

CHAPTER 15

15.1 The data can be tabulated and the sums computed as

| i | x | y | x^2 | x^3 | x^4 | xy | x^2y |
|----------|-----|------|-------|---------|----------|--------|----------|
| 1 | 10 | 25 | 100 | 1000 | 10000 | 250 | 2500 |
| 2 | 20 | 70 | 400 | 8000 | 160000 | 1400 | 28000 |
| 3 | 30 | 380 | 900 | 27000 | 810000 | 11400 | 342000 |
| 4 | 40 | 550 | 1600 | 64000 | 2560000 | 22000 | 880000 |
| 5 | 50 | 610 | 2500 | 125000 | 6250000 | 30500 | 1525000 |
| 6 | 60 | 1220 | 3600 | 216000 | 12960000 | 73200 | 4392000 |
| 7 | 70 | 830 | 4900 | 343000 | 24010000 | 58100 | 4067000 |
| 8 | 80 | 1450 | 6400 | 512000 | 40960000 | 116000 | 9280000 |
| Σ | 360 | 5135 | 20400 | 1296000 | 87720000 | 312850 | 20516500 |

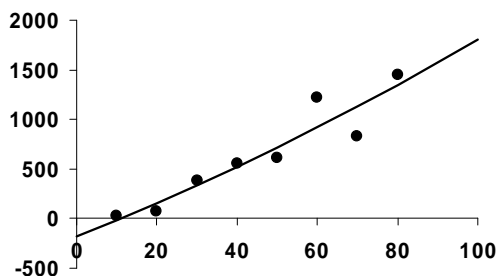
Normal equations:

$$\begin{bmatrix} 8 & 360 & 20400 \\ 360 & 20400 & 1296000 \\ 20400 & 1296000 & 87720000 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 5135 \\ 312850 \\ 20516500 \end{bmatrix}$$

which can be solved for the coefficients yielding the following best-fit polynomial

$$F = -178.4821 + 16.12202 v + 0.037202 v^2$$

Here is the resulting fit:



The predicted values can be used to determine the sum of the squares. Note that the mean of the y values is 641.875.

| i | x | y | y_{pred} | $(y_i - \bar{y})^2$ | $(y - y_{\text{pred}})^2$ |
|----------|-----|------|-------------------|---------------------|---------------------------|
| 1 | 10 | 25 | -13.5417 | 380535 | 1485 |
| 2 | 20 | 70 | 158.8393 | 327041 | 7892 |
| 3 | 30 | 380 | 338.6607 | 68579 | 1709 |
| 4 | 40 | 550 | 525.9226 | 8441 | 580 |
| 5 | 50 | 610 | 720.625 | 1016 | 12238 |
| 6 | 60 | 1220 | 922.7679 | 334229 | 88347 |
| 7 | 70 | 830 | 1132.351 | 35391 | 91416 |
| 8 | 80 | 1450 | 1349.375 | 653066 | 10125 |
| Σ | | | | 1808297 | 213793 |

The coefficient of determination can be computed as

$$r^2 = \frac{1808297 - 213793}{1808297} = 0.88177$$

The model fits the trend of the data nicely, but it has the deficiency that it yields physically unrealistic negative forces at low velocities.

15.2 The sum of the squares of the residuals for this case can be written as

$$S_r = \sum_{i=1}^n (y_i - a_1 x_i - a_2 x_i^2)^2$$

The partial derivatives of this function with respect to the unknown parameters can be determined as

$$\frac{\partial S_r}{\partial a_1} = -2 \sum [(y_i - a_1 x_i - a_2 x_i^2) x_i]$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum [(y_i - a_1 x_i - a_2 x_i^2) x_i^2]$$

Setting the derivative equal to zero and evaluating the summations gives

$$(\sum x_i^2) a_1 + (\sum x_i^3) a_2 = \sum x_i y_i$$

$$(\sum x_i^3) a_1 + (\sum x_i^4) a_2 = \sum x_i^2 y_i$$

which can be solved for

$$a_1 = \frac{\sum x_i y_i \sum x_i^4 - \sum x_i^2 y_i \sum x_i^3}{\sum x_i^2 \sum x_i^4 - (\sum x_i^3)^2}$$

$$a_2 = \frac{\sum x_i^2 \sum x_i^2 y_i - \sum x_i y_i \sum x_i^3}{\sum x_i^2 \sum x_i^4 - (\sum x_i^3)^2}$$

The model can be tested for the data from Table 14.1.

| x | y | x^2 | x^3 | x^4 | xy | x^2y |
|----------|------|-------|---------|----------|--------|----------|
| 10 | 25 | 100 | 1000 | 10000 | 250 | 2500 |
| 20 | 70 | 400 | 8000 | 160000 | 1400 | 28000 |
| 30 | 380 | 900 | 27000 | 810000 | 11400 | 342000 |
| 40 | 550 | 1600 | 64000 | 2560000 | 22000 | 880000 |
| 50 | 610 | 2500 | 125000 | 6250000 | 30500 | 1525000 |
| 60 | 1220 | 3600 | 216000 | 12960000 | 73200 | 4392000 |
| 70 | 830 | 4900 | 343000 | 24010000 | 58100 | 4067000 |
| 80 | 1450 | 6400 | 512000 | 40960000 | 116000 | 9280000 |
| Σ | | 20400 | 1296000 | 87720000 | 312850 | 20516500 |

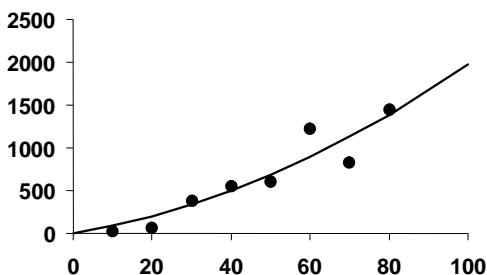
$$a_1 = \frac{312850(87720000) - 20516500(1296000)}{20400(87720000) - (1296000)^2} = 7.771024$$

$$a_2 = \frac{20400(20516500) - 312850(1296000)}{20400(87720000) - (1296000)^2} = 0.119075$$

Therefore, the best-fit model is

$$y = 7.771024x + 0.119075x^2$$

The fit, along with the original data can be plotted as



15.3 The data can be tabulated and the sums computed as

| i | x | y | x^2 | x^3 | x^4 | x^5 | x^6 | xy | x^2y | x^3y |
|----------|-----|------|-------|-------|-------|--------|---------|-------|--------|--------|
| 1 | 3 | 1.6 | 9 | 27 | 81 | 243 | 729 | 4.8 | 14.4 | 43.2 |
| 2 | 4 | 3.6 | 16 | 64 | 256 | 1024 | 4096 | 14.4 | 57.6 | 230.4 |
| 3 | 5 | 4.4 | 25 | 125 | 625 | 3125 | 15625 | 22 | 110 | 550 |
| 4 | 7 | 3.4 | 49 | 343 | 2401 | 16807 | 117649 | 23.8 | 166.6 | 1166.2 |
| 5 | 8 | 2.2 | 64 | 512 | 4096 | 32768 | 262144 | 17.6 | 140.8 | 1126.4 |
| 6 | 9 | 2.8 | 81 | 729 | 6561 | 59049 | 531441 | 25.2 | 226.8 | 2041.2 |
| 7 | 11 | 3.8 | 121 | 1331 | 14641 | 161051 | 1771561 | 41.8 | 459.8 | 5057.8 |
| 8 | 12 | 4.6 | 144 | 1728 | 20736 | 248832 | 2985984 | 55.2 | 662.4 | 7948.8 |
| Σ | 59 | 26.4 | 509 | 4859 | 49397 | 522899 | 5689229 | 204.8 | 1838.4 | 18164 |

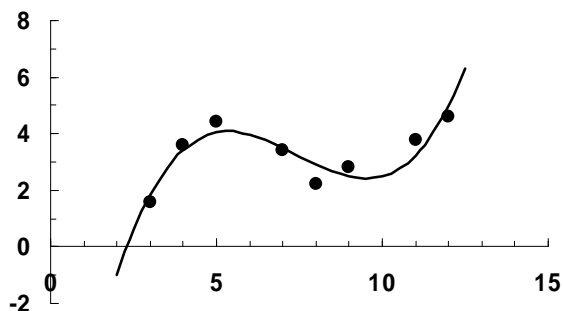
Normal equations:

$$\begin{bmatrix} 8 & 59 & 509 & 4859 \\ 59 & 509 & 4859 & 49397 \\ 509 & 4859 & 49397 & 522899 \\ 4859 & 49397 & 522899 & 5689229 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} 26.4 \\ 204.8 \\ 1838.4 \\ 18164 \end{Bmatrix}$$

which can be solved for the coefficients yielding the following best-fit polynomial

$$y = -11.4887 + 7.143817x - 1.04121x^2 + 0.046676x^3$$

Here is the resulting fit:



The predicted values can be used to determine the sum of the squares. Note that the mean of the y values is 3.3.

| i | x | y | y_{pred} | $(y_i - \bar{y})^2$ | $(y - y_{\text{pred}})^2$ |
|----------|-----|-----|-------------------|---------------------|---------------------------|
| 1 | 3 | 1.6 | 1.83213 | 2.8900 | 0.0539 |
| 2 | 4 | 3.6 | 3.41452 | 0.0900 | 0.0344 |
| 3 | 5 | 4.4 | 4.03471 | 1.2100 | 0.1334 |
| 4 | 7 | 3.4 | 3.50875 | 0.0100 | 0.0118 |
| 5 | 8 | 2.2 | 2.92271 | 1.2100 | 0.5223 |
| 6 | 9 | 2.8 | 2.4947 | 0.2500 | 0.0932 |
| 7 | 11 | 3.8 | 3.23302 | 0.2500 | 0.3215 |
| 8 | 12 | 4.6 | 4.95946 | 1.6900 | 0.1292 |
| Σ | | | | 7.6000 | 1.2997 |

The coefficient of determination can be computed as

$$r^2 = \frac{7.6 - 1.2997}{7.6} = 0.829$$

Note that the above solution can also be easily obtained with the following MATLAB script:

```
clear,clc,format short g
x=[3 4 5 7 8 9 11 12]';
y=[1.6 3.6 4.4 3.4 2.2 2.8 3.8 4.6]';
Z = [ones(size(x)) x x.^2 x.^3];
a = (Z'*Z)\(Z'*y)
Sr = sum((y-Z*a).^2)
r2 = 1-Sr/sum((y-mean(y)).^2)
```

Running this script yields:

```
a =
   -11.489
    7.1438
   -1.0412
    0.046676
Sr =
    1.2997
r2 =
    0.82898
```

15.4

```
function p = polyreg(x,y,m)
% polyreg(x,y,m):
%   Polynomial regression.
% input:
%   x = independent variable
%   y = dependent variable
%   m = order of polynomial
% output:
%   p = vector of coefficients

n = length(x);
if length(y)~=n, error('x and y must be same length'); end
for i = 1:m+1
    for j = 1:i
        k = i+j-2;
        s = 0;
```

```

    for l = 1:n
        s = s + x(l)^k;
    end
    A(i,j) = s;
    A(j,i) = s;
end
s = 0;
for l = 1:n
    s = s + y(l)*x(l)^(i-1);
end
b(i) = s;
end
p = A\b';

```

Script to test by solving Prob. 15.3:

```

clear,clc,format short g
x = [3 4 5 7 8 9 11 12];
y = [1.6 3.6 4.4 3.4 2.2 2.8 3.8 4.6];
polyreg(x,y,3)

```

```

ans =

    -11.489
     7.1438
    -1.0412
     0.046676

```

15.5 Because the data is curved, a linear regression will undoubtedly have too much error. Therefore, as a first try, fit a parabola,

```

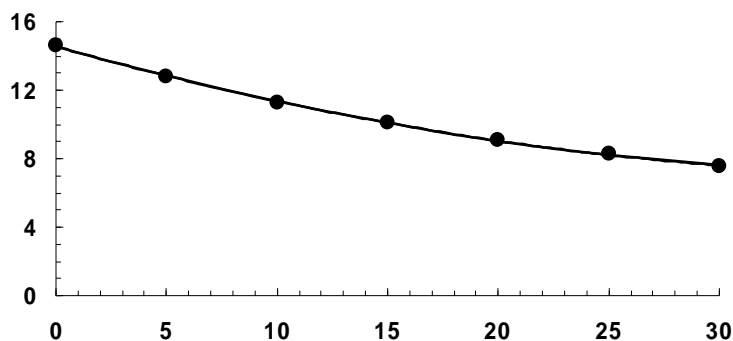
>> clear,clc,format long
>> T = [0 5 10 15 20 25 30];
>> c = [14.6 12.8 11.3 10.1 9.09 8.26 7.56];
>> p = polyfit(T,c,2)

p =
    0.00439523809524   -0.36335714285714   14.55190476190477

```

Thus, the best-fit parabola would be

$$c = 14.55190476 - 0.36335714T + 0.0043952381T^2$$



We can use this equation to generate predictions corresponding to the data. When these values are rounded to the same number of significant digits the results are

| T | c-data | c-pred | rounded |
|-----|--------|----------|---------|
| 0 | 14.6 | 14.55190 | 14.6 |
| 5 | 12.8 | 12.84500 | 12.8 |
| 10 | 11.3 | 11.35786 | 11.4 |
| 15 | 10.1 | 10.09048 | 10.1 |
| 20 | 9.09 | 9.04286 | 9.04 |
| 25 | 8.26 | 8.21500 | 8.22 |
| 30 | 7.56 | 7.60690 | 7.61 |

Thus, although the plot looks good, discrepancies occur in the third significant digit. We can, therefore, fit a third-order polynomial

```
>> p = polyfit(T,c,3)
p =
-0.00006444444444    0.00729523809524   -0.39557936507936   14.60023809523810
```

Thus, the best-fit cubic would be

$$c = 14.600238095 - 0.395579365T + 0.007295238T^2 - 0.000064444T^3$$

We can use this equation to generate predictions corresponding to the data. When these values are rounded to the same number of significant digits the results are

| T | c-data | c-pred | rounded |
|-----|--------|----------|---------|
| 0 | 14.6 | 14.60020 | 14.6 |
| 5 | 12.8 | 12.79663 | 12.8 |
| 10 | 11.3 | 11.30949 | 11.3 |
| 15 | 10.1 | 10.09044 | 10.1 |
| 20 | 9.09 | 9.09116 | 9.09 |
| 25 | 8.26 | 8.26331 | 8.26 |
| 30 | 7.56 | 7.55855 | 7.56 |

Thus, the predictions and data agree to three significant digits.

15.6 The multiple linear regression model to evaluate is

$$o = a_0 + a_1T + a_2c$$

The $[Z]$ and y matrices can be set up using MATLAB commands in a fashion similar to Example 15.4,

```
>> format long
>> t = [0 5 10 15 20 25 30];
>> T = [t t t]';
>> c = [zeros(size(t)) 10*ones(size(t)) 20*ones(size(t))]' ;
>> Z = [ones(size(T)) T c];
>> y = [14.6 12.8 11.3 10.1 9.09 8.26 7.56 12.9 11.3 10.1 9.03 8.17 7.46 6.85
11.4 10.3 8.96 8.08 7.35 6.73 6.2]';
```

The coefficients can be evaluated as

```
>> a = Z\y
a =
13.52214285714286
-0.20123809523810
-0.10492857142857
```

Thus, the best-fit multiple regression model is

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

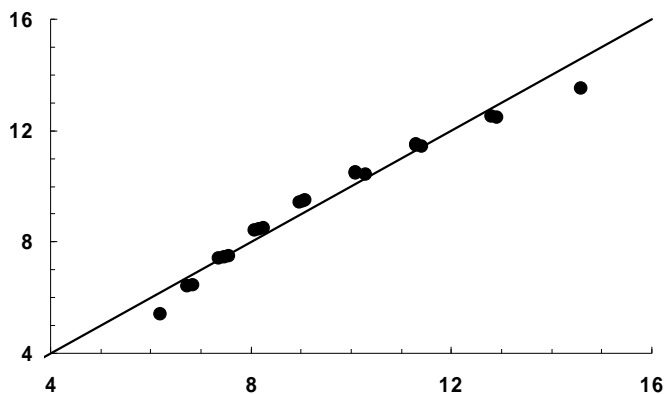
$$o = 13.52214285714286 - 0.20123809523810T - 0.10492857142857c$$

We can evaluate the prediction at $T = 12$ and $c = 15$ and evaluate the percent relative error as

```
>> cp = a(1)+a(2)*12+a(3)*15
cp =
    9.53335714285714

>> ea = abs((9.09-cp)/9.09)*100
ea =
    4.87741631305987
```

Thus, the error is considerable. This can be seen even better by generating predictions for all the data and then generating a plot of the predictions versus the data. A one-to-one line is included to show how the predictions diverge from a perfect fit.



The cause for the discrepancy is because the dependence of oxygen concentration on the unknowns is significantly nonlinear. It should be noted that this is particularly the case for the dependency on temperature.

15.7 The multiple linear regression model to evaluate is

$$y = a_0 + a_1T + a_2T^2 + a_3T^3 + a_4c$$

The $[Z]$ matrix can be set up as in

```
clear,clc,clf,format long
T = 0:5:30;
T = [T T T]';
c = [0 0 0 0 0 0 0 10 10 10 10 10 10 10 20 20 20 20 20 20]';
o = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]';
Y = [14.6 12.8 11.3 10.1 9.09 8.26 7.56 12.9 11.3 10.1 9.03 8.17 ...
     7.46 6.85 11.4 10.3 8.96 8.08 7.35 6.73 6.2]';
Z = [o T T.^2 T.^3 c];
```

Then, the coefficients can be generated by solving Eq.(15.10)

```
a = (Z'*Z)\[Z'*Y]

a =
    14.02714285714287
    -0.33642328042328
```

```

0.00574444444444
-0.00004370370370
-0.10492857142857

```

Thus, the least-squares fit is

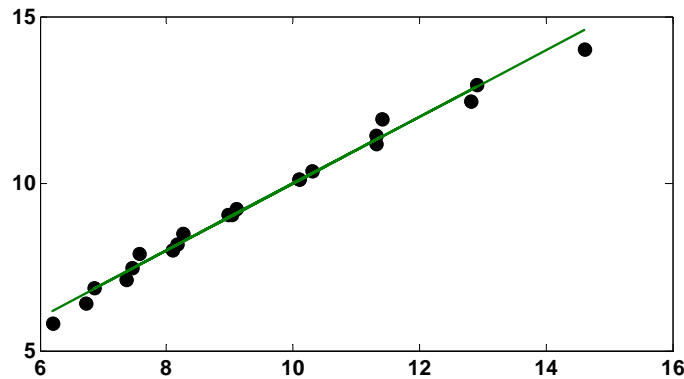
$$y = 14.027143 - 0.336423T + 0.00574444T^2 - 0.000043704T^3 - 0.10492857c$$

The model can then be used to predict values of oxygen at the same values as the data. These predictions can be plotted against the data to depict the goodness of fit.

```

>> yp = Z*a;
>> plot(Y,yp,'o',Y,Y)

```



Finally, the prediction can be made at $T = 12$ and $c = 15$,

```

>> a(1)+a(2)*12+a(3)*12^2+a(4)*12^3+a(5)*15
ans =
    9.16781492063485

```

which compares favorably with the true value of 9.09 mg/L.

15.8 The multiple linear regression model to evaluate is

$$y = a_0 + a_1x_1 + a_2x_2$$

The $[Z]$ matrix can be set up as in

```

>> x1 = [0 1 1 2 2 3 3 4 4]';
>> x2 = [0 1 2 1 2 1 2 1 2]';
>> y = [15.1 17.9 12.7 25.6 20.5 35.1 29.7 45.4 40.2]';
>> o = [1 1 1 1 1 1 1 1 1]';
>> Z = [o x1 x2];

```

Then, the coefficients can be generated by solving Eq.(15.10)

```

>> a = (Z'*Z)\[Z'*y]
a =
    14.4609
     9.0252
    -5.7043

```

Thus, the least-squares fit is

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$y = 14.4609 + 9.0252x_1 - 5.7043x_2$$

The model can then be used to predict values of the unknown at the same values as the data. These predictions can be used to determine the correlation coefficient and the standard error of the estimate.

```
>> yp = Z*a
>> SSR = sum((yp - y).^2)
SSR =
    4.7397

>> SST = sum((y - mean(y)).^2)
SST =
    1.0587e+003

>> r2 = (SST - SSR)/SST
r2 =
    0.9955

>> r = sqrt(r2)
r =
    0.9978

>> syx = sqrt(SSR/(length(y)-3))
syx =
    0.8888
```

15.9 The multiple linear regression model to evaluate is

$$\log Q = \log \alpha_0 + \alpha_1 \log(D) + \alpha_2 \log(S)$$

The [Z] matrix can be set up as in

```
>> D = [.3 .6 .9 .3 .6 .9 .3 .6 .9]';
>> S = [.001 .001 .001 .01 .01 .01 .05 .05 .05]';
>> Q = [.04 .24 .69 .13 .82 2.38 .31 1.95 5.66]';
>> o = [1 1 1 1 1 1 1 1 1]';
>> Z = [o log10(D) log10(S)]
```

Then, the coefficients can be generated by solving Eq.(15.10)

```
>> a = (Z'*Z)\[Z'*log10(Q)]
a =
    1.5609
    2.6279
    0.5320
```

Thus, the least-squares fit is

$$\log Q = 1.5609 + 2.6279 \log(D) + 0.5320 \log(S)$$

Taking the inverse logarithm gives

$$Q = 10^{1.5609} D^{2.6279} S^{0.5320} = 36.3813 D^{2.6279} S^{0.5320}$$

15.10 The linear regression model to evaluate is

$$p(t) = Ae^{-1.5t} + Be^{-0.3t} + Ce^{-0.05t}$$

The unknowns can be entered and the [Z] matrix can be set up as in

```
>> p = [6 4.4 3.2 2.7 2 1.9 1.7 1.4 1.1]';
>> t = [0.5 1 2 3 4 5 6 7 9]';
>> Z = [exp(-1.5*t) exp(-0.3*t) exp(-0.05*t)];
```

Then, the coefficients can be generated by solving Eq.(15.10)

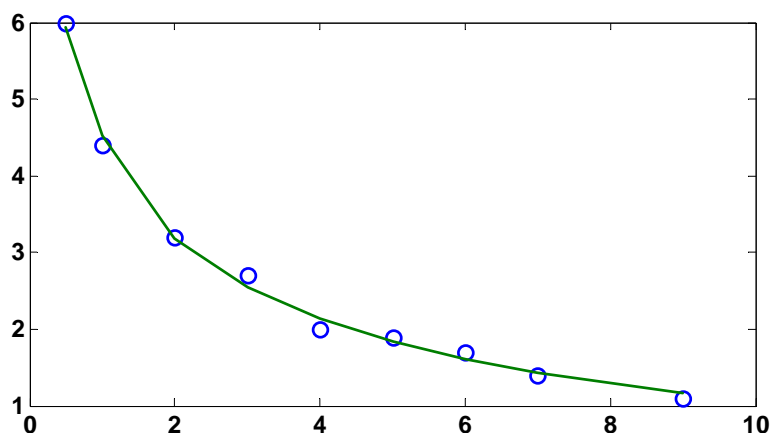
```
>> a = (Z'*Z)\[Z'*p]
a =
    4.1375
    2.8959
    1.5349
```

Thus, the least-squares fit is

$$p(t) = 4.1375e^{-1.5t} + 2.8959e^{-0.3t} + 1.5349e^{-0.05t}$$

The fit and the data can be plotted as

```
>> pp = Z*a;
>> plot(t,p,'o',t,pp)
```



15.11 First, an M-file function must be created to compute the sum of the squares,

```
function f = fSSR(a,Im,Pm)
Pp = a(1)*Im/a(2).*exp(-Im/a(2)+1);
f = sum((Pm-Pp).^2);
```

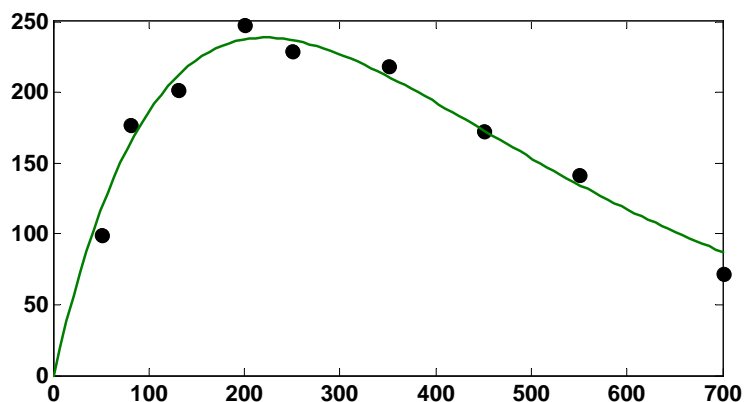
The following script then generates the best fit and the plot:

```
clear,clc,clf,format short g
I = [50 80 130 200 250 350 450 550 700];
P = [99 177 202 248 229 219 173 142 72];
a = fminsearch(@fSSR1511, [200, 200], [], I, P)
Ip=linspace(0,max(I));
Pp = a(1)*Ip/a(2).*exp(-Ip/a(2)+1);
plot(I,P,'o',Ip,Pp)
```

a =
238.71 221.82

The best-fit model is therefore

$$P = 238.71 \frac{I}{221.82} e^{-\frac{I}{221.82} + 1}$$



15.12 The following script solves the problem and generates a plot of the results:

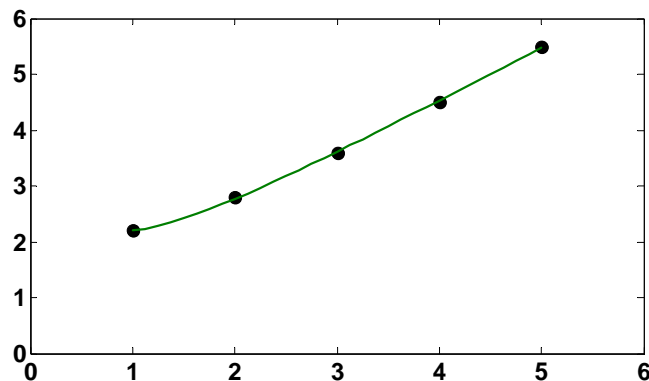
```
clear,clc,format short g
x =[1 2 3 4 5]';
y =[2.2 2.8 3.6 4.5 5.5]';
Z = [ones(size(x)) x 1./x];
a = (Z'*Z)\(Z'*y)
Sr = sum((y-Z*a).^2)
r2 = 1-Sr/sum((y-mean(y)).^2)
syx = sqrt(Sr/(length(x)-length(a)))
clf
xp=[1:0.125:5];
yp=a(1)+a(2)*xp+a(3)./xp;
plot(x,y,'o',xp,yp)
xlim([0 6]),ylim([0 6])
```

Results:

a =
0.3745
0.98644
0.84564
Sr =
0.0027651
r2 =
0.9996
syx =
0.037183

Therefore, the final model is

$$y = 0.3745 + 0.98644x + \frac{0.84564}{x}$$



15.13 First, an M-file function must be created to compute the sum of the squares,

```
function f = fSSR(a,xm,ym)
yp = a(1)*xm.*exp(a(2)*xm);
f = sum((ym-yp).^2);
```

The data can then be entered as

```
>> x = [.1 .2 .4 .6 .9 1.3 1.5 1.7 1.8];
>> y = [0.75 1.25 1.45 1.25 0.85 0.55 0.35 0.28 0.18];
```

The minimization of the function is then implemented by

```
>> a = fminsearch(@fSSR, [1, 1], [], x, y)
```

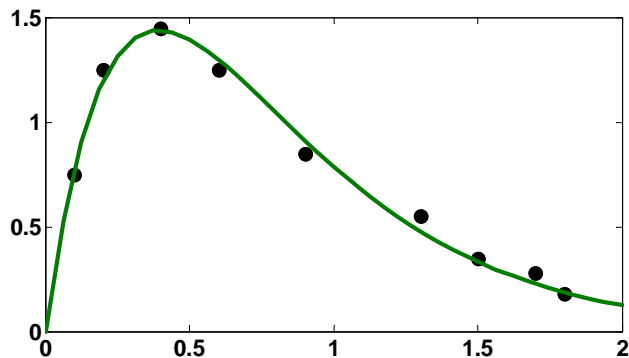
```
a =
    9.8974    -2.5319
```

The best-fit model is therefore

$$y = 9.8974xe^{-2.5319x}$$

The fit along with the data can be displayed graphically.

```
>> xp=[0:2/32:2];
>> yp = a(1)*xp.*exp(a(2)*xp);
>> plot(x,y,'o',xp,yp)
```



15.14 (a) The model can be linearized by inverting it,

$$\frac{1}{v_0} = \frac{K}{k_m} \frac{1}{[S]^3} + \frac{1}{k_m}$$

If this model is valid, a plot of $1/v_0$ versus $1/[S]^3$ should yield a straight line with a slope of K/k_m and an intercept of $1/k_m$. The slope and intercept can be implemented in MATLAB using the M-file function `linregr` (Fig. 14.12),

```
>> S = [.01 .05 .1 .5 1 5 10 50 100];
>> v0 = [6.078e-11 7.595e-9 6.063e-8 5.788e-6 1.737e-5 2.423e-5 ...
        2.43e-5 2.431e-5 2.431e-5];
>> a = linregr(1./S.^3,1./v0)

a =
        16453        41400
```

These results can then be used to compute k_m and K ,

```
>> km=1/a(2)
km =
    2.4155e-005

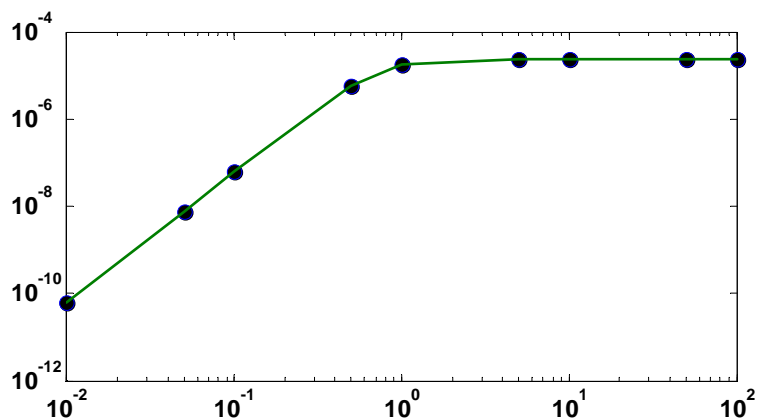
>> K=km*a(1)
K =
    0.39741
```

Thus, the best-fit model is

$$v_0 = \frac{2.4155 \times 10^{-5} [S]^3}{0.39741 + [S]^3}$$

The fit along with the data can be displayed graphically. We will use a log-log plot because of the wide variation of the magnitudes of the values being displayed,

```
>> v0p = km*S.^3./(K+S.^3);
>> loglog(S,v0,'o',S,v0p)
```



(b) An M-file function must be created to compute the sum of the squares,

```
function f = fSSR1514(a,Sm,v0m)
v0p = a(1)*Sm.^3./(a(2)+Sm.^3);
f = sum((v0m-v0p).^2);
```

The data can then be entered as

```
>> S = [.01 .05 .1 .5 1 5 10 50 100];
>> v0 = [6.078e-11 7.595e-9 6.063e-8 5.788e-6 1.737e-5 ...
        2.423e-5 2.43e-5 2.431e-5 2.431e-5];
```

The minimization of the function is then implemented by

```
>> a = fminsearch(@fSSR1514, [2e-5, 1], [], S, v0)
```

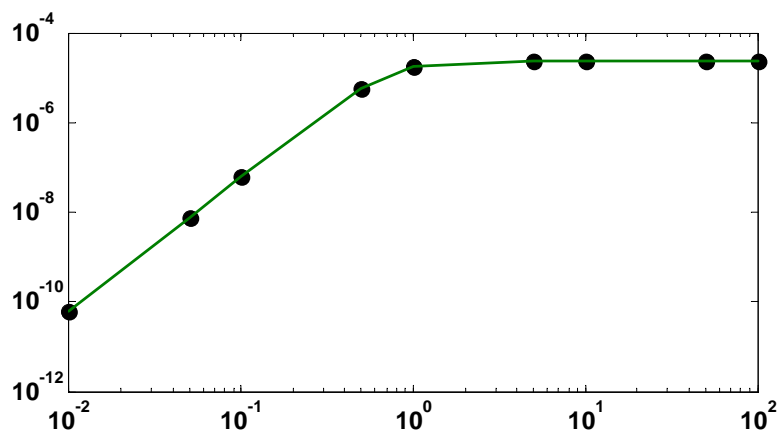
```
a =
    2.431e-005    0.39976
```

The best-fit model is therefore

$$v_0 = \frac{2.431 \times 10^{-5} [S]^3}{0.399763 + [S]^3}$$

The fit along with the data can be displayed graphically. We will use a log-log plot because of the wide variation of the magnitudes of the values being displayed,

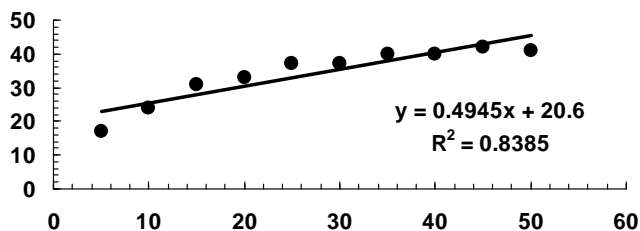
```
>> v0p = a(1)*S.^3./(a(2)+S.^3);
>> loglog(S,v0,'o',S,v0p)
```



15.15 (a) We regress y versus x to give

$$y = 20.6 + 0.494545 x$$

The model and the data can be plotted as



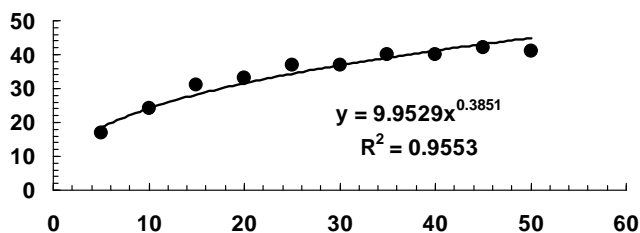
(b) We regress $\log_{10}y$ versus $\log_{10}x$ to give

$$\log_{10} y = 0.99795 + 0.385077 \log_{10} x$$

Therefore, $\alpha_2 = 10^{0.99795} = 9.952915$ and $\beta_2 = 0.385077$, and the power model is

$$y = 9.952915x^{0.385077}$$

The model and the data can be plotted as



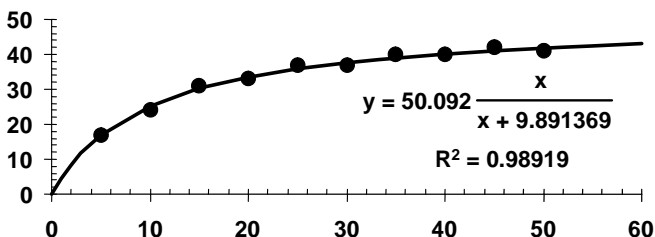
(c) We regress $1/y$ versus $1/x$ to give

$$\frac{1}{y} = 0.019963 + 0.197464 \frac{1}{x}$$

Therefore, $\alpha_3 = 1/0.01996322 = 50.09212$ and $\beta_3 = 0.19746357(50.09212) = 9.89137$, and the saturation-growth-rate model is

$$y = 50.09212 \frac{x}{9.89137 + x}$$

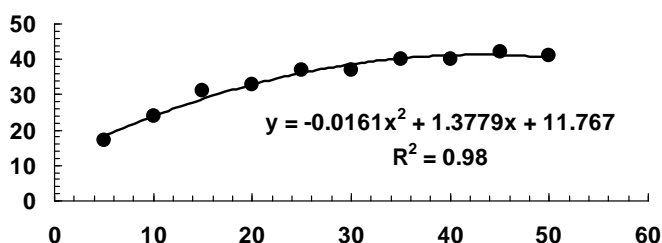
The model and the data can be plotted as



(d) We employ polynomial regression to fit a parabola

$$y = -0.01606x^2 + 1.377879x + 11.76667$$

The model and the data can be plotted as



Comparison of fits: The linear fit is obviously inadequate. Although the power fit follows the general trend of the data, it is also inadequate because (1) the residuals do not appear to be randomly distributed around the best fit line and (2) it has a lower r^2 than the saturation and parabolic models.

The best fits are for the saturation-growth-rate and the parabolic models. They both have randomly distributed residuals and they have similar high coefficients of determination. The saturation model has a slightly higher r^2 . Although the difference is probably not statistically significant, in the absence of additional information, we can conclude that the saturation model represents the best fit.

15.16 The following script can be developed to solve this problem:

```
clear,clc,format short g
format compact
t = [0 4 8 12 16 20]';
B = [67.38 74.67 82.74 91.69 101.60 112.58]';

disp('linear:')
Z=[ones(size(t)) t];
a=(Z'*Z)\(Z'*B);a'
Sr=sum((B-Z*a).^2);
r2=1-Sr/sum((B-mean(B)).^2)
syx=sqrt(Sr/(length(B)-length(a)))
tp=[0:30];
Bplin=a(1)+a(2)*tp;
Bp30lin=a(1)+a(2)*30;
pause

disp('parabolic:')
Z=[ones(size(t)) t t.^2];
a=(Z'*Z)\(Z'*B);a'
Sr=sum((B-Z*a).^2);
r2=1-Sr/sum((B-mean(B)).^2)
syx=sqrt(Sr/(length(B)-length(a)))
tp=[0:30];
Bpquad=a(1)+a(2)*tp+a(3)*tp.^2;
Bp30quad=a(1)+a(2)*30+a(3)*30^2
pause

disp('exponential:')
Z=[ones(size(t)) t];
a=(Z'*Z)\(Z'*log(B));a'
Sr=sum((log(B)-Z*a).^2);
r2=1-Sr/sum((log(B)-mean(log(B))).^2)
syx=sqrt(Sr/(length(log(B))-length(a)))
tp=[0:30];
Bpexp=exp(a(1)+a(2)*tp);
Bp30exp=exp(a(1)+a(2)*30);
```



```

Bp30lin
Bp30quad
Bp30exp
clf
plot(t,B,'o',tp,Bplin,tp,Bpquad,tp,Bpexp)
legend('data','linear','quadratic','exponential','location','Best')

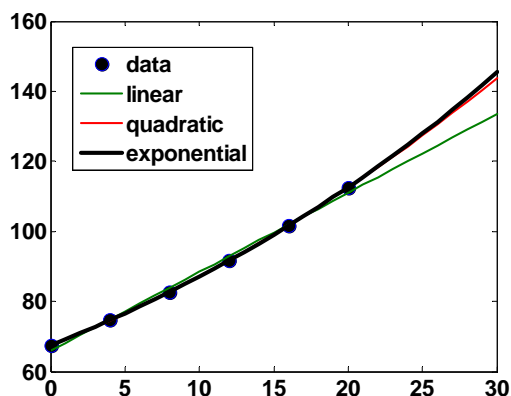
```

When this script is run, the result is

```

linear:
ans =
    65.89    2.2553
r2 =
    0.99445
syx =
    1.4095
parabolic:
ans =
    67.427    1.6792    0.028806
r2 =
    0.99999
syx =
    0.072757
Bp30quad =
    143.73
exponential:
ans =
    4.2104    0.025666
r2 =
    1
syx =
    3.6082e-005
Bp30lin =
    133.55
Bp30quad =
    143.73
Bp30exp =
    145.53

```



The linear model is inadequate since it does not capture the curving trend of the data. At face value, the parabolic and exponential models appear to be equally good. However, inspection of the standard error of the estimate indicates that the exponential model is somewhat better. In addition, knowledge of bacterial growth might lead you to choose the exponential model as it is commonly used to simulate the growth of microorganism populations. Interestingly, the choice matters when the models are used for prediction. If the exponential model is used, the result is

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

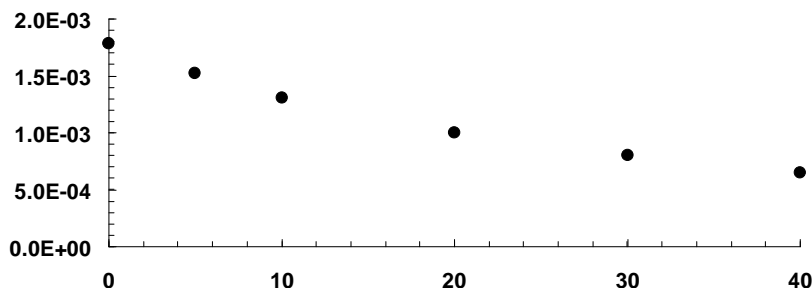
$$B = 67.383e^{0.025666(30)} = 145.53$$

For the parabolic model, the prediction is

$$B = 0.028806t^2 + 1.6792t + 67.427 = 143.73$$

Thus, even though the models would yield very similar results within the data range, they yield different results for extrapolation outside the range.

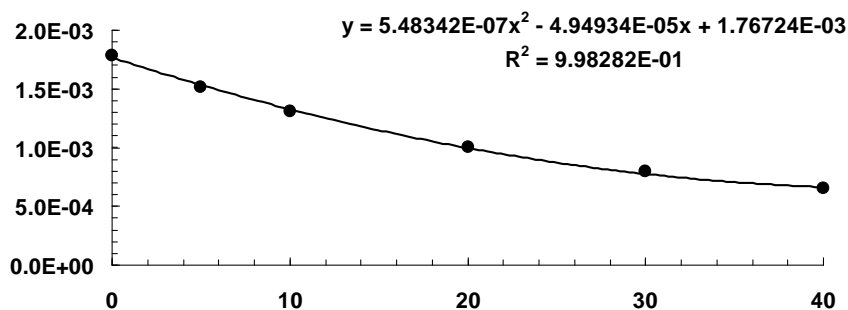
15.17 (a) The data can be plotted as



(b) Linear interpolation yields.

$$\mu = 1.519 \times 10^{-3} + \frac{1.307 \times 10^{-3} - 1.519 \times 10^{-3}}{10 - 5} (7.5 - 5) = 1.413 \times 10^{-3}$$

(c) Polynomial regression yields



This leads to a prediction of

$$\mu = 5.48342 \times 10^{-7} (7.5)^2 - 4.94934 \times 10^{-5} (7.5) + 1.76724 \times 10^{-3} = 1.4269 \times 10^{-3}$$

15.18 The equation can be expressed in the form of the general linear least-squares model,.

$$\frac{PV}{RT} - 1 = A_1 \frac{1}{V} + A_2 \frac{1}{V^2}$$

```
>> R=82.05; T=303;
>> P=[0.985 1.108 1.363 1.631]';
>> V=[25000 22200 18000 15000]';
>> y=P.*V/R/T-1;
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```
>> Z=[1./V 1./V.^2];
>> a=(Z'*Z)\(Z'*y)
```

```
a =
    -231.67
   -1.0499e+005
```

15.19 We can use general linear least squares to generate the best-fit equation. The [Z] and y matrices can be set up using MATLAB commands as

```
>> format long
>> x = [273.15 283.15 293.15 303.15 313.15]';
>> Kw = [1.164e-15 2.950e-15 6.846e-15 1.467e-14 2.929e-14]';
>> y=-log10(Kw);
>> Z = [(1./x) log10(x) x ones(size(x))];
```

The coefficients can be evaluated as

```
>> a=inv(Z'*Z)*(Z'*y)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.457873e-020.
a =
    1.0e+003 *
    5.18067187500000
    0.01342456054688
    0.00000562858582
   -0.03827636718750
```

Note the warning that the results are ill-conditioned. According to this calculation, the best-fit model is

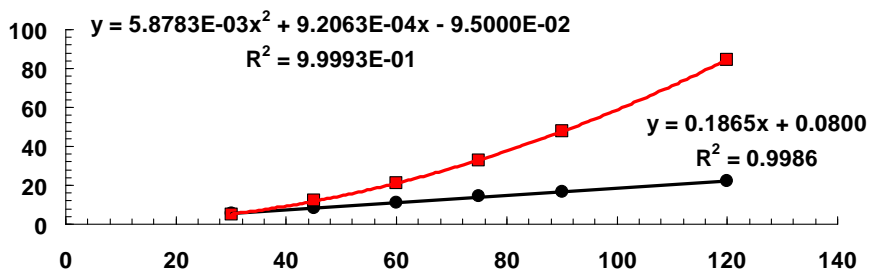
$$-\log_{10} K_w = \frac{5180.67}{T_a} + 13.42456 \log_{10} T_a + 0.00562859 T_a - 38.276367$$

We can check the results by using the model to make predictions at the values of the original data

```
>> yp=10.^-(a(1)./x+a(2)*log10(x)+a(3)*x+a(4))
yp =
    1.0e-013 *
    0.01161193308242
    0.02943235714551
    0.06828461729494
    0.14636330575049
    0.29218444886852
```

These results agree to about 2 or 3 significant digits with the original data.

15.20 The “Thinking” curve can be fit with a linear model whereas the “Braking” curve can be fit with a quadratic model as in the following plot.



A prediction of the total stopping distance for a car traveling at 110 km/hr can be computed as

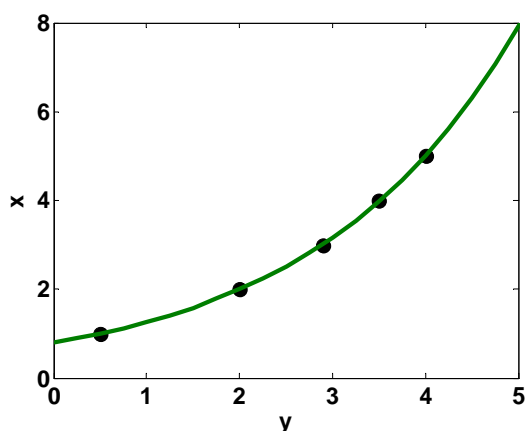
$$d = 0.1865(110) + 0.0800 + 5.8783 \times 10^{-3}(110)^2 + 9.2063 \times 10^{-4}(110) - 9.5000 \times 10^{-2} = 91.726 \text{ m}$$

15.21 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

```
function f = fSSR1521(a,ym,xm)
xp = exp((ym-a(2))/a(1));
f = sum((xm-xp).^2);

clear,clc,clf,format short g
x=[1 2 3 4 5];
y=[0.5 2 2.9 3.5 4];
a = fminsearch(@fSSR1521, [2,0.5], [], y, x)
yp=[0:0.25:6];
xp = a(1)*yp.*exp(a(2)*yp);
xp = exp((yp-a(2))/a(1));
plot(y,x,'o',yp,xp)
xlabel('y'),ylabel('x')
axis([0 5 0 8])
```

```
a =
    2.173    0.49924
```



15.22 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

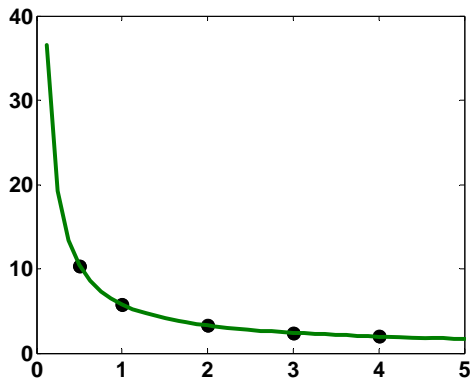
```
function f = fSSR1522(a,xm,ym)
yp = ((a(1)+sqrt(xm))/a(2))./sqrt(xm)).^2;
f = sum((ym-yp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear, clc, clf, format short g
x=[0.5 1 2 3 4];
y=[10.4 5.8 3.3 2.4 2];
a = fminsearch(@fSSR1522, [5,2], [], x, y)
yy = ((a(1)+sqrt(1.6))/a(2)./sqrt(1.6)).^2;
fprintf('y(1.6) =%9.5f\n',yy)
xp=[1/8:1/8:5];
yp = ((a(1)+sqrt(xp))/a(2)./sqrt(xp)).^2;
plot(x,y, 'o', xp, yp)
```

```
a =
    4.8375    2.4303
```

```
y(1.6) =  3.94057
```



15.23 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

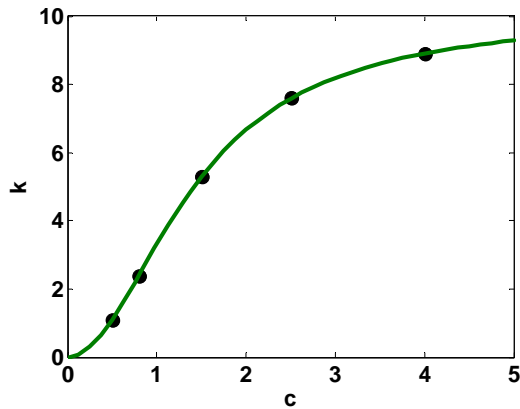
```
function f = fSSR1523(a,cm,km)
kp = a(1)*cm.^2./(a(2)+cm.^2);
f = sum((km-kp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear, clc, clf, format short g
c=[0.5 0.8 1.5 2.5 4];
k=[1.1 2.4 5.3 7.6 8.9];
a = fminsearch(@fSSR1523, [10,2], [], c, k)
kk = a(1)*2^2/(a(2)+2^2);
fprintf('k(2) =%9.5f\n',kk)
cp=[0:0.125:5];
kp = a(1)*cp.^2./(a(2)+cp.^2);
plot(c,k, 'o', cp, kp)
```

```
a =
    10.036    2.0179
```

```
k(2) =  6.67058
```



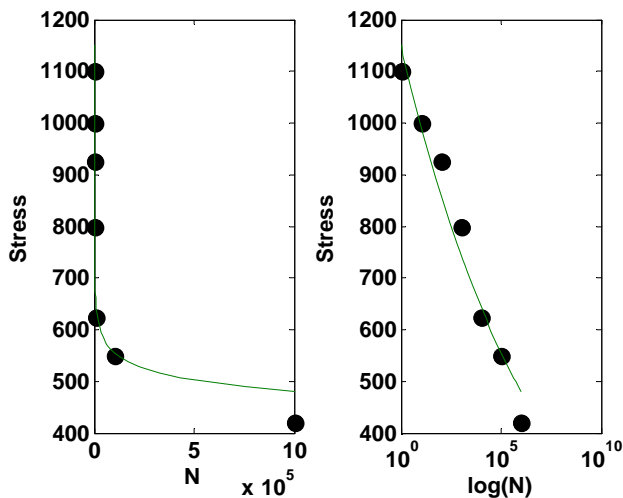
15.24 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

```
function f = fSSR1524(a,Nm,Stressm)
Stressp = a(1)*Nm.^a(2);
f = sum((Stressm-Stressp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear, clc, clf, format short g
N=[1 10 100 1000 10000 100000 1000000];
Stress=[1100 1000 925 800 625 550 420];
a = fminsearch(@fSSR1524, [10,2], [], N, Stress)
Np=logspace(0,6);
Stressp = a(1)*Np.^a(2);
subplot(1,2,1)
plot(N,Stress,'o',Np,Stressp)
xlabel('N'),ylabel('Stress')
subplot(1,2,2)
semilogx(N,Stress,'o',Np,Stressp)
xlabel('log(N)'),ylabel('Stress')
```

```
a =
    1150.3    -0.063082
```



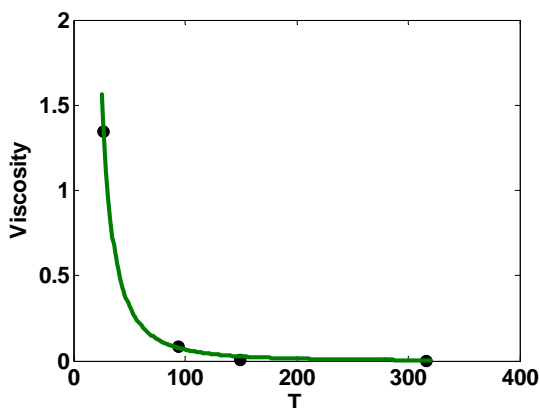
15.25 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

```
function f = fSSR1525(a,Tm,Viscm)
Viscp = a(1)*Tm.^a(2);
f = sum((Viscm-Viscp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear,clc,clf,format short g
T=[26.67 93.33 148.89 315.56];
Visc=[1.35 0.085 0.012 0.00075];
a = fminsearch(@fSSR1525, [10,2], [], T, Visc)
Tp=[25:320];
Viscp = a(1)*Tp.^a(2);
plot(T,Visc,'o',Tp,Viscp)
xlabel('T'),ylabel('Viscosity')
```

```
a =
    2416.2    -2.281
```



15.26 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

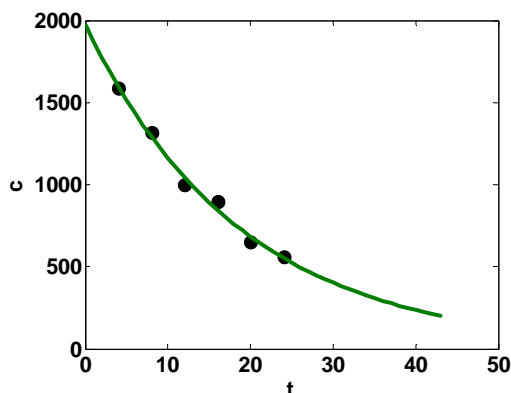
```
function f = fSSR1526(a,tm,cm)
cp = a(1)*exp(a(2)*tm);
f = sum((cm-cp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear,clc,clf,format short g
t=[4 8 12 16 20 24];
c=[1590 1320 1000 900 650 560];
a = fminsearch(@fSSR1526, [10,2], [], t, c)
fprintf('c(0) =%9.5f\n',a(1))
ft=@(t) a(1)*exp(a(2)*t)-200;
t200=fzero(ft,50);
fprintf('t(200) =%9.5f\n',t200)
tp=[0:t200];
cp = a(1)*exp(a(2)*tp);
plot(t,c,'o',tp,cp)
xlabel('t'),ylabel('c')
```

```
a =
    1975.5    -0.053005
```

$c(0) = 1975.54653$
 $t(200) = 43.20874$



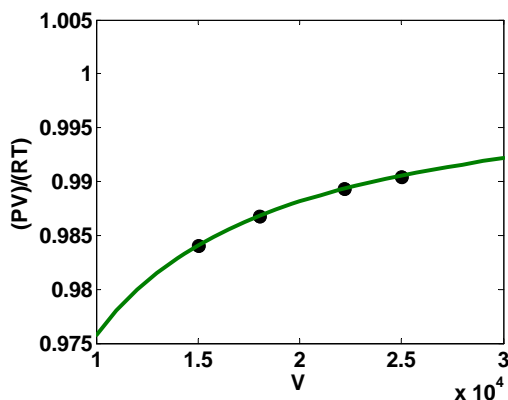
15.27 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

```
function f = fSSR1527(a,V,ym)
yp=1+a(1)./V+a(2)./V.^2;
f = sum((ym-yp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear,clc,clf,format short g
R=82.05;T=303;
P=[0.985 1.108 1.363 1.631]';
V=[25000 22200 18000 15000]';
y=P.*V/R/T;
a=fminsearch(@fSSR1527,[-200 1],[],V,y)
Vp=[10000:1000:30000];
yp=1+a(1)./Vp+a(2)./Vp.^2;
plot(V,y,'o',Vp,yp)
xlabel('V'),ylabel('(PV)/(RT)')
```

```
a =
-231.67 -1.0499e+005
```



15.28 Nonlinear regression can be used to evaluate the coefficients. First, a function can be set up to compute the sum of the squares of the residuals,

```
function f = fSSR1528(a,t,pm)
```

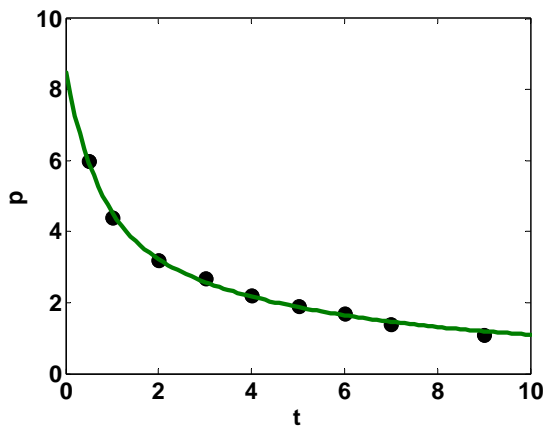
PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.


```
pp=a(1)*exp(-1.5*t)+a(2)*exp(-0.3*t)+a(3)*exp(-0.05*t);
f = sum((pm-pp).^2);
```

Then, the `fminsearch` function can be employed to determine the coefficients,

```
clear,clc,clf,format short g
t=[0.5 1 2 3 4 5 6 7 9];
p=[6.0 4.4 3.2 2.7 2.2 1.9 1.7 1.4 1.1];
a=fminsearch(@fSSR1528,[1 1 1],[],t,p)
tp=[0:0.1:10];
pp=a(1)*exp(-1.5*tp)+a(2)*exp(-0.3*tp)+a(3)*exp(-0.05*tp);
plot(t,p,'o',tp,pp)
xlabel('t'),ylabel('p')
```

```
a =
    4.0046    2.9213    1.5647
```



Therefore, $A = 4.0046$, $B = 2.9213$, and $C = 1.5647$, and the final model is

$$p(t) = 4.0046e^{-1.5t} + 2.9213e^{-0.3t} + 1.5647e^{-0.05t}$$