mathbf{y}^{\text{T}} &= [[m_1', \ldots, m_M'], [m_1'', \ldots, m_M''], [\alpha_1, \ldots, \alpha_N], [x_1, \ldots, x_N], [x_1', \ldots, x_N']] \\ \mathbf{c}^{\text{T}} &= [[0, \ldots, 0], [0, \ldots, 0], [1, \ldots, 1], [0, \ldots, 0], [0, \ldots, 0]] \\ \mathbf{A} &= \begin{bmatrix} \mathbf{G}_{\text{N}\times\text{M}} & -\mathbf{G}_{\text{N}\times\text{M}} & -\mathbf{I}_{\text{N}\times\text{N}} & \mathbf{I}_{\text{N}\times\text{N}} & \mathbf{O}_{\text{N}\times\text{N}} \\ \mathbf{G}_{\text{N}\times\text{M}} & -\mathbf{G}_{\text{N}\times\text{M}} & \mathbf{I}_{\text{N}\times\text{N}} & \mathbf{O}_{\text{N}\times\text{N}} & -\mathbf{I}_{\text{N}\times\text{N}} \end{bmatrix} \\ \mathbf{b}^{\text{T}} &= [[d_1, \ldots, d_N], [d_1, \ldots, d_N]] \end{aligned} \tag{12.22}$$

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

$$\begin{aligned}
\mathbf{y}^{\mathrm{T}} &= [[m_1', \ldots, m_M'], [m_1'', \ldots, m_M''], [\alpha], [x_1, \ldots, x_N], [x_1', \ldots, x_N']] \\
\mathbf{c}^{\mathrm{T}} &= [[0, \ldots, 0], [0, \ldots, 0], [1], [0, \ldots, 0], [0, \ldots, 0]] \\
\mathbf{A} &= \begin{bmatrix} \mathbf{G}_{\mathrm{N\times M}} & -\mathbf{G}_{\mathrm{N\times M}} & -\mathbf{1} & \mathbf{I}_{\mathrm{N\times N}} & \mathbf{O}_{\mathrm{N\times N}} \\ \mathbf{G}_{\mathrm{N\times M}} & -\mathbf{G}_{\mathrm{N\times M}} & \mathbf{1} & \mathbf{O}_{\mathrm{N\times N}} & -\mathbf{I}_{\mathrm{N\times N}} \end{bmatrix} \\
\mathbf{b}^{\mathrm{T}} &= [[d_1, \ldots, d_N], [d_1, \ldots, d_N]]
\end{aligned} \tag{12.23}$$

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

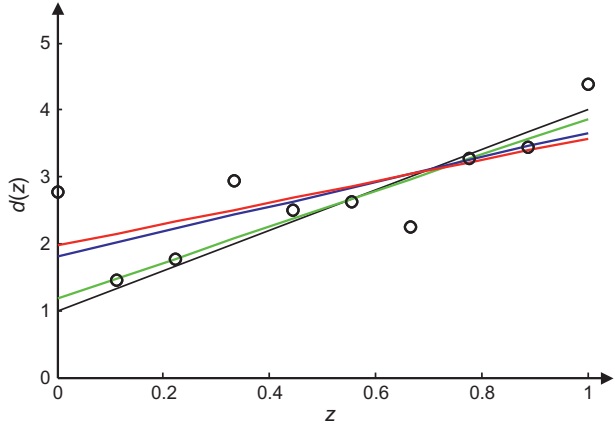

FIGURE 12.11 Fitting a straight line to data $d(z)$. The true model (black line) is the straight line, $d^{\mathrm{true}}(z) = 1 + 3z$. The $N = 10$ observations d_i^{obs} are the true data perturbed with exponentially distributed noise with variance ${\sigma_d}^2 = (0.4)^2$. Three different fits have 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.

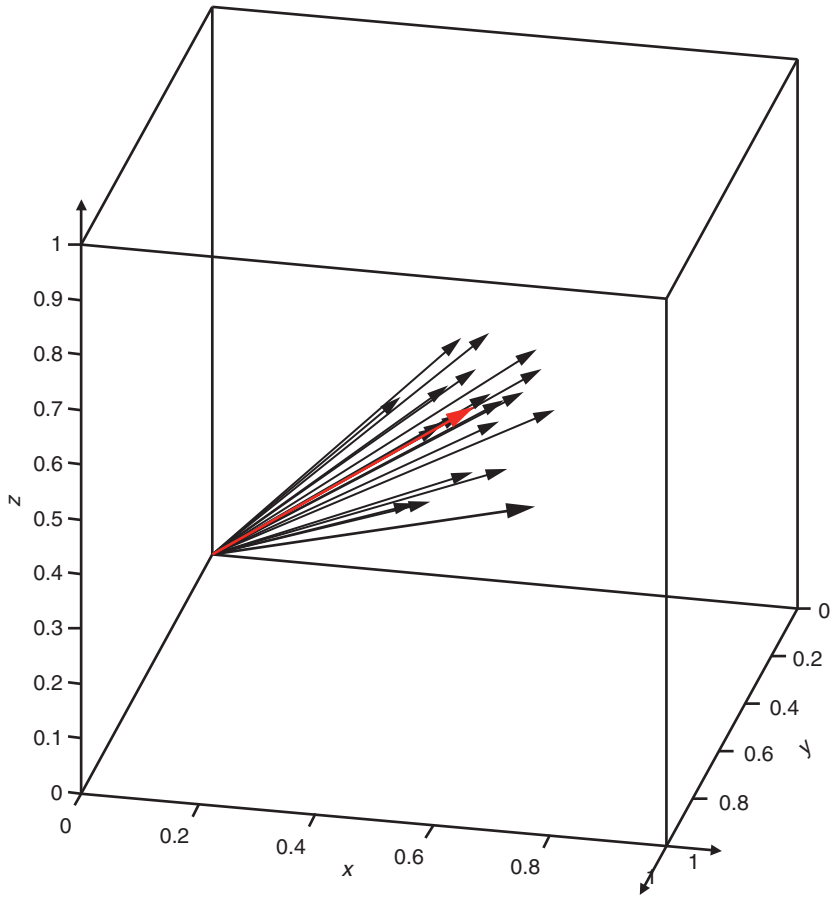

FIGURE 12.12 Several unit vectors (black) scattering about a central direction (red). *MatLab* script gda12_08.

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

$$p(\theta, \phi) = \frac{\kappa}{4\pi \sinh(\kappa)} \exp[\kappa \cos(\theta)] \tag{12.24}$$

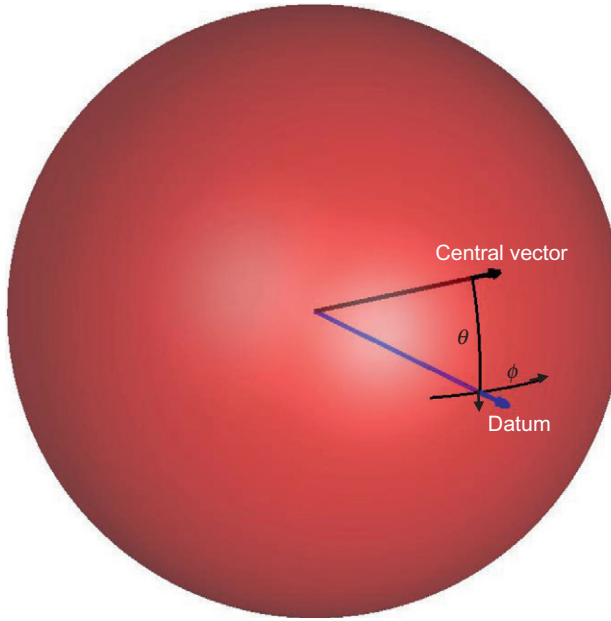

FIGURE 12.13 Unit sphere, showing coordinate system used in Fisher distribution. *MatLab* script gda12_09.

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.

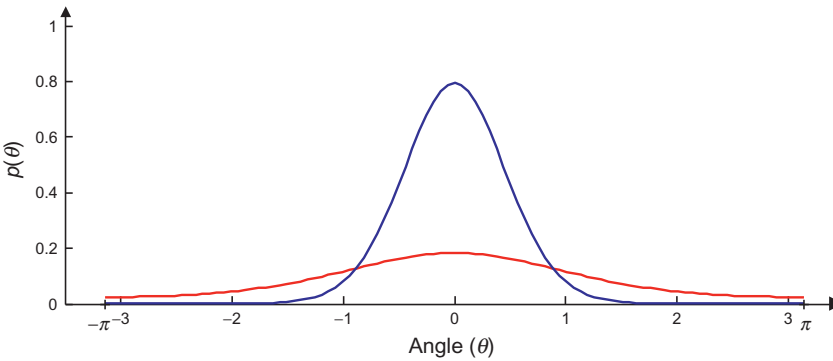

FIGURE 12.14 Fisher distribution, $p(\theta, \phi)$, for a large precision parameter ($\kappa = 5$, blue curve) and a small precision parameter ($\kappa = 1$, red curve). *MatLab* script gda12_10.

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

$$p(\theta, \phi) = \left[\frac{\kappa}{4\pi \sinh(\kappa)}\right]^N \exp\left[\kappa \sum_{i=1}^{N} \cos(\theta_i)\right] \prod_{i=1}^{N} \sin(\theta_i) \tag{12.25}$$

where $\cos(\theta_i) = \mathbf{x}^{\mathrm{T}}\mathbf{m} = [x_i m_1 + y_i m_2 + z_i m_3]$. The likelihood function is

$$\begin{aligned} L = \log(p) &= N \log(\kappa) - \mathrm{N}\, \log(4\pi) - N \log[\sinh(\kappa)] \\ &+ \kappa \sum_{i=1}^{N} [x_i m_1 + y_i m_2 + z_i m_3] + \sum_{i=1}^{N} \log[\sin(\theta_i)] \end{aligned} \tag{12.26}$$

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

$$\begin{aligned} \kappa \sum_i x_i - 2\lambda m_1 &= 0 \\ \kappa \sum_i y_i - 2\lambda m_2 &= 0 \\ \kappa \sum_i z_i - 2\lambda m_3 &= 0 \\ \frac{N}{\kappa} - N\frac{\cosh(\kappa)}{\sinh(\kappa)} + \sum_{i=1}^{N} [x_i m_1 + y_i m_2 + z_i m_3] &= 0 \end{aligned} \tag{12.27}$$

where λ is the Lagrange multiplier. The first three equations can be solved simultaneously along with the constraint equation for the model parameters as

$$[m_1, m_2, m_3]^{\mathrm{T}} = \frac{\left[\sum_i x_i, \sum_i y_i, \sum_i z_i\right]^{\mathrm{T}}}{\left\{\left(\sum_i x_i\right)^2 + \left(\sum_i y_i\right)^2 + \left(\sum_i z_i\right)^2\right\}^{1/2}} \tag{12.28}$$

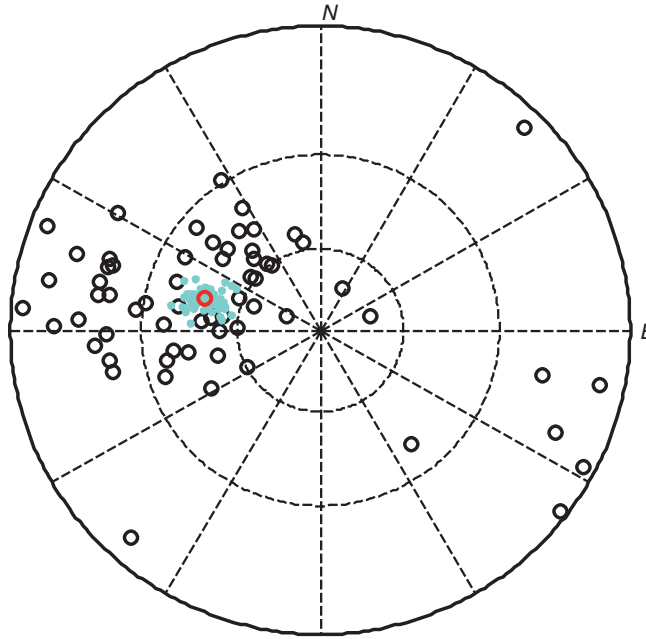

FIGURE 12.15 Lower hemisphere stereonet showing *P*-axes of deep (300–600 km) earthquakes in the Kurile-Kamchatka subduction zone. The axes mostly dip to the west, parallel to the direction of subduction, indicating that the subducting slab is in down-dip compression. (Black circles) Axes of individual 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

$$\kappa \approx \frac{N}{N - \sum_i \cos(\theta_i)} \tag{12.29}$$

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 $\mathbf{m} = [A_1, f_1, c_1, \ldots, A_q, f_q, c_q]^{\mathrm{T}}$. The model is nonlinear and of the form $\mathbf{d} = \mathbf{g}(\mathbf{m})$

$$\begin{aligned} \text{Gaussian:} \quad d_i &= \sum_{j=1}^{q} \frac{A_j}{(2\pi)^{1/2} c_j} \exp\left[-\frac{(z_i - f_j)^2}{2c_j^2}\right] \\ \text{Lorentzian:} \quad d_i &= \sum_{j=1}^{q} \frac{A_j c_j^2}{(z_i - f_j)^2 + c_j^2} \end{aligned} \tag{12.30}$$

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

$$[\partial g_i/\partial A_1 \quad \partial g_i/\partial f_1 \quad \partial g_i/\partial c_1 \quad \cdots \quad \partial g_i/\partial A_p \quad \partial g_i/\partial f_p \quad \partial g_i/\partial c_p] \tag{12.31}$$

In this problem the Gaussian and Lorentzian models are simple enough for the derivatives to be computed analytically as

Gaussian:

$$\begin{aligned} \partial g_i/\partial A_j &= [1/(2\pi)^{1/2} c_j] \exp[-(z_i - f_j)^2/2c_j^2] \\ \partial g_i/\partial f_j &= \left[\frac{A_j}{(2\pi)^{1/2} c_j}\right] [(z_i - f_j)/c_j^2] \exp[-(z_i - f_j)^2/2c_j^2] \\ \partial g_i/\partial c_j &= \left[\frac{A_j}{(2\pi)^{1/2} c_j^2}\right] [((z_i - f_j)^2/c_j^2) - 1] \exp[-(z_i - f_j)^2/2c_j^2] \end{aligned} \tag{12.32}$$

Lorentzian:

$$\begin{aligned} \partial g_i/\partial A_j &= c_j^2/[(z_i - f_j)^2 + c_j^2] \\ \partial g_i/\partial f_j &= 2A_j c_j^2 (z_i - f_j)/[(z_i - f_j)^2 + c_j^2]^2 \\ \partial g_i/\partial c_j &= 2A_j c_j/[(z_i - f_j)^2 + c_j^2] - 2A_j c_j^3/[(z_i - f_j)^2 + c_j^2]^2 \end{aligned} \tag{12.33}$$

These derivatives are evaluated at the trial solution $\mathbf{m}^{(\mathrm{p})}$ 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 $\mathbf{G}\Delta\mathbf{m} = \mathbf{d} - \mathbf{g}(\mathbf{m}^{(\mathrm{p})})$ and $\mathbf{m}^{(\mathrm{p}+1)} = \mathbf{m}^{(\mathrm{p})} + \Delta\mathbf{m}$, 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.

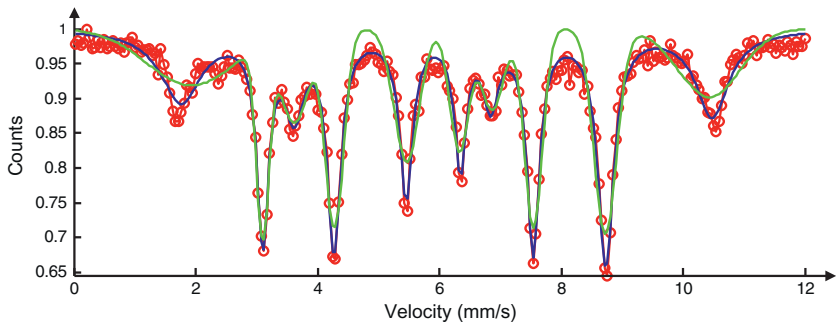

FIGURE 12.16 Example of fitting the sum of 10 Lorentzian (blue) or Normal (green) curves to Mossbauer spectroscopic data (red). The Lorentzian curves are better able to fit the shape of the curve, with the ratio of estimated variances being about 4. An F-test indicates that a null hypothesis that any 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 $\mathbf{Hm} = \mathbf{h}$

$$\begin{bmatrix} \mathbf{G}^{(p)} \\ \varepsilon \mathbf{H} \end{bmatrix} \mathbf{m}^{(p+1)} = \begin{bmatrix} \left\{ \mathbf{d} - \mathbf{g}(\mathbf{m}^{(p)}) + \mathbf{G}^{(p)} \mathbf{m}^{(p)} \right\} \\ \varepsilon \mathbf{h} \end{bmatrix} \tag{12.34}$$

$$\mathbf{H} = [\cdots \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0 \quad -1 \quad 0 \quad \cdots], \quad \text{and} \quad \mathbf{h} = \left[\langle s_{ij} \rangle \right]$$

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 *hypocenter* (location, $\mathbf{x}^{(0)} = [x_0, y_0, z_0]^{\mathrm{T}}$) 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 $i = 1, \ldots, N$ *receivers* (seismometers) located at $\mathbf{x}^{(\mathrm{i})} = [x_i, y_i, z_i]^{\mathrm{T}}$. 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

$$t_i^P = T_i^P(\mathbf{x}^{(0)}, \mathbf{x}^{(\mathrm{i})}) + t_0 \qquad \text{and} \qquad t_i^S = T_i^S(\mathbf{x}^{(0)}, \mathbf{x}^{(\mathrm{i})}) + t_0 \tag{12.35}$$

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 $\nabla T = [\partial T_i/\partial x_0, \partial T_i/\partial y_0, \partial T_i/\partial y_0]^{\mathrm{T}}$ 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

$$\begin{aligned}
\frac{\partial T_i^P}{\partial x_0} &\approx \frac{T_i^P(x_0 + \Delta x, y_0, z_0, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta x} \\
\frac{\partial T_i^P}{\partial y_0} &\approx \frac{T_i^P(x_0, y_0 + \Delta y, z_0, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta y} \\
\frac{\partial T_i^P}{\partial z_0} &\approx \frac{T_i^P(x_0, y_0, z_0 + \Delta z, x_i, y_i, z_i) - T_i^P(x_0, y_0, z_0, x_i, y_i, z_i)}{\Delta z}
\end{aligned} \tag{12.36}$$

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

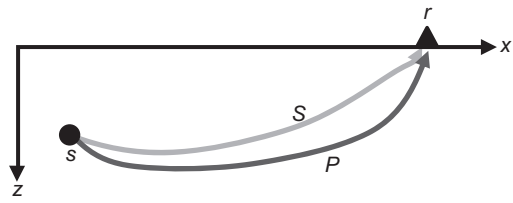

FIGURE 12.17 Compressional P and shear S waves travel along rays from earthquake source s (circle) to receiver r (triangle).

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 ith receiver (of a total of N receivers), and similarly $\mathbf{s}^{(i)S}$ is for the S wave, the linearized problem is

$$\begin{bmatrix} -s_1^{(1)P}/v_0^P & -s_2^{(1)P}/v_0^P & -s_3^{(1)P}/v_0^P & 1 \\ \cdots & \cdots & \cdots & \cdots \\ -s_1^{(N)P}/v_0^P & -s_2^{(N)P}/v_0^P & -s_3^{(N)P}/v_0^P & 1 \\ -s_1^{(1)S}/v_0^S & -s_2^{(1)S}/v_0^S & -s_3^{(1)S}/v_0^S & 1 \\ \cdots & \cdots & \cdots & \cdots \\ -s_1^{(N)S}/v_0^S & -s_2^{(N)S}/v_0^S & -s_3^{(N)S}/v_0^S & 1 \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \\ \Delta z_0 \\ t_0 \end{bmatrix} = \begin{bmatrix} t_1^P - T_1^P \\ \cdots \\ t_N^P - T_N^P \\ t_1^S - T_1^S \\ \cdots \\ t_N^S - T_N^S \end{bmatrix} \tag{12.37}$$

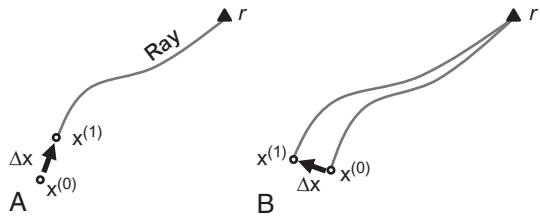

FIGURE 12.18 (A) Moving the source a distance $\Delta\mathbf{x}$ from $\mathbf{x}^{(0)}$ to $\mathbf{x}^{(1)}$ parallel to the ray path leads to a large change in travel time. (B) Moving the source perpendicular to the ray path leads to no (first order) change in travel time. The partial derivative of 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 $[1, 1, 1, \ldots, 1]^T$ 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 $[1, 1, 1, \ldots, 1, v_S/v_P, \ldots]^T$ 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.

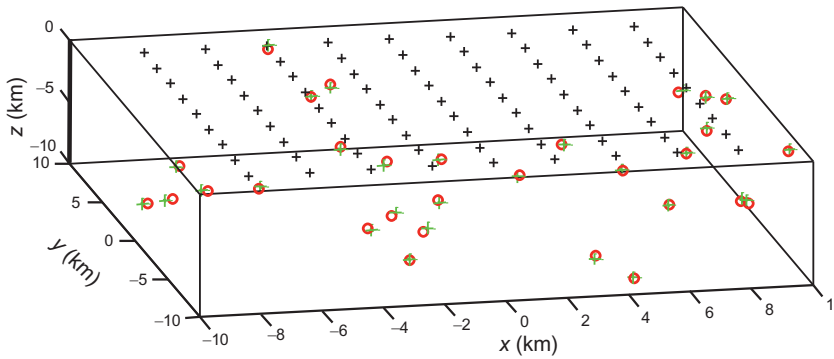

FIGURE 12.19 Earthquake location example. Arrival times of P and S waves from earthquakes (red circles) are recorded on an array of 81 stations (black crosses). The observed arrival times include random noise with variance $\sigma_d^2 = (0.1)^2$s. The estimated locations (green crosses) are computed using Geiger's method. *MatLab* script gda12_13.

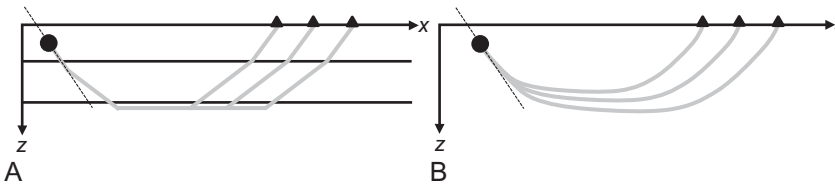

FIGURE 12.20 (A) In layered media, rays follow "refracted" paths, such that the rays to all receivers (triangles) leave the source (circle) at the same angle. The position and travel time of the source on the dashed line therefore trade off. (B) This phenomenon also occurs in nonlayered media if the 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

$$-\omega_n^2 p_n(\mathbf{x}) = v^2(\mathbf{x})\nabla^2 p_n(\mathbf{x}) \tag{12.38}$$

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

$$\int p_n(\mathbf{x}) p_m(\mathbf{x}) v^{-2}(\mathbf{x}) \mathrm{d}^3 x = \delta_{nm} \tag{12.39}$$

(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

$$v(\mathbf{x}) = v^{(0)}(\mathbf{x}) + \varepsilon v^{(1)}(\mathbf{x}) \tag{12.40}$$

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

$$\begin{aligned} \omega_n &= \omega_n^{(0)} + \varepsilon\omega_n^{(1)} + \varepsilon^2\omega_n^{(2)} + \cdots \qquad \text{and} \\ p_n(\mathbf{x}) &= p_n^{(0)}(\mathbf{x}) + \varepsilon p_n^{(1)}(\mathbf{x}) + \varepsilon^2 p_n^{(2)}(\mathbf{x}) + \cdots \end{aligned} \tag{12.41}$$

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

$$p_m^{(1)} = \sum_{\substack{m \\ \omega_m \neq \omega_n}}^{\infty} b_{nm} p_m^{(0)} \tag{12.42}$$

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)

$$\begin{aligned} \omega_n^{(1)} &= \omega_n^{(0)} \int [p_n^{(0)}(\mathbf{x})]^2 [v^{(0)}(\mathbf{x})]^{-3} v^{(1)}(\mathbf{x}) \mathrm{d}^3x \\ b_{nm} &= \frac{2(\omega_m^{(0)})^2}{(\omega_m^{(0)})^2 - (\omega_n^{(0)})^2} \int p_n^{(0)}(\mathbf{x}) p_m^{(0)}(\mathbf{x}) [v^{(0)}(\mathbf{x})]^{-3} v^{(1)}(\mathbf{x}) \mathrm{d}^3x \end{aligned} \tag{12.43}$$

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 $v^{(1)}(\mathbf{x})$,

$$\omega_n^{(1)} = \int G_n(\mathbf{x}) v^{(1)}(\mathbf{x}) \, \mathrm{d}^3x \quad \text{with} \quad G_n(\mathbf{x}) = \omega_n^{(0)} [p_n^{(0)}(\mathbf{x})]^2 [v^{(0)}(\mathbf{x})]^{-3} \tag{12.44}$$

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 $p|_{x=0} = \mathrm{d}p/\mathrm{d}x|_{x=h} = 0$). We consider the unperturbed problem of a constant velocity structure, which has modes and eigenfrequencies given by

$$\begin{aligned} p_n^{(0)}(x) &= \frac{2[v^{(0)}]^2}{h} \sin\left\{ \frac{(n - {}^1\!/_2)\pi}{h} x \right\} \quad \text{and} \\ \omega_n^{(0)} &= \frac{\pi(n - {}^1\!/_2) v^{(0)}}{h} \quad \text{with} \quad n = 1, 2, 3, \ldots \end{aligned} \tag{12.45}$$

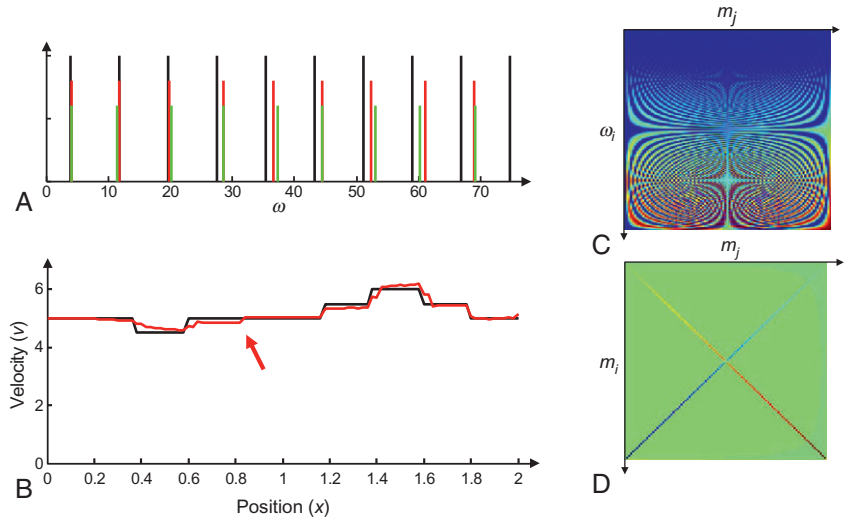

FIGURE 12.21 Organ pipe example. (A) Ladder diagram for unperturbed (black), true (red), and observed (green) eigenfrequencies of the organ pipe. (B) True (black) and estimated (red) velocity structure. Arrow points to an artifact in the estimated solution. (C) Data kernel **G**. (D) Model resolution 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, $d_n = \omega_n^{(1)} = \omega_n - \omega_n^{(0)}$. The model parameters are the continuous function $m(x) = v^{(1)}(x) = v(x) - v^{(0)}$, which we will discretize into a vector **m** by sampling it at M evenly spaced values, with spacing Δx. The data kernel in the discrete equation $\mathbf{d} = \mathbf{Gm}$ is the discrete version of Equation (12.44) (Figure 12.21C)

$$G_{nm} = G_n(x_m)\Delta x = \omega_n^{(0)}[p_n^{(0)}(x_m)]^2[v^{(0)}]^{-3}\Delta x \tag{12.46}$$

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 **bicg()** with the weighted least squares function `weightedleastsquaresfcn().`]

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.