Chapter 7
Least Squares and Data Models

7.1 Introduction

In Chapter 6 the concept of interpolation played a major role in the computation of polynomial and spline models. The mathematical development in this chapter will eliminate the interpolation requirement in favor of a different criterion. Figure 7.1 shows six data points that could be modeled with an interpolating polynomial, $P_5(x)$; however, given the rapid increase in the values, $P_5(x)$ may not provide an acceptable approximation. A graph of the data suggests that an exponential model may be more appropriate than an interpolating polynomial.

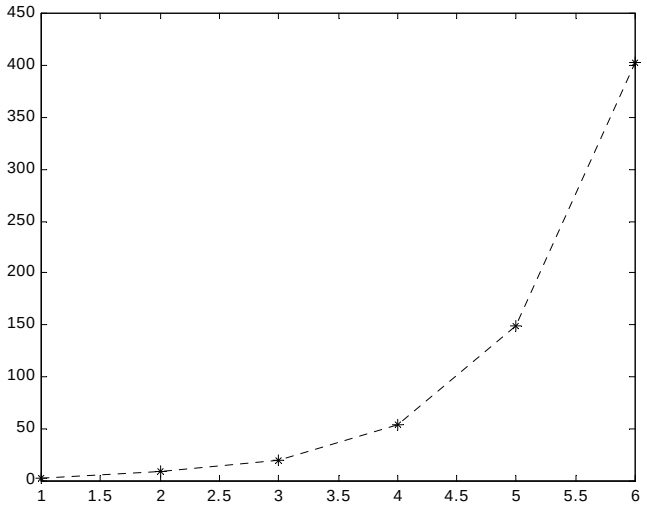

Figure 7.1 Possible Exponential Data

We begin by considering a set of data points (x_i, y_i) for $i = 1$ to n. Our goal is to construct an approximating function, say $F(x)$, to model the data. As before, $F(x)$ will contain unknown parameters or coefficients, $c_1, c_2, \ldots c_k$; however, in this case $F(x)$ need not interpolate the data. Typically, the number of parameters, k, will be much smaller than the number of data points, n. With this simple description, the possible structure of $F(x)$ is unlimited. To simplify our analysis, we assume that $F(x)$ has the linear form used in Chapter 6,

$$F(x) = c_1 g_1(x) + c_2 g_2(x) + c_3 g_3(x) + \ldots + c_k g_k(x), \tag{7.1}$$

where the $g_j(x)$'s are known linearly independent functions. Frequently, the choice of the $g_j(x)$'s is suggested by a knowledge of the problem under consideration. For example, we may suspect that there is a linear relationship between the independent and dependent variables represented by

the data. In which case a choice of $k = 2$ and $F(x) = c_1 x + c_2 1$ provides a reasonable model. This idea will be explored in Section 7.3.

Since interpolation is no longer required we should expect to find differences between $F(x_i)$ and y_i. The value of each difference, $d_i = F(x_i) - y_i$, will depend on the specified $g_j(x)$'s and the parameters, the c_j's. The n differences, denoted by d_i, are the key factors in determining the parameters. For example, it may be possible to minimize the maximum of $|d_i|$ or to minimize a sum that represents the total of the absolute values of the differences, $\sum_{i=1}^{n} |d_i|$. In each case positive numbers are involved.

7.2 The Least Squares Method

Gauss proposed that the parameters be computed by minimizing the sum of differences squared, $\sum_{i=1}^{n} d_i^2$. This choice eliminates the need for absolute values and, as we shall see, simplifies computations. The value of this sum will depend on the parameters in the model. In other words

$$E(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{n} \Big[F(x_i) - y_i\Big]^2. \tag{7.2}$$

The parameters are determined by the solution of a multiple variable calculus problem: minimize the sum of differences squared, $E(c_1, c_2, \ldots c_k)$, with respect to the parameters.

To simplify our efforts set $k = 2$ so that (7.2) becomes

$$E(c_1, c_2) = \sum_{i=1}^{n} \Big[F(x_i) - y_i\Big]^2 = \sum_{i=1}^{n} \Big[c_1 g_1(x_i) + c_2 g_2(x_i) - y_i\Big]^2. \tag{7.3}$$

From calculus, the values for c_1 and c_2 that minimize $E(c_1, c_2)$ are solutions of the following equations

$$\frac{\partial E(c_1, c_2)}{\partial \mathrm{c}_1} = 0 \quad \text{and} \quad \frac{\partial E(c_1, c_2)}{\partial \mathrm{c}_2} = 0. \tag{7.4}$$

Computing the partial derivatives yields

$$\frac{\partial E(c_1, c_2)}{\partial \mathrm{c}_1} = \sum_{i=1}^{n} 2\Big[c_1 g_1(x_i) + c_2 g_2(x_i) - y_i\Big] g_1(x_i) = 0$$

$$\frac{\partial E(c_1, c_2)}{\partial \mathrm{c}_2} = \sum_{i=1}^{n} 2\Big[c_1 g_1(x_i) + c_2 g_2(x_i) - y_i\Big] g_2(x_i) = 0$$

Distributing $g_1(x_i)$ and $g_2(x_i)$, using three sums in each equation, and rearranging terms leads to the so-called *normal* equations

$$c_1 \sum_{i=1}^{n} g_1(x_i)g_1(x_i) + c_2 \sum_{i=1}^{n} g_2(x_i)g_1(x_i) = \sum_{i=1}^{n} y_i g_1(x_i) \tag{7.5}$$

$$c_1 \sum_{i=1}^{n} g_1(x_i)g_2(x_i) + c_2 \sum_{i=1}^{n} g_2(x_i)g_2(x_i) = \sum_{i=1}^{n} y_i g_2(x_i) \tag{7.6}$$

The *symmetric* linear system in (7.5) and (7.6) is a direct result of the linear structure of $F(x)$ and Gauss' use of the differences squared. Other values for k will give comparable results. With $k = 3$, the normal equations are

$$c_1 \sum_{i=1}^{n} g_1(x_i)g_1(x_i) + c_2 \sum_{i=1}^{n} g_2(x_i)g_1(x_i) + c_3 \sum_{i=1}^{n} g_3(x_i)g_1(x_i) = \sum_{i=1}^{n} y_i g_1(x_i)$$

$$c_1 \sum_{i=1}^{n} g_1(x_i)g_2(x_i) + c_2 \sum_{i=1}^{n} g_2(x_i)g_2(x_i) + c_3 \sum_{i=1}^{n} g_3(x_i)g_2(x_i) = \sum_{i=1}^{n} y_i g_2(x_i) \tag{7.7}$$

$$c_1 \sum_{i=1}^{n} g_1(x_i)g_3(x_i) + c_2 \sum_{i=1}^{n} g_2(x_i)g_3(x_i) + c_3 \sum_{i=1}^{n} g_3(x_i)g_3(x_i) = \sum_{i=1}^{n} y_i g_3(x_i)$$

7.3 Least Squares Polynomials and MATLAB

The scatter plot shown in Figure 7.2 suggests a linear relationship; thus, a linear least squares model seems appropriate.

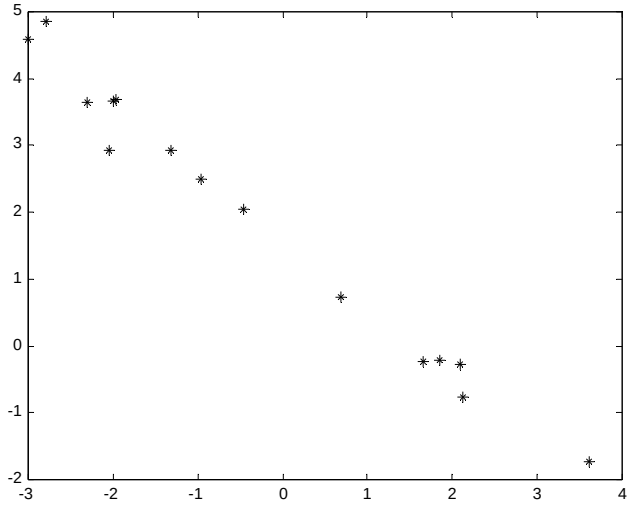

Figure 7.2 Scatter Plot of Data

A linear least squares model will have the form $F(x) = c_1 x + c_2 1$. In other words, $g_1(x) = x$ and $g_2(x) = 1$. Substituting into the normal equations (7.5) and (7.6) gives, in matrix form,

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{7.8}$$

Constructing the various sums in (7.8) and solving for c_1 and c_2 is a somewhat tedious task. MATLAB's **polyfit** will complete the task with one command. In Chapter 6 the third argument in **polyfit** represented the degree of an interpolating polynomial. To compute the coefficients of a linear least squares model the third argument must be set to one.

Assuming that the data shown in Figure 7.2 is contained in the vectors ***xd*** and ***yd***, the following MATLAB commands will compute the linear least squares coefficients, plot the data and also plot the least squares line. Note the use of **polyval** to find the y values of points on the line.

```
>> lscoef1 = polyfit(xd,yd,1)
lscoef1 =
 -9.8557e-001  1.5737e+000
>> plot(xd,yd,'k*',xd,polyval(lscoef1,xd),'k')
```

lscoef1 provides the data for the least squares line: $y = -0.98557x + 1.5737$.

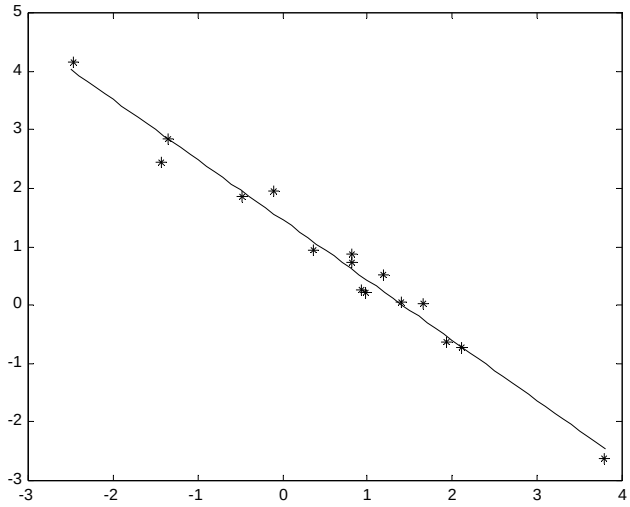

Figure 7.3 Data and Linear Least Squares Approximation

Recall that the least squares line has minimized the sum of differences squared. It is a simple task to compute this minimum. See >> **help sum**. Any other line close to the least squares line, for example $y = -1x + 1.5$, will produce a larger value for the sum of differences squared. Study the following commands carefully to see that equation (7.3) is being computed for two different lines: $y = -0.98557x + 1.5737$ and $y = -1x + 1.5$.

```
>> diffsq = sum((polyval(lscoef1,xd) - yd).^2)
diffsq =
  1.0298e+000
>> diffsq = sum((polyval([-1,1.5],xd) - yd).^2)
diffsq =
  1.1146e+000
```

Although the linear least squares approximation appears satisfactory, we might ask if there is a significant quadratic component in the data. A quadratic least squares model will have the form $F(x) = c_1x^2 + c_2x + c_3 1$.

In this special case, the matrix form of the normal equations (7.7) becomes

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i x_i^2 \\ \sum_{i=1}^{n} y_i x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{7.9}$$

Once again **polyfit** will compute the coefficients, leading to a quadratic least squares model for the data. In the quadratic case, the third argument is set equal to two.

```
>> lscoef2 = polyfit(xd,yd,2)
lscoef2 =
  2.9356e-002 -9.9002e-001  1.4440e+000
```

Although $F(x) = 0.029356x^2 - 0.99002x + 1.444$ does contain a small quadratic term the approximation is dominated by the linear term which is not too much different from the linear model determined previously. Graphical results, shown in Figure 7.4, show a minor difference.

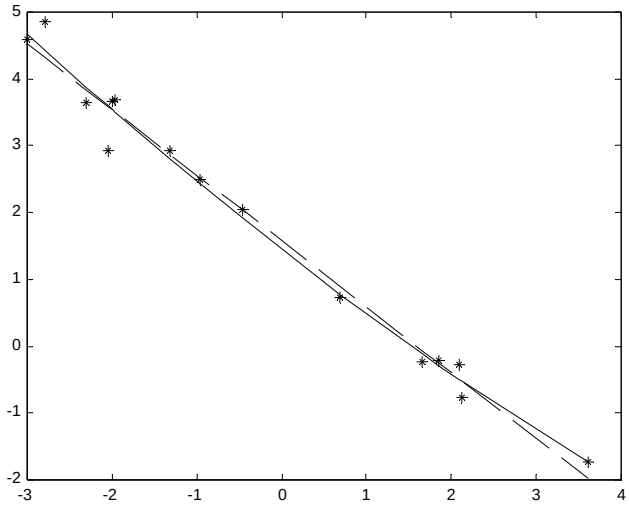

Figure 7.4 Linear (dashed) and Quadratic (solid) Models

In terms of differences squared, the least squares quadratic model, specified by the parabola $0.029356x^2 - 0.99002x + 1.444$, is slightly better that the linear model.

```
>> diffsq = sum((polyval(lscoef2,xd)-yd).^2)
diffsq =
  8.9035e-001
```

It is important to realize that **polyfit** is designed to compute the coefficients for polynomial least squares models. As we have seen, polynomial models allow the normal equations to take on a very simple form, such as (7.8) or (7.9). Higher order polynomial least squares models may be constructed; however, there is a source of difficulty. Depending on the data, terms in the coefficient matrix, for example $\sum_{i=1}^{n} x_i^4$ in (7.9), may become very large leading to an ill-conditioned matrix. Typically, polynomial least squares models are of low order.

7.4 Non polynomial Least Squares

In the previous section, the $g_j(x)$'s were selected as powers of x. As you may expect, other functions may be chosen to build least squares models. The normal equations will contain numerous terms of the form $\sum_{i=1}^{n} g_j(x_i)g_k(x_i)$ and $\sum_{i=1}^{n} y_i g_j(x_i)$. It is necessary to compute these sums in order to determine the coefficients in a least squares approximation.

Suppose we wish to build a three term least squares model in the standard form, $F(x) = c_1 g_1(x) + c_2 g_2(x) + c_3 g_3(x)$, for a given data set $(\, x_i,\, y_i\,)$ for $i = 1$ to n. As mentioned above, with $g_1(x)$, $g_2(x)$ and $g_3(x)$ specified, the normal equations may be constructed from the various sums; however, let's take a different approach. Although interpolation has been forgone in this chapter, assume for the moment that $F(x)$ must interpolate the data. In other words, $F(x_i) = y_i$. The result is n equations in 3 unknowns, each of the form

$$c_1 g_1(x_i) + c_2 g_2(x_i) + c_3 g_3(x_i) \;=\; y_i.$$

In matrix form the equations are

$$\begin{bmatrix} g_1(x_1) & g_2(x_1) & g_3(x_1) \\ g_1(x_2) & g_2(x_2) & g_3(x_2) \\ \vdots & \vdots & \vdots \\ g_1(x_n) & g_2(x_n) & g_3(x_n) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{7.10}$$

Equation (7.10), denoted $\boldsymbol{Gc = y}$, is an *over determined* system – more equations than unknowns. The matrix $\boldsymbol{G}$ is $n \times 3$. Multiplying both sides of (7.10) by $\boldsymbol{G}^T$, a $3 \times n$ matrix, will give the 3×3 system $\boldsymbol{G^T G c = G^T y}$. The matrix product on the left hand side,

$$\boldsymbol{G}^T\boldsymbol{G} = \begin{bmatrix} g_1(x_1) & g_2(x_1) & g_3(x_1) \\ g_1(x_2) & g_2(x_2) & g_3(x_2) \\ \vdots & \vdots & \vdots \\ g_1(x_{\mathrm{n}}) & g_2(x_{\mathrm{n}}) & g_3(x_{\mathrm{n}}) \end{bmatrix}^T \begin{bmatrix} g_1(x_1) & g_2(x_1) & g_3(x_1) \\ g_1(x_2) & g_2(x_2) & g_3(x_2) \\ \vdots & \vdots & \vdots \\ g_1(x_{\mathrm{n}}) & g_2(x_{\mathrm{n}}) & g_3(x_{\mathrm{n}}) \end{bmatrix}, \tag{7.11}$$

is a 3×3 matrix that is equal to the coefficient matrix in the normal system (7.7). In a similar fashion

$$\boldsymbol{G}^T\boldsymbol{Y} = \begin{bmatrix} g_1(x_1) & g_2(x_1) & g_3(x_1) \\ g_1(x_2) & g_2(x_2) & g_3(x_2) \\ \vdots & \vdots & \vdots \\ g_1(x_{\mathrm{n}}) & g_2(x_{\mathrm{n}}) & g_3(x_{\mathrm{n}}) \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{7.12}$$

is a 3×1 matrix that is equal to the right hand side of (7.7). In other words, $\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{c} = \boldsymbol{G}^T\boldsymbol{y}$ represents the standard 3×3 normal system for the computation of the least squares coefficients. It should be clear that the result holds in general. From a computational point of view, the matrix $\boldsymbol{G}$, along with the data vector $\boldsymbol{y}$ are the critical elements.

With the columns of $\boldsymbol{G}$ populated by the appropriate values, the MATLAB command **>>lscoef = (g'*g)\(g'*y)** will compute the least squares coefficients. Actually, in the case of an over determined system such as $\boldsymbol{G}\boldsymbol{c} = \boldsymbol{y}$, MATLAB recognizes the mismatch and defaults to the least squares solution. In other words, if we simply enter the over determined system, $\boldsymbol{G}\boldsymbol{c} = \boldsymbol{y}$, as **>> lscoef = g\y** MATLAB will, by default, compute the solution to **(g'*g)\(g'*y)**.

As an example, consider a least squares approximation to data using the model $F(x) = c_1\exp(-x) + c_2/x + c_3\sin(x)$. Assuming that the data is in the column vectors $\boldsymbol{xd}$ and $\boldsymbol{yd}$, our first task is to construct the matrix $\boldsymbol{G}$. For example, the first column of $\boldsymbol{G}$ should contain the values $g_1(xd_i) = \exp(-xd_i)$. The following MATLAB commands will construct the matrix $\boldsymbol{G}$ and solve for the coefficients. Note the use of the colon operator to fill all rows in a column with one MATLAB command.

```
>> xd = (1:.25:4)';
>> g(:,1) = exp(-xd);
>> g(:,2) = 1./xd;
>> g(:,3) = sin(xd);
>> c = g\yd
c =
 -1.7815e+002
  9.9946e+001
  2.0397e+002
```

The non polynomial model is $F(x) = -178.15\exp(-x) + 99.946/x + 203.97\sin(x)$. A graph of $F(x)$ along with the data is displayed in Figure 7.5 using the commands

```
>> xi=(1:.1:4)';
>> yi=c(1)*exp(-xi)+c(2)./xi+c(3)*sin(xi);
>> plot(xd,yd,'k*',xi,yi,'k')
```

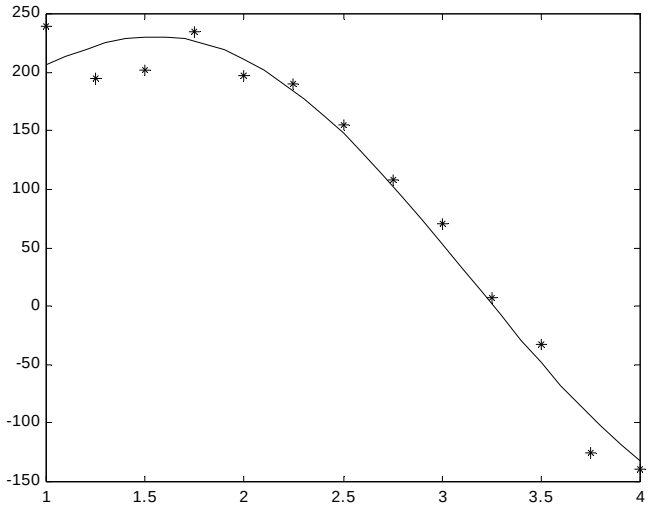

Figure 7.5 Non Polynomial Least Squares Model

In the preceding example, the vector ***yd*** was obtained by adding normally distributed random numbers to the function $y = -10e^{-x} + 50/x + 200\sin(x)$. MATLAB's **randn** was used to generate the random numbers producing ***yd*** = ***y*** + 20∗**randn(size(xd))**. You may wish to determine how well the least squares model for ***yd*** approximates the data determined by ***y*** itself. Since **randn** is random, your graph will not be identical to Figure 7.5.

7.5 Nonlinear Models

In many cases the structure of our approximating function, (7.1), wherein the coefficients and functions appear in a linear manner, is not appropriate. A knowledge of the phenomenon or system producing the data may indicate that an exponential model, $F(x) = \beta\, e^{\alpha x}$, should be constructed. Other possible simple models are power functions, βx^{α}, rational functions, $1/(\alpha x + \beta)$, or log functions, $\beta \ln(\alpha x)$.

Exponential Functions

Suppose that the data is from an exponential process, $y = \beta\, e^{\alpha x}$. Since $\ln(y) = \alpha x + \ln(\beta)$, plotting $\ln(y)$ versus x, using a linear scale on both axes, will result in the graph of a line. As an alternative, plotting y versus x, with a logarithmic scale on the vertical axis, will also give a line. MATLAB contains numerous plot options including **semilogy**. See **>> help semilogy** for further information and other plot options. Since the exponential model can be written as $\log_{10}(y) = \alpha x \log_{10}(e) + \log_{10}(\beta)$ the use of base 10 logarithms by **semilogy** poses no problems. To illustrate these ideas consider the exponential function $y = 1.5e^{-2.3x}$ on the interval $[\ 0,\ 3\ ]$.

Figure 7-6 shows two subplots for the exponential function. To aid our understanding of these unfamiliar plots, note that when $x = 1$, $y = 0.1504$, and $\ln(y) = -1.89$. The points $(1, -1.89)$ and (1, $1.504 \cdot 10^{-1}$) are plotted on the graphs for reference. You should be able to verify that the points corresponding to $x = 0$ are at the proper locations. The MATLAB

command **ylabel** prints a label on the vertical axis. See also >> **help xlabel**. Recall that MATLAB uses **log** for ln and **log10** for $\log_{10}$.

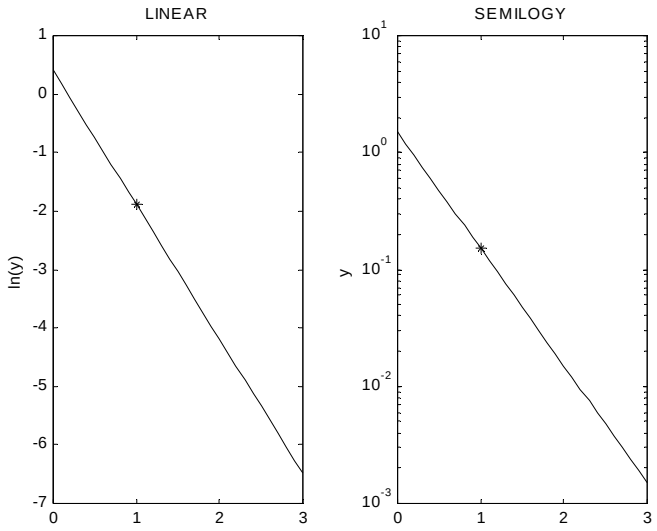

Figure 7.6 The Exponential Function $1.5e^{-2.3x}$

The scatter plot of a data set from an exponential process should resemble a line if the data is plotted with a logarithmic scale on the vertical axis or if the vertical scale is $\ln(y)$. Consider the following data

x	0.6	1.2	2.5	3.2	4.2	5.7
y	4.7	6.9	19.3	30.1	61.2	170.5

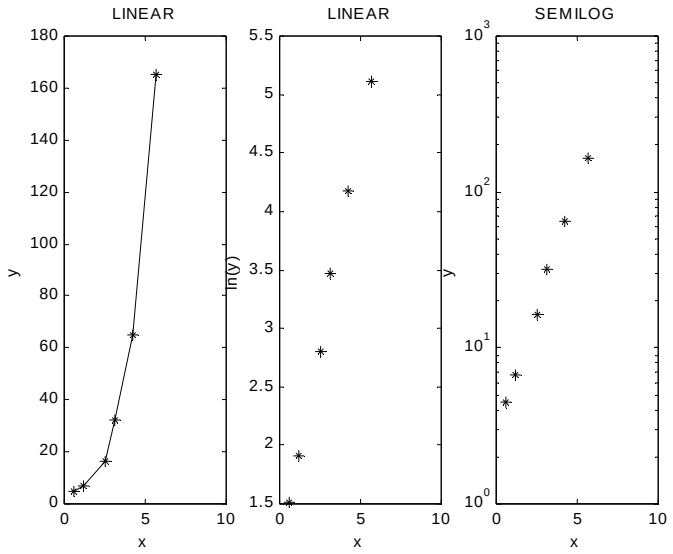

Figure 7.7 Exponential Data

Figure 7.7 shows three plots of the data using the following MATLAB commands.

```
>> subplot(131),plot(x,y,'k*',x,y,'k'),title('LINEAR'),xlabel('x'),ylabel('y')
>> subplot(132),plot(x,log(y),'k*'),title('LINEAR'),xlabel('x'),ylabel('ln(y)')
>> subplot(133),semilogy(x,y,'k*'),title('SEMILOG'),xlabel('x'),ylabel('y')
```

Subplot(131) suggests exponential growth in the data. Subplots(132) and (133) appear to be linear supporting the exponential conjecture. Since the exponential model, $y = \beta\, e^{\alpha x}$, may be restructured as $\ln(y) = \alpha x + \ln(\beta)$, Gauss' least squares method may be used to compute the parameters, α and $\ln(\beta)$, for the line. The normal equation, (7.8), may be modified to determine α and $\ln(\beta)$ as follows:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \ln(\beta) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \ln(y_i) x_i \\ \sum_{i=1}^{n} \ln(y_i) \end{bmatrix} \tag{7.13}$$

MATLAB will compute the linear least squares coefficients, α and $\ln(\beta)$, using

```
>> lscoef = polyfit(x,log(y),1)
lscoef =
  7.2084e-001  1.0824e+000
```

Note that **lscoef(1)** $= \alpha$ and **lscoef(2)** $= \ln(\beta)$. The numerical value of β is

```
>> beta = exp(lscoef(2))
beta =
  2.9517e+000
```

Using **lscoef(1)** and **beta** our exponential model for the data is $y = \beta\, e^{\alpha x} = 2.9517\, e^{0.72084x}$.

Figure 7-8 is a graph of the data and least squares exponential model.

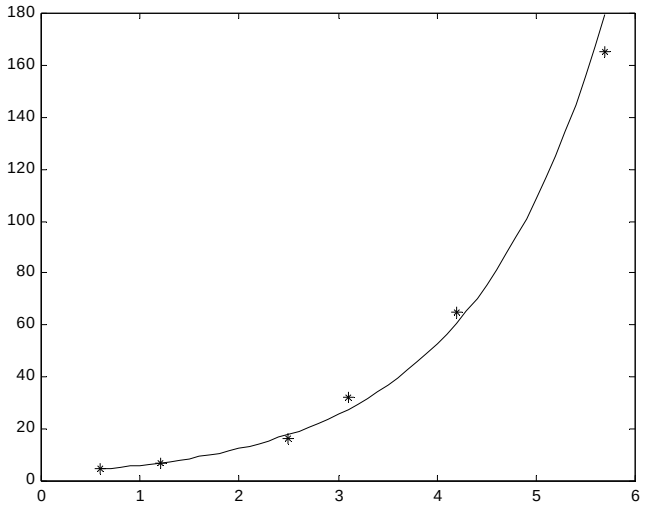

Figure 7.8 Exponential Data and Model

It is tempting to use **polyval** to compute values for y_i. The MATLAB command **polyval(lscoef,xi)** will actually evaluate the linear term $\alpha x_i + \ln(\beta)$. In other words, values of the logarithm of y_i, not y_i itself, will be computed. The exponential function may be used to produce the correct values for y_i with the command **exp(polyval(lscoef,xi))**.

Power Functions

As a second example, consider a power function model $y = \beta x^{\alpha}$. Taking logarithms of both sides gives $\ln(y) = \alpha \ln(x) + \ln(\beta)$. Plotting $\ln(y)$ versus $\ln(x)$, using a linear scale on both axes, will produce a line. Clearly, plotting y versus x, using logarithmic scales for both the vertical and horizontal axes, will also result in a line. The MATLAB plot command **loglog** will implement the second approach. The normal equations for this model are

$$\begin{bmatrix} \sum_{i=1}^{n} \ln^2(x_i) & \sum_{i=1}^{n} \ln(x_i) \\ \sum_{i=1}^{n} \ln(x_i) & \sum_{i=1}^{n} 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \ln(\beta) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \ln(y_i)\ln(x_i) \\ \sum_{i=1}^{n} \ln(y_i) \end{bmatrix}. \tag{7.14}$$

The following MATLAB commands will plot the power function $y = 6.6x^{0.23}$ on the interval $[\,0.2, 7.7\,]$ in two different formats.

```
>> x = (.2:.5:7.7)';
>> y = 6.6*x.^(.23);
>> subplot(121),plot(log(x),log(y),'k*'),title('LINEAR'),xlabel('ln(x)'),ylabel('ln(y)')
>> subplot(122),loglog(x,y,'k*'),title('LOGLOG'),xlabel('x'),ylabel('y')
```

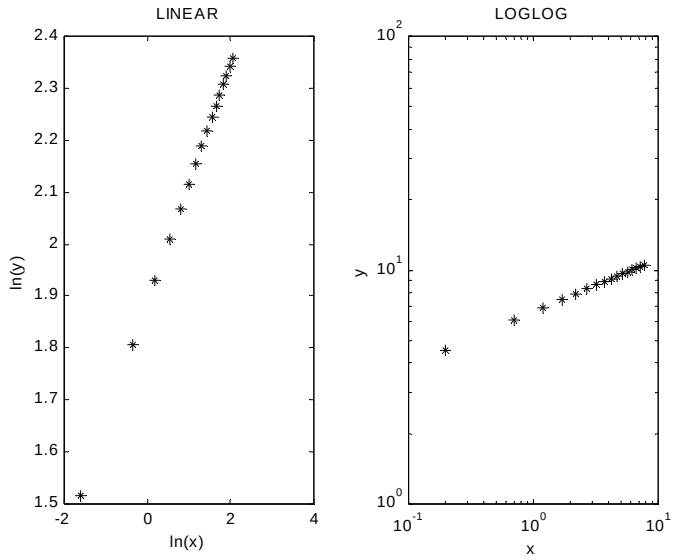

Figure 7.9 Plots of $y = 6.6x^{0.23}$ on $[\,0.2, 7.7\,]$

Suppose that the values of $y = 6.6x^{0.23}$ have been perturbed by random values producing a data set (x_i, z_i). We may construct a least squares approximation, $\ln(z) = \alpha\ln(x) + \ln(\beta)$, using **polyfit** on logarithms of the data.

```
>> z = y + .3*randn(size(x)); % perturbed power function data
>> lscoef = polyfit(log(x),log(z),1)
lscoef =
  2.4760e-001  1.8692e+000
>> lnz = polyval(lscoef,log(x));
>> subplot(121),plot(x,z,'k*',x,exp(lnz),'k'),title('LINEAR'),xlabel('x'),ylabel('z')
>> subplot(122),loglog(x,z,'k*',x,exp(lnz),'k'),title('LOGLOG'),xlabel('x'), ...
 ylabel('z')
```

lscoef(2) is actually $\ln(\beta)$; thus $\beta = e^{1.8692} = 6.4831$ and the least squares model is given by $z = 6.4831x^{0.2476}$. Since **polyval** has been used to compute **lnz**, the actual values of z are given by **exp(lnz)**. Figure 7.10 displays the perturbed data and the least squares approximation in two different formats.

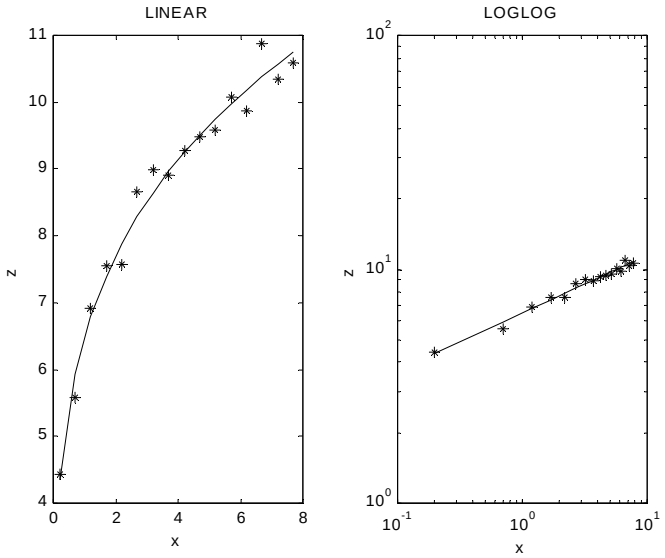

Figure 7.10 Least Squares Power Function Approximation

Rational Functions

At first thought it does not seem possible that the least squares method could be applied to simple rational functions such as $y = 1/(\alpha x + \beta)$ or $y = 1/(\alpha x^2 + \beta x + \gamma)$. If y is never equal to zero, both functions may be studied by inverting the expressions: $1/y = \alpha x + \beta$ or $1/y = \alpha x^2 + \beta x + \gamma$. In either case **polyfit** may be used to compute the coefficients. In will be necessary to enter the data values for $\boldsymbol{y}$ using array division. The commands **polyfit(x,1./y,1)** or **polyfit(x,1./y,2)** will compute the least squares coefficients for a linear or quadratic model.

If the coefficients computed by **polyfit** are used in **polyval** it is important to realize that, given x_i, the reciprocal $1/y_i$ will be calculated. In other words, an additional inversion (array division) will be needed to determine the predicted model value for y_i.

7.6 Problems

7-1. The *root-mean-square* error, or RMS error, is defined by $\sqrt{\frac{1}{n}\sum_{i=1}^{n}[\,F(x_i)-y_i]^2}$.
Find the equations that will minimize the RMS error if $F(x) = c_1 g_1(x) + c_2 g_2(x)$.

7-2. Consider the data $(\,0,\,1\,),(\,1,\,1\,),\,(\,2,\,2\,)$ and $(\,4,\,5\,)$ and a linear least squares approximation, $y = c_1 x + c_2$.
a. Find the explicit form for the error term $E(c_1,\,c_2) = \sum_{i=1}^{n}\Big[c_1 x_i + c_2 - y_i\Big]^2$.
b. Differentiate your $E(c_1,\,c_2)$ from part a to determine two linear equations whose solution minimizes $E(c_1,\,c_2)$.
c. How do your equations compare to the normal equations given in equation (7.8)?
d. Solve the system in part b and graph the data and least squares line.

7-3. Repeat Problem 7-2 using the data $(0,-1),(1,1),(2,1)$ and $(4,0)$.

7-4. Recall the spring data from Chapter 6.
The force needed to stretch a spring from its normal length is given in the table.

$x\,(cm)$	3.0	3.5	4.0	4.5	5.0	6.0	7.0	8.0
$F\,(dynes)$	2.37	8.25	14.10	20.00	22.45	24.82	26.27	28.61

Using MATLAB,
a. Construct a linear least squares model for the data.
b. Plot the model and spring data.
c. Estimate the spring constant.

7-5. Recall the corrected thermocouple data from Chapter 6.

$T\,(°F)$	45	50	55	60	70	80	90	100
$V\,(mV)$	3.49	3.17	2.93	2.73	2.37	2.08	1.85	1.65

Using MATLAB,
a. Construct a quadratic least squares model for the data.
b. Plot the model and thermocouple data.
c. Approximate $V(T)$ and dV/dT at $T = 66$ and 76.

7-6. Repeat Problem 7-5 using an exponential least squares model.

7-7. In matrix form, write out the normal equations for a least squares model using $F(x) = c_1\exp(-x) + c_2/x + c_3\sin(x)$.

7-8. Construct normal equations for the least squares method that will allow you to determine the parameters, a and b, in the model $y = \sqrt{(ax^2 + be^{-x})}$.

7-9. Given the following data

>> x = (.4:.2:1.8)';

>> y = [.189,.202,.265,.300,.345,.563,.691,1.249]';

a. Construct four least squares models: quadratic, exponential, power and reciprocal, $y = 1/(mx + b)$, for the data.

b. Which model appears to provide the best approximation. Explain your reasoning. Note: The sum of differences squared, (7.2), is one way to compare various models.

7-10. Suppose that we have a data set, (t_i, y_i) for $i = 1$ to n, that represents the numerical solution of an initial value problem in differential equations. The solution of the initial value problem is thought to be of the form $y(t) = (at^2 + bt + c)^{-1}$. Show how MATLAB may be used to compute the least squares values for the parameters.

7-11. Given the following data

x	-1	-0.5	0	$.5$	1	1.5	2
y	$.080$	$.156$	$.537$	$.838$	$.936$	$.817$	$.267$

Determine the coefficients in a least squares model, $y = \exp(ax^2 + bx + c)$, for the data. Plot the data and model results.

7-12. Consider the data

x	$.273$	$.450$	$.106$	$.706$	$.368$	$.142$
y	$-.546$	$.531$	-2.53	1.22	$.018$	-1.97

a. Plot the data

b. Is an exponential model a reasonable choice? Explain.

c. Determine a least squares model in the form $y = b\ln(ax)$.

d. Plot the data and model results.

7-13. Given the data

x	1.0	1.4	2.1	2.6	3.7	4.6	5.2	6.8
y	-4.71	-3.53	-1.88	-1.31	0.11	0.18	0.97	1.89

Choose two likely models for the data and compute the coefficients. Plot graphs of your results and explain why you selected the particular models. Is one better than the other? Explain why. Perhaps MATLAB has the answer. Try **>> why**.