Parameter Estimation and Inverse Problems

Second Edition

Instructor Solutions Manual

Richard C. Aster

Brian Borchers

Clifford H. Thurber

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Academic Press is an imprint of Elsevier

Academic Press is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA
The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, UK

ISBN: 978-0-12-385048-5

For information on all Academic Press publications, visit our website: www.elsevierdirect.com

Preface

This instructor's guide has been prepared to help instructors who are teaching from the second edition of "Parameter Estimation and Inverse Problems." In addition to solutions for all of the exercises in the book, this guide contains summary suggestions to the instructor about how to approach each chapter, suggested homework assignments, and common student questions.

In addition to assigning homework sets, we have found it valuable for each student in the course to give an approximately 20 minute in-class presentation and prepare an associated concise term paper on some subject related to the course. Example presentation topics (see class website below for additional examples) reflect the wide range of interests in our classes, and have included such topics as "Solving L1 Regression Problems by Interior Methods", "Calculation of Electromagnetic Sensitivities by the Adjoint Method", "Inversion of Free Surface Reflection Data", "Validation of a Distributed Parameter Watershed Model to Observations of Soil Moisture Using Inverse Methods", "Borehole Temperature Inversion", and "Limitations of the ForWARD Algorithm in Reconstructing Interferometric Images."

As with the text itself, we would greatly appreciate comments, corrections, or suggestions on this solutions manual.

Rick Aster, Brian Borchers, and Cliff Thurber

aster@ees.nmt.edu
borchers@nmt.edu
thurber@geology.wisc.edu

`http://www.ees.nmt.edu/outside/Geop/Classes/GEOP529_class.html`
`http://www.ees.nmt.edu/outside/Geop/Classes/GEOP529_book.html`

January, 2012

Contents

Chapter 1

Introduction

1.1 Discussion of the Chapter

The main purpose of this chapter is to introduce some of the basic concepts and terminology associated with inverse problems. In the authors' experience, students who are "shopping around" for a course can read the first chapter and attend our lectures to get a good idea of what the course will be about. By discussing the material in this chapter with students we also get the opportunity to learn some details about their mathematical background.

A second purpose of this chapter is to introduce the students to several specific examples that will be used and re-used throughout the course. Thus it would be wise to cover all of these examples.

The technical level of the material in Chapter 1 is fairly low, so the material can be presented relatively quickly. We typically take about three one hour lectures to cover Chapter 1. After Chapter 1, we move on to a review of mathematical topics in appendices A, B, and C.

Section 1.5 is somewhat more mathematically technical than the other sections in this chapter. The point of this section is to introduce the ideas of existence, uniqueness, and stability of solutions to inverse problems. These ideas are central to an understanding of inverse problems, and the material in this section is designed to make it clear to students that inverse problems can be extremely hard if not impossible to solve.

Exercise 1.1 is a theoretical exercise that helps to introduce the concept of linearity and make the connection between linear problems and matrix–vector formulation of a forward problem. This exercise can be surprisingly challenging, even for students who have done well in a typical introductory course in linear algebra. An important point here is that the matrix vector product $\mathbf{Ax}$ can be written as

$$\mathbf{Ax} = x_1\mathbf{A}_{\cdot,1} + \ldots + x_n\mathbf{A}_{\cdot,n}.$$

Exercise 1.2 is a basic thought experiment exercise to better illustrate inconsistency of linear equations. Exercise 1.3 is a practical exercise that illustrates discretization of a continuous problem and ill-posedness. Exercise 1.4 is a good way to connect the material in this course to student's research interests. The course as taught at NMT includes a final term paper, which might be based on the paper that the student identifies in this exercise.

1.2 Exercises

1. Consider a mathematical model of the form $G(\mathbf{m}) = \mathbf{d}$, where $\mathbf{m}$ is a vector of length n, and $\mathbf{d}$ is a vector of length m. Suppose that the model obeys the superposition and scaling laws and is thus linear. Show that $G(\mathbf{m})$ can be written in the form

$$G(\mathbf{m}) = \mathbf{\Gamma m} \tag{1.1}$$

   where $\mathbf{\Gamma}$ is an m by n matrix. What are the elements of $\mathbf{\Gamma}$? Hint: Consider the standard basis, and write $\mathbf{m}$ as a linear combination of the vectors in the standard basis. Apply the superposition and scaling laws. Finally, recall the definition of matrix–vector multiplication.

2. Can (1.14) be inconsistent, even with only $m = 3$ data points? How about just $m = 2$ data points? If the system can be inconsistent, give an example. If not, explain why not.

3. Consider the borehole vertical seismic profile problem of Examples 1.3 and 1.9 for $n = 100$ equally spaced seismic sensors located at depths of $z = 0.2,\ 0.4,\ \ldots,\ 20$ m, and for a model $\mathbf{m}$ describing n corresponding equal length seismic slowness values for 0.2 m intervals having midpoints at $z - 0.1$ m.

   (a) Calculate the appropriate system matrix, $\mathbf{G}$ for discretizing the integral equation (1.21) using the midpoint rule.

   (b) For a linear seismic velocity depth gradient model specified by

$$v = v_0 + kz \tag{1.2}$$

   where the velocity at $z = 0$ is $v_0 = 1$ km/s and the gradient is $k = 40$ m/s per m, calcuate the true slowness values at the midpoints of the n intervals, $\mathbf{m}_{true}$. Additionally, integrate the corresponding slowness function for (1.2) using (1.21) to calculate a noiseless synthetic data vector, $\mathbf{d}$, of predicted seismic travel times at the sensor depths.

   (c) Solve for the slowness, $\mathbf{m}$, as a function of depth using your $\mathbf{G}$ matrix and analytically calculated noiseless travel times using the MATLAB backslash operator. Compare your result graphically with $\mathbf{m}_{true}$.

   (d) Generate a noisy travel time vector where independent normally distributed noise with a standard deviation of 0.05 ms is added to the elements of $\mathbf{d}$. Resolve the system for $\mathbf{m}$ and again compare your result graphically with $\mathbf{m}_{true}$. How has the model changed?

(e) Repeat the problem, but for just $n = 4$ sensor depths and corresponding equal length slowness intervals. Is the recovery of the true model improved? Explain in terms of the condition numbers of your **G** matrices.

4. Find a journal article that discusses the solution of an inverse problem in a discipline of special interest to you. What are the data? Are the data discrete or continuous? Have the authors discussed possible sources of noise in the data? What is the model? Is the model continuous or discrete? What physical laws determine the forward operator G? Is G linear or nonlinear? Do the authors discuss any issues associated with existence, uniqueness, or instability of solutions?

1.3 Solutions

1. Following the hint, we use the standard basis in R^n, $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$, along with superposition and scaling to write

$$G(\mathbf{m}) = G(m_1\mathbf{e}_1 + m_2\mathbf{e}_2 + \ldots + m_n\mathbf{e}_n)$$

$$G(\mathbf{m}) = m_1 G(\mathbf{e}_1) + m_2 G(\mathbf{e}_2) + \ldots + m_n G(\mathbf{e}_n) \ .$$

Let the ith column of $\mathbf{\Gamma}$ be

$$\mathbf{\Gamma}_{\cdot,i} = G(\mathbf{e}_i) \ .$$

Then

$$\begin{aligned} G(\mathbf{m}) &= m_1 G(\mathbf{e}_1) + m_2 G(\mathbf{e}_2) + \ldots + m_n G(\mathbf{e}_n) \\ &= m_1 \mathbf{\Gamma}_{\cdot,1} + m_2 \mathbf{\Gamma}_{\cdot,2} + \ldots + m_n \mathbf{\Gamma}_{\cdot,n} \\ &= \mathbf{\Gamma}\mathbf{m} \ . \end{aligned}$$

2. The system of equations is

$$\begin{bmatrix} 1 & t_1 & -(1/2)t_1^2 \\ 1 & t_2 & -(1/2)t_2^2 \\ 1 & t_3 & -(1/2)t_3^2 \\ . & . & . \\ . & . & . \\ . & . & . \\ 1 & t_m & -(1/2)t_m^2 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_m \ . \end{bmatrix} .$$

A straightforward way to make this system inconsistent is to require that the ballistic body be in two places at the same time, e.g., $t_1 = t_2$ and $y_1 \neq y_2$. This is a physically impossible situation that the parabolic trajectory law can never satisfy, and the mathematical result is an inconsistent set of equations.

If the t_i are distinct, then it can be shown, by examining the reduced row echelon form of the augmented matrix, that there are always exact solutions for $m = 2$ and $m = 3$.

For the $m = 2$ case we have

$$RREF \begin{bmatrix} 1 & t_1 & -\frac{1}{2}t_1^2 & y_1 \\ 1 & t_2 & -\frac{1}{2}t_2^2 & y_2 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & -\frac{1}{2}\left(t_1^2 - \frac{t_1(t_1^2-t_2^2)}{t_2-t_1}\right) & y_1 - \frac{t_1(y_2-y_1)}{t_2-t_1} \\ 0 & 1 & -\frac{1}{2}\frac{t_2^2-t_1^2}{t_2-t_1} & \frac{y_2-y_1}{t_2-t_1} \end{bmatrix} . \quad (1.3)$$

If the t_i are distinct, we can always solve for m_1 and m_2 in terms of the free parameter m_3, and there are infinitely many solutions that fit the data. This shows that there are an infinite number of t-symmetric parabolas that fit through any two points in the $t - y$ plane, provided the t_i are distinct.

For $m = 3$, the corresponding expression is

$$RREF \begin{bmatrix} 1 & t_1 & -\frac{1}{2}t_1^2 & y_1 \\ 1 & t_2 & -\frac{1}{2}t_2^2 & y_2 \\ 1 & t_3 & -\frac{1}{2}t_3^2 & y_3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & D^{-1}\left(y_1(t_3t_2^2 - t_2t_3^2) + y_2(t_3^2t_1 - t_1^2t_3) + y_3(t_1^2t_2 - t_2^2t_1)\right) \\ 0 & 1 & 0 & D^{-1}\left(y_1(t_3^2 - t_2^2) + y_2(t_1^2 - t_3^2) + y_3(t_2^2 - t_1^2)\right) \\ 0 & 0 & 1 & 2D^{-1}\left(y_1(t_3 - t_2) + y_2(t_1 - t_3) + y_3(t_2 - t_1)\right) \end{bmatrix} .$$

where

$$D = t_1^2(t_2 - t_3) + t_2^2(t_3 - t_1) + t_3^2(t_1 - t_2) \; .$$

If the t_i are distinct, there is always a unique solution. This shows that there is always a unique t-symmetric parabola through any three points in the $t - y$ plane, provided that the t_i are distinct.

3. The **G** matrix is just an upper-triangular matrix with nonzero entries of the data discretization interval (0.2). The data vector, **d**, is analytically calculated from the definite interval from zero to z of the slowness functions, so that

$$d(z_i) = (1/g)(\ln(v_0 + kz_i) = \ln(v_0) \; . \quad (1.4)$$

The following MATLAB code produces plots of theoretical travel times, noise-free solutions, and noisy solutions. The condition number of the $n = 100$ system is approximately 128, but for $n = 4$ it is only about 5.4. The coarser discretization thus produces a much more stable solution method, but at a cost of brutally discretizing the model so that any fine-scale detail of the slowness structure with depth will not be recovered. This is a demonstration of regularization by discretization, such as is also demonstrated in Example 3.3.

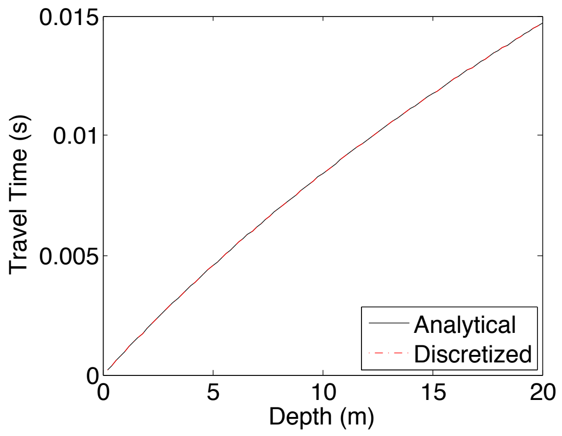

Figure 1.1: Analytical and discretized travel time calculations, $n = 100$.

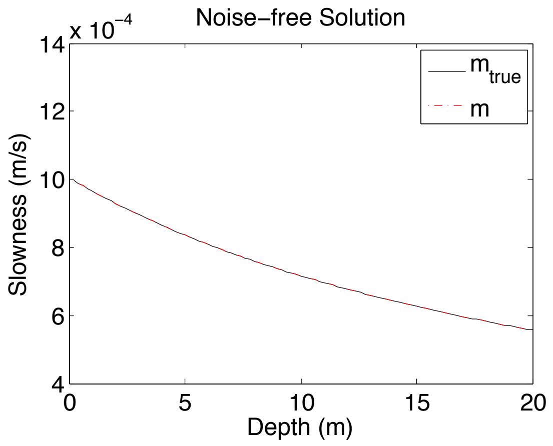

Figure 1.2: True and recovered models for noise-free data, $n = 100$.

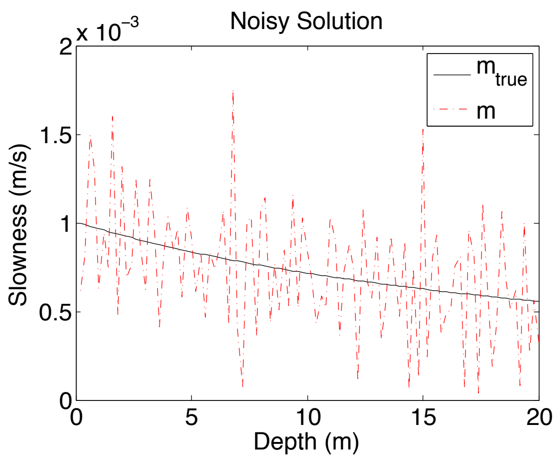

Figure 1.3: True and recovered models for noisy ($\sigma = 0.05$ ms) data, $n = 100$.

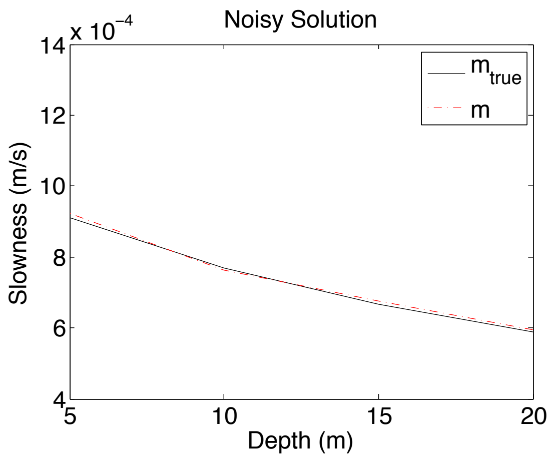

Figure 1.4: True and recovered models for noisy ($\sigma = 0.05$ ms) data, $n = 4$.

```
%Solution to borehole discretization Exercise 1.3
clear
%number and location of seismometer depths (change n to 4 for the part (e)).
n=100;
%n=4;
z=linspace(0,20,n+1)';
z=z(2:end);
Deltaz=z(2)-z(1);
%velocity gradient
g=40;
%velocity at z=0;
v0=1000;
%true velocity at midpoints
v=v0+(z-Deltaz/2)*g;
%true slowness
s=1./v;
%perfect data (analytic solution)
t=(1/g)*(log(v0+z*g)-log(v0));
%G matrix
G=tril(ones(n,n))*Deltaz;

figure(1)
bookfonts
plot(z,t,'k',z,G*s,'r-.')
xlabel('Depth (m)')
ylabel('Travel Time (s)')
legend('Analytical','Discretized','location','southeast')

figure(2)
bookfonts
plot(z,s,'k',z,G\t,'r-.')
xlabel('Depth (m)')
ylabel('Slowness (m/s)')
legend('m_{true}','m')
title('Noise-free Solution')

%add noise to the travel time data vector
tn=t+0.00005*randn(size(t));
figure(3)
bookfonts
plot(z,s,'k',z,G\tn,'r-.')
xlabel('Depth (m)')
ylabel('Slowness (m/s)')
legend('m_{true}','m')
title('Noisy Solution')
```

4. Answers to this question will vary tremendously depending on the choice of topic and specific paper.

Chapter 2

Linear Regression

2.1 Discussion of the Chapter

The main point of Chapter 2 is to summarize the solution of well conditioned discrete linear inverse problems (linear regression problems). This class of parameter estimation problems can easily be solved and statistically analyzed. Linear regression problems are introduced in Section 2.1. The statistical aspects of least squares linear regression problems are discussed in Section 2.2. In Section 2.3 we further explore the 95% confidence ellipsoid for least-squares problems. In some cases measurement standard deviations are not available. A methodology for dealing with this situation is described in Section 2.4. In Section 2.5 we discuss methods for linear regression that are robust in the face of incorrect data points by introducing L_1 residual vector norm minimization as a robust regression technique and show how to solve such problems with iteratively reweighted least squares. Monte Carlo methods for propagating data uncertainties into uncertainties in estimated model parameters are covered in Section 2.6.

Exercise 2.1 is an extended case study in that uses linear regression techniques to analyze a simple seismic profiling experiment. The problem is solved using both least squares and L_1 residual vector norm minimization. Exercise 2.2 is a theoretical exercise in which students work out how to formulate and solve a linear regression problem with correlated data errors. Exercise 2.3 is a simulation where students will observe the statistical properties of parameter estimates under assumptions that the data noise is known or unknown. Exercise 2.4 is a theoretical exercise to demonstrate that p-values in a linear regression problem are uniformly distributed. Exercise 2.5 demonstrates that some seemingly easy linear regression problems can be extremely poorly conditioned in practice. It turns out that this problem can be solved easily after switching to a more appropriate basis. This problem reappears in Exercise 5.5.

This is the first really "technical" chapter in the book, and students tend to find this material more challenging, both conceptually and in terms of programming, than the material in Chapter 1 and the appendices. Most of the MATLAB programming can be gleaned from the Example scripts for beginning programmers. We typically cover the material in this chapter in about four one-hour lectures.

2.2 Exercises

1. A seismic profiling experiment is performed where the first arrival times of seismic energy from a mid–crustal refractor are observed at distances (in kilometers) of

$$\mathbf{x} = \begin{bmatrix} 6.0000 \\ 10.1333 \\ 14.2667 \\ 18.4000 \\ 22.5333 \\ 26.6667 \end{bmatrix} \tag{2.1}$$

from the source, and are found to be (in seconds after the source origin time)

$$\mathbf{t} = \begin{bmatrix} 3.4935 \\ 4.2853 \\ 5.1374 \\ 5.8181 \\ 6.8632 \\ 8.1841 \end{bmatrix} . \tag{2.2}$$

These vectors can also be found in the MATLAB data file **profile.mat**. A two–layer flat Earth structure gives the mathematical model

$$t_i = t_0 + s_2 x_i \tag{2.3}$$

where the intercept time, t_0 depends on the thickness and slowness of the upper layer, and s_2 is the slowness of the lower layer. The estimated noise in the first arrival time measurements is believed to be independent and normally distributed with expected value 0 and standard deviation $\sigma = 0.1$ s.

   (a) Find the least squares solution for the model parameters t_0 and s_2. Plot the data, the fitted model, and the residuals.

   (b) Calculate and comment on the model parameter correlation matrix (e.g., 2.43). How are the correlations manifested in the general appearance of the error ellipsoid in $(t_0,\ s_2)$ space?

   (c) Plot the error ellipsoid in the (t_0, s_2) plane and calculate conservative 95% confidence intervals for t_0 and s_2 for the appropriate value of Δ^2. Hint: The following MATLAB function will plot a two–dimensional covariance ellipse about the model parameters, where **C** is the covariance matrix, DELTA2 is Δ^2, and **m** is the 2–vector of model parameters.

```
%set the number of points on the ellipse to generate and plot
function plot_ellipse(DELTA2,C,m)
n=100;
%construct a vector of n equally-spaced angles from (0,2*pi)
theta=linspace(0,2*pi,n)';
%corresponding unit vector
xhat=[cos(theta),sin(theta)];
Cinv=inv(C);
%preallocate output array
r=zeros(n,2);
for i=1:n
%store each (x,y) pair on the confidence ellipse
%in the corresponding row of r
r(i,:)=sqrt(DELTA2/(xhat(i,:)*Cinv*xhat(i,:)'))*xhat(i,:);
end
plot(m(1)+r(:,1), m(2)+r(:,2));
axis equal
```

(d) Evaluate the p–value for this model. You may find the library function **chi2cdf** to be useful here.

(e) Evaluate the value of χ^2 for 1000 Monte Carlo simulations using the data prediction from your model perturbed by noise that is consistent with the data assumptions. Compare a histogram of these χ^2 values with the theoretical χ^2 distribution for the correct number of degrees of freedom. You may find the library function **chi2pdf** to be useful here.

(f) Are your p–value and Monte Carlo χ^2 distribution consistent with the theoretical modeling and the data set? If not, explain what is wrong.

(g) Use IRLS to find 1–norm estimates for t_0 and s_2. Plot the data predictions from your model relative to the true data and compare with (a).

(h) Use Monte Carlo error propagation and IRLS to estimate symmetric 95% confidence intervals on the 1–norm solution for t_0 and s_2.

(i) Examining the contributions from each of the data points to the 1–norm misfit measure, can you make a case that any of the data points are statistical outliers?

2. In this chapter we have largely assumed that the data errors are independent. Suppose instead that the data errors have an MVN distribution with expected value **0** and a covariance matrix $\mathbf{C}_D$. It can be shown that

the likelihood function is then

$$L(\mathbf{m}|\mathbf{d}) = \frac{1}{(2\pi)^{m/2}} \frac{1}{\sqrt{\det(\mathbf{C}_D)}} e^{-(\mathbf{Gm}-\mathbf{d})^T \mathbf{C}_D^{-1} (\mathbf{Gm}-\mathbf{d})/2} . \tag{2.4}$$

(a) Show that the maximum likelihood estimate can be obtained by solving the minimization problem

$$\min \ (\mathbf{Gm} - \mathbf{d})^T \mathbf{C}_D^{-1} (\mathbf{Gm} - \mathbf{d}) . \tag{2.5}$$

(b) Show that (2.5) can be solved using the system of equations

$$\mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{Gm} = \mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{d} . \tag{2.6}$$

(c) Show that (2.5) is equivalent to the linear least squares problem

$$\min \|\mathbf{C}_D^{-1/2} \mathbf{Gm} - \mathbf{C}_D^{-1/2} \mathbf{d}\|_2 \tag{2.7}$$

where $\mathbf{C}_D^{-1/2}$ is the matrix square root of $\mathbf{C}_D^{-1}$.

(d) The Cholesky factorization of $\mathbf{C}_D^{-1}$ can also be used instead of the matrix square root. Show that (2.5) is equivalent to the linear least squares problem

$$\min \|\mathbf{RGm} - \mathbf{Rd}\|_2 \tag{2.8}$$

where $\mathbf{R}$ is the Cholesky factor of $\mathbf{C}_D^{-1}$.

3. Use MATLAB to generate 10,000 realizations of a data set of $m = 5$ points $\mathbf{d} = a + b\mathbf{x} + \boldsymbol{\eta}$, where $\mathbf{x} = [1,\ 2,\ 3,\ 4,\ 5]^T$, the $n = 2$ true model parameters are $a = b = 1$, and $\boldsymbol{\eta}$ is an m-element vector of independent $N(0, 1)$ noise.

(a) Assuming that the noise standard deviation is known *a priori* to be 1, solve for the parameters a and b using least squares for each realization and histogram them in 100 bins.

(b) Calculate the parameter covariance matrix, $\mathbf{C} = \sigma^2(\mathbf{G}^T\mathbf{G})^{-1}$, assuming independent $N(0,\ 1)$ data errors, and give standard deviations, σ_a and σ_b, for your estimates of a and b estimated from $\mathbf{C}$,

(c) Calculate standardized parameter estimates

$$a' = \frac{a - 1}{\sqrt{C_{1,1}}} \tag{2.9}$$

and

$$b' = \frac{b - 1}{\sqrt{C_{2,2}}} \tag{2.10}$$

for your solutions for a and b. Demonstrate using a Q–Q plot (Appendix B) that your estimates for a' and b' are distributed as $N(0,\ 1)$.

(d) Show using a Q–Q plot that the squared residual lengths

$$\|\mathbf{r}\|_2^2 = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 \tag{2.11}$$

for your solutions in (a) are distributed as χ^2 with $m - n = \nu = 3$ degrees of freedom.

(e) Assume that the noise standard deviation for the synthetic data set is not known, and instead estimate it for each realization, k, as

$$s_k = \sqrt{\frac{1}{n-m}\sum_{i=1}^{m} r_i^2} \ . \tag{2.12}$$

Histogram your standardized solutions

$$a' = \frac{a - \bar{a}}{\sqrt{C'_{1,1}}} \tag{2.13}$$

and

$$b' = \frac{b - \bar{b}}{\sqrt{C'_{2,2}}} \tag{2.14}$$

where $\mathbf{C}' = s_k^2(\mathbf{G}^T\mathbf{G})^{-1}$ is the covariance matrix estimation for the k^{th} realization.

(f) Demonstrate using a Q–Q plot that your estimates for a' and b' are distributed as the Student's t distribution with $\nu = 3$ degrees of freedom.

4. Suppose that we analyze a large number of data sets $\mathbf{d}$ in a linear regression problem and compute p–values for each data set. The χ^2_{obs} values should be distributed according to a χ^2 distribution with $m - n$ degrees of freedom. Show that the corresponding p–values will be uniformly distributed between 0 and 1.

5. Use linear regression to fit a polynomial of the form

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_{19} x_i^{19} \tag{2.15}$$

to the noise–free data points

$$(x_i,\ y_i) = (-0.95,\ -0.95),\ (-0.85,\ -0.85),\ \ldots, (0.95,\ 0.95)\ . \tag{2.16}$$

Use the normal equations to solve the least squares problem.

Plot the data and your fitted model, and list the parameters, a_i obtained in your regression. Clearly, the correct solution has $a_1 = 1$, and all other $a_i = 0$. Explain why your answer differs.

2.3 Solutions

1. (a) The least squares model is easily found, using either the normal equations or the MATLAB backslash operator, to be $\mathbf{m}_{L_2} = [2.0323,\ 0.2203]$. Figure 2.1 shows the fit of the predicted data from this model relative to the actual data and standard errors. The fit isn't too bad, although the last data point is perhaps a bit suspicious.

   (b) Figure 2.2 shows the appropriate error ellipsoid for 95% confidence (2.45σ) appropriate for two joint parameters in $n = 2$ dimensions. The associated covariance matrix is

$$\mathbf{C} = \sigma^2(\mathbf{G}^T\mathbf{G})^{-1} = \begin{bmatrix} 0.01058966 & -0.00054631 \\ -0.00054630 & 0.00003345 \end{bmatrix} .$$

   The error ellipsoid gives the axes for the MVN distribution that describe the parameter uncertainty. Parameter standard errors are given by the square roots of the diagonal elements of $\mathbf{C}$. For single parameters, which will be normally distributed for least-squares problems with normal data noise, the 95% confidence intervals are defined by $\pm 1.96\sigma$. However, for an MVN distribution in 2 dimensions, the more conservative approach is to go out to about $\pm 2.45\sigma$ along the error ellipsoid principal axes to encompass 95% of the joint probability within the 2-dimensional MVN distribution. This gives

$$\sigma_{1,2.45} = 2.45\sqrt{C_{1,1}} \approx 0.2521$$

   and

$$\sigma_{2,2.45} = 2.45\sqrt{C_{2,2}} \approx 0.0142 ,$$

   which give parameter ranges of

$$t_0 \approx [1.78022,\ 2.28446]$$

   and

$$s_2 \approx [0.20611,\ 0.23445] .$$

   (c) The correlation matrix is

$$\begin{bmatrix} 1.0000 & -0.9179 \\ -0.9179 & 1.0000 \end{bmatrix} ,$$

   consistent with the error ellipsoid being tilted negatively in the (m_1,m_2) plane (Figure 2.2) and demonstrating a strong tradeoff between increasing/decreasing s_2 and decreasing/increasing t_0.

   (d) The p value is about 8.7×10^{-4}, which isn't outrageously small, but shows that a realization of the data with this relatively large value of χ^2 (18.75) for 4 degrees of freedom would only occur less than 1% of the time if our data were really drawn from a linear model and had the advertised standard errors.

(e) See Figure 2.3.

(f) Figure 2.3 demonstrates that the Monte Carlo values do indeed look like they are drawn from the theoretical PDF (this could be checked more thoroughly with a Q-Q plot). The p-value is way out on the tail of the distribution for this data set, and thus the modeling and/or data should be regarded with some suspicion. Possibilities include a data outlier or outliers (or equivalently, underestimation of the true data noise), or the data could be generated by some other model than a linear relationship between x and t. For example, we might have used the travel time for a phase from some unmodeled seismic ray path.

(g) The 1-norm solution is $m_{L_1} = [2.1786,\ 0.2079]$, which has a smaller x vs t slope (lesser slowness,or higher velocity) than for the 2-norm solution. The biggest difference in comparing the data fit is that the last data point looks even more like an outlier due to its large misfit (Figure 2.4)

(h) Sorting the 1-norm Monte Carlo parameter solutions and finding symmetric intervals where 95% of them are contained gives individual parameter confidence intervals of

$$[\pm 0.26188,\ \pm 0.014698]$$

for parameters t_0 and s_2, which are larger than the corresponding individual parameter 2-norm values of

$$\pm 1.96\sqrt{\mathrm{diag}(\mathbf{C})} = [\pm 0.2017,\ \pm 0.0113]\ .$$

The corresponding 1-norm 95% confidence intervals are $(1.9168,\ 2.4405)$ and $(0.1932,\ 0.2226)$.

A conservative estimate of the 95% confidence intervals for the IRLS solution using the same approach as the 2.45σ bounds for the $\mathbf{m_{L_2}}$ solution involves finding the necessary scaling factor, k, for the error ellipsoid defined by the empirical covariance matrix

$$\mathbf{C}_{\mathrm{emp}} = (\mathbf{m_{L_1}} - \bar{\mathbf{m}}_{\mathbf{L_1}})^{\mathbf{T}}(\mathbf{m_{L_1}} - \bar{\mathbf{m}}_{\mathbf{L_1}})\ ,$$

(where $\mathbf{m_{L_1}}$ is a 1000×2 matrix of Monte Carlo IRLS solutions) so that about 950 (95%) of the models are enclosed. The easiest way to do this is to work in the principal coordinate system of the error ellipsoid, defined by the eigenvectors and eigenvalues of $\mathbf{C}_{\mathrm{emp}}$

$$\mathbf{C}_{\mathrm{emp}} = \mathbf{U\Lambda U^T}\ .$$

Rotating the IRLS Monte Carlo models into the ellipsoid principal coordinate system

$$\mathbf{m}'_{\mathbf{L_1}} = \mathbf{m_{L_1}U}$$

and counting the models where

$$\frac{m'^2_{L_1(i,1)}}{\Lambda_{1,1}} + \frac{m'^2_{L_1(i,2)}}{\Lambda_{2,2}} \leq k^2$$

gives the confidence level for including points in an ellipsoid extending to $\pm k \Lambda_{i,i}^{1/2}$ in its principal directions. For this problem, about 950 of the models are included for $k \approx 2.5$, which gives a range of IRLS 1-norm solutions

$$t_0 \approx [1.8734,\ 2.4839]$$

and

$$s_2 \approx [0.1911,\ 0.2247]\ .$$

(i) The 1-norm residual contributions are the absolute value of the misfits normalized by σ

$$r_{L_1,i} = \left| \frac{(\mathbf{d} - \mathbf{Gm})_{(i)}}{\sigma} \right| = \begin{bmatrix} 0.0675 \\ 0.0000 \\ 0.0072 \\ 0.1858 \\ 0.0000 \\ 0.4616 \end{bmatrix}$$

and the 2-norm residual contributions are the squared misfits normalized by σ

$$r_{L_2,i} = \left(\frac{(\mathbf{d} - \mathbf{Gm})_{(i)}}{\sigma} \right)^2 = \begin{bmatrix} 0.1395 \\ 0.0208 \\ 0.0376 \\ 0.2674 \\ 0.1328 \\ 0.2776 \end{bmatrix}$$

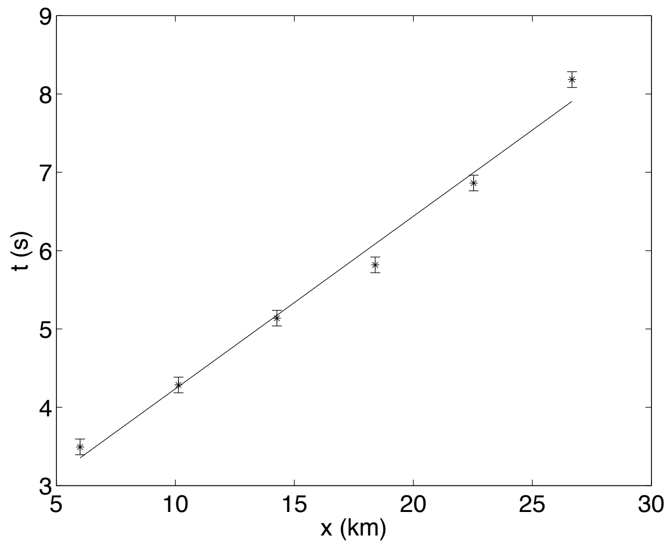

Figure 2.1: Fitted model.

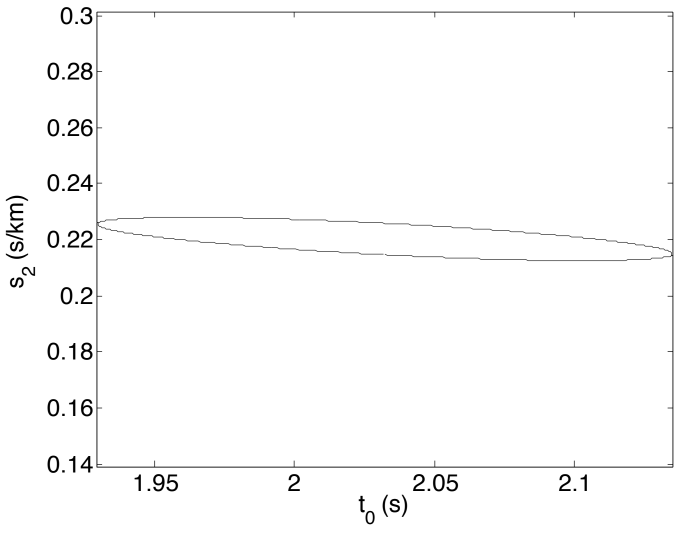

Figure 2.2: Error Ellipsoid.

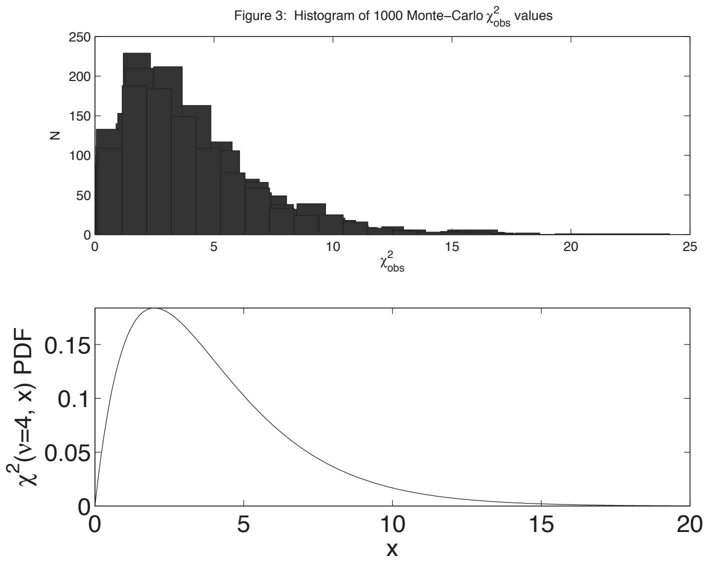

Figure 2.3: Histogram of observed χ^2 values compared to the χ^2 distribution for ν=4.

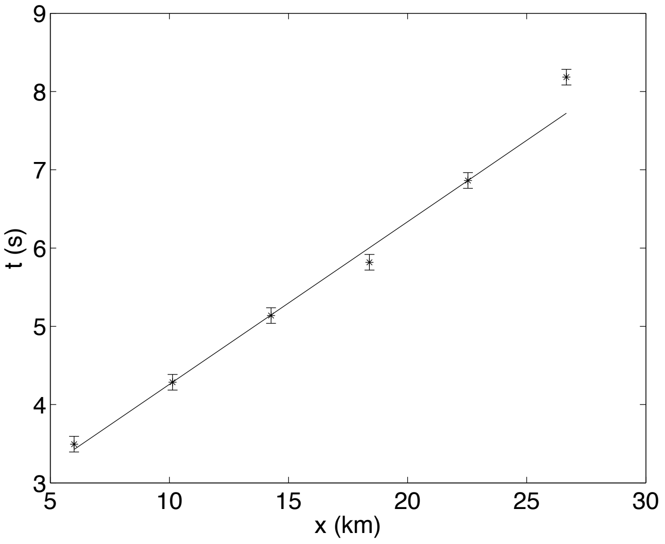

Figure 2.4: Data and fitted model.

```
%Solution to seismic profile Exercise 2.1
clear
%generate a linear trend from a refraction line from
%a two-layer model
d0=2;
v1=3;
v2=5;
%critical angle
thetac=asin(v1/v2);
%horizontal offset to critical refraction
x0=d0/tan(thetac);
%start of refraction
xstart=2*x0;

%end of refraction
xmax=5*xstart;
%x intercept
%t offset
%true parameters
t0=2*sqrt(d0^2+x0^2)/v1;
s2=1/v2;

%number of data points
M=6;

x=linspace(ceil(xstart),xmax,M)';

%solve the least squares regression
%generate a data set

G=[ones(M,1),x];

dof=M-2;

sig=0.1;

%data noise vector
noise=sig*ones(M,1);

%generate a new data vector
%t=t0+x/v2+noise.*randn(M,1);
%add an outlier
%t(M)=t(M)+0.8;

%load canned data (for HW)
t=[3.4935 ; 4.2853 ; 5.1374 ; 5.8181 ; 6.8632 ; 8.1841 ];

%weight the system
for i=1:M
Gw(i,:)=G(i,:)/noise(i);
end
tw=t./noise;

%a) find the 2-norm solution
disp('2-norm solution:')
m2=Gw\tw
pause
figure(1)
bookfonts
%plot the data predictions and data for the 2-norm solution
errorbar(x,t,noise,'*')
hold on
plot(x,m2(1)+x*m2(2),'r')
xlabel('x (km)')
ylabel('t (s)')
hold off
print -deps2 figure1.eps
pause

%b Generate the covariance and correlation matrices; plot the error ellipsoid
disp('covariance matrix:')
covm = sig^2*inv(G'*G)
disp('correlation matrix:')
s=sqrt(diag(covm));
rhom = covm./(s*s')
pause

figure(2)
bookfonts
[u,lam]=eig(covm);
theta=(0:.01:2*pi)';
%x component of the ellipsoid
r(:,1)=2.45*sqrt(lam(1,1))*u(1,1)*cos(theta)+sqrt(lam(2,2))*u(1,2)*sin(theta);
%y component of the ellipsoid
r(:,2)=2.45*sqrt(lam(1,1))*u(2,1)*cos(theta)+sqrt(lam(2,2))*u(2,2)*sin(theta);
plot(m2(1)+r(:,1),m2(2)+r(:,2))
axis tight
axis equal
xlabel(['t_0 (s)'])
ylabel(['s_2 (s/km)'])
print -deps2 figure2.eps
```

```
pause

%c do the chi-square calculations
disp('chi-square value:')
chi2=norm((G*m2-t)./noise)^2
disp('chi-square p value:')
p = 1 - chi2cdf(chi2,dof)
pause

%d Monte Carlo Calculations

%The baseline data set generated by the 2-norm model
disp('2-norm model predicted data:')
db=G*m2;

%number of realizations for Monte Carlo
NR=1000;

for i=1:NR
%generate the data vector for the ith Monte Carlo data set
dr=db+noise.*randn(M,1);
%calculate the weighted data vector
dw=dr./noise;
%find the 2-norm model for the ith Monte Carlo data set, stored as a column of mr2
mr2(:,i)=Gw\dw;
%calculate the chi-square value for the ith Monte Carlo data set
chi2r(i)=norm((G*mr2(:,i)-dr)./noise)^2;
end

%histogram the chi2 values for the Monte Carlo realizations
%and compare them with the chi2 PDF
figure(3)
bookfonts
NBIN=20;
%get the population of each of the NBIN bins for scaling the chi2pdf plot
subplot(2,1,1)
hist(chi2r,NBIN);
xlabel('\chi^2_{obs}')
ylabel('N')

hold on
xx=0:.1:20;
subplot(2,1,2)
chitheo=chi2pdf(xx,dof);
plot(xx,chitheo)
ylabel('\chi^2(\nu=4, x) PDF')
xlabel('x')
axis([0 20 0 max(chitheo)]);
hold off
print -deps figure3.eps
pause

%solve and plot the L-1 solution
disp('1-norm solution:')
m1=irls(Gw,tw,1.0e-8,1.0e-6,1,100)
pause

figure(4)
bookfonts
%plot the data predictions and data for the 1-norm solution
errorbar(x,t,noise,'*')
hold on
plot(x,m1(1)+x*m1(2),'r')
xlabel('x (km)')
ylabel('t (s)')

hold off
print -deps figure4.eps

for i=1:NR
%data vector for the ith Monte Carlo data set
dr=db+noise.*randn(M,1);
%weighted data vector
dw=dr./noise;
%2-norm model for the ith Monte Carlo data set, stored as a column of mr
mr1(:,i)=irls(Gw,dw,1.0e-8,1.0e-6,1,100);
end

%estimate L1 confidence intervals
%mean model
mrmean1=mean(mr1');
%find a range where 95% of the Monte Carlo models are found
%relative to the mean
m1sort=sort(abs(mr1(1,:)-mrmean1(1)));
m2sort=sort(abs(mr1(2,:)-mrmean1(2)));
m11conf=m1sort(round(0.95*NR));
m12conf=m2sort(round(0.95*NR));

disp(['m1 Monte Carlo modeling 95% confidence width (indiv. parameters):  (',  ...
num2str(m11conf),', ',num2str(m12conf),')'])
m11range = [ m1(1) - m11conf , m1(1) , m1(1) + m11conf ]
```

```
m12range = [ m1(2) - m12conf , m1(2) , m1(2) + m12conf ]

%Next we will evaluate the 95% joint confidence intervals
%Evaluate the empirical covariance matrix
covmemp=mr1'-[(mean(mr1(1,:))*ones(NR,1)),(mean(mr1(2,:))*ones(NR,1))];
covmemp=(covmemp'*covmemp)/NR;
%and diagonalize it
[u,lam]=eig(covmemp);
%rotate the model estimates into the ellipsoid principal coordinate system
mr1rot=mr1'*u;
%subtract the residual rotated parameter mean
mr1rotmean=mean(mr1rot);
mr1rot=mr1rot-[mr1rotmean(1)*ones(NR,1), mr1rotmean(2)*ones(NR,1)];
count=0;
%count the number of points out to some k*sigma
k=2.5;
for i=1:NR
if (mr1rot(i,1)^2/lam(1,1)+mr1rot(i,2)^2/lam(2,2) < k^2)
%count=count+1 if the NRth model is in the ellipsoid
count=count+1;
end
end
disp([num2str(k),' sigma confidence interval inclusion:'])
count/NR
sig11=sqrt(covmemp(1,1));
sig12=sqrt(covmemp(2,2));
m11conf=k*sig11;
m12conf=k*sig12;

disp(['m1 Monte Carlo modeling 95% confidence width (joint parameters):  (',  ...
num2str(m11conf),', ',num2str(m12conf),')'])
m11range = [ m1(1) - m11conf , m1(1) , m1(1) + m11conf ]
m12range = [ m1(2) - m12conf , m1(2) , m1(2) + m12conf ]
pause

%estimate L2 confidence intervals for comparison
%mean model
mrmean2=mean(mr2')'';
%find a range where 95% of the Monte Carlo models are found
%relative to the mean
m1sort=sort(abs(mr2(1,:)-mrmean2(1)));
m2sort=sort(abs(mr2(2,:)-mrmean2(2)));
m21conf=m1sort(round(0.95*NR));
m22conf=m2sort(round(0.95*NR));

disp(['m2 Monte Carlo modeling 95% confidence widths (indiv. parameters):  (',  ...
num2str(m21conf),', ',num2str(m22conf),')'])
m21range= [ m2(1) - m21conf , m2(1) , m2(1) + m21conf ]
m22range= [ m2(2) - m22conf , m2(2) , m2(2) + m22conf ]

disp(['m2 Monte Carlo modeling 95% confidence widths (joint parameters):  (',  ...
num2str(m21conf),', ',num2str(m22conf),')'])
m21range= [ m2(1)-m21conf , m2(1) , m2(1)+m21conf ]
m22range= [ m2(2)-m22conf , m2(2) , m2(2)+m22conf ]

disp('m2 covariance-derived 95% confidence (indiv. parameters)')
m21range = [ m2(1)-1.96*sqrt(covm(1,1)) , m2(1) , m2(1)+1.96*sqrt(covm(1,1)) ]
m22range = [ m2(2)-1.96*sqrt(covm(2,2)) , m2(2) , m2(2)+1.96*sqrt(covm(2,2)) ]

disp('m2 covariance-derived 95% confidence (joint parameters)')
m21range = [ m2(1)-2.45*sqrt(covm(1,1)), m2(1), m2(1)+2.45*sqrt(covm(1,1)) ]
m22range = [ m2(2)-2.45*sqrt(covm(2,2)), m2(2), m2(2)+2.45*sqrt(covm(2,2)) ]

%examine the 1-norm residuals
disp('1-norm residuals terms')
abs(G*m1-t)
disp('2-norm residual (squared) terms')
abs(G*m2-t)
```

2. (a) To maximize the likelihood, we need to maximize the exponent. This is equivalent (as can be seen by inspection) to minimizing

$$f(\mathbf{m}) = (\mathbf{Gm} - \mathbf{d})^T \mathbf{C}_D^{-1} (\mathbf{Gm} - \mathbf{d}) \ .$$

(b) Multiplying out the expression gives

$$f(\mathbf{m}) = \mathbf{m}^T \mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{Gm} - \mathbf{m}^T \mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{d} - \mathbf{d}^T \mathbf{C}_D^{-1} \mathbf{Gm} + \mathbf{d}^T \mathbf{C}_D^{-1} \mathbf{d} \ .$$

Because f($\mathbf{m}$) is a scalar function, the second and third terms (which are transposes of each other) are equal, so that

$$f(\mathbf{m}) = \mathbf{m}^T \mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{Gm} - 2\mathbf{d}^T \mathbf{C}_D^{-1} \mathbf{Gm} + \mathbf{d}^T \mathbf{C}_D^{-1} \mathbf{d} \ .$$

Taking the gradient (applying Theorem C.2 to the first term, and noting that the last term is a constant) gives

$$\nabla f(\mathbf{m}) = 2\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} - 2\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{d} \ .$$

Setting $\nabla f(\mathbf{m}) = \mathbf{0}$, we obtain the desired result

$$\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} = \mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{d} \ .$$

(c) Minimizing $\|\mathbf{C}_D^{-1/2}\mathbf{G}\mathbf{m} - \mathbf{C}_D^{-1/2}\mathbf{d}\|_2$ is equivalent to minimizing

$$f(\mathbf{m}) = \|\mathbf{C}_D^{-1/2}\mathbf{G}\mathbf{m} - \mathbf{C}_D^{-1/2}\mathbf{d}\|_2^2$$

$$f(\mathbf{m}) = (\mathbf{C}_D^{-1/2}\mathbf{G}\mathbf{m} - \mathbf{C}_D^{-1/2}\mathbf{d})^T(\mathbf{C}_D^{-1/2}\mathbf{G}\mathbf{m} - \mathbf{C}_D^{-1/2}\mathbf{d})$$

Distributing the transpose and multiplying out the terms gives

$$f(\mathbf{m}) = \mathbf{m}^T\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} - \mathbf{m}^T\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{d} - \mathbf{d}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} + \mathbf{d}^T\mathbf{C}_D^{-1}\mathbf{d}$$

which is the same as the expression in part (b).

(d) Minimizing $\|\mathbf{R}\mathbf{G}\mathbf{m} - \mathbf{R}\mathbf{d}\|_2$ is equivalent to minimizing

$$f(\mathbf{m}) = \|\mathbf{R}\mathbf{G}\mathbf{m} - \mathbf{R}\mathbf{d}\|_2^2$$

$$f(\mathbf{m}) = (\mathbf{R}\mathbf{G}\mathbf{m} - \mathbf{R}\mathbf{d})^T(\mathbf{R}\mathbf{G}\mathbf{m} - \mathbf{R}\mathbf{d})$$

Distributing the transpose and using $\mathbf{C}_D^{-1} = \mathbf{R}^T\mathbf{R}$ gives

$$f(\mathbf{m}) = \mathbf{m}^T\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} - \mathbf{m}^T\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{d} - \mathbf{d}^T\mathbf{C}_D^{-1}\mathbf{G}\mathbf{m} + \mathbf{d}^T\mathbf{C}_D^{-1}\mathbf{d}$$

which is the same as the expression in part (b).

3. (a) Figure 2.5 shows the histograms of the model parameters.

(b) The covariance matrix is

$$\mathbf{C} = \begin{bmatrix} 1.1000 & -0.3000 \\ -0.3000 & 0.1000 \end{bmatrix}$$

and the corresponding model standard deviations are given by the square roots of the diagonals

$$(\sigma_a, \ \sigma_b) = (1.0488, \ 0.3162) \ .$$

(c) The appropriate Q-Q plots are shown in Figure 2.6.

(d) See Figure 2.7.

(e) See Figures 2.9 and 2.10 (below).

(f) The appropriate plot is Figure 2.8. Figures 2.9 and 2.10 also show histograms of the standardized solutions plotted relative to the $N(0, \ 1)$ and $t(\nu = 3)$ distributions (the t distributions are wider) .

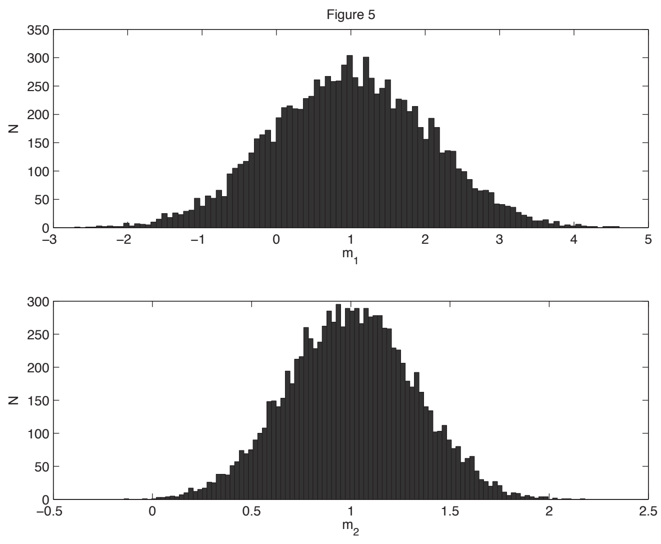

Figure 2.5: Histogram of the model parameters.

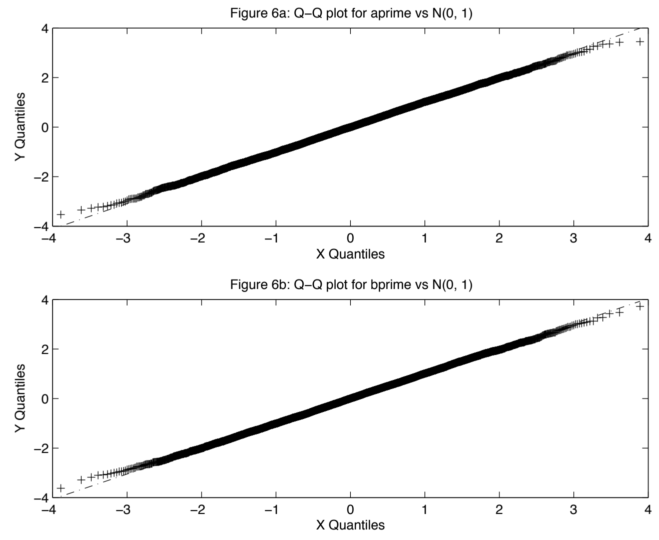

Figure 2.6: Q-Q plots.

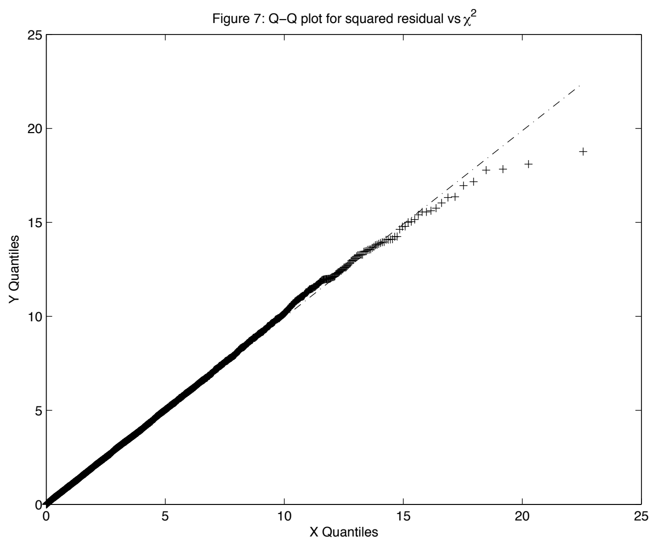

Figure 2.7: χ^2 Q-Q plot.

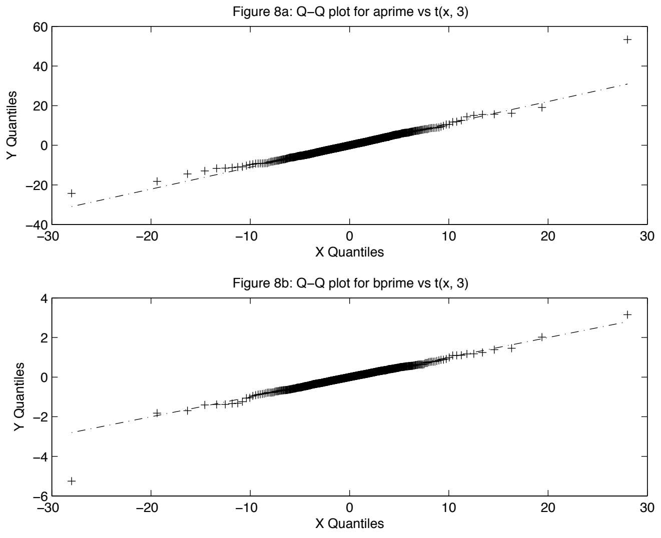

Figure 2.8: t distribution Q-Q plots.

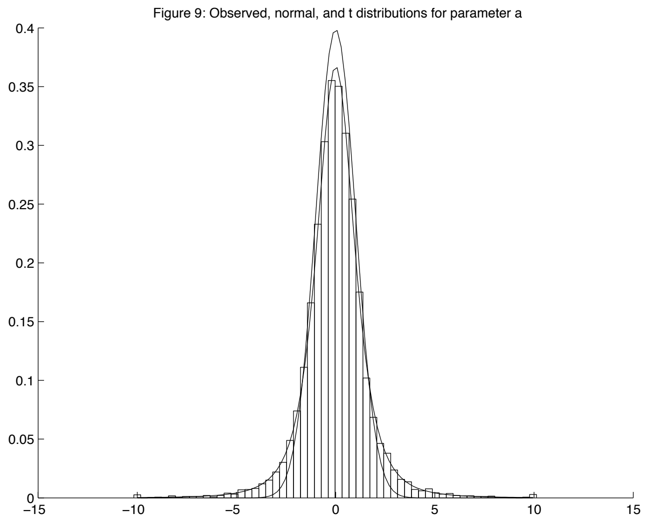

Figure 2.9: Histogram of solutions versus a N(0,1) distribution.

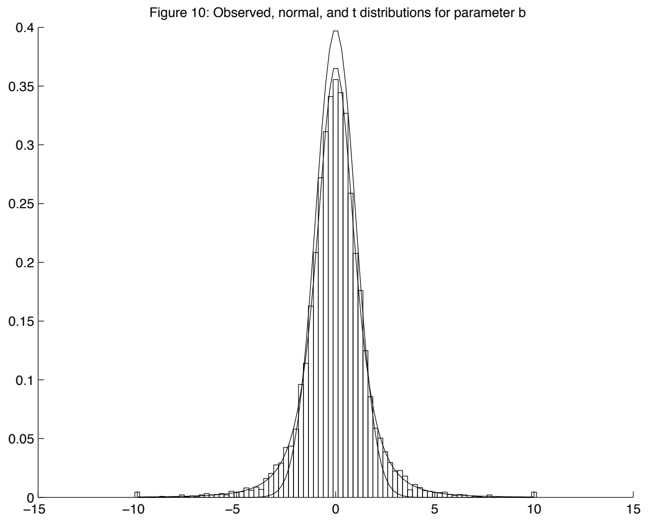

Figure 2.10: Histogram of solutions versus a t distribution with $\nu = 3$.

```
%Solution to Exercise 2.3
%Test t, Chi2, and normal distributions
%using a simple linear regression example
%note that qqplot, chi-square and t distribution calls utilize the MATLAB
%statistics toolbox
clear

%true intercept
a=1;
%true slope
b=1;

%data standard deviation
sigma=1.;

%set number of realizations
nreal=10000;

%NDATA, NDOF
ndata=5;
dof=ndata-2;

x=linspace(1,ndata,ndata)';
G=[ones(ndata,1),linspace(1,ndata,ndata)'];

%covariance
disp('covariance matrix:')
covm=inv(G'*G)

disp('parameter standard deviations:')
sigm1=sigma*sqrt(covm(1,1))
sigm2=sigma*sqrt(covm(2,2))

%generate the synthetic data sets
for NR=1:nreal

  d=a+b*x+sigma*randn(ndata,1);

%least-squares solution for realization NR.
  m(:,NR)=G\d;

%least-squares solution residual for realization NR.
  r=G*m(:,NR)-d;
  residsq(NR)=norm(r)^2/sigma^2;

%standard deviation estimate for this realization NR.
  s=std(r)*sqrt((ndata-1)/(ndata-2));
  sest(NR)=s;

end

%Histograms of the 2-norm models
figure(5)
subplot(2,1,1)
hist(m(1,:),100)
xlabel('m_1');
ylabel('N');
title('Figure 5')
subplot(2,1,2)
hist(m(2,:),100)
xlabel('m_2');
ylabel('N');
print -deps figure5.eps

%Q-Q plots for standard errors
figure(6)
aprime=(m(1,:)-mean(m(1,:)))/sigm1;
bprime=(m(2,:)-mean(m(2,:)))/sigm2;
x= norminv(((1:NR)-0.5)/NR);
y=sort(aprime);
subplot(2,1,1)
qqplot(x,y)
title('Figure 6a: Q-Q plot for aprime vs N(0, 1)')
subplot(2,1,2)
y=sort(bprime);
qqplot(x,y)
title('Figure 6b: Q-Q plot for bprime vs N(0, 1)')
print -deps figure6.eps

%Q-Q plot for chi-square
figure(7)
y = sort(residsq);
pp=((1:NR)-0.5)/NR;
%loop here in case non-vectorized chi2inv function is being used
for i=1:length(pp)
x(i) = chi2inv(pp(i),dof);
end
plot(x,y,'*');
title('Figure 7: Q-Q plot for squared residual vs \chi^2')
print -deps figure7.eps
```

```
%estimated standard deviation Q-Q plots
figure(8)
aprime=(m(1,:)-mean(m(1,:)))./sest*sqrt(covm(1,1));
bprime=(m(2,:)-mean(m(2,:)))./sest*sqrt(covm(2,2));
pp=((1:NR)-0.5)/NR;
%loop here in case non-vectorized chi2inv function is being used
for i=1:length(pp)
x(i) = tinv(pp(i),dof);
end
y=sort(aprime);
subplot(2,1,1)
qqplot(x,y)
title('Figure 8a: Q-Q plot for aprime vs t(x, 3)')
subplot(2,1,2)
y=sort(bprime);
plot(x,y,'*')
title('Figure 8b: Q-Q plot for bprime vs t(x, 3)')
print -deps figure8.eps

%
% Examine the distribution of estimates for intercept (a).
%
figure(9)
%
% Use the following bins for histograms.
%
bins=-10:0.35:10.0;
[Nh,xh]=hist((m(1,:)-ones(size(m(1,:)))).(sest*sigm1),bins);
sp=xh(2)-xh(1);
clf
hold on
for j=1:length(xh),
 plot([xh(j)-sp/2 xh(j)-sp/2 xh(j)+sp/2 xh(j)+sp/2],...
[0 Nh(j)/(sp*sum(Nh)) Nh(j)/(sp*sum(Nh)) 0]);
end;

xd=linspace(min(xh),max(xh),100);
tp=tpdf(xd,dof);
plot(xd,tp,'r')
np= normpdf(xd,0,1);
plot(xd,np,'g')
hold off
title('Figure 9: Observed, normal, and t distributions',...
      ' for parameter a');
print -deps figure9.eps
%
% Examine the distribution of estimates for slope (b).
%
figure(10)
%
% Use the following bins for histograms.
%
bins=-10:0.25:10.0;
[Nh,xh]=hist((m(2,:)-ones(size(m(2,:)))).(sest*sigm2),bins);
sp=xh(2)-xh(1);
clf
hold on
for j=1:length(xh),
 plot([xh(j)-sp/2 xh(j)-sp/2 xh(j)+sp/2 xh(j)+sp/2],...
[0 Nh(j)/(sp*sum(Nh)) Nh(j)/(sp*sum(Nh)) 0]);
end;

xd=linspace(min(xh),max(xh),100);
tp=tpdf(xd,dof);
plot(xd,tp,'r')
np= normpdf(xd,0,1);
plot(xd,np,'g')
hold off
title('Figure 10: Observed, normal, and t distributions',...
      ' for parameter b');
print -deps figure10.eps
```

4. Let $F(x)$ be the χ^2 CDF for $m-n$ degrees of freedom. For $0 \leq a \leq 1$,

$$P(p \leq a) = P(\int_{\chi^2_{\text{obs}}}^{\infty} f_{\chi^2}(x)dx \leq a).$$

$$P(p \leq a) = P(1 - F(\chi^2_{\text{obs}}) \leq a).$$

$$P(p \leq a) = P(-F(\chi^2_{\text{obs}}) \leq a - 1).$$

$$P(p \leq a) = P(F(\chi^2_{\text{obs}}) \geq 1 - a).$$

Since F is monotone increasing,

$$P(p \leq a) = P(\chi^2_{\text{obs}} \geq F^{-1}(1-a)).$$

$$P(p \leq a) = 1 - P(\chi^2_{\text{obs}} \leq F^{-1}(1-a)).$$

$$P(p \leq a) = 1 - F(F^{-1}(1-a)).$$

$$P(p \leq a) = 1 - (1-a).$$

$$P(p \leq a) = a.$$

Thus p is uniformly distributed between 0 and 1!

5. There are a couple of interesting numerical problems here. First, the $\mathbf{G}^T\mathbf{G}$ matrix is extremely ill-conditioned (with a condition number greater than 5×10^{17}). As a result, the solution to the normal equations is extremely inaccurate and the straightforward solution method produces a wildly inaccurate result.

   Note that the condition number of $\mathbf{G}$ by itself is about 5×10^8, so that by using the QR factorization or SVD to solve the least squares problem we could obtain a much better estimate of the fitted parameters.

   The fitted coefficients and relevant condition number are:

```
yfit =


  Columns 1 through 7

    0.0000    1.0000   -0.0000    0.0000    0.0000    0.0003   -0.0000

  Columns 8 through 14

   -0.0274    0.0006   -0.0935   -0.0031   -0.2789    0.0094   -2.5239

  Columns 15 through 20

   -0.0138   -2.2797    0.0041    0.9930   -0.0027    0.1201

cond(G'*G) is

ans =

   5.6876e+17
```

   Second, the obvious way to evaluate the fitted polynomial leads to even more roundoff error. A more sophisticated (well-posed) approach is used by the MATLAB **polyval** function, which produces a more reasonable (but still imperfect) result.

Figure 2.11 shows the results plotted using a straightforward evaluation of the polynomial coefficients using the normal equations. The true solution has $a_1 = 1$ and the rest of the coefficients equal to zero. However, the obtained solution has significant higher-degree polynomial coefficients that lead to a very poor result for the regression. Figure 2.12 shows the results obtained using polyval, which uses a much better posed formulation.

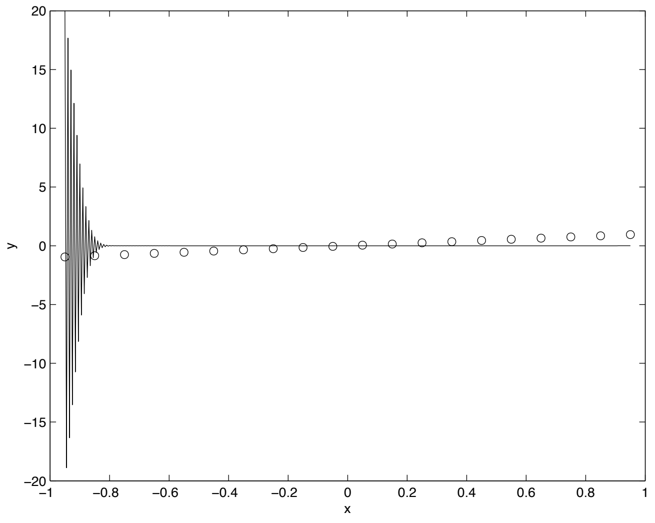

Figure 2.11: Fitted polynomial using straight forward evaluation.

```
%Ill-posed regression Exercise 2.5
%Set up the problem.
%
x=(-0.95:0.1:0.95)';
ytrue=x;
%
% Construct the G matrix.
%
G=zeros(20,20);
for i=0:19,
  G(:,i+1)=x.^i;
end;
%
% Find the least squares solution.
%
yfit=inv(G'*G)*G'*ytrue;
yfit'
disp('cond(G''*G) is ');
cond(G'*G)
disp('cond(G) is ');
cond(G)
%
% Plot out the results.
%
figure(1);
xdetailed=(-0.95:0.005:0.95)';
yfitval=zeros(size(xdetailed));
for i=1:length(xdetailed),
  for j=1:20,
    yfitval(i)=yfitval(i)+xdetailed(i)^(i-1);
  end;
end;
plot(xdetailed,yfitval,'k');
hold on
plot(x,ytrue,'ko');
xlabel('x');
```

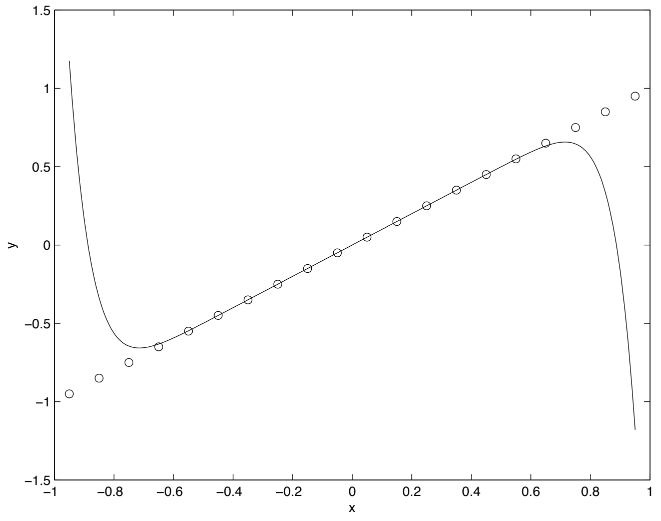

Figure 2.12: Fitted polynomial using polyval.

```
ylabel('y');
print -deps prob5a.eps
%
% Use a better-posed scheme for computing the values of the
% polynomial.
%
figure(2);
yfitp=flipud(yfit);
plot(xdetailed,polyval(yfitp,xdetailed),'k');
hold on
plot(x,ytrue,'ko');
xlabel('x');
ylabel('y');
print -deps prob5b.eps
```

Chapter 3

Rank Deficiency and Ill–Conditioning

3.1 Discussion of the Chapter

In this chapter we begin to analyze poorly conditioned linear parameter estimation problems. The basic analysis tool used in this chapter is the singular value decomposition (SVD). In Section 3.1, we introduce the SVD and the associated Moore-Penrose pseudoinverse, describing its relationship to least squares and minimum length solutions. In Section 3.2 we discuss the covariance of the pseudoinverse solution and introduce the concepts of model and data resolution. In Section 3.3 we consider the instability issues that can arise in the inverse solution and show how the presence of small singular values can make the solution extremely unstable. The SVD is applied to representative rank deficient tomography problems in Section 3.4 and to discrete ill–posed problems in Section 3.5.

The mathematical level of this chapter is somewhat higher than in Chapters 1 and 2. We begin to make extensive use of operations from linear algebra (reviewed in Appendix A). In particular, we make extensive use of the properties of orthogonal matrices and other aspects of algebraic matrix operations. In lecturing on this material, we have found that it is sometimes necessary to go through the derivations step by step and/or illustrate them with MATLAB. We typically spend five lectures on the material in this chapter.

Exercise 3.1 is a theoretical exercise that develops some of the properties of the Moore–Penrose pseudoinverse. Exercise 3.2 is a computational exercise in which students build upon one of the examples in the chapter and consider the resolution of the solution that was obtained. Exercise 3.3 is an extended case study in which students develop data for over determining, exactly determining and under determining the same system. Exercise 3.4 is an extended case study of a rank deficient problem. This exercise can be very time consuming, but students gain a lot from solving such a problem from start to finish. Exercise

3.5 is a computational exercise in which the methods of Chapter 3 are applied to problems that have previously been examined in earlier chapters.

3.2 Exercises

1. The pseudoinverse of a matrix $\mathbf{G}$ was originally defined by Moore and Penrose as the unique matrix $\mathbf{G}^\dagger$ with the properties

   (a) $\mathbf{G}\mathbf{G}^\dagger\mathbf{G} = \mathbf{G}$.

   (b) $\mathbf{G}^\dagger\mathbf{G}\mathbf{G}^\dagger = \mathbf{G}^\dagger$.

   (c) $(\mathbf{G}\mathbf{G}^\dagger)^T = \mathbf{G}\mathbf{G}^\dagger$.

   (d) $(\mathbf{G}^\dagger\mathbf{G})^T = \mathbf{G}^\dagger\mathbf{G}$.

   Show that $\mathbf{G}^\dagger$ as given by (3.20) satisfies these four properties.

2. Another resolution test commonly performed in tomography studies is a **checkerboard test**, which consists of using a test model composed of alternating positive and negative perturbations. Perform a checkerboard test on the tomography problem in Example 3.1 using the test model

$$\mathbf{m}_{\text{true}} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} . \qquad (3.1)$$

   Evaluate the difference between the true (checkerboard) model and the recovered model in your test, and interpret the pattern of differences. Are any block values recovered exactly? If so, does this imply perfect resolution for these model parameters?

3. Using the parameter estimation problem described in Example 1.1 for determining the three parameters defining a ballistic trajectory, construct synthetic examples that demonstrate the following four cases using the SVD. In each case, display and interpret the SVD components $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{S}$ in terms of the rank, p, of your forward problem $\mathbf{G}$ matrix. Display and interpret any model and data null space vector(s) and calculate model and data space resolution matrices.

   (a) Three data points that are exactly fit by a unique model. Plot your data points and the predicted data for your model.

   (b) Two data points that are exactly fit by an infinite suite of parabolas. Plot your data points and the predicted data for a suite of these models.

   (c) Four data points that are only approximately fit by a parabola. Plot your data points and the predicted data for the least squares model.

(d) Two data points that are only approximately fit by any parabola, and for which there are an infinite number of least squares solutions. Plot your data points and the predicted data for a suite of least squares models.

4. A large north-south by east-west oriented, nearly square plan view, sandstone quarry block (16 m by 16 m) with a bulk compressional wave seismic velocity of approximately 3000 m/s is suspected of harboring higher-velocity dinosaur remains. An ultrasonic tomography scan is performed in a horizontal plane bisecting the boulder, producing a data set consisting of 16 E$\rightarrow$W, 16 N$\rightarrow$S, 31 NE$\rightarrow$SW, and 31 NW$\rightarrow$SE travel times. See Figure 3.1. The travel time data (units of s) have statistically independent errors and the travel time contribution for a uniform background model (with a velocity of 3000 m/s) has been subtracted from each travel time measurement.

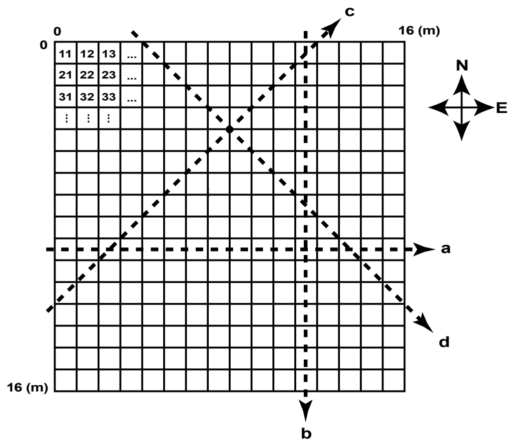

Figure 3.1: Tomography exercise, showing block discretization, block numbering convention, and representative ray paths going east-west (a), north-south (b), southwest-northeast (c), and northwest-southeast (d).

The MATLAB travel time data files that you will need to load are: **rowscan.mat**, **colscan.mat**, **diag1scan.mat**, and **diag2scan.mat**. The standard deviations of all data measurements are 1.5×10^{-5} s. Because the travel time contributions for a uniform background model (with a velocity of 3000 m/s) have been subtracted from each travel time measurement, you will be solving for slowness and velocity perturbations relative to a uniform slowness model of 1/3000 s/m. Use a row–by–row mapping between the slowness grid and the model vector (e.g., Example 1.12). The row format of each data file is $(x_1,\ y_1,\ x_2,\ y_2,\ t)$ where the starting point coordinate of each source is $(x_1,\ y_1)$, the end point coordinate is $(x_2,\ y_2)$, and the travel time along a ray path between the source and receiver points is a path integral (in seconds).

Parameterize the slowness structure in the plane of the survey by dividing the boulder into a 16 by 16 grid of 256 1-m-square, N by E blocks and construct a linear system for the forward problem (Figure 3.3.1). Assume that the ray paths through each homogeneous block can be represented by straight lines, so that the travel time expression is

$$t = \int_{\ell} s(\mathbf{x})\, d\ell \tag{3.2}$$

$$= \sum_{\text{blocks}} s_{\text{block}} \cdot \Delta l_{\text{block}} \tag{3.3}$$

where Δl_{block} is 1 m for the row and column scans and $\sqrt{2}$ m for the diagonal scans.

Use the SVD to find a minimum-length/least squares solution, $\mathbf{m}_{\dagger}$, for the 256 block slowness perturbations that fit the data as exactly as possible. Perform two inversions in this manner:

(A) Using the row and column scans only, and

(B) Using the complete data set.

For each inversion:

(a) Note the rank of your **G** matrix relating the data and model.

(b) State and discuss the general solution and/or data fit significance of the elements and dimensions of the data and model null spaces. Plot and interpret an element of each space and contour or otherwise display a nonzero model that fits the trivial data set $\mathbf{Gm} = \mathbf{d} = \mathbf{0}$ exactly.

(c) Note whether there are any model parameters that have perfect resolution.

(d) Produce a 16 by 16 element contour or other plot of your slowness perturbation model, displaying the maximum and minimum slowness perturbations in the title of each plot. Interpret any internal structures geometrically and in terms of seismic velocity (in m/s).

(e) Show the model resolution by contouring or otherwise displaying the 256 diagonal elements of the model resolution matrix, reshaped into an appropriate 16 by 16 grid.

(f) Describe how one could use solutions to $\mathbf{Gm} = \mathbf{d} = \mathbf{0}$ to demonstrate that very rough models exist that will fit any data set just as well as a generalized inverse model. Show one such wild model.

5. Consider the data in Table 3.1 (also found in the file **ifk.mat**).

y	0.0250	0.0750	0.1250	0.1750	0.2250
$d(y)$	0.2388	0.2319	0.2252	0.2188	0.2126
y	0.2750	0.3250	0.3750	0.4250	0.4750
$d(y)$	0.2066	0.2008	0.1952	0.1898	0.1846
y	0.5250	0.5750	0.6250	0.6750	0.7250
$d(y)$	0.1795	0.1746	0.1699	0.1654	0.1610
y	0.7750	0.8250	0.8750	0.9250	0.9750
$d(y)$	0.1567	0.1526	0.1486	0.1447	0.1410

Table 3.1: Data for Exercise 3.5.

The function $d(y)$, $0 \leq y \leq 1$, is related to an unknown function $m(x)$, $0 \leq x \leq 1$, by the mathematical model

$$d(y) = \int_0^1 x e^{-xy} m(x)\, dx \; . \tag{3.4}$$

(a) Using the data provided, discretize the integral equation using simple collocation to create a square $\mathbf{G}$ matrix and solve the resulting system of equations.

(b) What is the condition number for this system of equations? Given that the data $d(y)$ are only accurate to about 4 digits, what does this tell you about the accuracy of your solution?

(c) Use the TSVD to compute a solution to this problem. You may find a plot of the Picard ratios $\mathbf{U}_{\cdot,i}^T \mathbf{d}/s_i$ to be especially useful in deciding how many singular values to include.

3.3 Solutions

1. The key to this problem is utilizing that $\mathbf{U}_p^T \mathbf{U}_p = \mathbf{I}$, $\mathbf{V}_p^T \mathbf{V}_p = \mathbf{I}$, and the formulas for $\mathbf{G}$ and $\mathbf{G}^\dagger$ in terms of the compact form of the SVD.

$$\begin{aligned}
\mathbf{G}\mathbf{G}^\dagger\mathbf{G} &= \mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{U}_p\mathbf{S}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{U}_p\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{G}\; .
\end{aligned}$$

$$\begin{aligned}
\mathbf{G}^\dagger\mathbf{G}\mathbf{G}^\dagger &= \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{S}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{V}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{G}^\dagger \ .
\end{aligned}$$

$$\begin{aligned}
(\mathbf{G}\mathbf{G}^\dagger)^T &= (\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T)^T \\
&= (\mathbf{U}_p\mathbf{S}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T)^T \\
&= (\mathbf{U}_p\mathbf{U}_p^T)^T \\
&= \mathbf{U}_p\mathbf{U}_p^T \\
&= \mathbf{U}_p\mathbf{S}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T \\
&= \mathbf{G}\mathbf{G}^\dagger \ .
\end{aligned}$$

Note that we could have stopped where we had $\mathbf{U}_p\mathbf{U}_p^T$, because this expression is obviously symmetric.

$$\begin{aligned}
(\mathbf{G}^\dagger\mathbf{G})^T &= (\mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T)^T \\
&= \mathbf{V}_p\mathbf{S}_p\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p^{-1}\mathbf{V}_p^T \\
&= \mathbf{V}_p\mathbf{S}_p\mathbf{S}_p^{-1}\mathbf{V}_p^T \\
&= \mathbf{V}_p\mathbf{V}_p^T \\
&= \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \\
&= \mathbf{G}^\dagger\mathbf{G} \ .
\end{aligned}$$

Note that we could have stopped where we had $\mathbf{V}_p\mathbf{V}_p^T$, because this expression is obviously symmetric.

2. We set up the problem using the **G** matrix of (3.93). The generalized inverse model is given by the product of the resolution matrix and the true model (3.60).

   For a checkerboard model,

   $$\mathbf{m}_{\text{check}} = [-1,\ 1,\ -1,\ 1,\ -1,\ 1,\ -1,\ 1,\ -1]^T \ ,$$

   the generalized inverse model is thus

   $$\begin{aligned}
   \mathbf{m}_\dagger &= \mathbf{R}_m\mathbf{m}_{\text{check}} \\
   &= \mathbf{G}^\dagger\mathbf{G}\mathbf{m}_{\text{check}} \\
   &\approx [-1.67,\ 1,\ -0.33,\ 1,\ -0.33,\ 0.33,\ -0.33,\ 0.33,\ -1]^T \ .
   \end{aligned}$$

The difference between the true and recovered models is

$$\mathbf{m}_{\text{check}} - \mathbf{m}_{\dagger} \approx [-0.67,\ 0,\ 0.67,\ 0,\ 0.67,\ -0.67,\ 0.67,\ -0.67,\ 0]^T$$

which is depicted in Figure 2. The pattern of differences between the

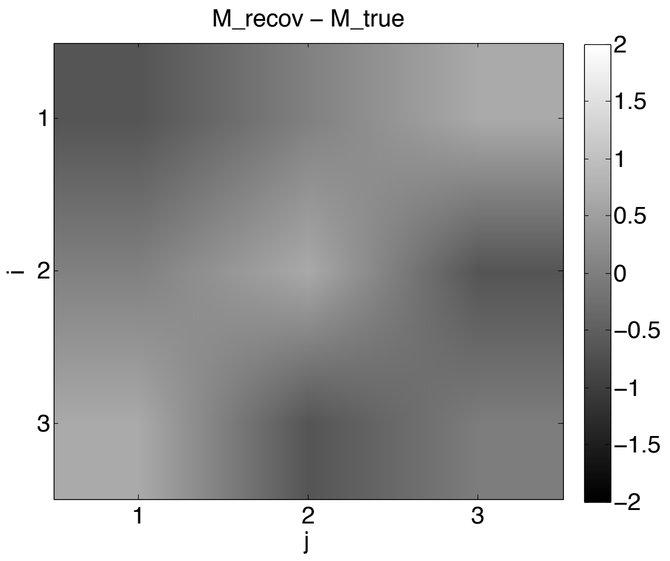

Figure 3.2: Difference between the generalized inverse solution and the true solution, checkerboard test.

generalized inverse model and the true model is insightful in the sense that it has zero sum along all of the ray paths in the problem. This must occur because the generalized inverse model fits the predicted data $\mathbf{d}_{\text{check}} = \mathbf{G}\mathbf{m}_{\text{check}}$ exactly. Because there is a nontrivial null space to this problem, however, the solution is nonunique. As shown in section 3.1, the generalized inverse solution is the smallest norm solution that fits the data in the least squares sense. Because the checkerboard model is not the smallest norm model that fits the data, it is not exactly recovered by the pseudoinverse matrix operating on the data. In fact, the checkerboard model is recovered exactly for only three of the model blocks in this case, and the slowness anomalies for the rest of the blocks are either over- or under-estimated.

In the case of a general model, only the (uniquely determined) $m_9 = s_{3,3}$ block will always be recovered exactly for noise-free data given this ray path geometry (you can show this by experimenting with random models, or by simply noting that the two null space models, $\mathbf{V}_{\cdot,8}$ and $\mathbf{V}_{\cdot,9}$ both have zero values for the element m_9). The correlation matrix (Figure 3.3) shows that the strongest (largest absolute value) correlations, 0.5755, arise between elements m_8 and m_6, and between elements m_3 and m_5. The weakest (smallest absolute value) correlations (-0.1039) occur between elements m_2 and m_3, and between elements m_3 and m_6. Correlations are measures of the tendencies of model elements to trade off against each other for a solution arising from a general data set that includes noise.

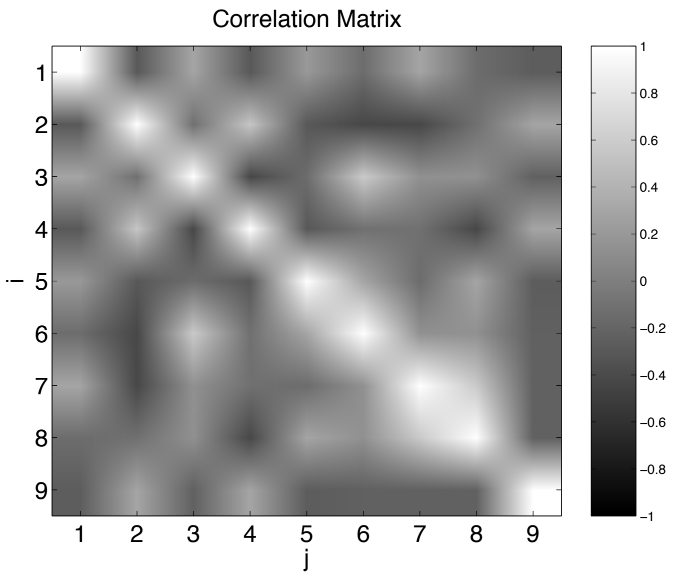

Figure 3.3: Correlation matrix for the generalized inverse solution.

```
%Exercise 3.2 Checkerboard Resolution Test
clear
t=sqrt(2);
%load the system matrix for the appropriate ray paths
G = [1,0,0,1,0,0,1,0,0;
     0,1,0,0,1,0,0,1,0;
     0,0,1,0,0,1,0,0,1;
     1,1,1,0,0,0,0,0,0;
     0,0,0,1,1,1,0,0,0;
     0,0,0,0,0,0,1,1,1;
     t,0,0,0,t,0,0,0,t;
     0,0,0,0,0,0,0,0,t];

[m,n]=size(G);

%get the svd of the system matrix
[U,s,V] = svd(G);

%checkerboard test model
mcheck=[-1 1 -1  1 -1 1 -1 1 -1 ]';
disp('Checkerboard Model')
mcheckmat=reshape(mcheck,3,3)

figure(1)
bookfonts;
colormap('gray')
imagesc(mcheckmat,[-2 2])
colorbar;
set(gca,'xtick',[1,2,3]);
set(gca,'ytick',[1,2,3]);
xlabel('j')
ylabel('i')
title ('Checkerboard Test Model')
```

```
% calculate the recovered model for the checkerboard
% data using the model resolution matrix
r=rank(G);
Vp=V(:,1:r);
Rm=Vp*Vp';
mrecov=Rm*mcheck;
mrecovmat=reshape(mrecov,3,3);

figure(2)
bookfonts;
colormap('gray')
imagesc(mrecovmat,[-2 2])
colorbar;
set(gca,'xtick',[1,2,3]);
set(gca,'ytick',[1,2,3]);
xlabel('j')
ylabel('i')
title ('Recovered Checkerboard Test Model')

figure(3)
bookfonts;
colormap('gray')
imagesc(Rm,[-2 2])
colorbar;
set(gca,'xtick',[1,2,3,4,5,6,7,8,9]);
set(gca,'ytick',[1,2,3,4,5,6,7,8,9]);
xlabel('j')
ylabel('i')
title ('Resolution Matrix')

Rmdiag=reshape(diag(Rm),3,3);

figure(4)
colormap('gray')
bookfonts;
imagesc(Rmdiag,[-2 2])
colorbar;
set(gca,'xtick',[1,2,3]);
set(gca,'ytick',[1,2,3]);
xlabel('j')
ylabel('i')
title ('Resolution Matrix Diagonal Elements')

% calculate the covariance matrix (unit standard
% deviation assumed for the generalized inverse
% solution)
C = zeros(n,n);
for i=1:r
C=C+V(:,i)*V(:,i)'/s(i,i);
end

%calculate the correlation matrix

for i=1:n
for j=1:n
Cor(i,j)=C(i,j)/sqrt(C(i,i)*C(j,j));
end
end
disp('Correlation Matrix:')
Cor

figure(5)
bookfonts;
colormap('gray')
imagesc(Cor,[-1 1])
colorbar;
set(gca,'xtick',[1,2,3,4,5,6,7,8,9]);
set(gca,'ytick',[1,2,3,4,5,6,7,8,9]);
xlabel('j')
ylabel('i')
title ('Correlation Matrix')

%Compare the results in a plot

disp('Difference between recovered and checkerboard model')
mdiffmat=mrecovmat-mcheckmat
figure(6)
colormap('gray')
bookfonts;
%imagesc(mdiffmat,[-2 2])
colorbar;
set(gca,'xtick',[1,2,3]);
set(gca,'ytick',[1,2,3]);
xlabel('j')
ylabel('i')
title ('M\_{recov} - M\_{true}')
```

3. This problem first involves setting up synthetic problems with varying

data or model null spaces. The number of parameters is $n = 3$, and the number of data points will be $m = 2$, 3, or 4. Numerical results will vary depending on how the synthetic examples are chosen.

For part (a), one can simply generate a forward $\mathbf{G}$ matrix for a three-point parameter fitting problem. Here we chose the true model to be $[m_1, m_2, m_3]^T = [10, 20, 9.8]^T$ and generated altitude data at times $t = 1$, 2, and 3. The SVD pseudoinverse solution for the parameters given the data and $\mathbf{G}$ matrix (Figure 3.4) is a single solution that fits the data points exactly.

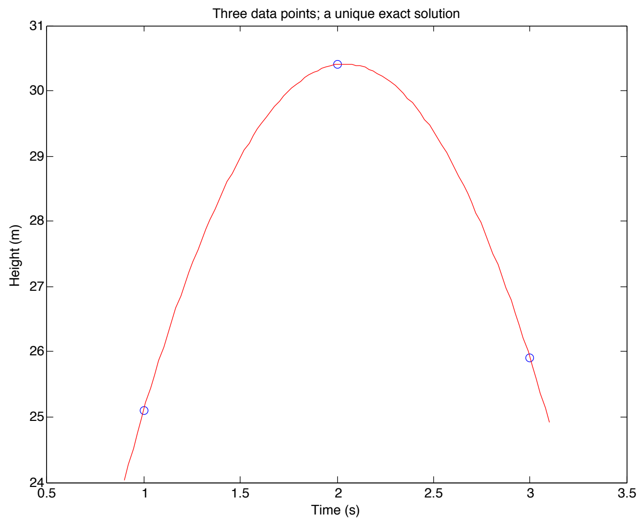

Figure 3.4: Parabola fit linear regression solution that exactly fits three data points. Both the model and data null spaces are trivial.

For part (b), the desired result can be obtained by generating a data set and $\mathbf{G}$ matrix to solve for the three parameters using only data points acquired at $t = 1$ and $t = 2$. The rank of the $\mathbf{G}$ matrix (the value of p for the SVD) is only 2, so there is a nontrivial null space vector $\mathbf{V}_{.,3}$ which is equal to $[-0.4851, 0.7276, 0.4851]^T$ for the sample problem as set up in part (a). Adding $\alpha\mathbf{V}_{.,3}$ to the pseudoinverse solution for any scalar α will not change the fit to the data when the model is used in the forward problem. A representative suite of models is plotted in Figure 3.5.

In part (c), one needs to generate four data points (here we used points at $t = 1$, 2, 2.1, 3) that do not fit any parabola. The pseudoinverse solution is a least squares solution that is unique (because the model null space is trivial). The forward problem can never exactly fit a data set with a nonzero projection onto the data null space vector $\mathbf{U}_{.,4} = [0.0319, -0.7026, 0.7097, -0.0390]$. The component of the data vector pro-

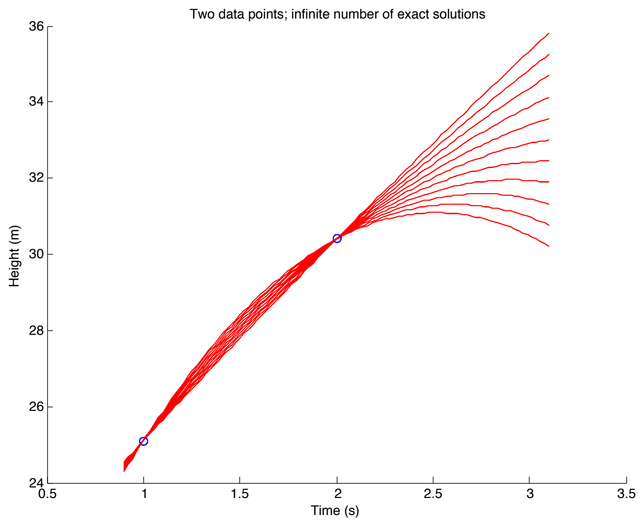

Figure 3.5: Parabola fit linear regression solutions that exactly fit two data points. The model null space is nontrivial, and a suite of equally well-fitting models is constructed by adding scaled null space vectors to the pseudoinverse solution.

jected onto $\alpha\mathbf{U}_{\cdot,4}$ cannot be fit by any parabola. The least squares solution is plotted in Figure 3.6.

In part (d), one needs to construct an example that is both impossible to fit with any parabola, yet has a nontrivial model null space, and is thus nonunique. For two data points, the solution is to fit parabolas to two distinct elevation measurements that occur at the same time. The SVD value of p is only one in this case (Figure 3.7).

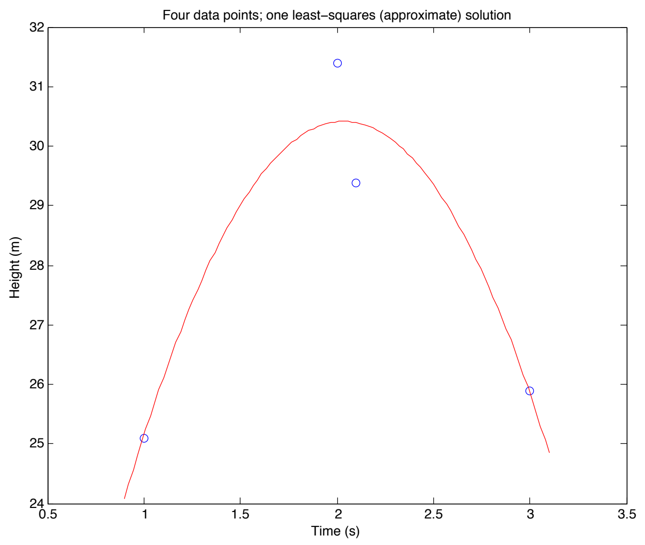

Figure 3.6: Parabola fit linear regression solution that best fits four data points in the least squares sense. The data null space is nontrivial, resulting in a nonzero residual.

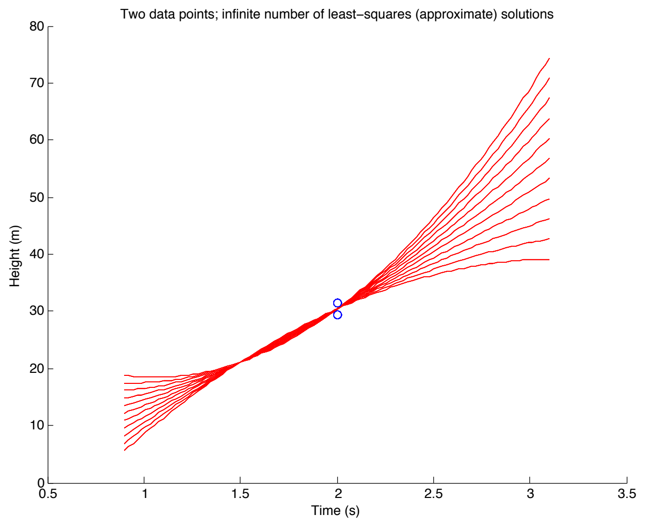

Figure 3.7: Parabola fit linear regression solution that is both least squares and nonunique.

```
%Exercise 3.3 exploration of SVD spaces and solutions using parabola fits
%to sets of points
%
clear

%true parameters (m1, m2, m3) in the form of Example 1.1
m_true=[10;20;9.8]

%t vector for subsequent plots
t_vec=linspace(0.9,3.1,100);

%forward problem matrix for full, consistent data set, three data points

%independent variable
t1=[1;2;3];
G1=[ones(size(t1)),t1,-0.5*t1.^2];

%noise free predicted data
y_true=G1*m_true;

[U1,S1,V1]=svd(G1);

%solution
m_recov1=(V1*inv(S1)*U1')*y_true

disp('Plotting case (a); three data points that are exactly fit by a unique parabola')
figure(1)
plot(t1,y_true,'o',t_vec,m_recov1(1)+m_recov1(2)*t_vec-0.5*m_recov1(3)*t_vec.^2,'r');
xlabel('Time (s)')
ylabel('Height (m)')
title('Three data points; a unique exact solution')


%nonunique case, two data points (nontrivial model null space)

%independent variable
t2=[1;2];
G2=[ones(size(t2)),t2,-0.5*t2.^2];
%noise free predicted data
y_true=G2*m_true;
[U2,S2,V2]=svd(G2);
%solution
p=rank(G2);
m_recov2=(V2(:,1:p)*inv(S2(1:p,1:p))*U2(:,1:p)')*y_true

disp('Plotting case (b); two data points that are exactly fit by an infinite number of parabolas')
figure(2)
hold on
%add in scaled null space vectors here to show nonuniqueness
for alpha=-5:5
    m=m_recov2+alpha*V2(:,3);
plot(t2,y_true,'o',t_vec,m(1)+m(2)*t_vec-0.5*m(3)*t_vec.^2,'r');
mnorm(alpha+6,1)=norm(m);
resid(alpha+6,1)=norm(G2*m-y_true);
end
hold off
xlabel('Time (s)')
ylabel('Height (m)')
title('Two data points; infinite number of exact solutions')
disp(['p = ',num2str(p),'; model null space vector:'])
null_m=V2(:,3)

%inconsistent case (nontrivial data null space)

%independent variable
t3=[1;2;2.1;3];
G3=[ones(size(t3)),t3,-0.5*t3.^2];
%noise free predicted data
y_true=G3*m_true;
y_true(2)=y_true(2)+1;
y_true(3)=y_true(3)-1;
[U3,S3,V3]=svd(G3);
%solution
p=rank(G3);
m_recov3=(V3(:,1:p)*inv(S3(1:p,1:p))*U3(:,1:p)')*y_true

disp('Plotting case (c); Four data points; least-squares (approximate) solution')

figure(3)
plot(t3,y_true,'o',t_vec,m_recov3(1)+m_recov3(2)*t_vec-0.5*m_recov3(3)*t_vec.^2,'r');
xlabel('Time (s)')
ylabel('Height (m)')
title('Four data points; one least-squares (approximate) solution')

disp(['p = ',num2str(p)])
disp('data null space vector:')
null_d=U3(:,4)

%both null spaces nontrivial

%independent variable
```

```
t4=[2;2];
G4=[ones(size(t4)),t4,-0.5*t4.^2];
%noise free predicted data
y_true=G4*m_true;
y_true(1)=y_true(1)+1;
y_true(2)=y_true(2)-1;
[U4,S4,V4]=svd(G4);
%solution
p=rank(G4);
m_recov4=(V4(:,1:p)*inv(S4(1:p,1:p))*U4(:,1:p)')*y_true

disp('Plotting case (d); Two data points; infinite number of least-squares (approximate) solution')
figure(4)
hold on
%add in scaled null space vectors here to produce a suite of equally good models and this
%show nonuniqueness of the least-squares solution.
for alpha=-5:5
    m=m_recov4+alpha*(10*V4(:,2)+V4(:,3));
plot(t4,y_true,'o',t_vec,m(1)+m(2)*t_vec-0.5*m(3)*t_vec.^2,'r');
mnorm(alpha+6,1)=norm(m);
resid(alpha+6,1)=norm(G2*m-y_true);
end
hold off
xlabel('Time (s)')
ylabel('Height (m)')
title('Two data points; infinite number of least-squares (approximate) solutions')

%display p and the model space null vectors

disp(['p = ',num2str(p)])
disp('data null space vector:')
null_d=U3(:,4)
disp('model null space vectors:')
null_m_1=V3(:,2)
null_m_2=V3(:,3)
```

4. This problem is a more complex and interesting version of the straight-line tomography Example 3.1. The initial challenge is to properly set up the **G** matrix. This is most easily done by recognizing the systematic patterns of mapping the column/row and diagonal ray paths into the 256 parameter model vector. Basic results are:

```
Part A: Inverting using only the row and column scan travel times (CR):
the rank of the G matrix is: 31
the dimension of the data space is: 32
the dimension of the P space is: 31
the dimension of the data null space is: 1
the dimension of the model null space is: 225
the rank of the generalized inverse matrix is: 31
the smallest and largest nonzero singular values are: 4 5.6569

Part B: Inverting using all available data (CRD):
the rank of the system matrix G is:87
the dimension of the data space is: 94
the dimension of the P space is: 87
the dimension of the data null space is: 7
the dimension of the model null space is: 169
the rank of the generalized inverse matrix is:87
smallest and largest nonzero singular values are: 1.3574 8.7055
```

For the row/column scan data set of 32 travel time measurements, the pseudoinverse solution is shown in Figure 3.8.

a) The rank of **G** is 31, and the 32 by 256 element **G** is thus rank deficient. The nonzero singular values range between approximately 4 and 5.66.

b) The data null space is 1-dimensional, so there is only one vector in **U**, $\mathbf{U}_{\cdot,32}$, that is not in the column space of the (32 by 256) **G** matrix. This data null space vector has a simple form (Figure 3.9) that reflects that, for this ray path geometry, no slowness model can jointly fit travel time data that describe uniform positive/negative slowness variation on the N-S axis coupled with uniform negative/positive slowness variation along the E-W axis. To fit such data, the slowness structure would need to be anisotropic (slowness within a block would have to depend on ray direction), which is a property that is not considered in the physics of the problem.

The model null space is large in the sense that 256 - 31 = 225 out of the 256 basis vectors that one would need to construct a general model reside in the null space. All models in this null space have N-S and E-W raypath slowness integrals that equal zero. c) The model resolution matrix diagonal is constant and has diagonal elements that are equal to about 0.12, implying that all model elements are imperfectly resolved. This uniformity reflects the geometric similarity between all of the ray path/block geometries; all blocks have just two ray paths going through them at right angles.

d) The row/column scan solution (Figure 3.8) shows that there is clearly some high velocity (low slowness) body located symmetrically near the interior of the slab. The central slowness anomaly is around -2.9×10^{-5} s/m, which, incorporating the background velocity of 3000 m/s, corresponds to a velocity of approximately $(1/3000 - 2.9 \times 10^{-5})^{-1} \approx 3286$ m/s. The limited resolution arising from the symmetry of the raypaths does not allow for distinguishing N-S or E-W reflected central structures, however.

e) An imagesc-generated plot of the diagonal of the model resolution matrix (Figure 3.10) simply shows that it is constant, with $R_{i,i} \approx 0.12$.

f) We can create a wild model simply by superimposing scaled basis vectors from the model null space onto our generalized inverse solution. We used 100 times the first basis vector from the model null space to add to (e) to generate a model (Figure 3.11) with outrageous slowness values that fits the data equally as well as the reasonable solution shown in Figure 3.8.

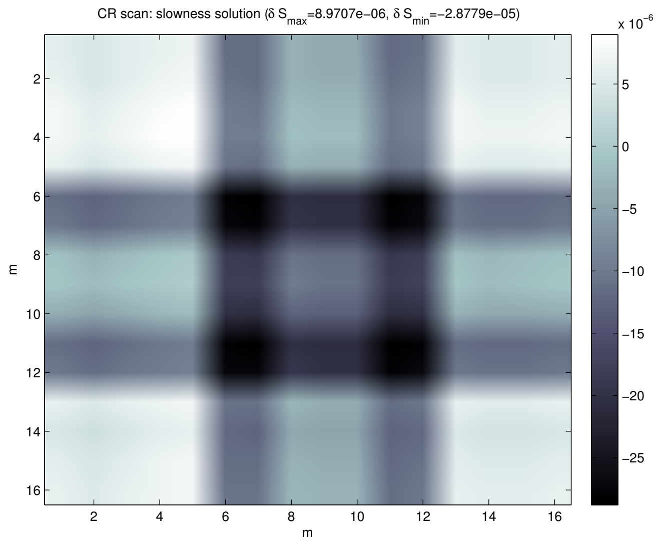

Figure 3.8: SVD solution for the row/column scan data.

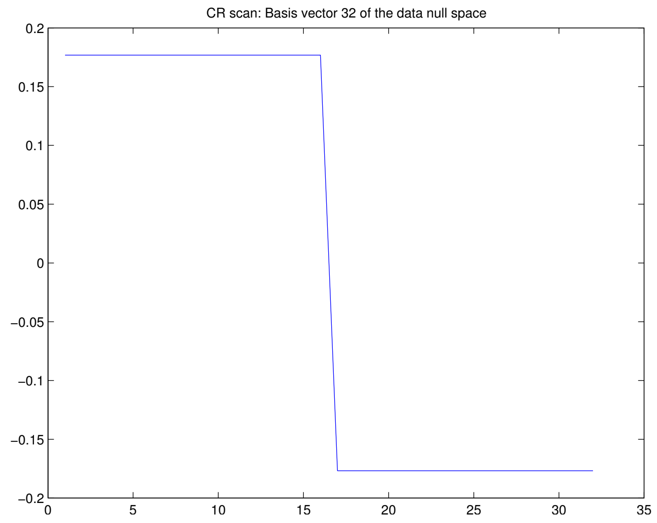

Figure 3.9: Data null space vector $\mathbf{U}_{\cdot,32}$.

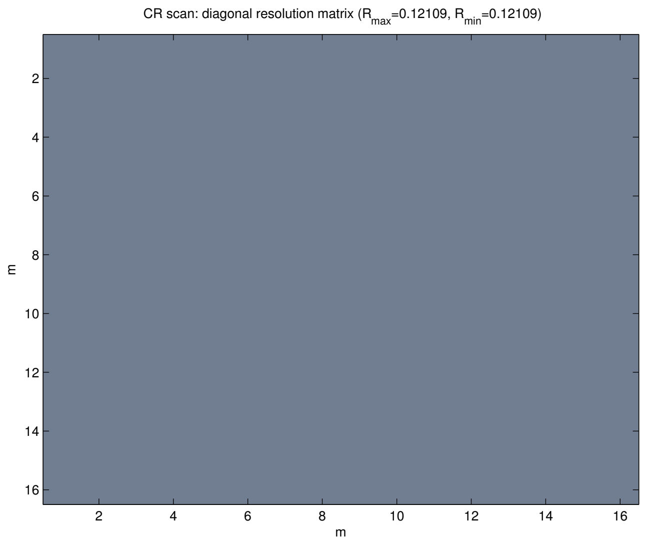

Figure 3.10: Model diagonal resolution matrix elements for the row/column scan data. All blocks have equal values for this ray path geometry.

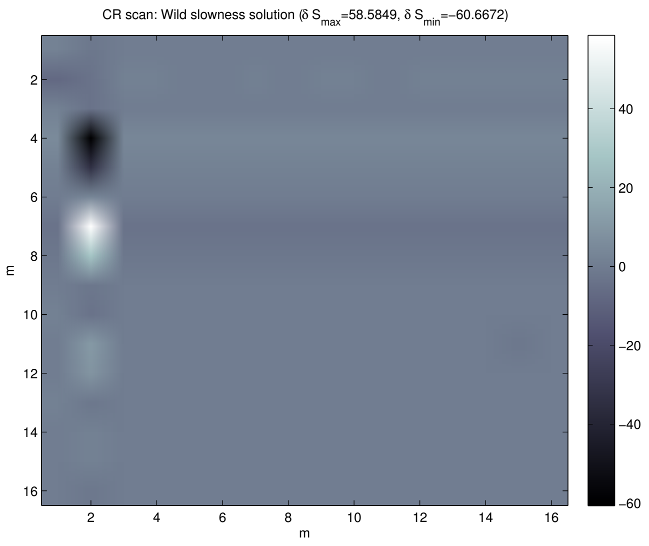

Figure 3.11: A wild model constructed by adding a scaled vector ($100\mathbf{V}_{.,32}$) from the model null space to the solution shown in Figure 3.8.

For the full row/column/diagonal scan data set of 94 travel time measurements, the pseudoinverse solution is shown in Figure 3.12.

a) The rank of $\mathbf{G}$ is 87, and the 94 by 256 element $\mathbf{G}$ is thus still rank deficient. The nonzero singular values range between about 1.36 and 8.70.

b) The data null space has dimension 7. There are thus 7 orthogonal vectors (e.g., Figure 3.13) that specify projections of the data that cannot be fit by any model. The addition of 62 additional ray paths relative to the row/column scan data set increases the number of ways in which perturbations to block slownesses can lead to inconsistent equations in $\mathbf{Gm} = \mathbf{d}$.

The dimension of the model null space is 256 - 87 = 169. All models in this null space have N-S and E-W raypath slowness integrals that equal zero.

c) The model resolution matrix diagonal is now spatially variable due to the more complex set of ray paths. An imagesc plot of the diagonal elements show that the resolution is perfect (one) for the corner blocks but is imperfect (less than one) everywhere else, reaching a minimum of approximately 0.26 near the center of the image.

d) The solution (Figure 3.12) shows that there is a diagonally-oriented high velocity body near the interior of the slab with a slowness perturbation of around -9.8×10^{-5} s/m, or a velocity estimate of approximately $(1/3000-9.8\times10^{-5})^{-1} \approx 4250$ m/s.

e) An imagesc plot of the diagonal of the model resolution matrix is shown, note that this is what we might intuitively expect for our ability to sort out information in the interior of the slab, given the raypath geometry.

f) Following the same procedure as before, adding 100 times the first element in the model null space to the pseudoinverse solution (Figure 3.12) creates a wild model with totally non-physical slowness perturbation values.

The true slowness perturbation model used to generate the data sets is shown in Figure 3.16. The true bone velocity was set to 5.0 km/s, which corresponds to a slowness perturbation with respect to a 3.0 km/s background velocity of 1.3×10^{-4}. Note that both of our models gave peak slowness perturbation decreases that were less than this. This underestimation is characteristic of limited resolution problems. Note that the smearing was more dramatic when we used only the row and column scan data, in which case we could not even discriminate between models reflected through the diagonals. As a result, the pseudoinverse solution “splits the difference” to give a smeared superposition of both possibilities for the 45-degree canted bone (and a lesser estimate of the degree of central slowness decrease relative to the complete data set inversion). The resolution is much improved for the complete data set which includes the diagonal scans, although the true model is clearly still smeared out, and the maximum slowness perturbation is under-recovered.

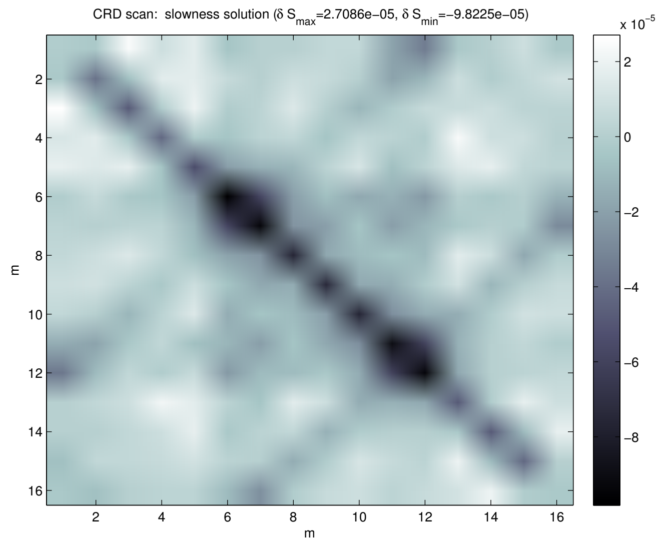

Figure 3.12: SVD solution for the row/column/diagonal scan data.

```
%dinosaur tomography problem solved as a 16x16 grid of blocks
```

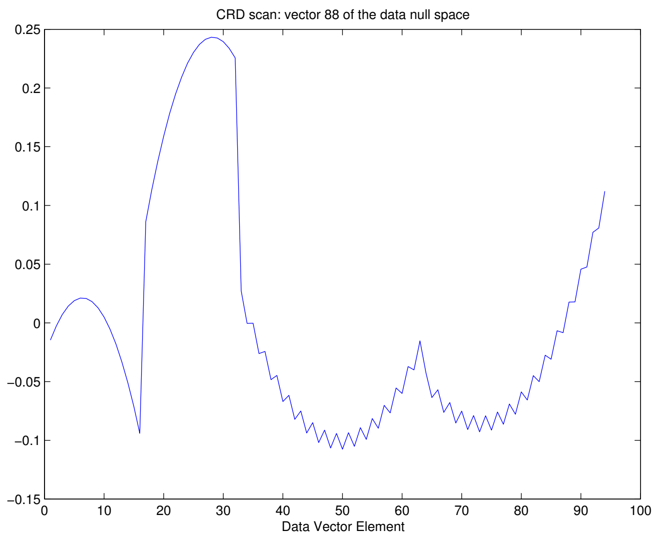

Figure 3.13: Representative data null space vector $\mathbf{U}_{.,88}$.

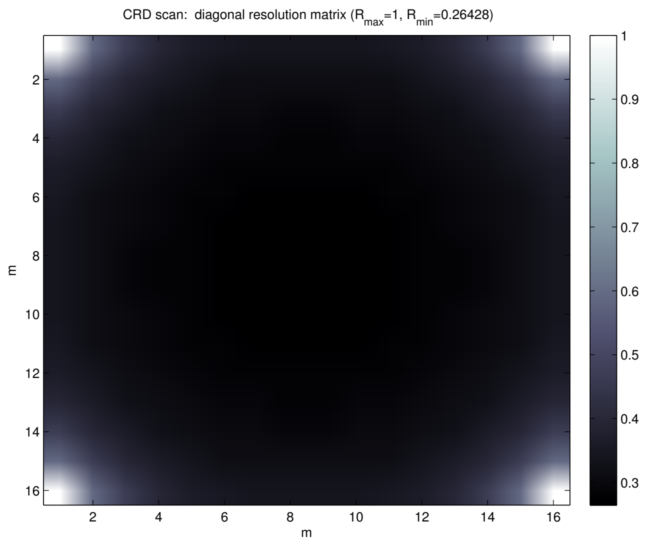

Figure 3.14: Model diagonal resolution matrix elements for the row/column/diagonal scan data.

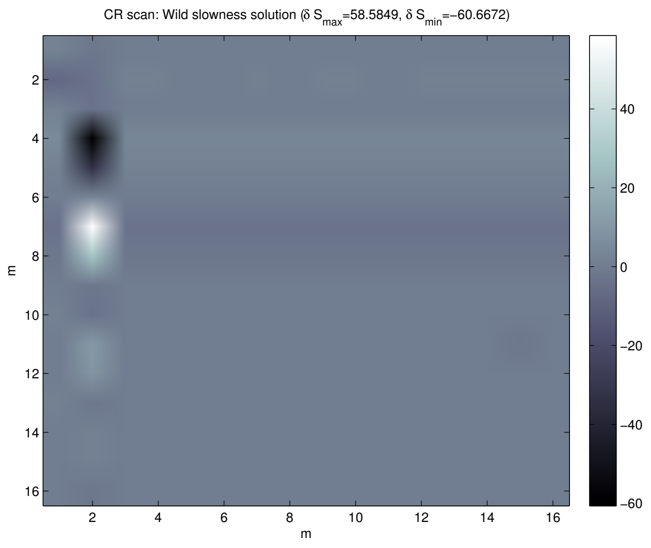

Figure 3.15: A wild model constructed by adding a scaled vector ($100\mathbf{V}_{\cdot,88}$) from the model null space to the solution shown in Figure 3.8).

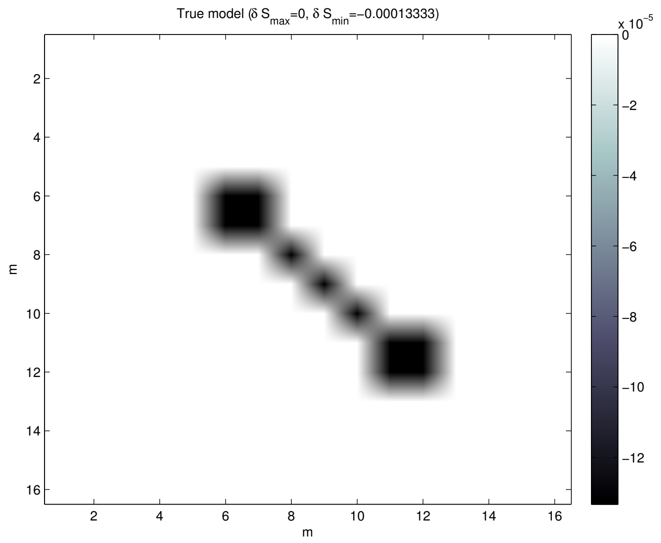

Figure 3.16: The true model.

```
%initialize the workspace
clear

%initialize the graphics
clf
axis('square')

%load the data files
load diag1scan.mat
load diag2scan.mat
load colscan.mat
load rowscan.mat

%stated standard deviation of travel time estimates
%not used in this problem
load std.mat;

%extract the travel time measurements
dinp(1:16) = colscan(1:16,5);
dinp(17:32) = rowscan(1:16,5);
d=dinp';


%initialize the design matrix
G = zeros(32,256);

%number of model parameters in the problem
m = 256;

%parameter indices increase going down each column
%m(1) is in the NW corner, m(16) is in the SW corner
%m(241) in the NE corner, and m(256) is in the SE corner

%design matrix entries for the column scan
%ones where each ray hits a block, zeros otherwise
for i=1:16
for j=(i-1)*16+1:i*16
G(i,j) = 1.;
end
end

%design matrix for the row scan
%ones where each ray hits a block, zeros otherwise
for i=1:16
for j=i:16:240+i
G(i+16,j) = 1.;
end
end

%inverting using the row and column scans only
disp('Part A: Inverting using only the row and column scan travel times (CR):')

%get the rank of G
disp(['the rank of the G matrix is: ',num2str(rank(G))]);

%evaluate the svd of G
[u,lam,v] = svd(G);

%partition the P and null spaces from u, lam and v
p = rank(lam);


%number of data values in the problem
n = length(d);

disp(['the dimension of the data space is: ',num2str(n)])
disp(['the dimension of the P space is: ',num2str(p)])
disp(['the dimension of the data null space is: ',num2str(n-p)])
disp(['the dimension of the model null space is: ',num2str(m-p)])

lamp = lam(1:p,1:p);
up = u(:,1:p);
vp = v(:,1:p);

%evaluate the generalized inverse
Ginv = vp*inv(lamp)*up';
disp(['the rank of the generalized inverse matrix is: ',...
num2str(rank(Ginv))]);
disp(['the smallest and largest nonzero singular values are: ',...
num2str(min(diag(lamp))),' ',num2str(max(diag(lamp)))])

%evaluate the generalized inverse solution
mg = Ginv*d;


sol = reshape(mg,16,16);
%convert the slownesses in the solution to a grid of slowness

%get the maximim and minimum slowness for the solution
maxs = max(max(sol));
```

```
mins = min(min(sol));

figure(1)
imagesc(sol);
colorbar;
colormap('bone');
title(['CR scan: slowness solution (\delta S_{max}=',num2str(maxs),...
', \delta S_{min}=',num2str(mins),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig1rc.eps
pause

%get the resolution matrix (rounded)
res = round(reshape(diag(vp*vp'),16,16)*100000)/100000;
%get the maximim and minimum values for the resolution
maxr = max(max(res));
minr = min(min(res));
figure(2)
imagesc(res);
colormap('bone');
title(['CR scan: diagonal resolution matrix (R_{max}=',num2str(maxr),...
', R_{min}=',num2str(minr),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig2rc.eps
pause

%evaluate the model standard deviations (rounded)

%show an element of the model null space
figure(3)
imagesc(reshape(v(:,p+1),16,16));
colorbar;
colormap('bone')
title(['CR scan: An element of the row and column scan model null space']);
xlabel('m'),ylabel('m');
print -depsc2 fig3rc.eps
pause

%show an exampl3 wild model
figure(4)
wsol=sol+100*reshape(v(:,p+1),16,16);
maxs = max(max(wsol));
mins = min(min(wsol));
imagesc(wsol);
colorbar;
colormap('bone');
title(['CR scan: Wild slowness solution (\delta S_{max}=',num2str(maxs),...
', \delta S_{min}=',num2str(mins),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig4rc.eps
pause

%show the first 10 elements of the model (solution) space
for i=1:10
figure(5)
imagesc(reshape(v(:,i),16,16))
colorbar;
colormap('bone');
title(['CR scan: Basis vector ',num2str(i),' of the model space'])
pause
end

%show some basis vectors of the model null space
for i=p+1:p+11
figure(6)
imagesc(reshape(v(:,i),16,16))
colorbar;
colormap('bone');
title(['CR scan: Basis vector ',num2str(i),' of the model null space'])
pause
end

%show the basis vector of the data null space
figure(7)
plot(reshape(u(:,32),1,32))
title(['CR scan: Basis vector 32 of the data null space'])
print -depsc2 fig7rc.eps
pause

%part B...
%inverting using the entire data set
%including the diagonal scan data
dinp(33:63) = diag1scan(1:31,5);
dinp(64:94) = diag2scan(1:31,5);

d=dinp';

%enlarge the G design matrix to accommodate the diagonal scans

%G matrix for the SW to NE diagonal scan, upper part
%because we're now going through blocks on the diagonal, we
```

```
%have entries of sqrt(2) where we intersect a block
for i=1:16
for j=0:i-1
G(i+32,i+j*15) = sqrt(2.);
end
end

%g matrix for the SW to NE diagonal scan, lower part
for i=1:15
for j=0:15-i
G(i+48,(i+1)*16+j*15) = sqrt(2.);
end
end

%g matrix for the NW to SE diagonal scan, lower part
for i=1:16
for j=0:i-1
G(i+63,17-i+17*j) = sqrt(2.);
end
end

%g matrix for the NW to SE diagonal scan, upper part
for i=1:15
for j=0:15-i
G(i+79,(i*16)+1+17*j) = sqrt(2.);
end
end

disp('Part B: Inverting using all available data (CRD):')

%get the rank of G
disp(['the rank of the system matrix G is:',num2str(rank(G))]);

%svd again
[u,lam,v] = svd(G);
p = rank(lam);

n = length(d);

disp(['the dimension of the data space is: ',num2str(n)])
disp(['the dimension of the P space is: ',num2str(p)])
disp(['the dimension of the data null space is: ',num2str(n-p)])
disp(['the dimension of the model null space is: ',num2str(m-p)])

lamp = lam(1:p,1:p);
up = u(:,1:p);
vp = v(:,1:p);

Ginv = vp*inv(lamp)*up';
disp(['the rank of the generalized inverse matrix is:',...
num2str(rank(Ginv))]);
disp(['smallest and largest nonzero singular values are: ',...
num2str(min(diag(lamp))),' ',num2str(max(diag(lamp)))])
mg = Ginv*d;

rescrd = reshape(diag(vp*vp'),16,16);
sol = reshape(mg,16,16);
maxs = max(max(sol));
mins = min(min(sol));

%solution
figure(8)
imagesc(sol);
colorbar;
colormap('bone');
title(['CRD scan:  slowness solution (\delta S_{max}=',num2str(maxs),', \delta S_{min}=',...
num2str(mins),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig8rcd.eps
pause
maxr = max(max(rescrd));
minr = min(min(rescrd));

%resolution
figure(9)
imagesc(rescrd);
colorbar;
colormap('bone');
title(['CRD scan:  diagonal resolution matrix (R_{max}=',num2str(maxr),...
', R_{min}=',num2str(minr),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig9rcd.eps
pause;

figure(10)
%show an element of the row, column, diag scan model null space
imagesc(reshape(v(:,p+1),16,16));
xlabel('m'),ylabel('m');
colorbar;
colormap('bone')
title('An element of the row, column, diag scan model null space');
```

```
print -depsc2 fig10.eps
pause;

figure(11)
%wild model
wsol=sol+100*reshape(v(:,p+1),16,16);
maxs = max(max(wsol));
mins = min(min(wsol));
imagesc(wsol);
colorbar;
colormap('bone');
title(['CRD Scan: wild slowness solution (\delta S_{max}=',num2str(maxs),', \delta S_{min}=',...
num2str(mins),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig11rcd.eps
pause

%plot the true model for comparison purposes
load tomo.mat
maxs = max(max(tomo));
mins = min ( min ( tomo ) );

figure(12)
imagesc(tomo);
colorbar;
colormap('bone');
title(['True model (\delta S_{max}=',num2str(maxs),', \delta S_{min}=',num2str(mins),')']);
xlabel('m'),ylabel('m');
print -depsc2 fig12true.eps
pause

%show the first 10 elements of the model (solution) space
for i=1:10
figure(13)
imagesc(reshape(v(:,i),16,16))
xlabel('m'),ylabel('m');
colorbar;
colormap('bone');
title(['CRD scan:  basis vector ',num2str(i),' of the model space'])
pause
end

%show some basis vectors of the model null space
for i=p+1:p+10
figure(14)
imagesc(reshape(v(:,i),16,16))
xlabel('m'),ylabel('m');
colorbar;
colormap('bone');
title(['CRD scan: vector ',num2str(i),' of the model null space'])
pause
end

figure(15)
%show some basis vectors of the data null space (save a plot of the first one)
for i=p+1:94
plot(reshape(u(:,i),94,1))
xlabel('Data Vector Element')
title(['CRD scan: vector ',num2str(i),' of the data null space'])
if i==p+1
    print -depsc2 fig15rcd.eps
end
pause
end
```

5. Discretizing the integral equation using simple collocation for $x_i = y_i = 0.025, 0.075, \ldots, 0.975$, we create a 20 by 20 $\mathbf{G}$ matrix with elements

$$G_{i,j} = 0.05 x_j e^{-x_j y_i} \ .$$

Taking the SVD of $\mathbf{G}$, and plotting the singular values (Figure 3.17), we see that the s_i are effectively zero for $i \geq 9$. A plot of the absolute values of the Picard ratios, $\mathbf{U}_{\cdot,i}^T \mathbf{d}/s_i$ (Figure 3.18) shows that the coefficients in the series SVD solution increase abruptly at $i = 5$ (to $\approx 10^2$) and reach $\approx 10^4$ by i=6.

Given that the data are stated to be accurate to 4 digits, we will conservatively only include the first four singular values in the TSVD solution. Figure 3.19 shows the TSVD solution obtained using four singular values, and Figure 3.20 shows the actual model used to generate these slightly noisy data. Including only one additional singular value results in a model that bears essentially no resemblance to the true model and has an amplitude that is approximately 20 times too large.

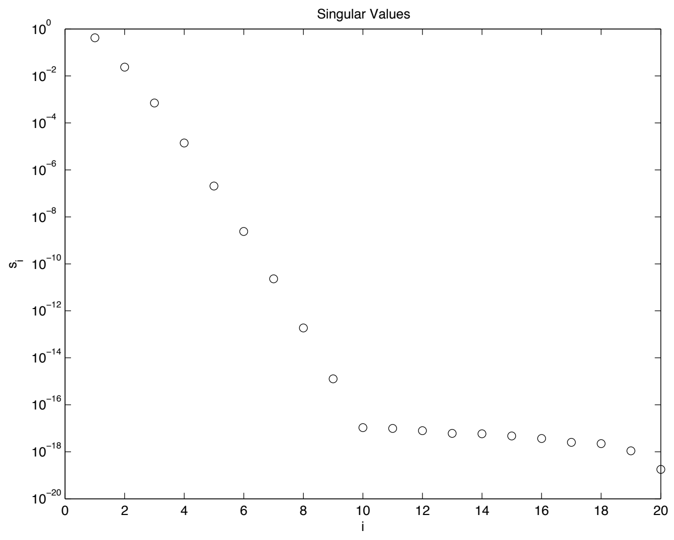

Figure 3.17: Singular Values.

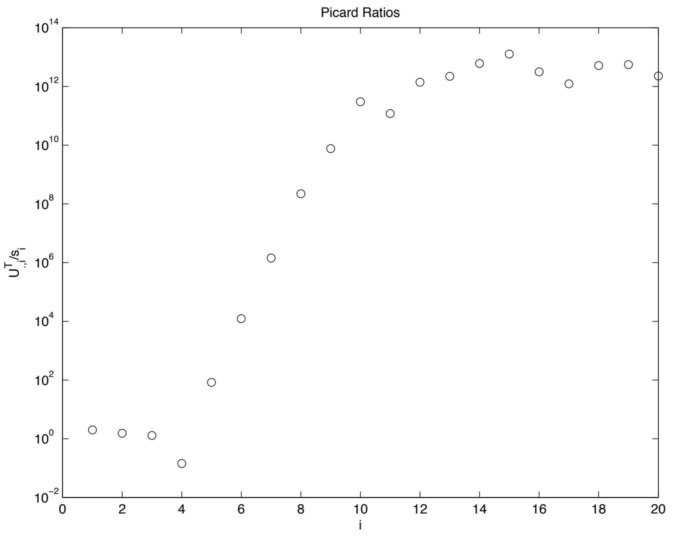

Figure 3.18: Absolute values of the Picard ratios.

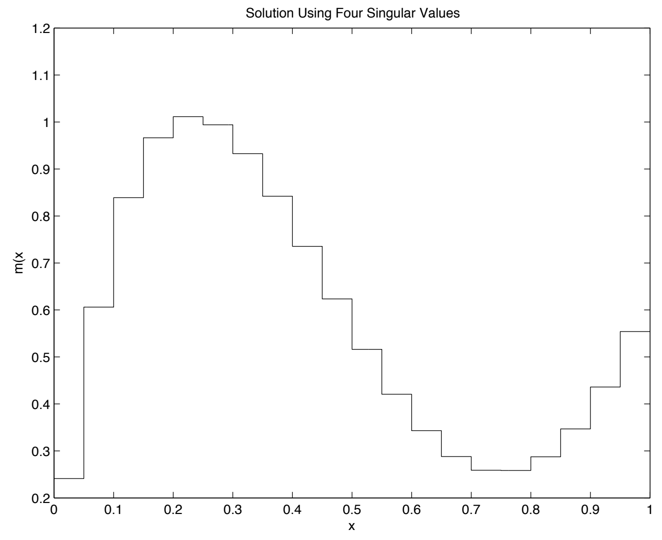

Figure 3.19: TSVD solution obtained using four singular values in the TSVD solution.

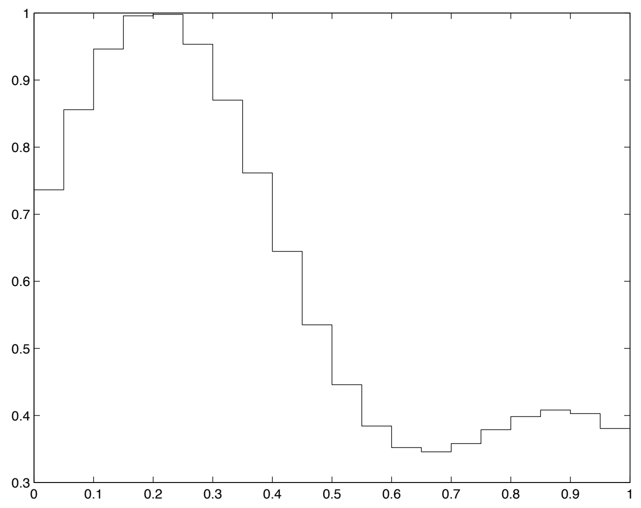

Figure 3.20: True model.

```
%Exercise 3.5; TSVD problem
%
% Load the data in.
%
clear
load ifk.mat
%
%Setup vectors of the x and y collocation points.
%
x=(0.025:0.05:0.975)';
y=x;
%
%Set up the G matrix.
%
for i=1:20, for j=1:20, G(i,j)=0.05*x(j)*exp(-x(j)*y(i)); end; end;
%
%Compute the SVD.
%
[U,S,V]=svd(G);
%
% Examine the Singular Values
%
s=diag(S);
figure(1)
semilogy(s,'o')
title('Singular Values')
xlabel('i')
ylabel('s_i')
print -deps fig1sing.eps

%
% Examine the Picard Ratios
%
for i=1:20
    pr(i)=U(:,i)'*d/s(i);
end

figure(2)
semilogy(abs(pr),'o')
title('Picard Ratios')
xlabel('i')
ylabel('U^T_{.,i}/s_i')
print -deps fig2pic.eps

%
%Use just the first four singular values in the TSVD solution.
%
p=4;
mtsvd=zeros(20,1);
for i=1:p
```

```
  mtsvd=mtsvd+((U(:,i)'*d)/s(i))*V(:,i);
end
%
%Plot the solution.
%
figure(3);
plotconst(mtsvd,0,1);
title('Solution Using Four Singular Values')
xlabel('x')
ylabel('m(x')'
print -deps fig3sol.eps
%
%Plot the true model.
%
figure(4)
mtrue=exp(-10*(x-0.2).^2)+0.4*exp(-10*(x-0.9).^2)
plotconst(mtrue,0,1);
title('True Solution')
xlabel('x')
ylabel('m(x')'
print -deps fig4mtrue.eps
```

Chapter 4

Tikhonov Regularization

4.1 Discussion of the Chapter

In this chapter we expand the analysis of discrete ill-posed problems. The technique of Tikhonov regularization is introduced and Tikhonov solutions are computed using the SVD. Higher order Tikhonov regularization is introduced in sections 4.4 and 4.5. The generalized singular value decomposition (GSVD) is introduced in section 4.6. In teaching from the first edition we found that many students focused on the GSVD (which is but one way of solving the optimization problems that arise from Tikhonov regularization with a roughening operator L) and missed the very important point that even in higher order regularization we're still solving least squares problems. This point becomes more clear in chapter 6 when we discuss iterative methods for computing regularized solutions. Section 4.7 introduced generalized cross validation, an underutilized method for determining the regularization parameter. In section 4.8 we present some bounds on the difference between the regularized solution and $\mathbf{m}_{true}$. These results are quite technical, but the important point here is that without making assumptions (the so called "source conditions) about the true model, it is impossible to bound the error.

The mathematical level of this chapter is generally similar to Chapter 3, although sections 4.7 and 4.8 are at a distinctly higher level. This is a rather long chapter, but Sections 4.6 through 4.8 could be made optional. We typically spend about 7 lectures on this chapter and cover all of the sections.

In Exercise 4.1 students are asked to show an important theoretical connection between two equivalent formulations of Tikhonov regularization. Exercises 4.2 and 4.3 are computational case studies. In Exercise 4.4 Tikhonov regularization is applied to effect functional interpolation. Exercise 4.5 extends Tikhonov regularization to operate on departures from a preferred or reference model $\mathbf{m}_0$.

4.2 Exercises

1. Use the method of Lagrange Multipliers (Appendix C) to derive the damped least squares problem (4.4) from the discrepancy principle problem (4.2), and demonstrate that (4.4) can be written as (4.5).

2. Consider the integral equation and data set from Problem 3.5. You can find a copy of this data set in the file **ifk.mat**.

   (a) Discretize the problem using simple collocation.

   (b) Using the data supplied, and assuming that the numbers are accurate to four significant figures, determine a reasonable bound δ for the misfit.

   (c) Use zeroth-order Tikhonov regularization to solve the problem. Use GCV, the discrepancy principle and the L-curve criterion to pick the regularization parameter.

   (d) Use first-order Tikhonov regularization to solve the problem. Use GCV, the discrepancy principle and the L-curve criterion to pick the regularization parameter.

   (e) Use second-order Tikhonov regularization to solve the problem. Use GCV, the discrepancy principle and the L-curve criterion to pick the regularization parameter.

   (f) Analyze the resolution of your solutions. Are the features you see in your inverse solutions unambiguously real? Interpret your results. Describe the size and location of any significant features in the solution.

3. Consider the following problem in **cross-well tomography**. Two vertical wells are located 1600 meters apart. A seismic source is inserted in one well at depths of 50, 150, ..., 1550 m. A string of receivers is inserted in the other well at depths of 50 m, 150 m, ..., 1550 m. See Figure 4.1. For each source-receiver pair, a travel time is recorded, with a measurement standard deviation of 0.5 ms. There are 256 ray paths and 256 corresponding data points. We wish to determine the velocity structure in the two-dimensional plane between the two wells.

   Discretizing the problem into a 16 by 16 grid of 100 meter by 100 meter blocks gives 256 model parameters. The **G** matrix and noisy data, **d**, for this problem (assuming straight ray paths) are in the file **crosswell.mat**. The order of parameter indexing from the slowness grid to the model vector is row-by-row (e.g., Example 1.12).

   (a) Use the TSVD to solve this inverse problem using an L-curve. Plot the result.

   (b) Use zeroth-order Tikhonov regularization to solve this problem and plot your solution. Explain why it is hard to use the discrepancy

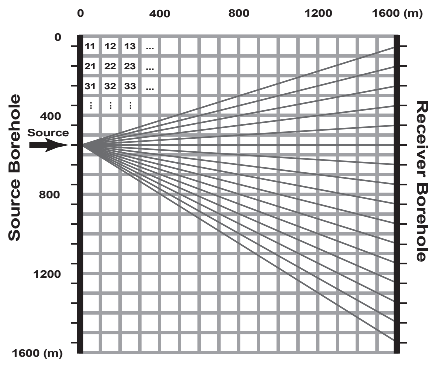

Figure 4.1: Cross-well tomography problem, showing block discretization, block numbering convention, and one set of straight source-receiver ray paths.

principle to select the regularization parameter. Use the L-curve criterion to select your regularization parameter. Plot the L-curve as well as your solution.

(c) Use second-order Tikhonov regularization to solve this problem and plot your solution. Because this is a two-dimensional problem, you will need to implement a finite-difference approximation to the Laplacian (second derivative in the horizontal direction plus the second derivative in the vertical direction) in the roughening matrix. The **L** matrix can be generated using the following MATLAB code:

```
L=zeros(14*14,256);
k=1;
for i=2:15,
  for j=2:15,
    M=zeros(16,16);
    M(i,j)=-4;
    M(i,j+1)=1;
    M(i,j-1)=1;
    M(i+1,j)=1;
    M(i-1,j)=1;
    L(k,:)=reshape(M,256,1)';
    k=k+1;
  end
end
```

What, if any, problems did you have in using the L-curve criterion on this problem? Plot the L-curve as well as your solution.

(d) Discuss your results. If vertical bands appeared in some of your solutions, can you explain why?

4. Apply second-order Tikhonov regularization to solve the problem of finding a smooth curve that approximately fits a set of noisy data points. Write a MATLAB program to find a function specified at the 191 points $x = 1, 1.1, 1.2, \ldots, 20$ that approximately fits the 20 data points specified at $x = 1, 2, 3, \ldots, 20$ given in the file **interpdata.mat**. Construct an appropriate 20 by 191 $\mathbf{G}$ matrix,and use the library function **get_l_rough** to obtain the necessary second-order roughening matrix, $\mathbf{L}$. Produce solutions for regularization parameter values of $\alpha = 0.2, 0.4, 0.6, \ldots, 10$ and show the tradeoff curve between 2-norm data misfit and model seminorm on a linear-linear plot. If the data noise is independent and normally distributed with a standard deviation of 0.3, use the discrepancy principle to find and plot an optimal interpolated curve along with the data points. What is the χ^2 value of this solution? Is it reasonable?

5. In some situations it is appropriate to bias the regularized solution towards a particular model $\mathbf{m}_0$. In this case, we would solve

$$\min \|\mathbf{Gm} - \mathbf{d}\|_2^2 + \alpha^2 \|\mathbf{L}(\mathbf{m} - \mathbf{m}_0)\|_2^2 \ . \tag{4.1}$$

Write this as an ordinary linear least squares problem. What are the normal equations? Can you find a solution for this problem using the GSVD?

4.3 Solutions

1. First, notice that since taking the square of $\|\mathbf{m}\|$ is a monotone increasing transformation, (4.2) is equivalent to

$$\begin{array}{lcl} \min & \|\mathbf{m}\|_2^2 & \\ & \|\mathbf{Gm} - \mathbf{d}\|_2^2 & \leq \quad \delta^2 \ . \end{array}$$

Let

$$f(\mathbf{m}) = \|\mathbf{m}\|_2^2$$

and

$$g(\mathbf{m}) = \|\mathbf{Gm} - \mathbf{d}\|_2^2 - \delta^2.$$

Then, (4.2) is shown to be equivalent to

$$\begin{array}{lcl} \min & f(\mathbf{m}) & \\ & g(\mathbf{m}) & \leq \quad 0 \ . \end{array}$$

By Theorem C.5, a minimizer of this problem can only occur at a point where

$$\nabla f(\mathbf{m}) + \lambda \nabla g(\mathbf{m}) = \mathbf{0}$$

and $\lambda \geq 0$.

Next, consider (4.4):

$$\min \quad \|\mathbf{Gm} - \mathbf{d}\|_2^2 + \alpha^2 \|\mathbf{m}\|_2^2 \ .$$

This is equivalent to

$$\min \quad g(\mathbf{m}) + \delta^2 + \alpha^2 f(\mathbf{m}).$$

The constant δ^2 will not affect the optimal $\mathbf{m}$, so we have

$$\min \quad g(\mathbf{m}) + \alpha^2 f(\mathbf{m}).$$

A minimum of this problem can only occur where

$$\nabla g(\mathbf{m}) + \alpha^2 \nabla f(\mathbf{m}) = \mathbf{0}.$$

Ignoring the special case $\alpha = 0$, we can divide through by α^2 to get:

$$\frac{1}{\alpha^2} \nabla g(\mathbf{m}) + \nabla f(\mathbf{m}) = \mathbf{0}.$$

This is exactly the same as the Lagrange multiplier optimality condition for (4.2) with $\lambda = 1/\alpha^2$ or $\alpha = \sqrt{1/\lambda}$.

So, for any optimal solution to (4.2), we can find a corresponding value of λ, and then set $\alpha = \sqrt{1/\lambda}$. An optimal solution to (4.4) with this value of the parameter α will also solve (4.2).

It can be shown that for $\alpha > 0$, (4.4) is strictly convex, and thus has a unique optimal solution.

For the special case $\alpha = 0$, (4.4) is not necessarily strictly convex, and there may be many optimal solutions. This corresponds to the limit as $\delta \to \infty$ in (4.2).

Finally, we'll show that (4.4) is equivalent to (4.5).

$$h(m) = \sum_{i=1}^{m} (\mathbf{Gm} - \mathbf{d})_i^2 + \alpha^2 \sum_{j=1}^{n} m_j^2.$$

$$h(m) = \sum_{i=1}^{m} (\mathbf{Gm} - \mathbf{d})_i^2 + \sum_{j=1}^{n} (\alpha m_j)^2.$$

$$h(m) = \left\| \begin{array}{c} \mathbf{Gm} - \mathbf{d} \\ \alpha \mathbf{m} \end{array} \right\|_2^2$$

$$h(m) = \left\| \left[\begin{array}{c} \mathbf{G} \\ \alpha \mathbf{I} \end{array} \right] \mathbf{m} - \left[\begin{array}{c} \mathbf{d} \\ \mathbf{0} \end{array} \right] \right\|_2^2 .$$

2. (a) This problem was previously discretized in the same manner in Exercise 3.5.

(b) Note that since we have 20 equations and 20 model parameters, our χ^2 distribution actually has 0 degrees of freedom, and we should theoretically be able to get a perfect fit to the data. However, because of the harsh ill conditioning of the problem, near exact fits to the data will tend to produce wild results, and regularization is necessary to produce reasonable models. Given that the data are accurate to four digits past the decimal point, a reasonable estimate of the standard deviation for each measurement is 0.00005. Using our rule of thumb, a reasonable value for δ is

$$\delta = 0.00005\sqrt{n} \approx 2.2361 \times 10^{-4} \ .$$

(c) Figure 4.2 shows the GCV curve for the zeroth-order Tikhonov regularization solution, and the corresponding model is shown in Figure 4.3. Note that it may be easier to see the full range of GCV curves in ill posed problems when they are plotted on log-log axes. Using the value of δ from part (b), we found the discrepancy principle solution shown in Figure 4.4. Figure 4.5 shows the L-curve for zeroth-order Tikhonov regularization, which has a very well defined corner when plotted on log-log axes. The corresponding solution is shown in Figure 4.6. The three solutions are reasonably similar.

(d) Figures 4.8 and 4.9 show the solutions obtained using first-order Tikhonov regularization. The L-curve corner is still well defined for first-order regularization, and the corresponding solutions tend to increase the value of the model near $x = 0$ to avoid the sharp change in slope visible there for the zeroth-order solutions.

(e) Figure 4.13 and 4.14 show the solutions obtained using second-order Tikhonov regularization. These solutions, being smoother still, all tend to further reduce the excursions visible in the zeroth-order solutions. The L-curve is not particularly well defined in the second–order problem, but has a relatively subtle corner near $\alpha = 3.19 \times 10^{-5}$ that produces a smooth solution.

(f) Figures 4.17 through 4.19 show the diagonal resolution values for the various solutions using the appropriate α values of the Tikhonov regularization parameter, showing generally poor resolution everywhere and exceptionally poor resolution near $x = 0$. The rightmost elements are relatively well resolved.

Because the problem is so ill-posed, only the grossest characteristics of the true model will be recovered by our inverse procedure. All of our solutions showed that values were generally larger on the left half of the model and smaller on the right half of the model, and there was agreement that the peak value was around 1. There was also agreement that the lower values on the right hand side of the model were around 0.5. Beyond that, the inverse solutions could not consistently detect features at finer levels of detail. In particular, it isn't possible to tell whether the peak on the left edge of the model is at $x = 0$ or a bit further to the right (say $x = 0.3$.) It also isn't possible to tell for sure whether or not there is a peak towards the right end of the solution.

Figure 4.20 shows the true model. In fact, it does have a peak at around $x = 0.2$, with a peak value of about 1. There is a second peak at $x = 0.9$.

Notice that the first- and second-order Tikhonov regularized solutions tended to miss that the peak was distinctly to the right of $x = 0$. The problem here was that in obtaining a very smooth solution, the top of the peak had to be rounded off. The zeroth-order solutions did not suffer from this, because zeroth-order regularization does not enforce smoothness. In general, it may be best to use the lowest order regularization which "successfully" regularizes the solution to best recover such features if there is reason to believe that distinct peaks exist in the true model.

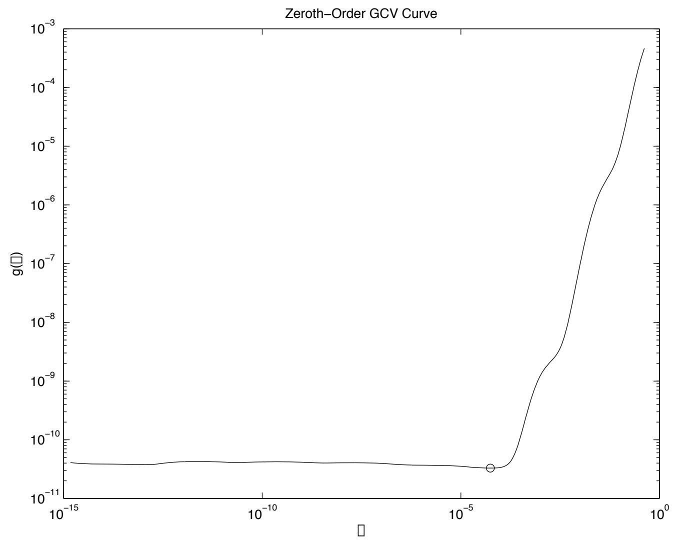

Figure 4.2: GCV curve for zeroth-order regularization.

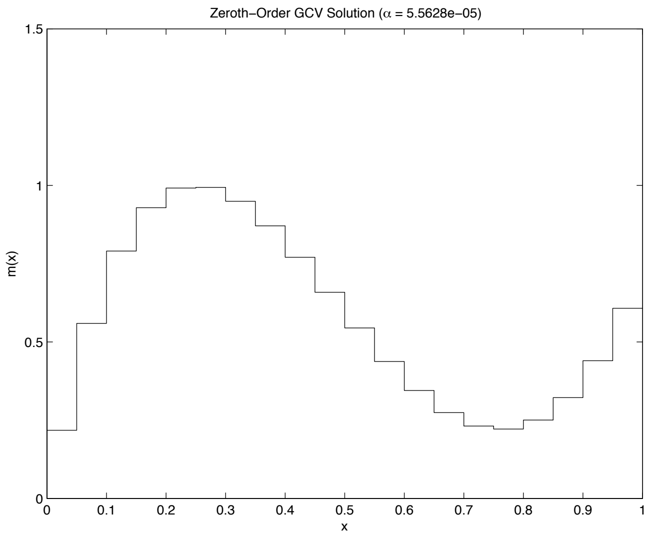

Figure 4.3: Zeroth-order Tikhonov solution, GCV α.

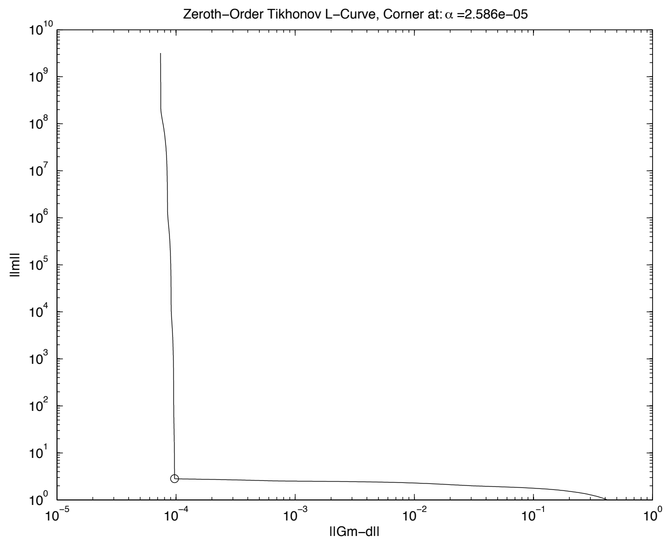

Figure 4.4: Zeroth-order Tikhonov solution, discrepancy principle α.

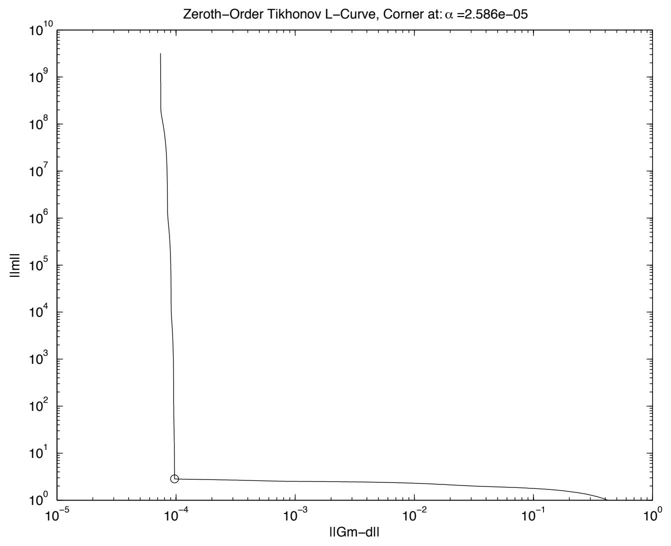

Figure 4.5: L-curve for zeroth-order regularization.

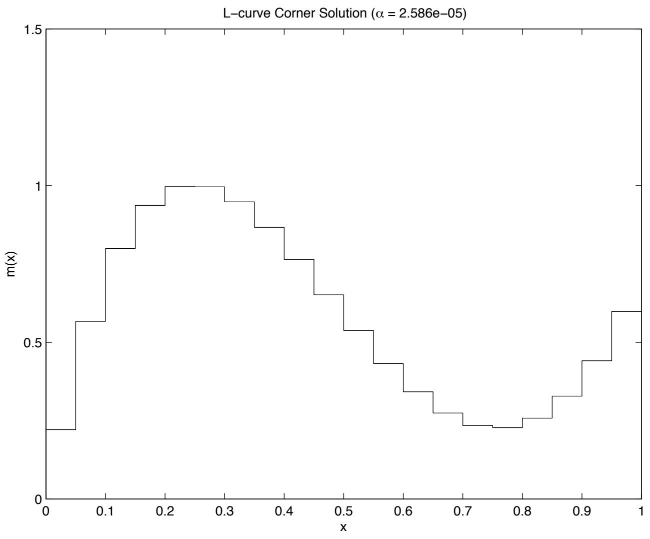

Figure 4.6: Zeroth-order Tikhonov solution, L-curve α.

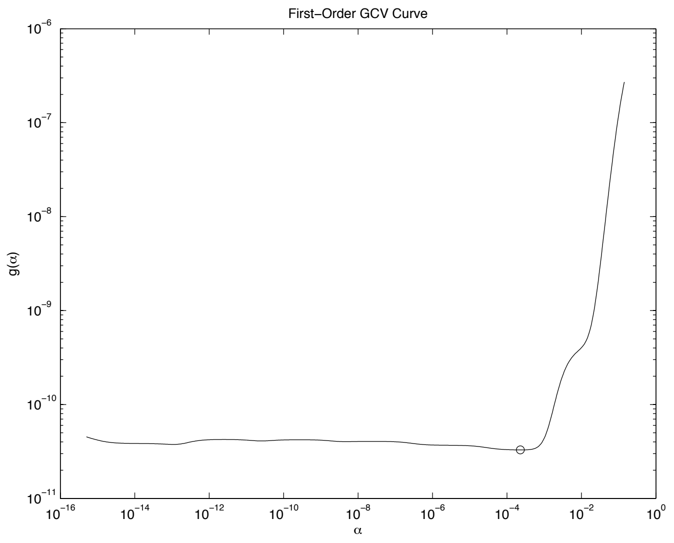

Figure 4.7: GCV curve for first-order regularization.

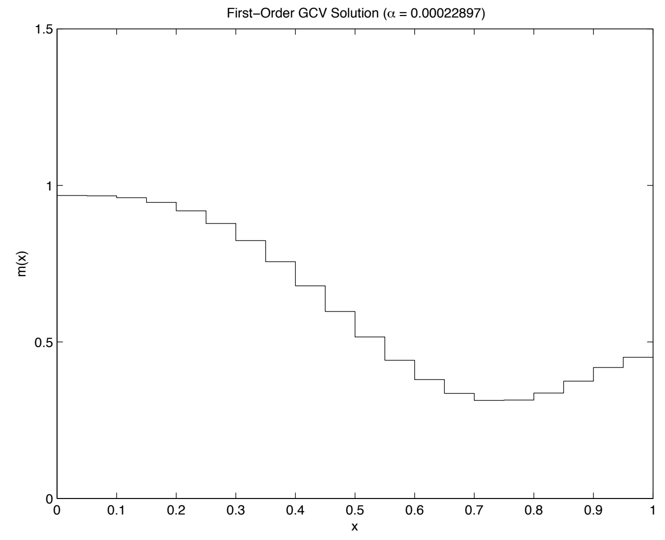

Figure 4.8: First-order Tikhonov solution, GCV α.

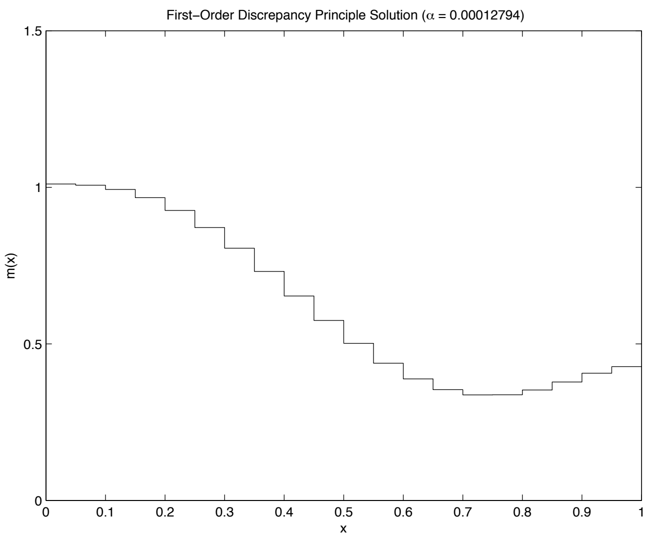

Figure 4.9: First-order Tikhonov solution, discrepancy principle α.

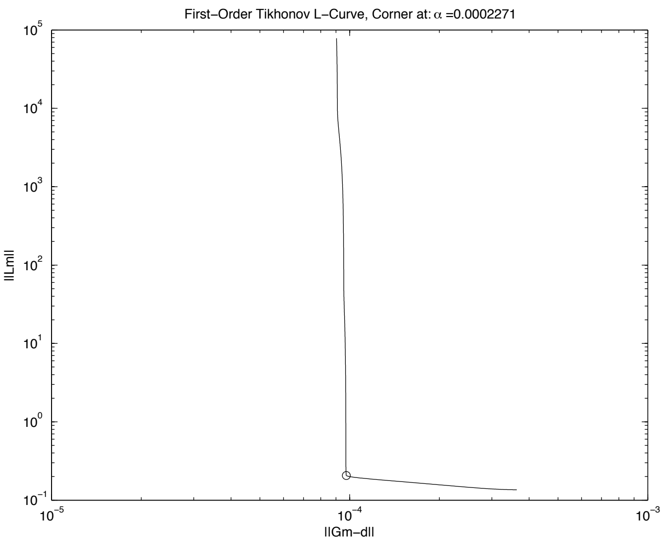

Figure 4.10: L-curve for first-order regularization.

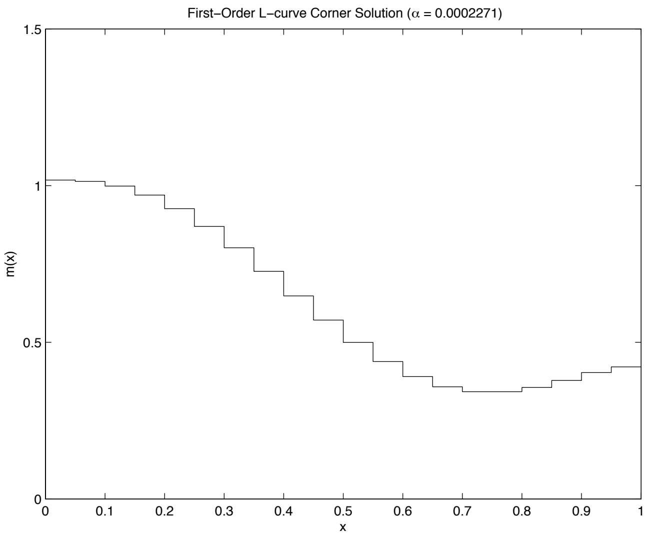

Figure 4.11: First-order Tikhonov solution, L-curve α.

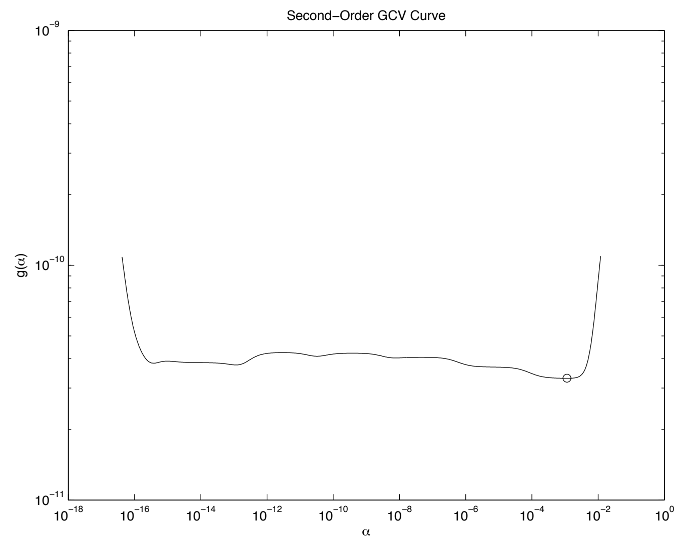

Figure 4.12: GCV curve for second-order regularization.

```
%Chapter 4, problem 2; Ill posed Tikhonov regularization problem.
%
% Load the data in.
%
clear
load ifk.mat
%
%Setup vectors of the x and y collocation points.
%
x=(0.025:0.05:0.975)';
y=x;
%
%Set up the G matrix as in Exercise 3.5.
%
%Part (a) discretize the model via simple collocation
for i=1:20, for j=1:20, G(i,j)=0.05*x(j)*exp(-x(j)*y(i)); end; end;
%
%Compute the SVD and G'*G.
%
[U,S,V]=svd(G);
GTG=G'*G;

% Part (c) Solve the zeroth order problem, selecting the regularization
% parameter in three different ways
%
%
%Calculate the zeroth order L-curve and find its corner
%
figure(1);
[rho,eta,reg_param]=l_curve_tikh_svd(U,diag(S),d,1000);
[alpha_corner,ireg_corner,kappa]=l_curve_corner(rho,eta,reg_param);
loglog(rho,eta)
hold on
loglog(rho(ireg_corner),eta(ireg_corner),'o')
hold off
xlabel('||Gm-d||')
ylabel('||m||')
title(['Zeroth-Order Tikhonov L-Curve, Corner at: \alpha =',num2str(alpha_corner)]);
print -deps lcurve0.eps
%
%Solve by discrepancy principle.
%
delta = 5e-5*sqrt(20);
%find the corresponding regularization parameter via interpolation
alpha_disc0=interp1(reg_param,rho,delta);
Gsharp_disc0=(GTG+alpha_disc0^2*eye(20))\G';
mdisc0=Gsharp_disc0*d;
```

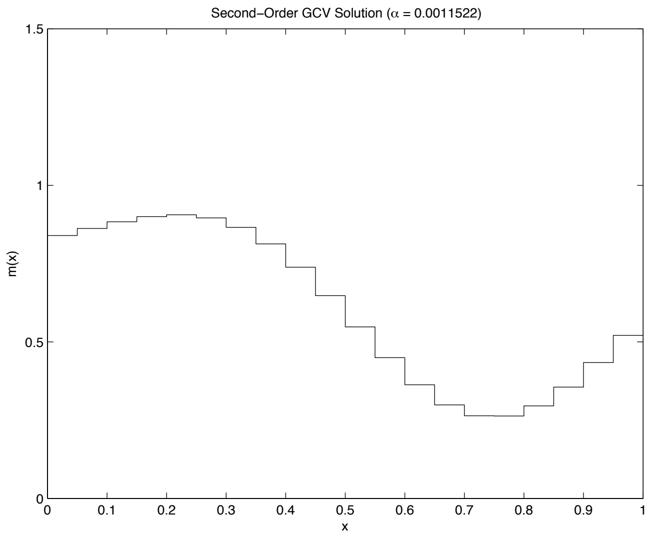

Figure 4.13: Sceond-order Tikhonov solution, GCV α.

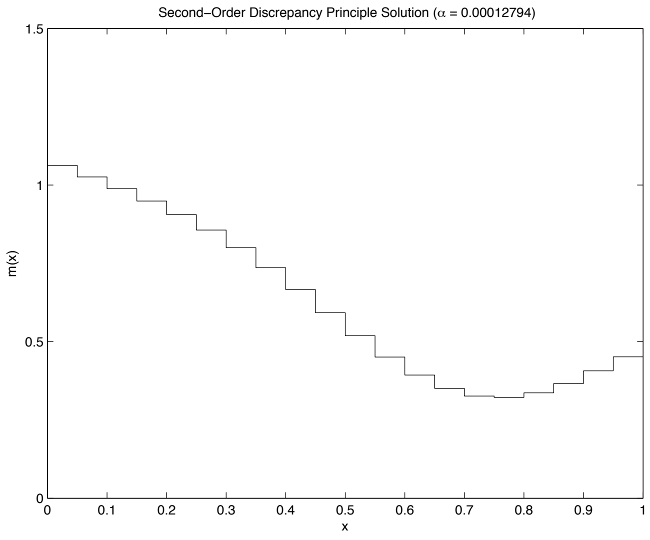

Figure 4.14: Second-order Tikhonov solution, discrepancy principle α.

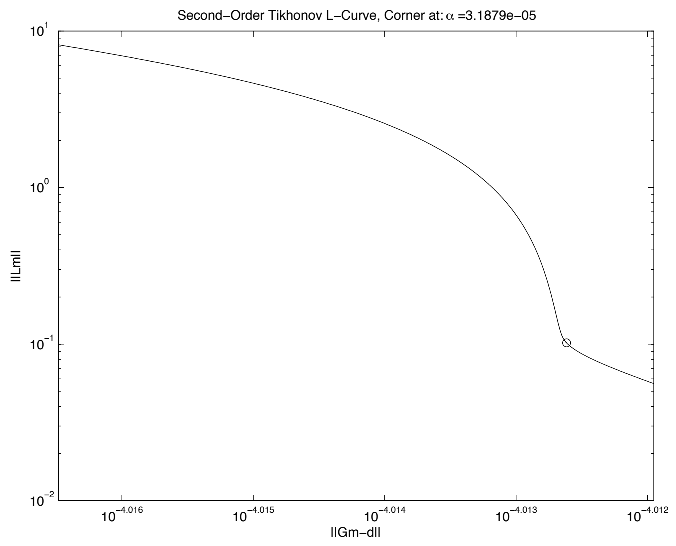

Figure 4.15: L-curve for second-order regularization.

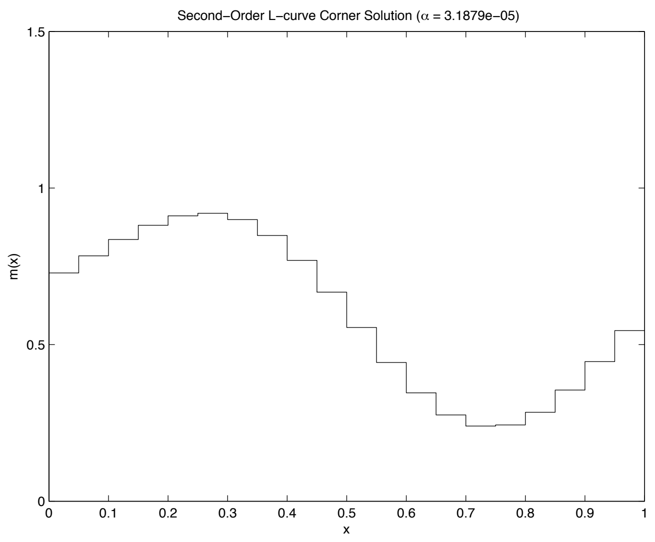

Figure 4.16: Second-order Tikhonov solution, L-curve α.

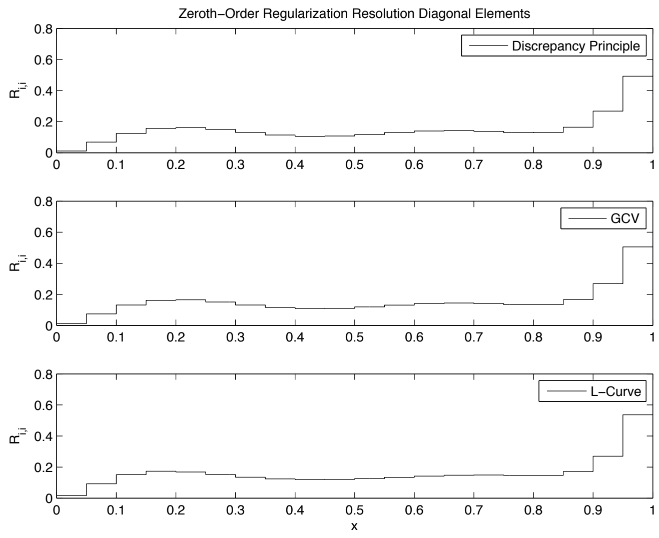

Figure 4.17: Resolution diagonal elements for zeroth-order solutions.

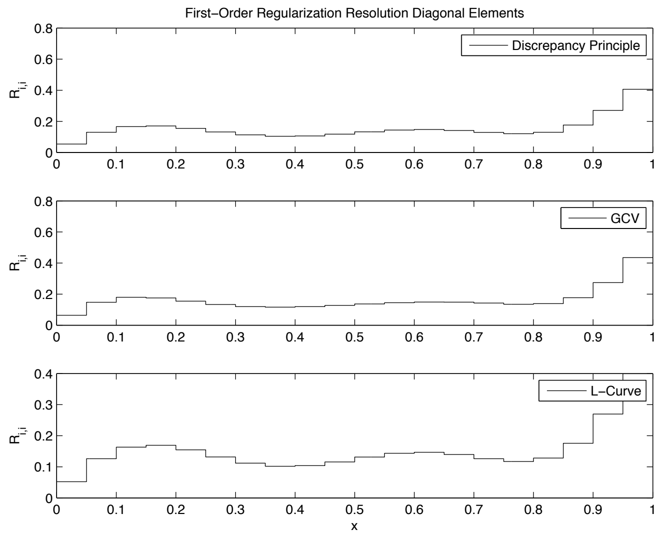

Figure 4.18: Resolution diagonal elements for first-order solutions.

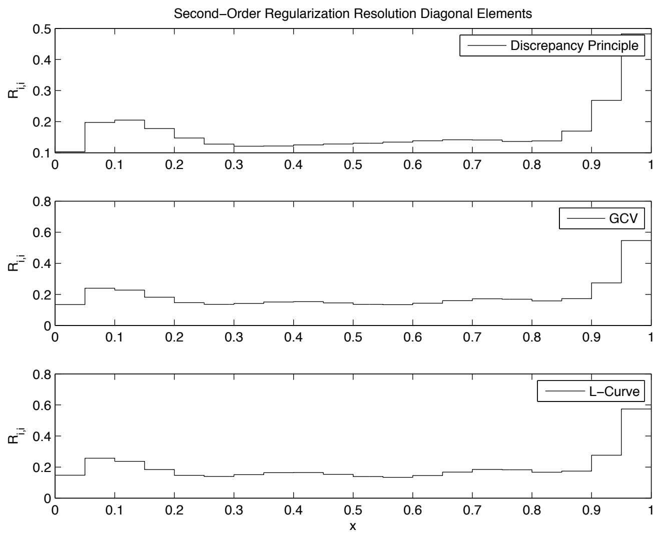

Figure 4.19: Resolution diagonal elements for second-order solutions.

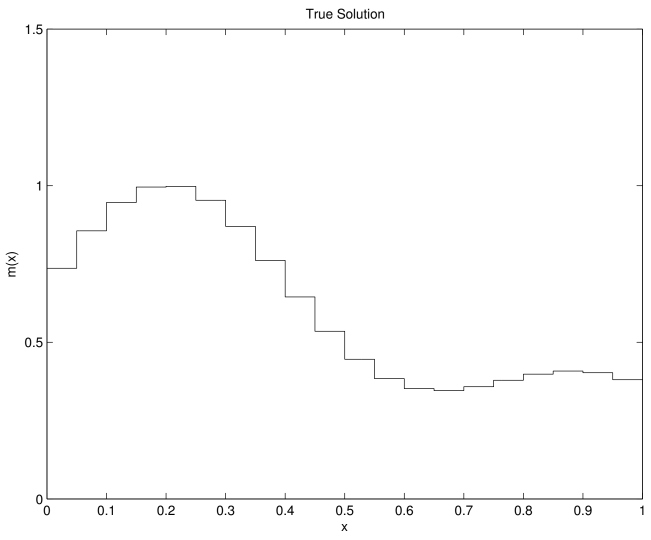

Figure 4.20: True solution.

```
%resolution matrix
R_disc0=Gsharp_disc0*G;
figure(2);
plotconst(mdisc0,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['Zeroth-Order Discrepancy Principle Solution (\alpha = ',num2str(alpha_disc0),')'])
print -deps mdisc0.eps
%
%Solve with GCV
%
%L0 is just the identity matrix ("roughening" matrix for zeroth order
%regularization)
L0 = get_l_rough(20,0);
[U0,V0,X0,Lam0,Mu0]=gsvd(G,L0);
lam=sqrt(diag(Lam0'*Lam0));
mu=sqrt(diag(Mu0'*Mu0));
p=rank(L0);
sm0=[lam(1:p),mu(1:p)];
%get the gcv values varying alpha
[alpha0,g0,reg_param0]=gcval(U0,sm0,d,1000);

%find the minimum value of of the g functions
[ming0,iming0] = min(g0);
alpha_gcv0=reg_param0(iming0);

%plot the zeroth order GCV function and indicate its minimum (it is subtle,
%so we will use a log-log plot)
figure(3);
loglog(reg_param0,g0);
hold on
loglog(alpha_gcv0,ming0,'o')
hold off
title('Zeroth-Order GCV Curve')
xlabel('\alpha')
ylabel('g(\alpha)')
print -deps gcv0.eps

%find the GCV solution and plot it
Gsharp_gcv0=(GTG+alpha_gcv0^2*eye(20))\G';
mgcv0=Gsharp_gcv0*d;
%resolution matrix
R_gcv0=Gsharp_gcv0*G;
figure(4);
plotconst(mgcv0,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['Zeroth-Order GCV Solution (\alpha = ',num2str(alpha_gcv0),')'])
print -deps mgcv0.eps

%
%Solve from the L-curve corner
%
Gsharp_L0=(GTG+alpha_corner^2*eye(size(GTG)))\G';
mL0=Gsharp_L0*d;
R_L0=Gsharp_L0*G;
figure(5);
plotconst(mL0,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['L-curve Corner Solution (\alpha = ',num2str(alpha_corner),')'])
print -deps mltik0.eps;

%Resolution Matrix Diagonal Element Plot
figure(6)
subplot(3,1,1)
plotconst(diag(R_disc0),0,1)
legend('Discrepancy Principle')
ylabel('R_{i,i}')
title('Zeroth-Order Regularization Resolution Diagonal Elements')
subplot(3,1,2)
plotconst(diag(R_gcv0),0,1)
ylabel('R_{i,i}')
legend('GCV')
subplot(3,1,3)
plotconst(diag(R_L0),0,1)
legend('L-Curve')
ylabel('R_{i,i}')
xlabel('x')
print -deps res0.eps
%
%
% Part (d) Solve the first-order problem, selecting the regularization
% parameter in three different ways
%
%
%Calculate the first-order L-curve and find its corner
%
```

```
L1=get_l_rough(20,1);
[U1,V1,X1,Lam1,Mu1]=gsvd(G,L1);
p=rank(L1);
sm1=[lam(1:p),mu(1:p)];

figure(7);
[rho,eta,reg_param]=l_curve_tikh_gsvd(U1,d,X1,Lam1,Mu1,G,L1,1000,1e-10,0.15);
[alpha_corner,ireg_corner,kappa]=l_curve_corner(rho,eta,reg_param);
loglog(rho,eta)
hold on
loglog(rho(ireg_corner),eta(ireg_corner),'o')
hold off
xlabel('||Gm-d||')
ylabel('||Lm||')
title(['First-Order Tikhonov L-Curve, Corner at: \alpha =',num2str(alpha_corner)]);
print -deps lcurve1.eps
%
%Solve by discrepancy principle.
%
%find the corresponding regularization parameter via interpolation
alpha_disc1=interp1(reg_param,rho,delta);
Gsharp_disc1=(GTG+alpha_disc0^2*(L1'*L1))\G';
mdisc1=Gsharp_disc1*d;
%resolution matrix
R_disc1=Gsharp_disc1*G;
figure(8);
plotconst(mdisc1,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['First-Order Discrepancy Principle Solution (\alpha = ',num2str(alpha_disc0),')'])
print -deps mdisc1.eps
%
%
%Solve with GCV
%
%L0 is just the identity matrix ("roughening" matrix for zeroth order
%regularization)
lam=sqrt(diag(Lam1'*Lam1));
mu=sqrt(diag(Mu1'*Mu1));
p=rank(L1);
sm1=[lam(1:p),mu(1:p)];
%get the gcv values varying alpha
[alpha1,g1,reg_param1]=gcval(U1,sm1,d,1000);

%find the minimum value of of the g functions
[ming1,iming1] = min(g1);
alpha_gcv1=reg_param1(iming1);

%plot the zeroth order GCV function and indicate its minimum (it is subtle,
%so we will use a log-log plot)
figure(9);
loglog(reg_param1,g1);
hold on
loglog(alpha_gcv1,ming1,'o')
hold off
title('First-Order GCV Curve')
xlabel('\alpha')
ylabel('g(\alpha)')
print -deps gcv1.eps

%find the GCV solution and plot it
Gsharp_gcv1=(GTG+alpha_gcv0^2*(L1'*L1))\G';
mgcv1=Gsharp_gcv1*d;
%resolution matrix
R_gcv1=Gsharp_gcv1*G;
figure(10);
plotconst(mgcv1,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['First-Order GCV Solution (\alpha = ',num2str(alpha_gcv1),')'])
print -deps mgcv1.eps

%
%Solve from the L-curve corner
%
Gsharp_L1=(GTG+alpha_corner^2*(L1'*L1))\G';
mL1=Gsharp_L1*d;
R_L1=Gsharp_L1*G;
figure(11);
plotconst(mL1,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['First-Order L-curve Corner Solution (\alpha = ',num2str(alpha_corner),')'])
print -deps mltik1.eps;

%Resolution Matrix Diagonal Element Plot
figure(12)
subplot(3,1,1)
```

```
plotconst(diag(R_disc1),0,1)
legend('Discrepancy Principle')
ylabel('R_{i,i}')
title('First-Order Regularization Resolution Diagonal Elements')
subplot(3,1,2)
plotconst(diag(R_gcv1),0,1)
ylabel('R_{i,i}')
legend('GCV')
subplot(3,1,3)
plotconst(diag(R_L1),0,1)
legend('L-Curve')
ylabel('R_{i,i}')
xlabel('x')
print -deps res1.eps
%
%
% Part (e) Solve the first-order problem, selecting the regularization
% parameter in three different ways
%
%
%Calculate the first-order L-curve and find its corner
%
L2=get_l_rough(20,2);
[U2,V2,X2,Lam2,Mu2]=gsvd(G,L2);
%p=rank(L2);

figure(13);
%[rho,eta,reg_param]=l_curve_tikh_gsvd(U2,d,X2,Lam2,Mu2,G,L2,1000,0.000000001,100000);
[rho,eta,reg_param]=l_curve_tikh_gsvd(U2,d,X2,Lam2,Mu2,G,L2,1000,1e-6,0.0001);
[alpha_corner,ireg_corner,kappa]=l_curve_corner(rho,eta,reg_param);
loglog(rho,eta)
hold on
loglog(rho(ireg_corner),eta(ireg_corner),'o')
hold off
xlabel('||Gm-d||')
ylabel('||Lm||')
title(['Second-Order Tikhonov L-Curve, Corner at: \alpha =',num2str(alpha_corner)]);
print -deps lcurve2.eps

%
%Solve by discrepancy principle.
%
%find the corresponding regularization parameter via interpolation
alpha_disc2=interp1(reg_param,rho,delta);
Gsharp_disc2=(GTG+alpha_disc0^2*(L2'*L2))\G';
mdisc2=Gsharp_disc2*d;
%resolution matrix
R_disc2=Gsharp_disc2*G;
figure(14);
plotconst(mdisc2,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['Second-Order Discrepancy Principle Solution (\alpha = ',num2str(alpha_disc0),')'])
print -deps mdisc2.eps
%
%
%Solve with GCV
%
%L0 is just the identity matrix ("roughening" matrix for zeroth order
%regularization)
lam=sqrt(diag(Lam2'*Lam2));
mu=sqrt(diag(Mu2'*Mu2));
p=rank(L2);
sm2=[lam(1:p),mu(1:p)];
%get the gcv values varying alpha
[alpha2,g2,reg_param2]=gcval(U2,sm2,d,1000);

%find the minimum value of of the g functions
[ming2,iming2] = min(g2);
alpha_gcv2=reg_param2(iming2);

%plot the zeroth order GCV function and indicate its minimum (it is subtle,
%so we will use a log-log plot)
figure(15);
loglog(reg_param2,g2);
hold on
loglog(alpha_gcv2,ming2,'o')
hold off
title('Second-Order GCV Curve')
xlabel('\alpha')
ylabel('g(\alpha)')
print -deps gcv2.eps

%find the GCV solution and plot it
Gsharp_gcv2=(GTG+alpha_gcv0^2*(L2'*L2))\G';
mgcv2=Gsharp_gcv2*d;
%resolution matrix
R_gcv2=Gsharp_gcv2*G;
figure(16);
plotconst(mgcv2,0,1);
```

```
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['Second-Order GCV Solution (\alpha = ',num2str(alpha_gcv2),')'])
print -deps mgcv2.eps

%
%Solve from the L-curve corner
%
Gsharp_L2=(GTG+alpha_corner^2*(L2'*L2))\G';
mL2=Gsharp_L2*d;
R_L2=Gsharp_L2*G;
figure(17);
plotconst(mL2,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['Second-Order L-curve Corner Solution (\alpha = ',num2str(alpha_corner),')'])
print -deps mltik2.eps;

%Resolution Matrix Diagonal Element Plot
figure(18)
subplot(3,1,1)
plotconst(diag(R_disc2),0,1)
legend('Discrepancy Principle')
ylabel('R_{i,i}')
title('Second-Order Regularization Resolution Diagonal Elements')
subplot(3,1,2)
plotconst(diag(R_gcv2),0,1)
ylabel('R_{i,i}')
legend('GCV')
subplot(3,1,3)
plotconst(diag(R_L2),0,1)
legend('L-Curve')
ylabel('R_{i,i}')
xlabel('x')
print -deps res2.eps

%Show the True Solution
load mtrue.mat
figure(19);
plotconst(mtrue,0,1);
axis([0 1 0 1.5]);
ylabel('m(x)')
xlabel('x')
title(['True Solution)'])
print -deps mtrue.eps
```

3. (a) Figure 4.21 shows the singular values of the $\mathbf{G}$ matrix. Note that there is a sharp drop after the 243rd singular value. Figure 4.22 shows the truncated SVD solution obtained by using the first 243 singular values. Notice the vertical "bands" in the solution and the general noise level. In an attempt to clean up the noise and produce a simpler model, Figure 4.23 shows the solution using only the first 200 singular values. This solution clearly shows a high velocity/reduced slowness feature near location (9,5).

   (b) Figure 4.24 shows the L-curve for zeroth-order Tikhonov regularization. The corner is quite distinct. Figure 4.25 shows the corresponding solution. This solution is similar to the TSVD solution with 243 singular values.

   The difficulty with using the discrepancy principle on this problem is that we have $m = 256$ and $n = 256$. Thus our χ^2 distribution has zero degrees of freedom, and theoretically, the misfit should be exactly zero. In fact, the misfit of our solution is far smaller than the bound that we might establish by taking $\sqrt{256}$ times 0.0005 (the error level of the measurements).

   (c) Figure 4.26 shows the L-curve for second order Tikhonov regularization. This time, the corner of the L-curve is not very distinct. Figure

4.27 shows a resulting solution. Next, we consider the regularization parameter corresponding to the second "corner" of the L-curve, near 0.059. This solution is shown in Figure 4.28. Figure 4.29 shows the diagonal of the resolution matrix for this solution. The resolution is generally quite good, except for areas at the top and bottom of the image, where there are few ray paths. Figure 4.30 shows the resolution of a spike at location (9,6). The spike is quite clearly resolved. This gives us confidence that the high velocity region near (9,5) in Figure 4.28 is a real feature.

(d) Of all the solutions obtained here, the one shown in Figure 4.28 seems most reasonable if we expect the true model to be smooth. The other solutions tend to exhibit vertical bands. These appear because it is possible to shift slowness from one column of the model to a neighboring column without much affecting the observed travel times. Such shifts fall within the model null space and are difficult if not impossible to eliminate using zeroth-order regularization. It is possible to eliminate them using second order regularization, but it takes a substantial amount of regularization to accomplish this. The resolution of this second-order solution (4.28) is quite good in most parts of the model, as seen by the diagonal elements of **R** shown in corresponding model locations in Figure 4.29, with degraded resolution near the top and bottom of the model. Representative spike tests shown in Figures 4.30 and 4.31 reflect this. The true model is shown in Figure 4.32, showing that the solution in Figure 4.28 is indeed reasonably close to the true (smooth) model.

```
%Chapter 4, problem 3; Crosswell tomography Exercise 3
% First clear the workspace
%
clear
%
% Next, make the G matrix.
%
G=zeros(256,256);
k=1;
for i=1:16,
  for j=1:16,
    M=zeros(16,16);
    slope=((j-0.5)-(i-0.5))/16;
    xstart=0;
    xend=16;
    ystart=i-0.5;
    yend=j-0.5;
    deltax=0.001;
    deltay=slope*deltax;
    lbit=sqrt(deltax^2+deltay^2);
    for x=0:deltax:16,
      y=ystart+x*slope;
      if ((ceil(x)==ceil(x+deltax)) && (ceil(y)==ceil(y+deltay))),
        M(ceil(x),ceil(y))=M(ceil(x),ceil(y))+lbit;
      end;
    end;
    V=reshape(M',256,1);
    G(k,:)=V';
    k=k+1;
  end;
end;
G=G*100;
%
%model axes
Xa=linspace(50,1550,16);
Ya=Xa;
%
% Make the true model and true data.
%
```

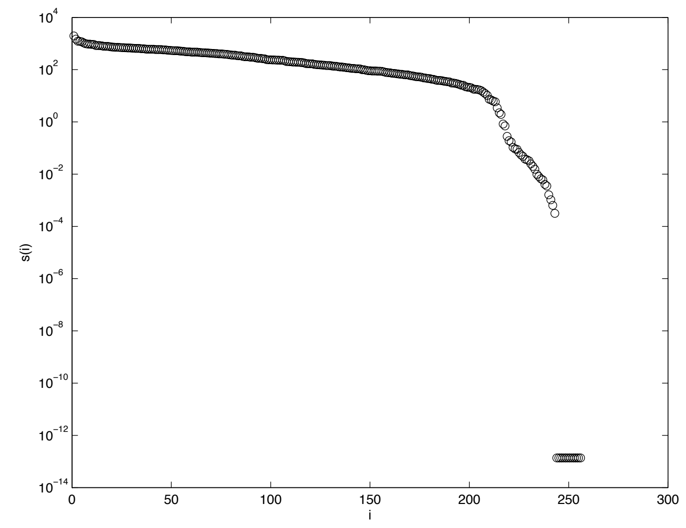

Figure 4.21: Singular values of the G matrix.

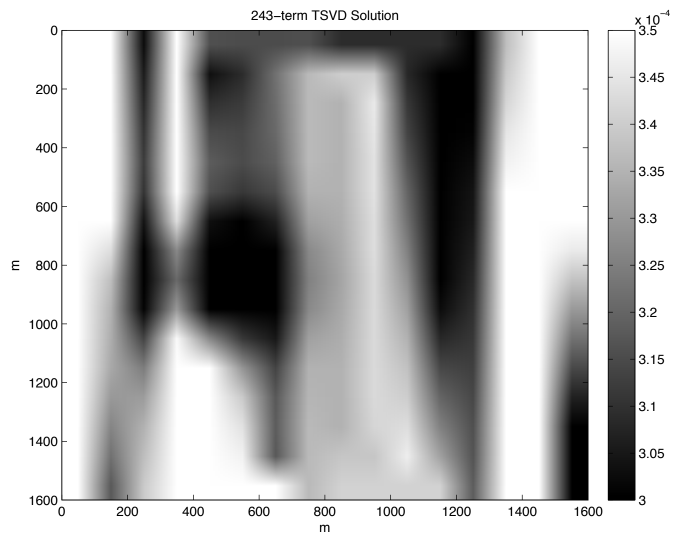

Figure 4.22: Truncated SVD solution, 243 singular values.

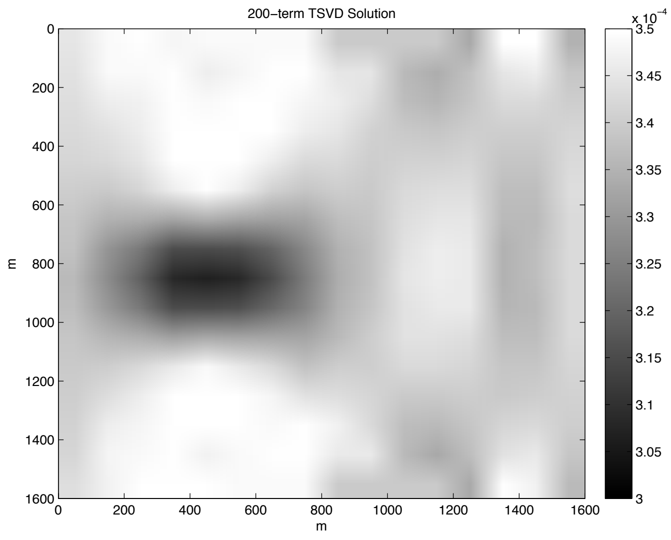

Figure 4.23: Truncated SVD solution, 200 singular values.

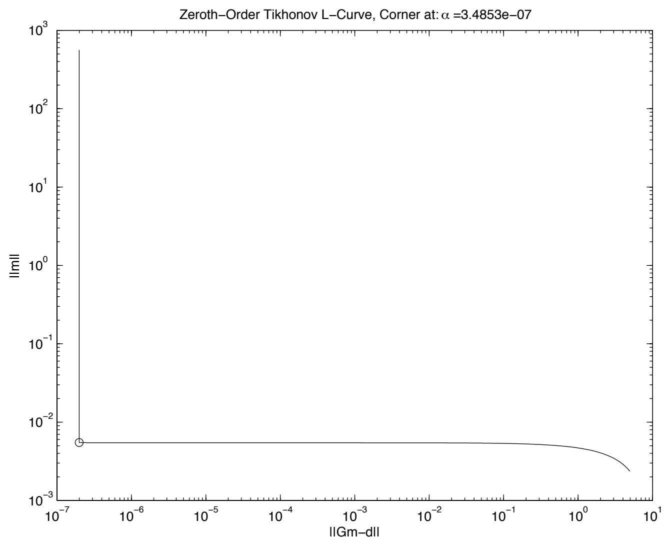

Figure 4.24: L-curve for zeroth-order Regularization.

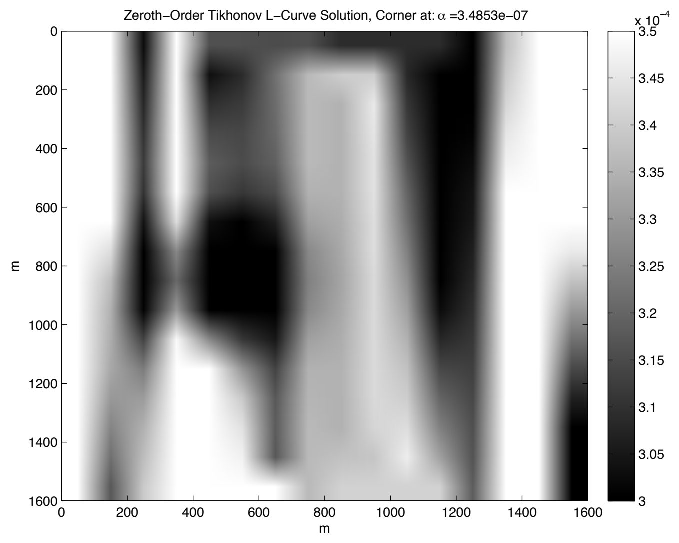

Figure 4.25: Zeroth-order Tikhonov L-curve sSolution.

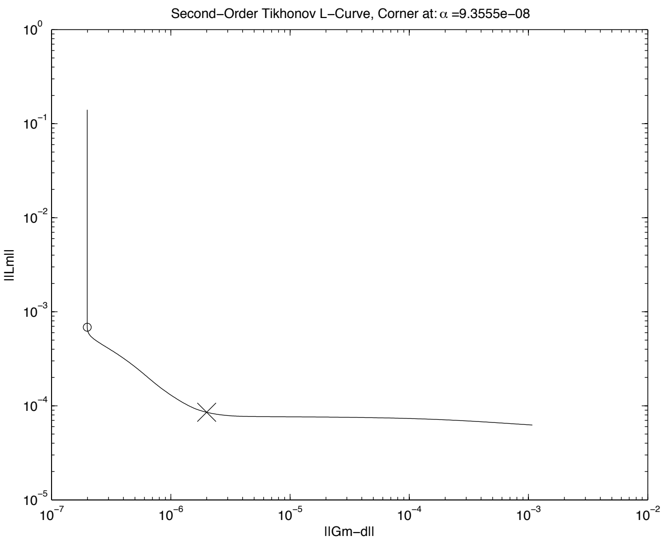

Figure 4.26: L-curve for second-order regularization. The two indicated corners marked with a circle and an 'x' correspond to the solutions shown in Figures 4.27 and 4.28, respectively.

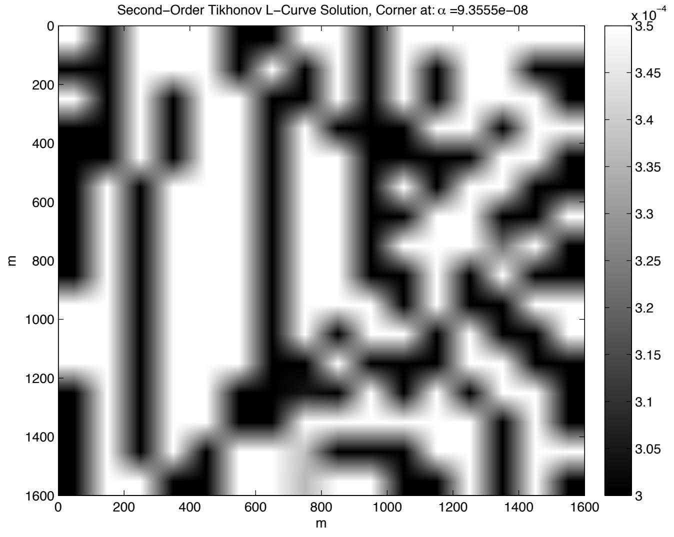

Figure 4.27: Second-order Tikhonov solution, $\alpha = 9.4 \times 10^{-8}$.

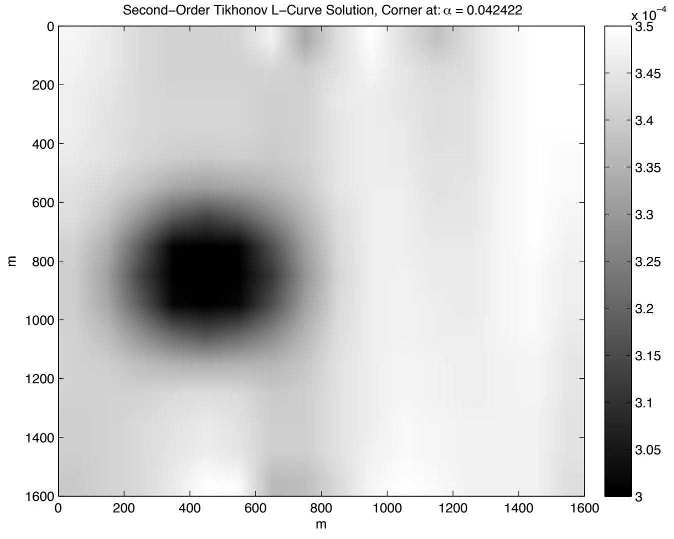

Figure 4.28: Second-order Tikhonov solution, $\alpha = 0.042$.

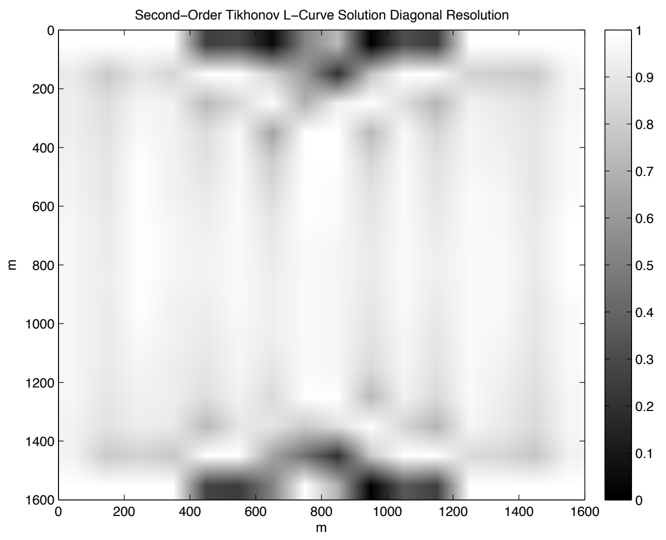

Figure 4.29: Diagonal elements of **R** for the solution shown in Figure 4.28.

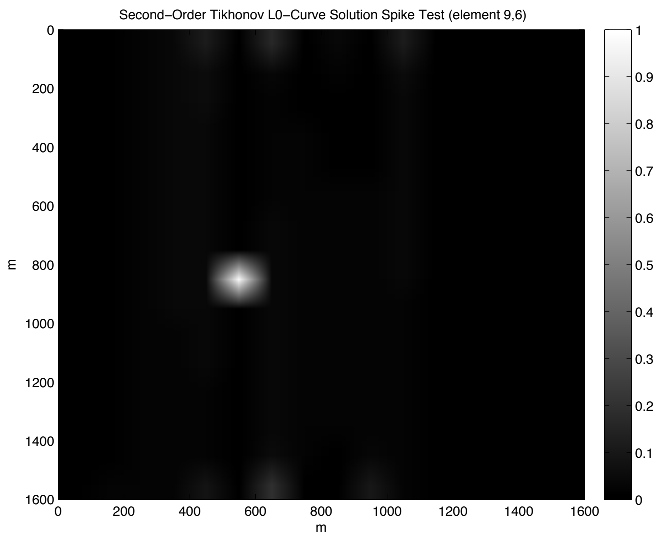

Figure 4.30: Resolution of a spike at (9,6) for the solution shown in Figure 4.28.

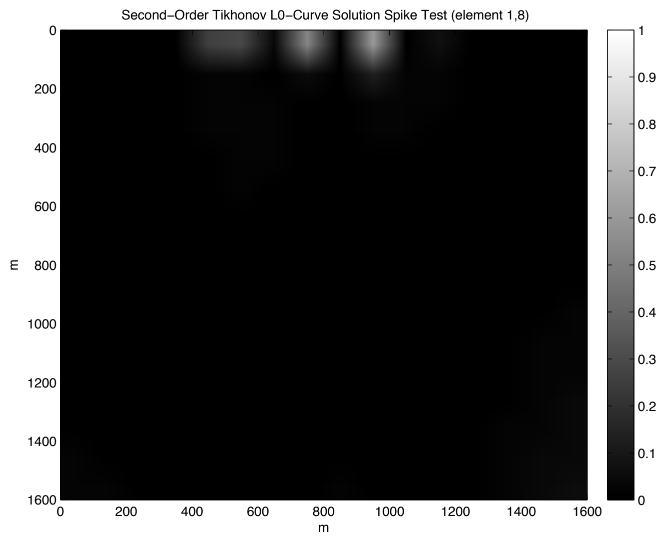

Figure 4.31: Resolution of a spike at (1,8) for the solution shown in Figure 4.28.

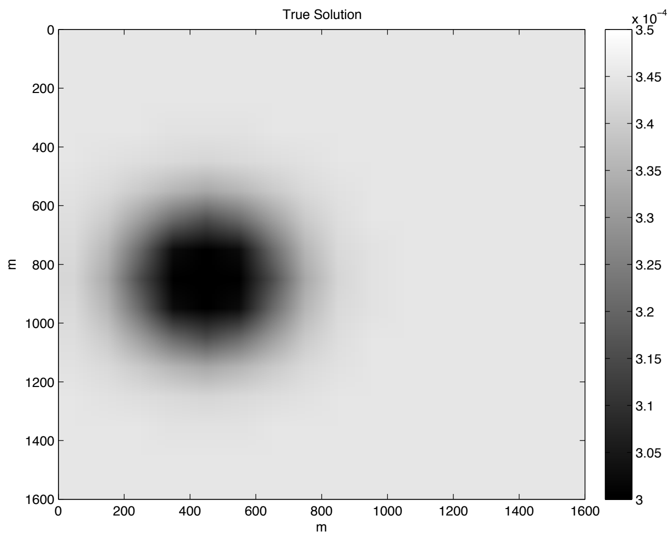

Figure 4.32: The true model.

```
mtruem=zeros(16,16);
for i=1:16,
  for j=1:16,
    mtruem(i,j)=1/(2900+600*exp(-0.2*((i-9)^2+(j-5)^2)));
  end;
end;
mtruev=reshape(mtruem,256,1);
dtruev=G*mtruev;
%
% Construct the noisy data.  Standard deviation is 0.5 milliseconds.
%
randn('state',0);
dn=dtruev+0.0005*randn;
%
% Save data for the students to use in solving the problem.
%
%save prob.mat G dn
%
% Find the SVD of G.
%
[UU,SS,VV]=svd(G);
%
%precalculate G'*G;
GTG=G'*G;
%
% Look at the singular values.
%
figure(1)
semilogy(diag(SS),'ko');
xlabel('i')
ylabel('s(i)')
print -deps singvals.eps

disp('singularvalues 243 and 244')
SS(243,243)
SS(244,244)
%
% It looks like p=243 may be good for the generalized inverse solution.
%
%calculate the TSVD solution
m243=zeros(length(V),1);
s=diag(SS);
for i=1:243
    m243=m243+((UU(:,i)'*dn)/s(i))*VV(:,i);
end
figure(2);
imagesc(Xa,Ya,reshape(m243,16,16),[3.0e-4 3.5e-4]);
title('243-term TSVD Solution')
ylabel('m')
xlabel('m')
colorbar
colormap(gray);
print -deps m243.eps
%
% This solution is a bit noisy, so we'll go back to the svd log "corner" near p=200
% and look at that solution too.
%
%calculate the TSVD solution
m200=zeros(length(V),1);
s=diag(SS);
for i=1:200
    m200=m200+((UU(:,i)'*dn)/s(i))*VV(:,i);
end
figure(3);
imagesc(Xa,Ya,reshape(m200,16,16),[3.0e-4 3.5e-4]);
title('200-term TSVD Solution')
ylabel('m')
xlabel('m')
colorbar
colormap(gray);
print -deps m200.eps
%
% Next, try 0th order Tikhonov regularization.
%
%
figure(4);
[rho,eta,reg_param]=l_curve_tikh_svd(UU,s,dn,1000);
[alpha_corner,ireg_corner,kappa]=l_curve_corner(rho,eta,reg_param);
loglog(rho,eta)
hold on
loglog(rho(ireg_corner),eta(ireg_corner),'o')
hold off
xlabel('||Gm-d||')
ylabel('||m||')
title(['Zeroth-Order Tikhonov L-Curve, Corner at: \alpha =',num2str(alpha_corner)]);

print -deps lcurve0.eps
%
% Show zeroth-order solution
I=eye(size(GTG));
figure(5);
```

```
%We'll calculate this using the filter factor series
m0=zeros(256,1);
for i=1:256
    f(i)=s(i)^2/(alpha_corner^2+s(i)^2);
    m0=m0+f(i)*((UU(:,i)'*dn)/s(i))*VV(:,i);
end

imagesc(Xa,Ya,reshape(m0,16,16),[3.0e-4 3.5e-4]);
colorbar
colormap(gray);
title(['Zeroth-Order Tikhonov L-Curve Solution, Corner at: \alpha =',num2str(alpha_corner)]);
ylabel('m')
xlabel('m')
colorbar;
colormap(gray);
print -deps mtik0.eps
%
% Next, apply 2nd order Tikhonov regularization.
%
%Make the L matrix.
%
L2=zeros(14*14,256);
k=1;
for i=2:15,
  for j=2:15,
    M=zeros(16,16);
    M(i,j)=-4;
    M(i,j+1)=1;
    M(i,j-1)=1;
    M(i+1,j)=1;
    M(i-1,j)=1;
    L2(k,:)=reshape(M,256,1)';
    k=k+1;
  end;
end;
L22=L2'*L2;
%
[U2,V2,X2,Lam2,Mu2]=gsvd(G,L2);

figure(6);
[rho,eta,reg_param]=l_curve_tikh_gsvd(U2,dn,X2,Lam2,Mu2,G,L2,1000,1e-10,60);
[alpha_corner,ireg_corner,kappa]=l_curve_corner(rho,eta,reg_param);
loglog(rho,eta)
hold on
loglog(rho(ireg_corner),eta(ireg_corner),'o')
hold off
xlabel('||Gm-d||')
ylabel('||Lm||')
title(['Second-Order Tikhonov L-Curve, Corner at: \alpha = ',num2str(alpha_corner)]);
%
% Examine the second-order solution
Gsharp2=inv(GTG+alpha_corner^2*L22)*G';
m2=Gsharp2*dn;
figure(7)
imagesc(Xa,Ya,reshape(m2,16,16),[3.0e-4 3.5e-4]);
colorbar
colormap(gray);
title(['Second-Order Tikhonov L-Curve Solution, Corner at: \alpha = ',num2str(alpha_corner)]);
ylabel('m')
xlabel('m')
print -deps mtik2a.eps
%
% That was awful! However, the misfit was tiny, the pseudoinverse calculation was very poorly conditioned,
% and the model amplitudes were huge, all indicating that the
% solution is under-regularized.  Try using the larger value
% of alpha at the second corner of the l-curve, near where ||Gm-d|| = 2e-6??
%
alpha_corner=interp1(rho,reg_param,2e-6);
eta_interp=interp1(rho,eta,2e-6);
%plot the second corner on the L-curve
figure(6)
hold on
loglog(2e-6,eta_interp,'x','markersize',20)
hold off
print -deps lcurve2.eps
%

Gsharp2=inv(GTG+alpha_corner^2*L22)*G';
m2=Gsharp2*dn;
figure(8);
imagesc(Xa,Ya,reshape(m2,16,16),[3.0e-4 3.5e-4]);
colorbar
colormap(gray);
title(['Second-Order Tikhonov L-Curve Solution, Corner at: \alpha = ',num2str(alpha_corner)]);
ylabel('m')
xlabel('m')
colorbar;
colormap(gray);
print -deps mtik2b.eps
%
% Look at the resolution of this solution (diagonal resolution elements)
```

```
%
R=Gsharp2*G;
figure(9);
imagesc(Xa,Ya,reshape(diag(R),16,16),[0 1]);
title('Second-Order Tikhonov L-Curve Solution Diagonal Resolution')
ylabel('m')
xlabel('m')
colorbar
colormap(gray);
print -deps res.eps
%
% Look at the resolution of a spike at (9,5)
%
spikem=zeros(16,16);
spikem(9,6)=1;
spike=reshape(spikem,256,1);
spiker=R*spike;
figure(10);
imagesc(Xa,Ya,reshape(spiker,16,16),[0 1]);
title('Second-Order Tikhonov LO-Curve Solution Spike Test (element 9,6)')
ylabel('m')
xlabel('m')
colorbar
colormap(gray);
print -deps res2.eps
%
% Look at the resolution of a spike at (1,8)
%
spikem=zeros(16,16);
spikem(1,8)=1;
spike=reshape(spikem,256,1);
spiker=R*spike;
figure(11);
imagesc(Xa,Ya,reshape(spiker,16,16),[0 1]);
title('Second-Order Tikhonov LO-Curve Solution Spike Test (element 1,8)')
ylabel('m')
xlabel('m')
colorbar
colormap(gray);
print -deps res3.eps
%
% Plot the true solution.
%
figure(12);
imagesc(Xa,Ya,mtruem,[3.0e-4 3.5e-4]);
colorbar;
colormap(gray);
title('True Solution')
ylabel('m')
xlabel('m')
print -deps mtrue.eps
```

4. The **G** matrix for this problem has 20 rows and 191 columns. Because we only have data constraints for 20 our of the 191 points in the function that we wish to determine, and rely on regularization to determine the other points, the matrix is an identity matrix for the 20 known ponts, and has zero columns corresponding to the remaining 171 points. The regularization matrix is a standard second-order roughing matrix. We can easily solve this regularized system by a variety of Tikhonov solution methods. Here we simply construct an augmented matrix $[\mathbf{G};\mathbf{L}]$ and solve for the 191-point model using the MATLAB backslash operator. A range of solutions for various values of α is shown in Figure 4.33. The recovered model is increasingly smooth in the seminorm sense of reduced $||\mathbf{L}m||$ as α is progressively increased. The corresponding tradeoff curve (which is essentially linear on a linear-linear plot) between smoothness and data misfit at the 20 given points is shown in Figure 4.34. The discrepancy principle guideline of fitting the 20 data points indicates that our target χ^2 value should be $\sqrt{20} \approx 4.47$. The actual value of the standard error-normalized misfit at the 20 data points is easily confirmed to be very close to this.

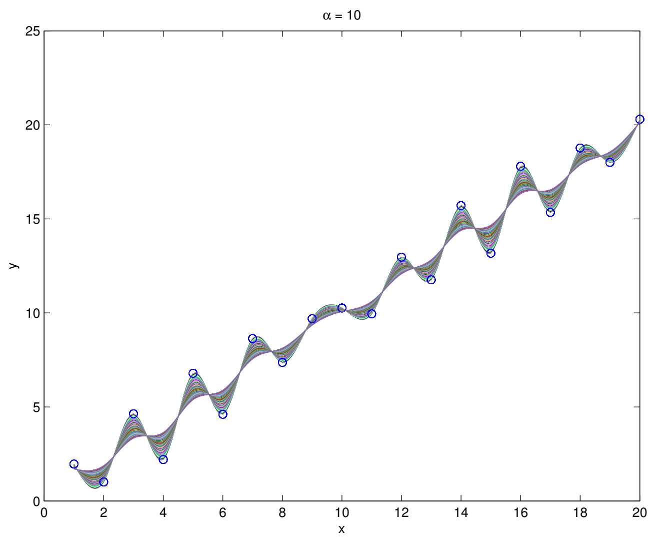

Figure 4.33: Second-order Tikhonov solutions for $0.2 \leq \alpha \leq 10$. Higher values of α produce progressively smoother models that fit the data less well (e.g., $\alpha = 10$ is the smoothest curve).

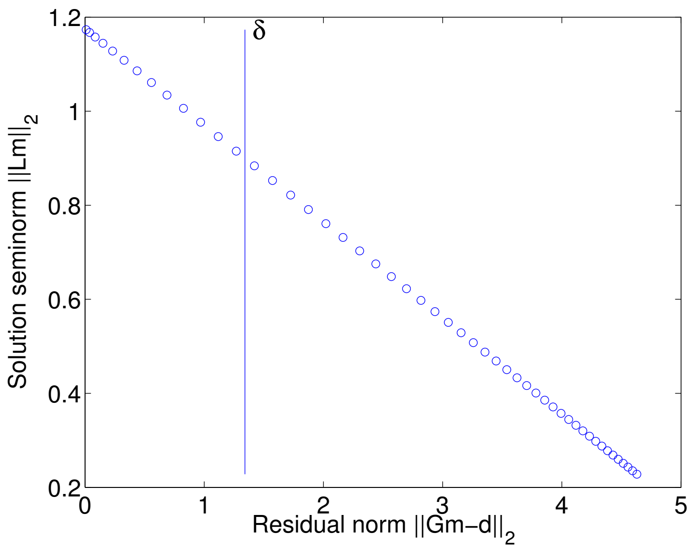

Figure 4.34: Second-order Tikhonov tradeoff curve for $0.2 \leq \alpha \leq 10$ with discrepancy principle residual norm indicated.

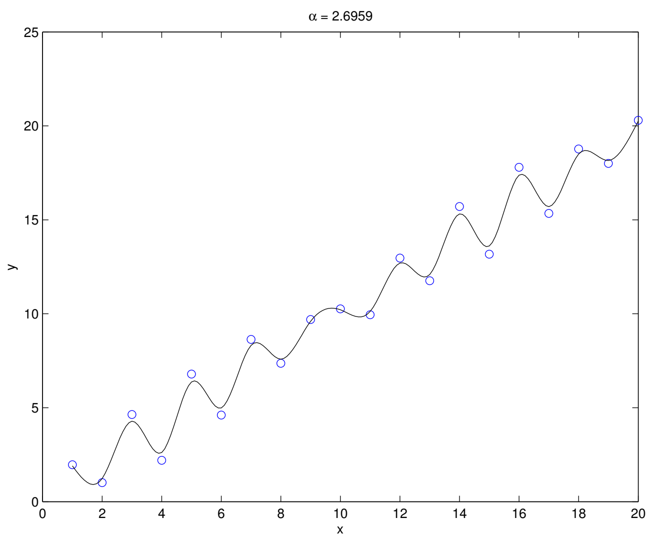

Figure 4.35: Second-order Tikhonov discrepancy principle model and observed data points ($\alpha \approx 2.7$).

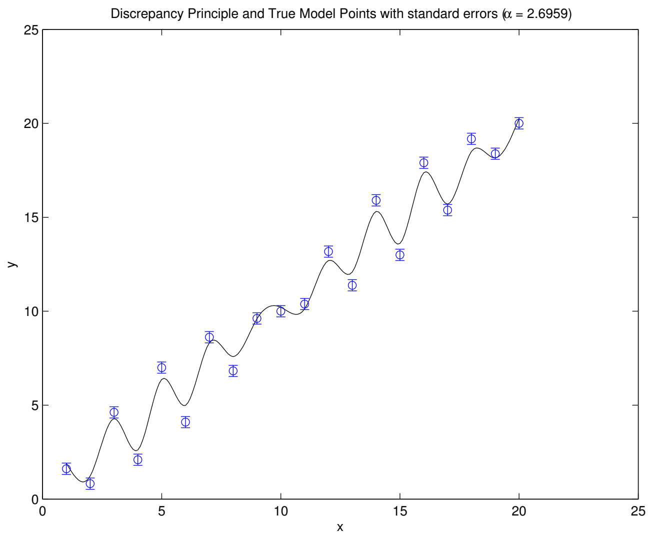

Figure 4.36: Second-order Tikhonov discrepancy principle model and true data points with standard errors.

```
%Tikhonov interpolation/Smoothing Exercise_4_4
clear
%Data for problem are generated here
randn('seed',0);
sigma=0.3;
M=20;
x=(1:M)';
y=2*sin(2*pi*0.45*x)+x;
noise=sigma*randn(size(y));
ynn=y;
y=y+noise;
%save interpdata.mat x y

%construct a 20 by 191 G matrix for the problem
%Every 10th column of G will have a 1 in the next element
xx=(1:0.1:M);
N=length(xx);
G=zeros(M,N);

for i=1:M
    j=find(xx==x(i));
    G(i,j)=1;
end

%second order Tikhonov regularization
torder=2;
L=get_l_rough(N,torder);
d=[y;zeros(N-torder,1)];
ialpha=1;

%examine a range of alpha values
alpha=0.2:.2:10;
for a=alpha
%construct an augmented G matrix
GG=[G;a*L];
%solve for m using the backslash operator
m(:,ialpha)=GG\d;
%calculate model seminorm, eta, and residual norm, rho
eta(ialpha)=norm(L*m(:,ialpha));
rho(ialpha)=norm(G*m(:,ialpha)-y);

%plot the data and model fit for each value of alpha (random colors)
figure(1)
c=rand(1,3);
plot(x,y,'o');
xlabel('x')
ylabel('y')
hold on
plot(xx,m(:,ialpha),'k','color',c)
title(['\alpha = ',num2str(alpha(ialpha))])
pause(0.1)
ialpha=ialpha+1;
end
hold off
print -depsc2 allmodels.eps

%plot the tradeoff curve
figure(2)
plot(rho,eta,'o')
xlabel('Residual norm ||Gm-d||_2')
ylabel('Solution seminorm ||Lm||_2')

%plot discrepancy principal value on the tradeoff curve
delta = sqrt(M)*sigma;
hold on
plot([delta,delta],[min(eta),max(eta)]);
text(delta,max(eta),' \delta','fontsize',20)
hold off
print -depsc2 tradoffcurve.eps

%Interpolate to find a solution closest to the discrepancy criterion
alpha_disc=interp1(rho,alpha,delta);
GG=[G;alpha_disc*L];
%solve for m using the backslash operator
m_disc=GG\d;
figure(3)
plot(x,y,'o');
xlabel('x')
ylabel('y')
hold on
plot(xx,m_disc,'k')
hold off
title(['\alpha = ',num2str(alpha_disc)])
print -depsc2 discmodel.eps

figure(4)
plot(xx,m_disc,'k')
title(['Discrepancy Principle and True Model Points with standard errors (\alpha = ',num2str(alpha_disc),')'])
xlabel('x')
ylabel('y')
hold on
```

```
errorbar(x,ynn,sigma*ones(size(x)),'o');
hold off
print -depsc2 tdiscmodel.eps

%misfit calculated at the 20 data points
m_test=m_disc(1:10:191);
display(['Chi-square measured at the 20 data points and predicted values: ',num2str(norm(y-m_test)/sigma)]);
display(['Chi-square target for uncorrelated data: ',num2str(sqrt(20))]);
```

5. Writing this as a least-squares problem gives:

$$\min \left\| \begin{bmatrix} \mathbf{G} \\ \alpha\mathbf{L} \end{bmatrix} \mathbf{m} - \begin{bmatrix} \mathbf{d} \\ \alpha\mathbf{L}\mathbf{m}_0 \end{bmatrix} \right\|_2^2 .$$

The normal equations for this least squares problem are

$$(\mathbf{G}^T\mathbf{G} + \alpha^2\mathbf{L}^T\mathbf{L})\mathbf{m} = \mathbf{G}^T\mathbf{d} + \alpha^2\mathbf{L}^T\mathbf{L}\mathbf{m}_0 \ .$$

These equations are not quite in a form that allows them to be solved by the GSVD. However, a bit of algebra fixes this. Subtracting $\mathbf{m}_0$ from $\mathbf{m}$ on the right-hand side and subtracting identical terms on the right hand side, we have

$$(\mathbf{G}^T\mathbf{G} + \alpha^2\mathbf{L}^T\mathbf{L})(\mathbf{m} - \mathbf{m}_0) = \mathbf{G}^T\mathbf{d} + \alpha^2\mathbf{L}^T\mathbf{L}\mathbf{m}_0 - \mathbf{G}^T\mathbf{G}\mathbf{m}_0 - \alpha^2\mathbf{L}^T\mathbf{L}\mathbf{m}_0$$

$$(\mathbf{G}^T\mathbf{G} + \alpha^2\mathbf{L}^T\mathbf{L})(\mathbf{m} - \mathbf{m}_0) = \mathbf{G}^T\mathbf{d} - \mathbf{G}^T\mathbf{G}\mathbf{m}_0$$

$$(\mathbf{G}^T\mathbf{G} + \alpha^2\mathbf{L}^T\mathbf{L})(\mathbf{m} - \mathbf{m}_0) = \mathbf{G}^T(\mathbf{d} - \mathbf{G}\mathbf{m}_0) \ .$$

This last system of equations is precisely in the form that we have previously solved using the GSVD or via other methods. The only difference is in the (given) right hand side. Once we've solved for the unknown $\mathbf{m}-\mathbf{m}_0$, we simply add $\mathbf{m}_0$ to obtain the desired model $\mathbf{m}$.