Chapter 6

Curve Fitting and Interpolation

Core Topics

(i) Curve fitting with a linear equation (6.2)

(ii) Curve fitting with a non-linear equation by writing the equation in linear form (6.3)

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

(iv) Interpolation using a single polynomial (6.5)

(v) Lagrange polynomials (6.5.1)

(vi) Newton's polynomials (6.5.2)

(vii) Piecewise (spline) interpolation (6.6)

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

(ix) Curve fitting with a linear combination of nonlinear functions (6.8)

Background:

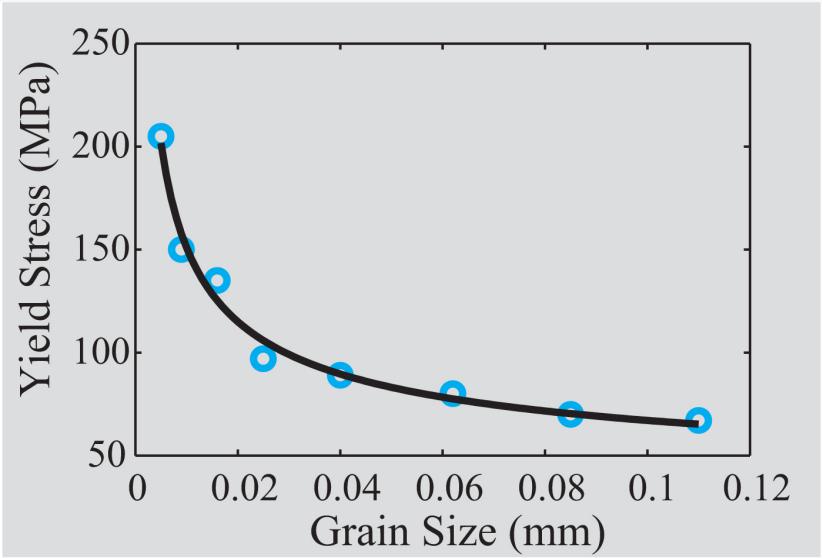
Experimental data is usually discrete having been performed for a discrete set of data points.

Oftentimes we need to approximate these data with a function that can be used to extrapolate data (give data outside the range of measurement) or indeed to estimate the behaviour of the function for a change in function parameters. This process is known as curve fitting. In other words curve fitting is obtaining a mathematical function that governs the results.
By way of example we have the data for the strength of a metal versus grain size:

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

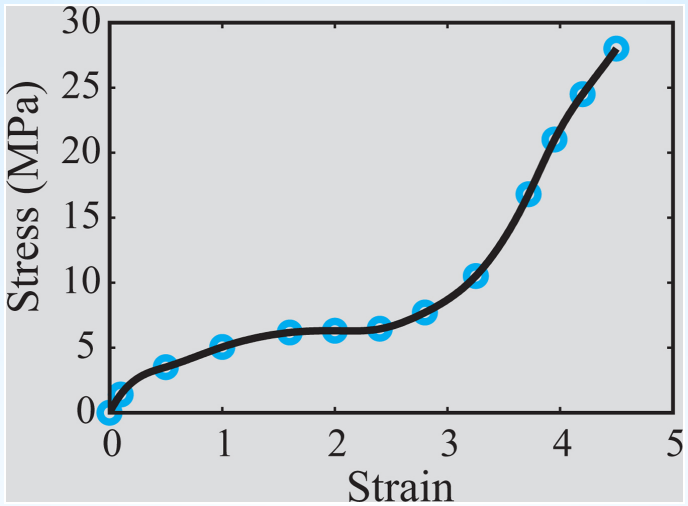
Background:

A plot of the data points and a curve fit is given here:



Note how the chosen curve does not pass through most points exactly but it does give us a good picture of the overall behaviour of the data. Knowing the function we can extrapolate the data and also see how it would behave for a change in parameters.

Background:



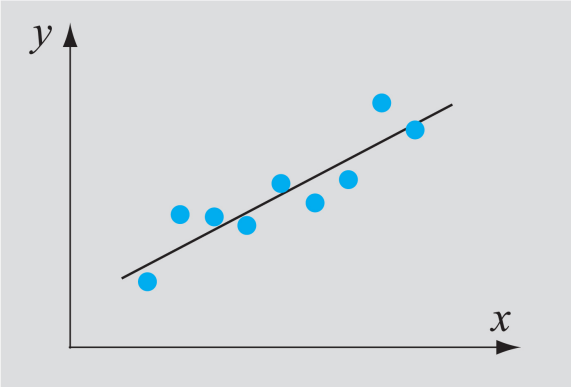
Oftentimes we need to estimate the value of the function between data points. This is best done by finding a function or functions that fit the data exactly. This process is known as interpolation. Interpolation is used to find intermediate values. Note that interpolation can also be done on the basis of curve fitting but the answer will not be as accurate.

Curve Fitting with a Linear Equation:

The simplest curve fit is a line. This is done when it is expected that the experimental variables are proportional to each other. The equation of a line (first degree polynomial) is:

$$y = a_1 x + a_0 \qquad (6.1)$$

We now need to determine the constants a_1 and a_0 but first we need a definition of what is meant by best fit since a number of different lines might appear to fit the data well.



Curve Fitting with a Linear Equation:

To achieve this objective we define the residual thus:

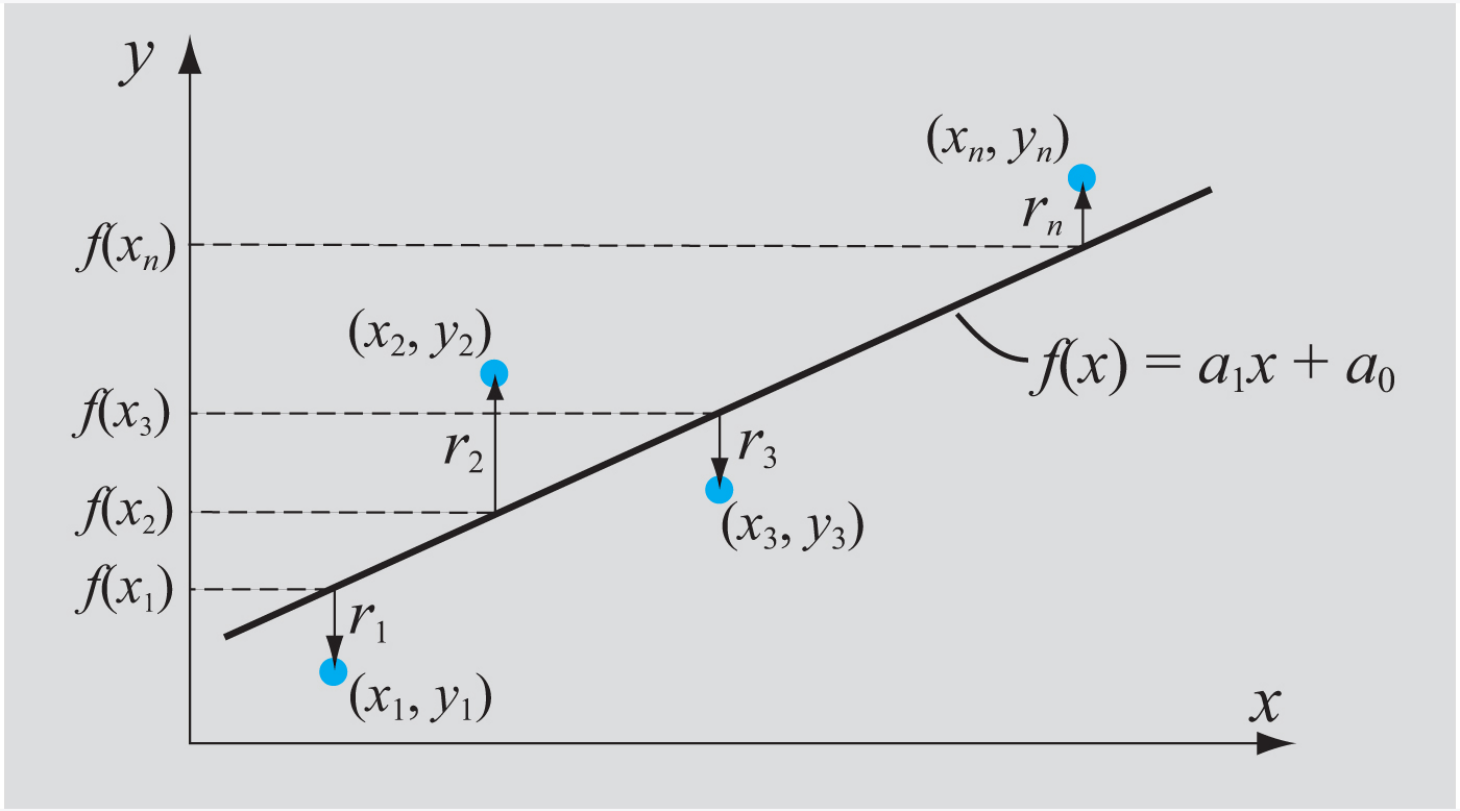
$$r_i = y_i - f(x_i) \qquad (6.2)$$

which is simply the difference between the value y_i of the data point and the value of the function $f(x_i)$ used to approximate it.

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

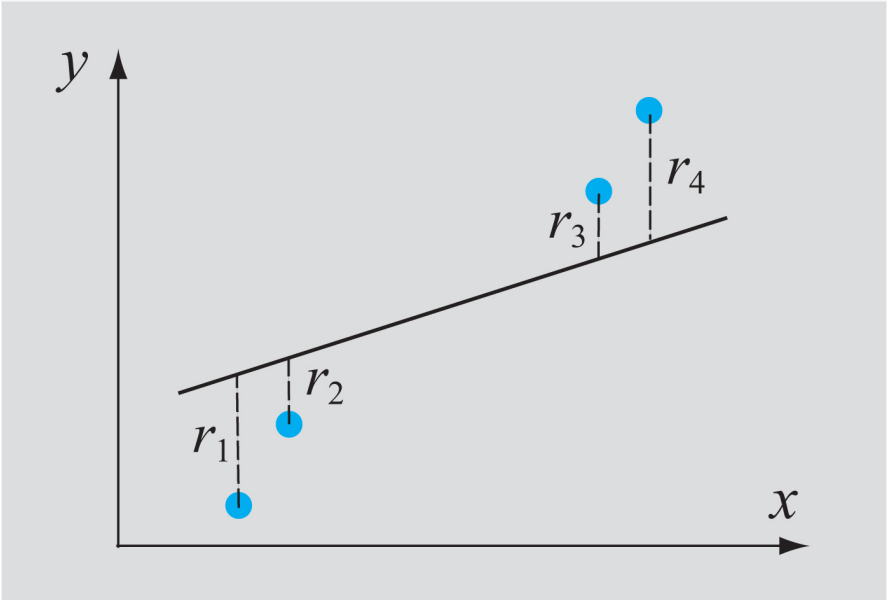
$$E = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)] \qquad (6.3)$$

Curve Fitting with a Linear Equation:



Curve Fitting with a Linear Equation:

The total error does not however give a good measure of the overall fit. This is because a bad fit with positive and negative residuals (both of which could be large) may yield a small overall error because positive and negative residuals may cancel. This scenario is illustrated in the following figure where it is clear that a line with a higher slope would better fit the data.



Curve Fitting with a Linear Equation:

A way to circumvent this problem is to make the overall error equal to the sum of the absolute values of the residuals:

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

This can be a useful approach but it cannot be used to obtain the values of a_0 and a_1. This is because the measure is not unique meaning that for the same set of data points there can be several functions that give the same overall error.

Curve Fitting with a Linear Equation:

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

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

With this approach the overall error is a positive number (positive and negative residuals do not cancel each other). Also larger errors have a relatively larger effect on the total error.

We now use this approach to determine the unique line that best fits the data (i.e. gives the smallest overall error).

Curve Fitting with a Linear Equation:

Linear Least Squares Regression:

This is a procedure that allows one to determine the line of best fit for a given set of data points. From Equation (6.5) we have the following expression for the overall error:

$$E = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]^2 \qquad (6.6)$$

All the values x_i and y_i are known (by experiment) and so Equation (6.6) is a non-linear function of the two variables a_0 and a_1. We now find the values of a_0 and a_1 for which E is a minimum. This is done by taking the partial derivatives of E w.r.t. a_0 and a_1 and setting them to zero – we know that there is no maximum for E!

Curve Fitting with a Linear Equation:

$$\frac{\partial E}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_1 x_i - a_0) = 0 \qquad (6.7)$$

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

This yields two simultaneous equations in a_0 and a_1:

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

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

Curve Fitting with a Linear Equation:

Solving these simultaneous equations for a_0 and a_1 yields:

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

and

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

Curve Fitting with a Linear Equation:

Since the summations are the same in Equations (6.11) and (6.12) it is convenient to perform these summations first and then substitute the results into the equations:

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

Then:

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

Then $y = a_1 x + a_0$ is the line of best fit for the given data points in a least squares sense.

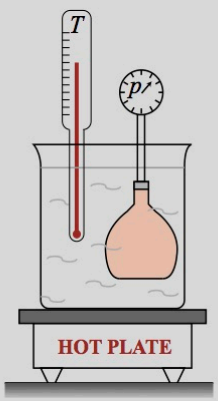
Example:

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.

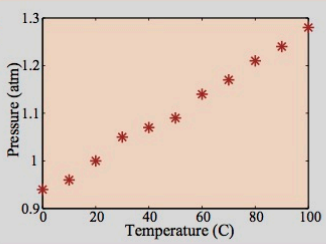
(*c*) Plot the function, and extend the line (extrapolate) until it crosses the horizontal (T) axis. This point is an estimate of the absolute zero temperature, T_0. Determine the value of T_0 from the function.

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:
```

Example:

```
% 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).

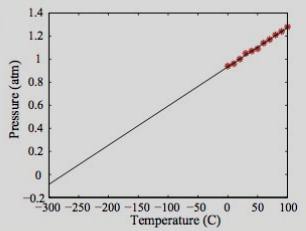
The user-defined function `LinearRegression` is next used in Command Window for determining the best fit line to the given points in the problem.

```
>> 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
```



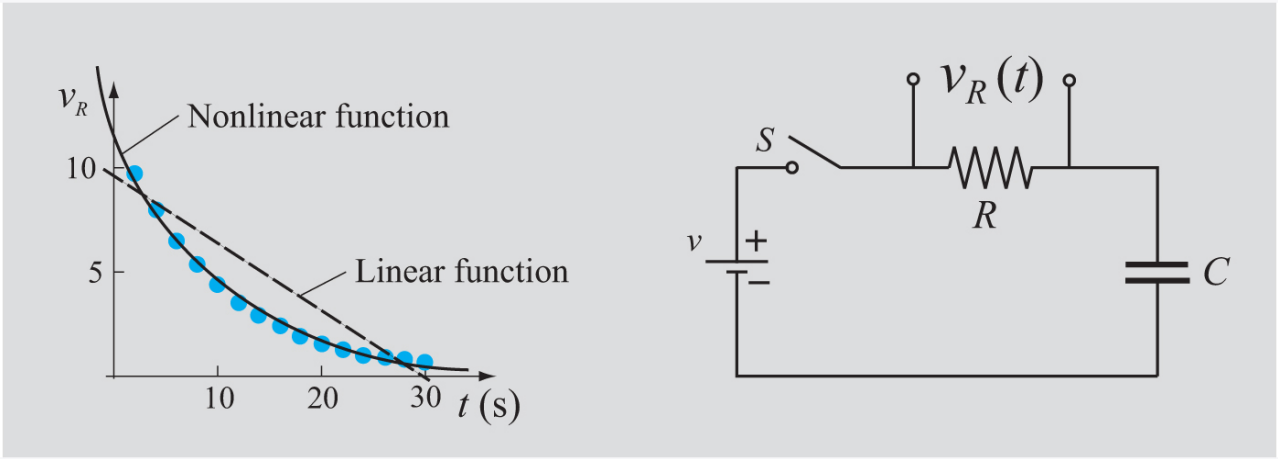
When this script file is executed, the figure shown on the right is displayed, and the value of the calculated absolute zero temperate is displayed in the Command Window, as shown below.

```
T0 =
 -274.5882
```

This result is close to the handbook value of –273.15 °C.

Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

Many times the relationship between quantities in science and engineering is nonlinear. For example consider the case of an RC circuit that is switched in at $t = 0$.



Electrical current flows and the capacitor charges up (causing the voltage across it to increase) and the capacitor voltage then levels off approaching the supply voltage. The electrical current decreases to zero.

Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

At $t = 0$ the maximum electrical current flows (because the capacitor is uncharged) and consequently the voltage across the resistor is at its maximum ($V = IR$). The voltage across the resistor at $t = 0$ equals the supply voltage and it decreases to zero as the capacitor voltage increases until it reaches the supply voltage. After that there is no change in current or voltage in the circuit unless the switch is opened.

It is clear from the plot that a nonlinear function will give a better fit than a linear function.

Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

To write non-linear equations in linear form we take for example the power function:

$$y = bx^m$$

Then:

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

This equation is linear for $\ln(y)$ in terms of $\ln(x)$. Using (linear) least-squares regression we can determine that:

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

Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

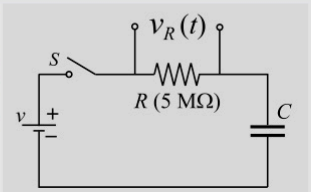
Examples of nonlinear functions that can be used with least-squares regression to give a best fit:

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.
$y = \frac{mx}{b + x}$	$\frac{1}{y} = \frac{b}{m}\frac{1}{x} + \frac{1}{m}$	$Y = \frac{1}{y},\quad X = \frac{1}{x}$ $a_1 = \frac{b}{m},\quad a_0 = \frac{1}{m}$	$1/x_i$ and $1/y_i$	$1/y$ vs. $1/x$ plot on linear x and y axes.

Example:

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):

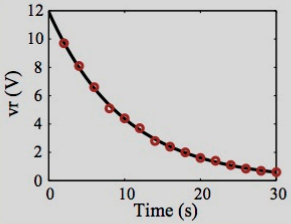
Program 6-2: Script file. Curve fitting with a nonlinear function.

```
texp=2:2:30;                                  % 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 function LinearRegression in Example 6-1.
b=exp(a0)                                     % Calculate b, since a_0 = ln(b) (see Table 6-2).
C=-1/(R*a1)                                   % Calculate C using Eq. (6.18).
t=0:0.5:30;
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
```

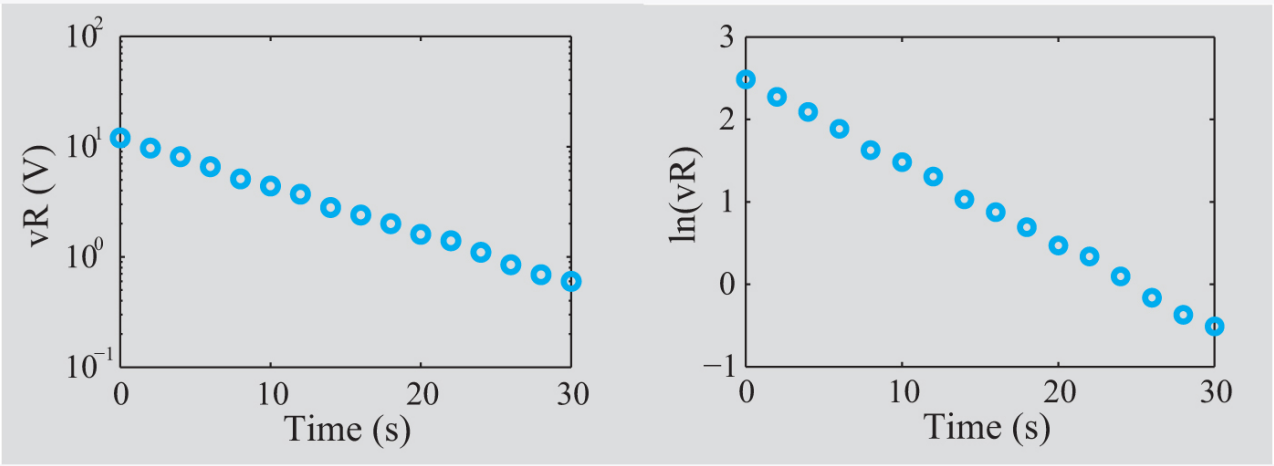
The capacitance is approximately 2 μF.



Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

Choosing the appropriate nonlinear function to fit the data is more often than not guided by an *a priori* knowledge of what it is likely to be a best fit. This stems from a knowledge of the underlying theory that governs the problem.

In the example under consideration if we plot v_R versus time t on a plot with a logarithmic vertical axis (for v_R) and a linear horizontal axis (for t) we should get (roughly) a straight line. Alternatively we could plot $\ln(v_R)$ vs t with linear axes and also get a straight line. Now we know that our choice of nonlinear equation is good.



Curve Fitting with a Nonlinear Equation by writing the Equation in Linear Form:

Other considerations when choosing a nonlinear function are:

1. Exponential functions cannot pass through the origin.

2. Exponential functions can only fit data with all +ve or all -ve ys.

3. Logarithmic functions cannot include $x = 0$ or x -ve.

4. For the power function $y = 0$ when $x = 0$.

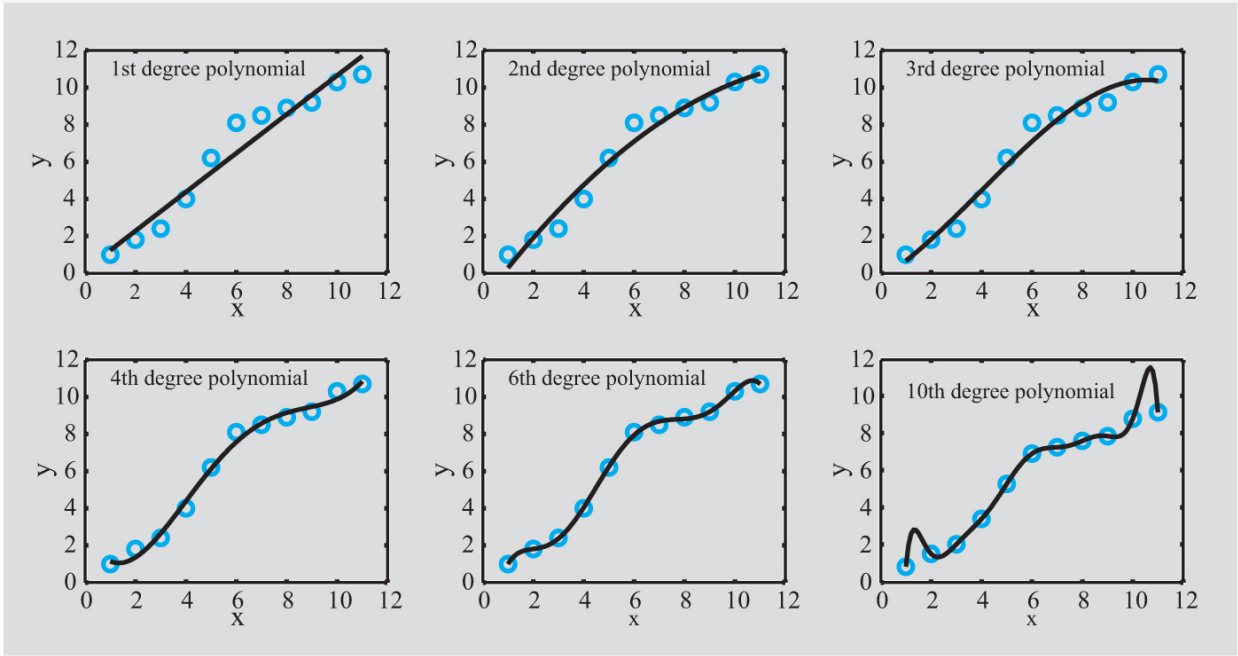
5. The reciprocal equation cannot include $y = 0$.

Curve Fitting with Quadratic and Higher Order Polynomials:

Polynomials are functions of the form:

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

For a given set of n data points can be curve fit with polynomials up to the order of $n - 1$.



Curve Fitting with Quadratic and Higher Order Polynomials:

The higher the order of the polynomial chosen the more data points it will pass through.

It is very important to note however that higher order polynomials may pass through more points but may deviate significantly from the overall trend between points thus making interpolation and extrapolation inaccurate.

Curve Fitting with Quadratic and Higher Order Polynomials:

Polynomial Regression:

This is a procedure for determining the coefficients of a polynomial of a second degree of higher that best fits the data points. The procedure is based on the same approach as was used with linear regression (finding a polynomial of order 1 (a line!) to fit the data – that is to minimise the overall error in a least squares sense.
If the polynomial that is used for curve fitting is:

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

Then, for a given set of n data points (x_i, y_i) (where $m < n - 1$) the total error is:

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

Curve Fitting with Quadratic and Higher Order Polynomials:

Taking for example $m = 2$ (a quadratic polynomial) we have:

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

For a minimum error the partial derivatives must be zero:

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

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

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

Curve Fitting with Quadratic and Higher Order Polynomials:

Equations (6.23) to (6.25) are a system of three linear equations in terms of the unknowns a_0, a_1 and a_2 and can be written thus:

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

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

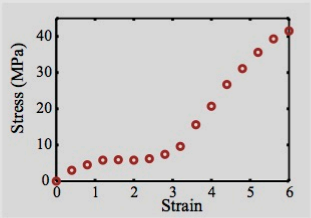
$$\left(\sum_{i=1}^{n} x_i^2\right) a_0 + \left(\sum_{i=1}^{n} x_i^3\right) a_1 + \left(\sum_{i=1}^{n} x_i^4\right) a_2 = \sum_{i=1}^{n} x_i^2 y_i \qquad (6.28)$$

The solution of this system of equations 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 is derived in the same way.

Example:

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 $[b]$.

Example:

```
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)'
for i = 1:m + 1
    Pcoef(i)= p(m + 2 - i);
end
epsilon = 0:0.1:6;
stressfit = polyval(Pcoef,epsilon);
plot(x,y,'ro',epsilon,stressfit,'k','linewidth',2)
xlabel('Strain','fontsize',20)
ylabel('Stress (MPa)','fontsize',20)
```

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

Solve the system $[a][p] = [b]$ for $[p]$. Transpose the solution such that $[p]$ is a row vector.

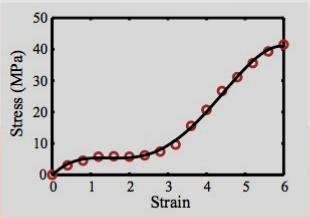
Create a new vector for the coefficients of the polynomial, to be used in MATLAB's `polyval` built-in function (see note at the end of the example).

Define a vector of strain to be used for plotting the polynomial.

Stress calculated by the polynomial.

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.

Curve Fitting with a Linear Combination of Nonlinear Functions:

To generalise least squares regression for any type (or types) of function (e.g. the exponential function) we write:

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

$f_1, f_2 \ \ldots \ f_n$ are prescribed functions (usually chosen with a fore-knowledge of how the function being fit ought to behave). $C_1 \ \ldots \ C_m$ are the unknown coefficients. Using least squares regression Equation (6.91) is used to fit a given set of n points $(x_1, y_1), \ldots, (x_n, y_n)$ by minimising the total error that is given by the sum of the squares of the residuals:

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

Curve Fitting with a Linear Combination of Nonlinear Functions:

Minimising E gives:

$$\frac{\partial E}{\partial C_k} = 0 \quad \text{for } k = 1, 2, \ldots, m \tag{6.93}$$

Substituting Equation (6.92) into (6.93) gives:

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

$$\text{for } k = 1, 2, \ldots, m \tag{6.94}$$

Curve Fitting with a Linear Combination of Nonlinear Functions:

Now define:

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

then Equation (6.94) becomes:

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

This can be rewritten:

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

This latter equation is a system of m linear equations in m unknowns (C_k) and so Equation (6.91) can be satisfied.

Example:

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)
```

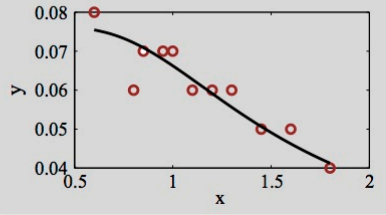
When the program is executed, the solution for the coefficients is displayed in the Command Window (the two elements of the vector `AB`), and a plot with the data points and the curve-fitted function is created.

Command Window:

```
AB =
    0.0743
   -0.0597
```

The coefficient A.

The coefficient B.



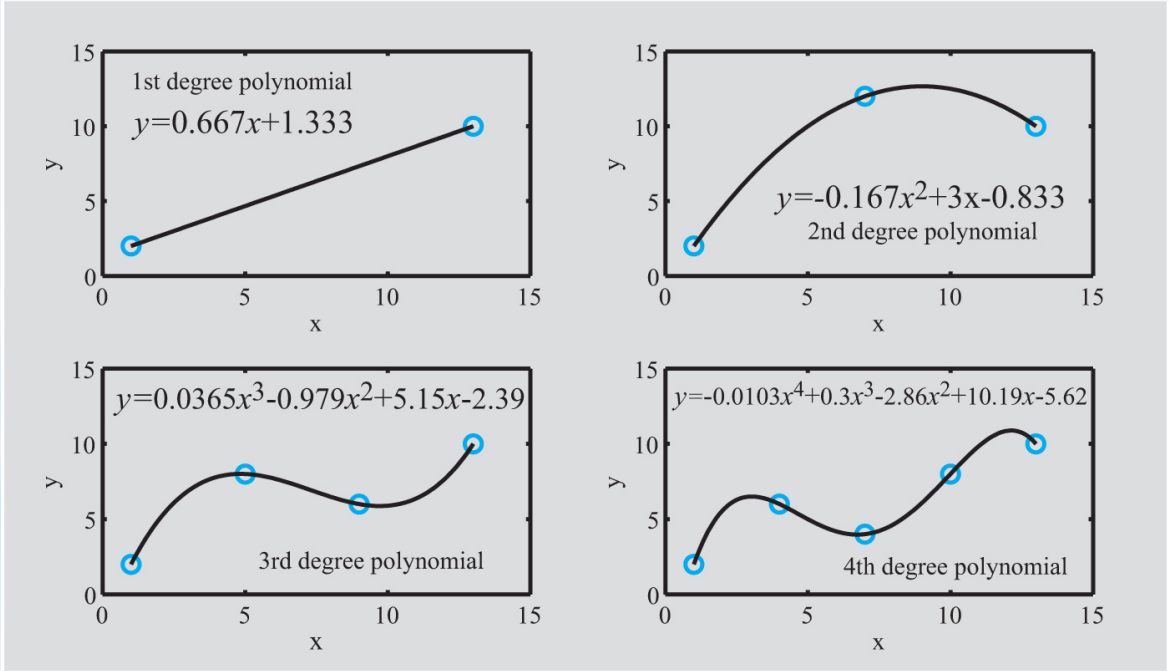
Interpolation using a Single Polynomial:

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 the data points and an estimated value between the points.

In contrast to this, curve fitting is the process of obtaining a mathematical formula, from the experimental data, that governs the phenomenon under observation. It is not necessary that the curve fit pass through all data points but rather gives a minimum overall error. The formula can then be used in theoretical calculations regarding this phenomenon including extrapolation.

Interpolation using a Single Polynomial:

For any n points there is a unique polynomial of order $n-1$ that passes through all of these points. For two points the order of the polynomial is 1 – giving us a line. For three points the order of the polynomial is 2 giving us a parabola connecting the points and so on.



Interpolation using a Single Polynomial:

The polynomial can be written in three different forms: standard, Lagrange and Newton's.

The standard form of an n^{th} order polynomial is:

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

The equations are written explicitly for each data point by substituting the value of each data point in turn in the polynomial.

This results in a system of linear equations in terms of the polynomial coefficients to be determined.

Interpolation using a Single Polynomial:

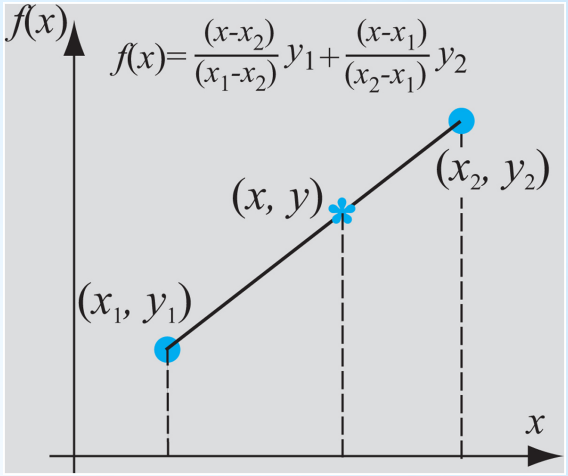
For example we have five points which means we have to use a fourth order polynomial to guarantee that it passes through all data points. The data is: (1,2), (4,6), (7,4), 10,8).

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

This system of equations can be solved for the coefficients of the polynomial using any of the methods described previously. However this process is not efficient and the coefficients matrix can be ill-conditioned. Hence we look further to polynomial representation in Lagrange and Newton's forms.

Interpolation using a Single Polynomial:

Lagrange Interpolating polynomials:



The coefficients of these polynomials can be carried out readily by hand or with a computer.

For a line the first order Lagrange polynomial that passes through two points has the form:

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

This can be shown to be equivalent to the standard definition for a line passing through two known points

Interpolation using a Single Polynomial:

Substituting two known points into the above equation gives:

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

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

Substituting the coefficients back into Equation (6.37) gives:

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

or

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

- which is the familiar equation for a line. Note that if $x = x_1$ is substituted into Equation (6.40) or (6.41) then the value of the polynomial is y_1. Ditto for $x = x_2$.

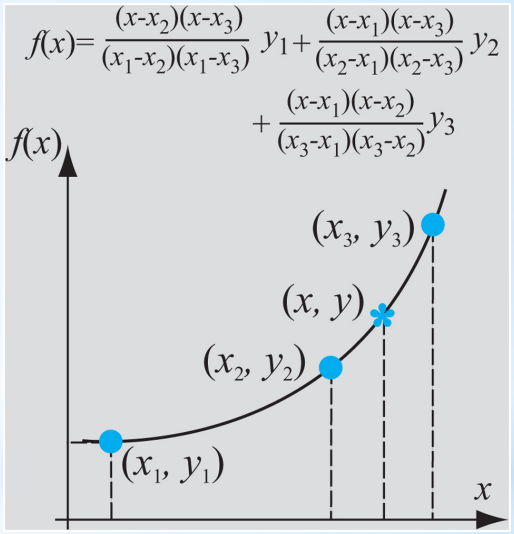
Interpolation using a Single Polynomial:

For three points a second order Lagrange polynomial is used:

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

Once the coefficients are determined such that the polynomial passes through the three data points the polynomial becomes a quadratic function of x as given in Equation (6.43).

Note when we substitute x_1 then $f(x) = y_1$.
This is because the the coefficient corresponding to y_1 is 1 and the other coefficients are zero.
This is characteristic of all Lagrange polynomials.



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

Interpolation using a Single Polynomial:

The general form of a Lagrange polynomial that passes through n points is:

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

or

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

where $L_i(x)=\prod_{j=1,j\neq i}^{n}\frac{(x-x_j)}{(x_i-x_j)}$ are known as Lagrange functions.

Interpolation using a Single Polynomial:

Notes on Lagrange Polynomials:

1. The spacing between the data points does not have to be equal.

2. For a given set of data points the whole expression for the interpolation polynomial has to be calculated for each interpolation value of x.

3. If a data point is added then the whole expression for the polynomial has to be calculated again. This does not have to be done with, for example, Newton's polynomials.

Example:

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);                          % The length of the vector x gives the number of terms in the polynomial.
for i = 1:n
  L(i) = 1;
  for j = 1:n
    if j ~= i                           % Calculate the product terms Li.
       L(i)= L(i)*(Xint-x(j))/(x(i)-x(j));
    end
  end
end
Yint = sum(y .*L);                      % Calculate the value of the polynomial f(x) = sum_{i=1}^{n} yi Li.
```

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
```

Interpolation using a Single Polynomial:

Newton's Interpolating Polynomials:

The general form of an $n-1$ order Newton's polynomials that passes through n points is:

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

The coefficients a_i are easily calculated. The major advantage of Newton's polynomials is that if a data point is added then the previous coefficients remain the same and only the new coefficient needs to be calculated so that the resulting polynomial passes through all points.

Interpolation using a Single Polynomial:

First Order Newton's Polynomial:

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

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

The coefficients can be calculated by evaluating the polynomial for (x_1, y_1) and (x_2, y_2) giving:

$$y_1 = a_1 + a_2(x_1 - x_1) \Longrightarrow a_1 = y_1$$

and

$$y_2 = a_1 + a_2(x_2 - x_1) \Longrightarrow a_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

Interpolation using a Single Polynomial:

Second Order Newton's Polynomial:

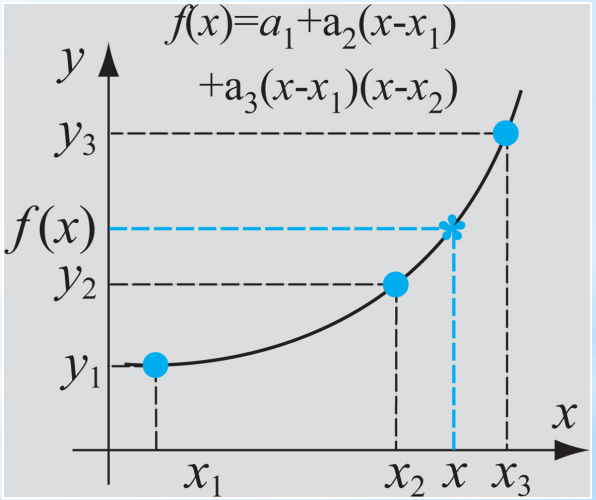
For three data points the second order Newton's polynomial has the form:

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

Substituting (x_1, y_1) gives $a_1 = y_1$ (as before).

Substituting (x_2, y_2) gives $a_2 = \frac{y_2 - y_1}{x_2 - x_1}$ (as before)

and substituting (x_3, y_3) gives:



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

Interpolation using a Single Polynomial:

Third Order Newton's Polynomial:

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

a_1, a_2 and a_3 are the same as for Newton's second order polynomial and

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

Interpolation using a Single Polynomial:

A General Form of Newton's Polynomial and its Coefficients:

For two points (a line):

$$a_1 = y_1$$

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

where $f[x_2, x_1]$ is known as the first divided difference.

Interpolation using a Single Polynomial:

For three points (a_1 and a_2 as before):

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

For four points (a_1, a_2 and a_3 as before):

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1}$$

$$= \frac{\dfrac{f[x_4, x_3] - f[x_3, x_2]}{x_4 - x_2} - \dfrac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1}}{(x_4 - x_1)} \quad (6.59)$$

$$= \frac{\dfrac{\dfrac{y_4 - y_3}{x_4 - x_3} - \dfrac{y_3 - y_2}{x_3 - x_2}}{(x_4 - x_2)} - \dfrac{\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}}{(x_3 - x_1)}}{(x_4 - x_1)} = a_4$$

Interpolation using a Single Polynomial:

In general the k^{th} divided difference for second and higher divided differences up to the $(n-1)$ divided difference is:

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

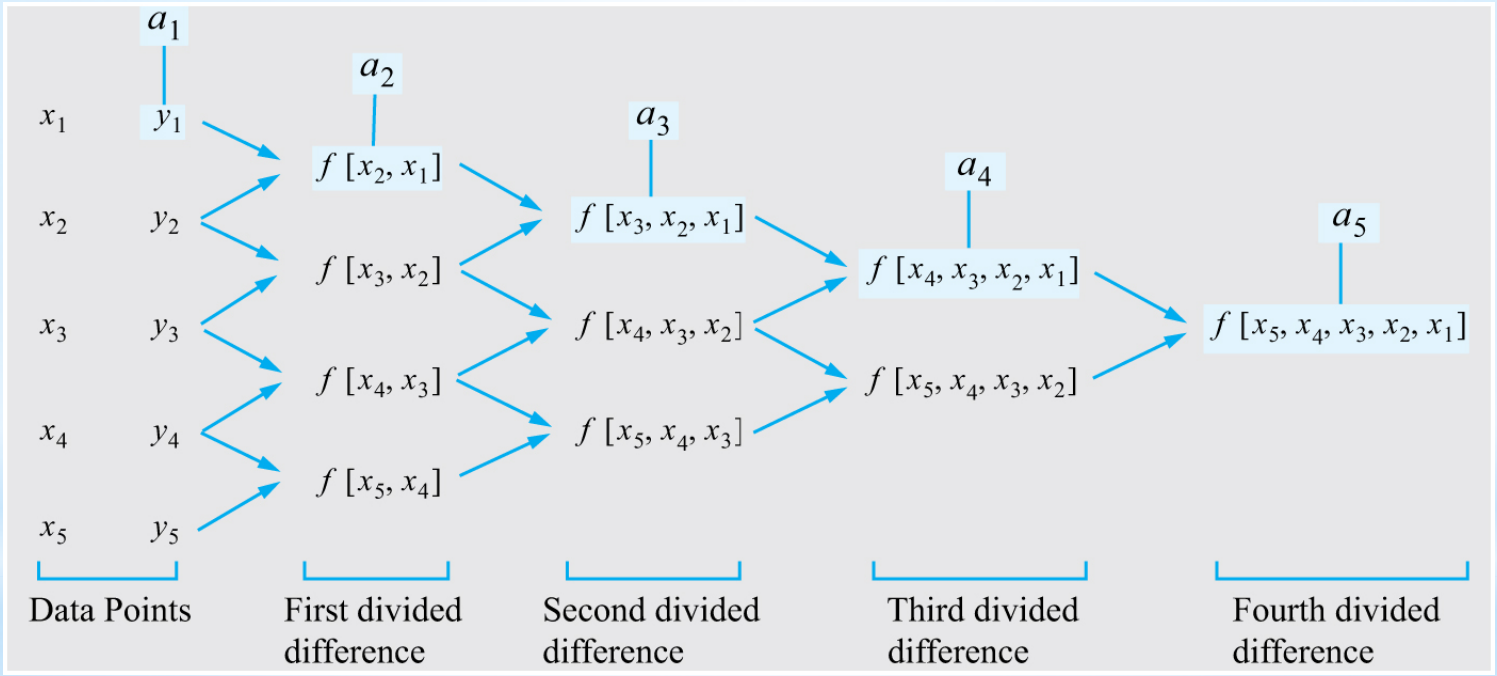
With these definitions the $(n-1)$ order Newton's polynomial is:

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

a_1 a_2 a_3 a_n

Interpolation using a Single Polynomial:

The manner in which the divided differences are related is given in the following diagram:



Interpolation using a Single Polynomial:

Notes on Newton's Polynomials:

1. The spacing between data points does not have to be the same.

2. Data points can be subsequently added and only one coefficient per extra data point needs to be calculated.

3. The data points do not have to be in any particular order.

For these reasons Newton's polynomials are a popular choice.

Example:

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

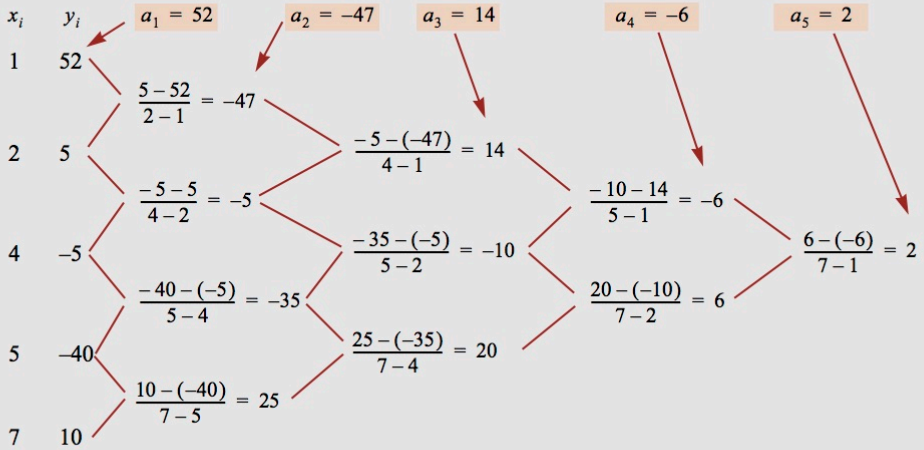
(*a*) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
(*b*) Use the polynomial obtained in part (*a*) to determine the interpolated value for $x=3$.
(*c*) Write a MATLAB user-defined function that interpolates 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:



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);
a(1)= y(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
```

The length of the vector x gives the number of coefficients (and terms) of the polynomial.

The first coefficient a_1.

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`.

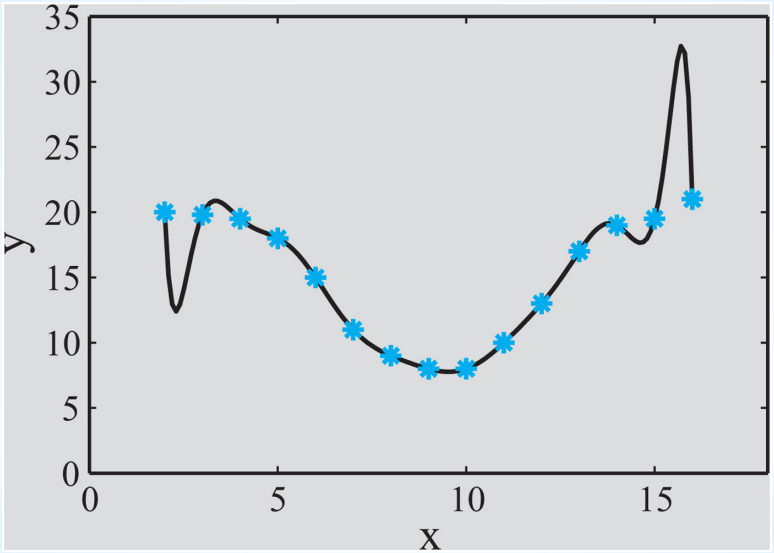
Calculate the interpolated value of `Xint`. The first term in the polynomial is a_1. The following terms are added by using a loop.

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
```

Piecewise (Spline) Interpolation:

When there are a large number of data points a single order polynomial interpolation may result in large deviations from the general trend (despite the resulting curve passing through all data points) as shown in the figure below:



To circumvent this problem it is possible to use lower order polynomials in a piecewise manner to interpolate. This form of interpolation is called piecewise of spline interpolation and the data points where the polynomials from two adjacent intervals meet are called knots.

Piecewise (Spline) Interpolation:

Linear Splines:

Interpolation is carried out using first order polynomials (linear functions) between the points. In other words the data points are connected with straight lines. Using the Lagrange form, the equation of a straight line that connects two points is given by:

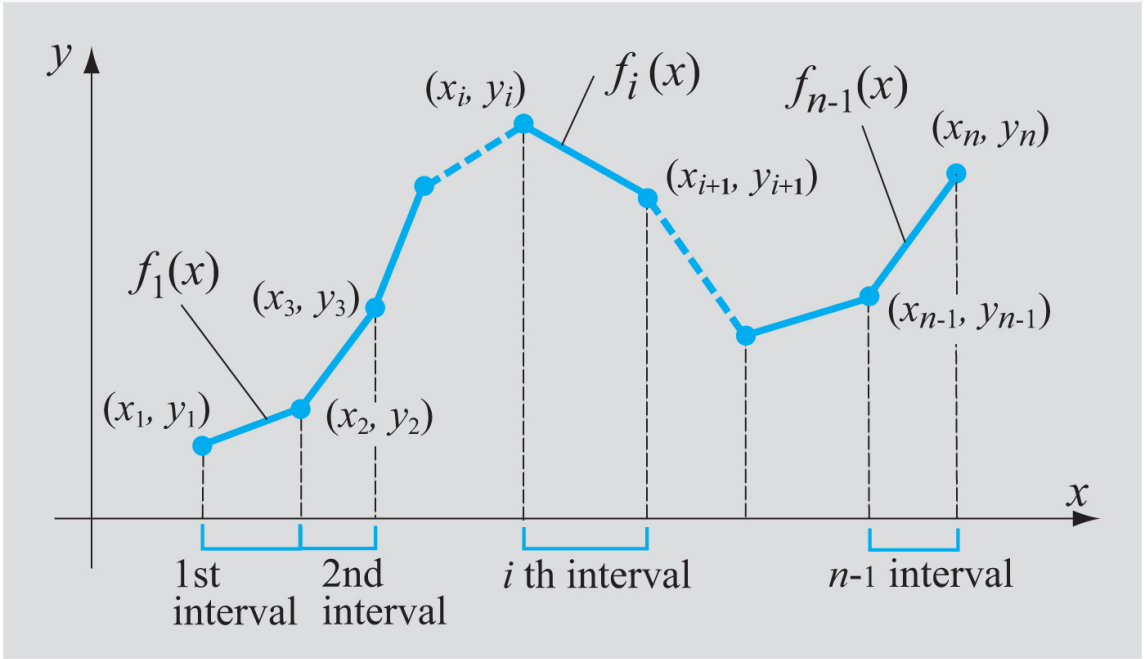
$$f_1(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2 \qquad (6.64)$$

Generalising for successive data points ((x_i, y_i) and (x_{i+1}, y_{i+1})) gives:

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

Piecewise (Spline) Interpolation:

Interpolation with linear splines is easy to do and gives good results when the data points are closely spaced. For widely spaced points it becomes less accurate because of large changes in the slope of the lines connecting the data points. In this latter respect it is preferable to use higher order polynomials.



Example:

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:

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

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

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

(*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);                          % The length of the vector x gives the number of terms in the data.
for i = 2:n                             % Find the interval that includes Xint.
   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));   % 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
```

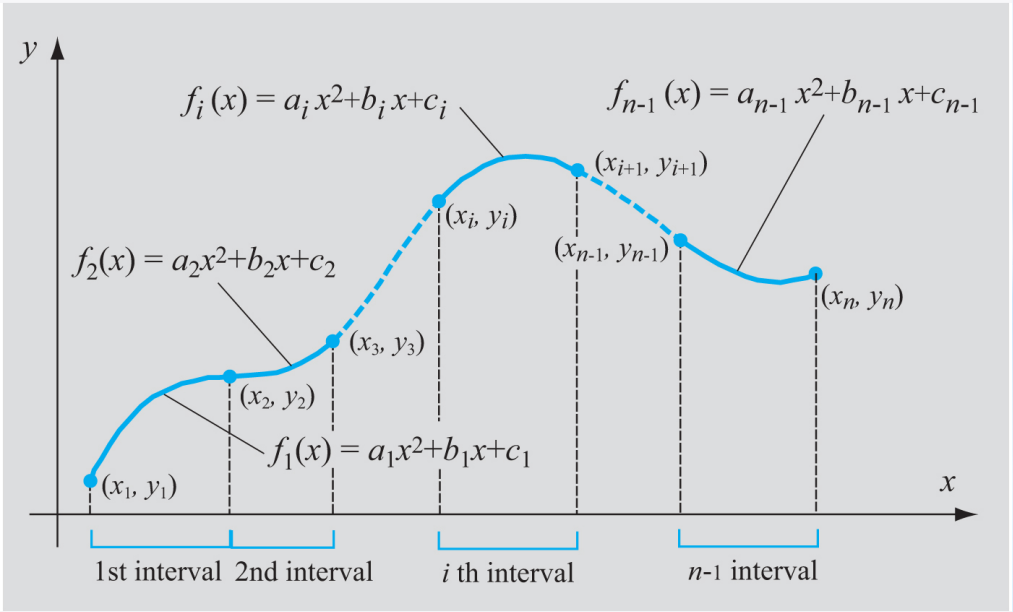
Piecewise (Spline) Interpolation:

Quadratic Splines:

Here second order polynomials are used for interpolation. The polynomial in the i^{th} interval between points (x_i, y_i) and (x_{i+1}, y_{i+1}) is:

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

For n data points there are $n-1$ intervals and consequently $n-1$ polynomials. Each polynomial has three coefficients so a total of $3(n-1)$ coefficients have to be determined.



Piecewise (Spline) Interpolation:

The coefficients are determined 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}). This means that $f_i(x_i, y_i) = y_i$ and $f_{i+1}(x_{i+1}, y_{i+1}) = y_{i+1}$. Put alternately:

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

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

Since there are $n - 1$ intervals these conditions give $2(n - 1)$ equations (two equations per interval).

Piecewise (Spline) Interpolation:

2. At the interior knots (there are $n-2$ of them) the slopes (first derivatives) from adjacent intervals are equal. In general the first derivative of the i^{th} polynomial is:

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

Equating the successive first derivatives at all of the interior points gives:

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

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

Piecewise (Spline) Interpolation:

From 1. and 2. we now have $2(n-1)+n-2=3n-4$ equations. However the $n-1$ polynomials have $3n-3$ coefficients so we need one more condition so that we can solve for the coefficients. This gives us our third condition:

3. We apply the condition that the second derivative at the first point is zero. The polynomial in the first interval is:

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

Hence $f_1''(x) = 2a_1$. Equating to zero gives $a_1 = 0$ which means that a straight line connects the first two points.

We now have $3n-3$ linear equations (in terms of the coefficients) that allow us to solve for all coefficients.

Example:

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 ($i = 1, \ldots, 4$). The quadratic equation for the ith 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):

$$i = 1 \quad f_1(x) = a_1x_1^2 + b_1x_1 + c_1 = b_1 8 + c_1 = 5$$
$$f_1(x) = a_1x_2^2 + b_1x_2 + c_1 = b_1 11 + c_1 = 9$$
$$i = 2 \quad f_2(x) = a_2x_2^2 + b_2x_2 + c_2 = a_2 11^2 + b_2 11 + c_2 = 9$$
$$f_2(x) = a_2x_3^2 + b_2x_3 + c_2 = a_2 15^2 + b_2 15 + c_2 = 10$$
$$i = 3 \quad f_3(x) = a_3x_3^2 + b_3x_3 + c_3 = a_3 15^2 + b_3 15 + c_3 = 10$$
$$f_3(x) = a_3x_4^2 + b_3x_4 + c_3 = a_3 18^2 + b_3 18 + c_3 = 8$$
$$i = 4 \quad f_4(x) = a_4x_4^2 + b_4x_4 + c_4 = a_4 18^2 + b_4 18 + c_4 = 8$$
$$f_4(x) = a_4x_5^2 + b_4x_5 + c_4 = a_4 22^2 + b_4 22 + c_4 = 7$$

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).

$$i = 2 \quad 2a_1x_2 + b_1 = 2a_2x_2 + b_2 \longrightarrow b_1 = 2a_2 11 + b_2 \quad \text{or:} \quad b_1 - 2a_2 11 - b_2 = 0$$
$$i = 3 \quad 2a_2x_3 + b_2 = 2a_3x_3 + b_3 \longrightarrow 2a_2 15 + b_2 = 2a_3 15 + b_3 \quad \text{or:} \quad 2a_2 15 + b_2 - 2a_3 15 - b_3 = 0$$
$$i = 4 \quad 2a_3x_4 + b_3 = 2a_4x_4 + b_4 \longrightarrow 2a_3 18 + b_3 = 2a_4 18 + b_4 \quad \text{or:} \quad 2a_3 18 + b_3 - 2a_4 18 - b_4 = 0$$

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

$$\begin{bmatrix} 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11^2 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22^2 & 22 & 1 \\ 1 & 0 & -22 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 1 & 0 & -30 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 36 & 1 & 0 & -36 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 9 \\ 10 \\ 10 \\ 8 \\ 8 \\ 7 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.72}$$

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 15^2 15 1 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
```

b_1 c_1 a_2 b_2 c_2 a_3 b_3 c_3 a_4 b_4 c_4

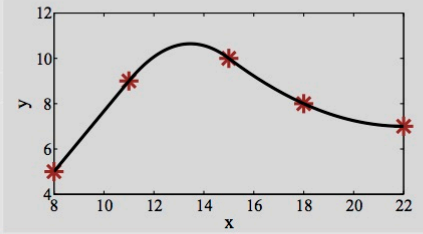
With the coefficients known, the polynomials are:

$$f_1(x) = 1.333x - 5.6667 \text{ for } 8 \le x \le 11, \qquad f_2(x) = (-0.2708)x^2 + 7.2917x - 38.4375 \text{ for } 11 \le x \le 15$$
$$f_3(x) = 0.0556x^2 - 2.5x + 35 \text{ for } 15 \le x \le 18, \qquad f_4(x) = 0.0625x^2 - 2.75x + 37.25 \text{ for } 18 \le x \le 22$$

(*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).



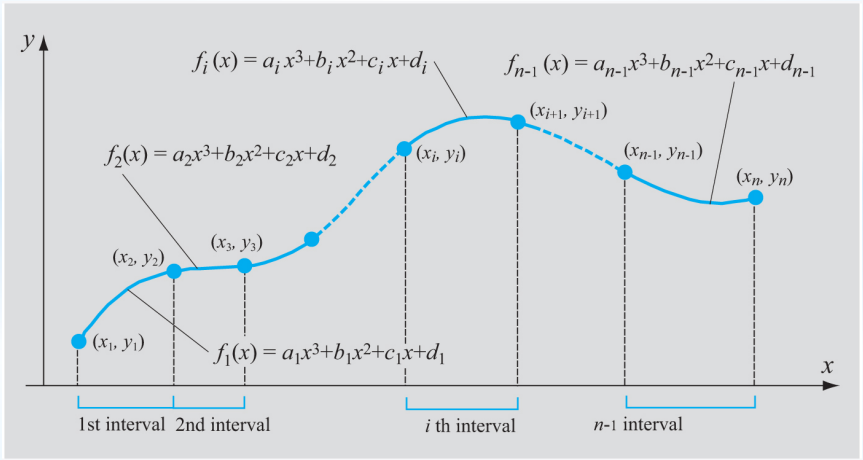
Piecewise (Spline) Interpolation:

Cubic Splines:

The equation of the polynomial in the i^{th} interval between points x_i and x_{i+1} is:

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

Overall there are $n-1$ such equations (corresponding to $n-1$ intervals) Since each equation has four coefficients then we need $4(n-1)$ equations (in terms of the coefficients) to solve for these. In like manner with the use of quadratic splines this is performed by imposing $4(n-1)$ conditions on the equations.



Piecewise (Spline) Interpolation:

Conditions imposed on the cubic polynomials to yield the necessary number of simultaneous equations $(4(n-1))$ in terms of the polynomial coefficients:

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}). This means that $f_i(x_i, y_i) = y_i$ and $f_{i+1}(x_{i+1}, y_{i+1}) = y_{i+1}$. Put alternately:

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

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

This gives us $2(n-1)$ equations.

Piecewise (Spline) Interpolation:

2. At the interior points the first derivatives (slopes) of the polynomials from adjacent intervals are equal. The first derivative of the i^{th} polynomial is:

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

Equating the first derivatives at each interior point gives:

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

Since there are $n-2$ interior points we get $n-2$ additional equations.

Piecewise (Spline) Interpolation:

3. At the interior points (knots) the second derivatives of the polynomials from the adjacent intervals must be equal. This means that the rate of change of the slopes at the knots (curvature) must be equal. The second derivative of the polynomial in the i^{th} interval is:

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

Equating the second derivatives at each interior points gives:

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

Since there are $n-2$ interior points then we have $n-2$ additional equations.

Piecewise (Spline) Interpolation:

4. Lastly we enforce the condition that the second derivatives at the first and last points equal zero. This gives the two remaining equations that we need to solve for all of the coefficients:

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

Cubic splines whose second derivatives are equal to zero at the end points are known as natural cubic splines.

Piecewise (Spline) Interpolation:

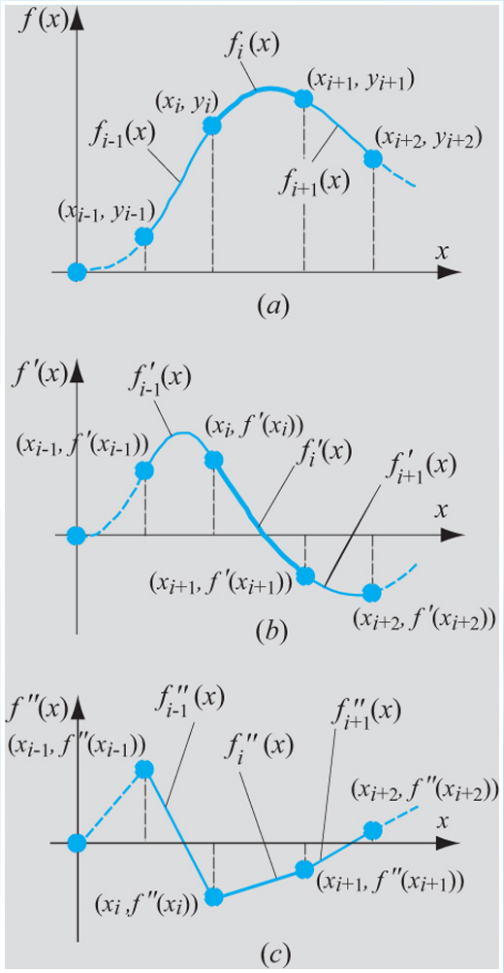
Cubic Splines based on Lagrange form Polynomials:

The derivation of cubic splines using the Lagrange form starts with the second derivative of the polynomial. The second derivative of a cubic polynomial is a linear function.
For the i^{th} interval this function can be written in the Lagrange form:

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

Hence the third order polynomial in the i^{th} interval can be obtained by integrating the above equation twice. This results in two constants of integration which can be obtained from the condition that the values of the polynomial at the knots are known:

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



Piecewise (Spline) Interpolation:

Once the constants of integration are determined the third order polynomial in the i^{th} interval is:

$$f_i(x) = \frac{f_i''(x_i)}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1}-x_i)}(x-x_i)^3 + \left[\frac{y_i}{x_{i+1}-x_i} - \frac{f_i''(x_i)(x_{i+1}-x_i)}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_i''(x_{i+1})(x_{i+1}-x_i)}{6}\right](x-x_i) \quad \text{for } x_i \le x \le x_{i+1} \quad \text{and } i = 1, 2, \ldots, n-1 \qquad (6.82)$$

Piecewise (Spline) Interpolation:

We obtain $n-2$ ***linear*** equations by enforcing continuity of the first derivatives at the interior points:

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

This condition is applied by differentiating Equation (6.82) and substituting these derivatives into Equation (6.83) giving the interpolating polynomial for each interval:

$$(x_{i+1}-x_i)f''(x_i) + 2(x_{i+2}-x_i)f''(x_{i+1}) + (x_{i+2}-x_{i+1})f''(x_{i+2}) = 6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} - \frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right] \quad \text{for} \quad i = 1, 2, \ldots, n-2 \qquad (6.84)$$

This is a system of $n-2$ equations with n unknowns ($f_i''(x_i)$ and $f_{i+1}''(x_{i+1})$).

Piecewise (Spline) Interpolation:

There remains two equations that are needed to solve for $f_i''(x_i)$ and $f_{i+1}''(x_{i+1})$. As before we enforce the condition that the second derivatives at the endpoints are equal to zero (giving us what are known as natural cubic splines):

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

Between Equations (6.84) and (6.85) we now have the $2(n-1)$ equations needed to express Equation (6.82).

Piecewise (Spline) Interpolation:

The process of solving for $f_i''(x_i)$ and $f_{i+1}''(x_{i+1})$ can be simplified by defining h_i as the length of the i^{th} interval and a_i as the second derivative of the polynomial at the point x_i. That is:

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

and

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

Piecewise (Spline) Interpolation:

With these definitions the equation of the polynomial in the i^{th} interval is:

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

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

and the system of linear equations that have to be solved for the a_is is:

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

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

a_1 and a_n are zero – natural splines.

Example:

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 ($i = 1, \ldots, 4$). The cubic equation in the ith spline is:

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

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_is are: $h_1 = x_2 - x_1 = 11 - 8 = 3$, $h_2 = x_3 - x_2 = 15 - 11 = 4$

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

$i = 1$ $\quad h_1 a_1 + 2(h_1 + h_2)a_2 + h_2 a_3 = 6\left[\frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1}\right]$

$$3 \cdot 0 + 2(3+4)a_2 + 4a_3 = 6\left[\frac{10-9}{4} - \frac{9-5}{3}\right] \longrightarrow 14a_2 + 4a_3 = -6.5$$

$i = 2$ $\quad h_2 a_2 + 2(h_2 + h_3)a_3 + h_3 a_4 = 6\left[\frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2}\right]$

$$4a_2 + 2(4+3)a_3 + 3a_4 = 6\left[\frac{8-10}{3} - \frac{10-9}{4}\right] \longrightarrow 4a_2 + 14a_3 + 3a_4 = -5.5$$

$i = 3$ $\quad h_3 a_3 + 2(h_3 + h_4)a_4 + h_4 a_5 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right]$

$$3a_3 + 2(3+4)a_4 + 4 \cdot 0 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right] \longrightarrow 3a_3 + 14a_4 = 2.5$$

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

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

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
```

a_2 $\quad a_3$ $\quad a_4$

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

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

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

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

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

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

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

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

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

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

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

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

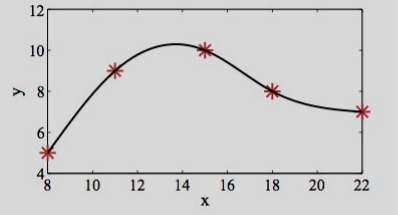
$$f_4(x) = 0.0105(22-x)^3 + 1.832(22-x) + 1.75(x-18) \quad \text{for} \quad 18 \le x \le 22$$

(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.



Recommended Problems:

Problems to be solved by hand (do at least 2):

6.3, 6.8, 6.13.

Problems to be programmed in MATLAB:

6.21

Problems in Science and Engineering (do 2):

6.31, 6.41

You should pick out problems that you find interesting/challenging and do these too.