

## Chapter 6

# Curve Fitting and Interpolation

### **Core Topics**

Curve fitting with a linear equation (6.2).

Curve fitting with nonlinear equation by writing the equation in linear form (6.3).

Curve fitting with quadratic and higher order polynomials (6.4).

Interpolation using a single polynomial (6.5).

Lagrange polynomials (6.5.1).

Newton's polynomials (6.5.2).

Piecewise (spline) interpolation (6.6).

Use of MATLAB built-in functions for curve fitting and interpolation (6.7).

### **Complementary Topics**

Curve fitting with linear combination of nonlinear functions (6.8).

## 6.1 BACKGROUND

Many scientific and engineering observations are made by conducting experiments in which physical quantities are measured and recorded. The experimental records are typically referred to as data points. For example, the strength of many metals depends on the size of the grains. Testing specimens with different grain sizes yields a discrete set of numbers ( $d$  – average grain diameter,  $\sigma_y$  – yield strength) as shown in Table 6-1.

**Table 6-1: Strength-grain size data.**

$d$ (mm)	0.005	0.009	0.016	0.025	0.040	0.062	0.085	0.110
$\sigma_y$ (MPa)	205	150	135	97	89	80	70	67

Sometimes measurements are made and recorded continuously with analog devices, but in most cases, especially in recent years with the wide use of computers, the measured quantities are digitized and stored as a set of discrete points.

Once the data is known, scientists and engineers can use it in different ways. Often the data is used for developing, or evaluating, mathematical formulas (equations) that represent the data. This is done by curve fitting in which a specific form of an equation is assumed, or provided by a guiding theory, and then the parameters of the equation are determined such that the equation best fits the data points. Sometimes the data points are used for estimating the expected values between the

known points, a procedure called ***interpolation***, or for predicting how the data might extend beyond the range over which it was measured, a procedure called ***extrapolation***.

### Curve fitting

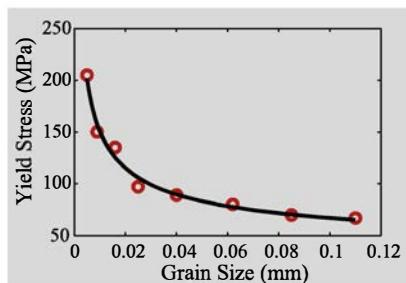


Figure 6-1: Curve fitting.

Curve fitting is a procedure in which a mathematical formula (equation) is used to best fit a given set of data points. The objective is to find a function that fits the data points overall. This means that the function does not have to give the exact value at any single point, but fits the data well overall. For example, Fig. 6-1 shows the data points from Table 6-1 and a curve that shows the best fit of a power function ( $\sigma = Cd^m$ ) to the data points. It can be observed that the curve fits the general trend of the data but does not match any of the data points exactly. Curve fitting is typically used when the values of the data points have some error, or scatter. Generally, all experimental measurements have built-in errors or uncertainties, and requiring a curve fit to go through every data point is not beneficial. The procedure is also used for determining the validity of proposed equations used to represent the data and for determining the values of parameters (coefficients) in the equations. Curve fitting can be carried out with many types of functions and with polynomials of various orders.

### Interpolation

Interpolation is a procedure for estimating a value *between* known values of data points. It is done by first determining a polynomial that gives the exact value at the data points, and then using the polynomial for calculating values between the points. When a small number of points is involved, a single polynomial might be sufficient for interpolation over the whole domain of the data points. Often, however, when a large number of points are involved, different polynomials are used in the intervals between the points in a process that is called spline interpolation. For example, Fig. 6-2 shows a plot of the stress-strain relationship for rubber. The red markers show experimental points that were measured very accurately, and the solid curve was obtained by using spline interpolation. It can be observed that the curve passes through the points precisely and gives a good estimate of values between the points.

The next three sections cover curve fitting. Section 6.2 describes how to curve-fit a set of data points with a linear function using least-squares regression analysis. In Section 6.3 data points are curve fit with nonlinear functions by rewriting the functions in a linear form. In Section 6.4 curve fitting is carried out with second and higher-order polynomials. Interpolation is covered in the next two sections. Section 6.5 shows how to find the equation of a single polynomial that passes through a given set of data points (Lagrange and Newton's polynomials), and Section 6.6 covers piecewise (spline) interpolation in which

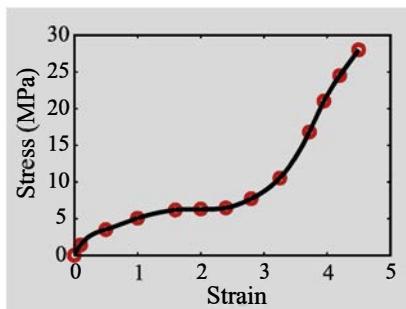


Figure 6-2: Interpolation.

different polynomials are used for interpolation in the intervals between the data points. Section 6.7 describes the tools that MATLAB has for curve fitting and interpolation. In Section 6.8 curve fitting is done in a more general way by using a linear combination of nonlinear functions.

## 6.2 CURVE FITTING WITH A LINEAR EQUATION

Curve fitting using a linear equation (first degree polynomial) is the process by which an equation of the form:

$$y = a_1x + a_0 \quad (6.1)$$

is used to best fit given data points. This is done by determining the constants  $a_1$  and  $a_0$  that give the smallest error when the data points are substituted in Eq. (6.1). If the data comprise only two points, the constants can be determined such that Eq. (6.1) gives the exact values at the points. Graphically, as shown in Fig. 6-3, it means that the straight line that corresponds to Eq. (6.1) passes through the two points.

When the data consists of more than two points, obviously, a straight line cannot pass through all of the points. In this case, the constants  $a_1$  and  $a_0$  are determined such that the line has the best fit overall, as illustrated in Fig. 6-4.

The process of obtaining the constants that give the best fit first requires a definition of best fit (Section 6.2.1) and a mathematical procedure for deriving the value of the constants (Section 6.2.2).

### 6.2.1 Measuring How Good Is a Fit

A criterion that measures how good a fit is between given data points and an approximating linear function is a formula that calculates a number that quantifies the overall agreement between the points and the function. Such a criterion is needed for two reasons. First, it can be used to compare two different functions that are used for fitting the same data points. Second, and even more important, the criterion itself is used for determining the coefficients of the function that give the best fit. This is shown in Section 6.2.2.

The fit between given data points and an approximating linear function is determined by first calculating the error, also called the residual, which is the difference between a data point and the value of the approximating function, at each point. Subsequently, the residuals are used for calculating a total error for all the points. Figure 6-5 shows a general case of a linear function (straight line) that is used for curve fitting  $n$  points. The residual  $r_i$  at a point,  $(x_i, y_i)$ , is the difference between the value  $y_i$  of the data point and the value of the function  $f(x_i)$  used to approximate the data points:

$$r_i = y_i - f(x_i) \quad (6.2)$$

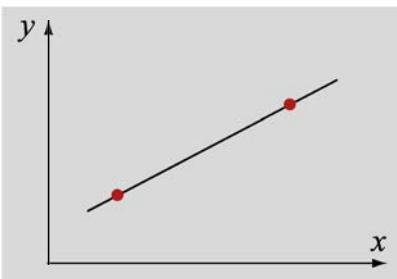


Figure 6-3: Two data points.

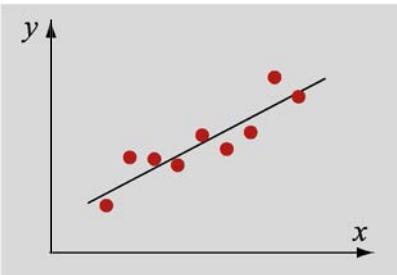
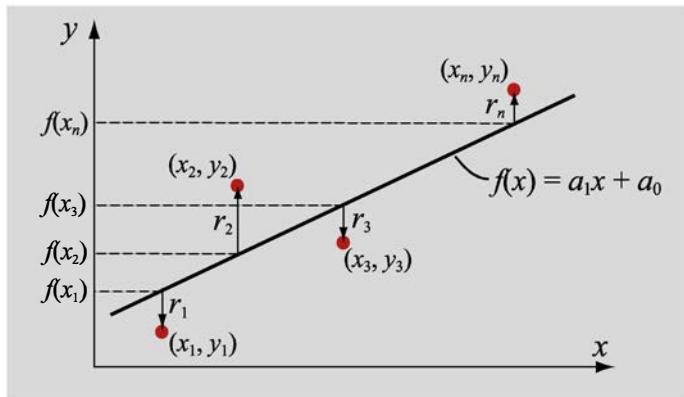


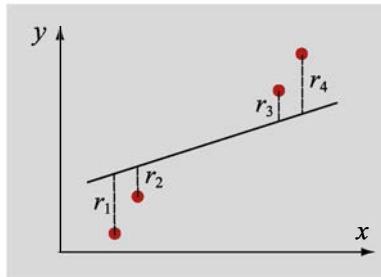
Figure 6-4: Many data points.



**Figure 6-5: Curve-fitting points with a linear equation.**

A criterion that measures how well the approximating function fits the given data can be obtained by calculating a total error  $E$  in terms of the residuals. The overall error can be calculated in different ways. One simple way is to add the residuals of all the points:

$$E = \sum_{i=1}^n r_i = \sum_{i=1}^n [y_i - (a_1 x_i + a_0)] \quad (6.3)$$

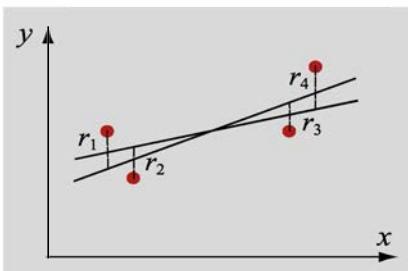


**Figure 6-6: Fit with no error according to Eq. (6.3).**

The error that is calculated in this way does not provide a good measure of the overall fit. This is because a bad fit with positive residuals and negative residuals (both can be large) can sum up to give a zero (or very close to zero) error, implying a good fit. A situation like this is shown in Fig. 6-6, where  $E$  according to Eq. (6.3) is zero since  $r_1 = -r_4$  and  $r_2 = -r_3$ .

Another possibility is to make the overall error  $E$  equal to the sum of the absolute values of the residuals:

$$E = \sum_{i=1}^n |r_i| = \sum_{i=1}^n |y_i - (a_1 x_i + a_0)| \quad (6.4)$$



**Figure 6-7: Two fits with the same error according to Eq. (6.4).**

With this definition, the total error is always a positive number since the residuals cannot cancel each other. A smaller  $E$  in Eq. (6.4) indicates a better fit. This measure can be used to evaluate or compare proposed fits, but it cannot be used for determining the constants of the function that give the best fit. This is because the measure is not unique, which means that for the same set of points there can be several functions that give the same total error. This is shown in Fig. 6-7 where total error  $E$  according to Eq. (6.4) is the same for the two approximating lines.

A definition for the overall error  $E$  that gives a good measure of the total error and can also be used for determining a unique linear function that has the best fit (i.e., smallest total error) is obtained by making  $E$  equal to the sum of the squares of the residuals:

$$E = \sum_{i=1}^n r_i^2 = \sum_{i=1}^n [y_i - (a_1 x_i + a_0)]^2 \quad (6.5)$$

With this definition, the overall error is always a positive number (positive and negative residuals do not cancel each other). In addition, larger residuals have a relatively larger effect (weight) on the total error. As already mentioned, Eq. (6.5) can be used to calculate the coefficients  $a_1$  and  $a_0$  in the linear function  $y = a_1x + a_0$  that give the smallest total error. This is done by using a procedure called linear least-squares regression, which is presented in the next section.

### 6.2.2 Linear Least-Squares Regression

Linear least-squares regression is a procedure in which the coefficients  $a_1$  and  $a_0$  of a linear function  $y = a_1x + a_0$  are determined such that the function has the best fit to a given set of data points. The best fit is defined as the smallest possible total error that is calculated by adding the squares of the residuals according to Eq. (6.5).

For a given set of  $n$  data points  $(x_i, y_i)$ , the overall error calculated by Eq. (6.5) is:

$$E = \sum_{i=1}^n [y_i - (a_1x_i + a_0)]^2 \quad (6.6)$$

Since all the values  $x_i$  and  $y_i$  are known,  $E$  in Eq. (6.6) is a nonlinear function of the two variables  $a_1$  and  $a_0$ . The function  $E$  has a minimum at the values of  $a_1$  and  $a_0$  where the partial derivatives of  $E$  with respect to each variable is equal to zero. Taking the partial derivatives and setting them equal to zero gives:

$$\frac{\partial E}{\partial a_0} = -2 \sum_{i=1}^n (y_i - a_1x_i - a_0) = 0 \quad (6.7)$$

$$\frac{\partial E}{\partial a_1} = -2 \sum_{i=1}^n (y_i - a_1x_i - a_0)x_i = 0 \quad (6.8)$$

Equations (6.7) and (6.8) are a system of two linear equations for the unknowns  $a_1$  and  $a_0$ , and can be rewritten in the form:

$$na_0 + \left( \sum_{i=1}^n x_i \right) a_1 = \sum_{i=1}^n y_i \quad (6.9)$$

$$\left( \sum_{i=1}^n x_i \right) a_0 + \left( \sum_{i=1}^n x_i^2 \right) a_1 = \sum_{i=1}^n x_i y_i \quad (6.10)$$

The solution of the system is:

$$a_1 = \frac{n \sum_{i=1}^n x_i y_i - \left( \sum_{i=1}^n x_i \right) \left( \sum_{i=1}^n y_i \right)}{n \sum_{i=1}^n x_i^2 - \left( \sum_{i=1}^n x_i \right)^2} \quad (6.11)$$

$$a_0 = \frac{\left(\sum_{i=1}^n x_i^2\right)\left(\sum_{i=1}^n y_i\right) - \left(\sum_{i=1}^n x_i y_i\right)\left(\sum_{i=1}^n x_i\right)}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} \quad (6.12)$$

Since Eqs. (6.11) and (6.12) contain summations that are the same, it is convenient to calculate the summations first and then to substitute them in the equations. To do this the summations are defined by:

$$S_x = \sum_{i=1}^n x_i, \quad S_y = \sum_{i=1}^n y_i, \quad S_{xy} = \sum_{i=1}^n x_i y_i, \quad S_{xx} = \sum_{i=1}^n x_i^2 \quad (6.13)$$

With these definitions, the equations for the coefficients  $a_1$  and  $a_0$  are:

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} \quad a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - (S_x)^2} \quad (6.14)$$

Equations (6.14) give the values of  $a_1$  and  $a_0$  in the equation  $y = a_1 x + a_0$  that has the best fit to  $n$  data points  $(x_i, y_i)$ . Example 6-1 shows how to use Eqs. (6.11) and (6.12) for fitting a linear equation to a set of data points.

### Example 6-1: Determination of absolute zero temperature.

According to Charles's law for an ideal gas, at constant volume, a linear relationship exists between the pressure,  $p$ , and temperature,  $T$ . In the experiment shown in the figure, a fixed volume of gas in a sealed container is submerged in ice water ( $T = 0^\circ\text{C}$ ). The temperature of the gas is then increased in ten increments up to  $T = 100^\circ\text{C}$  by heating the water, and the pressure of the gas is measured at each temperature. The data from the experiment is:

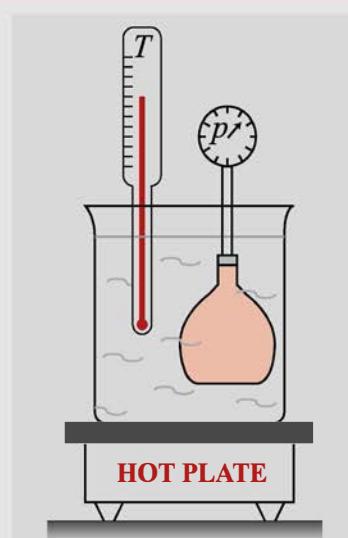
$T$ ( $^\circ\text{C}$ )	0	10	20	30	40	50	60
$p$ (atm.)	0.94	0.96	1.0	1.05	1.07	1.09	1.14

$T$ ( $^\circ\text{C}$ )	70	80	90	100
$p$ (atm.)	1.17	1.21	1.24	1.28

Extrapolate the data to determine the absolute zero temperature,  $T_0$ .

This can be done using the following steps:

- Make a plot of the data ( $p$  versus  $T$ ).
- Use linear least-squares regression to determine a linear function in the form  $p = a_1 T + a_0$  that best fits the data points. First calculate the coefficients by hand using only the four data points: 0, 30, 70, and 100  $^\circ\text{C}$ . Then write a user-defined MATLAB function that calculates the coefficients of the linear function for any number of data points and use it with all the data points to determine the coefficients.



- (c) Plot the function, and extend the line (extrapolate) until it crosses the horizontal ( $T$ ) axis. This point is an estimate of the absolute zero temperature,  $T_0$ . Determine the value of  $T_0$  from the function.

### SOLUTION

(a) A plot ( $p$  versus  $T$ ) of the data is created by MATLAB (Command Window):

```
>> T=0:10:100;
p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
>> plot(T,p,'*r')
```

The plot that is obtained is shown on the right (axes titles were added using the Plot Editor). The plot shows, as expected, a nearly linear relationship between the pressure and the temperature.

(b) Hand calculation of least-squares regression of the four data points:

$$(0, 0.94), \quad (30, 1.05), \quad (70, 1.17), \quad (100, 1.28)$$

The coefficients  $a_1$  and  $a_0$  of the equation  $p = a_1T + a_0$  that best fits the data points are determined by using Eqs. (6.14). The summations, Eqs. (6.13), are calculated first.

$$S_x = \sum_{i=1}^4 x_i = 0 + 30 + 70 + 100 = 200 \quad S_y = \sum_{i=1}^4 y_i = 0.94 + 1.05 + 1.17 + 1.28 = 4.44$$

$$S_{xx} = \sum_{i=1}^4 x_i^2 = 0^2 + 30^2 + 70^2 + 100^2 = 15800$$

$$S_{xy} = \sum_{i=1}^4 x_i y_i = 0 \cdot 0.94 + 30 \cdot 1.05 + 70 \cdot 1.17 + 100 \cdot 1.28 = 241.4$$

Substituting the  $S$ s in Eqs. (6.14) gives:

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{4 \cdot 241.4 - (200 \cdot 4.44)}{4 \cdot 15800 - 200^2} = 0.003345$$

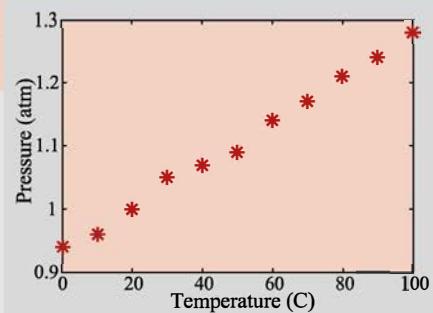
$$a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - (S_x)^2} = \frac{15800 \cdot 4.44 - (241.4 \cdot 200)}{4 \cdot 15800 - 200^2} = 0.9428$$

From this calculation, the equation that best fits the data is:  $p = 0.003345T + 0.9428$ .

Next, the problem is solved by writing a MATLAB user-defined function that calculates the coefficients of the linear function for any number of data points. The inputs to the function are two vectors with the coordinates of the data points. The outputs are the coefficients  $a_1$  and  $a_0$  of the linear equation, which are calculated with Eqs. (6.14).

#### Program 6-1: User-defined function. Linear least-squares regression.

```
function [a1,a0] = LinearRegression(x, y)
% LinearRegression calculates the coefficients a1 and a0 of the linear
% equation y = a1*x + a0 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:
```



```
% a1 The coefficient a1.
% a0 The coefficient a0.

nx=length(x);
ny=length(y);
if nx ~ = ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    a1='Error';
    a0='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);
end
```

Check if the vectors  $x$  and  $y$  have the same number of elements.

If yes, MATLAB displays an error message and the constants are not calculated.

Calculate the summation terms in Eqs. (6.13).

Calculate the coefficients  $a_1$  and  $a_0$  in Eqs. (6.14).

The user-defined function `LinearRegression` is next used in Command Window for determining the best fit line to the given points in the problem.

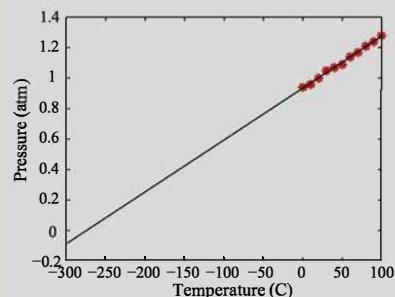
```
>> T=0:10:100;
>> p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
>> [a1, a0]=LinearRegression(T,p)
a1 =
    0.0034
a0 =
    0.9336
```

The equation that best fit the data is:  

$$p = 0.0034T + 0.9336$$

(c) The solution is done in the following script file that plots the function, the points, and calculates the value of  $T_0$  from the function.

```
T=0:10:100;
p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
Tplot=[-300 100];
pplot=0.0034*Tplot+0.9336;
plot(T,p,'*r','markersize',12)
hold on
plot(Tplot,pplot,'k')
xlabel('Temperature (C)', 'fontsize',20)
ylabel('Pressure (atm)', 'fontsize',20)
T0=-0.9336/0.0034
```



When this script file is executed, the figure shown on the right is displayed, and the value of the calculated absolute zero temperate is displayed in the Command Window, as shown below.

```
T0 =
-274.5882
```

This result is close to the handbook value of  $-273.15^\circ\text{C}$ .

### 6.3 CURVE FITTING WITH NONLINEAR EQUATION BY WRITING THE EQUATION IN A LINEAR FORM

Many situations in science and engineering show that the relationship between the quantities that are being considered is not linear. For example, Fig. 6-8 shows a plot of data points that were measured in an experiment with an  $RC$  circuit. In this experiment, the voltage across the resistor is measured as a function of time, starting when the switch is closed.

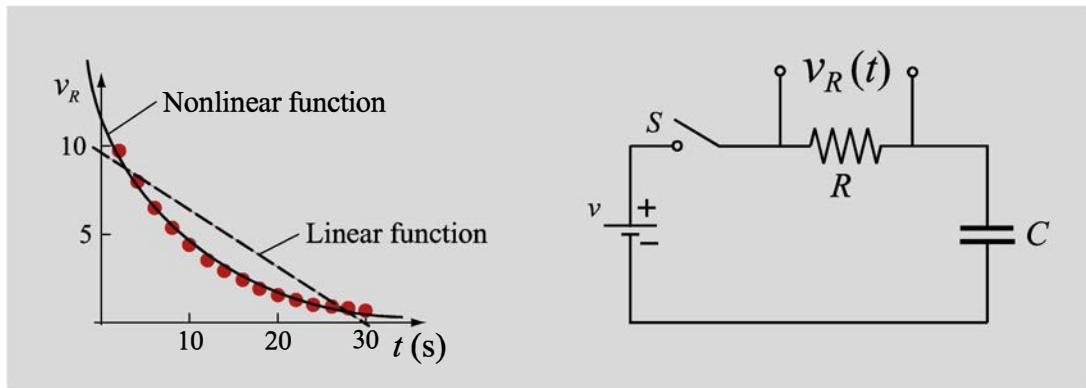


Figure 6-8: Curve-fitting points with linear equation.

The data points from the experiment are listed in Example 6-2. It is obvious from the plot that curve fitting the data points with a nonlinear function gives a much better fit than curve fitting with a linear function.

There are many kinds of nonlinear functions. This section shows curve fitting with nonlinear functions that can be written in a form for which the linear least-squares regression method can be used for determining the coefficients that give the best fit. Examples of nonlinear functions used for curve fitting in the present section are:

$$y = bx^m \quad (\text{power function})$$

$$y = be^{mx} \text{ or } y = b10^{mx} \quad (\text{exponential function})$$

$$y = \frac{1}{mx + b} \quad (\text{reciprocal function})$$

Polynomials of second, or higher, degree are also nonlinear functions. Curve fitting with such polynomials is covered separately in Section 6.4.

#### *Writing a nonlinear equation in linear form*

In order to be able to use linear regression, the form of a nonlinear equation of two variables is changed such that the new form is linear with terms that contain the original variables. For example, the power function  $y = bx^m$  can be put into linear form by taking the natural logarithm ( $\ln$ ) of both sides:

$$\ln(y) = \ln(bx^m) = m\ln(x) + \ln(b) \quad (6.15)$$

This equation is linear for  $\ln(y)$  in terms  $\ln(x)$ . The equation is in the form  $Y = a_1X + a_0$  where  $Y = \ln(y)$ ,  $a_1 = m$ ,  $X = \ln(x)$ , and  $a_0 = \ln(b)$ :

$$\begin{array}{c} \overline{\ln(y)} = \overline{m\ln(x) + \ln(b)} \\ \overline{Y} = \overline{a_1 X} + \overline{a_0} \end{array}$$

This means that linear least-squares regression can be used for curve fitting an equation of the form  $y = bx^m$  to a set of data points  $x_i, y_i$ . This is done by calculating  $a_1$  and  $a_0$  using Eqs. (6.11) and (6.12) [or (6.13) and (6.14)] while substituting  $\ln(y_i)$  for  $y_i$  and  $\ln(x_i)$  for  $x_i$ . Once  $a_1$  and  $a_0$  are known, the constants  $b$  and  $m$  in the exponential equation are calculated by:

$$m = a_1 \text{ and } b = e^{(a_0)} \quad (6.16)$$

Many other nonlinear equations can be transformed into linear form in a similar way. Table 6-2 lists several such equations.

**Table 6-2: Transforming nonlinear equations to linear form.**

Nonlinear equation	Linear form	Relationship to $Y = a_1X + a_0$	Values for linear least-squares regression	Plot where data points appear to fit a straight line
$y = bx^m$	$\ln(y) = m\ln(x) + \ln(b)$	$Y = \ln(y), \quad X = \ln(x)$ $a_1 = m, \quad a_0 = \ln(b)$	$\ln(x_i)$ and $\ln(y_i)$	$y$ vs. $x$ plot on logarithmic $y$ and $x$ axes. $\ln(y)$ vs. $\ln(x)$ plot on linear $x$ and $y$ axes.
$y = be^{mx}$	$\ln(y) = mx + \ln(b)$	$Y = \ln(y), \quad X = x$ $a_1 = m, \quad a_0 = \ln(b)$	$x_i$ and $\ln(y_i)$	$y$ vs. $x$ plot on logarithmic $y$ and linear $x$ axes. $\ln(y)$ vs. $x$ plot on linear $x$ and $y$ axes.
$y = b10^{mx}$	$\log(y) = mx + \log(b)$	$Y = \log(y), \quad X = x$ $a_1 = m, \quad a_0 = \log(b)$	$x_i$ and $\log(y_i)$	$y$ vs. $x$ plot on logarithmic $y$ and linear $x$ axes. $\log(y)$ vs. $x$ plot on linear $x$ and $y$ axes.
$y = \frac{1}{mx + b}$	$\frac{1}{y} = mx + b$	$Y = \frac{1}{y}, \quad X = x$ $a_1 = m, \quad a_0 = b$	$x_i$ and $1/y_i$	$1/y$ vs. $x$ plot on linear $x$ and $y$ axes.
$y = \frac{mx}{b+x}$	$\frac{1}{y} = \frac{b}{mx} + \frac{1}{m}$	$Y = \frac{1}{y}, \quad X = \frac{1}{x}$ $a_1 = \frac{b}{m}, \quad a_0 = \frac{1}{m}$	$1/x_i$ and $1/y_i$	$1/y$ vs. $1/x$ plot on linear $x$ and $y$ axes.

### **How to choose an appropriate nonlinear function for curve fitting**

A plot of the given data points can give an indication as to the relationship between the quantities. Whether the relationship is linear or nonlinear can be determined by plotting the points in a figure with linear axes. If in such a plot the points appear to line up along a straight line, then the relationship between the plotted quantities is linear.

A plot with linear axes in which the data points appear to line up along a curve indicates a nonlinear relationship between the plotted quantities. The question then is which nonlinear function to use for the curve fitting. Many times in engineering and science there is knowledge from a guiding theory of the physical phenomena and the form of the mathematical equation associated with the data points. For example, the process of charging a capacitor shown in Fig. 6-8 is modeled with an exponential function. If there is no knowledge of a possible form of the equation, choosing the most appropriate nonlinear function to curve-fit given data may be more difficult.

For given data points it is possible to foresee, to some extent, if a proposed nonlinear function has a potential for providing a good fit. This is done by plotting the data points in a specific way and examining whether the points appear to fit a straight line. For the functions listed in Table 6-2 this is shown in the fifth (last) column of the table. For power and exponential functions, this can be done by plotting the data using different combinations of linear and logarithmic axes. For all functions it can be done by plotting the transformed values of the data points in plots with linear axes.

For example, as was mentioned before, the data points from the experiment that are shown in Fig. 6-8 are expected to fit an exponential function. This means that a plot of the voltage  $v_R$  versus time  $t$  on a plot with a logarithmic vertical axis (for  $v_R$ ) and linear horizontal axis (for  $t$ ) should reveal that the data points will be fit by a straight line. Another option is to make a plot of  $\ln(v_R)$  vs.  $t$  on linear vertical and horizontal axes, which is also expected to show that the points line up along a straight line. Both of these plots are shown in Fig. 6-9. The figures con-

The script file for making the plots is:

```
tx=2:2:30;
vexp=[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];
vexpLOG=log(vexp)
subplot(1,2,1)
semilogy(tx,vexp,'or')
subplot(1,2,2)
plot(tx,vexpLOG,'or')
```

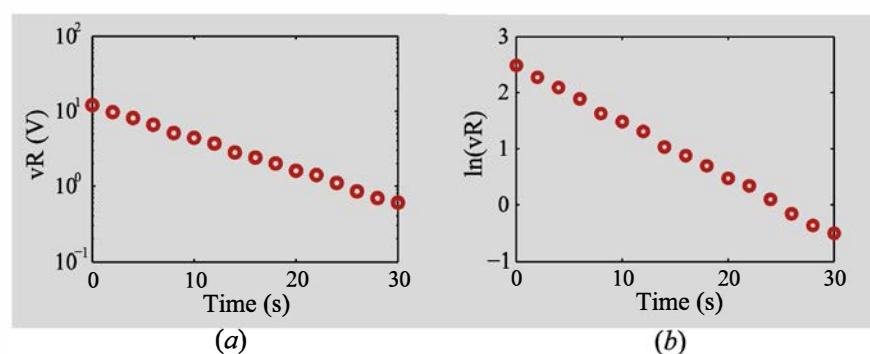


Figure 6-9: (a) A plot of  $v_R$  vs.  $t$  in a plot with a logarithmic vertical axis and linear horizontal axis. (b) A plot of  $\ln(v_R)$  vs.  $t$  in a plot with linear vertical and horizontal axes.

firm that the data from the capacitor charging experiment can be curve fit with an exponential function. The actual curve fitting is shown in Example 6-2.

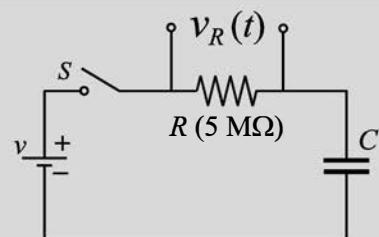
Other considerations when choosing a nonlinear function for curve fitting are as follows:

- Exponential functions cannot pass through the origin.
- Exponential functions can only fit data with all positive  $y$ s, or all negative  $y$ s.
- Logarithmic functions cannot include  $x = 0$  or negative values of  $x$ .
- For power function  $y = 0$  when  $x = 0$ .
- The reciprocal equation cannot include  $y = 0$ .

### Example 6-2: Curve fitting with a nonlinear function by writing the equation in a linear form.

An experiment with an  $RC$  circuit is used for determining the capacitance of an unknown capacitor. In the circuit, shown on the right and in Fig. 6-8, a  $5\text{-M}\Omega$  resistor is connected in series to the unknown capacitor  $C$  and a battery. The experiment starts by closing the switch and measuring the voltages,  $v_R$ , across the resistor every 2 seconds for 30 seconds. The data measured in the experiment is:

$t$ (s)	2	4	6	8	10	12	14	16	18
$v_R$ (V)	9.7	8.1	6.6	5.1	4.4	3.7	2.8	2.4	2.0
$t$ (s)	20	22	24	26	28	30			
$v_R$ (V)	1.6	1.4	1.1	0.85	0.69	0.6			



Theoretically, the voltage across the resistor as a function of time is given by the exponential function:

$$v_R = v e^{(-t/(RC))} \quad (6.17)$$

Determine the capacitance of the capacitor by curve fitting the exponential function to the data.

#### SOLUTION

It was shown in Fig. 6-9 that, as expected, an exponential function can fit the data well. The problem is solved by first determining the constants  $b$  and  $m$  in the exponential function  $v = be^{mt}$  that give the best fit of the function to the data. This is done by changing the equation to have a linear form and then using linear least-squares regression.

The linear least-squares regression is applied by using the user-defined function `LinearRegression` that was developed in the solution of Example 6-1. The inputs to the function are the values  $t_i$  and  $\ln((v_r)_i)$ . Once  $b$  and  $m$  are known, the value of  $C$  is determined by equating the coefficients in the exponent of  $e$ :

$$\frac{-1}{RC} = m \text{ solving for } C \text{ gives: } C = \frac{-1}{Rm} \quad (6.18)$$

The calculations are done by executing the following MATLAB program (script file):

**Program 6-2:** Script file. Curve fitting with a nonlinear function.

```

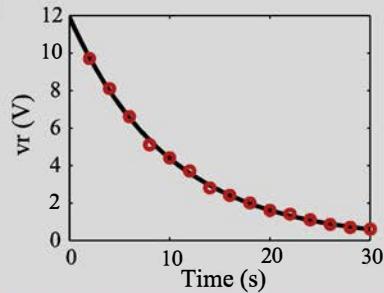
texp=2:2:30;                                         Enter the experimental data.
vexp=[9.7 8.1 6.6 5.1 4.4 3.7 2.8 2.4 2.0 1.6 1.4 1.1 0.85 0.69 0.6];
vexpLOG=log(vexp);                                     Calculate  $\ln(y_i)$  of the data points (to be used in the linear regression).
R=5E6;
[a1,a0]=LinearRegression(texp, vexpLOG)           Calculate coefficients  $a_1$  and  $a_0$  with the user-defined
b=exp(a0)                                           function LinearRegression in Example 6-1.
C=-1/(R*a1)                                         Calculate b, since  $a_0 = \ln(b)$  (see Table 6-2).
t=0:0.5:30;                                         Calculate C using Eq. (6.18).
v=b*exp(a1*t);                                     a1 is m in the equation  $v = be^{mt}$ .
plot(t,v,texp,vexp,'ro')
```

When the program is executed, the following values are displayed in the Command Window. In addition, the following plot of the data points and the curve-fitting function is displayed in the Figure Window (axes title were added interactively).

```

a1 =
-0.1002
a0 =
2.4776
b =
11.9131
C =
1.9968e-006
```

The capacitance is approximately 2  $\mu\text{F}$ .



## 6.4 CURVE FITTING WITH QUADRATIC AND HIGHER-ORDER POLYNOMIALS

### Background

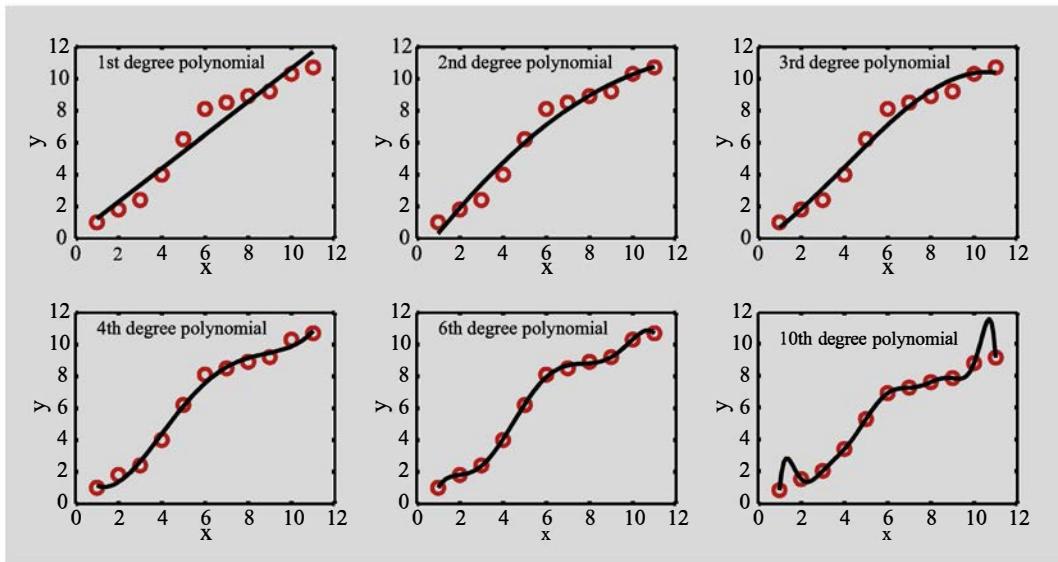
Polynomials are functions that have the form:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (6.19)$$

The coefficients  $a_n, a_{n-1}, \dots, a_1, a_0$  are real numbers, and  $n$ , which is a nonnegative integer, is the degree, or order, of the polynomial. A plot of the polynomial is a curve. A first-order polynomial is a linear function, and its plot is a straight line. Higher-order polynomials are nonlinear functions, and their plots are curves. A quadratic (second-order) polynomial is a curve that is either concave up or down (parabola). A third-order polynomial has an inflection point such that the curve can be concave up (or down) in one region, and concave down (or up) in another. In general, as the order of a polynomial increases, its curve can have more “bends.”

A given set of  $n$  data points can be curve-fit with polynomials of different order up to an order of  $(n - 1)$ . As shown later in this section, the coefficients of a polynomial can be determined such that the polynomial best fits the data by minimizing the error in a least squares

sense. Figure 6-10 shows curve fitting with polynomials of different order for the same set of 11 data points. The plots in the figure show that as the order of the polynomial increases the curve passes closer to the points. It is actually possible to have a polynomial that passes exactly through all of the points (at every point the value of the polynomial is equal to the value of the point). For  $n$  points the polynomial that



**Figure 6-10:** Curve fitting of the same set of data points with polynomials for different degrees.

passes through all of the points is one of order  $(n - 1)$ . In Fig. 6-10 it is the tenth degree polynomial (since there are 11 points).

Figure 6-10 shows that the same set of data points can be curve fit with polynomials of different order. The question as to which of the polynomials gives the best fit does not have a simple answer. It depends on the type and source of data, the engineering or science application associated with the data, and the purpose of the curve fitting. For example, if the data points themselves are not accurate (there is possibly a large error when the quantity is measured), it does not make a lot of sense to use a higher-order polynomial that follows the points closely. On the other hand, if the values of the data points are very accurate and the curve fitting is used for representing the data, curve fitting with a higher-order polynomial might be more appropriate. However, as explained in the ***important note*** that follows, use of higher-order polynomials for curve fitting is not recommended.

#### ***Important note***

As already mentioned, for any number of data points,  $n$ , it is possible to derive a polynomial (order of  $(n - 1)$ ) that passes exactly through all the points. However, when many points are involved, this polynomial is of a high degree. Although the high-order polynomial gives the exact values at all of the data points, often the polynomial deviates significantly

**between** some of the points. This can be seen in the plot with the tenth order polynomial in Fig. 6-10, where between the first two points and between the last two points the curve of the polynomial wanders away and does not follow the general trend of the data points. This means that even though the high-order polynomial gives the exact values at all the data points, it cannot be used reliably for interpolation or extrapolation. Appropriate methods for interpolation are described in Sections 6.5 and 6.6.

### Polynomial regression

Polynomial regression is a procedure for determining the coefficients of a polynomial of a second degree, or higher, such that the polynomial best fits a given set of data points. As in linear regression, the derivation of the equations that are used for determining the coefficients is based on minimizing the total error.

If the polynomial, of order  $m$ , that is used for the curve fitting is:

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x + a_0 \quad (6.20)$$

then, for a given set of  $n$  data points  $(x_i, y_i)$  ( $m$  is smaller than  $n - 1$ ), the total error is given by:

$$E = \sum_{i=1}^n [y_i - (a_m x_i^m + a_{m-1} x_i^{m-1} + \dots + a_1 x_i + a_0)]^2 \quad (6.21)$$

Since all the values  $x_i$  and  $y_i$  of the data points are known,  $E$  in Eq. (6.21) is a nonlinear function of the  $m + 1$  variables (the coefficients  $a_0$  through  $a_m$ ). The function  $E$  has a minimum at the values of  $a_0$  through  $a_m$  where the partial derivatives of  $E$  with respect to each of the variables is equal to zero. Taking the partial derivatives of  $E$  in Eq. (6.21) and setting them to zero gives a set of  $m + 1$  linear equations for the coefficients. To simplify the presentation here, the derivation for the case of  $m = 2$  (quadratic polynomial) is shown in detail. In this case Eq. (6.21) is:

$$E = \sum_{i=1}^n [y_i - (a_2 x_i^2 + a_1 x_i + a_0)]^2 \quad (6.22)$$

Taking the partial derivatives with respect to  $a_0$ ,  $a_1$ , and  $a_2$ , and setting them equal to zero gives:

$$\frac{\partial E}{\partial a_0} = -2 \sum_{i=1}^n (y_i - a_2 x_i^2 - a_1 x_i - a_0) = 0 \quad (6.23)$$

$$\frac{\partial E}{\partial a_1} = -2 \sum_{i=1}^n (y_i - a_2 x_i^2 - a_1 x_i - a_0) x_i = 0 \quad (6.24)$$

$$\frac{\partial E}{\partial a_2} = -2 \sum_{i=1}^n (y_i - a_2 x_i^2 - a_1 x_i - a_0) x_i^2 = 0 \quad (6.25)$$

Equations (6.23) through (6.25) are a system of three linear equations for the unknowns  $a_0$ ,  $a_1$ , and  $a_2$ , which can be rewritten in the form:

$$na_0 + \left( \sum_{i=1}^n x_i \right) a_1 + \left( \sum_{i=1}^n x_i^2 \right) a_2 = \sum_{i=1}^n y_i \quad (6.26)$$

$$\left( \sum_{i=1}^n x_i \right) a_0 + \left( \sum_{i=1}^n x_i^2 \right) a_1 + \left( \sum_{i=1}^n x_i^3 \right) a_2 = \sum_{i=1}^n x_i y_i \quad (6.27)$$

$$\left( \sum_{i=1}^n x_i^2 \right) a_0 + \left( \sum_{i=1}^n x_i^3 \right) a_1 + \left( \sum_{i=1}^n x_i^4 \right) a_2 = \sum_{i=1}^n x_i^2 y_i \quad (6.28)$$

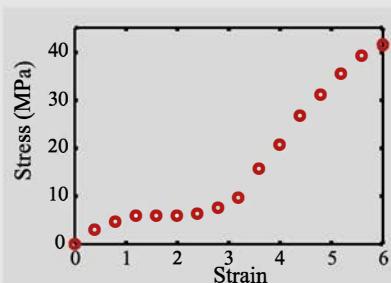
The solution of the system of equations (6.26)–(6.28) gives the values of the coefficients  $a_0$ ,  $a_1$ , and  $a_2$  of the polynomial  $y = a_2 x_i^2 + a_1 x_i + a_0$  that best fits the  $n$  data points  $(x_i, y_i)$ .

The coefficients for higher-order polynomials are derived in the same way. For an  $m$ th order polynomial, Eqs. (6.26)–(6.28) are extended to a set of  $m+1$  linear equations for the  $m+1$  coefficients. The equations for a fourth order polynomial are shown in Example 6-3.

### Example 6-3: Using polynomial regression for curve fitting of stress-strain curve.

A tension test is conducted for determining the stress-strain behavior of rubber. The data points from the test are shown in the figure, and their values are given below. Determine the fourth order polynomial that best fits the data points. Make a plot of the data points and the curve that corresponds to the polynomial.

Strain $\varepsilon$	0	0.4	0.8	1.2	1.6	2.0	2.4
Stress $\sigma$ (MPa)	0	3.0	4.5	5.8	5.9	5.8	6.2
Strain $\varepsilon$	2.8	3.2	3.6	4.0	4.4	4.8	5.2
Stress $\sigma$ (MPa)	7.4	9.6	15.6	20.7	26.7	31.1	35.6
							41.5



#### SOLUTION

A polynomial of the fourth order can be written as:

$$f(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0 \quad (6.29)$$

Curve fitting of 16 data points with this polynomial is done by polynomial regression. The values of the five coefficients  $a_0$ ,  $a_1$ ,  $a_2$ ,  $a_3$ , and  $a_4$  are obtained by solving a system of five linear equations. The five equations can be written by extending Eqs. (6.26)–(6.28).

$$na_0 + \left( \sum_{i=1}^n x_i \right) a_1 + \left( \sum_{i=1}^n x_i^2 \right) a_2 + \left( \sum_{i=1}^n x_i^3 \right) a_3 + \left( \sum_{i=1}^n x_i^4 \right) a_4 = \sum_{i=1}^n y_i \quad (6.30)$$

$$\left( \sum_{i=1}^n x_i \right) a_0 + \left( \sum_{i=1}^n x_i^2 \right) a_1 + \left( \sum_{i=1}^n x_i^3 \right) a_2 + \left( \sum_{i=1}^n x_i^4 \right) a_3 + \left( \sum_{i=1}^n x_i^5 \right) a_4 = \sum_{i=1}^n x_i y_i \quad (6.31)$$

$$\left( \sum_{i=1}^n x_i^2 \right) a_0 + \left( \sum_{i=1}^n x_i^3 \right) a_1 + \left( \sum_{i=1}^n x_i^4 \right) a_2 + \left( \sum_{i=1}^n x_i^5 \right) a_3 + \left( \sum_{i=1}^n x_i^6 \right) a_4 = \sum_{i=1}^n x_i^2 y_i \quad (6.32)$$

$$\left( \sum_{i=1}^n x_i^3 \right) a_0 + \left( \sum_{i=1}^n x_i^4 \right) a_1 + \left( \sum_{i=1}^n x_i^5 \right) a_2 + \left( \sum_{i=1}^n x_i^6 \right) a_3 + \left( \sum_{i=1}^n x_i^7 \right) a_4 = \sum_{i=1}^n x_i^3 y_i \quad (6.33)$$

$$\left( \sum_{i=1}^n x_i^4 \right) a_0 + \left( \sum_{i=1}^n x_i^5 \right) a_1 + \left( \sum_{i=1}^n x_i^6 \right) a_2 + \left( \sum_{i=1}^n x_i^7 \right) a_3 + \left( \sum_{i=1}^n x_i^8 \right) a_4 = \sum_{i=1}^n x_i^4 y_i \quad (6.34)$$

The calculations and the plot are done with MATLAB in a script file that is listed below. The computer program follows these 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  $x_i$ .

For example, the fourth element is:  $xsum(4) = \sum_{i=1}^n x_i^4$

**Step 3:** Set up the system of five linear equations (Eqs. (6.30)–(6.34)) in the form  $[a][p] = [b]$ , where  $[a]$  is the matrix with the summation terms of the powers of  $x_i$ ,  $[p]$  is the vector of the unknowns (the coefficients of the polynomial), and  $[b]$  is a vector of the summation terms on the right-hand side of Eqs. (6.30)–(6.34).

**Step 4:** Solve the system of five linear equations  $[a][p] = [b]$  (Eqs. (6.30)–(6.34)) for  $p$ , by using MATLAB's left division. The solution is a vector with the coefficients of the fourth order polynomial that best fits the data.

**Step 5:** Plot the data points and the curve-fitting polynomial.

**Program 6-3: Script file. Curve fitting using polynomial regression.**

```
clear all
x=0:0.4:6;
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);
m=4;
for i=1:2*m
    xsum(i)=sum(x.^i));
end
% Beginning of Step 3
a(1,1)=n;
b(1,1)=sum(y);
for j=2:m+1
    a(1,j)=xsum(j-1);
end
```

Assign the experimental data points to vectors  $x$  and  $y$ .

n is the number of data points.  
 m is the order of the polynomial.

Define a vector with the summation terms of the powers of  $x_i$ .

Assign the first row of the matrix  $[a]$  and the first element of the column vector  $[b]$ .

```

for i = 2:m + 1
    for j = 1:m + 1
        a(i,j) = xsum(j + i - 2);
    end
    b(i,1) = sum(x.^ (i - 1). *y);
end
% Step 4
p = (a \ b)'           Solve the system [a][p] = [b] for [p]. Transpose the solution such that [p] is a row vector.
for i = 1:m + 1
    Pcoef(i) = p(m + 2 - i); Create a new vector for the coefficients of the polynomial, to be used in
end                               MATLAB's polyval built-in function (see note at the end of the example).
epsilon = 0:0.1:6;               Define a vector of strain to be used for plotting the polynomial.
stressfit = polyval(Pcoef, epsilon); Stress calculated by the polynomial.
plot(x,y,'ro',epsilon, stressfit, 'k', 'linewidth', 2)   Plot the data points and the curve-fitting polynomial.
xlabel('Strain','fontsize',20)
ylabel('Stress (MPa)', 'fontsize',20)

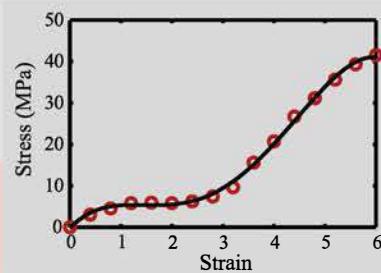
```

When the program is executed, the solution  $[p]$  is displayed in the Command Window. In addition, the plot of the data points and the curve-fitting polynomial is displayed in the Figure Window.

```
p =
-0.2746 12.8780 -10.1927 3.1185 -0.2644
```

The curve-fitting polynomial is:

$$f(x) = (-0.2644)x^4 + 3.1185x^3 - 10.1927x^2 + 12.878x - 0.2746$$



**Note:** In MATLAB a polynomial is represented by a vector whose elements are the polynomial's coefficients. The first element in the vector is the coefficient of the highest order term in the polynomial, and the last element in the vector is the coefficient  $a_0$ .

## 6.5 INTERPOLATION USING A SINGLE POLYNOMIAL

Interpolation is a procedure in which a mathematical formula is used to represent a given set of data points, such that the formula gives the exact value at all the data points and an estimated value *between* the points. This section shows how this is done by using a single polynomial, regardless of the number of points. As was mentioned in the previous section, for any number of points  $n$  there is a polynomial of order  $n - 1$  that passes through all of the points. For two points the polynomial is of first order (a straight line connecting the points). For three points the polynomial is of second order (a parabola that connects the points), and so on. This is illustrated in Fig. 6-11 which shows how first, second, third, and fourth-order polynomials connect two, three, four, and five points, respectively.

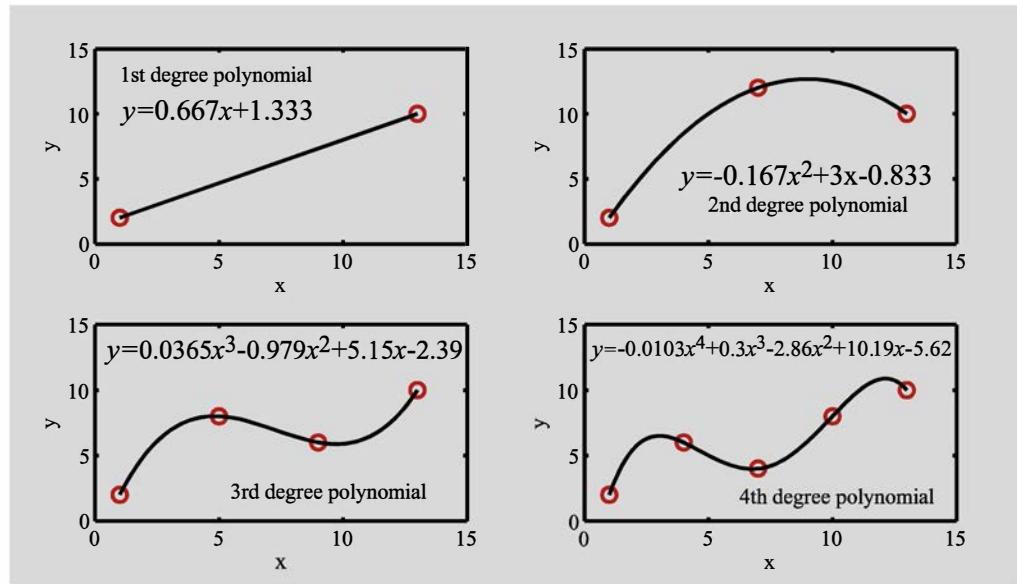


Figure 6-11: Various order polynomials.

Once the polynomial is determined, it can be used for estimating the  $y$  values between the known points simply by substituting for the  $x$  coordinate in the polynomial. Interpolation with a single polynomial gives good results for a small number of points. For a large number of points the order of the polynomial is high, and although the polynomial passes through all the points, it might deviate significantly between the points. This was shown in Fig. 6-10 for a polynomial of tenth degree and is shown later in Fig. 6-17, where a 15th-order polynomial is used for interpolation of a set of 16 data points. Consequently, interpolation with a single polynomial might not be appropriate for a large number of points. For a large number of points, better interpolation can be done by using piecewise (spline) interpolation (covered in Section 6.6) in which different lower-order polynomials are used for interpolation between different points of the same set of data points.

For a given set of  $n$  points, only one (unique) polynomial of order  $m$  ( $m = n - 1$ ) passes exactly through all of the points. The polynomial, however, can be written in different mathematical forms. This section shows how to derive three forms of polynomials (standard, Lagrange, and Newton's). The different forms are suitable for use in different circumstances.

The standard form of an  $m$ th-order polynomial is:

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x + a_0 \quad (6.35)$$

The coefficients in this form are determined by solving a system of  $m + 1$  linear equations. The equations are obtained by writing the polynomial explicitly for each point (substituting each point in the polynomial). For example, the five points ( $n = 5$ ) in the fourth degree ( $m = 4$ )

polynomial plot in Fig. 6-11 are: (1, 2), (4, 6), (7, 4), (10, 8), and (13, 10). Writing Eq. (6.35) for each of the points gives the following system of five equations for the unknowns  $a_0$ ,  $a_1$ ,  $a_2$ ,  $a_3$ , and  $a_4$ :

$$\begin{aligned} a_4 1^4 + a_3 1^3 + a_2 1^2 + a_1 1 + a_0 &= 2 \\ a_4 4^4 + a_3 4^3 + a_2 4^2 + a_1 4 + a_0 &= 6 \\ a_4 7^4 + a_3 7^3 + a_2 7^2 + a_1 7 + a_0 &= 4 \\ a_4 10^4 + a_3 10^3 + a_2 10^2 + a_1 10 + a_0 &= 8 \\ a_4 13^4 + a_3 13^3 + a_2 13^2 + a_1 13 + a_0 &= 10 \end{aligned} \quad (6.36)$$

The solution of this system of equations gives the values of the coefficients. A MATLAB solution of Eqs. (6.36) is:

```
>> a = [1 1 1 1 1; 4^4 4^3 4^2 4 1; 7^4 7^3 7^2 7 1; 10^4 10^3
10^2 10 1; 13^4 13^3 13^2 13 1]
a =
    1         1         1         1         1
    256       64        16        4         1
   2401      343       49        7         1
  10000     1000      100       10        1
  28561     2197      169       13        1
>> b = [2; 6; 4; 8; 10]
>> A = a\b
A =
-0.0103
  0.3004
-2.8580
  10.1893
 -5.6214
```

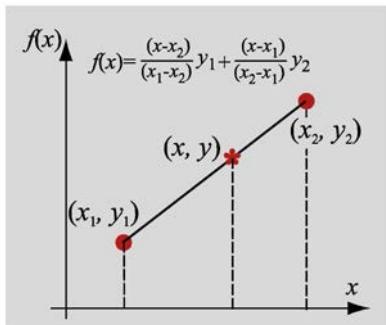
The polynomial that corresponds to these coefficients is:  
 $y = -0.0103x^4 + 0.3x^3 - 2.86x^2 + 10.19x - 5.62$   
(see Fig. 6-11).

In practice, solving the system of equations, especially for higher-order polynomials, is not efficient, and frequently the matrix of the coefficients is ill conditioned (see Section 4.11).

It is possible to write the polynomial in other forms that may be easier to use. Two such forms, the Lagrange and Newton forms, are described in the next two subsections.

### 6.5.1 Lagrange Interpolating Polynomials

Lagrange interpolating polynomials are a particular form of polynomials that can be written to fit a given set of data points by using the values at the points. The polynomials can be written right away and do not require any preliminary calculations for determining coefficients.



**Figure 6-12:** First-order Lagrange polynomial.

For two points,  $(x_1, y_1)$ , and  $(x_2, y_2)$ , the first-order Lagrange polynomial that passes through the points (Fig. 6-12) has the form:

$$f(x) = y = a_1(x - x_2) + a_2(x - x_1) \quad (6.37)$$

Substituting the two points in Eq. (6.37) gives:

$$y_1 = a_1(x_1 - x_2) + a_2(x_1 - x_1) \quad \text{or} \quad a_1 = \frac{y_1}{(x_1 - x_2)} \quad (6.38)$$

and

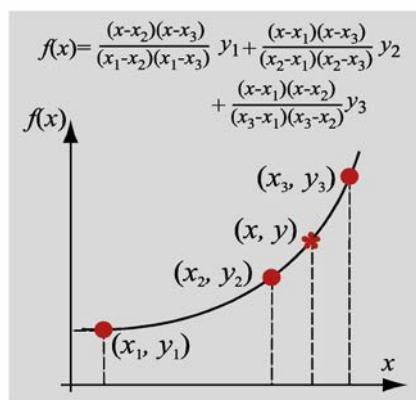
$$y_2 = a_1(x_2 - x_2) + a_2(x_2 - x_1) \quad \text{or} \quad a_2 = \frac{y_2}{(x_2 - x_1)} \quad (6.39)$$

Substituting the coefficients  $a_1$  and  $a_2$  back in Eq. (6.37) gives:

$$f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2 \quad (6.40)$$

Equation (6.40) is a linear function of  $x$  (an equation of a straight line that connects the two points). It is easy to see that if  $x = x_1$  is substituted in Eq. (6.40), the value of the polynomial is  $y_1$ , and if  $x = x_2$  is substituted, the value of the polynomial is  $y_2$ . Substituting a value of  $x$  between the points gives an interpolated value of  $y$ . Equation (6.40) can also be rewritten in the standard form  $f(x) = a_1x + a_0$ :

$$f(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)} x + \frac{x_2 y_1 - x_1 y_2}{(x_2 - x_1)} \quad (6.41)$$



**Figure 6-13:** Second-order Lagrange polynomial.

For three points,  $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$ , the second-order Lagrange polynomial that passes through the points (Fig. 6-13) has the form:

$$f(x) = y = a_1(x - x_2)(x - x_3) + a_2(x - x_1)(x - x_3) + a_3(x - x_1)(x - x_2) \quad (6.42)$$

Once the coefficients are determined such that the polynomial passes through the three points, the polynomial is:

$$f(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3 \quad (6.43)$$

Equation (6.43) is a quadratic function of  $x$ . When the coordinate  $x_1$ ,  $x_2$ , or  $x_3$  of one of the three given points is substituted in Eq. (6.43), the value of the polynomial is equal to  $y_1$ ,  $y_2$ , or  $y_3$ , respectively. This is because the coefficient in front of the corresponding  $y_i$  is equal to 1 and the coefficient of the other two terms is equal to zero.

Following the format of the polynomials in Eqs. (6.41) and (6.43), the general formula of an  $n - 1$  order Lagrange polynomial that passes through  $n$  points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  is:

$$\begin{aligned}
 f(x) = & \frac{(x-x_2)(x-x_3)\dots(x-x_n)}{(x_1-x_2)(x_1-x_3)\dots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\dots(x-x_n)}{(x_2-x_1)(x_2-x_3)\dots(x_2-x_n)} y_2 + \\
 & \dots + \frac{(x-x_1)(x-x_2)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_1)(x_i-x_2)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} y_i + \dots + \quad (6.44) \\
 & \frac{(x-x_1)(x-x_2)\dots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\dots(x_n-x_{n-1})} y_n
 \end{aligned}$$

On the right-hand side of Eq. (6.44) the numerator of the  $i$ th term does not contain  $(x-x_i)$ , and the denominator does not contain  $(x_i-x_i)$ . Consequently, when the coordinate  $x_i$  of one of the  $n$  points is substituted in Eq. (6.44), the value of the polynomial is equal to  $y_i$ . Equation (6.44) can be written in a compact form using summation and product notation as:

$$f(x) = \sum_{i=1}^n y_i L_i(x) = \sum_{i=1}^n y_i \prod_{\substack{j=1 \\ j \neq i}}^n \frac{(x-x_j)}{(x_i-x_j)} \quad (6.45)$$

where  $L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{(x-x_j)}{(x_i-x_j)}$  are called the Lagrange functions. This

form can easily be implemented in a computer program, as shown in Example 6-4.

#### ***Additional notes about Lagrange polynomials***

- The spacing between the data points does not have to be equal.
- For a given set of points, the whole expression of the interpolation polynomial has to be calculated for every value of  $x$ . In other words, the interpolation calculations for each value of  $x$  are independent of others. This is different from other forms (e.g., Eq. (6.35)) where once the coefficients of the polynomial are determined, they can be used for calculating different values of  $x$ .
- If an interpolated value is calculated for a given set of data points, and then the data set is enlarged to include additional points, all the terms of the Lagrange polynomial have to be calculated again. As shown in Section 6.5.2, this is different from Newton's polynomials where only the new terms have to be calculated if more data points are added.

Application of a Lagrange polynomial is shown in Example 6-4.

### Example 6-4: Lagrange interpolating polynomial.

The set of the following five data points is given:

$x$	1	2	4	5	7
$y$	52	5	-5	-40	10

- (a) Determine the fourth-order polynomial in the Lagrange form that passes through the points.
- (b) Use the polynomial obtained in part (a) to determine the interpolated value for  $x=3$ .
- (c) Develop a MATLAB user-defined function that interpolates using a Lagrange polynomial. The input to the function are the coordinates of the given data points and the  $x$  coordinate at the point at which the interpolated value of  $y$  is to be calculated. The output from the function is the interpolated value of  $y$  at  $x=3$ .

#### SOLUTION

- (a) Following the form of Eq. (6.44), the Lagrange polynomial for the five given points is:

$$f(x) = \frac{(x-2)(x-4)(x-5)(x-7)}{(1-2)(1-4)(1-5)(1-7)} 52 + \frac{(x-1)(x-4)(x-5)(x-7)}{(2-1)(2-4)(2-5)(2-7)} 5 + \frac{(x-1)(x-2)(x-5)(x-7)}{(4-1)(4-2)(4-5)(4-7)} (-5) + \frac{(x-1)(x-2)(x-4)(x-7)}{(5-1)(5-2)(5-4)(5-7)} (-40) + \frac{(x-1)(x-2)(x-4)(x-5)}{(7-1)(7-2)(7-4)(7-5)} 10$$

- (b) The interpolated value for  $x=3$  is obtained by substituting the  $x$  in the polynomial:

$$f(3) = \frac{(3-2)(3-4)(3-5)(3-7)}{(1-2)(1-4)(1-5)(1-7)} 52 + \frac{(3-1)(3-4)(3-5)(3-7)}{(2-1)(2-4)(2-5)(2-7)} 5 + \frac{(3-1)(3-2)(3-5)(3-7)}{(4-1)(4-2)(4-5)(4-7)} (-5) + \frac{(3-1)(3-2)(3-4)(3-7)}{(5-1)(5-2)(5-4)(5-7)} (-40) + \frac{(3-1)(3-2)(3-4)(3-5)}{(7-1)(7-2)(7-4)(7-5)} 10$$

$$f(3) = -5.778 + 2.667 - 4.444 + 13.333 + 0.222 = 6$$

- (c) The MATLAB user-defined function for interpolation using Lagrange polynomials is named `Yint=LagrangeINT(x,y,Xint)`.  $x$  and  $y$  are vectors with the coordinates of the given data points, and  $Xint$  is the coordinate of the point at which  $y$  is to be interpolated.

- The program first calculates the product terms in the Lagrange functions in Eq. (6.45). The terms are assigned to a variable (vector) named  $L$ .

$$L_i = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{(x-x_j)}{(x_i-x_j)} \quad \text{where} \quad x = Xint$$

- The program next calculates the value of the polynomial at  $x = Xint$ .

$$f(x) = \sum_{i=1}^n y_i L_i$$

#### Program 6-4: User-defined function. Interpolation using a Lagrange polynomial.

```
function Yint = LagrangeINT(x,y,Xint)
% LagrangeINT fits a Lagrange polynomial to a set of given points and
% uses the polynomial to determine 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 at which y is 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);
```

The length of the vector  $x$  gives the number of terms in the polynomial.

Calculate the product terms  $L_i$ .

Calculate the value of the polynomial  $f(x) = \sum_{i=1}^n y_i L_i$ .

The Lagrange( $x, y, Xint$ ) function is then used in the Command Window for calculating the interpolated value of  $x=3$ .

```
>> x = [1 2 4 5 7];
>> y = [52 5 -5 -40 10];
>> Yinterpolated = LagrangeINT(x,y,3)
Yinterpolated =
    6.0000
```

### 6.5.2 Newton's Interpolating Polynomials

Newton's interpolating polynomials are a popular means of exactly fitting a given set of data points. The general form of an  $n-1$  order Newton's polynomial that passes through  $n$  points is:

$$f(x) = a_1 + a_2(x-x_1) + a_3(x-x_1)(x-x_2) + \dots + a_n(x-x_1)(x-x_2)\dots(x-x_{n-1}) \quad (6.46)$$

The special feature of this form of the polynomial is that the coefficients  $a_1$  through  $a_n$  can be determined using a simple mathematical procedure. (Determination of the coefficients does not require a solution of a system of  $n$  equations.) Once the coefficients are known, the polynomial can be used for calculating an interpolated value at any  $x$ .

Newton's interpolating polynomials have additional desirable features that make them a popular choice. The data points do not have to be in descending or ascending order, or in any order. Moreover, after the  $n$  coefficients of an  $n-1$  order Newton's interpolating polynomial are determined for  $n$  given points, more points can be added to the data set and only the new additional coefficients have to be determined.

### First-order Newton's polynomial

For two given points,  $(x_1, y_1)$  and  $(x_2, y_2)$ , the first-order Newton's polynomial has the form:

$$f(x) = a_1 + a_2(x - x_1) \quad (6.47)$$

As shown in Fig. 6-14, it is an equation of a straight line that passes through the points. The coefficients  $a_1$  and  $a_2$  can be calculated by considering the similar triangles in Fig. 6-14.

$$\frac{DE}{CE} = \frac{AB}{CB}, \quad \text{or} \quad \frac{f(x) - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \quad (6.48)$$

Solving Eq. (6.48) for  $f(x)$  gives:

$$f(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \quad (6.49)$$

Comparing Eq. (6.49) with Eq. (6.47) gives the values of the coefficients  $a_1$  and  $a_2$  in terms of the coordinates of the points:

$$a_1 = y_1, \quad \text{and} \quad a_2 = \frac{y_2 - y_1}{x_2 - x_1} \quad (6.50)$$

Notice that the coefficient  $a_2$  is the slope of the line that connects the two points. As shown in Chapter 8,  $a_2$  is the two-point forward difference approximation for the first derivative at  $(x_1, y_1)$ .

### Second-order Newton's polynomial

For three given points,  $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$ , the second-order Newton's polynomial has the form:

$$f(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) \quad (6.51)$$

As shown in Fig. 6-15, it is an equation of a parabola that passes through the three points. The coefficients  $a_1$ ,  $a_2$ , and  $a_3$  can be determined by substituting the three points in Eq. (6.51). Substituting  $x = x_1$  and  $f(x_1) = y_1$  gives:  $a_1 = y_1$ . Substituting the second point,  $x = x_2$  and  $f(x_2) = y_2$ , (and  $a_1 = y_1$ ) in Eq. (6.51) gives:

$$y_2 = y_1 + a_2(x_2 - x_1) \quad \text{or} \quad a_2 = \frac{y_2 - y_1}{x_2 - x_1} \quad (6.52)$$

Substituting the third point,  $x = x_3$  and  $f(x_3) = y_3$  (as well as

$a_1 = y_1$  and  $a_2 = \frac{y_2 - y_1}{x_2 - x_1}$ ) in Eq. (6.51) gives:

$$y_3 = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2) \quad (6.53)$$

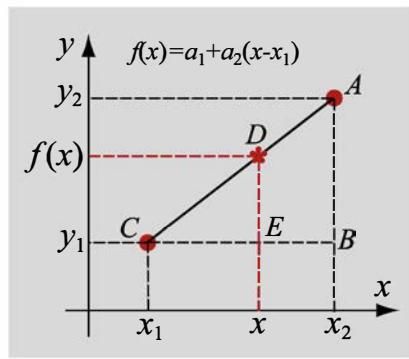


Figure 6-14: First-order Newton's polynomial.

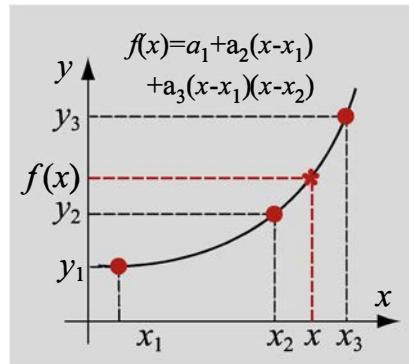


Figure 6-15: Second-order Newton's polynomial.

Equation (6.53) can be solved for  $a_3$  and rearranged to give (after some algebra):

$$a_3 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{(x_3 - x_1)} \quad (6.54)$$

The coefficients  $a_1$ , and  $a_2$  are the same in the first-order and second-order polynomials. This means that if two points are given and a first-order Newton's polynomial is fit to pass through those points, and then a third point is added, the polynomial can be changed to be of second-order and pass through the three points by only determining the value of one additional coefficient.

### **Third-order Newton's polynomial**

For four given points,  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$  and  $(x_4, y_4)$ , the third-order Newton's polynomial that passes through the four points has the form:

$$f(x) = y = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + a_4(x - x_1)(x - x_2)(x - x_3) \quad (6.55)$$

The formulas for the coefficients  $a_1$ ,  $a_2$ , and  $a_3$  are the same as for the second order polynomial. The formula for the coefficient  $a_4$  can be obtained by substituting  $(x_4, y_4)$ , in Eq. (6.55) and solving for  $a_4$ , which gives:

$$a_4 = \frac{\left( \frac{y_4 - y_3}{x_4 - x_3} - \frac{y_3 - y_2}{x_3 - x_2} \right) - \left( \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1} \right)}{\frac{(x_4 - x_2)}{(x_4 - x_1)}} \quad (6.56)$$

### **A general form of Newton's polynomial and its coefficients**

A careful examination of the equations for the coefficients  $a_2$  (Eq. (6.52)),  $a_3$ , (Eq. (6.54)) and  $a_4$ , (Eq. (6.56)) shows that the expressions follow a certain pattern. The pattern can be clarified by defining so-called **divided differences**.

For two points,  $(x_1, y_1)$ , and  $(x_2, y_2)$ , the first divided difference, written as  $f[x_2, x_1]$ , is defined as the slope of the line connecting the two points:

$$f[x_2, x_1] = \frac{y_2 - y_1}{x_2 - x_1} = a_2 \quad (6.57)$$

The first divided difference is equal to the coefficient  $a_2$ .

For three points  $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$  the second divided difference, written as  $f[x_3, x_2, x_1]$ , is defined as the difference between the first divided differences of points  $(x_3, y_3)$ , and  $(x_2, y_2)$ , and points  $(x_2, y_2)$ , and  $(x_1, y_1)$  divided by  $(x_3 - x_1)$ :

$$f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{(x_3 - x_1)} = a_3 \quad (6.58)$$

The second divided difference is thus equal to the coefficient  $a_3$ .

For four points  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$ , and  $(x_4, y_4)$  the third divided difference, written as  $f[x_4, x_3, x_2, x_1]$ , is defined as the difference between the second divided differences of points  $(x_2, y_2)$ ,  $(x_3, y_3)$  and  $(x_4, y_4)$ , and points  $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$  divided by  $(x_4 - x_1)$ :

$$\begin{aligned} f[x_4, x_3, x_2, x_1] &= \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1} \\ &= \frac{\frac{f[x_4, x_3] - f[x_3, x_2]}{x_4 - x_2} - \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1}}{(x_4 - x_1)} \\ &= \frac{\frac{y_4 - y_3}{x_4 - x_3} - \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{(x_4 - x_2) - (x_3 - x_1)} = a_4 \end{aligned} \quad (6.59)$$

The third divided difference is thus equal to the coefficient  $a_4$ .

The next (fourth) divided difference (when five data points are given) is:

$$f[x_5, x_4, x_3, x_2, x_1] = \frac{f[x_5, x_4, x_3, x_2] - f[x_4, x_3, x_2, x_1]}{x_5 - x_1} = a_5 \quad (6.60)$$

If more data points are given, the procedure for calculating higher differences continues in the same manner. In general, when  $n$  data points are given, the procedure starts by calculating  $(n - 1)$  first divided differences. Then,  $(n - 2)$  second divided differences are calculated from the first divided differences. This is followed by calculating  $(n - 3)$  third divided differences from the second divided differences. The process ends when one  $n$ th divided difference is calculated from two  $(n - 1)$  divided differences to give the coefficient  $a_n$ .

The procedure for finding the coefficients by using divided differences can be followed in a divided difference table. Such a table for the case of five data points is shown in Fig. 6-16.

In general terms, for  $n$  given data points,  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , the first divided differences between two points  $(x_i, y_i)$ , and  $(x_j, y_j)$  are given by:

$$f[x_j, x_i] = \frac{y_j - y_i}{x_j - x_i} \quad (6.61)$$

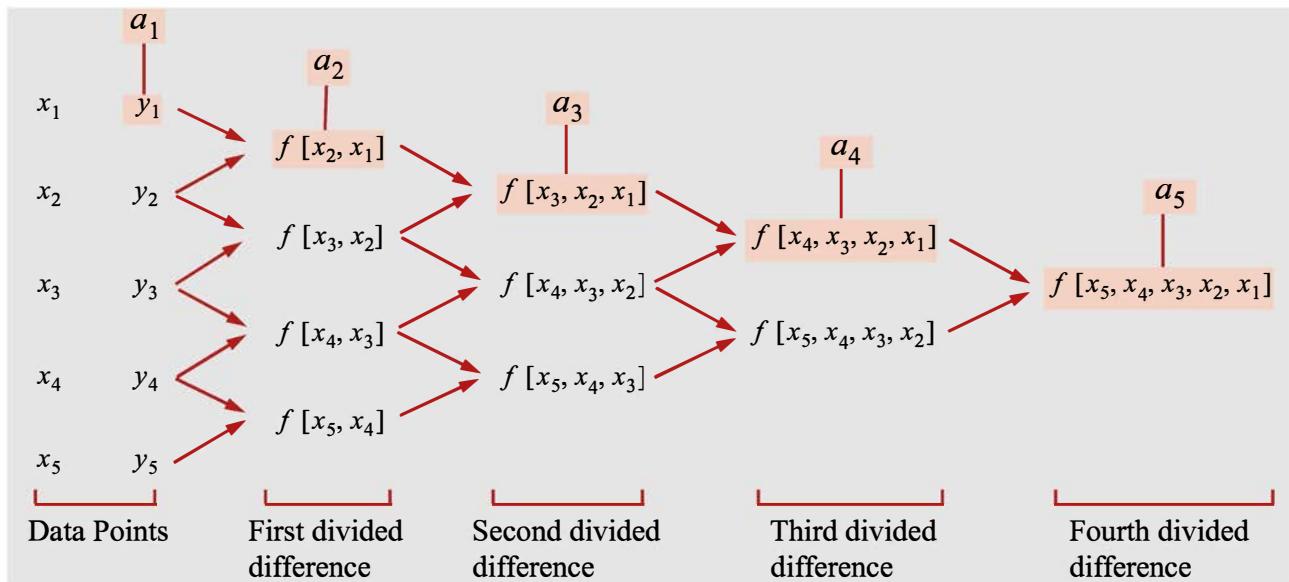


Figure 6-16: Table of divided differences for five data points.

The  $k$ th divided difference for second and higher divided differences up to the  $(n-1)$  difference is given by:

$$f[x_k, x_{k-1}, \dots, x_2, x_1] = \frac{f[x_k, x_{k-1}, \dots, x_3, x_2] - f[x_{k-1}, x_{k-2}, \dots, x_2, x_1]}{x_k - x_1} \quad (6.62)$$

With these definitions, the  $(n-1)$  order Newton's polynomial, Eq. (6.46) is given by:

$$f(x) = y = y_1 + f[x_2, x_1](x-x_1) + f[x_3, x_2, x_1](x-x_1)(x-x_2) + \dots + f[x_n, x_{n-1}, \dots, x_2, x_1](x-x_1)(x-x_2)\dots(x-x_{n-1}) \quad (6.63)$$

### Notes about Newton's polynomials

- The spacings between the data points do not have to be the same.
- For a given set of  $n$  points, once the coefficients  $a_1$  through  $a_n$  are determined, they can be used for interpolation at any point between the data points.

After the coefficients  $a_1$  through  $a_n$  are determined (for a given set of  $n$  points), additional data points can be added (they do not have to be in order), and only the additional coefficients have to be determined.

Example 6-5 shows application of Newton's interpolating polynomials.

### Example 6-5: Newton's interpolating polynomial.

The set of the following five data points is given:

$x$	1	2	4	5	7
$y$	52	5	-5	-40	10

- (a) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
- (b) Use the polynomial obtained in part (a) to determine the interpolated value for  $x=3$ .
- (c) Write a MATLAB user-defined function that interpolates using Newton's polynomial. The input to the function should be the coordinates of the given data points and the  $x$  coordinate of the point at which  $y$  is to be interpolated. The output from the function is the  $y$  value of the interpolated point.

#### SOLUTION

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

$$f(x) = y = a_1 + a_2(x-1) + a_3(x-1)(x-2) + a_4(x-1)(x-2)(x-4) + a_5(x-1)(x-2)(x-4)(x-5)$$

The coefficients can be determined by the following divided difference table:

$x_i$	$y_i$	$a_1 = 52$	$a_2 = -47$	$a_3 = 14$	$a_4 = -6$	$a_5 = 2$
1	52					
2	5	$\frac{5-52}{2-1} = -47$		$\frac{-5-(-47)}{4-1} = 14$		
4	-5	$\frac{-5-5}{4-2} = -5$	$\frac{-35-(-5)}{5-2} = -10$		$\frac{-10-14}{5-1} = -6$	
5	-40	$\frac{-40-(-5)}{5-4} = -35$	$\frac{25-(-35)}{7-4} = 20$		$\frac{20-(-10)}{7-2} = 6$	$\frac{6-(-6)}{7-1} = 2$
7	10	$\frac{10-(-40)}{7-5} = 25$				

With the coefficients determined, the polynomial is:

$$f(x) = y = 52 - 47(x-1) + 14(x-1)(x-2) - 6(x-1)(x-2)(x-4) + 2(x-1)(x-2)(x-4)(x-5)$$

- (b) The interpolated value for  $x=3$  is obtained by substituting for  $x$  in the polynomial:

$$f(3) = y = 52 - 47(3-1) + 14(3-1)(3-2) - 6(3-1)(3-2)(3-4) + 2(3-1)(3-2)(3-4)(3-5) = 6$$

- (c) The MATLAB user-defined function for Newton's interpolation is named `Yint=NewtonSINT(x, y, Xint)`.  $x$  and  $y$  are vectors with the coordinates of the given data points, and  $Xint$  is the coordinate of the point at which  $y$  is to be interpolated.

- The program starts by calculating the first divided differences, which are then used for calculating the higher divided differences. The values are assigned to a table named `divDIF`.
- The coefficients of the polynomial (first row of the table) are then assigned to a vector named `a`.

- The known polynomial is used for interpolation.

**Program 6-5: User-defined function. Interpolation using Newton's polynomial.**

```

function Yint = NewtonsINT(x,y,Xint)
% NewtonsINT fits a Newtons 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);           The length of the vector x gives the number of coefficients (and terms) of the polynomial.
a(1)= y(1);              The first coefficient  $a_1$ .
for i = 1:n - 1
    divDIF(i,1)=(y(i+1)-y(i))/(x(i+1)-x(i));
end
for j = 2:n - 1
    for i = 1:n - j
        divDIF(i,j)=(divDIF(i+1,j-1)-divDIF(i,j-1))/(x(j+i)-x(i));
    end
end
for j = 2:n
    a(j) = divDIF(1,j - 1);
end
Yint = a(1);
xn = 1;
for k = 2:n
    xn = xn*(Xint - x(k - 1));
    Yint = Yint + a(k)*xn;
end

```

Calculate the finite divided differences. They are assigned to the first column of `divDIF`.

Calculate the second and higher divided differences (up to an order of  $(n-1)$ ). The values are assigned in columns to `divDIF`.

Assign the coefficients  $a_2$  through  $a_n$ . to vector `a`.

Calculate the interpolated value of `Xint`. The first term in the polynomial is  $a_1$ . The following terms are added by using a loop.

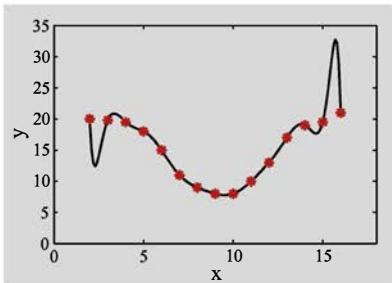
The `NewtonSINT(x,y,Xint)` Function is then used in the Command Window for calculating the interpolated value of  $x=3$ .

```

>> x = [1 2 4 5 7];
>> y = [52 5 -5 -40 10];
>> Yinterpolated = NewtonsINT(x,y,3)
Yinterpolated =

```

## 6.6 PIECEWISE (SPLINE) INTERPOLATION



**Figure 6-17:** Fitting 16 data points with a 15th order polynomial.

When a set of  $n$  data points is given and a single polynomial is used for interpolation between the points, the polynomial gives the exact values at the points (passes through the points) and yields estimated (interpolated) values between the points. When the number of points is small such that the order of the polynomial is low, typically the interpolated values are reasonably accurate. However, as already mentioned in Section 6.4, large errors might occur when a high-order polynomial is used for interpolation involving a large number of points. This is shown in Fig. 6-17 where a polynomial of 15th order is used for interpolation with a set of 16 data points. It is clear from the figure that near the ends the polynomial deviates significantly from the trend of the data, and thus cannot be reliably used for interpolation.

When a large number of points is involved, a better interpolation can be obtained by using many low-order polynomials instead of a single high-order polynomial. Each low-order polynomial is valid in one interval between two or several points. Typically, all of the polynomials are of the same order, but the coefficients are different in each interval. When first-order polynomials are used, the data points are connected with straight lines. For second-order (quadratic), and third-order (cubic) polynomials, the points are connected by curves. Interpolation in this way is called *piecewise*, or *spline*, interpolation. The data points where the polynomials from two adjacent intervals meet are called *knots*. The name “spline” comes from a draftsman’s spline, which is a thin flexible rod used to physically interpolate over discrete points marked by pegs.

The three types of spline interpolation are linear, quadratic, and cubic.

### 6.6.1 Linear Splines

With linear splines, interpolation is carried out by using a first-order polynomial (linear function) between the points (the points are connected with straight lines), as shown in Fig. 6-18. Using the Lagrange form, the equation of the straight line that connects the first two points is given by:

$$f_1(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2 \quad (6.64)$$

For  $n$  given points, there are  $n - 1$  intervals. The interpolation in interval  $i$ , which is between points  $x_i$  and  $x_{i+1}$  ( $x_i \leq x \leq x_{i+1}$ ), is done by using the equation of the straight line that connects point  $(x_i, y_i)$  with point  $(x_{i+1}, y_{i+1})$ :

$$f_i(x) = \frac{(x - x_{i+1})}{(x_i - x_{i+1})} y_i + \frac{(x - x_i)}{(x_{i+1} - x_i)} y_{i+1} \text{ for } i = 1, 2, \dots, n-1 \quad (6.65)$$

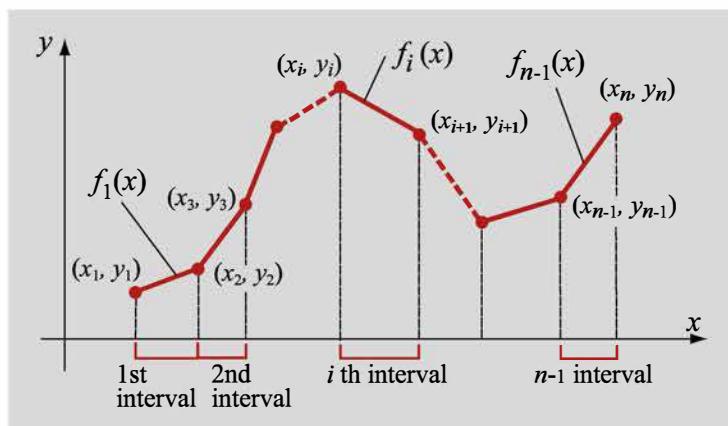


Figure 6-18: Linear splines.

It is obvious that linear splines give continuous interpolation since the two adjacent polynomials have the same value at a common knot. There is, however, a discontinuity in the slope of the linear splines at the knots.

Interpolation with linear splines is easy to calculate and program, and gives good results when the data points are closely spaced. Example 6-6 shows a numerical application of linear splines by hand and by using a user-defined MATLAB function.

### Example 6-6: Linear splines.

The set of the following four data points is given:

$x$	8	11	15	18
$y$	5	9	10	8

- (a) Determine the linear splines that fit the data.
- (b) Determine the interpolated value for  $x = 12.7$ .
- (c) Write a MATLAB user-defined function for interpolation with linear splines. The inputs to the function are the coordinates of the given data points and the  $x$  coordinate of the point at which  $y$  is to be interpolated. The output from the function is the interpolated  $y$  value at the given point. Use the function for determining the interpolated value of  $y$  for  $x = 12.7$ .

#### SOLUTION

- (a) There are four points and thus three splines. Using Eq. (6.65) the equations of the splines are:

$$f_1(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2 = \frac{(x - 11)}{(8 - 11)} 5 + \frac{(x - 8)}{(11 - 8)} 9 = \frac{5}{-3}(x - 11) + \frac{9}{2}(x - 8) \quad \text{for } 8 \leq x \leq 11$$

$$f_2(x) = \frac{(x - x_3)}{(x_2 - x_3)} y_2 + \frac{(x - x_2)}{(x_3 - x_2)} y_3 = \frac{(x - 15)}{(11 - 15)} 9 + \frac{(x - 11)}{(15 - 11)} 10 = \frac{9}{-4}(x - 15) + \frac{10}{4}(x - 11) \quad \text{for } 11 \leq x \leq 15$$

$$f_3(x) = \frac{(x - x_4)}{(x_3 - x_4)} y_3 + \frac{(x - x_3)}{(x_4 - x_3)} y_4 = \frac{(x - 18)}{(15 - 18)} 10 + \frac{(x - 15)}{(18 - 15)} 8 = \frac{10}{-3}(x - 18) + \frac{8}{3}(x - 15) \quad \text{for } 15 \leq x \leq 18$$

- (b) The interpolated value of  $y$  for  $x = 12.7$  is obtained by substituting the value of  $x$  in the equation for  $f_2(x)$  above:

$$f_2(12.7) = \frac{9}{4}(12.7 - 15) + \frac{10}{4}(12.7 - 11) = 9.425$$

(c) The MATLAB user-defined function for linear spline interpolation is named `Yint=LinearSpline(x, y, Xint)`. `x` and `y` are vectors with the coordinates of the given data points, and `Xint` is the coordinate of the point at which `y` is to be interpolated.

**Program 6-6: User-defined function. Linear splines.**

```
function Yint = LinearSpline(x, y, Xint)
% LinearSpline calculates interpolation using linear splines.
% 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 coordinate of the interpolated point.
% Output variable:
% Yint The y value of the interpolated point.

n = length(x);
for i = 2:n
    if Xint < x(i)
        break
    end
end
Yint=(Xint-x(i))*y(i-1)/(x(i-1)-x(i))+(Xint-x(i-1))*y(i)/(x(i)-x(i-1));
```

The length of the vector `x` gives the number of terms in the data.

Find the interval that includes `Xint`.

Calculate `Yint` with Eq. (6.65).

The `LinearSpline(x, y, Xint)` function is then used in the Command Window for calculating the interpolated value of  $x = 12.7$ .

```
>> x = [8 11 15 18];
>> y = [5 9 10 8];
>> Yint = LinearSpline(x,y,12.7)
Yint =
    9.4250
```

### 6.6.2 Quadratic Splines

In quadratic splines, second-order polynomials are used for interpolation in the intervals between the points (Fig. 6-19). For  $n$  given points there are  $n - 1$  intervals, and using the standard form, the equation of the polynomial in the  $i$ th interval, between points  $x_i$  and  $x_{i+1}$ , is given by:

$$f_i(x) = a_i x^2 + b_i x + c_i \text{ for } i = 1, 2, \dots, n-1 \quad (6.66)$$

Overall, there are  $n - 1$  equations, and since each equation has three coefficients, a total of  $3(n - 1) = 3n - 3$  coefficients have to be determined. The coefficients are determined by applying the following conditions:

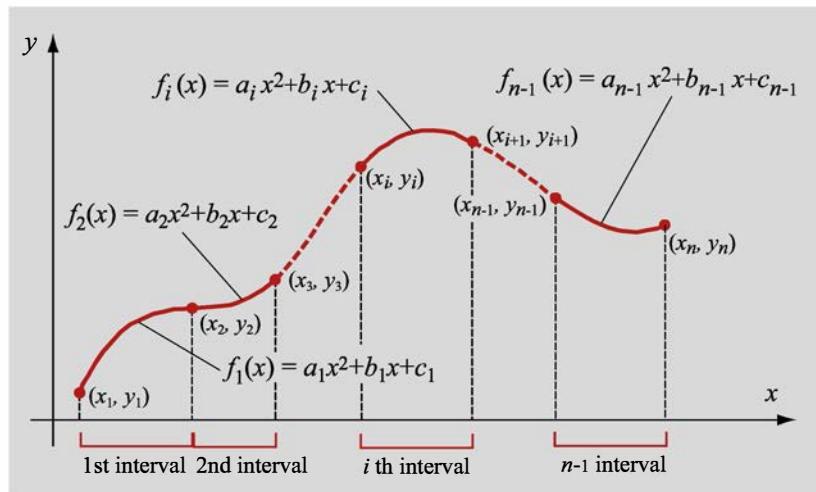


Figure 6-19: Quadratic splines.

- Each polynomial  $f_i(x)$  must pass through the endpoints of the interval,  $(x_i, y_i)$  and  $(x_{i+1}, y_{i+1})$ , which means that  $f_i(x_i) = y_i$  and  $f_i(x_{i+1}) = y_{i+1}$ :

$$a_i x_i^2 + b_i x_i + c_i = y_i \quad \text{for } i = 1, 2, \dots, n-1 \quad (6.67)$$

$$a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \quad \text{for } i = 1, 2, \dots, n-1 \quad (6.68)$$

Since there are  $n-1$  intervals, these conditions give  $2(n-1) = 2n-2$  equations.

- At the interior knots, the slopes (first derivative) of the polynomials from adjacent intervals are equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the slope is continuous. In general, the first derivative of the  $i$ th polynomial is:

$$f'(x) = \frac{df}{dx} = 2a_i x + b_i \quad (6.69)$$

For  $n$  points, the first interior point is  $i = 2$ , and the last is  $i = n-1$ . Equating the successive first derivatives at all of the interior points gives:

$$2a_{i-1}x_i + b_{i-1} = 2a_i x_i + b_i \quad \text{for } i = 2, 3, \dots, n-1 \quad (6.70)$$

Since there are  $n-2$  interior points, this condition gives  $n-2$  equations.

Together, the two conditions give  $3n-4$  equations. However, the  $n-1$  polynomials have  $3n-3$  coefficients so that one more equation (condition) is needed in order to solve for all of the coefficients. An additional condition that is commonly applied is that the second derivative at either the first point or the last point is zero. Consider the first choice:

3. The second derivative at the first point,  $(x_1, y_1)$ , is zero. The polynomial in the first interval (between the first and the second points) is:

$$f_1(x) = a_1x^2 + b_1x + c_1 \quad (6.71)$$

The second derivative of the polynomial is  $f_1''(x) = 2a_1$ , and equating it to zero means that  $a_1 = 0$ . This condition actually means that a straight line connects the first two points (the slope is constant).

#### A note on quadratic and cubic splines

Quadratic splines have a continuous first derivative at the interior points (knots), and for  $n$  given points they require the solution of a linear system of  $3n - 4$  equations for the coefficients of the polynomials. As is shown in the next section, cubic splines have continuous first and second derivatives at the interior points, and can be written in a form that requires the solution of a linear system of only  $n - 2$  equations for the coefficients.

Example 6-7 shows an application of quadratic splines for interpolation of a given set of five points.

#### Example 6-7: Quadratic splines.

The set of the following five data points is given:

$x$	8	11	15	18	22
$y$	5	9	10	8	7

- (a) Determine the quadratic splines that fit the data.
- (b) Determine the interpolated value of  $y$  for  $x = 12.7$ .
- (c) Make a plot of the data points and the interpolating polynomials.

#### SOLUTION

- (a) There are five points ( $n = 5$ ) and thus four splines ( $i = 1, \dots, 4$ ). 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, and since each polynomial has three coefficients, a total of 12 coefficients have to be determined. The 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 equal to zero (see condition 3). The other 11 coefficients are determined from a linear system of 11 equations.

Eight equations are obtained from the condition that the polynomial in each interval passes through the endpoints, Eqs. (6.67) and (6.68):

$$i = 1 \quad f_1(x) = a_1 x_1^2 + b_1 x_1 + c_1 = b_1 8 + c_1 = 5$$

$$f_1(x) = a_1 x_2^2 + b_1 x_2 + c_1 = b_1 11 + c_1 = 9$$

$$i = 2 \quad f_2(x) = a_2 x_2^2 + b_2 x_2 + c_2 = a_2 11^2 + b_2 11 + c_2 = 9$$

$$f_2(x) = a_2 x_3^2 + b_2 x_3 + c_2 = a_2 15^2 + b_2 15 + c_2 = 10$$

$$i = 3 \quad f_3(x) = a_3 x_3^2 + b_3 x_3 + c_3 = a_3 15^2 + b_3 15 + c_3 = 10$$

$$f_3(x) = a_3 x_4^2 + b_3 x_4 + c_3 = a_3 18^2 + b_3 18 + c_3 = 8$$

$$i = 4 \quad f_4(x) = a_4 x_4^2 + b_4 x_4 + c_4 = a_4 18^2 + b_4 18 + c_4 = 8$$

$$f_4(x) = a_4 x_5^2 + b_4 x_5 + c_4 = a_4 22^2 + b_4 22 + c_4 = 7$$

Three equations are obtained from the condition that at the interior knots the slopes (first derivative) of the polynomials from adjacent intervals are equal, Eq. (6.70).

$$i = 2 \quad 2a_1 x_2 + b_1 = 2a_2 x_2 + b_2 \longrightarrow b_1 = 2a_2 11 + b_2 \quad \text{or:} \quad b_1 - 2a_2 11 - b_2 = 0$$

$$i = 3 \quad 2a_2 x_3 + b_2 = 2a_3 x_3 + b_3 \longrightarrow 2a_2 15 + b_2 = 2a_3 15 + b_3 \quad \text{or:} \quad 2a_2 15 + b_2 - 2a_3 15 - b_3 = 0$$

$$i = 4 \quad 2a_3 x_4 + b_3 = 2a_4 x_4 + b_4 \longrightarrow 2a_3 18 + b_3 = 2a_4 18 + b_4 \quad \text{or:} \quad 2a_3 18 + b_3 - 2a_4 18 - b_4 = 0$$

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

$$\left[ \begin{array}{cccccccccc|c} 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b_1 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_1 \\ 0 & 0 & 11^2 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & a_2 \\ 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & c_2 \\ 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 & 0 & 0 & a_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & b_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22^2 & 22 & c_3 \\ 1 & 0 & -22 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & a_4 \\ 0 & 0 & 30 & 1 & 0 & -30 & -1 & 0 & 0 & 0 & b_4 \\ 0 & 0 & 0 & 0 & 0 & 36 & 1 & 0 & -36 & -1 & c_4 \end{array} \right] = \begin{bmatrix} 5 \\ 9 \\ 9 \\ 10 \\ 10 \\ 8 \\ 8 \\ 7 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (6.72)$$

The system in Eq. (6.72) is solved with MATLAB:

```
>> A = [8 1 0 0 0 0 0 0 0 0 0; 11 1 0 0 0 0 0 0 0 0 0; 0 0 11^2 11 1 0 0 0 0 0 0
0 0 15^2 15 1 0 0 0 0 0 0; 0 0 0 0 0 15^2 15 1 0 0 0; 0 0 0 0 0 0 18^2 18 1 0 0 0
0 0 0 0 0 0 0 0 18^2 18 1; 0 0 0 0 0 0 0 0 22^2 22 1; 1 0 -22 -1 0 0 0 0 0 0 0;
0 0 30 1 0 -30 -1 0 0 0 0; 0 0 0 0 0 36 1 0 -36 -1 0];
>> B = [5; 9; 9; 10; 10; 8; 8; 7; 0; 0; 0];
>> coefficients = (A\B)';
coefficients =
```

```
1.3333 -5.6667 -0.2708 7.2917 -38.4375 0.0556 -2.5000 35.0000 0.0625 -2.7500 37.2500
```

With the coefficients known, the polynomials are:

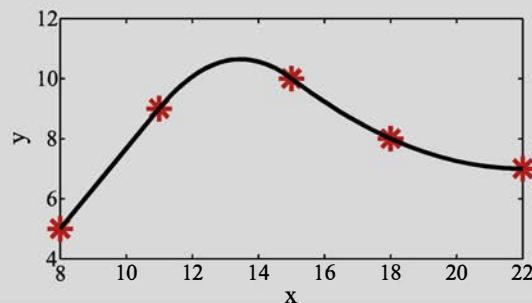
$$f_1(x) = 1.333x - 5.6667 \text{ for } 8 \leq x \leq 11, \quad f_2(x) = (-0.2708)x^2 + 7.2917x - 38.4375 \text{ for } 11 \leq x \leq 15$$

$$f_3(x) = 0.0556x^2 - 2.5x + 35 \text{ for } 15 \leq x \leq 18, \quad f_4(x) = 0.0625x^2 - 2.75x + 37.25 \text{ for } 18 \leq x \leq 22$$

(b) The interpolated value of  $y$  for  $x = 12.7$  is calculated by substituting the value of  $x$  in  $f_2(x)$ :

$$f_2(12.7) = (-0.2708) \cdot 12.7^2 + 7.2917 \cdot 12.7 - 38.4375 = 10.4898$$

(c) The plot on the right shows the data points and the polynomials. The plot clearly shows that the first spline is a straight line (constant slope).



### 6.6.3 Cubic Splines

In cubic splines, third-order polynomials are used for interpolation in the intervals between the points. For  $n$  given points there are  $n - 1$  intervals, and since each third-order polynomial has four coefficients the determination of all of the coefficients may require a large number of calculations. As was explained earlier in this chapter, polynomials can be written in different forms (standard, Lagrange, Newton) and theoretically any of these forms can be used for cubic splines. Practically, however, the amount of calculations that have to be executed for determining all the coefficients varies greatly with the form of the polynomial that is used. The presentation that follows shows two derivations of cubic splines. The first uses the standard form of the polynomials, and the second uses a variation of the Lagrange form. The derivation with the standard form is easier to follow, understand, and use (it is similar to the derivation of the quadratic splines), but it requires the solution of a system of  $4n - 4$  linear equations. The derivation that is based on the Lagrange form is more sophisticated, but requires the solution of a system of only  $n - 2$  linear equations.

#### Cubic splines with standard form polynomials

For  $n$  given points, as shown in Fig. 6-20, there are  $n - 1$  intervals, and using the standard form, the equation of the polynomial in the  $i$ th interval, between points  $x_i$  and  $x_{i+1}$  is given by:

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \quad (6.73)$$

Overall, there are  $n - 1$  equations, and since each equation has four coefficients, a total of  $4(n - 1) = 4n - 4$  coefficients have to be determined. The coefficients are found by applying the following conditions:

1. Each polynomial  $f_i(x)$  must pass through the endpoints of the interval,  $(x_i, y_i)$  and  $(x_{i+1}, y_{i+1})$ , which means that  $f_i(x_i) = y_i$  and  $f_i(x_{i+1}) = y_{i+1}$ :

$$a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = y_i \quad \text{for } i = 1, 2, \dots, n - 1 \quad (6.74)$$

$$a_i x_{i+1}^3 + b_i x_{i+1}^2 + c_i x_{i+1} + d_i = y_{i+1} \quad \text{for } i = 1, 2, \dots, n - 1 \quad (6.75)$$

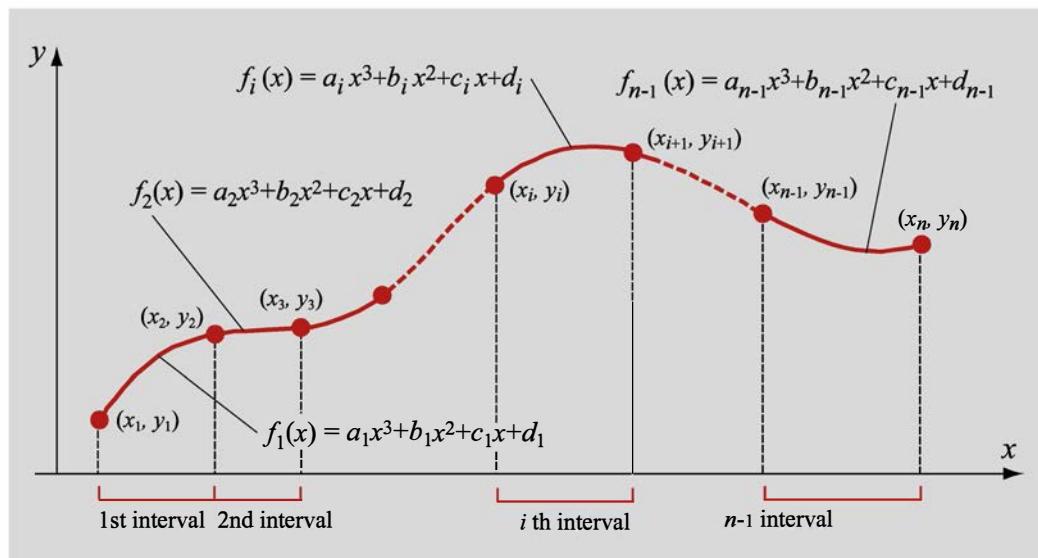


Figure 6-20: Cubic splines.

Since there are  $n - 1$  intervals, this condition gives  $2(n - 1) = 2n - 2$  equations.

2. At the interior knots, the slopes (first derivatives) of the polynomials from the adjacent intervals are equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the slope must be continuous. The first derivative of the  $i$ th polynomial is:

$$f_i'(x) = \frac{d f_i}{dx} = 3a_i x^2 + 2b_i x + c_i \quad (6.76)$$

For  $n$  points the first interior point is  $i = 2$ , and the last is  $i = n - 1$ . Equating the first derivatives at each interior point gives:

$$3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_{i-1} = 3a_i x_i^2 + 2b_i x_i + c_i \text{ for } i = 2, 3, \dots, n - 1 \quad (6.77)$$

Since there are  $n - 2$  interior points, this condition gives  $n - 2$  additional equations.

3. At the interior knots, the second derivatives of the polynomials from adjacent intervals must be equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the rate of change of the slope (curvature) must be continuous. The second derivative of the polynomial in the  $i$ th interval is:

$$f_i''(x) = \frac{d^2 f_i}{dx^2} = 6a_i x + 2b_i \quad (6.78)$$

For  $n$  points, the first interior point is  $i = 2$ , and the last is  $i = n - 1$ . Equating the second derivatives at each interior point gives:

$$6a_{i-1}x_i + 2b_{i-1} = 6a_i x_i + 2b_i \text{ for } i = 2, 3, \dots, n - 1 \quad (6.79)$$

Since there are  $n - 2$  interior points, this condition gives  $n - 2$  additional equations.

Together, the three conditions give  $4n - 6$  equations. However, the  $n - 1$  polynomials have  $4n - 4$  coefficients, and two more equations (conditions) are needed in order to solve for the coefficients. The additional conditions are usually taken to be that the second derivative is zero at the first point and at the last point. This gives two additional equations:

$$6a_1x_1 + 2b_1 = 0 \text{ and } 6a_{n-1}x_n + 2b_{n-1} = 0 \quad (6.80)$$

Cubic splines with the second derivatives at the endpoints set equal to zero are called **natural cubic splines**. Applying all the conditions gives a system of  $4n - 4$  linear equations for the  $4n - 4$  coefficients. The system can be solved using one of the methods from Chapter 4.

### Cubic splines based on Lagrange form polynomials

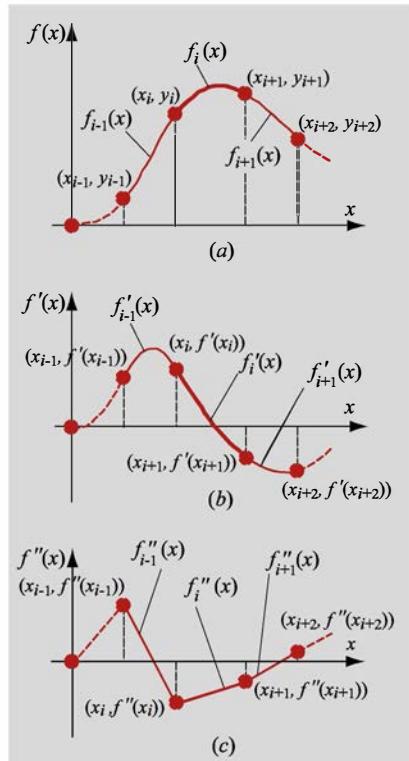
The derivation of cubic splines using the Lagrange form starts with the second derivative of the polynomial. Figure 6-21 shows spline interpolation with cubic polynomials in (a), the first derivatives of the polynomials in (b), and their second derivatives in (c). The figure shows an  $i$ th interval with the adjacent  $i - 1$  and  $i + 1$  intervals. The second derivative of a third-order polynomial is a linear function. This means that within each spline the second derivative is a linear function of  $x$  (see Fig. 6-21c). For the  $i$ th interval, this linear function can be written in the Lagrange form:

$$f_i''(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} f_i''(x_i) + \frac{x - x_i}{x_{i+1} - x_i} f_i''(x_{i+1}) \quad (6.81)$$

where the values of the second derivative of the third-ordered polynomial at the endpoints (knots) of the  $i$ th interval are  $f_i''(x_i)$  and  $f_i''(x_{i+1})$ . The third order polynomial in interval  $i$  can be determined by integrating Eq. (6.81) twice. The resulting expression contains two constants of integration. These two constants can be determined from the condition that the values of the polynomial at the knots are known:

$$f_i(x_i) = y_i \quad \text{and} \quad f_i(x_{i+1}) = y_{i+1}$$

Once the constants of integration are determined, the equation of the third-order polynomial in interval  $i$  is given by:



**Figure 6-21: Third-order polynomial (a) and its first (b) and second (c) derivatives.**

$$\begin{aligned}
 f_i(x) &= \frac{f_i''(x_i)}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1}-x_i)}(x-x_i)^3 \\
 &\quad + \left[ \frac{y_i}{x_{i+1}-x_i} - \frac{f_i''(x_i)(x_{i+1}-x_i)}{6} \right] (x_{i+1}-x) \\
 &\quad + \left[ \frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_i''(x_{i+1})(x_{i+1}-x_i)}{6} \right] (x-x_i)
 \end{aligned} \tag{6.82}$$

for  $x_i \leq x \leq x_{i+1}$  and  $i = 1, 2, \dots, n-1$

For each interval Eq. (6.82) contains two unknowns,  $f_i''(x_i)$  and  $f_i''(x_{i+1})$ . These are the values of the second derivative at the endpoints of the interval. Equations that relate the values of the second derivatives at the  $n-2$  interior points can be derived by requiring continuity of the first derivatives of polynomials from adjacent intervals at the interior knots:

$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \quad \text{for } i = 1, 2, \dots, n-2 \tag{6.83}$$

This condition is applied by using Eq. (6.82) to write the expressions for  $f_i(x)$  and  $f_{i+1}(x)$ , differentiating the expressions, and substituting the derivatives in Eq. (6.83). These operations give (after some algebra) the following equations:

$$\begin{aligned}
 (x_{i+1}-x_i)f''(x_i) + 2(x_{i+2}-x_i)f''(x_{i+1}) + (x_{i+2}-x_{i+1})f''(x_{i+2}) \\
 = 6 \left[ \frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} - \frac{y_{i+1}-y_i}{x_{i+1}-x_i} \right]
 \end{aligned} \tag{6.84}$$

for  $i = 1, 2, \dots, n-2$

This is a system of  $n-2$  linear equations that contains  $n$  unknowns.

### **How is the polynomial in each interval determined?**

- For  $n$  given data points there are  $n-1$  intervals. The cubic polynomial in each interval is given by Eq. (6.82), (total of  $n-1$  polynomials).
- The  $n-1$  polynomials contain  $n$  coefficients  $f''(x_1)$ , through  $f''(x_n)$ . These are the values of the second derivatives of the polynomials at the data points. The second derivative at the interior knots is taken to be continuous. This means that at the interior knots, the second derivatives of the polynomials from adjacent intervals are equal. Consequently, for  $n$  data points there are  $n$  values (the value of the second derivative at each point) that have to be determined.
- Equations (6.84) give a system of  $n-2$  linear equations for the  $n$  unknown coefficients  $f''(x_1)$  through  $f''(x_n)$ . To solve for the coefficients, two additional relations are required. Most commonly, the second derivative at the endpoints of the data (the first and last

points) is set to zero (natural cubic splines):

$$f''(x_1) = 0 \quad \text{and} \quad f''(x_n) = 0 \quad (6.85)$$

With these conditions, the linear system of Eqs. (6.84) can be solved, and the coefficients can be substituted in the equations for the polynomials (Eqs. (6.82)). Cubic splines with the second derivatives at the endpoints set equal to zero are called ***natural cubic splines***.

### Simplified form of the equations

The form of Eqs. (6.82) and (6.84) can be simplified by defining  $h_i$  as the length of the  $i$ th interval (the intervals do not have to be of the same length):

$$h_i = x_{i+1} - x_i \quad (6.86)$$

and defining  $a_i$  as the second derivative of the polynomial at point  $x_i$ :

$$a_i = f''(x_i) \quad (6.87)$$

With these definitions, the equation of the polynomial in the  $i$ th interval is:

$$\begin{aligned} f_i(x) &= \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3 \\ &+ \left[ \frac{y_i}{h_i} - \frac{a_i h_i}{6} \right] (x_{i+1}-x) + \left[ \frac{y_{i+1}}{h_i} - \frac{a_{i+1} h_i}{6} \right] (x-x_i) \end{aligned} \quad (6.88)$$

for  $x_i \leq x \leq x_{i+1}$       and       $i = 1, 2, \dots, n-1$

and the system of linear equations that has to be solved for the  $a_i$ 's is given by:

$$h_i a_i + 2(h_i + h_{i+1}) a_{i+1} + h_{i+1} a_{i+2} = 6 \left[ \frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i} \right] \quad (6.89)$$

for  $i = 1, 2, \dots, n-2$

To carry out cubic spline interpolation, Eq. (6.89) is used for writing a system of  $n-2$  equations with  $n-2$  unknowns,  $a_2$  through  $a_{n-1}$ . (Remember that with natural cubic splines  $a_1$  and  $a_n$  are equal to zero.) Equations (6.89) result in a tridiagonal system of equations that can be solved efficiently using the Thomas algorithm (see Chapter 4). Once the system of equations is solved, the cubic polynomials for each interval can be written using Eq. (6.88). Example 6-8 shows a solution of the problem that is solved in Example 6-7 with cubic splines.

### Note on using cubic splines in MATLAB

Cubic splines are available as a built-in function within MATLAB. However, the option labeled 'cubic' (also called 'pchip') is **not** the method of cubic splines. Rather, the option labeled 'spline' is the appropriate option to use for cubic splines. Even when using the option 'spline', the user is cautioned that it is not the natural splines

described in this chapter. The cubic splines available in MATLAB under the option 'spline' use the not-a-knot conditions at the endpoints, that is, at the first and last data points. The not-a-knot condition refers to the fact that the third derivatives are continuous at the second point and at the second to last point.

### Example 6-8: Cubic splines.

The set of the following five data points is given:

$x$	8	11	15	18	22
$y$	5	9	10	8	7

- (a) Determine the natural cubic splines that fit the data.
- (b) Determine the interpolated value of  $y$  for  $x = 12.7$ .
- (c) Plot of the data points and the interpolating polynomials.

#### SOLUTION

(a) There are five points ( $n = 5$ ), and thus four splines ( $i = 1, \dots, 4$ ). The cubic equation in the  $i$ th spline is:

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1} - x)^3 + \frac{a_{i+1}}{6h_i}(x - x_i)^3 + \left[ \frac{y_i}{h_i} - \frac{a_i h_i}{6} \right](x_{i+1} - x) + \left[ \frac{y_{i+1}}{h_i} - \frac{a_{i+1} h_i}{6} \right](x - x_i) \quad \text{for } i=1, \dots, 4$$

where  $h_i = x_{i+1} - x_i$ . The four equations contain five unknown coefficients  $a_1, a_2, a_3, a_4$ , and  $a_5$ . For natural cubic splines the coefficients  $a_1$  and  $a_5$  are set to be equal to zero. The other three coefficients are determined from a linear system of three equations given by Eq. (6.89).

The values of the  $h_i$ s are:  $h_1 = x_2 - x_1 = 11 - 8 = 3$ ,  $h_2 = x_3 - x_2 = 15 - 11 = 4$

$$h_3 = x_4 - x_3 = 18 - 15 = 3, \quad h_4 = x_5 - x_4 = 22 - 18 = 4$$

$$\begin{aligned} i = 1 \quad h_1 a_1 + 2(h_1 + h_2)a_2 + h_2 a_3 &= 6 \left[ \frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1} \right] \\ 3 \cdot 0 + 2(3 + 4)a_2 + 4a_3 &= 6 \left[ \frac{10 - 9}{4} - \frac{9 - 5}{3} \right] \rightarrow 14a_2 + 4a_3 = -6.5 \end{aligned}$$

$$\begin{aligned} i = 2 \quad h_2 a_2 + 2(h_2 + h_3)a_3 + h_3 a_4 &= 6 \left[ \frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2} \right] \\ 4a_2 + 2(4 + 3)a_3 + 3a_4 &= 6 \left[ \frac{8 - 10}{3} - \frac{10 - 9}{4} \right] \rightarrow 4a_2 + 14a_3 + 3a_4 = -5.5 \end{aligned}$$

$$\begin{aligned} i = 3 \quad h_3 a_3 + 2(h_3 + h_4)a_4 + h_4 a_5 &= 6 \left[ \frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3} \right] \\ 3a_3 + 2(3 + 4)a_4 + 4 \cdot 0 &= 6 \left[ \frac{5 - 8}{4} - \frac{8 - 10}{3} \right] \rightarrow 3a_3 + 14a_4 = 2.5 \end{aligned}$$

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

$$\begin{bmatrix} 14 & 4 & 0 \\ 4 & 14 & 3 \\ 0 & 3 & 14 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -6.5 \\ -5.5 \\ 2.5 \end{bmatrix} \quad (6.90)$$

The system in Eq. (6.90) is solved with MATLAB:

```
>> A = [14 4 0; 4 14 3; 0 3 14];
>> B = [-6.5; -5.5; 2.5];
```

```
>> coefficients = (A\B)'
coefficients =
-0.3665 -0.3421 0.2519
    ↑   ↑   ↑
  a2   a3   a4
```

With the coefficients known, the polynomials are (from Eq. (6.88)):

$$i = 1 \quad f_1(x) = \frac{a_1}{6h_1}(x_2 - x)^3 + \frac{a_2}{6h_1}(x - x_1)^3 + \left[ \frac{y_1}{h_1} - \frac{a_1 h_1}{6} \right](x_2 - x) + \left[ \frac{y_2}{h_1} - \frac{a_2 h_1}{6} \right](x - x_1)$$

$$f_1(x) = \frac{0}{6 \cdot 3}(11 - x)^3 + \frac{-0.3665}{6 \cdot 3}(x - 8)^3 + \left[ \frac{5}{3} - \frac{0 \cdot 3}{6} \right](11 - x) + \left[ \frac{9}{3} - \frac{-0.3665 \cdot 3}{6} \right](x - 8)$$

$$f_1(x) = (-0.02036)(x - 8)^3 + 1.667(11 - x) + 3.183(x - 8) \quad \text{for } 8 \leq x \leq 11$$

$$i = 2 \quad f_2(x) = \frac{a_2}{6h_2}(x_3 - x)^3 + \frac{a_3}{6h_2}(x - x_2)^3 + \left[ \frac{y_2}{h_2} - \frac{a_2 h_2}{6} \right](x_3 - x) + \left[ \frac{y_3}{h_2} - \frac{a_3 h_2}{6} \right](x - x_2)$$

$$f_2(x) = \frac{-0.3665}{6 \cdot 4}(15 - x)^3 + \frac{-0.3421}{6 \cdot 4}(x - 11)^3 + \left[ \frac{9}{4} - \frac{-0.3665 \cdot 4}{6} \right](15 - x) + \left[ \frac{10}{4} - \frac{-0.3421 \cdot 4}{6} \right](x - 11)$$

$$f_2(x) = (-0.01527)(15 - x)^3 + (-0.01427)(x - 11)^3 + 2.494(15 - x) + 2.728(x - 11) \quad \text{for } 11 \leq x \leq 15$$

$$i = 3 \quad f_3(x) = \frac{a_3}{6h_3}(x_4 - x)^3 + \frac{a_4}{6h_3}(x - x_3)^3 + \left[ \frac{y_3}{h_3} - \frac{a_3 h_3}{6} \right](x_4 - x) + \left[ \frac{y_4}{h_3} - \frac{a_4 h_3}{6} \right](x - x_3)$$

$$f_3(x) = \frac{-0.3421}{6 \cdot 3}(18 - x)^3 + \frac{0.2519}{6 \cdot 3}(x - 15)^3 + \left[ \frac{10}{3} - \frac{-0.3421 \cdot 3}{6} \right](18 - x) + \left[ \frac{8}{3} - \frac{0.2519 \cdot 3}{6} \right](x - 15)$$

$$f_3(x) = (-0.019)(18 - x)^3 + 0.014(x - 15)^3 + 3.504(18 - x) + 2.5407(x - 15) \quad \text{for } 15 \leq x \leq 18$$

$$i = 4 \quad f_4(x) = \frac{a_4}{6h_4}(x_5 - x)^3 + \frac{a_5}{6h_4}(x - x_4)^3 + \left[ \frac{y_4}{h_4} - \frac{a_4 h_4}{6} \right](x_5 - x) + \left[ \frac{y_5}{h_4} - \frac{a_5 h_4}{6} \right](x - x_4)$$

$$f_4(x) = \frac{0.2519}{6 \cdot 4}(22 - x)^3 + \frac{0}{6 \cdot 4}(x - 18)^3 + \left[ \frac{8}{4} - \frac{0.2519 \cdot 4}{6} \right](22 - x) + \left[ \frac{7}{4} - \frac{0 \cdot 4}{6} \right](x - 18)$$

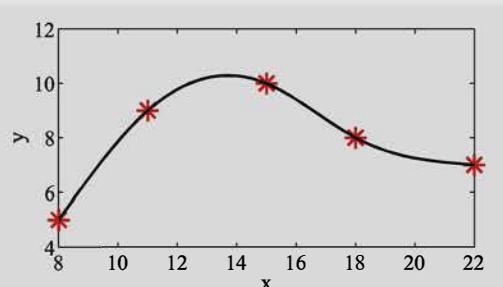
$$f_4(x) = 0.0105(22 - x)^3 + 1.832(22 - x) + 1.75(x - 18) \quad \text{for } 18 \leq x \leq 22$$

(b) The interpolated value of  $y$  for  $x = 12.7$  is calculated by substituting the value of  $x$  in  $f_2(x)$ :

$$f_2(x) = (-0.01527)(15 - 12.7)^3 + (-0.01427)(12.7 - 11)^3 + 2.494(15 - 12.7) + 2.728(12.7 - 11)$$

$$f_2(x) = 10.11$$

(c) The plot on the right shows the data points and the polynomial.

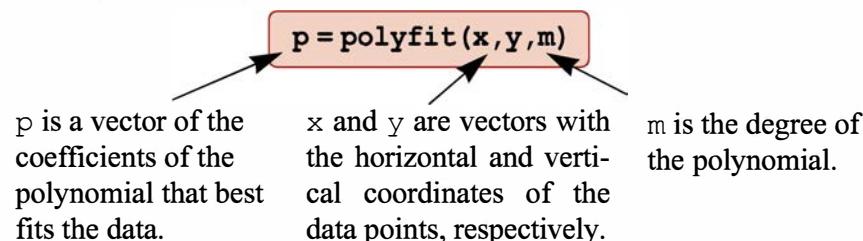


## 6.7 USE OF MATLAB BUILT-IN FUNCTIONS FOR CURVE FITTING AND INTERPOLATION

MATLAB has built-in functions for curve fitting and interpolation. In addition, MATLAB has an interactive tool for curve fitting, called the basic fitting interface. This section describes how to use the functions `polyfit` (for curve fitting) and `interp1` (for interpolation). Polynomials can be easily used and mathematically manipulated with MATLAB.

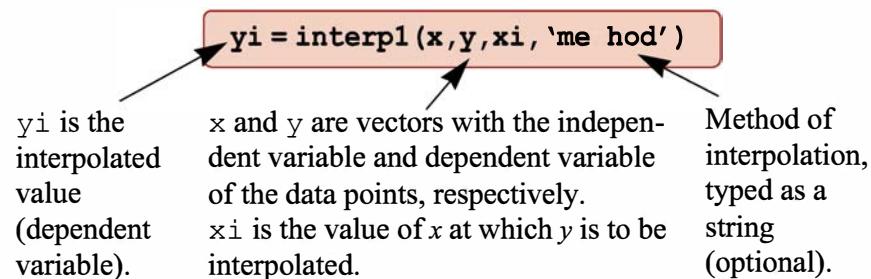
### *The polyfit command*

The `polyfit` command can be used for curve fitting a given set of  $n$  points with polynomials of various degrees and for determining the polynomial of order  $n - 1$  that passes through all the points. The form of the command is:



### *The interp1 command*

The `interp1` (the last character is the number one) command executes one-dimensional interpolation at one point. The format of the command is:



- The vector `x` must be monotonic (the elements must be in ascending or descending order).
- `xi` can be a scalar (interpolation at one point) or a vector (interpolation at several points). `yi` is a scalar or a vector with the corresponding interpolated values at the point(s) `xi`.
- MATLAB can interpolate using one of several methods that can be specified. These methods include:

<code>'nearest'</code> <code>'linear'</code>	returns the value of the data point that is nearest to the interpolated point. uses linear spline interpolation.
---	---

- 'spline' uses cubic spline interpolation with "not-a-knot" conditions where the third derivatives at the second and second to last points are continuous. This is not the natural spline presented in this chapter.
- 'pchip' also called 'cubic', uses piecewise cubic Hermite interpolation.
- When the 'nearest' and the 'linear' methods are used, the value(s) of  $xi$  must be within the domain of  $x$ . If the 'spline' or the 'pchip' methods are used,  $xi$  can have values outside the domain of  $x$  and the function `interp1` performs extrapolation.
  - The 'spline' method can give large errors if the input data points are nonuniform such that some points are much closer together than others.

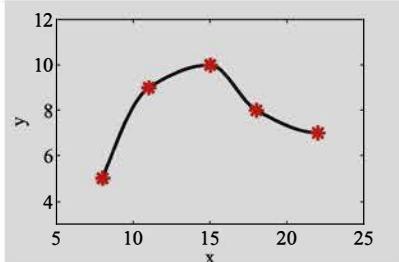
Specification of the method is optional. If no method is specified, the default is 'linear'.

Two examples of using MATLAB's built-in functions for curve fitting and interpolation are shown. First, the `polyfit` function is used for determining the fourth order polynomial that curve-fits the data points in Example 6-3:

```
>> x = 0:0.4:6;
>> 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];
>> p = polyfit(x,y,4)
p =
-0.2644    3.1185 -10.1927   12.8780  -0.2746
```

The polynomial that corresponds to these coefficients is:  
 $f(x)=(-0.2644)x^4+3.1185x^3-10.1927x^2+12.878x-0.2746$ .

In the second example, the `interp1` command is used for the interpolation in Example 6-8:



```
>> x = [8 11 15 18 22];
>> y = [5 9 10 8 7];
>> xint=8:0.1:22;
>> yint=interp1(x,y,xint,'pchip');
>> plot(x,y,'*',xint,yint)
```

Assign the data points to  $x$  and  $y$ .

Vector with points for interpolation.

Calculate the interpolated values.

Create a plot with the data points and interpolated values.

**Figure 6-22: Interpolation using MATLAB's `interp1` function.**

The resulting plot is shown in Fig. 6-22.

MATLAB also has an interactive tool for curve fitting and interpolation, called the basic fitting interface. To activate the interface, the user first has to make a plot of the data points and then in the Figure Window select **Basic Fitting** in the **Tools** menu. (A detailed description of the basic fitting interface is available in MATLAB, An Introduction with Applications, Fourth Edition, by Amos Gilat, Wiley, 2011.)

## 6.8 CURVE FITTING WITH A LINEAR COMBINATION OF NONLINEAR FUNCTIONS

The method of least squares that was applied for curve fitting with a linear function in Section 6.2, and with quadratic and higher-order polynomials in Section 6.4, can be generalized in terms of curve fitting with a linear combination of nonlinear functions. A linear combination of  $m$  nonlinear functions can be written as:

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + C_3 f_3(x) + \dots + C_m f_m(x) = \sum_{j=1}^m C_j f_j(x_i) \quad (6.91)$$

where  $f_1, f_2, \dots, f_m$  are prescribed functions, and  $C_1, C_2, \dots, C_m$  are unknown coefficients. Using least-squares regression, Eq. (6.91) is used to fit a given set of  $n$  points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  by minimizing the total error that is given by the sum of the squares of the residuals:

$$E = \sum_{i=1}^n \left[ y_i - \sum_{j=1}^m C_j f_j(x_i) \right]^2 \quad (6.92)$$

The function  $E$  in Eq. (6.92) has a minimum for those values of the coefficients  $C_1, C_2, \dots, C_m$  where the partial derivative of  $E$  with respect to each of the coefficients is equal to zero:

$$\frac{\partial E}{\partial C_k} = 0 \quad \text{for } k = 1, 2, \dots, m \quad (6.93)$$

Substituting Eq. (6.92) into Eq. (6.93) gives:

$$\frac{\partial E}{\partial C_k} = \sum_{i=1}^n 2 \left[ y_i - \sum_{j=1}^m C_j f_j(x_i) \right] \left[ -\frac{\partial}{\partial C_k} \left( \sum_{j=1}^m C_j f_j(x_i) \right) \right] = 0$$

for  $k = 1, 2, \dots, m$  (6.94)

Since the coefficients  $C_1, C_2, \dots, C_m$  are independent of each other,

$$\frac{\partial}{\partial C_k} \left( \sum_{j=1}^m C_j f_j(x_i) \right) = f_k(x_i) \quad (6.95)$$

and Eq. (6.94) becomes:

$$\frac{\partial E}{\partial C_k} = -\sum_{i=1}^n 2 \left[ y_i - \sum_{j=1}^m C_j f_j(x_i) \right] f_k(x_i) = 0 \quad (6.96)$$

The last equation can be rewritten in the form:

$$\sum_{i=1}^n \sum_{j=1}^m C_j f_j(x_i) f_k(x_i) = \sum_{i=1}^n y_i f_k(x_i) \text{ for } k = 1, 2, \dots, m \quad (6.97)$$

In Eq. (6.97),  $x_i$ ,  $y_i$ , and  $f_k(x_i)$  are all known quantities, and the  $C_1, C_2, \dots, C_m$  are the unknowns. The set of Eqs. (6.97) is a system of  $m$  linear equations for the unknown coefficients  $C_1, C_2, \dots, C_m$ .

The functions  $f_k(x)$  can be any functions. For example, if  $F(x) = C_1 f_1(x) + C_2 f_2(x)$  such that  $f_1(x) = 1$  and  $f_2(x) = x$ , then Eqs. (6.97) reduce to Eqs. (6.9) and (6.10). If the functions  $f_k(x)$  are chosen such that  $F(x)$  is quadratic (i.e.,  $f_1(x) = 1$ ,  $f_2(x) = x$ , and  $f_3(x) = x^2$ ), then Eqs. (6.97) reduce to Eqs. (6.23)–(6.25). In general, the functions  $f_k(x)$  are chosen because there is a guiding theory that predicts the trend of the data. Example 6-9 shows how the method is used for curve fitting data points with nonlinear approximating functions.

### Example 6-9: Curve fitting with linear combination of nonlinear functions.

The following data is obtained from wind-tunnel tests, for the variation of the ratio of the tangential velocity of a vortex to the free stream flow velocity  $y = V_\theta/V_\infty$  versus the ratio of the distance from the vortex core to the chord of an aircraft wing,  $x = R/C$ :

$x$	0.6	0.8	0.85	0.95	1.0	1.1	1.2	1.3	1.45	1.6	1.8
$y$	0.08	0.06	0.07	0.07	0.07	0.06	0.06	0.06	0.05	0.05	0.04

Theory predicts that the relationship between  $x$  and  $y$  should be of the form  $y = \frac{A}{x} + \frac{Be^{-2x^2}}{x}$ . Find the values of  $A$  and  $B$  using the least-squares method to fit the above data.

#### SOLUTION

In the notation of Eq. (6.91) the approximating function is  $F(x) = C_1 f_1(x) + C_2 f_2(x)$  with  $F(x) = y$ ,  $C_1 = A$ ,  $C_2 = B$ ,  $f_1(x) = \frac{1}{x}$ , and  $f_2(x) = \frac{e^{-2x^2}}{x}$ . The equation has two terms, which means that  $m = 2$ , and since there are 11 data points,  $n = 11$ . Substituting this information in Eq. (6.97) gives the following system of two linear equations for  $A$  and  $B$ .

$$\sum_{i=1}^{11} A \frac{1}{x_i} \frac{1}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i} \frac{1}{x_i} = \sum_{i=1}^{11} y_i \frac{1}{x_i} \quad \text{for } k = 1$$

$$\sum_{i=1}^{11} A \frac{1}{x_i} \frac{e^{-2x_i^2}}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i} \frac{e^{-2x_i^2}}{x_i} = \sum_{i=1}^{11} y_i \frac{e^{-2x_i^2}}{x_i} \quad \text{for } k = 2$$

These two equations can be rewritten as:

$$A \sum_{i=1}^{11} \frac{1}{x_i^2} + B \sum_{i=1}^{11} \frac{e^{-2x_i^2}}{x_i^2} = \sum_{i=1}^{11} y_i \frac{1}{x_i}$$

$$A \sum_{i=1}^{11} \frac{e^{-2x_i^2}}{x_i^2} + B \sum_{i=1}^{11} \frac{e^{-4x_i^2}}{x_i^2} = \sum_{i=1}^{11} y_i \frac{e^{-2x_i^2}}{x_i}$$

The system can be written in a matrix form:

$$\begin{bmatrix} \sum_{i=1}^{11} \frac{1}{x_i^2} & \sum_{i=1}^{11} \frac{e^{-2x_i^2}}{x_i^2} \\ \sum_{i=1}^{11} \frac{e^{-2x_i^2}}{x_i^2} & \sum_{i=1}^{11} \frac{e^{-4x_i^2}}{x_i^2} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{11} y_i \frac{1}{x_i} \\ \sum_{i=1}^{11} y_i \frac{e^{-2x_i^2}}{x_i} \end{bmatrix}$$

The system is solved by using MATLAB. The following MATLAB program in a script file solves the system and then makes a plot of the data points and the curve-fitted function.

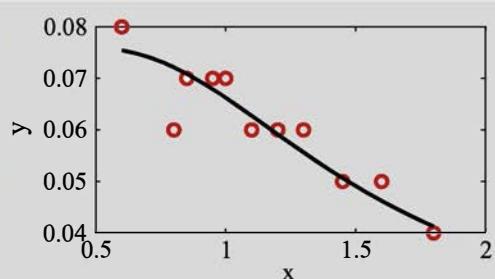
```
x = [0.6 0.8 0.85 0.95 1.0 1.1 1.2 1.3 1.45 1.6 1.8];
y = [0.08 0.06 0.07 0.07 0.07 0.06 0.06 0.06 0.05 0.05 0.04];
a(1,1) = sum(1./x.^2);
a(1,2) = sum(exp(-2*x.^2)./x.^2);
a(2,1) = a(1,2);
a(2,2) = sum(exp(-4*x.^2)./x.^2);
b(1,1) = sum(y./x);
b(2,1) = sum((y.*exp(-2*x.^2))./x);
AB = a\b;
xfit = 0.6:0.02:1.8;
yfit = AB(1)./xfit + AB(2)*exp(-2*xfit.^2)./xfit;
plot(x,y, 'o', xfit, yfit)
```

When the program is executed, the solution for the coefficients is displayed in the Command Window (the two elements of the vector AB), and a plot with the data points and the curve-fitted function is created.

Command Window:

```
AB =
0.0743
-0.0597
```

← The coefficient  $A$ .  
← The coefficient  $B$ .



## 6.9 PROBLEMS

### Problems to be solved by hand

Solve the following problems by hand. When needed, use a calculator, or write a MATLAB script file to carry out calculations. If using MATLAB, do not use built-in functions for curve fitting and interpolation.

**6.1** The following data is given:

<i>x</i>	1	3	4	6	9	12	14
<i>y</i>	2	4	5	6	7	9	11

- (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. (6.5) to determine the overall error.

**6.2** The following data is given:

<i>x</i>	-7	-4	-1	0	2	5	7
<i>y</i>	20	14	5	3	-2	-10	-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. (6.5) to determine the overall error.

**6.3** The following data give the approximate population of China for selected years from 1900 until 2010:

<i>Year</i>	1900	1950	1970	1980	1990	2000	2010
<i>Population (millions)</i>	400	557	825	981	1135	1266	1370

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 6.3), 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 1985.

**6.4** The following data is given:

<i>x</i>	0.2	0.5	1	2	3
<i>y</i>	3	2	1.4	1	0.6

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 6.3), and use linear least-squares regression to determine the value of the coefficients.

**6.5** The following data is given:

$x$	-2	-1	0	1	2
$y$	1.5	3.2	4.5	3.4	2

Determine the coefficients  $a$  and  $b$  in the function  $y = \frac{a}{x^2 + b}$  that best fit the data. (Write the function in a linear form (Section 6.3), and use linear least-squares regression to determine the value of the coefficients.) Once the coefficients are determined make a plot that shows the function and the data points.

**6.6** The following data is given:

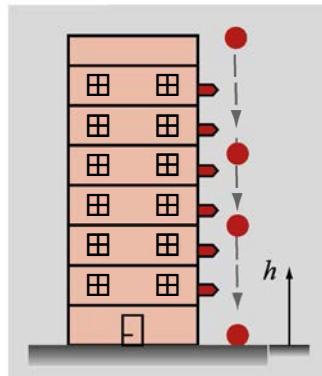
$x$	1	2	3	5	8
$y$	0.8	1.9	2.2	3	3.5

Determine the coefficients  $m$  and  $b$  in the function  $y = [m\sqrt{x} + b]^{1/2}$  that best fit the data. Write the equation in a linear form (Section 6.3), and use linear least-squares regression to determine the value of the coefficients.

**6.7** To measure  $g$  (the acceleration due to gravity), the following experiment is carried out. A ball is dropped from the top of a 100-m-tall building. As the object is falling down, the time  $t$  when it passes sensors mounted on the building wall is recorded. The data measured in the experiment is given in the table.

$h$ (m)	100	80	60	40	20	0
$t$ (s)	0	2.02	2.86	3.50	4.04	4.51

In terms of the coordinates shown in the figure, the position of the ball  $h$  as a function of the time  $t$  is given by  $h = h_0 - \frac{1}{2}gt^2$ , where  $h_0 = 100$  m is the initial position of the ball. Use linear regression to best fit the equation to the data and determine the experimental value of  $g$ .



**6.8** Water solubility in jet fuel,  $W_S$ , as a function of temperature,  $T$ , can be modeled by an exponential function of the form  $W_S = be^{mT}$ . The following are values of water solubility measured at different temperatures. Using linear regression, determine the constants  $m$  and  $b$  that best fit the data. Use the equation to estimate the water solubility at a temperature of 10° C. Make a plot that shows the function and the data points.

$T$ (°C)	-40	-20	0	20	40
$W_S$ (% wt.)	0.0012	0.002	0.0032	0.006	0.0118

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

$I$ (nA)	300	300	350	400	400	500	500	650	650
$d$ ( $\mu\text{m}$ )	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.

- 6.10** Determine the coefficients of the polynomial  $y = a_2x^2 + a_1x + a_0$  that best fit the data given in Problem 6.5.

- 6.11** Using the method in Section 6.8, determine the coefficients of the equation  $y = ax + b/x^2$  that best fit the following data:

$x$	0.8	1.6	2.4	3.2	4.0
$y$	6	3.6	4.1	5.1	6.2

- 6.12** Using the method in Section 6.8, determine the coefficients of the equation  $y = Ae^{x/2} + B\sqrt{x} + Cx^2$  that best fit the following data:

$x$	0.4	1.0	1.6	2.2	2.8
$y$	5.1	7.1	8	8.1	7.8

- 6.13** 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.

- 6.14** Determine the fourth-order Newton's interpolating polynomial that passes through the data points given in Problem 6.13. Use the polynomial to calculate the power at a wind speed of 26 mph.

- 6.15** The following data is given:

$x$	1	2.2	3.4	4.8	6	7
$y$	2	2.8	3	3.2	4	5

- (a) Write the polynomial in Lagrange form that passes through the points; then use it to calculate the interpolated value of  $y$  at  $x = 5.4$ .
- (b) Write the polynomial in Newton's form that passes through the points; then use it to calculate the interpolated value of  $y$  at  $x = 5.4$ .

**6.16** Use linear splines interpolation with the data in Problem 6.13 to calculate the power at the following wind speeds.

- (a) 24 mph (b) 35 mph.

**6.17** Use quadratic splines interpolation with the data in Problem 6.13 to calculate the power at the following wind speeds.

- (a) 24 mph (b) 35 mph.

**6.18** Use natural cubic splines interpolation (based on Lagrange-form polynomials [Eqs. (6.86)–(6.89)]) with the data in Problem 6.13; to calculate the power at the following wind speeds.

- (a) 24 mph (b) 35 mph.

### **Problems to be programmed in MATLAB**

*Solve the following problems using MATLAB environment. Do not use MATLAB's built-in functions for curve fitting and interpolation.*

**6.19** Modify the MATLAB user-defined function `LinearRegression` in Program 6-1. In addition to determining the constants  $a_1$  and  $a_0$ , the modified function should also calculate the overall error  $E$  according to Eq. (6.6). Name the function  $[a, Er] = \text{LinReg}(x, y)$ . 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 6-1.
- (b) Use the function to solve Problem 6.2.

**6.20** 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  $[b m] = \text{ExpoFit}(x, y)$ , 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 6.3 for determining the value of the coefficients. Use the function to solve Problem 6.8.

**6.21** 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  $[b m] = \text{PowerFit}(x, y)$ , 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 6.3 for determining the value of the coefficients. Use the function to solve Problem 6.3.

**6.22** 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  $a = \text{QuadFit}(x, y)$ , 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 6-2.
- (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.

**6.23** 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. (6.21). Name the function  $[a, Er] = \text{CubicPolyFit}(x, y)$ , 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.

- Use `CubicPolyFit` to determine the cubic polynomial that best fits the data in Example 6-3.
- Write a program in a script file that plots the data points and the curve of the cubic polynomial that best fits the data.

**6.24** Write a MATLAB user-defined function for interpolation with natural cubic splines. Name the function  $Yint = \text{CubicSplines}(x, y, xint)$ , 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.

- Use the function with the data in Example 6-8 for calculating the interpolated value at  $x = 12.7$ .
- Use the function with the data in Problem 6.39 for calculating the enthalpy per unit mass at  $T = 14000$  K and at  $T = 24000$  K.

**6.25** Write a MATLAB user-defined function for spline interpolation that uses third-order Lagrange polynomials. Name the function  $Yint = \text{CubicLagSplines}(x, y, xint)$ , 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  $x_{i+2}$ .

- Use the `CubicLagSplines` function with the data in Problem 6.13 to calculate the power at wind speeds of 26 mph and 42 mph.
- Use the `CubicLagSplines` function with the data in Example 6-3 to calculate the stress at strains of 0.2 and 3.

**6.26** A linear combination of three functions that is used for curve fitting has the form (see Section 6.8):

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

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  $C = \text{NonLinCombFit}(F1, F2, F3, x, y)$ , 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 6.40 (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.

### Problems in math, science, and engineering

Solve the following problems using the MATLAB environment. As stated, use the MATLAB programs that are presented in the chapter, programs developed in previously solved problems, or MATLAB's built-in functions.

- 6.27** 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  $\alpha$  is the temperature coefficient of resistance. Determine  $R_0$  and  $\alpha$  such that the equation will best fit the following data. Use  $T_0 = 20^\circ\text{C}$ .

$T (\text{ }^\circ\text{C})$	20	100	180	260	340	420	500
$R (\Omega)$	500	676	870	1060	1205	1410	1565

- (a) Use the user-defined function `LinReg` developed in Problem 6.19.  
 (b) Use MATLAB's built-in function `polyfit`.

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

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

- (a) Use the user-defined function `LinReg` developed in Problem 6.19.  
 (b) Use MATLAB's built-in function `polyfit`.

- 6.29** The amount of water in air measured at various temperatures at 100% humidity is displayed in the following table:

$T (\text{ }^\circ\text{C})$	0	10	20	30	40	50
$m_{\text{water}}$ (g/Kg of air)	5	8	15	28	51	94

- (a) Determines the coefficients of an exponential function of the form  $m_{\text{water}} = be^{mT}$  that best fits the data. Use the function to estimate the amount of water at  $35^\circ\text{C}$ . If available, use the user-defined function `ExpoFit` developed in Problem 6.20. In one figure, plot the exponential function and the data points.  
 (b) Use MATLAB's built-in function `polyfit` to determine the coefficients of a second-order polynomial that best fits the data. Use the polynomial to estimate the amount of water at  $35^\circ\text{C}$ . In one figure, plot the polynomial and the data points.

**6.30** 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,  $\sigma_t$ , and the true strain,  $\varepsilon_t$ , are defined by:

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

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 6.3 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  $L_0 = 0.0125 \text{ m}$ .

$F$ (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

**6.31** The percent of households that own at least one computer in selected years from 1981 to 2010, according to the U.S. census bureau, is listed in the following table:

Year	1981	1984	1989	1993	1997	2000	2001	2003	2004	2010
Household with computer [%]	0.5	8.2	15	22.9	36.6	51	56.3	61.8	65	76.7

The data can be modeled with a function in the form  $H_C = C/(1 + Ae^{-Bx})$  (logistic equation), where  $H_C$  is percent of households that own at least one computer,  $C$  is a maximum value for  $H_C$ ,  $A$  and  $B$  are constants, and  $x$  is the number of years after 1981. By using the method described in Section 6.3 and assuming that  $C = 90$ , determine the constants  $A$  and  $B$  such that the function best fit the data. Use the function to estimate the percent of ownership in 2008 and in 2013. In one figure, plot the function and the data points.

**6.32** Use MATLAB's built-in functions to determine the coefficients of the third-order polynomial,  $H_C = a_3x^3 + a_2x^2 + a_1x + a_0$  (where  $x$  is the number of years after 1981) that best fits the data in Problem 6.31. Use the polynomial to estimate the percent of computer ownership in 2008 and in 2013. In one figure, plot the polynomial and the data points.

**6.33** 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:

$v$ (mi/h)	12.5	25	37.5	50	62.5	75
$d$ (ft)	20	59	118	197	299	420

Determine the coefficients of a quadratic polynomial  $d = a_2v^2 + a_1v + a_0$  that best fits 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 6.22.
- (b) Use MATLAB's built-in function polyfit.

**6.34** Measurements of thermal conductivity,  $k$  (W/m K), of silicon at various temperatures,  $T$  (K), are:

$T$ (°K)	50	100	150	200	400	600	800	1000
$k$ (W/m K)	28	9.1	4.0	2.7	1.1	0.6	0.4	0.3

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 6-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

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

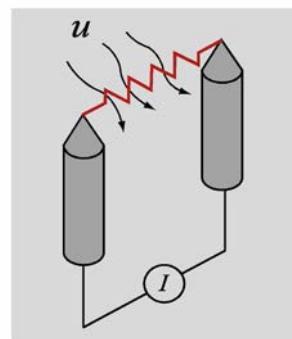
$$\frac{1}{T + 273.15} = C_1 + C_2 \ln(R) + C_3 \ln^3(R)$$

where  $T$  is the temperature in degrees Celsius,  $R$  is the thermistor resistance in  $\Omega$ , 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
$R$ ( $\Omega$ )	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.

**6.36** 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 6.20.
- (b) Use MATLAB built-in functions.

In each part make a plot that shows the data points (asterisk marker) and the equation (solid line).

**6.37** 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	2.40	3.00	2.55	2.24	1.72	1.18	0.82	0.56	0.42	0.25

**6.38** The yield stress of many metals,  $\sigma_y$ , varies with the size of the grains. Often, the relationship between the grain size,  $d$ , and the yield stress is modeled with the Hall–Pitch 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)	0.0018	0.0025	0.004	0.007	0.016	0.060	0.25
$\sigma_y$ (MPa)	530	450	380	300	230	155	115

- (a) Determine the constants  $\sigma_0$  and  $k$  such that the Hall–Pitch equation will best fit the data. Plot the data points (circle markers) and the Hall–Pitch equation as a solid line. Use the Hall–Pitch equation to estimate the yield stress of a specimen with a grain size of 0.003 mm.
- (b) Use the user-defined function QuadFit from Problem 6.22 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 0.003 mm.

**6.39** Values of enthalpy per unit mass,  $h$ , of an equilibrium Argon plasma ( $\text{Ar}$ ,  $\text{Ar}^+$ ,  $\text{A}^{++}$ ,  $\text{A}^{+++}$  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 6.24.
- (b) For interpolation use MATLAB's built-in function `interp1` with the spline option.

**6.40** The following are measurements of the rate coefficient,  $k$ , for the reaction  $\text{CH}_4 + \text{O} \rightarrow \text{CH}_3 + \text{OH}$  at different temperatures,  $T$ :

$T$ (K)	595	623	761	849	989	1076	1146	1202	1382	1445	1562
$k \times 10^{20}$ (m <sup>3</sup> /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 6.8).

- (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<sup>3</sup>/s) and  $E_a$  (J/mole) in the Arrhenius expression.

**6.41** 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 6-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

**6.42** The following data present ocean water salinity at different depths:

Depth (m)	0	100	200	300	400	500	600	700	800	900	1100	1400	2000	3000
Salinity (ppt)	35.5	35.35	35.06	34.65	34.48	34.39	34.34	34.32	34.33	34.36	34.45	34.58	34.73	34.79

- (a) Use interpolation to estimate the water salinity at depths of 250 m, 750 m, and 1800 m by using the user-defined function `CubicSplines` developed in Problem 6.24.
- (b) Use interpolation to estimate the water salinity at depths of 250 m, 750 m, and 1800 m by using the user-defined function `CubicLagSplines` developed in Problem 6.25.
- (c) Use MATLAB to create a vector with values of depth ranging from 0 to 3000 m with spacing of 10 m. Then use the MATLAB's built-defined function `interp1` with the option `spline` to calculate corresponding interpolated values of salinity. Make a plot that shows the data points and interpolated points.

**6.43** The following data present the power of a diesel engine at different engine speeds:

Engine Speed (rpm)	1200	1500	2000	2500	3000	3250	3500	3750	4000	4400
Engine Power (hp)	65	130	185	225	255	266	275	272	260	230

- (a) Estimate the engine power at speeds of 2300 rpm and 3650 rpm by using the user-defined function `CubicSplines` developed in Problem 6.24.
- (b) Estimate the engine power at speeds of 2300 rpm and 3650 rpm by using the user-defined function `CubicLagSplines` developed in Problem 6.25.
- (c) Use MATLAB to create a vector with values of engine speeds ranging from 1200 to 4400 rpm with spacing of 10 rpm. Use the MATLAB's built-defined function `interp1` with the option `spline` to calculate corresponding interpolated values of engine power. Make a plot that shows the data points and interpolated points.