5.1 The following data is given:

x	1	2	4	7	10	12	15
y	3	5	7	11	14	17	21

(*a*) Use linear least-squares regression to determine the coefficients m and b in the function $y = mx + b$ that best fit the data.
(*b*) Use Eq. (5.5) to determine the overall error.

Solution

(*a*) In the equation $y = mx + b$, m corresponds to a_1 and b corresponds to a_0 in the discussion of Section 5.2.2. Using Eq.(5.13),

$$S_x = \sum_{i=1}^{n} x_i = 1 + 2 + 4 + 7 + 10 + 12 + 15 = 51$$

$$S_y = \sum_{i=1}^{n} y_i = 3 + 5 + 7 + 11 + 14 + 17 + 21 = 78$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i = 1(3) + 2(5) + 4(7) + 7(11) + 10(14) + 12(17) + 15(21) = 777$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 = 1^2 + 2^2 + 4^2 + 7^2 + 10^2 + 12^2 + 15^2 = 539$$

Using Eq.(5.14) with $n = 7$:

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{7(777) - 51(78)}{7(539) - 51^2} = 1.2466$$

$$a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - (S_x)^2} = \frac{539(78) - 777(51)}{7(539) - 51^2} = 2.0606$$

The linear function that best fit the data is: $y = 1.2466x + 2.0606$.

(*b*) Using Eq.(5.5), the overall error is:

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]^2$$

$$E = (3 - 3.3072)^2 + (5 - 4.5538)^2 + (7 - 7.0470)^2 + (11 - 10.7868)^2 + (14 - 14.5266)^2 + (17 - 17.0198)^2 + (21 - 20.7596^2)$$

$$E = 0.6766$$

5.2 The following data is given:

x	-6	-4	-1	0	3	5	8
y	18	13	6	4	-1	-8	-15

(*a*) Use linear least-squares regression to determine the coefficients m and b in the function $y = mx + b$ that best fit the data.
(*b*) Use Eq. (5.5) to determine the overall error.

Solution

(*a*) In the equation $y = mx + b$, m corresponds to a_1 and b corresponds to a_0 in the discussion of Section 5.2.2. Using Eq.(5.13)

$$S_x = \sum_{i=1}^{n} x_i = -6-4-1-0+3+5+8 = 5$$

$$S_y = \sum_{i=1}^{n} y_i = 18+13+6+4-1-8-15 = 17$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i = -6(18)-4(13)-1(6)+0(4)+3(-1)+5(-8)+8(-15) = -329$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 = (-6)^2+(-4)^2+(-1)^2+0^2+3^2+5^2+8^2 = 151$$

Using Eq.(5.14) with $n = 7$:

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{(7)(-329)-(5)(17)}{(7)(151)-(5)^2} = -2.3140$$

$$a_0 = \frac{S_{xx}S_y - S_{xy}S_x}{nS_{xx} - (S_x)^2} = \frac{(151)(17)-(-329)(5)}{(7)(151)-(5)^2} = 4.0814$$

The linear function that best fit the data is: $y = (-2.324)x + 4.0814$.

(*b*) Using Eq.(5.5), the overall error is:

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]^2$$

$$= (18 - 17.9654)^2 + (13 - (13.3374))^2 + (6 - 6.3954)^2 + (4 - 4.0814)^2 + (-1 + 2.8606)^2 + (-8 + 7.4886)^2$$
$$\cdot (-15 + 14.4306)^2$$

$$E = 4.3256$$

5.3 The following data give the approximate population of the U.S. for selected years from 1845 until 2000:

Year	1845	1871	1915	1954	2000
Population (millions)	20	30	100	160	280

Assume that the population growth can be modeled with an exponential function $p = be^{mx}$, where x is the year and p is the population in millions. Write the equation in a linear form (Section 5.3), and use linear least-squares regression to determine the constants b and m for which the function best fits the data. Use the equation to estimate the population in the year 1970.

Solution

For $p = be^{mx}$, take the natural logarithm of both sides to yield $\ln(p) = \ln(b) + mx$. This equation is in the form of $y = a_1 x + a_0$ with $\ln(b)$ corresponds to a_0 and m corresponds to a_1. First, calculate $y = \ln(p)$:

x (year)	1845	1871	1915	1954	2000
y ($\ln(p)$)	2.9957	3.4012	4.6052	5.0752	5.6348

Using Eq.(5.13)

$$S_x = \sum_{i=1}^{n} x_i = 1845 + 1871 + 1915 + 1954 + 2000 = 9585$$

$$S_y = \sum_{i=1}^{n} y_i = 2.9957 + 3.4012 + 4.6052 + 5.0752 + 5.6348 = 21.7121$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i = 1845(2.9957) + 1871(3.4012) + 1915(4.6052) + 1954(5.0752) + 2000(5.6348) = 41896$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 = 1845^2 + 1871^2 + 1915^2 + 1954^2 + 2000^2 = 18390007$$

Using Eq.(5.14),

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{(5)(41896) - (9585)(21.7121)}{(5)(41896) - (9585)^2} = 0.0176$$

$$a_0 = \frac{S_{xx}S_y - S_{xy}S_x}{nS_{xx} - (S_x)^2} = \frac{(18390007)(21.7121) - (41896)(9585)}{(5)(41896) - (9585)^2} = -29.4238$$

Thus, the linear least-squares regression yields $y = (0.0176)x - 29.4238$. Substituting $y = \ln(p)$ gives $\ln(p) = (0.0176)x - 29.4238$, or:

$$p = e^{(0.0176)x - 29.4238} = e^{-29.4238} \cdot e^{(0.0176)x} = (1.665 \times 10^{-13})e^{(0.0176)x}$$

which is of the form $p = be^{mx}$ with $b = 1.665 \times 10^{-13}$ and $m = 0.0176$.

The population for the year 1970:

$$p(1970) = (1.665 \times 10^{-13})e^{0.0176 \cdot 1970} = 190.23 \text{ millions}$$

5.4 The following data is given:

x	-0.6	-0.4	-0.1	0.5	1.5
y	18	8	3	2	0.9

Determine the coefficients m and b in the function $y = \frac{1}{mx+b}$ that best fit the data. Write the equation in a linear form (Section 5.3), and use linear least-squares regression to determine the value of the coefficients.

Solution

The transforming equations from the nonlinear function $y = \frac{1}{mx+b}$ to the linear form are given in Table 5-2 as $Y = \frac{1}{y}$, $X = x$, $m = a_1$ and $b = a_0$. The transformation yields

X	-0.6	-0.4	-0.1	0.5	1.5
Y	0.0556	0.125	0.3333	0.5	1.1111

Using Eq.(5.13)

$$S_x = \sum_{i=1}^{n} x_i = -0.6 - 0.4 - 0.1 + 0.5 + 1.5 = 0.9$$

$$S_y = \sum_{i=1}^{n} y_i = 0.0556 + 0.125 + 0.3333 + 0.5 + 1.1111 = 2.125$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i = -0.6(0.0556) - 0.4(0.125) - 0.1(0.3333) + 0.5(0.5) + 1.5(1.1111) = 1.8$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 = (-0.6)^2 + (-0.4)^2 + (-0.1)^2 + 0.5^2 + 1.5^2 = 3.030$$

Using Eq.(5.14) for n=5,

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{(5)(1.8) - (0.9)(2.125)}{(5)(3.030) - (0.9)^2} = 0.4942$$

$$a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - (S_x)^2} = \frac{(3.030)(2.125) - (1.8)(0.9)}{(5)(3.030) - (0.9)^2} = 0.3360$$

Thus, the linear least-squares regression yields $Y = (0.4942)x + 0.3360$, where $m = a_1 = 0.4942$ and $b = a_0 = 0.3360$.

5.5 The following data is given:

x	0.5	1	1.5	2
y	2.3	3.2	3.7	3.8

Determine the coefficients m and b in the function $y = \frac{mx^2}{b + x^2}$ that best fit the data. Write the equation in a linear form (Section 5.3), and use linear least-squares regression to determine the value of the coefficients.

Solution

The equation is transformed to a linear form: $\frac{1}{y} = \frac{b}{m}\frac{1}{x^2} + \frac{1}{m}$

Linear least square regression is used to find the best fit between $\frac{1}{y}$ and $\frac{1}{x^2}$ in the form:

$$\frac{1}{y} = a_1\frac{1}{x^2} + a_0$$

to the linear form are given in Table 5-2 as $Y = \frac{1}{y}$, $X = \frac{1}{x}$, $a_1 = \frac{b}{m}$ and $a_0 = \frac{1}{m}$. Writing m and b in terms of a_1 and a_0 gives $m = \frac{1}{a_0}$ and $b = a_1 m$. The calculations are done by executing the following MATLAB program (script file):

```
clear all; clc;
x=[0.5 1 1.5 2];
y=[2.3 3.2 3.7 3.8];
% From Table 5-2
Y=1./y;
X=1./x.^2;
% Equation 5-13
SX=sum(X);
SY=sum(Y);
SXY=sum(X.*Y);
SXX=sum(X.*X);
% Equation 5-14
```

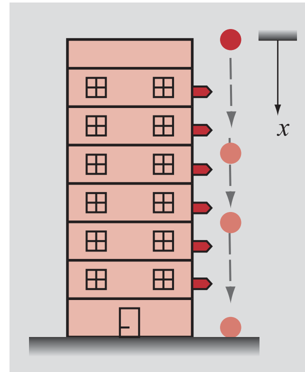
```
n=length(X);
a1=(n*SXY-SX*SY)/(n*SXX-SX^2)
a0=(SXX*SY-SXY*SX)/(n*SXX-SX^2)

m=1/a0
b=a1*m
```

When the program is executed, the following values are displayed in the Command Window:

```
a1 =
    0.0453
a0 =
    0.2557
m =
    3.9102
b =
    0.1770
```

5.6 To measure g (the acceleration due to gravity) the following experiment is carried out. A ball is dropped from the top of a 30-m-tall building. As the object is falling down, its speed v is measured at various heights by sensors that are attached to the building. The data measured in the experiment is given in the table.



x (m)	0	5	10	15	20	25
v (m/s)	0	9.85	14.32	17.63	19.34	22.41

In terms of the coordinates shown in the figure (positive down), the speed of the ball v as a function of the distance x is given by $v^2 = 2gx$. Using linear regression, determine the experimental value of g.

Solution

The equation $v^2 = 2gx$ can be transformed into linear form by setting $Y = v^2$. The resulting equation, $Y = 2gx$, is linear in Y and x with $m = 2g$ and $b = 0$. Therefore, once m is determined, g can be calculating using $g = \frac{m}{2}$. The calculations are done by executing the following MATLAB program (script file):

```
clear all; clc;
x=[0 5 10 15 20 25];
y=[0 9.85 14.32 17.63 19.34 22.41];
Y=y.^2;
X=x;
% Equation 5-13
SX=sum(X);
SY=sum(Y);
SXY=sum(X.*Y);
SXX=sum(X.*X);
% Equation 5-14
n=length(X);
a1=(n*SXY-SX*SY)/(n*SXX-SX^2)
a0=(SXX*SY-SXY*SX)/(n*SXX-SX^2)

m=a1
```

```
b=a0
g=m/2
```

When the program is executed, the following values are displayed in the Command Window:

```
a1 =
   19.7019
a0 =
    1.9170
m =
   19.7019
b =
    1.9170
g =
    9.8510
```

Thus, the measured value of g is 9.8510 m/s^2.

5.7 Air density, ρ, as a function of height, *h*, can be modeled by an exponential function of the form $\rho = be^{mh}$. The following are values of air density measured at different heights. Using linear regression, determine the constants *m* and *b* that best fit the data. Use the equation to estimate the air density at a height of 7,000 m.

h (m)	1	4,000	8,000	12,000	16,000	20,000
ρ (kg/m^3)	1.225	0.820	0.525	0.309	0.168	0.092

Solution

The nonlinear function $\rho = be^{mh}$ is transferred to a linear function $Y = a_1X + a_0$ (see Table 5-2) by substituting $Y = \ln y$, $X = x$, $a_1 = m$ and $a_0 = \ln b$. Writing m and b in terms of a_1 and a_0 gives $m = a_1$ and $b = e^{a_0}$. The calculations are done by executing the following MATLAB program (script file):

```
clear all; clc;
h=[1 4000 8000 12000 16000 20000];
r=[1.225 0.820 0.525 0.309 0.168 0.092];
X=h;
Y=log(r);
% Equation 5-13
SX=sum(X);
SY=sum(Y);
SXY=sum(X.*Y);
SXX=sum(X.*X);
% Equation 5-14
n=length(X);
a1=(n*SXY-SX*SY)/(n*SXX-SX^2);
a0=(SXX*SY-SXY*SX)/(n*SXX-SX^2);
m=a1
b=exp(a0)
% Estimated air density at 7,000 m
```

```
r7000=b*exp(m*7000)
```

When the program is executed, the following values are displayed in the Command Window:

```
m =
 -1.3022e-004
b =
    1.3565
r7000 =
    0.5452
```

Therefore, linear regression yields the relation $\rho = 1.3565\exp(-1.3022\times10^{-4}h)$. An estimate of the atmospheric pressure at $h = 7000$ m, is determined by substituting the value of h into the equation:

$$\rho|_{h\,=\,7000} = 1.3565\exp(-1.3022\times10^{-4}\times 7000)$$

$$p|_{h\,=\,7000} = 0.5452\ \ \text{kg/m}^3$$

5.8 In an electrophoretic fiber making process, the diameter of the fiber, *d*, is related to the current flow, *I*. The following is measured during production:

I (nA)	300	300	350	400	400	500	500	650	650
d (μm)	22	26	27	30	34	33	33.5	37	42

The relationship between the current and the diameter can be modeled with an equation of the form $d = a + b\sqrt{I}$. Use the data to determine the constants *a* and *b* that best fit the data.

Solution

The equation $d = a + b\sqrt{I}$ is linear in $\sqrt{I}$ and d. The coefficients a and b are found by setting $Y = d$ and $X = \sqrt{I}$ and using the method of linear regression. The calculations are done by executing the following MATLAB program (script file):

```
clear all; clc;
I=[300 300 350 400 400 500 500 650 650];
d=[22 26 27 30 34 33 33.5 37 42];
X=sqrt(I);
Y=d;
% Equation 5-13
SX=sum(X);
SY=sum(Y);
SXY=sum(X.*Y);
SXX=sum(X.*X);
% Equation 5-14
n=length(X);
a1=(n*SXY-SX*SY)/(n*SXX-SX^2);
a0=(SXX*SY-SXY*SX)/(n*SXX-SX^2);

b=a1
a=a0
```

When the program is executed, the following values are displayed in the Command Window:

```
b =
    1.7998
a =
   -6.1967
```

5.9 Determine the coefficients of the polynomial $y = a_2x^2 + a_1x + a_0$ that best fit the data given in Problem 5.5.

Solution

The data points from Problem 5.5 are:

x	0.5	1	1.5	2
y	2.3	3.2	3.7	3.8

Curve fitting of these data points with the second-order polynomial is done by polynomial regression. The values of the three coefficients a_2, a_1 and a_0 are determined by solving a system of three linear equations, which given by Eqs. (5.26)-(5.28):

$$na_0 + \left(\sum_{i=1}^{n} x_i\right)a_1 + \left(\sum_{i=1}^{n} x_i^2\right)a_2 = \sum_{i=1}^{n} y_i$$

$$\left(\sum_{i=1}^{n} x_i\right)a_0 + \left(\sum_{i=1}^{n} x_i^2\right)a_1 + \left(\sum_{i=1}^{n} x_i^3\right)a_2 = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum_{i=1}^{n} x_i^2\right)a_0 + \left(\sum_{i=1}^{n} x_i^3\right)a_1 + \left(\sum_{i=1}^{n} x_i^4\right)a_2 = \sum_{i=1}^{n} x_i^2 y_i$$

The calculations can be performed using a MATLAB script that performs the following steps:
Step 1: Create vectors `x` and `y` with the data points.
Step 2: Create a vector `xsum` in which the elements are the summation terms of the powers of x_i.
Step 3: Set up the system of three linear equations in the form $[X][a] = [Y]$, where $[X]$ is the 3×3 matrix with the summation terms of the powers of x_i, $[a]$ is the 3×1 vector of the unknowns and $[Y]$ is the 3×1 vector of the summation terms on the right-hand side of Eqs. (5.26)-(5.28).
Step 4: Solve the system of three linear equations for $[a]$ using MATLAB's left division. The solution is a vector with the coefficients of the second-order polynomial that best fits the data.

These steps are performed using the following MATLAB script:

```
clear all; clc;
% Step 1
x=[0.5 1 1.5 2];
y=[2.3 3.2 3.7 3.8];
```

```
n=length(x);
m=4;
% Step 2
for i=1:m
    xsum(i)=sum(x.^(i));
end
% Step 3
% First row of matrix [X] and first element of column vector [Y]
X(1,1)=n;
Y(1,1)=sum(y);
for j=2:(m-1)
    X(1,j)=xsum(j-1);
end
% Rows 2 and 3 of matrix [X] and column vector [Y]
for i=2:(m-1)
    for j=1:(m-1)
        X(i,j)=xsum(j+i-2);
    end
   Y(i,1)=sum(x.^(i-1).*y);
end
% Step 4
a=X\Y
```

When the program is executed, the vector $[a]$ is displayed in the Command Window:

```
a =
    1.0000
    3.0000
   -0.8000
```

The second-order polynomial that best fits the data is $y = -0.8x^2 + 3x + 1$.

5.11 Using the method in Section 5.8, determine the coefficients of the equation $y = Ae^{0.8x} + B\sqrt{x} + Cx$ that best fit the following data:

x	0.5	1.0	1.5	2.0	2.5
y	-3.1	-3.6	-3.4	-2.8	-1.9

Solution

In the notation of Eq. (5.91) the approximating function is $F(x) = C_1f_1(x) + C_2f_2(x) + C_3f_3(x)$, where $f_1(x) = e^{0.8x}$, $f_2(x) = \sqrt{x}$ and $f_3(x) = x$. The equation has three terms, so $m = 3$, and there are six data points, so $n = 6$. Substituting this information into Eq. (5.97) gives the following system of three linear equations for C_1, C_2, C_3, C_4:

$$\sum_{i=1}^{6} C_1 e^{0.8x_i} e^{0.8x_i} + \sum_{i=1}^{6} C_2 e^{0.8x_i}\sqrt{x_i} + \sum_{i=1}^{6} C_3 e^{0.8x_i} x_i = \sum_{i=1}^{6} y_i e^{0.8x_i}$$

$$\sum_{i=1}^{6} C_1 e^{0.8x_i}\sqrt{x_i} + \sum_{i=1}^{6} C_2 \sqrt{x_i}\sqrt{x_i} + \sum_{i=1}^{6} C_3 \sqrt{x_i} x_i = \sum_{i=1}^{6} y_i \sqrt{x_i}$$

$$\sum_{i=1}^{6} C_1 e^{0.8x_i} x_i + \sum_{i=1}^{6} C_2 \sqrt{x_i} x_i + \sum_{i=1}^{6} C_3 x_i x_i = \sum_{i=1}^{6} y_i x_i$$

These equations can be rewritten in matrix form:

$$\begin{bmatrix} \sum_{i=1}^{6} e^{0.8x_i} e^{0.8x_i} & \sum_{i=1}^{6} e^{0.8x_i}\sqrt{x_i} & \sum_{i=1}^{6} e^{0.8x_i} x_i \\ \sum_{i=1}^{6} e^{0.8x_i}\sqrt{x_i} & \sum_{i=1}^{6} \sqrt{x_i}\sqrt{x_i} & \sum_{i=1}^{6} \sqrt{x_i} x_i \\ \sum_{i=1}^{6} e^{0.8x_i} x_i & \sum_{i=1}^{6} \sqrt{x_i} x_i & \sum_{i=1}^{6} x_i x_i \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{6} y_i e^{0.8x_i} \\ \sum_{i=1}^{6} y_i \sqrt{x_i} \\ \sum_{i=1}^{6} y_i x_i \end{bmatrix}$$

This system is solved using the following MATLAB script. The script solves the system in the form $[X][C] = [Y]$, where $[X]$ is the 3×3 matrix with the summation terms, $[C]$ is the 3×1 vector of the unknowns and $[Y]$ is the 3×1 vector on the right-hand side.

```
clear all; clc;
x=0.5:0.5:2.5
xf=0.5:0.1:2.5;
%yf=0.2*exp(0.8*xf)-7*sqrt(xf)+3*xf
y=[-3.1 -3.6 -3.4 -2.8 -1.9]
F1=@ (x) exp(0.8*x);
F2=@ (x) sqrt(x);
F3=@ (x) x;
a(1,1)=sum(F1(x).*F1(x));
a(1,2)=sum(F1(x).*F2(x));
a(1,3)=sum(F1(x).*F3(x));
a(2,1)=sum(F2(x).*F1(x));
a(2,2)=sum(F2(x).*F2(x));
a(2,3)=sum(F2(x).*F3(x));
a(3,1)=sum(F3(x).*F1(x));
a(3,2)=sum(F3(x).*F2(x));
a(3,3)=sum(F3(x).*F3(x));
b(1,1)=sum(y.*F1(x));
b(2,1)=sum(y.*F2(x));
b(3,1)=sum(y.*F3(x));
c=a\b
```

When the program is executed, the vector $[C]$ is displayed in the Command Window:

```
c =
    0.2884
   -7.0514
    2.8538
```

Thus, the equation that best fits the data is $y = 0.2884e^{0.8x} + (-7.0514)\sqrt{x} + 2.8538x$.

5.12 The power generated by a windmill varies with the wind speed. In an experiment, the following five measurements were obtained:

Wind Speed (mph)	14	22	30	38	46
Electric Power (W)	320	490	540	500	480

Determine the fourth-order polynomial in the Lagrange form that passes through the points. Use the polynomial to calculate the power at a wind speed of 26 mph.

Solution

Lagrange polynomials are given by Eq.(5.45):

$$f(x) = \sum_{i=1}^{n} y_i L_i(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

Here, the fuel economy is y and the speed is x. Thus,

$$y(x) = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x) + y_4 L_4(x) + y_5 L_5(x)$$

where

$$L_1(x) = \frac{(x-22)(x-30)(x-38)(x-46)}{(14-22)(14-30)(14-38)(14-46)} = \frac{(x-22)(x-30)(x-38)(x-46)}{98304}$$

$$L_2(x) = \frac{(x-14)(x-30)(x-38)(x-46)}{(22-14)(22-30)(22-38)(22-46)} = -\frac{(x-14)(x-30)(x-38)(x-46)}{24576}$$

$$L_3(x) = \frac{(x-14)(x-22)(x-38)(x-46)}{(30-14)(30-22)(30-38)(30-46)} = \frac{(x-14)(x-22)(x-38)(x-46)}{16384}$$

$$L_4(x) = \frac{(x-14)(x-22)(x-30)(x-46)}{(38-14)(38-22)(38-30)(38-46)} = -\frac{(x-14)(x-22)(x-30)(x-46)}{24576}$$

$$L_5(x) = \frac{(x-14)(x-22)(x-30)(x-38)}{(46-14)(46-22)(46-30)(46-38)} = \frac{(x-14)(x-22)(x-30)(x-38)}{98304}$$

Therefore, the 4th order polynomial is:

$$y(x) = \frac{320(x-22)(x-30)(x-38)(x-46)}{98304} - \frac{490(x-14)(x-30)(x-38)(x-46)}{24576} + \frac{540(x-14)(x-22)(x-38)(x-46)}{16384} - \frac{500(x-14)(x-22)(x-30)(x-46)}{24576} + \frac{480(x-14)(x-22)(x-30)(x-38)}{98304}$$

Evaluating the polynomial at $x = 26 \text{ mph}$:

$$y(26) = \frac{320(26-22)(26-30)(26-38)(26-46)}{98304} - \frac{490(26-14)(26-30)(26-38)(26-46)}{24576}$$
$$+ \frac{540(26-14)(26-22)(26-38)(26-46)}{16384} - \frac{500(26-14)(26-22)(26-30)(26-46)}{24576}$$
$$+ \frac{480(26-14)(26-22)(26-30)(26-38)}{98304} = 530$$

5.13 Determine the fourth-order Newton's interpolating polynomial that passes through the data points given in Problem 5.12. Use the polynomial to calculate the power at a wind speed of 26 mph.

Solution

Write the data in column form as follows, and use the method of divided differences:

x y

14 320

$$\frac{(490-320)}{(22-14)} = 21.25$$

22 490

$$\frac{(6.25-21.25)}{(30-14)} = -0.9375$$

$$\frac{(540-490)}{(30-22)} = 6.25 \qquad \frac{(-0.70313-(-0.9375))}{(38-14)} = 0.0097654$$

30 540

$$\frac{(-5-6.25)}{(38-22)} = -0.70313 \qquad \frac{(0.035808-0.0097654)}{(46-14)} = 8.1383\times 10^{-4}$$

$$\frac{(500-540)}{(38-30)} = -5 \qquad \frac{(0.15625--0.70313)}{(46-22)} = 0.035808$$

38 500

$$\frac{(-2.5--5)}{(46-30)} = 0.15625$$

$$\frac{(480-500)}{(46-38)} = -2.5$$

46 480

Thus, the polynomial is

$$y(x) = 320+21.25(x-14)-0.9375(x-14)(x-22)+0.0097654(x-14)(x-22)(x-30)$$
$$+8.1383\times 10^{-4}(x-14)(x-22)(x-30)(x-38)$$

Evaluating at $x = 26$ mph :

$$y(26) = 320 + 21.25(26-14) - 0.9375(26-14)(26-22) + 0.0097654(26-14)(26-22)(26-30)$$
$$+ 8.1383 \times 10^{-4}(26-14)(26-22)(26-30)(26-38) = 530$$

5.14 The following data is given:

x	0.5	1	2	3	4	6
y	2	3.5	4.5	3.5	4	6

(*a*) Write the polynomial in Lagrange form that passes through the points; then use it to calculate the interpolated value of y at $x = 3.5$.

(*b*) Write the polynomial in Newton's form that passes through the points; then use it to calculate the interpolated value of y at $x = 3.5$.

Solution

(*a*) Following the form of Eq. (5.44), the Lagrange polynomial for the six given data points is:

$$f(x) = \frac{(x-1)(x-2)(x-3)(x-4)(x-6)}{(0.5-1)(0.5-2)(0.5-3)(0.5-4)(0.5-6)}2 + \frac{(x-0.5)(x-2)(x-3)(x-4)(x-6)}{(1-0.5)(1-2)(1-3)(1-4)(1-6)}3.5$$
$$+\frac{(x-0.5)(x-1)(x-3)(x-4)(x-6)}{(2-0.5)(2-1)(2-3)(2-4)(2-6)}4.5 + \frac{(x-0.5)(x-1)(x-2)(x-4)(x-6)}{(3-0.5)(3-1)(3-2)(3-4)(3-6)}3.5$$
$$+\frac{(x-0.5)(x-1)(x-2)(x-3)(x-6)}{(4-0.5)(4-1)(4-2)(4-3)(4-6)}4 + \frac{(x-0.5)(x-1)(x-2)(x-3)(x-4)}{(6-0.5)(6-1)(6-2)(6-3)(6-4)}6$$

The interpolated value of y at $x = 3.5$ is obtained by substituting this value of x into the polynomial:

$$f(3.5) = \frac{(3.5-1)(3.5-2)(3.5-3)(3.5-4)(3.5-6)}{(0.5-1)(0.5-2)(0.5-3)(0.5-4)(0.5-6)}2 + \frac{(3.5-0.5)(3.5-2)(3.5-3)(3.5-4)(3.5-6)}{(1-0.5)(1-2)(1-3)(1-4)(1-6)}3.5$$
$$+\frac{(3.5-0.5)(3.5-1)(3.5-3)(3.5-4)(3.5-6)}{(2-0.5)(2-1)(2-3)(2-4)(2-6)}4.5 + \frac{(3.5-0.5)(3.5-1)(3.5-2)(3.5-4)(3.5-6)}{(3-0.5)(3-1)(3-2)(3-4)(3-6)}3.5$$
$$+\frac{(3.5-0.5)(3.5-1)(3.5-2)(3.5-3)(3.5-6)}{(4-0.5)(4-1)(4-2)(4-3)(4-6)}4 + \frac{(3.5-0.5)(3.5-1)(3.5-2)(3.5-3)(3.5-4)}{(6-0.5)(6-1)(6-2)(6-3)(6-4)}6$$

$$f(3.5) = -0.13 + 0.6563 - 1.7578 + 3.2813 + 1.3393 - 0.0256 = 3.3635$$

(b) Newton's polynomial for the given points has the form:

$$f(x) = a_1 + a_2(x-0.5) + a_3(x-0.5)(x-1) + a_4(x-0.5)(x-1)(x-2) + a_5(x-0.5)(x-1)(x-2)(x-3)$$
$$+ a_6(x-0.5)(x-1)(x-2)(x-3)(x-4)$$

The coefficients can be found by constructing a divided difference table:

x_i	y_i	$a_1 = 2$	$a_2 = 3$	$a_3 = -1.3333$	$a_4 = 0.13332$	$a_5 = 0.12856$	$a_6 = -0.049889$
0.5	2						
			$\frac{3.5-2}{1-0.5} = 3$				
1	3.5			$\frac{1-3}{2-0.5} = -1.3333$			
			$\frac{4.5-3.5}{2-1} = 1$		$\frac{-1--1.3333}{3-0.5} = 0.13332$		
2	4.5			$\frac{-1-1}{3-1} = -1$		$\frac{0.5833-0.13332}{4-0.5} = 0.12856$	
			$\frac{3.5-4.5}{3-2} = -1$		$\frac{0.75--1}{4-1} = 0.5833$		$\frac{-0.14583-0.12856}{6-0.5} = -0.049889$
3	3.5			$\frac{0.5--1}{4-2} = 0.75$		$\frac{-0.14584-0.5833}{6-1} = -0.14583$	
			$\frac{4-3.5}{4-3} = 0.5$		$\frac{0.16666-0.75}{6-2} = -0.14584$		
4	4			$\frac{1-0.5}{6-3} = 0.16666$			
			$\frac{6-4}{6-4} = 1$				
6	6						

With the coefficients determined, the polynomial is

$$f(x) = 2 + 3(x-0.5) + (-1.3333)(x-0.5)(x-1) + 0.13332(x-0.5)(x-1)(x-2)$$
$$+ 0.12856(x-0.5)(x-1)(x-2)(x-3) + (-0.049889)(x-0.5)(x-1)(x-2)(x-3)(x-4)$$

The interpolated value of y at $x = 3.5$ is obtained by substituting this value of x into the polynomial:

$$f(3.5) = 2 + 3(3.5-0.5) + (-1.3333)(3.5-0.5)(3.5-1) + 0.13332(3.5-0.5)(3.5-1)(3.5-2)$$
$$+ 0.12856(3.5-0.5)(3.5-1)(3.5-2)(3.5-3) + (-0.049889)(3.5-0.5)(3.5-1)(3.5-2)(3.5-3)(3.5-4)$$

$$f(3.5) = 3.3636$$

5.15 Use linear splines interpolation with the data in Problem 5.12 to calculate the power at the following wind speeds.
(*a*) 26 mph (*b*) 42 mph.

Solution

The data in Problem 5.12 is

Wind Speed (mph)	14	22	30	38	46
Electric Power (W)	320	490	540	500	480

There are five points and thus four splines. Using Eq.(5.65), the equations of the splines are:

$$f_i(x) = \frac{(x - x_{i+1})}{(x_i - x_{i+1})}y_i + \frac{(x - x_i)}{(x_{i+1} - x_i)}y_{i+1}$$

(*a*) To estimate the fuel economy at 26 mph, it is only necessary to calculate the linear spline between the points (22 mph, 30 mpg) and (490 mph, 540 mpg):

$$(x) = \frac{(x-30)}{(22-30)}490 + \frac{(x-22)}{(30-22)}540 = -\frac{490}{8}(x-30) + \frac{540}{8}(x-22)$$

To find the fuel economy at 26 mph, substitute 26 mph for *x* in the equation:

$$f(30) = -\frac{490}{8}(26-30) + \frac{540}{8}(26-22)$$

$$f(26) = 515 \text{ mpg}$$

(*b*) To estimate the fuel economy at 42 mph, calculate the linear spline between the points (38 mph, 46 mpg) and (500 mph, 480 mpg):

$$f(x) = \frac{(x-46)}{(38-46)}500 + \frac{(x-38)}{(46-38)}480 = -62.5(x-46) + 60(x-38)$$

To find the fuel economy at 42 mph, substitute 42 mph for *x* in the equation:

$$f(42) = -62.5(42-46) + 60(42-38)$$

$$f(65) = 490 \text{ mpg}$$

5.16 Use quadratic splines interpolation with the data in Problem 5.12 to calculate the power at the following wind speeds.
(*a*) 26 mph (*b*) 42 mph.

Solution

The data in Problem 5.12 is:

Wind Speed (mph)	14	22	30	38	46
Electric Power (W)	320	490	540	500	480

There are five points, and therefore four splines. The quadratic equation for the *i*th spline is

$$f_i(x) = a_i x^2 + b_i x + c_i$$

There are four polynomials with three coefficients each, for a total of 12 coefficients. These coefficients are $a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3, a_4, b_4$ and c_4. The coefficient a_1 is set to zero (i.e. the second derivative at the first point is set to zero, see condition 3 on p. 213). A system of 11 linear equations can be written to find the other 11 unknowns.
Eight equations are obtained by applying Eqs. (5.67) and (5.68) to the endpoints of each interval:

$i = 1$ $$a_1 x_1^2 + b_1 x_1 + c_1 = y_1 \rightarrow b_1 14 + c_1 = 320$$

$$a_1 x_2^2 + b_1 x_2 + c_1 = y_2 \rightarrow b_1 22 + c_1 = 490$$

$i = 2$ $$a_2 x_2^2 + b_2 x_2 + c_2 = y_2 \rightarrow a_2 22^2 + b_2 22 + c_2 = 490$$

$$a_2 x_3^2 + b_2 x_3 + c_2 = y_3 \rightarrow a_2 30^2 + b_2 30 + c_2 = 540$$

$i = 3$ $$a_3 x_3^2 + b_3 x_3 + c_3 = y_3 \rightarrow a_3 30^2 + b_3 30 + c_3 = 540$$

$$a_3 x_4^2 + b_3 x_4 + c_3 = y_3 \rightarrow a_3 38^2 + b_3 38 + c_3 = 500$$

$i = 4$ $$a_4 x_4^2 + b_4 x_4 + c_4 = y_4 \rightarrow a_4 38^2 + b_4 38 + c_4 = 500$$

$$a_4 x_5^2 + b_4 x_5 + c_4 = y_5 \rightarrow a_4 46^2 + b_4 46 + c_4 = 480$$

The final three equations are obtained from the condition that at the interior knots the slopes (first derivatives) of the polynomials from adjacent intervals are equal. From Eq. (5.70),

$i = 2$ $$2a_1 x_2 + b_1 = 2a_2 x_2 + b_2 \rightarrow b_1 - 2(22)a_2 - b_2 = 0$$

$i = 3$ $$2a_2 x_3 + b_2 = 2a_3 x_3 + b_3 \rightarrow 2(30)a_2 + b_2 - 2(30)a_3 - b_3 = 0$$

$i = 4$ $$2a_3 x_4 + b_3 = 2a_4 x_4 + b_4 \rightarrow 2(38)a_3 + b_3 - 2(38)a_4 - b_4 = 0$$

The system of 11 linear equations can be written in matrix form:

$$\begin{bmatrix} 14 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 22^2 & 22 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30^2 & 30 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30^2 & 30 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 38^2 & 38 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 38^2 & 38 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50^2 & 50 & 1 \\ 1 & 0 & -44 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 60 & 1 & 0 & -60 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 76 & 1 & 0 & -76 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 320 \\ 490 \\ 490 \\ 540 \\ 540 \\ 500 \\ 500 \\ 480 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This system of equations can be solved using the following MATLAB script:

```
clear all
A = [14 1 0 0 0 0 0 0 0 0 0; 22 1 0 0 0 0 0 0 0 0 0; 0 0 22^2 22 1 0 0 0 0 0 0
    0 0 30^2 30 1 0 0 0 0 0 0; 0 0 0 0 0 30^2 30 1 0 0 0; 0 0 0 0 0 38^2 38 1 0
0 0
    0 0 0 0 0 0 0 0 38^2 38 1; 0 0 0 0 0 0 0 0 50^2 50 1; 1 0 -44 -1 0 0 0 0 0
0 0
    0 0 60 1 0 -60 -1 0 0 0 0; 0 0 0 0 0 76 1 0 -76 -1 0]
B = [320; 490; 490; 540; 540; 500; 500; 480; 0; 0; 0];
coefficients = (A\B)'
c = (A\B)
```

When the program is executed, the coefficients are displayed in the Command Window:

```
c =
  1.0e+003 *
   0.021250000000000
   0.022500000000000
  -0.001875000000000
   0.103750000000000
```

```
-0.885000000000000
 0.000468750000000
-0.036875000000000
 1.224375000000000
-0.000034722222222
 0.001388888888889
 0.497361111111110
```

The coefficients for the quadratic splines are as follows:
$a_1 = 0$, $b_1 = 21.25$, $c_1 = 22.5$, $a_2 = -1.875$, $b_2 = 103.75$, $c_2 = -885$, $a_3 = 0.46875$, $b_3 = -36.875$, $c_3 = 1224.4$, $a_4 = -0.034722$, $b_4 = 1.3889$, $c_4 = 497.36$.

(*a*) To find the power at 26 mph, consider the polynomial between the points (22 mph, 490 W) and (30 mph, 520 W). This corresponds to $i = 2$, so this polynomial has the coefficients a_2, b_2 and c_2:

$$f_2(x) = -1.875x^2 + 103.75x - 885$$

To calculate the fuel economy at 26 mph, substitute $x = 26$ mph into the equation:

$$f_2(26) = -1.875 \cdot 26^2 + 103.75 \cdot 26 - 885$$

$$f_2(26) = 545$$

(*b*) To find the fuel economy at 42 mph, consider the polynomial between the points (38 mph, 500 W) and (46 mph, 480 W). This corresponds to $i = 4$, so this polynomial has the coefficients a_4, b_4 and c_4:

$$f_4(x) = -0.034722x^2 + 1.3889x + 497.36$$

To calculate the fuel economy at 42 mph, substitute $x = 42$ mph into the equation:

$$f_4(42) = -0.034722 \cdot 42^2 + 1.3889 \cdot 42 + 497.36$$

$$f_4(42) = 494.44$$

5.17 Use natural cubic splines interpolation (based on Lagrange-form polynomials [Eqs. (5.86)–(5.89)]) with the data in Problem 5.12; to calculate the power at the following wind speeds.
(*a*) 26 mph (*b*) 42 mph.

Solution

The data in Problem 5.12 is:

Wind Speed (mph)	14	22	30	38	46
Electric Power (W)	320	490	540	500	480

we seek to fit the function $f(x)$ given by Eq.(5.88) for the electric power in W as a function of the speed x in mph using cubic splines. Since the x values are equally spaced, $h_i = h = 8$ mph Eq.(5.89) can be simplified to:

$$a_i + 4a_{i+1} + a_{i+2} = 6\left[\frac{y_{i+2} - 2y_{i+1} + y_i}{h^2}\right] \qquad \text{for} \qquad i = 1, 2, 3$$

For natural cubic splines, $a_1 = 0$ and $a_5 = 0$. Thus, the above system of equations reduces to:

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -11.25 \\ -8.4375 \\ 1.875 \end{bmatrix}$$

This can be solved in the MATLAB command window as follows:

```
>> A=[4 1 0;1 4 1;0 1 4]; b=[-11.25;-8.4375;1.875];
>> x=A\b
x =
   -2.3772
   -1.7411
    0.9040
>>
```

Thus, $\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix} = \begin{bmatrix} 0 & -2.3772 & -1.7411 & 0.9040 & 0 \end{bmatrix}$. These are the values of the second derivatives at each point. The approximating cubic spline functions for each interval are found from Eq.(5.88) for the case $h_i = h = 8$ mph .

(*a*) For 26 mph the interval is between $[x_2, x_3]$:

$$f_2(x) = \frac{a_2}{6(8)}(30 - x)^3 + \frac{a_3}{6(8)}(x - 22)^3 + \left[\frac{490}{8} - \frac{a_2(8)}{6}\right](30 - x) + \left[\frac{540}{8} - \frac{a_3(8)}{6}\right](x - 22)$$

or $f_2(x) = (-0.049525)(30 - x)^3 + (-0.036335)(x - 22)^3 + [64.42](30 - x) + [69.82](x - 22)$. Substituting $x = 26\ \text{mph}$ yields $f_2(26) = 531.5\ \text{W}$.

(b) For 42 mph the interval is between $[x_4, x_5]$:

$$f_4(x) = \frac{a_4}{6(8)}(46-x)^3 + \frac{a_5}{6(8)}(x-38)^3 + \left[\frac{500}{8} - \frac{a_4(8)}{6}\right](46-x) + \left[\frac{480}{8} - \frac{a_5(8)}{6}\right](x-38)$$

or $f_4(x) = (0.018833)(46-x)^3 + [61.29](46-x) + [60](x-38)$. Substituting $x = 42\ \text{mph}$ yields $f_4(42) = 486.3\ \text{W}$.

5.18 Modify the MATLAB user-defined function `LinearRegression` in Program 5-1. In addition to determining the constants a_1 and a_0, the modified function should also calculate the overall error E according to Eq. (5.6). Name the function `[a, Er] = LinReg(x,y)`. The input arguments `x` and `y` are vectors with the coordinates of the data points. The output argument `a` is a two-element vector with the values of the constants a_1 and a_0. The output argument `Er` is the value of the overall error.
(*a*) Use the function to solve Example 5-1.
(*b*) Use the function to solve Problem 5.2.

Solution

The listing of the user-defined function `LinReg` is:

```
function [a,Er] = LinReg(x, y)
% LinReg calculates the coefficients a1 and a0 of the linear
% equation y = a1*x + a0 that best fits n data points, and the overall
% error according to Eq. (5.6).
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variables:
% a   Two elements vector with the coefficients a1 and a0.
% Er  The overall error.

nx = length(x);
ny = length(y);
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    a = 'Error';
    Er = 'Error';
else
    Sx = sum(x);
    Sy = sum(y);
    Sxy = sum(x.*y);
    Sxx = sum(x.^2);
    a1 = (nx*Sxy - Sx*Sy)/(nx*Sxx - Sx^2);
    a0 = (Sxx*Sy - Sxy*Sx)/(nx*Sxx - Sx^2);
    a=[a1; a0];
    % Eq. (5.6)
```

```
    Er=sum((y-(a1.*x+a0)).^2);
end
```

(*a*) The user-defined function `LinReg` is used in the Command Window to solve Problem 5-1.

```
>> x=[1 2 4 7 10 12 15];
>> y=[3 5 7 11 14 17 21];
>> [a,Er] = LinReg(x, y)
a =
    1.2466
    2.0606
Er =
    0.6766
```

(*b*) The user-defined function `LinReg` is used in the Command Window to solve Problem 5-2.

```
>> x=[-6 -4 -1 0 3 5 8];
>> y=[18 13 6 4 -1 -8 -15];
>> [a,Er] = LinReg(x, y)
a =
   -2.3140
    4.0814
Er =
    4.3256
```

5.19 Write a MATLAB user-defined function that determines the best fit of an exponential function of the form $y = be^{mx}$ to a given set of data points. Name the function `[b m]= ExpoFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output arguments `b` and `m` are the values of the coefficients. The function `ExpoFit` should use the approach that is described in Section 5.3 for determining the value of the coefficients. Use the function to solve Problem 5.7.

Solution

The listing of the user-defined function `ExpoFit` is:

```
function [b,m] = ExpoFit(x, y)
% ExpoFit calculates the coefficients b and m of the exponential
% equation y = b*exp(m*x) that best fits n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variables:
% b   The coefficient b.
% m   The coefficient m.

nx = length(x);
ny = length(y);
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    b = 'Error';
    m = 'Error';
else
    Y=log(y); X=x;
    SX = sum(X);
    SY = sum(Y);
    SXY = sum(X.*Y);
    SXX = sum(X.^2);
    a1 = (nx*SXY - SX*SY)/(nx*SXX - SX^2);
    a0 = (SXX*SY - SXY*SX)/(nx*SXX - SX^2);
```

```
    b=exp(a0);
    m=a1;
end
```

The user-defined function `ExpoFit` is used to solve Problem 5-7 in the Command Window:

```
>> h=[1 4000 8000 12000 16000 20000];
>> ro=[1.225 0.82 0.525 0.309 0.168 0.092];
>> [b,m] = ExpoFit(h, ro)
b =
    1.3565
m =
 -1.3022e-004
>> ro7000=b*exp(m*7000)
ro7000 =
    0.5452
```

5.20 Write a MATLAB user-defined function that determines the best fit of a power function of the form $y = bx^m$ to a given set of data points. Name the function `[b m]= PowerFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output arguments `b` and `m` are the values of the coefficients. The function `PowerFit` should use the approach that is described in Section 5.3 for determining the value of the coefficients. Use the function to solve Problem 5.3.

Solution

The listing of the user-defined function `PowerFit` is:

```
function [b,m] = PowerFit(x, y)
% PowerFit calculates the coefficients b and m of the exponential
% equation y = b*x^m that best fits n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variables:
% b   The coefficient b.
% m   The coefficient m.

nx = length(x);
ny = length(y);
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    b = 'Error';
    m = 'Error';
else
    Y=log(y); X=log(x);
    SX = sum(X);
    SY = sum(Y);
    SXY = sum(X.*Y);
    SXX = sum(X.^2);
    a1 = (nx*SXY - SX*SY)/(nx*SXX - SX^2);
    a0 = (SXX*SY - SXY*SX)/(nx*SXX - SX^2);
    b=exp(a0);
    m=a1;
end
```

The user-defined function `PowerFit` is used in the Command Window to fit a power function to the data from Problem 5.3:

```
>> y=[1845 1871 1915 1954 2000];
>> p=[20 30 100 160 280];
>> [b,m] = PowerFit(y, p)
b =
  4.4839e-110
m =
   33.8877
>> p1970=b*1970^m
p1970 =
  196.5490
```

The power function that best fits the data from Problem 5.3 is: $y = 4.4839 \times 10^{-110} x^{33.8877}$.

5.21 Write a MATLAB user-defined function that determines the coefficients of a quadratic polynomial, $f(x) = a_2x^2 + a_1x + a_0$, that best fits a given set of data points. Name the function `a = QuadFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output argument `a` is a three-element vector with the values of the coefficients a_2, a_1 and a_0.
(*a*) Use the function to find the quadratic polynomial that best fits the data in Example 5-2.
(*b*) Write a program in a script file that plots the data points and the curve of the quadratic polynomial that best fits the data.

Solution

The listing of the user-defined function `QuadFit` is:

```
function a = QuadFit(x, y)
% QuadFit calculates the coefficients a2,a1 and a0 of the quadratic
% polynomial that best fits n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variables:
% a    A vector with the values of a2,a1 and a0
% The vector [a] is determined by solving the system [X][a]=[Y]

nx = length(x);
ny = length(y);
m = 3; %number of coefficients
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    a = 'Error';
else
    for i=1:(m+1)
    xsum(i)=sum(x.^(i));
    end
    % First row of matrix [X] and first element of column vector [Y]
    X(1,1)=nx;
```

```
    Y(1,1)=sum(y);
    for j=2:m
        X(1,j)=xsum(j-1);
    end
    % Rows 2 and 3 of matrix [X] and column vector [Y]
    for i=2:m
        for j=1:m
            X(i,j)=xsum(j+i-2);
        end
       Y(i,1)=sum(x.^(i-1).*y);
    end
    a=X\Y;
end
```

(*a*) The user-defined function `QuadFit` is used in the Command Window to find the quadratic polynomial that best fits the data from Example 5-2:

```
>> t=[2 4 6 8 10 12 14 16 18 20 22 24 26 28 30];
>> V=[9.7 8.1 6.6 5.1 4.4 3.7 2.8 2.4 2.0 1.6 1.4 1.1 0.85 0.69 0.6];
>> a = QuadFit(t, V)
a =
   10.7215
   -0.7480
    0.0141
```

The quadratic polynomial that best fits the data from Example 5-2 is:

$$f(x) = 0.0141x^2 - 0.7480x + 10.7215$$

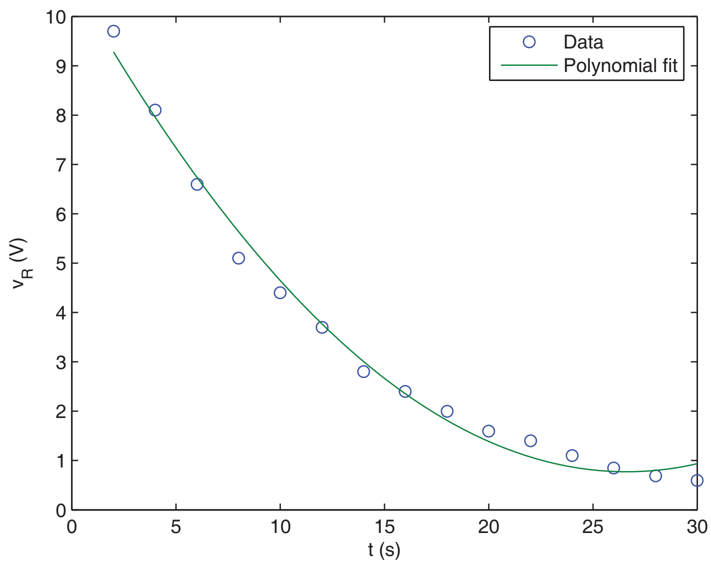
(*b*) The following program in a script file plots the data points and the curve of the quadratic polynomial that best fits the data.

```
clear all; close all; clc;
t=[2 4 6 8 10 12 14 16 18 20 22 24 26 28 30];
V=[9.7 8.1 6.6 5.1 4.4 3.7 2.8 2.4 2.0 1.6 1.4 1.1 0.85 0.69 0.6];
```

```
n=length(t);
a = QuadFit(t, V);

tt=linspace(t(1),t(n),50);
VV=a(3).*tt.^2+a(2).*tt+a(1);
plot(t,V,'o',tt,VV,':');
xlabel('t (s)'); ylabel('v_R (V)');
legend('Data','Polynomial fit');
```

When the program is executed, the following plot is displayed in the Figure Window:



5.22 Write a MATLAB user-defined function that determines the coefficients of a cubic polynomial, $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, that best fits a given set of data points. The function should also calculate the overall error E according to Eq. (5.21). Name the function `[a,Er] = CubicPolyFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output argument `a` is a four-element vector with the values of the coefficients a_3, a_2, a_1, and a_0. The output argument `Er` is the value of the overall error.

(*a*) Use `CubicPolyFit` to determine the cubic polynomial that best fits the data in Example 5-3.

(*b*) Write a program in a script file that plots the data points and the curve of the cubic polynomial that best fits the data.

Solution

The listing of the user-defined function `CubicPolyFit` is:

```
function [a,Er] = CubicPolyFit(x, y)
% CubicPolyFit calculates the coefficients a3,a2,a1 and a0 of the cubic
% polynomial that best fits n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variables:
% a    A vector with the values of a3,a2,a1 and a0
% Er   The overall error
% The vector [a] is determined by solving the system [X][a]=[Y]

nx = length(x);
ny = length(y);
m = 4; %number of coefficients
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    a = 'Error';
else
    for i = 1:(2*m-2)
        xsum(i) = sum(x.^(i));
```

```
    end
    % First row of matrix [X] and first element of column vector [Y]
    X(1,1) = nx;
    Y(1,1) = sum(y);
    % Remaining rows of [X] and [Y]
    for j = 2:m
        X(1,j) = xsum(j - 1);
    end
    for i = 2:m
        for j = 1:m
            X(i,j) = xsum(j + i - 2);
        end
        Y(i,1) = sum(x.^(i - 1).*y);
    end
    % Solve for [a] and Er
     a = (X\Y)';
     Er=sum((y-a(4).*x.^3-a(3).*x.^2-a(2).*x-a(1)).^2);
end
```

(*a*) The user-defined function `CubicPolyFit` can be used in the Command Window to find the cubic polynomial that best fits the data from Example 5-3 and the error associated with that polynomial:

```
>> x=[0 0.4 0.8 1.2 1.6 2 2.4 2.8 3.2 3.6 4 4.4 4.8 5.2 5.6 6];
>> y=[0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
>> [a,Er] = CubicPolyFit(x,y)
a =
    2.8930   -1.9882    1.8030   -0.0541
Er =
   69.9256
```

The cubic polynomial which provides the best fit with the data is:

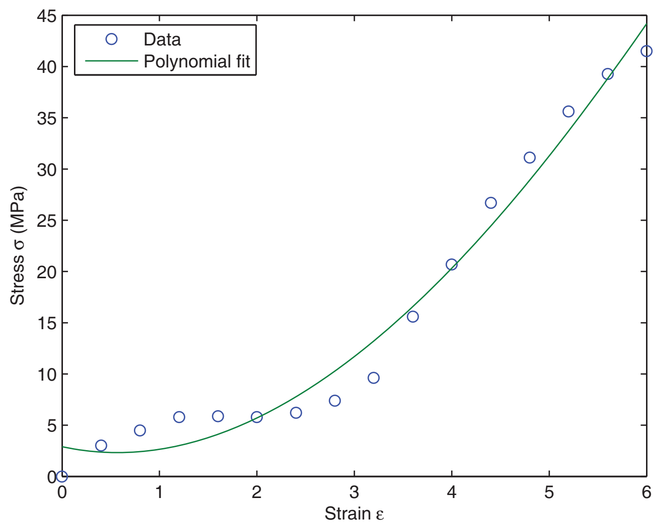
$$f(x) = -0.0541x^3 + 1.8030x^2 - 1.9882x + 2.8930$$

(*b*) The following program in a script file plots the data points and the curve of the cubic polynomial that best fits the data.

```
clear all; close all; clc;
x=[0 0.4 0.8 1.2 1.6 2 2.4 2.8 3.2 3.6 4 4.4 4.8 5.2 5.6 6];
y=[0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n=length(x);
[a,Er] = CubicPolyFit(x,y)

xx=linspace(x(1),x(n),50);
yy=a(4).*xx.^3+a(3).*xx.^2+a(2).*xx+a(1);
plot(x,y,'o',xx,yy);
xlabel('Strain \epsilon'); ylabel('Stress \sigma (MPa)');
legend('Data','Polynomial fit',2);
```

When the program is executed, the following plot is displayed in the Figure Window:



5.23 Write a MATLAB user-defined function for interpolation with natural cubic splines. Name the function `Yint = CubicSplines(x,y,xint)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and `xint` is the *x* coordinate of the interpolated point. The output argument `Yint` is the *y* value of the interpolated point.

(*a*) Use the function with the data in Example 5-8 for calculating the interpolated value at $x = 12.7$.

(*b*) Use the function with the data in Problem 5.37 for calculating the enthalpy per unit mass at $T = 14000$ K and at $T = 24000$ K.

Solution

The listing of the user-defined function `CubicSplines` is:

```
function Yint=CubicSplines(x,y,xint)
% CubicSplines fits a set of cubic splines to n data points and then
% returns the interpolated value Yint at the desired coordinate Xint.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% xint The x-value where interpolation is desired
% Output variables:
% Yint The interpolated y-value at x=xint

n=length(x); interval=1;
if n ~= length(y)
    disp('ERROR: x and y do not have the same number of points');
    stop
end
% calculate h_i
for i = 1:n-1
    h(i) = x(i+1)-x(i);
end
%Start Thomas Algorithm
for i=2:n-2
    b(i)=h(i);
```

```
end
b(1)=0;
for i=1:n-3
    a(i)=h(i+1);
end
a(n-2)=0;
for i=1:n-2
    d(i)=2*(h(i)+h(i+1)); r(i)=6*(((y(i+2)-y(i+1))/h(i+1))-((y(i+1)-y(i))/
h(i)));
end
A=zeros(n-2,n-2);
for i=2:n-3
    A(i,i)=d(i); A(i,i+1)=a(i); A(i,i-1)=b(i);
end
A(1,2)=a(1); A(1,1)=d(1); A(n-2,n-3)=b(n-2); A(n-2,n-2)=d(n-2);
a(1)=a(1)/d(1);
r(1)=r(1)/d(1);
for i=2:n-3
    denom=d(i)-b(i)*a(i-1);
    if(denom==0), error('zero in denominator'), end
    a(i)=a(i)/denom;
    r(i)=(r(i)-b(i)*r(i-1))/denom;
end
r(n-2)=(r(n-2)-b(n-2)*r(n-3))/(d(n-2)-b(n-2)*a(n-3));
ans(n-2)=r(n-2);
for i=n-3:-1:1
    ans(i) = r(i) - a(i)*ans(i+1);
end
acoeff(1)=0; acoeff(n)=0;
for i=2:n-1
        acoeff(i)=ans(i-1);
```

```
end
for j=1:n-1
    if xint >= x(j) & xint <= x(j+1)
        interval=j;
    end
end
i=interval;
YintA=(acoeff(i)*((x(i+1)-xint)^3)/6/h(i));
YintB=(acoeff(i+1)*((xint-x(i))^3)/6/h(i));
YintC=((y(i)/h(i))-(acoeff(i)*h(i)/6))*(x(i+1)-xint);
YintD=((y(i+1)/h(i))-(acoeff(i+1)*h(i)/6))*(xint-x(i));
Yint=YintA+YintB+YintC+YintD;
```

(*a*) The user-defined function `CubicSplines` is used in the Command Window to calculate the interpolated value at $x = 12.7$ for the data in Example 5-8:

```
>> x=[8 11 15 18 22];
>> y=[5 9 10 8 7];
>> Yint=CubicSplines(x,y,12.7)
Yint =
   10.1189
>>
```

(*b*) The user-defined function is used in the Command Window to calculate the enthalpy per unit mass at $T = 14000$ K and at $T = 24000$ K using the data from Problem 5.27:

```
>> x=[5 7.5 10 12.5 15 17.5 20 22.5 25 27.5 30]*10^3;
>> y=[3.3 7.5 41.8 51.8 61 101.1 132.9 145.5 171.4 225.8 260.9];
>> Yint=CubicSplines(x,y,14000)
Yint =
   54.1160
>> Yint=CubicSplines(x,y,24000)
Yint =
  157.0608
```

>>

Thus, at $T = 14000$ K, the enthalpy per unit mass is $h = 54.1160$ MJ/kg. At $T = 24000$ K, the enthalpy per unit mass is $h = 157.0608$ MJ/kg.

5.24 Write a MATLAB user-defined function for spline interpolation that uses third-order Lagrange polynomials. Name the function `Yint = CubicLagSplines(x,y,xint)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and `xint` is the x coordinate of the interpolated point. The output argument `Yint` is the y value of the interpolated point. The function uses the following scheme for the interpolation. If `xint` is in the first interval of the data points, the function uses a second-order polynomial that passes through the first three data points. If `xint` is in the last interval of the data points, the function uses a second-order polynomial that passes through the last three data points. If `xint` is in any other interval, lets say interval i between point x_i and point x_{i+1}, the function uses a third-order polynomial for the interpolation. The third-order polynomial is written such that it passes through the data points: x_{i-1}, x_i, x_{i+1}, and x_{i+2}.

(*a*) Use the `CubicLagSplines` function with the data in Problem 5.12 to calculate the power at wind speeds of 26 mph and 42 mph.

(*b*) Use the `CubicLagSplines` function with the data in Example 5-3 to calculate the stress at strains of 0.2 and 3.

Solution

The listing of the user-defined function `CubicLagSplines` is:

```
function Yint=CubicLagSplines(x,y,Xint)
n=length(x);
for i=2:n
    if Xint < x(i)
        break
    end
end
if i == 2
    xa=x(1:3);
    ya=y(1:3);
    Yint=LagrangeINT(xa,ya,Xint);
elseif i == n
    xa=x(n-2:n);
    ya=y(n-2:n);
    Yint=LagrangeINT(xa,ya,Xint);
else
```

```
    xa=x(i-2:i+1);
    ya=y(i-2:i+1);
    Yint=LagrangeINT(xa,ya,Xint);
end


function Yint = LagrangeINT(x,y,Xint)
% LagrangeINT fits a Lagrange polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x  A vector with the x coordinates of the given points.
% y  A vector with the y coordinates of the given points.
% Xint  The x coordinate of the point to be interpolated.
% Output variable:
% Yint  The interpolated value of Xint.

n = length(x);
for i = 1:n
    L(i) = 1;
    for j = 1:n
        if j ~= i
            L(i)= L(i)*(Xint-x(j))/(x(i)-x(j));
        end
    end
end
Yint = sum(y.*L);
```

(*a*) A script file that solves Part (*a*):

```
clear; clc
S=[14 22 30 38 46];
P=[320 490 540 500 480];
```

```
% Interpolation for 26 mph
Pfor26=CubicLagSplines(S,P,26)
% Interpolation for 42 mph
Pfor42=CubicLagSplines(S,P,42)
```

When executes the following results are displayed in the Command Window:

```
Pfor26 =
  528.1250
Pfor42 =
  487.5000
```

(*b*) A script file that solves Part (*b*):

```
clear; clc
Strain=[0 0.4 0.8 1.2 1.6 2 2.4 2.8 3.2 3.6 4 4.4 4.8 5.2 5.6 6];
Stress=[0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
% Interpolation for strain 0.2
Stress02=CubicLagSplines(Strain,Stress,0.2)
% Interpolation for strain 3
Stress3=CubicLagSplines(Strain,Stress,3)
```

When executes the following results are displayed in the Command Window:

```
Stress02 =
    1.6875
Stress3 =
    8.200
```

5.25 A linear combination of three functions that is used for curve fitting has the form (see Section 5.8):

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + C_3 f_3(x)$$

Write a MATLAB user-defined function that determines the coefficients C_1, C_2, and C_3 that best fits a given set of data points. Name the function `C = NonLinCombFit(F1,F2,F3,x,y)`, where the input arguments `F1`, `F2`, and `F3` are handles of the three functions (user-defined or anonymous) $f_1(x)$, $f_2(x)$, and $f_3(x)$, and `x` and `y` are vectors with the coordinates of the data points. The output argument `C` is a three-element row vector with the values of the coefficients C_1, C_2, and C_3.
Use `NonLinCombFit` to solve Problem 5.38 (*a*). Write a program in a script file that uses `NonLinCombFit` and plots the data points and the curve of $F(x)$ that best fits the data.

Solution

The listing of the user-defined function `NonLinCombFits` is:

```
function c = NonLinCombFit(F1,F2,F3,x,y)
% NonLinComdFit determines the three coefficients C1, C2, C3 of the function
% C1*f1(x)+C2*f2(x)+C3*f3(x)that best fit given data points.
% Input arguments:
% F1  Anonymos function of f1(x).
% F2  Anonymos function of f2(x).
% F3  Anonymos function of f3(x).
% x  A vector with the x coordinates of the given points.
% x  A vector with the y coordinates of the given points.
% Output arguments:
% c  A column vector with the values of C1, C2, and C3.

a(1,1)=sum(F1(x).*F1(x));
a(1,2)=sum(F1(x).*F2(x));
a(1,3)=sum(F1(x).*F3(x));
a(2,1)=sum(F2(x).*F1(x));
a(2,2)=sum(F2(x).*F2(x));
a(2,3)=sum(F2(x).*F3(x));
a(3,1)=sum(F3(x).*F1(x));
```

```
a(3,2)=sum(F3(x).*F2(x));
a(3,3)=sum(F3(x).*F3(x));
b(1,1)=sum(y.*F1(x));
b(2,1)=sum(y.*F2(x));
b(3,1)=sum(y.*F3(x));
c=a\b;
```

A script file that solves Part (*a*) of Problem 5.38:

```
% HW5_25se Script
clear; clc
T=[595 623 761 849 989 1076 1146 1202 1382 1445 1562];
k=[2.12 3.12 14.4 30.6 80.3 131 186 240 489 604 868]*1e-20;
yk=log(k);
FA=@ (x) x./x;
FB=@ (x) log(x);
FC=@ (x) -1./x;
c = NonLinCombFit(FA,FB,FC,T,yk)
```

When executes the following results are displayed in the Command Window:

```
c =
  1.0e+003 *
  -0.052442723579328
   0.002121597543987
   3.815341691904303
```

5.26 The resistance R of a tungsten wire as a function of temperature can be modeled with the equation $R = R_0[1 + \alpha(T - T_0)]$, where R_0 is the resistance corresponding to temperature T_0, and α is the temperature coefficient of resistance. Determine R_0 and α such that the equation will best fit the following data. Use $T_0 = 20° C$.

T (°C)	20	100	180	260	340	420	500
R (Ω)	500	676	870	1060	1205	1410	1565

(*a*) Use the user-defined function `LinReg` developed in Problem 5.18.
(*b*) Use MATLAB's built-in function `polyfit`.

Solution

Rewriting the equation: $R = \alpha R_0 T + R_0(1 - \alpha T_0)$
Linear regression form: $R = a_1 T + a_0$
Thus: $a_1 = \alpha R_0$ and $a_0 = R_0(1 - \alpha T_0)$ or: $a_0 = R_0 - a_1 T_0$
Once a_1 and a_0 are calculated, R_0 and α are determined by $R_0 = a_0 + a_1 T_0$ and $\alpha = \frac{a_1}{R_0}$.

(*a*)
A script file that solves Part (*a*):

```
T0=20;
T=[20 100 180 260 340 420 500];
R=[500 676 870 1060 1205 1410 1565];
[a,Er] = LinReg(T, R);
R0=a(2)+a(1)*T0
Alpha=a(1)/R0
```

When the script is executed the following results are displayed in the Command Window:

```
R0 =
    5.053571428571428e+002
Alpha =
   0.004415194346290
```

(*b*)
A script file that solves Part (*b*):

```
T0=20;
```

```
T=[20 100 180 260 340 420 500];
R=[500 676 870 1060 1205 1410 1565];
p=polyfit(T,R,1);
R0=p(2)+p(1)*T0
Alpha=p(1)/R0
```

When the script is executed the following results are displayed in the Command Window:

```
R0 =
    5.053571428571431e+002
Alpha =
   0.004415194346290
```

5.27 Bacteria growth rate can be modeled with the equation $\ln N_t - \ln N_0 = \mu(t - t_0)$, where μ is the growth rate constant, and N_t and N_0 are the numbers of bacteria at times t and t_0, respectively. Determine μ and N_0 such that the equation will best fit the following data. Use $t_0 = 0$.

t (h)	0	2	4	6	8
N (cells/ml)	35	1990	70,800	2,810,000	141,250,000

(*a*) Use the user-defined function `LinReg` developed in Problem 5.18.
(*b*) Use MATLAB's built-in function `polyfit`.

Solution

Rewriting the equation: $\ln N_t = \mu t + \ln N_0 - \mu t_0$
Linear regression form: $y = a_1 t + a_0$
Thus: $y = \ln N_t$ $a_1 = \mu$ and $a_0 = \ln N_0 - \mu t_0$.
Once a_1 and a_0 are calculated, a_0 and α are determined by $\mu = a_1$ and $N_0 = e^{a_0 + \mu t_0}$.

(*a*)
A script file that solves Part (*a*):

```
clear, clc
t0=0;
t=0:2:8;
Nt=[35 1990 70800 2810000 141250000];
y=log(Nt);
[a,Er] = LinReg(t, y);
mu=a(1)
N0=exp(a(2)+mu*t0)
```

When the script is executed the following results are displayed in the Command Window:

```
mu =
   1.883709642829710
N0 =
  38.54703848340930
```

(*b*)

A script file that solves Part (*b*):

```
clear, clc
t0=0;
t=0:2:8;
Nt=[35 1990 70800 2810000 141250000];
y=log(Nt);
p=polyfit(t,y,1);
mu=p(1)
N0=exp(p(2)+mu*t0)
```

When the script is executed the following results are displayed in the Command Window:

```
mu =
   1.883709642829710
N0 =
  38.547038483409182
```

5.28 In an experiment for measuring the thermal expansion coefficient, a 2.5-m-long stainless steel bar is placed in an oven. Starting at $20°$C the temperature of the bar is increased in increments of $100°$C up to temperature of $820°$C, and the increase in length at each temperature is measured. The change in length ΔL is related to the temperature change ΔT by the function $\Delta L = \alpha L_o \Delta T$, where α is the thermal expansion coefficient, and L_o is the initial length. Use linear least-squares regression to determine the thermal expansion coefficient if the following data was measured:

ΔT ($°$C)	100	200	300	400	500	600	700	800
ΔL (mm)	4.2	8.9	17.3	14.8	23.5	28	30.8	34.2

(*a*) Use the user-defined function `LinReg` developed in Problem 5.18.
(*b*) Use MATLAB's built-in function `polyfit`.

Solution

(*a*)
A script file that solves Part (*a*):

```
clear, clc
L0=2.5;
DT=100:100:800;
DL=[4.2 8.9 17.3 14.8 23.5 28 30.8 34.2];
[a,Er] = LinReg(DT, DL);
Alpha=a(1)/L0
```

When the script is executed the following results are displayed in the Command Window:

```
Alpha =
   0.017157142857143
```

(*b*)
A script file that solves Part (*b*):

```
clear, clc
L0=2.5;
DT=100:100:800;
DL=[4.2 8.9 17.3 14.8 23.5 28 30.8 34.2];
p=polyfit(DT,DL,1);
```

```
Alpha=p(1)/L0
DTp=linspace(100,800,100);
DLp=Alpha*L0*DTp+p(2);
plot(DTp,DLp,DT,DL,'*')0
```

When the script is executed the following results are displayed in the Command Window:

```
Alpha =
   0.017157142857143
```

5.29 Measurements of thermal conductivity, k (W/m-K), of iron at various temperatures, T (K), are:

T (°K)	70	100	200	300	400	500	600	800	1000
k (W/m-K)	215	134	93	80.8	70	61.2	54.8	43.4	32.2

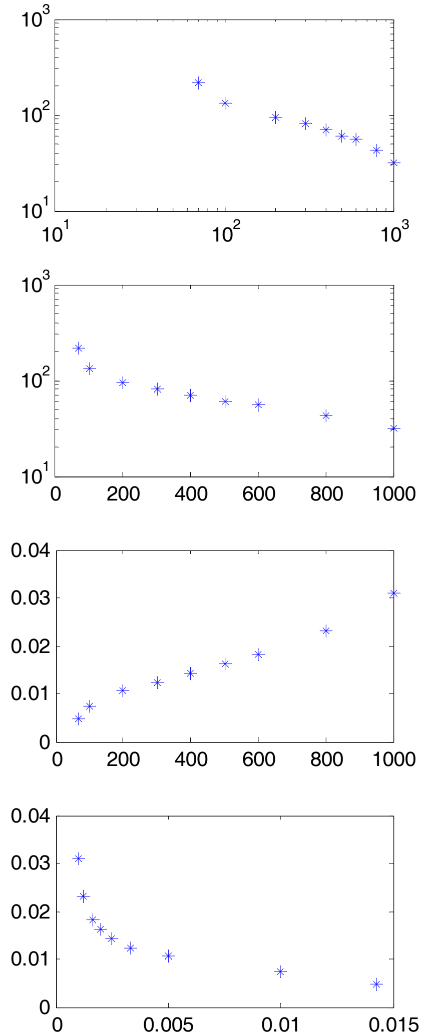
The data is to be fitted with a function of the form $k = f(T)$. Determine which of the nonlinear equations that are listed in Table 5-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

Solution

First, the following four plots are made. k vs. T on a log-log plot. k vs. T on a log k (vertical) linear T (horizontal) axes. $1/k$ vs. T with linear axes. $1/k$ vs. $1/T$ with linear axes.

```
clear, clc
T=[70 100:100:600 800 1000];
k=[215 134 93 80.8 70 61.2 54.8 43.4 32.2];
subplot(4,1,1)
loglog(T,k,'*')
subplot(4,1,2)
semilogy(T,k,'*')
subplot(4,1,3)
plot(T,1./k,'*')
subplot(4,1,4)
plot(1./T,1./k,'*')
```

When the script is executed, the following plots are displayed:



10^3
10^2
10^1
10^1
10^2
10^3
10^3
10^2
10^1
0
200
400
600
800
1000
0.04
0.03
0.02
0.01
0
0
200
400
600
800
1000
0.04
0.03
0.02
0.01
0
0
0.005
0.01
0.015

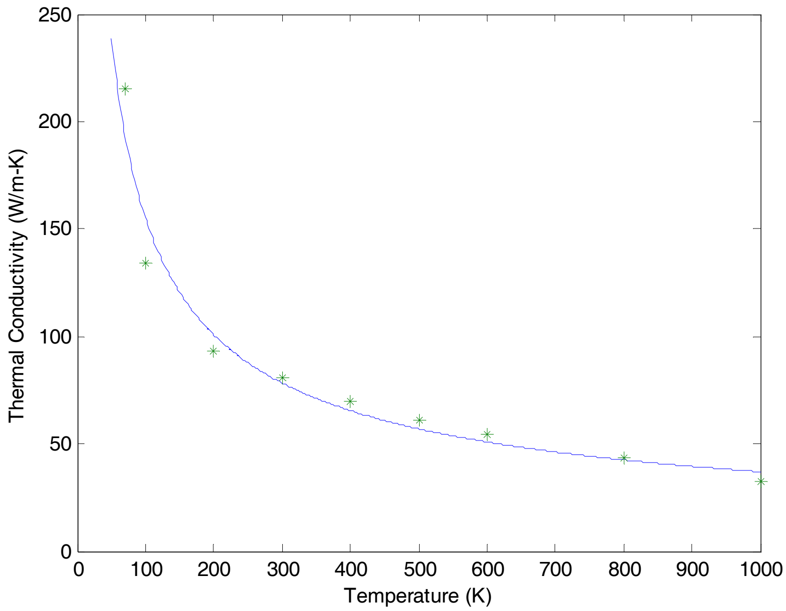
In the top figure the data points appear to fit a straight line. This means that equation of the form: $k = bT^m$ will best fit the data points.

The constants *b* and *m* are then determined by:

```
clear, clc
T=[70 100:100:600 800 1000];
k=[215 134 93 80.8 70 61.2 54.8 43.4 32.2];
p=polyfit(log(T),log(k),1);
Tp=linspace(50,1000,500);
m=p(1)
b=exp(p(2))
kp=b*Tp.^m;
plot(Tp,kp,T,k,'*')
xlabel('Temperature (K)')
ylabel('Thermal Conductivity (W/m-K)')
```

When the script is executed the following values of the constants are displayed in the Command Window, and the following figure is displayed:

```
m =
  -0.622207870413872
b =
    2.723917796389088e+003
```



250
200
150
100
50
0
Thermal Conductivity (W/m-K)
0
100
200
300
400
500
600
700
800
900
1000
Temperature (K)

5.30 In a uniaxial tension test, a dog-bone-shaped specimen is pulled in a machine. During the test, the force applied to the specimen, F, and the length of a gage section, L, are measured. The true stress, σ_t, and the true strain, ε_t, are defined by:

$$\sigma_t = \frac{F}{A_0}\frac{L}{L_0} \quad \text{and} \quad \varepsilon_t = \ln\frac{L}{L_0}$$

where A_0 and L_0 are the initial cross-sectional area and gage length, respectively. The true stress–strain curve in the region beyond the yield stress is often modeled by:

$$\sigma_t = K\varepsilon_t^m$$

The following are values of F and L measured in an experiment. Use the approach from Section 5.3 for determining the values of the coefficients K and m that best fit the data. The initial cross-sectional area and gage length are $A_0 = 1.25 \times 10^{-4}\ \text{m}^2$, and $L_0 = 0.0125$ m.

F (kN)	24.6	29.3	31.5	33.3	34.8	35.7	36.6	37.5	38.8	39.6	40.4
L (mm)	12.58	12.82	12.91	12.95	13.05	13.21	13.35	13.49	14.08	14.21	14.48

Solution

To solve the problem the equation $\sigma_t = K\varepsilon_t^m$ is written in the form:

$$\ln(\sigma_t) = m\ln(\varepsilon_t) + \ln(K)$$

Then, linear least-squares regression is used for finding the coefficients m and K that best fit the data. (See equations in the first row of Table 5-2.)

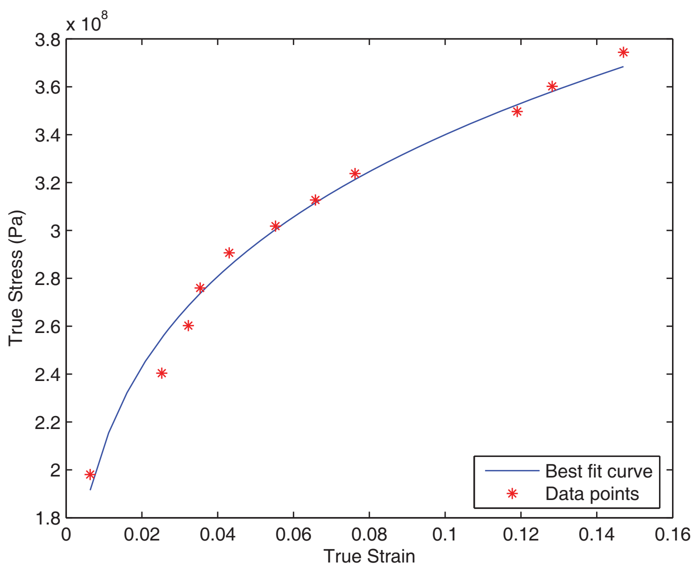
The following program written in a script file determines the constants and plot the data points and the curve that best fit the data.

```
% Solution of HW5_30 script file
clear all
A0= 1.25e-4; L0=0.0125;
F=[24.6 29.3 31.5 33.3 34.8 35.7 36.6 37.5 38.8 39.6 40.4]*1000
L=[12.58 12.82 12.91 12.95 13.05 13.21 13.35 13.49 14.08 14.21 14.48]*0.001
n=length(F)
St=F.*L/(A0*L0)
EPt=log(L/L0)
Y=log(St);
X=log(EPt);
```

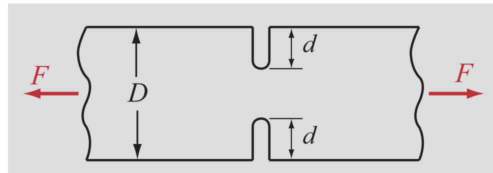
```
p=polyfit(X,Y,1)
m=p(1)
K=exp(p(2))
x=linspace(EPt(1),EPt(n),30)
y=K*x.^m
plot(x,y)
hold on
plot(EPt,St,'*r')
xlabel('True Strain'), ylabel('True Stress (Pa)')
```

When the program is executed, the following values of *m* and *K* are displayed in the Command Window, and the figure that follows is displayed in the Figure Window.

```
m =
    0.2085
K =
  5.4948e+008
```



5.31 The stress concentration factor k is the ratio between the maximum stress σ_{max} and the average stress σ_{ave}, $k = \sigma_{max}/\sigma_{ave}$. For a notched plate of width D with notches of length d loaded with an axial force F (see figure) the maximum stress is at the tip of the notch. The average stress is given by $\sigma_{ave} = F/[t(D-2d)]$, where t is the thickness of the plate. The stress concentration factor measured in five tests with plates with various ratios of d/D is shown in the table.



d/D	0.05	0.15	0.25	0.35	0.45
k	3.84	4.63	4.4	3.63	2.22

Use a cubic polynomial to model the relationship between k and d/D. Plot the data points and the curve-fitted model. Use the model to predict the stress concentration factor for $d/D = 0.21$.
(*a*) Use the user-defined function `CubicPolyFit` developed in Problem 5.22.
(*b*) Use MATLAB's built-in function `polyfit`.

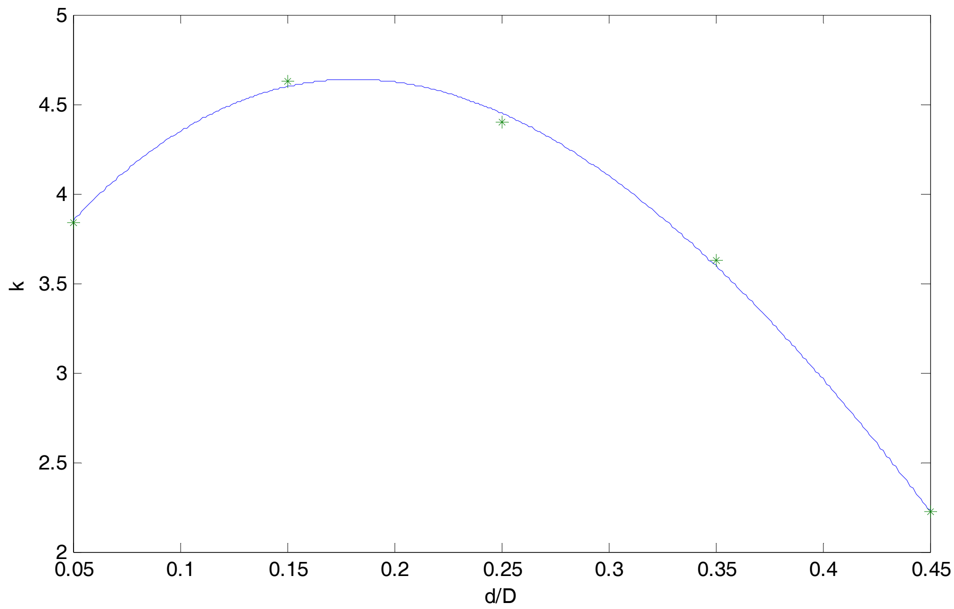
Solution

(*a*)
The problem is solved in the following script:

```
clear; clc
dD=[0.05 0.15 0.25 0.35 0.45];
k=[3.84 4.63 4.4 3.63 2.22];
[a,Er] = CubicPolyFit(dD, k);
kModel=@ (x) a(4)*x.^3+a(3)*x.^2+a(2)*x+a(1);
dDp=0.05:0.001:0.45;
kp=kModel(dDp);
plot(dDp,kp,dD,k,'*')
xlabel('d/D'), ylabel('k')
dD021=kModel(0.21)
```

When the script is executed the following results are displayed in the Command Window, and the following figure is displayed.

```
dD021 =
    4.6039
```



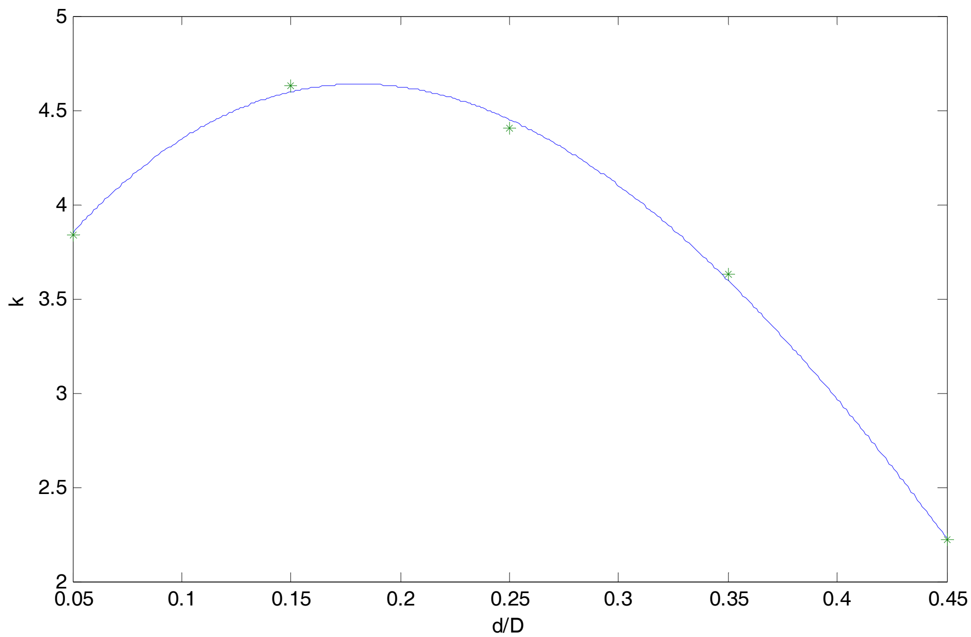
(*b*)
The problem is solved in the following script:

```
clear; clc
dD=[0.05 0.15 0.25 0.35 0.45];
k=[3.84 4.63 4.4 3.63 2.22];
p=polyfit(dD,k,3);
kModel=@ (x) p(1)*x.^3+p(2)*x.^2+p(3)*x+p(4);
dDp=0.05:0.001:0.45;
kp=kModel(dDp);
plot(dDp,kp,dD,k,'*')
xlabel('d/D'), ylabel('k')
```

```
dD021=kModel(0.21)
```

When the script is executed the following results are displayed in the Command Window, and the following figure is displayed.

```
dD021 =
    4.6039
```



5.32 The following data was obtained when the stopping distance *d* of a car on a wet road was measured as a function of the speed *v* when the brakes were applied.

v (km/h)	20	40	60	80	100	120
d (m)	6	18	36	60	91	128

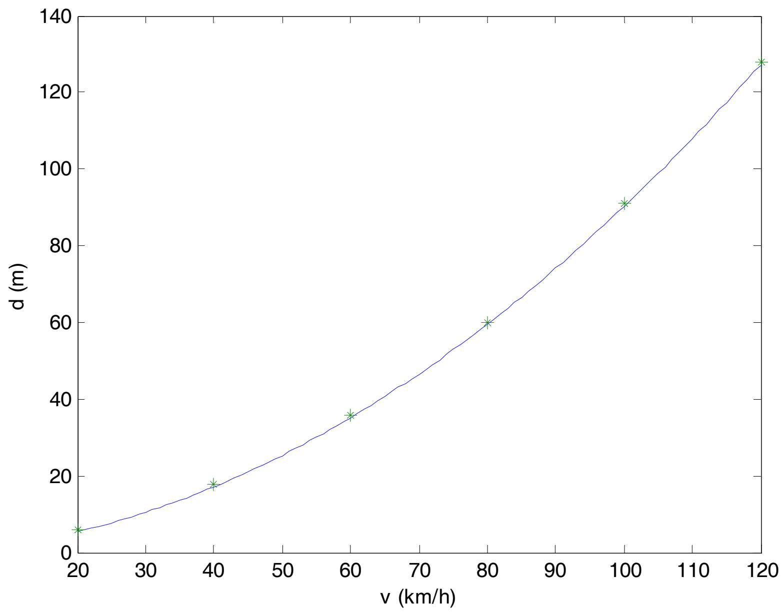
Determine the coefficients of a quadratic polynomial $d = a_2 v^2 + a_1 v + a_0$ that best fit the data. Make a plot that show the data points (asterisk marker) and polynomial (solid line).
(*a*) Use the user-defined function `QuadFit` developed in Problem 5.21.
(*b*) Use MATLAB's built-in function `polyfit`.

Solution

(*a*)
The problem is solved in the following script:

```
clear; clc
v=20:20:120;
d=[6 18 36 60 91 128];
a = QuadFit(v, d);
kModel=@ (x) a(3)*x.^2+a(2)*x+a(3);
vp=20:120;
dp=kModel(vp);
plot(vp,dp,v,d,'*')
xlabel('v (km/h)'), ylabel('d (m)')
```

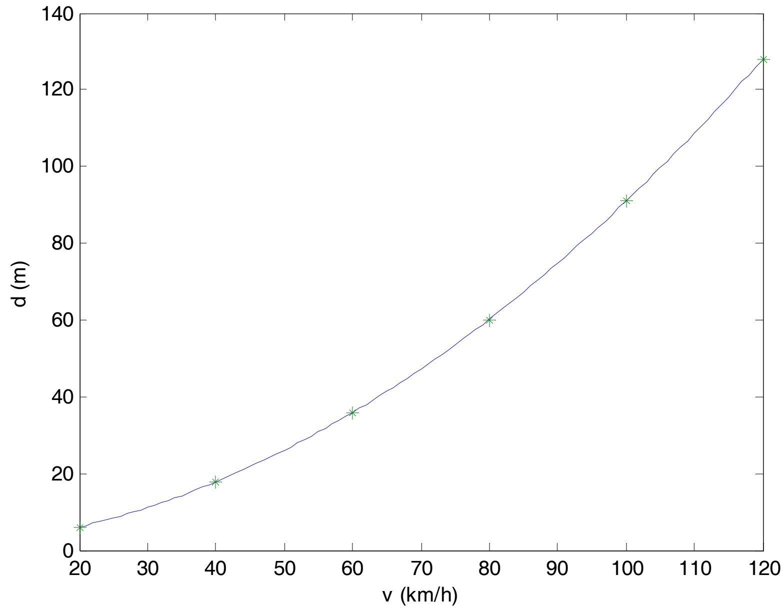
When the script is executed the following figure is displayed.



(*b*)
The problem is solved in the following script:

```
clear; clc
v=20:20:120;
d=[6 18 36 60 91 128];
p=polyfit(v,d,2);
kModel=@ (x) p(1)*x.^2+p(2)*x+p(3);
vp=20:120;
dp=kModel(vp);
plot(vp,dp,v,d,'*')
xlabel('v (km/h)'), ylabel('d (m)')
```

When the script is executed the following figure is displayed.



5.33 Thermistors are resistors that are used for measuring temperature. The relationship between temperature and resistance is given by the Steinhart-Hart equation:

$$\frac{1}{T + 273.15} = C_1 + C_2 \ln(R) + C_3 \ln^3(R)$$

where T is the temperature in degrees Celsius, R is the thermistor resistance in Ω, and C_1, C_2, and C_3, are constants. In an experiment for characterizing a thermistor, the following data was measured:

T (C)	360	320	305	298	295	290	284	282	279	276
R (Ω)	950	3100	4950	6960	9020	10930	13100	14950	17200	18950

Determine the constants C_1, C_2, and C_3 such that the Steinhart-Hart equation will best fit the data.

Solution

The problem is solved by using the user-defined function `NonLinCombFit` written in Homework Problem 5.25.
The problem is solved in the following script file. When the program is executed the value of the constants is displayed in the Command Window. The program create also a figure that display the equation and the data points.

```
clear; clc
T=[360 320 305 298 295 290 284 282 279 276];
R=[950 3100 4950 6960 9020 10930 13100 14950 17200 18950];
Rfit=log(R);
Tfit=1./(T+273.15);
FA=@ (x) x./x;
FB=@ (x) x;
FC=@ (x) x.^3;
c = NonLinCombFit(FA,FB,FC,Rfit,Tfit)
Rp=950:200:18950;
Tp=1./(c(1)+c(2)*log(Rp)+c(3)*log(Rp).^3)-273.15;
plot(Tp,Rp,T,R,'*')
xlabel('T (C)'), ylabel('R (Ohms)')
```

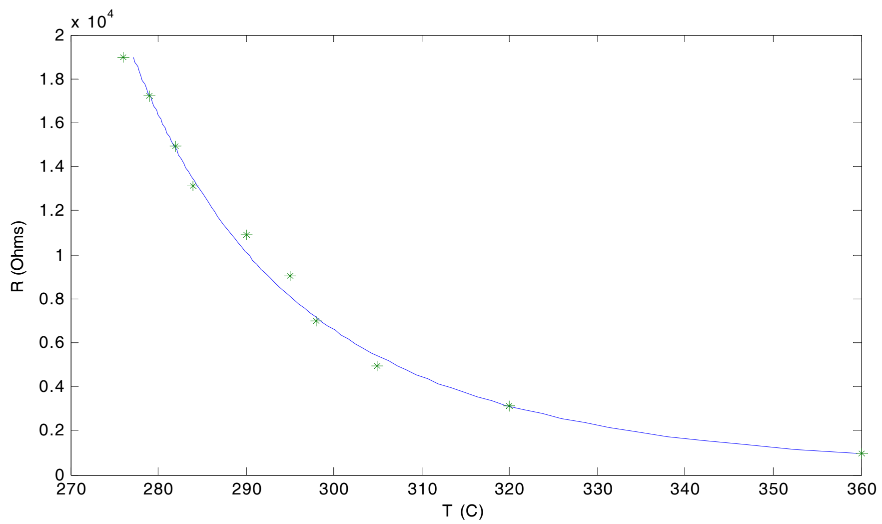
The display in the Command Window is:

```
c =
  1.0e-003 *
   0.792513950961458
   0.125196250130898
  -0.000218430449324
```

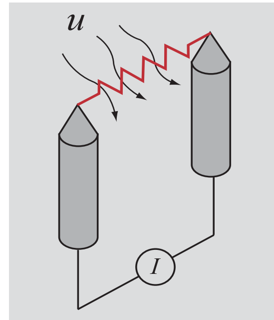
Thus, the constants are:

$C_1 = 0.79251 \times 10^{-3}$ $C_2 = 0.1252 \times 10^{-3}$ $C_3 = -0.21843 \times 10^{-6}$

The figure is:



5.34 A hot-wire anemometer is a device for measuring flow velocity, by measuring the cooling effect of the flow on the resistance of a hot wire. The following data are obtained in calibration tests:



u (ft/s)	4.72	12.49	20.03	28.33	37.47	41.43	48.38	55.06
V (Volt)	7.18	7.3	7.37	7.42	7.47	7.5	7.53	7.55

u (ft/s)	66.77	59.16	54.45	47.21	42.75	32.71	25.43	8.18
V (Volt)	7.58	7.56	7.55	7.53	7.51	7.47	7.44	7.28

Determine the coefficients of the exponential function $u = Ae^{BV}$ that best fit the data.

(*a*) Use the user-defined function `ExpoFit` developed in Problem 5.19.

(*b*) Use MATLAB built-in functions.

In each part plot the data points and the fitting function.

Solution

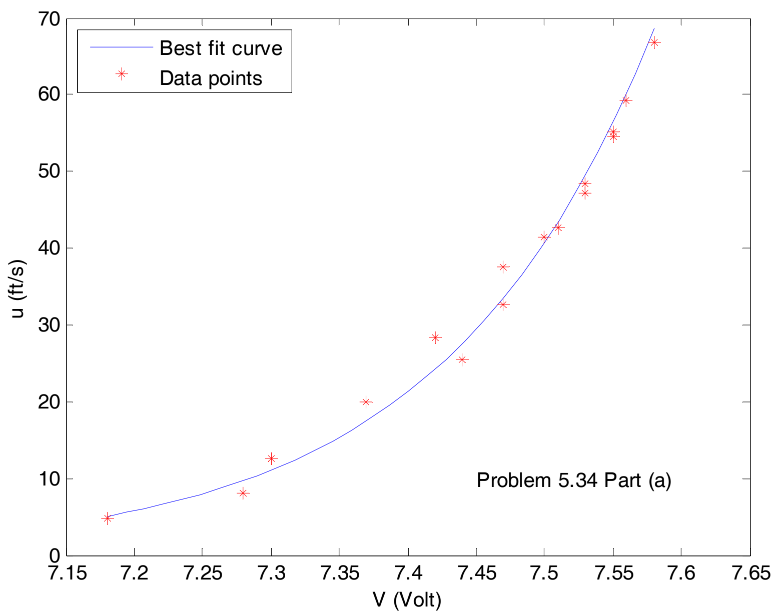
(*a*) The user-defined function `ExpoFit` that was developed in Problem 5-18 fits an exponential function of the form $y = be^{mx}$ to a given set of data points. The following program written in a script file uses `ExpoFit` to determine the constants and plot the data points and the curve that best fit the data.

```
clear, clc
u=[4.72 12.49 20.03 28.33 37.47 41.43 48.38 55.06 ...
    66.77 59.16 54.45 47.21 42.75 32.71 25.43 8.18];
V=[7.18 7.3 7.37 7.42 7.47 7.5 7.53 7.55 7.58 7.56 ...
    7.55 7.53 7.51 7.47 7.44 7.28];
[A,B] = ExpoFit(V, u)
x=linspace(min(V),max(V),30);
y=A*exp(B*x);
plot(x,y)
hold on
plot(V,u,'*r')
xlabel('V (Volt)'), ylabel('u (ft/s)')
legend('Best fit curve','Data points',2)
text(7.45,10,'Problem 5.25 Part (a)')
```

```
hold off
```

When the program is executed, the following values of B and A are displayed in the Command Window, and the figure that follows is displayed in the Figure Window.

```
A =
  2.4825e-020
B =
    6.5133
```



(b) To solve the problem the equation $u = Ae^{BV}$ is written in the form:

$$\ln(u) = B(d/D) + \ln(A)$$

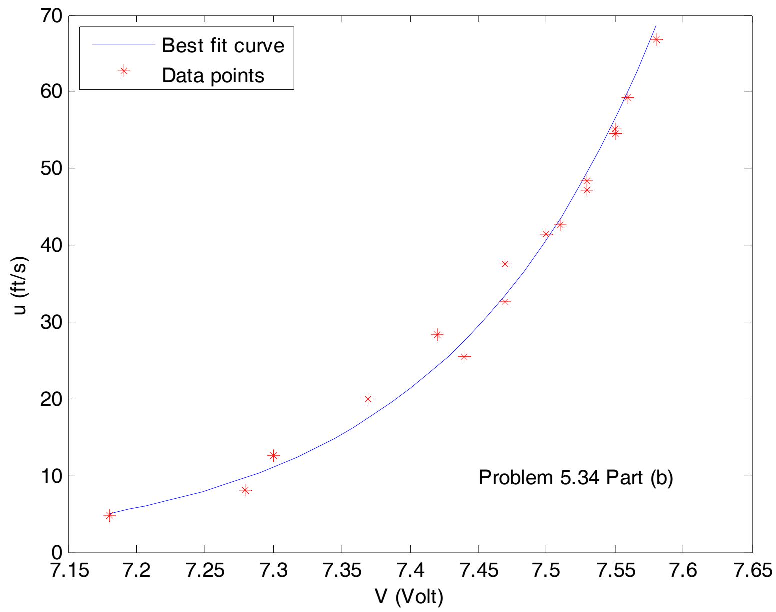
Then, linear least-squares regression is used for finding the coefficients B and A that best fit the data. (See equations in the second row of Table 5-2.)

The following program written in a script file uses MATLAB built-in functions to determine the constants and plot the data points and the curve that best fit the data.

```
% Solution of HW5_34b script file
clear all
u=[4.72 12.49 20.03 28.33 37.47 41.43 48.38 55.06 ...
    66.77 59.16 54.45 47.21 42.75 32.71 25.43 8.18];
V=[7.18 7.3 7.37 7.42 7.47 7.5 7.53 7.55 7.58 7.56 ...
    7.55 7.53 7.51 7.47 7.44 7.28];
n=length(u);
Y=log(u);
p=polyfit(V,Y,1);
B=p(1)
A=exp(p(2))
x=linspace(min(V),max(V),30);
y=A*exp(B*x);
plot(x,y)
hold on
plot(V,u,'*r')
xlabel('V (Volt)'), ylabel('u (ft/s)')
legend('Best fit curve','Data points',2)
text(7.45,10,'Problem 5.34 Part (b)')
hold off
```

When the program is executed, the following values of *B* and *A* are displayed in the Command Window, and the figure that follows is displayed in the Figure Window.

```
B =
    6.5133
A =
  2.4825e-020
```


Best fit curve
Data points
Problem 5.34 Part (b)
u (ft/s)
V (Volt)
0
10
20
30
40
50
60
70
7.15
7.2
7.25
7.3
7.35
7.4
7.45
7.5
7.55
7.6
7.65

5.35 The data given is to be curve-fitted with the equation $y = axe^{mx}$. Transform the equation to a linear form and determine the constants a and m by using linear least-square regression. (Hint: substitute $v = \ln(y/x)$ and $u = x$.) Make a plot that shows the points (circle markers) and the equation (solid line).

x	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
y	0	2.4	3.00	2.55	2.24	1.72	1.18	0.82	0.56	0.42	0.25

Solution

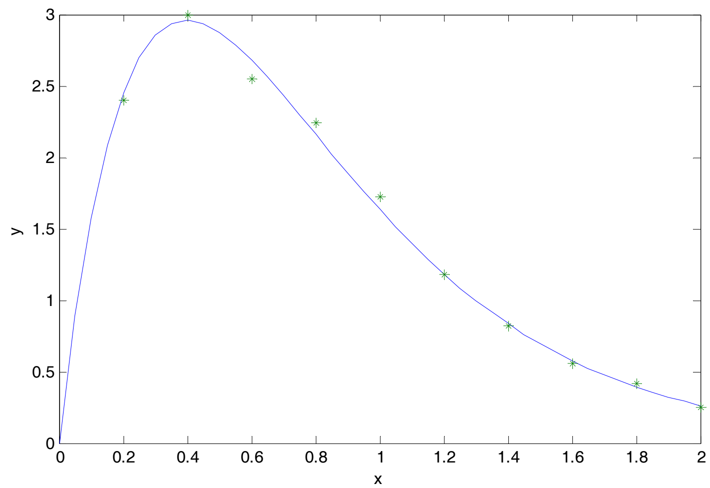
The linear form of the equation is: $\ln\frac{y}{x} = mx + \ln a$ or $v = mu + \ln a$

The first point (0, 0) is excluded from the regression analysis since by definition $y(0) = 0$.
The problem is solved in the following script file:

```
clear, clc
x=0.2:0.2:2;
y=[2.4 3 2.55 2.24 1.72 1.18 0.82 0.56 0.42 0.25];
u=x;
v=log(y./x);
p=polyfit(u,v,1);
m=p(1)
a=exp(p(2))
xp=0:0.05:2;
yp=a*xp.*exp(m*xp);
plot(xp,yp,x,y,'*')
xlabel('x'), ylabel('y')
```

When the script is executed the values of m and a are displayed in the Command Window, and the following figure is displayed in the Figure Window.

```
m =
   -2.5184
a =
   20.2564
```



y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.5
1.5
2.5
3

5.36 The yield stress of many metals, σ_y, varies with the size of the grains. Often, the relationship between the grain size, *d*, and the yield stress is modeled with the Hall–Petch equation:

$$\sigma_y = \sigma_0 + kd^{\left(-\frac{1}{2}\right)}$$

The following are results from measurements of average grain size and yield stress:

d (mm)	0.006	0.011	0.017	0.025	0.039	0.060	0.081	0.105
σ_y (MPa)	334	276	249	235	216	197	194	182

(*a*) Determine the constants σ_0 and *k* such that the Hall–Petch equation will best fit the data. Plot the data points (circle markers) and the Hall–Petch equation as a solid line. Use the Hall–Petch equation to estimate the yield stress of a specimen with a grain size of 0.05 mm.

(*b*) Use the user-defined function `QuadFit` from Problem 5.21 to find the quadratic function that best fits the data. Plot the data points (circle markers) and the quadratic equation as a solid line. Use the quadratic equation to estimate the yield stress of a specimen with a grain size of 0.05 mm.

Solution

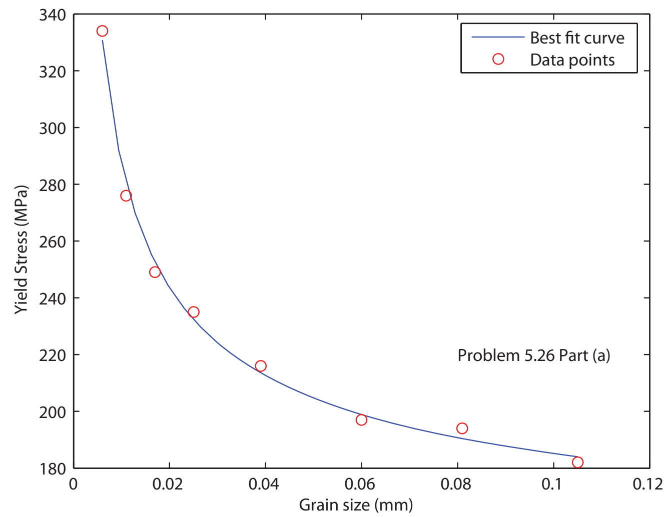
(*a*) The coefficients σ_0 and *k* that best fit the data in the equation $\sigma_y = \sigma_0 + kd^{\left(-\frac{1}{2}\right)}$ are determined by using linear least-squares regression with $d^{\left(-\frac{1}{2}\right)}$ as the independent variable and σ_y as the dependent variable. The following program written in a script file uses MATLAB built-in functions to determine the constants, plot the data points and the curve that best fit the data, and estimating the yield stress for grain size of 0.05 mm.

```
clear all
d=[0.006 0.011 0.017 0.025 0.039 0.06 0.081 0.105];
Sy=[334 276 249 235 216 197 194 182];
n=length(d);
X=d.^(-0.5);
p=polyfit(X,Sy,1);
k=p(1)
S0=p(2)
x=linspace(min(d),max(d),30);
y=S0+k*x.^(-0.5);
```

```
plot(x,y)
hold on
plot(d,Sy,'or')
xlabel('Grain size (mm)'), ylabel('Yield Stress (MPa)')
legend('Best fit curve','Data points')
hold off
% Model prediction for d = 0.05
SigmayFor05=S0+k*0.05^(-0.5)
```

When the program is executed, the values σ_0 and k and the prediction for grain size of 0.05 mm are displayed in the Command Window. The figure that follows is displayed in the Figure Window.

```
k =
   14.9410
S0 =
  137.9135
SigmayFor05 =
  204.7318
```



(*b*) Quadratic function that fits the data has the form:

$$\sigma_y = a_2 d^2 + a_1 d + a_0$$

The following program written in a script file uses the user-defined function `QuadFit` from Problem 5.20 to find the quadratic function that best fits the data. It also plots the data points and the curve that best fit the data, and estimates the yield stress for grain size of 0.05 mm.

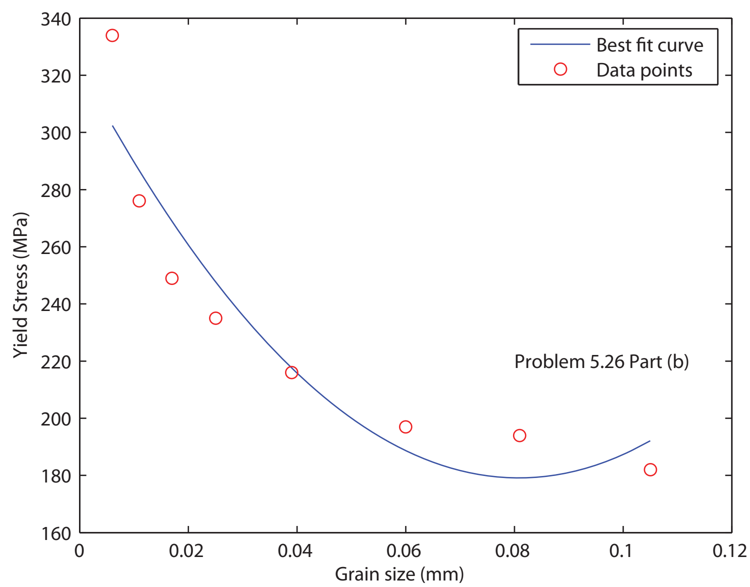
```
clear all
d=[0.006 0.011 0.017 0.025 0.039 0.06 0.081 0.105];
Sy=[334 276 249 235 216 197 194 182];
n=length(d);
a = QuadFit(d, Sy)
x=linspace(min(d),max(d),30);
y=a(3)*x.^2+a(2)*x+a(1);
plot(x,y)
hold on
plot(d,Sy,'or')
xlabel('Grain size (mm)'), ylabel('Yield Stress (MPa)')
legend('Best fit curve','Data points')
text(0.08,220,'Problem 5.26 Part (b)')
```

```
hold off
% Model prediction for d = 0.05
SigmayFor05=a(3)*0.05^2+a(2)*0.05+a(1)
```

When the program is executed, the values a_0, a_1, and a_2, and the prediction for grain size of 0.05 mm are displayed in the Command Window. The figure that follows is displayed in the Figure Window.

```
>> format short e
a =
  3.2301e+002
 -3.5624e+003
  2.2052e+004
SigmayFor05 =
  2.0002e+002
```

The quadratic equation is: $\sigma_y = 22052d^2 - 356.24d + 323.01$



5.37 Values of enthalpy per unit mass, *h*, of an equilibrium Argon plasma (Ar, Ar^+, A^{++}, A^{+++} ions and electrons) versus temperature are:

$T \times 10^3$ (K)	5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30
h (MJ/kg)	3.3	7.5	41.8	51.8	61	101.1	132.9	145.5	171.4	225.8	260.9

Write a program in a script file that uses interpolation to calculate *h* at temperatures ranging from 5000 K to 30000 K in increments of 500 K. The program should generate a plot that shows the interpolated points, and the data points from the table (use an asterisk marker).

(*a*) For interpolation use the user-defined function `CubicSplines` from Problem 5.23.

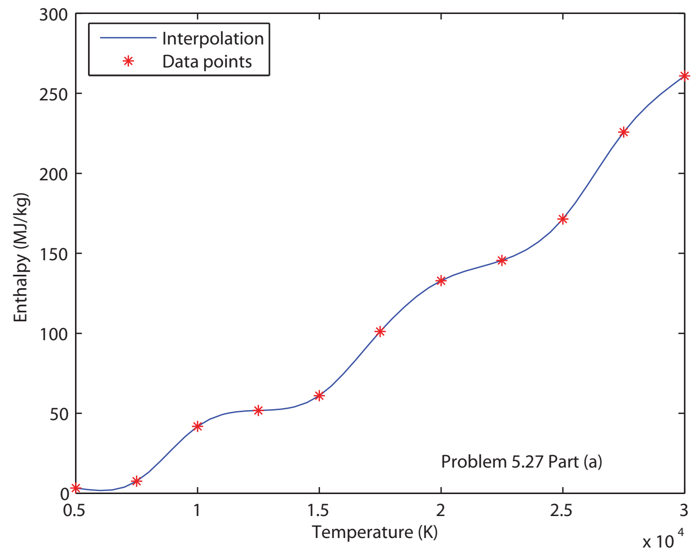
(*b*) For interpolation use MATLAB's built-in function `interp1` with the `spline` option.

Solution

(*a*) The following program in a script file solves the problem using the user-defined function `CubicSplines` from Problem 5.22.

```
clear all
T=[5:2.5:30]*1000;
h=[3.3 7.5 41.8 51.8 61 101.1 132.9 145.5 171.4 225.8 260.9];
Tint=5000:500:30000;
n=length(Tint);
for i=1:n
    hint(i)=CubicSplines(T,h,Tint(i));
end
plot(Tint,hint)
hold on
plot(T,h,'*r')
xlabel('Temperature (K)')
ylabel('Enthalpy (MJ/kg)')
legend('Interpolation','Data points',2)
text(20000,20,'Problem 5.27 Part (a)')
hold off
```

When the script file is executed, the following plot is generated:

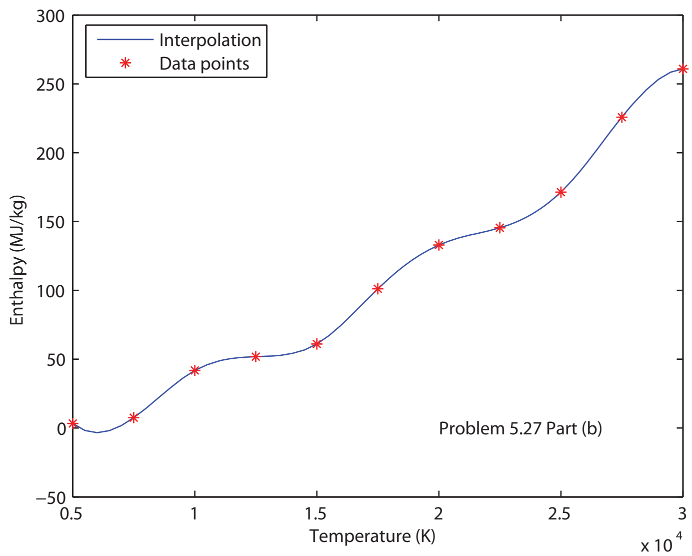


(*b*) The following program in a script file solves the problem using MATLAB's built-in function `interp1` with the `spline` option.

```
clear all
T=[5:2.5:30]*1000;
h=[3.3 7.5 41.8 51.8 61 101.1 132.9 145.5 171.4 225.8 260.9];
Tint=5000:500:30000;
n=length(Tint);
hint=interp1(T,h,Tint,'spline');
plot(Tint,hint)
hold on
plot(T,h,'*r')
xlabel('Temperature (K)')
ylabel('Enthalpy (MJ/kg)')
legend('Interpolation','Data points',2)
```

```
hold off
```

When the script file is executed, the following plot is generated:



5.38 The following are measurements of the rate coefficient, k, for the reaction $CH_4 + O \rightarrow CH_3 + OH$ at different temperatures, T:

T (K)	595	623	761	849	989	1076	1146	1202	1382	1445	1562
$k \times 10^{20}$ (m^3/s)	2.12	3.12	14.4	30.6	80.3	131	186	240	489	604	868

(*a*) Use the method of least-squares to best fit a function of the form $\ln(k) = C + b\ln(T) - \frac{D}{T}$ to the data. Determine the constants C, b, and D by curve fitting a linear combination of the functions $f_1(T) = 1$, $f_2(T) = \ln(T)$, and $f_3(T) = \frac{-1}{T}$ to the given data (Section 5.8).

(*b*) Usually, the rate coefficient is expressed in the form of an Arrhenius equation $k = AT^b e^{-E_a/(RT)}$, where A and b are constants, $R = 8.314$ J/mole/K is the universal gas constant, and E_a is the activation energy for the reaction. Having determined the constants C, b, and D in part (*a*), deduce the values of A (m^3/s) and E_a (J/mole) in the Arrhenius expression.

Solution

(*a*) Using Eq. (5.97), the following system of three linear equations for the unknowns C, b, and D can be written:

$$C \cdot 11 + b\sum_{i=1}^{11}\ln(T_i) + D\sum_{i=1}^{11}\frac{-1}{T_i} = \sum_{i=1}^{11}\ln(k_i)$$

$$C \cdot \sum_{i=1}^{11}\ln(T_i) + b\sum_{i=1}^{11}[\ln(T_i)]^2 + D\sum_{i=1}^{11}\frac{-1}{T_i}\ln(T_i) = \sum_{i=1}^{11}\ln(k_i)\ln(T_i)$$

$$C \cdot \sum_{i=1}^{11}\frac{-1}{T_i} + b\sum_{i=1}^{11}\frac{-1}{T_i}\ln(T_i) + D\sum_{i=1}^{11}\left(\frac{-1}{T_i}\right)^2 = \sum_{i=1}^{11}\ln(k_i)\frac{-1}{T_i}$$

In the program listed below, the system of equations is solved. In the program the unknowns (solution) is a three element vector named `Cnst` such that $C = Cnst(1)$, $b = Cnst(2)$, and $D = Cnst(3)$. The program generates also a plot of $\ln(k)$ versus T. The plot shows the data points and line of the function $C + b\ln(T) - \frac{D}{T}$.

(b) The equation $\ln(k) = C + b\ln(T) - \frac{D}{T}$ can be rewritten in the form $k = AT^b e^{-E_a/(RT)}$:

$$k = e^C T^b e^{-D/T}$$

The constants A, b, and E_a are related to the constants C, b, and D that were determined in Part (a) by:

$A = e^C$, $b = b$, and $D = \frac{E_a}{R}$

The computer program calculates the constants and then makes a plot of k versus T. The plot shows the data points and the line predicted by the equation.

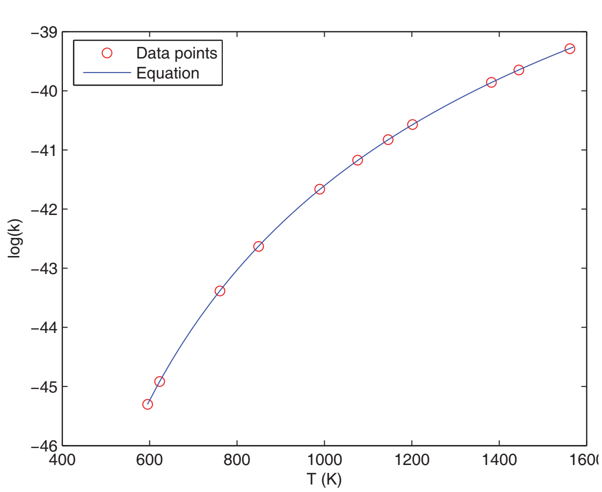
```
clear all
T = [595 623 761 849 989 1076 1146 1202 1382 1445 1562];
k = [2.12 3.12 14.4 30.6 80.3 131 186 240 489 604 868]*1e-20;
n=length(k);
a(1,1) = n;
a(1,2) = sum(log(T));
a(1,3) = sum(-1./T);
a(2,1) = a(1,2);
a(2,2) = sum(log(T).^2);
a(2,3) = sum(-log(T)./T);
a(3,1) = a(1,3);
a(3,2) = a(2,3);
a(3,3) = sum(1./(T.^2));
b(1,1) = sum(log(k));
b(2,1) = sum(log(k).*log(T));
b(3,1) = sum(-log(k)./T);
Cnst = a\b
Tfit = 595:25:1575;
LogkFit = Cnst(1)+ Cnst(2)*log(Tfit)-Cnst(3)./Tfit;
plot(T,log(k),'or',Tfit,LogkFit)
xlabel('T (K)'), ylabel('log(k)')
legend('Data points','Equation',2)
```

```
%   Part b
R=8.314;
A=exp(Cnst(1))
bb = Cnst(2)
Ea=Cnst(3)*R
kfit=A*Tfit.^bb.*exp(-Ea./(R*Tfit));
figure
plot(T,k,'or',Tfit,kfit)
xlabel('T (K)'), ylabel('k (m^3/s)')
legend('Data points','Equation',2)
```

When the program is executed, the values of the constants is displayed in the Command Window, and the following two figures are displayed.

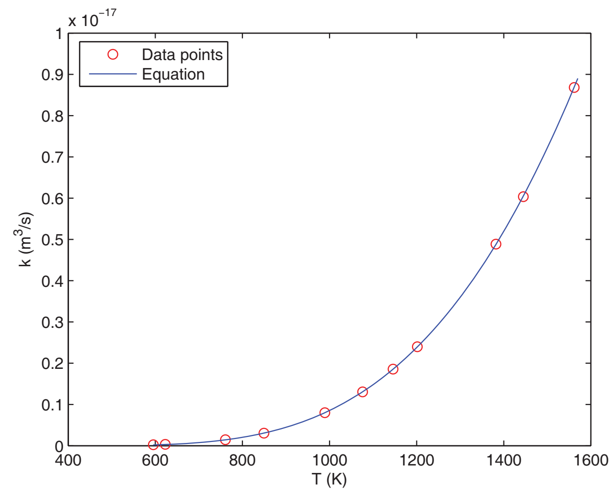
Command Window:

```
>> format short e
Cnst =
 -5.2443e+001
  2.1216e+000
  3.8153e+003
A =
  1.6765e-023
bb =
  2.1216e+000
Ea =
  3.1721e+004
```



Data points
Equation
log(k)
T (K)
−39
−40
−41
−42
−43
−44
−45
−46
400
600
800
1000
1200
1400
1600

Part (*a*)



x 10^−17
Data points
Equation
k (m^3/s)
T (K)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
400
600
800
1000
1200
1400
1600

Part (*b*)

5.39 The following measurements were recorded in a study on the growth of trees.

Age (year)	5	10	15	20	25	30	35
Height (m)	5.2	7.8	9	10	10.6	10.9	11.2

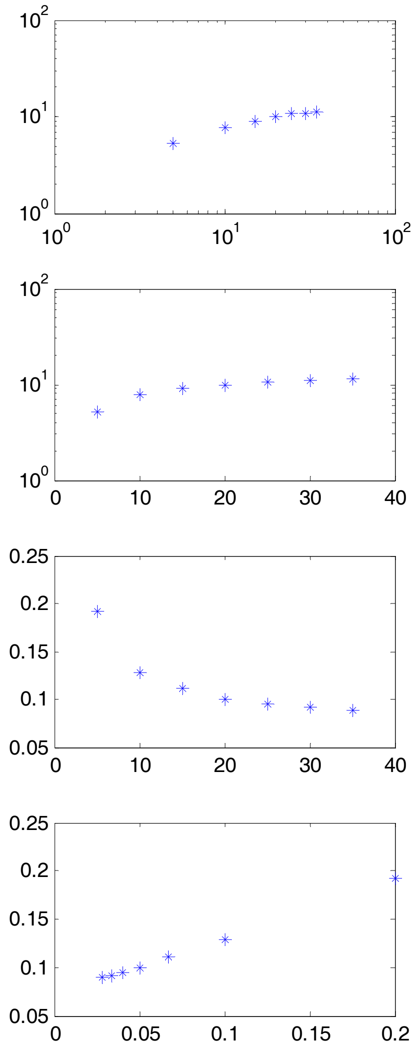
The data is used for deriving an equation $H = H(Age)$ that can predict the height of the trees as a function of their age. Determine which of the nonlinear equations that are listed in Table 5-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

Solution

First, the following four plots are made. H (height) vs. A (age) on a log-log plot. H vs. A on a log H (vertical) linear A (horizontal) axes. $1/H$ vs. A with linear axes. $1/H$ vs. $1/A$ with linear axes.

```
clear, clc
A=5:5:35;
H=[5.2 7.8 9 10 10.6 10.9 11.2];
subplot(4,1,1)
loglog(A,H,'*')
subplot(4,1,2)
semilogy(A,H,'*')
subplot(4,1,3)
plot(A,1./H,'*')
subplot(4,1,4)
plot(1./A,1./H,'*')
```

When the script is executed, the following plots are displayed:


10^2
10^1
10^0
10^0
10^1
10^2
10^2
10^1
10^0
0
10
20
30
40
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
0.25
0.2
0.15
0.1
0.05
0
0.05
0.1
0.15
0.2

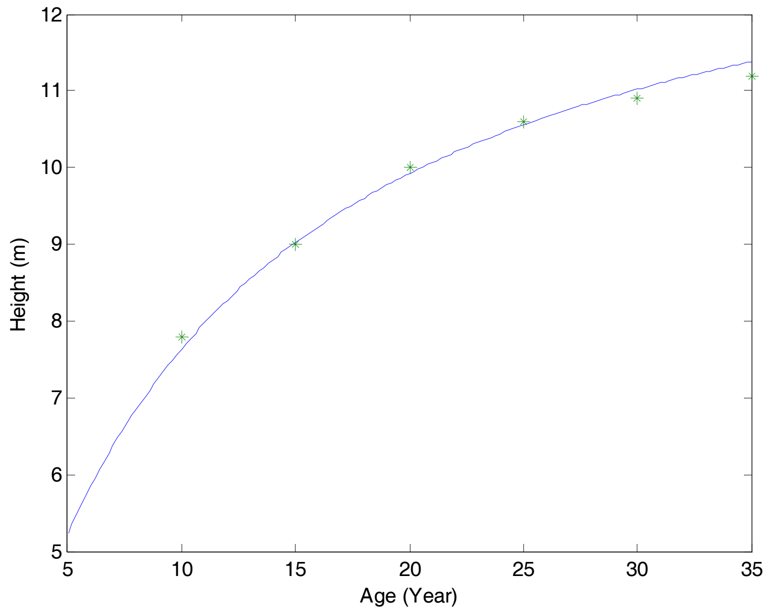
In the last figure (bottom) the data points appear to fit a straight line. This means that equation of the form: $H = \frac{mA}{b+A}$ will best fit the data points.

The constants *b* and *m* are then determined by:

```
clear, clc
p=polyfit(1./A,1./H,1);
Ap=5:0.2:35;
m=1/p(2)
b=p(1)*m
Hp=m*Ap./(b+Ap);
figure
plot(Ap,Hp,A,H,'*')
xlabel('Age (Year)')
ylabel('Height (m)')
```

When the script is executed the following values of the constants are displayed in the Command Window, and the following figure is displayed:

```
m =
   14.1546
b =
    8.5285
```


12
11
10
9
8
7
6
5
Height (m)
5
10
15
20
25
30
35
Age (Year)

5.40 The following data present ocean water temperature at different depths:

Depth (m)	0	100	300	400	500	750	1000	1250	1500	1750
Water temp (^{o}C)	22	21.8	21	18.7	16	10	5.6	4.9	4.6	4.4

Use interpolation to estimate the temperature at depths of 50 m, 600 m, and 1400 m.
(*a*) Use the user-defined function `CubicSplines` developed in Problem 5.23.
(*b*) Use the user-defined function `CubicLagSplines` developed in Problem 5.24.
(*c*) Use the MATLAB's built-defined function `interp1` with the option `spline`.

Solution

The problem is solved in the following script file:

```
clear, clc
D=[0 100 300 400 500 750 1000 1250 1500 1750];
WT=[22 21.8 21 18.7 16 10 5.6 4.9 4.6 4.4];

% Part a
D50Parta=CubicSplines(D,WT,50)
D600Parta=CubicSplines(D,WT,600)
D1400Parta=CubicSplines(D,WT,1400)
% Part b
D50Partb=CubicLagSplines(D,WT,50)
D600Partb=CubicLagSplines(D,WT,600)
D51400Partb=CubicLagSplines(D,WT,1400)
% Part c
D50Partc=interp1(D,WT,50,'spline')
D600Partc=interp1(D,WT,600,'spline')
D51400Partc=interp1(D,WT,1400,'spline')
```

When the script is executed, the following results are displayed in the Command Window:

```
D50Parta =
   21.8701
D600Parta =
```

```
   13.4681
D1400Parta =
    4.7464
D50Partb =
   21.9167
D600Partb =
   13.4503
D51400Partb =
    4.6912
D50Partc =
   21.8033
D600Partc =
   13.4668
D51400Partc =
    4.7598
```