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, σ_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
σ_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

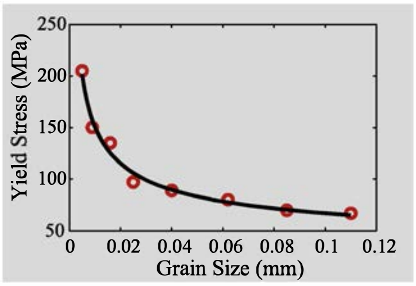


Figure 6-1: Curve fitting.

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

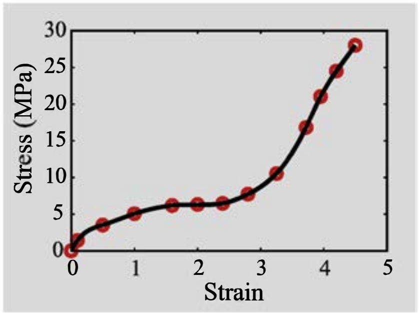


Figure 6-2: Interpolation.

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 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

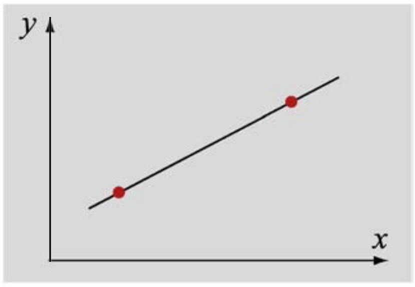


Figure 6-3: Two data points.

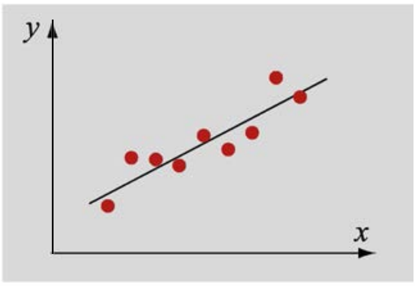


Figure 6-4: Many data points.

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

$$y = a_1 x + a_0 \tag{6.1}$$

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:

$$r_i = y_i - f(x_i) \tag{6.2}$$

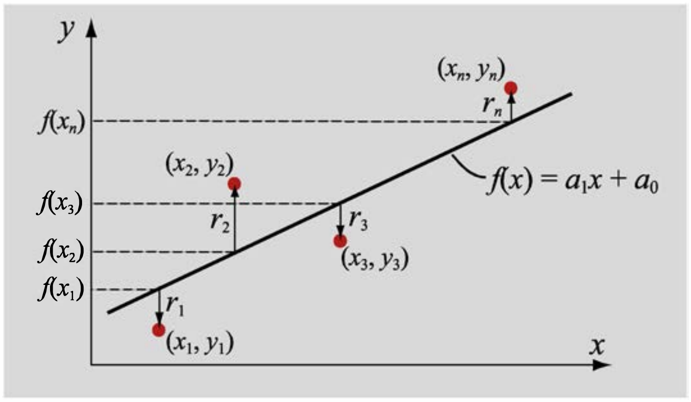


Figure 6-5: Curve-fitting points with a linear equation.

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:

$$E = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} [y_i - (a_1x_i + a_0)] \tag{6.3}$$

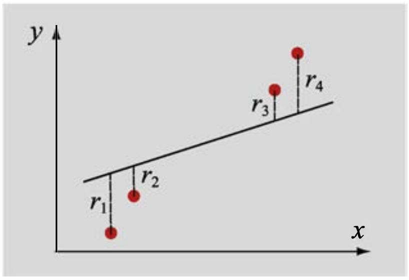


Figure 6-6: Fit with no error according to Eq. (6.3).

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:

$$E = \sum_{i=1}^{n} |r_i| = \sum_{i=1}^{n} |y_i - (a_1x_i + a_0)| \tag{6.4}$$

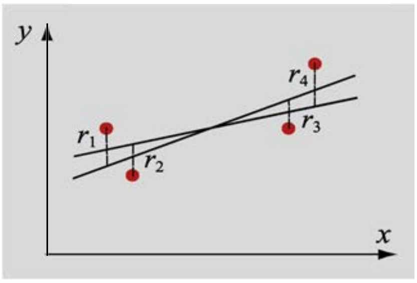


Figure 6-7: Two fits with the same error according to Eq. (6.4).

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:

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} [y_i - (a_1x_i + a_0)]^2 \tag{6.5}$$

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:

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

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:

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

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

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:

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

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

The solution of the system is:

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

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

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:

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

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

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

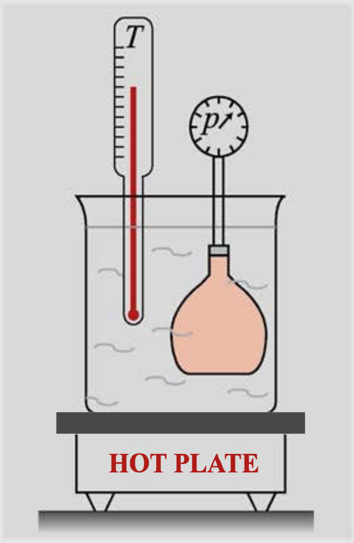
Equations (6.14) give the values of a_1 and a_0 in the equation $y = a_1 x + a_0$ that has the best fit to n data points (x_i, y_i). Example 6-1 shows how to use Eqs. (6.11) and (6.12) for fitting a linear equation to a set of data points.

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 ($T = 0$°C). The temperature of the gas is then increased in ten increments up to $T = 100$°C by heating the water, and the pressure of the gas is measured at each temperature. The data from the experiment is:

T (°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 (°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:

(*a*) Make a plot of the data (p versus T).

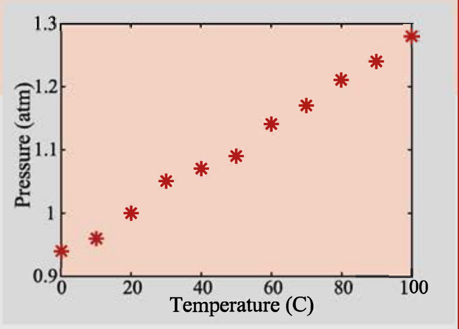
(*b*) Use linear least-squares regression to determine a linear function in the form $p = a_1 T + a_0$ that best fits the data points. First calculate the coefficients by hand using only the four data points: 0, 30, 70, and 100°C. Then write a user-defined MATLAB function that calculates the coefficients 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), (30, 1.05), (70, 1.17), (100, 1.28)

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

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

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

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

Substituting the Ss 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).

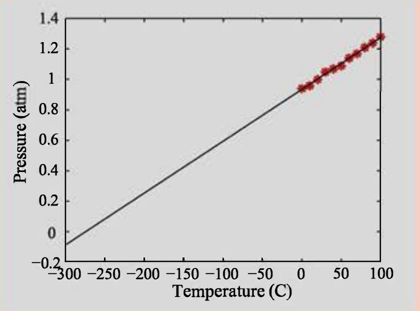
The user-defined function `LinearRegression` is next used in Command Window for determining the best fit line to the given points in the problem.

```
>> 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 °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.

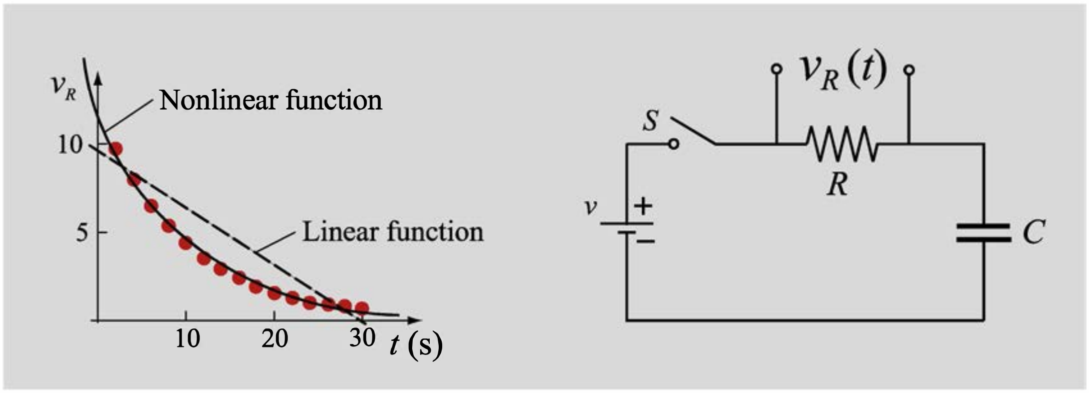


Figure 6-8: Curve-fitting points with linear equation.

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:

$y = bx^m$	(power function)
$y = be^{mx}$ or $y = b10^{mx}$	(exponential function)
$y = \dfrac{1}{mx+b}$	(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:

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

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 $a_0 = \ln(b)$:

$$\underbrace{\ln(y)}_{Y} = \underbrace{m}_{a_1}\underbrace{\ln(x)}_{X} + \underbrace{\ln(b)}_{a_0}$$

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:

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

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
$y = bx^m$	$\ln(y) = m\ln(x) + \ln(b)$	$Y = \ln(y)$, $X = \ln(x)$ $a_1 = m$, $a_0 = \ln(b)$	$\ln(x_i)$ 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.
$y = be^{mx}$	$\ln(y) = mx + \ln(b)$	$Y = \ln(y)$, $X = x$ $a_1 = m$, $a_0 = \ln(b)$	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.
$y = b10^{mx}$	$\log(y) = mx + \log(b)$	$Y = \log(y)$, $X = x$ $a_1 = m$, $a_0 = \log(b)$	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.
$y = \frac{1}{mx + b}$	$\frac{1}{y} = mx + b$	$Y = \frac{1}{y}$, $X = x$ $a_1 = m$, $a_0 = b$	x_i and $1/y_i$	$1/y$ vs. x plot on linear x and y axes.
$y = \frac{mx}{b + x}$	$\frac{1}{y} = \frac{b}{m}\frac{1}{x} + \frac{1}{m}$	$Y = \frac{1}{y}$, $X = \frac{1}{x}$ $a_1 = \frac{b}{m}$, $a_0 = \frac{1}{m}$	$1/x_i$ 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')
```

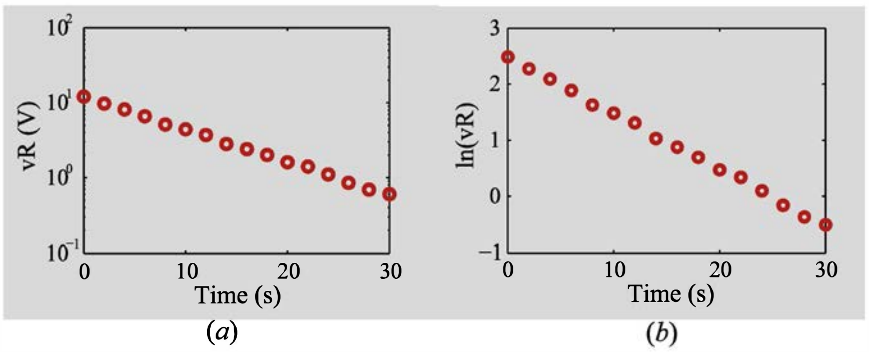


Figure 6-9: (*a*) A plot of v_R vs. t in a plot with a logarithmic vertical axis and linear horizontal axis. (*b*) A plot of $\ln(v_R)$ vs. t in a plot with linear vertical and horizontal axes.

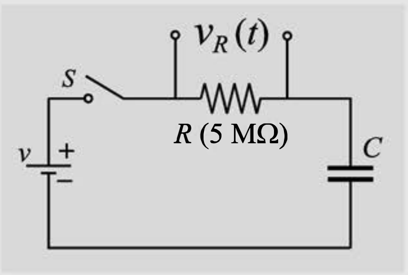
firm that the data from the capacitor charging experiment can be curve fit with an exponential function. The actual curve fitting is shown in Example 6-2.

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-MΩ 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:

$$v_R = ve^{(-t/(RC))} \tag{6.17}$$

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:

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

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;
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);
R=5E6;
[a1,a0]=LinearRegression(texp, vexpLOG)
b=exp(a0)
C=-1/(R*a1)
t=0:0.5:30;
v=b*exp(a1*t);
plot(t,v,texp,vexp,'ro')
```

Enter the experimental data.

Calculate $\ln(y_i)$ of the data points (to be used in the linear regression.

Calculate coefficients a_1 and a_0 with the user-defined function `LinearRegression` in Example 6-1.

Calculate b, since $a_0 = \ln(b)$ (see Table 6-2).

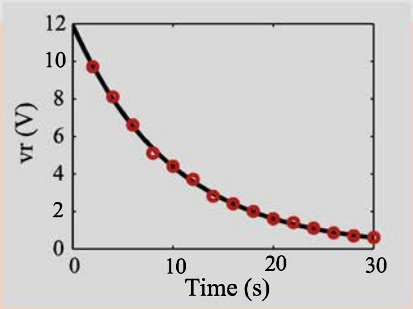
Calculate C using Eq. (6.18).

`a1` is *m* in the equation $v = be^{mt}$.

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 μF.



6.4 CURVE FITTING WITH QUADRATIC AND HIGHER-ORDER POLYNOMIALS

Background

Polynomials are functions that have the form:

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

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

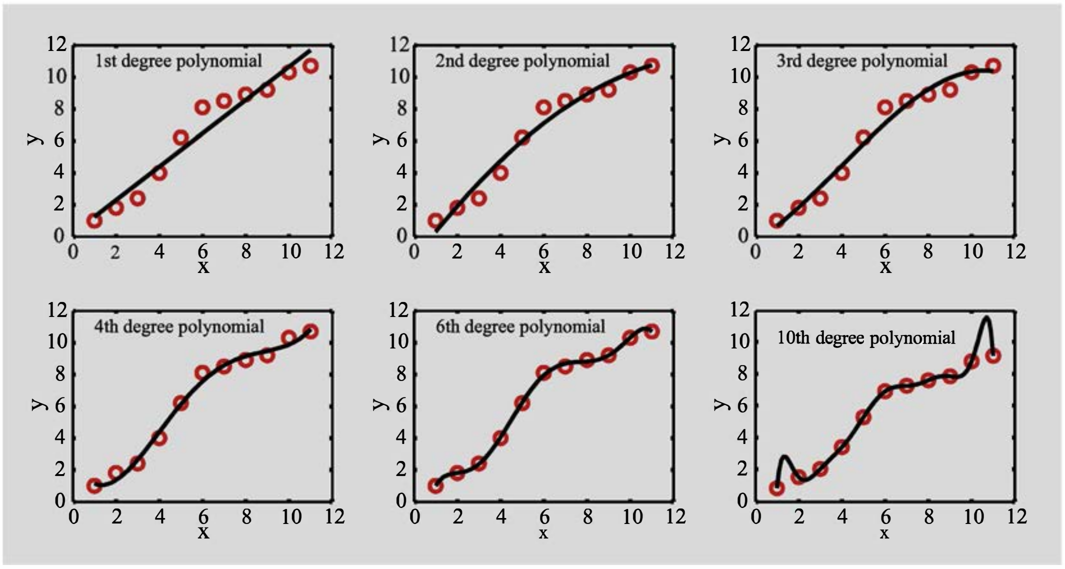


Figure 6-10: Curve fitting of the same set of data points with polynomials for different degrees.

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:

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

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:

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

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:

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

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

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

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

$$\frac{\partial E}{\partial a_2} = -2\sum_{i=1}^{n}(y_i - a_2x_i^2 - a_1x_i - a_0)x_i^2 = 0 \tag{6.25}$$

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:

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

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

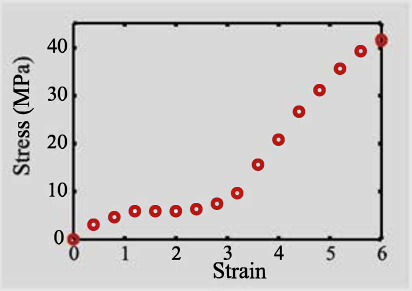
$$\left(\sum_{i=1}^{n}x_i^2\right)a_0 + \left(\sum_{i=1}^{n}x_i^3\right)a_1 + \left(\sum_{i=1}^{n}x_i^4\right)a_2 = \sum_{i=1}^{n}x_i^2y_i \tag{6.28}$$

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 $y = a_2x_i^2 + a_1x_i + a_0$ 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 mth 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 ε	0	0.4	0.8	1.2	1.6	2.0	2.4		
Stress σ (MPa)	0	3.0	4.5	5.8	5.9	5.8	6.2		
Strain ε	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0
Stress σ (MPa)	7.4	9.6	15.6	20.7	26.7	31.1	35.6	39.3	41.5

SOLUTION

A polynomial of the fourth order can be written as:

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \tag{6.29}$$

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).

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

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

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

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

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

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
```

Create rows 2 through 5 of the matrix $[a]$ and elements 2 through 5 of the column vector $[b]$.

```
% 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);
end
```

Create a new vector for the coefficients of the polynomial, to be used in 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.

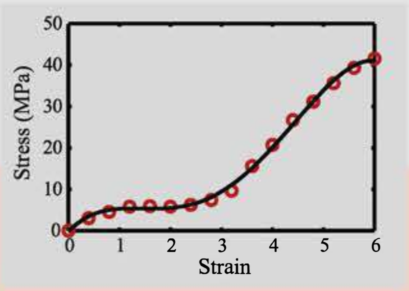
```
stressfit = polyval(Pcoef,epsilon);
```

Stress calculated by the polynomial.

```
plot(x,y,'ro',epsilon,stressfit,'k','linewidth',2)
xlabel('Strain','fontsize',20)
ylabel('Stress (MPa)','fontsize',20)
```

Plot the data points and the curve-fitting polynomial.

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.

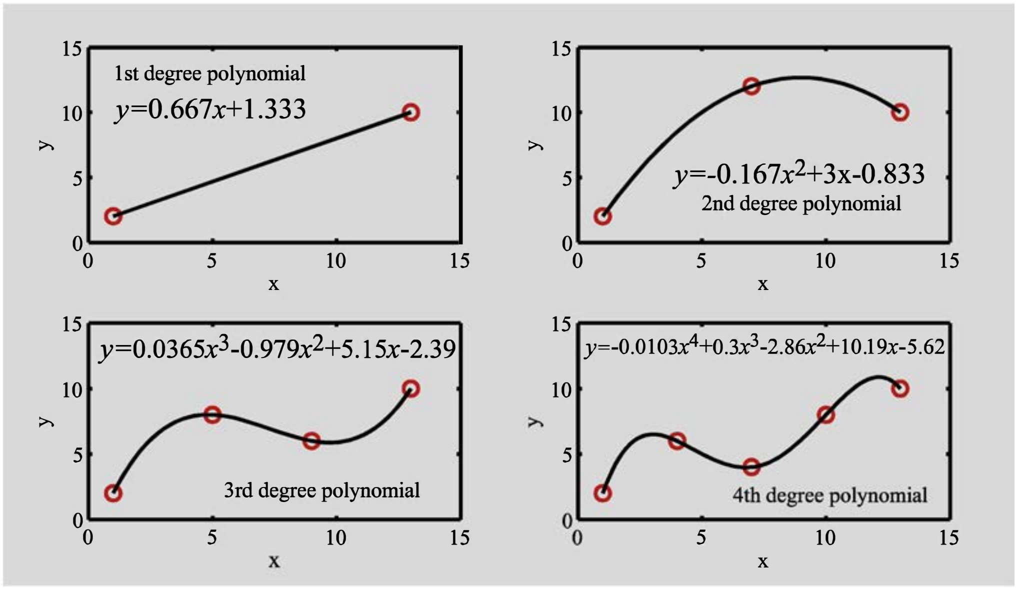


Figure 6-11: Various order polynomials.

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 mth-order polynomial is:

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

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:

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

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
```

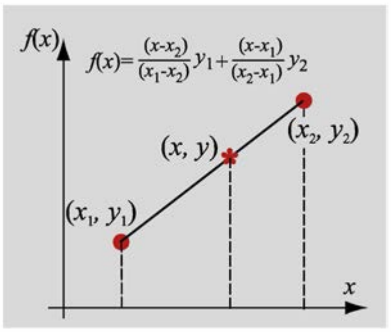
The polynomial that corresponds to these coefficients is: $y = -0.0103x^4 + 0.3x^3 - 2.86x^2 + 10.19x - 5.62$ (see Fig. 6-11).

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.



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

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:

$$f(x) = y = a_1(x-x_2)+a_2(x-x_1) \tag{6.37}$$

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

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

and

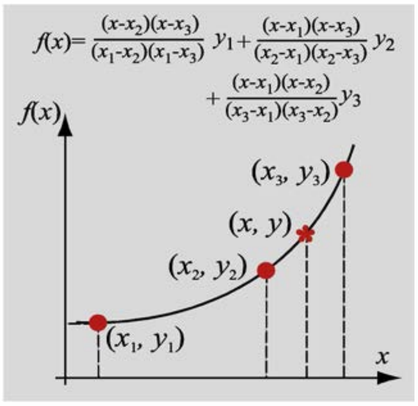
$$y_2 = a_1(x_2-x_2)+a_2(x_2-x_1) \quad \text{or} \quad a_2 = \frac{y_2}{(x_2-x_1)} \tag{6.39}$$

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

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

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 $f(x) = a_1x+a_0$:

$$f(x) = \frac{(y_2-y_1)}{(x_2-x_1)}x+\frac{x_2y_1-x_1y_2}{(x_2-x_1)} \tag{6.41}$$



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

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:

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

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

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

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), …, (x_n, y_n) is:

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

On the right-hand side of Eq. (6.44) the numerator of the ith 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:

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

where $L_i(x) = \prod_{\substack{j=1 \\ j \ne i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$ 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`.

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

- 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:

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

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:

$$f(x) = a_1 + a_2(x - x_1) \tag{6.47}$$

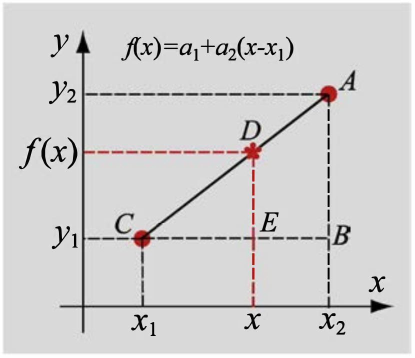


Figure 6-14: First-order Newton's polynomial.

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.

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

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

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

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:

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

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:

$$f(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) \tag{6.51}$$

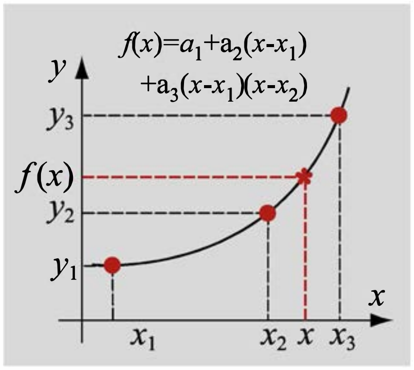


Figure 6-15: Second-order Newton's polynomial.

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:

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

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:

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

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

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

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:

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

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:

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

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:

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

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 $(x_3 - x_1)$:

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

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 $(x_4 - x_1)$:

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

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

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

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

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 nth 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, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the first divided differences between two points (x_i, y_i), and (x_j, y_j) are given by:

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

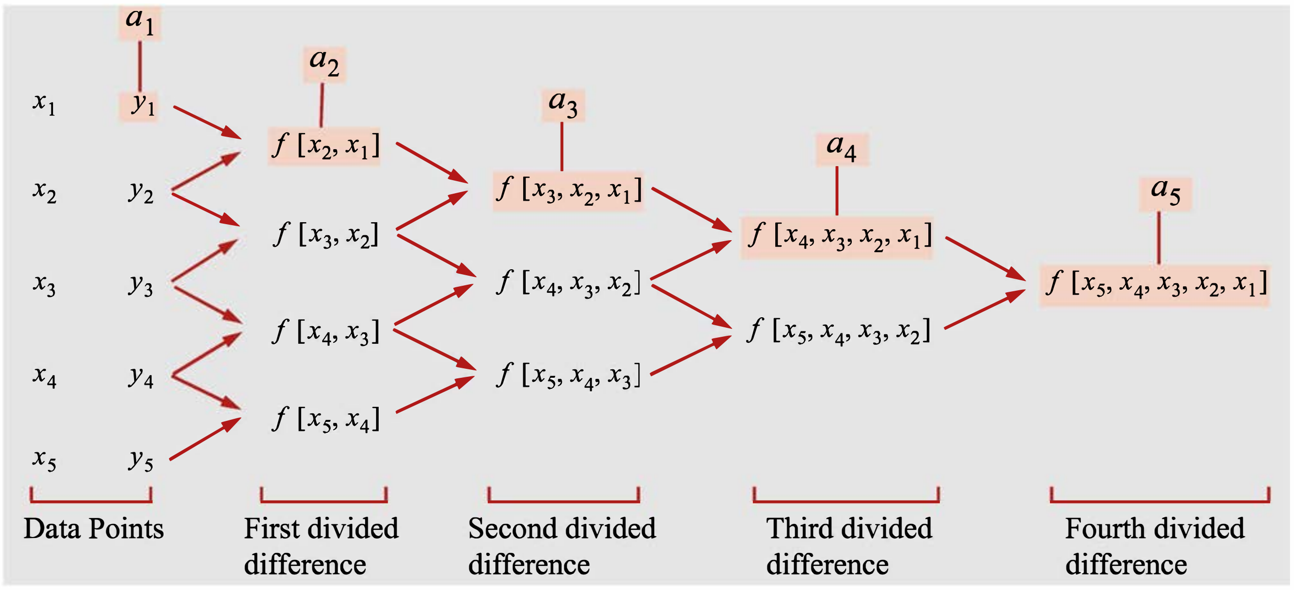


Figure 6-16: Table of divided differences for five data points.

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

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

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

$$f(x) = y = \underbrace{y_1}_{a_1} + \underbrace{f[x_2, x_1]}_{a_2}(x-x_1) + \underbrace{f[x_3, x_2, x_1]}_{a_3}(x-x_1)(x-x_2) + \ldots + \underbrace{f[x_n, x_{n-1}, \ldots, x_2, x_1]}_{a_n}(x-x_1)(x-x_2)\ldots(x-x_{n-1}) \qquad (6.63)$$

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

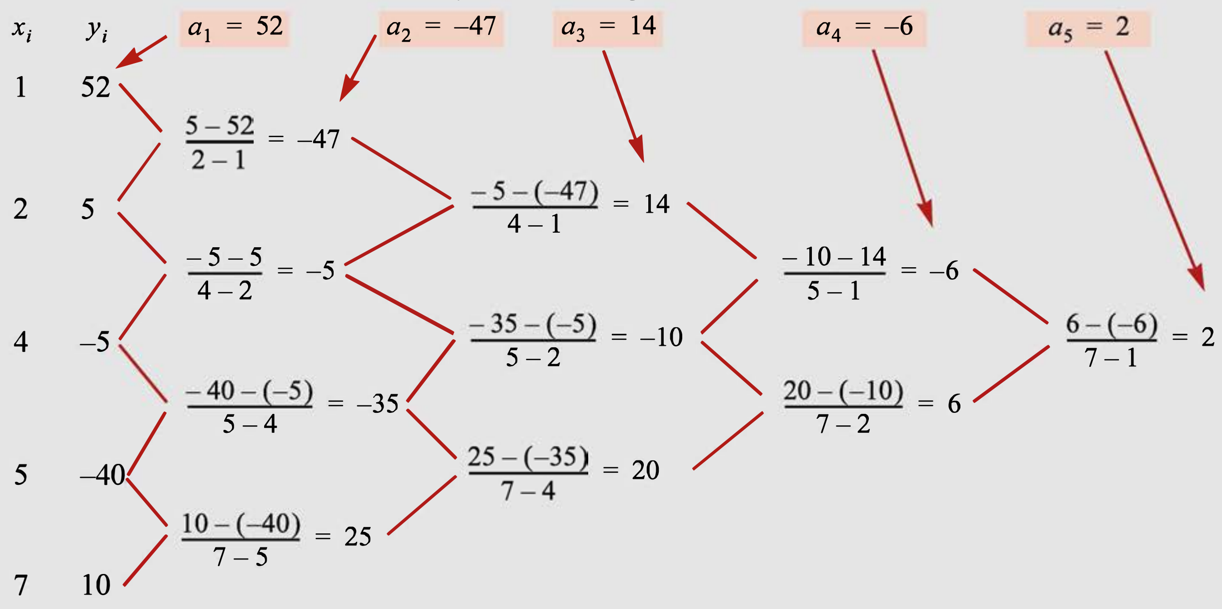
(*a*) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
(*b*) Use the polynomial obtained in part (*a*) to determine the interpolated value for $x = 3$.
(*c*) Write a MATLAB user-defined function that 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
4 –5
5 –40
7 10
$\frac{5-52}{2-1} = -47$
$\frac{-5-5}{4-2} = -5$
$\frac{-40-(-5)}{5-4} = -35$
$\frac{10-(-40)}{7-5} = 25$
$\frac{-5-(-47)}{4-1} = 14$
$\frac{-35-(-5)}{5-2} = -10$
$\frac{25-(-35)}{7-4} = 20$
$\frac{-10-14}{5-1} = -6$
$\frac{20-(-10)}{7-2} = 6$
$\frac{6-(-6)}{7-1} = 2$

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`.