Chapter 8

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):

$$\underset{\text{Gaussian}}{p(d) = \frac{1}{(2\pi)^{1/2}\,\sigma}\,\exp\left\{-\frac{1}{2}\frac{(d-\langle d\rangle)^2}{\sigma^2}\right\}} \qquad \underset{\text{exponential}}{p(d) = \frac{1}{(2)^{1/2}\,\sigma}\,\exp\left\{-(2)^{1/2}\frac{|d-\langle d\rangle|}{\sigma}\right\}} \tag{8.1}$$

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).

Geophysical Data Analysis: Discrete Inverse Theory. DOI: 10.1016/B978-0-12-397160-9.00008-4

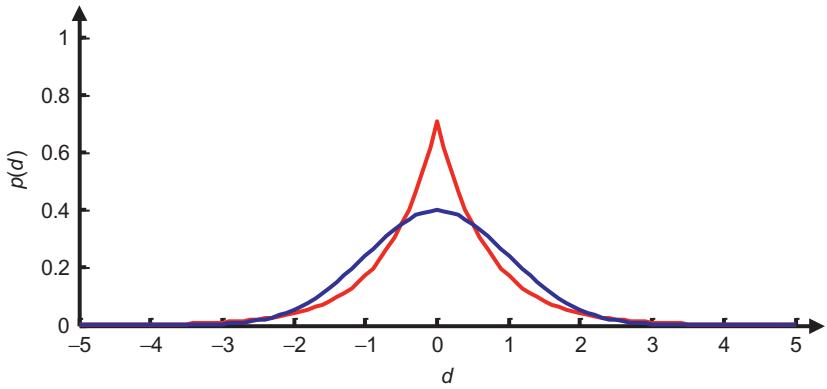

FIGURE 8.1 Comparison of exponential probability density distribution (red) with a Gaussian probability density distribution (blue) of equal variance, $\sigma^2 = 1$. The exponential probability density distribution is longer tailed. *MatLab* script gda08_01.

TABLE 8.1 The area beneath $\langle d \rangle \pm n\sigma$ for the Gaussian and exponential distributions

	Gaussian	Exponential
n	*Area (%)*	*Area (%)*
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 `dbar` and variance `sd^2` is calculated as

```
pE = (1/sqrt(2))*(1/sd)*exp(-sqrt(2)*abs((d-dbar))/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:

$$p(d) = (2)^{-N/2}\,\sigma^{-N}\exp\left[-\frac{(2)^{1/2}}{\sigma}\sum_{i=1}^{N}|d_i - m_1|\right] \tag{8.2}$$

To maximize the likelihood of $p(d)$, we must maximize the argument of the exponential, which involves minimizing the sum of absolute residuals as

$$\text{minimize} \quad E = \sum_{i=1}^{N}|d_i - m_1| \tag{8.3}$$

This is the L_1 norm of the prediction error in a linear inverse problem of the form $\mathbf{Gm} = \mathbf{d}$, where $M = 1$, $\mathbf{G} = [1, 1, \ldots, 1]^{\mathrm{T}}$, and $\mathbf{m} = [m_1]$. Applying the principle of maximum likelihood, we obtain

$$\text{maximize} \quad L = \log P = -\frac{N}{2}\log(2) - N\log(\sigma) - \frac{(2)^{1/2}}{\sigma}\sum_{i=1}^{N}|d_i - m_1| \tag{8.4}$$

Setting the derivatives to zero yields

$$\begin{aligned}\frac{\partial L}{\partial m_1} &= 0 = \frac{(2)^{1/2}}{\sigma}\sum_{i=1}^{N}\text{sign}(d_i - m_1)\\ \frac{\partial L}{\partial \sigma} &= 0 = \frac{N}{\sigma} - \frac{(2)^{1/2}}{\sigma^2}\sum_{i=1}^{N}|d_i - m_1|\end{aligned} \tag{8.5}$$

The sign function sign(x) equals $+1$ if $x > 0$, -1 if $x < 0$ and 0 if $x = 0$. Note that the derivative $\mathrm{d}|x|/\mathrm{d}x = \text{sign}(x)$, since if x is positive, then it is just equal to $\mathrm{d}x/\mathrm{d}x = 1$ and if it is negative, then to -1. The first equation yields the implicit expression for $\mathrm{m}_1 = \langle d\rangle^{\text{est}}$ for which $\Sigma_i\ \text{sign}(d_i - m_1) = 0$. The second equation can then be solved for an estimate of the variance as

$$\sigma^{\text{est}} = \frac{(2)^{1/2}}{N}\sum_{i=1}^{N}|d_i - m_1| \tag{8.6}$$

The equation for m_1^{est} is exactly the sample median; one finds an m_1 such that half the ds 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*.

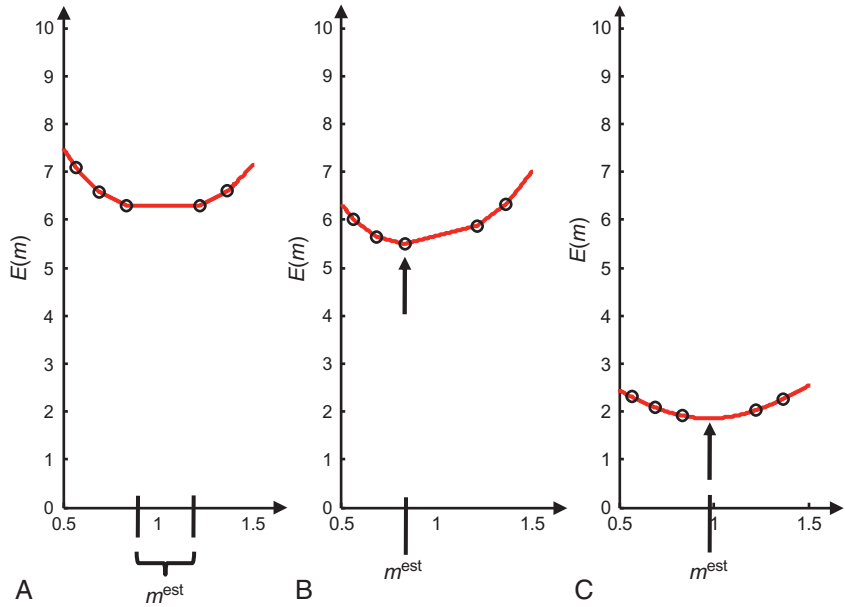

FIGURE 8.2 Inverse problem to estimate the mean m^{est} of N observations. (A) L_1 error $E(m)$ (red curve) with even N. The error has a flat minimum, bounded by two observations (circles), and the solution is nonunique. (B) L_1 error with odd N. The error is minimum at one of the observation points, 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 $\mathbf{Gm}=\mathbf{d}$ 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 $\mathbf{Gm}=\mathbf{d}$, 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 `Nr`.

8.3 THE GENERAL LINEAR PROBLEM

Consider the linear inverse problem $\mathbf{Gm}=\mathbf{d}$ in which the data and *a priori* model parameters are uncorrelated with known means $\mathbf{d}^{\text{obs}}$ and $\langle \mathbf{m} \rangle$ and known variances σ_d^2 and σ_m^2, respectively. Their joint distribution is then

$$p(\mathbf{d},\mathbf{m}) = 2^{-(M+N)/2} \prod_{i=1}^{N} \sigma_{d_i}^{-1} \exp\left[-2^{1/2} \frac{|e_i|}{\sigma_{d_i}}\right] \prod_{j=1}^{M} \sigma_{mj}^{-1} \exp\left[-2^{1/2} \frac{|l_j|}{\sigma_{mj}}\right] \quad (8.7)$$

where the prediction error is given by $\mathbf{e}=\mathbf{d}-\mathbf{Gm}$ 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:

$$\text{minimize} \quad E+L = \sum_{i=1}^{N} \frac{|e_i|}{\sigma_{d_i}} + \sum_{j=1}^{M} \frac{|l_j|}{\sigma_{mj}} \quad (8.8)$$

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)

$$\begin{aligned}&\text{find } \mathbf{x} \text{ that minimizes } z = \mathbf{f}^{\mathrm{T}}\mathbf{x}\\ &\text{with the constraints } \mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \text{and} \quad \mathbf{C}\mathbf{x} = \mathbf{d} \quad \text{and} \quad \mathbf{x}^{(\mathrm{l})} \leq \mathbf{x} \leq \mathbf{x}^{(\mathrm{u})}\end{aligned} \tag{8.9}$$

We first define the L_1 versions of the weighted solution length L and the weighted prediction error E

$$L = \sum_{i=1}^{M} \frac{|m_i - \langle m_i \rangle|}{\sigma_{mi}} \quad \text{and} \quad E = \sum_{i=1}^{N} \frac{|d_i^{\mathrm{obs}} - d_i^{\mathrm{pre}}|}{\sigma_{d_i}} \quad \text{with} \quad \mathbf{d}^{\mathrm{pre}} = \mathbf{G}\mathbf{m} \tag{8.10}$$

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, α_i, x_i, and x'_i, where $i = 1, \ldots, M$. The linear programming problem may be stated as follows (Cuer and Bayer, 1980)

$$\text{minimize } z = \sum_{i=1}^{M} \frac{\alpha_i}{\sigma_{m_i}} \quad \text{subject to the constraints}$$

$$\mathbf{G}(\mathbf{m}' - \mathbf{m}'') = \mathbf{d} \quad \text{and} \quad \mathbf{m}' - \mathbf{m}'' + \mathbf{x}_{\mathbf{i}} - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \quad \mathbf{m}' - \mathbf{m}'' - \mathbf{x}' - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle$$

and

$$\mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0 \tag{8.11}$$

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

$$\boldsymbol{\alpha} - \mathbf{x} = [\mathbf{m} - \langle \mathbf{m} \rangle] \quad \text{and} \quad \boldsymbol{\alpha} - \mathbf{x}' = -[\mathbf{m} - \langle \mathbf{m} \rangle] \tag{8.12}$$

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 α_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):

$$\text{minimize} \quad E=\sum_{i=1}^{N}\frac{\alpha_i}{\sigma_{d_i}} \quad \text{subject to the constraints}$$

$$\mathbf{G}[\mathbf{m}'-\mathbf{m}'']+\mathbf{x}-\boldsymbol{\alpha}=\mathbf{d} \quad \text{and} \quad \mathbf{G}[\mathbf{m}'-\mathbf{m}'']-\mathbf{x}'+\boldsymbol{\alpha}=\mathbf{d} \tag{8.13}$$

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.

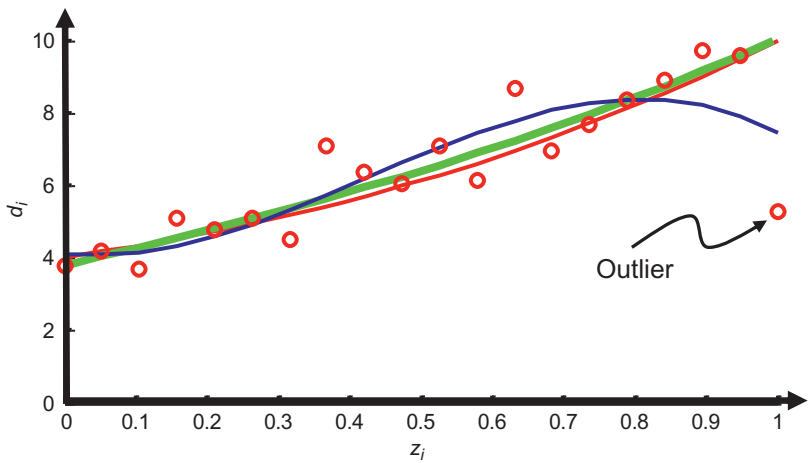

FIGURE 8.3 Curve fitting using the L_1 norm. The true data (red curve) follow a cubic polynomial in an auxiliary variable, z. Observations (red circles) have additive noise with zero mean and variance $\sigma^2 = 1$ drawn from an exponential probability density function. Note the outlier at $z \approx 1$. 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 ($5M+2N$):

$$\begin{aligned}&\text{minimize} \quad \sum_{i=1}^{M} \frac{\alpha_i}{\sigma_{m_i}} + \sum_{i=1}^{N} \frac{\alpha_i'}{\sigma_{d_i}} \quad \text{subject to the constraints} \\ &[\mathbf{m}' - \mathbf{m}''] + \mathbf{x} - \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \quad [\mathbf{m}' - \mathbf{m}''] - \mathbf{x}' + \boldsymbol{\alpha} = \langle \mathbf{m} \rangle \quad \text{and} \\ &\mathbf{G}[\mathbf{m}' - \mathbf{m}''] + \mathbf{x}'' - \boldsymbol{\alpha}' = \mathbf{d} \quad \text{and} \quad \mathbf{G}[\mathbf{m}' - \mathbf{m}''] - \mathbf{x}'' + \boldsymbol{\alpha}' = \mathbf{d} \quad \text{and} \\ &\mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \boldsymbol{\alpha}' \geq 0 \\ &\text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0 \quad \text{and} \quad \mathbf{x}'' \geq 0 \quad \text{and} \quad \mathbf{x}'' \geq 0\end{aligned} \tag{8.14}$$

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

$$\mathbf{m}^{\text{est}} = \sum_{i=1}^{p} a_i \mathbf{v}_p^{(i)} + \sum_{i=p+1}^{M} b_i \mathbf{v}_0^{(i)} = \mathbf{V}_p \mathbf{a} + \mathbf{V}_0 \mathbf{b} \tag{8.15}$$

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

$$\text{find } \mathbf{a} \text{ that minimizes} \quad E = ||\mathbf{d} - \mathbf{G}\mathbf{m}||_1 = \sum_{i=1}^{N} \frac{\left| d_i - [\mathbf{U}_p \boldsymbol{\Lambda}_p \mathbf{a}]_i \right|}{\sigma_{d_i}} \tag{8.16}$$

Next, one uses the underdetermined algorithm to determine $\mathbf{b}$:

$$\text{find } \mathbf{b} \text{ that minimizes} \quad E = ||\mathbf{m} - \langle \mathbf{m} \rangle||_1 = \sum_{i=1}^{N} \frac{|[\mathbf{V}_0 \mathbf{b}]_i - (\langle m_i \rangle - [\mathbf{V}_p \mathbf{a}]_i)|}{\sigma_{m_i}} \tag{8.17}$$

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:

$$\text{minimize} \quad L + E = ||\mathbf{e}||_\infty + ||\mathbf{l}||_\infty = \max_i \frac{|e_i|}{\sigma_{d_i}} + \max_i \frac{|l_i|}{\sigma_{m_i}} \tag{8.18}$$

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

$$\begin{aligned}&\text{minimize} \quad \alpha \quad \text{subject to the constraints}\\ &\mathbf{G}[\mathbf{m}' - \mathbf{m}''] = \mathbf{d} \quad \text{and}\\ &[m'_i - m''_i] + x_i - \alpha\sigma_{m_i} = \langle m_i \rangle \quad \text{and} \quad [\mathrm{m}'_i - \mathrm{m}''_i] - \mathrm{x}'_i + \alpha\sigma_{m_i} = \langle m_i \rangle\\ &\text{and} \quad \mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \alpha \geq 0 \quad \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0\end{aligned} \tag{8.19}$$

where $\mathbf{m} = \mathbf{m}' - \mathbf{m}''$. We note that the new constraints can be written as

$$\alpha - \frac{x_i}{\sigma_{m_i}} = \frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \quad \text{and} \quad \alpha - \frac{x'_i}{\sigma_{m_i}} = -\frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \tag{8.20}$$

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 $\alpha \geq |[m_i - \langle m_i \rangle]/\sigma_{mi}|$ for all i. Since this problem has but a single parameter α, it must therefore satisfy

$$\alpha \geq \max_i \left| \frac{[m_i - \langle m_i \rangle]}{\sigma_{m_i}} \right| \tag{8.21}$$

Minimizing α yields the L_∞ solution.

The linear programming problem for the overdetermined case is

$$\begin{aligned}&\text{minimize} \quad \alpha \quad \text{subject to the constraints}\\ &\sum_{j=1}^{M} G_{ij}\,[m'_j - m''_j] + x_i - \alpha\sigma_{d_i} = d_i \quad \text{and} \quad \sum_{j=1}^{M} G_{ij}\,[m'_j - m''_j] - x'_i + \alpha\sigma_{d_i} = d_i\\ &\text{and} \quad \mathbf{m}' \geq 0 \quad \text{and} \quad \mathbf{m}'' \geq 0 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0 \quad \text{and} \quad \mathbf{x} \geq 0 \quad \text{and} \quad \mathbf{x}' \geq 0\end{aligned} \tag{8.22}$$

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!
```

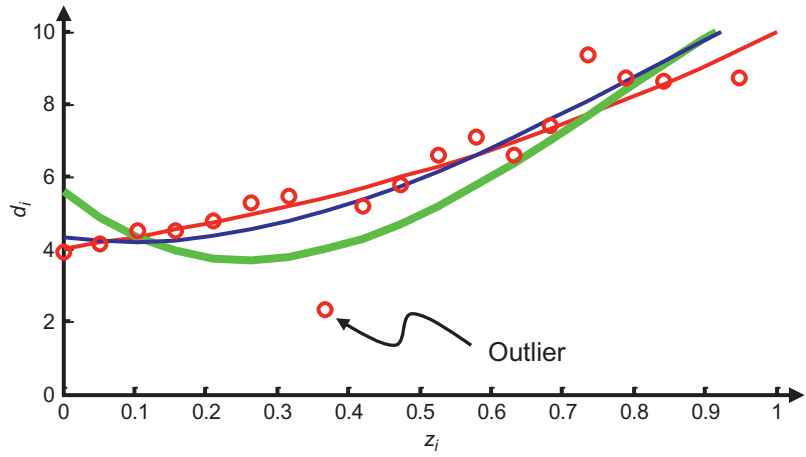

FIGURE 8.4 Curve fitting using the L_∞ norm. The true data (red curve) follow a cubic polynomial in an auxiliary variable z. Observations (red circles) have additive noise with zero mean and variance $\sigma^2 = 1$ drawn from an exponential probability density function. Note the outlier at $z \approx 0.37$. 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 zs 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.