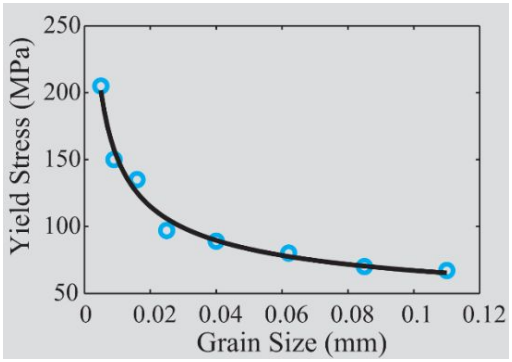
Formulae that are highlighted in green will be given in the exam, unless they are to be derived.

06 Curve-Fitting & Interpolation

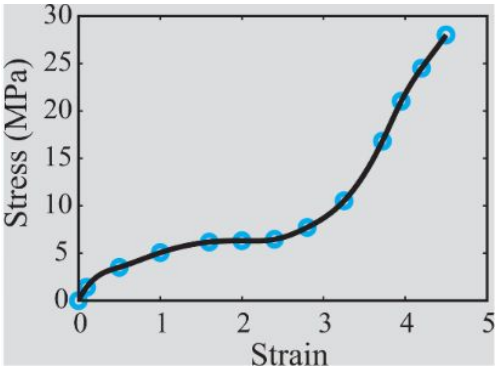
Experimental data is usually **discrete**, having been performed for discrete set of data points.

Oftentime we need to approximate these data with a function that can be used to **extrapolate** the 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 called **curve-fitting**.

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



Oftentimes, we need to estimate the value of the function **between data points**. This is done by finding a function or functions that **fit the data exactly**. This process is called **interpolation**.



Interpolation is used to find **intermediate values**. Interpolation can be 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. The is done when it is expected that the experimental variables are **proportional** to each other.

The equation of a line (first-degree polynomial) is:
$y = a_1 x + a_0$

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

The Residual

We define the **residual** as:
$r_1 = y_1 - f(x_i)$

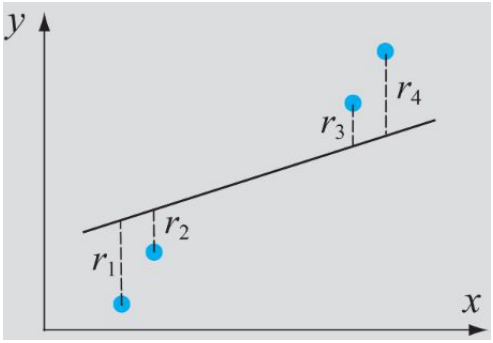
which is simply the difference between the data point y_1 and the value of the function $f(x_i)$ used to approximate it.

Total Error

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 a number of different ways.

This simplest way is to **add the residuals** over all data points:
$$E = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} \left[y_1 - (a_1 x_i + a_0) \right]$$

The total error does **not** give a good measure of the overall fit. This is because a bad fit with large positive and negative residuals may yield a small overall error.



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

$$E = \sum_{i=1}^{n} \left| r_i \right| = \sum_{i=1}^{n} \left| y_1 - (a_1 x_i + a_0) \right|$$

This can be a useful approach, however it now cannot be used to obtain the values of a_1 and a_0. 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.

A **good measure** for the overall error E that can be used to obtain a unique line of best fit is obtained by making E equal to the **sum of the squares** of the residuals:

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \left[y_1 - (a_1 x_i + a_0) \right]^2 \qquad \textbf{(6.6)}$$

With this approach, the overall error is a positive number.
Also, larger errors have a relatively larger effect on the total error.

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

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.

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \left[y_1 - (a_1 x_i + a_0) \right]^2 \qquad \textbf{(6.6 again)}$$

All the values x_i and y_i are known by experiment, and so equation (6.6) is a nonlinear 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 with respect to a_0 and a_1 and setting them to zero - we know that there is no maximum for E.

$$\frac{\partial E}{\partial a_0} = -2 \sum_{i=1}^{n} (y_i - a_1 x_i - a_0) = 0 \qquad \textit{(Partial derivative of \textbf{(6.6)} with respect to } a_0 \textit{)}$$

$$\frac{\partial E}{\partial a_1} = -2 \sum_{i=1}^{n} (y_i - a_1 x_i - a_0)\, x_i = 0 \qquad \textit{(Partial derivative of \textbf{(6.6)} with respect to } a_1 \textit{)}$$

This yields two simultaneous equations in a_0 and a_1:

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

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

Solving these simultaneous equations for a_0 and a_1 yields:

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

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

Since the summations are the same in these equations, it is convenient to perform these summations first, and then substitute the results into the equations:

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

Then:

$$a_1 = \frac{n S_{xy} - S_x S_y}{n S_{xx} - (S_x)^2}$$

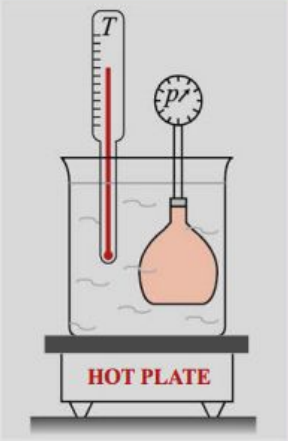
$$a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{n S_{xx} - (S_x)^2}$$

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

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).

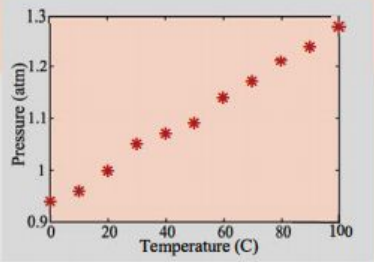
(*b*) Use linear least-squares regression to determine a linear function in the form $p = a_1 T + a_0$ that best fits the data points. First calculate the coefficients by hand using only the four data points: 0, 30, 70, and 100°C. Then write a user-defined MATLAB function that calculates the coefficients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

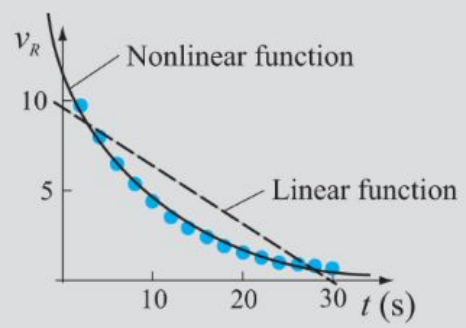
$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{4 \cdot 241.4 - (200 \cdot 4.44)}{4 \cdot 15800 - 200^2} = 0.003345$$

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

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

Curve-Fitting with a Nonlinear Equation by Rewriting in Linear Form

Many times the relationships between quantities is nonlinear.



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

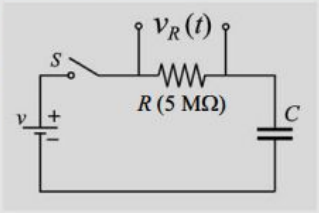
To write nonlinear equations in linear form, we take for example the power function:
$y = bx^m$

Then:
$ln(y) = ln(bx^m) = m\ ln(x) + ln(b)$ *(Using* $y = a_1 x + a_0$ *)*

This equation is linear for $ln(y)$ in terms of $ln(x)$. Using linear least-squares regression we can determine:
$a_1 = m,\ a_2 = ln(b)$
$\Rightarrow m = a_1,\ b = e^{(a_0)}$

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).

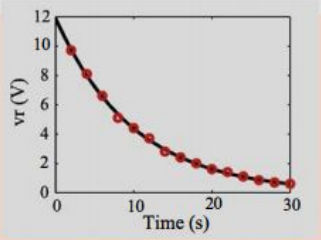
Calculate C using Eq. (6.18).

`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).

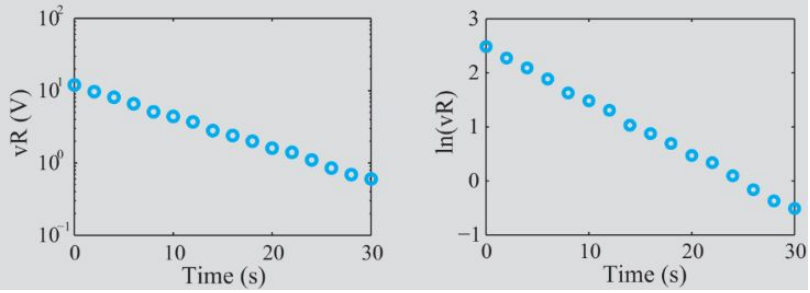
```
a1 =
   -0.1002
a0 =
    2.4776
b =
   11.9131
C =
   1.9968e-006
```

The capacitance is approximately 2 μF.



Choosing the appropriate nonlinear function to fit the data is often guided by a prior knowledge of what it is likely to be a best fit.

In the following example, we could plot v_R vs. time t with a logarithmic scale, or we could plot $ln(v_R)$ vs. time t to get a good straight line.



Other considerations when choosing a nonlinear function:

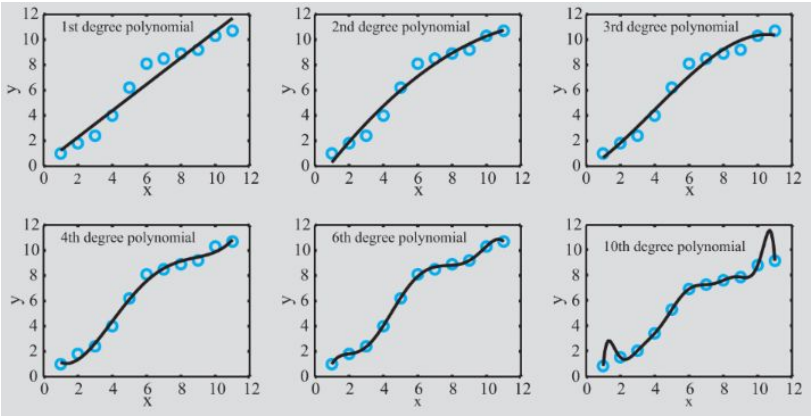
1. Exponential functions **cannot pass through the origin**.
2. Exponential functions can only fit data with all $+ve$ or $-ve$ ys.
3. Logarithmic functions cannot include $x = 0$ or x $-ve$.
4. For the **power function** $y = 0$ when $x = 0$.
5. The **reciprocal equation cannot include** $y = 0$.

Curve-Fitting with Quadratic and Higher-Order Polynomials

Polynomials are functions of the form:

$f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$

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



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

It is important to note that higher-order polynomials may pass through more points, but **may deviate significantly** from the overall trend thus making the approximate governing equation **inaccurate**.

Polynomial Regression

This is a procedure for determining the coefficients of a polynomial of second degree or higher, that best fits the data points.

The procedure is based on the same approach taken by linear regression (finding a polynomial of order 1 - a line) to fit the data. The goal is again to **minimise the overall error** in a **least squares** sense.

If the polynomial that is used for curve fitting is:

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

For a given set of n data points (x_i, y_i) the total error is:

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

where $m \leq n - 1$

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

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

For a minimum error, the partial derivatives must be zero:

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

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

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

This system of linear equations in terms of the unknowns a_0, a_1 and a_2 can be rewritten:

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

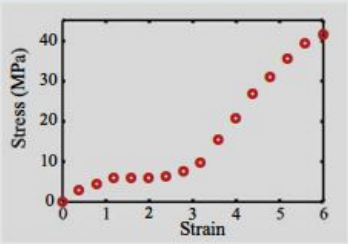
$$(\sum_{i=1}^{n} x_i) a_0 + (\sum_{i=1}^{n} x_i^2) a_1 + (\sum_{i=1}^{n} x_i^3) a_2 = \sum_{i=1}^{n} x_i y_i$$

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

The solution to 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 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
% 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.

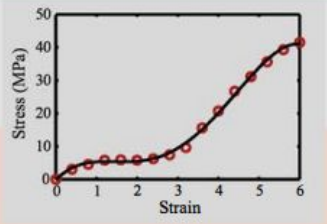
Create a new vector for the coefficients of the polynomial, to be used in MATLAB's `polyval` built-in function (see note at the end of the example).

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:

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + ... + C_m f_m(x) = \sum_{j=1}^{m} C_j f_j(x_i) \qquad \textbf{(6.91)}$$

where $f_1, f_2, ..., f_n$ are predescribed functions, and $C_1, C_2, ..., C_n$ are unknown coefficients.

Using **least squares regression**, equation (6.91) is used to fit a given set of n points $(x_1,y_1), ..., (x_n,y_n)$ by **minimising the total error** that is given by the **sum of squares** of the **residuals**:

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

Minimising E gives:

$$\frac{\partial E}{\partial C_k} = 0 \qquad for\ k = 1, 2, ..., m \qquad \textbf{(6.93)}$$

Substituting equation (6.92) into (6.93) gives:

$$\frac{\partial E}{\partial C_k} = \sum_{i=1}^{n} 2 \left[y_i - \sum_{j=1}^{m} C_j f_j(x_i) \right] \left[- \frac{\partial}{\partial C_k} \left(\sum_{j=1}^{m} C_j f_j(x_i) \right) \right] = 0 \qquad for\ k = 1, 2, ..., m \qquad \textbf{(6.94)}$$

Now define:

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

Equation (6.94) becomes:

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

This can be rewritten:

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

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

Example 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)
```

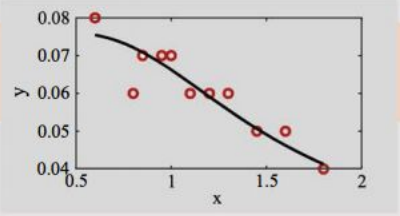
When the program is executed, the solution for the coefficients is displayed in the Command Window (the two elements of the vector AB), and a plot with the data points and the curve-fitted function is created.

Command Window:

```
AB =
   0.0743
  -0.0597
```

The coefficient A.

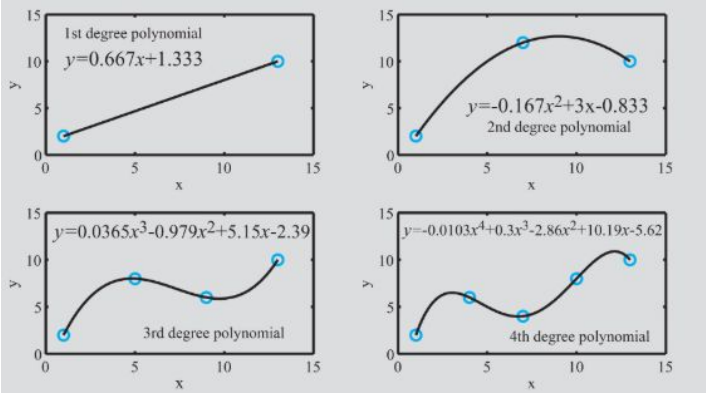
The coefficient B.



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**.

For any n points, there is a unique polynomial of order $n-1$ that passes through **all** the points. For two points the order of the polynomial is 1 - giving us a line.



The polynomial can be written in three forms:

1. Standard
2. Lagrange
3. Newton's

Standard Form

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

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

The equations are written explicitly for each data point by substituting the value of each data point in turn into the polynomial. This results in a **system of linear equations** in terms of the polynomial coefficients to be determined.

Example:
We have five points, so we need to use a 4th order polynomial to guarantee that it passes through all data points.

The data is: $(1,2),\ (4,6),\ (7,4),\ (10,8),\ (13,10)$

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

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.

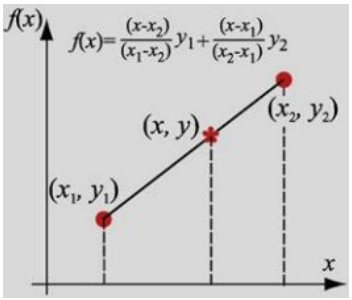
Hence, we look further to polynomial representation in Lagrange and Newton's forms.

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 the two points has the form:

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



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

Substituting two known points into the above equation gives:

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

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

Substituting the coefficients back into equation (6.37) gives:

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

or

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

which is a familiar equation for a line through two known points (x_1, y_1) and (x_2, y_2).

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 $x = x_2$.

Second Order Lagrange Polynomials

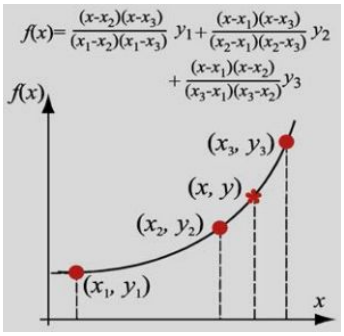
For **three points**, a second order Lagrange polynomial is used:

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

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).

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

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



General Form

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

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

or

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

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

Notes on Lagrange Polynomials

- The **spacing** between data points does **not** have to be equal.
- For a given set of data points, the whole expression for the interpolation polynomial has to be calculated for each interpolation value of x.
- 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 Newton's polynomials.

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:

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

$$f(3) = -5.778 + 2.\ \ 7 - 4.444 + 13.\ \ 33 + 0.222 = 6$$

(*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`.

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

```
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
```

Newton's Interpolating Polynomials

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

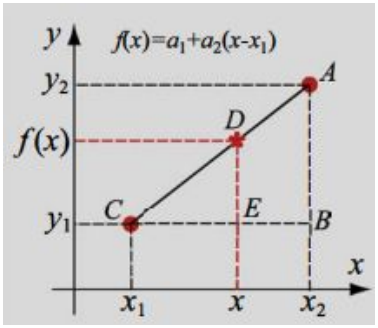
$$f(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + ... + a_n(x - x_1)(x - x_2)...(x - x_{n-1})$$

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.

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:

$$f(x) = a_1 + a_2(x - x_1)$$



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

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

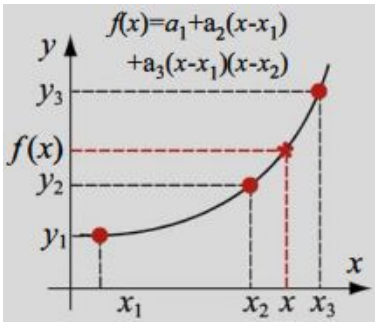
and

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

Second Order Newton's Polynomial

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

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



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.

Substituting (x_3, y_3) gives:

$$a_3 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}$$

Third Order Newton's Polynomial

$$f(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + a_4(x - x_1)(x - x_2)(x - x_3)$$

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

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

General Form for Newton's Polynomial and Coefficients

For two points (a line):

$a_1 = y_1$

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

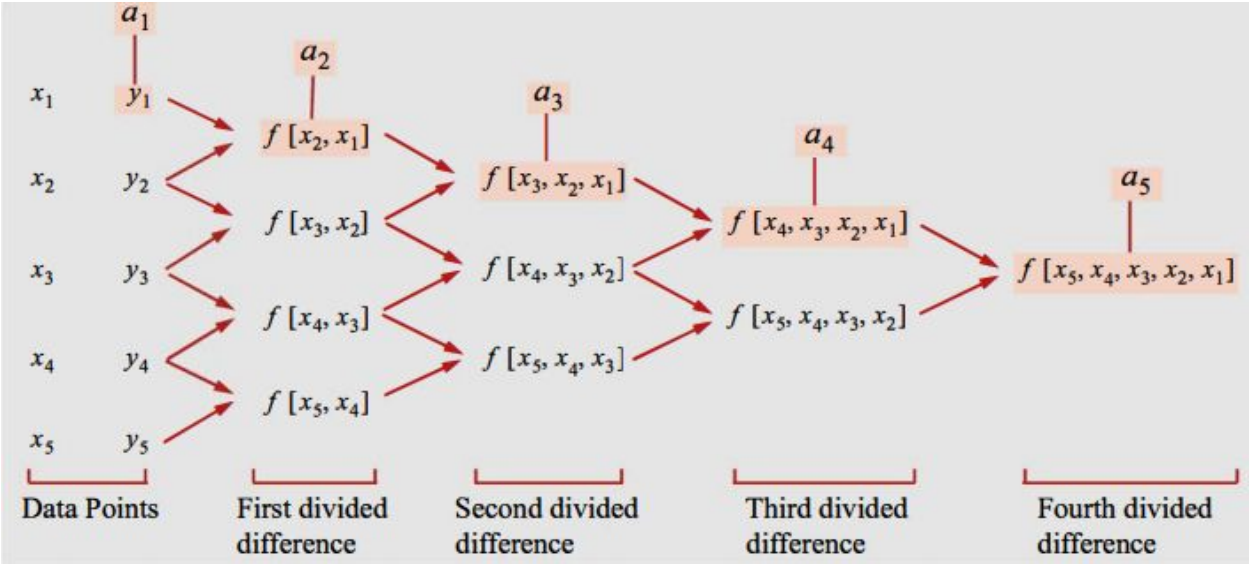
where $f[x_2, x_1]$ is known as the **first divided difference**.

For three points (a_1 and a_2 as before):

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

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

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1} = \frac{\frac{f[x_4, x_3] - f[x_3, x_2]}{x_4 - x_2} - \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1}}{x_3 - x_1} = \ldots = a_4$$



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

$$f\left[x_k, x_{k-1}, \ldots, x_2, x_1\right] = \frac{f\left[x_k, x_{k-1}, \ldots, x_3, x_2\right] - f\left[x_{k-1}, x_{k-2}, \ldots, x_2, x_1\right]}{x_k - x_1}$$

With these definitions, the $n-1^{\text{th}}$ polynomial is:

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

a_1 a_2 a_3 a_n

Notes on Newton Polynomials

- The spacing between the data points does not have to be the same.
- Data points can be subsequently added and only one coefficient per extra data point needs to be calculated.
- The data points do not have to be in any particular order.

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

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

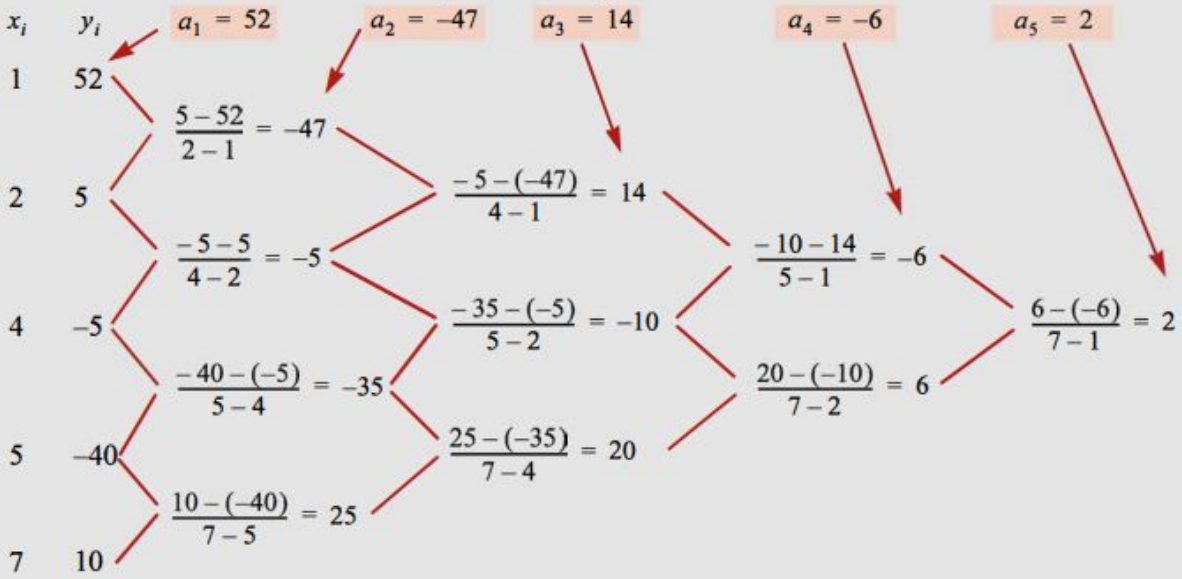
(*a*) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
(*b*) Use the polynomial obtained in part (*a*) to determine the interpolated value for $x=3$.
(*c*) Write a MATLAB user-defined function that interpolates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

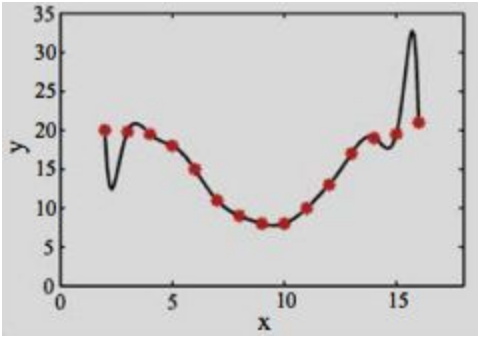
Calculate the interpolated value of `Xint`. The first term in the polynomial is a_1. The following terms are added by using a loop.

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 (Spinewise) Interpolation

When there are a large number of data points, a single-order polynomial interpolation may result in large deviations from the general trend.



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** or **spline interpolation**, and the data points where the polynomials from **two adjacent intervals meet** are called **knots**.

Linear Splines

Interpolation is carried out using first-order polynomials (linear functions) between the points. In other words, the data points are connected by straight lines.

Using **Lagrange form**, the equation of the straight line that connects two points is given by:

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

Generalising for successive data points (x_1, y_1) and (x_2, y_2) gives:

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

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 the large changes in slope of the lines connecting the data points. In this case it is preferable to use higher-order polynomials.

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

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

Find the interval that includes Xint.

Calculate Yint with Eq. (6.65).

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

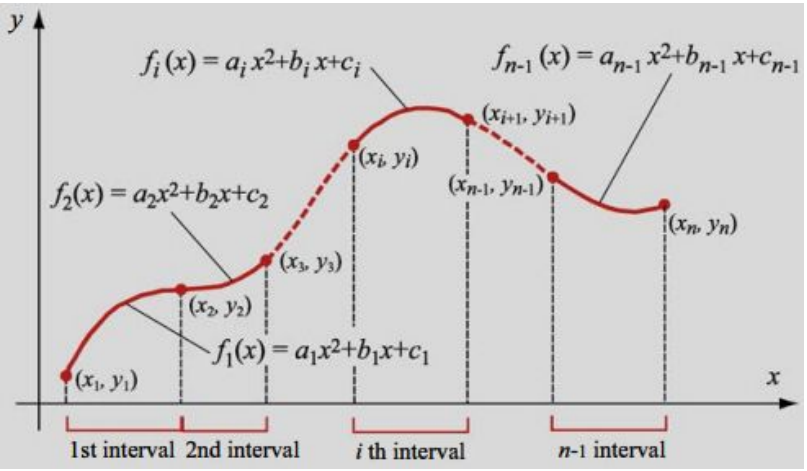
```
>> x = [8 11 15 18];
>> y = [5 9 10 8];
>> Yint = LinearSpline(x,y,12.7)
Yint =
    9.4250
```

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:

$f_i(x) = a_i x^2 + b_i x + c_i \quad for\ i = 1, 2, ..., n-1$

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.



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}).

   $a_i x_i^2 + b_i x_i + c_i = y_i \qquad for\ i = 1, 2, ..., n-1$
   $a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \qquad for\ i = 1, 2, ..., n-1$

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

2. At the interior points (knots) (there are $n-2$ of them), the **slopes** (first derivative) **from adjacent intervals are equal**.

   In general, the first derivative of the i^{th} polynomial is:
   $f'(x) = \frac{df}{dx} = 2a_i x + b_i$

   Equating the successive first derivatives at all of the interior points gives:
   $2a_{i-1} x + b_{i-1} = 2a_i x + b_i \qquad for\ i = 2, 3, ..., n-1$

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

From *1.* and *2.* we now have $2(n-1)+(n-2)=3n-4$ equations.
However, the $n-1$ polynomials have $n-3$ coefficients, so we need one more condition so that we can solve for the coefficients.

3. We apply the condition that the second derivative at the first point is zero.
   The polynomial in the first interval is:
   $f_1(x)=a_1x^2+b_1x+c_1$
   Hence $f_1''(x)=2a_1$

   Equating this to zero gives $a_1=0$ which means that a **straight line** connects the first two points.

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

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):

$$\begin{aligned}
i = 1 \quad & f_1(x) = a_1 x_1^2 + b_1 x_1 + c_1 = b_1 8 + c_1 = 5 \\
& f_1(x) = a_1 x_2^2 + b_1 x_2 + c_1 = b_1 11 + c_1 = 9 \\
i = 2 \quad & f_2(x) = a_2 x_2^2 + b_2 x_2 + c_2 = a_2 11^2 + b_2 11 + c_2 = 9 \\
& f_2(x) = a_2 x_3^2 + b_2 x_3 + c_2 = a_2 15^2 + b_2 15 + c_2 = 10
\end{aligned}$$

$$\begin{aligned}
i = 3 \quad & f_3(x) = a_3 x_3^2 + b_3 x_3 + c_3 = a_3 15^2 + b_3 15 + c_3 = 10 \\
& f_3(x) = a_3 x_4^2 + b_3 x_4 + c_3 = a_3 18^2 + b_3 18 + c_3 = 8 \\
i = 4 \quad & f_4(x) = a_4 x_4^2 + b_4 x_4 + c_4 = a_4 18^2 + b_4 18 + c_4 = 8 \\
& f_4(x) = a_4 x_5^2 + b_4 x_5 + c_4 = a_4 22^2 + b_4 22 + c_4 = 7
\end{aligned}$$

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).

$$\begin{aligned}
i = 2 \quad & 2a_1 x_2 + b_1 = 2a_2 x_2 + b_2 \longrightarrow b_1 = 2a_2 11 + b_2 \quad \text{or:} \quad b_1 - 2a_2 11 - b_2 = 0 \\
i = 3 \quad & 2a_2 x_3 + b_2 = 2a_3 x_3 + b_3 \longrightarrow 2a_2 15 + b_2 = 2a_3 15 + b_3 \quad \text{or:} \quad 2a_2 15 + b_2 - 2a_3 15 - b_3 = 0 \\
i = 4 \quad & 2a_3 x_4 + b_3 = 2a_4 x_4 + b_4 \longrightarrow 2a_3 18 + b_3 = 2a_4 18 + b_4 \quad \text{or:} \quad 2a_3 18 + b_3 - 2a_4 18 - b_4 = 0
\end{aligned}$$

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

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

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

```
>> A = [8 1 0 0 0 0 0 0 0 0 0; 11 1 0 0 0 0 0 0 0 0 0; 0 0 11^2 11 1 0 0 0 0 0 0
0 0 15^2 15 1 0 0 0 0 0 0; 0 0 0 0 0 15^2 15 1 0 0 0; 0 0 0 0 0 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 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 = (
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:

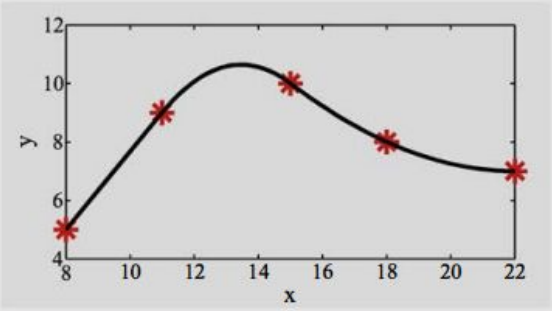
$f_1(x) = 1.333x - 5.6667$ for $8 \le x \le 11$, $f_2(x) = (-0.2708)x^2 + 7.2917x - 38.4375$ for $11 \le x \le 15$

$f_3(x) = 0.0556x^2 - 2.5x + 35$ for $15 \le x \le 18$, $f_4(x) = 0.0625x^2 - 2.75x + 37.25$ 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 $f_2(x)$:

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

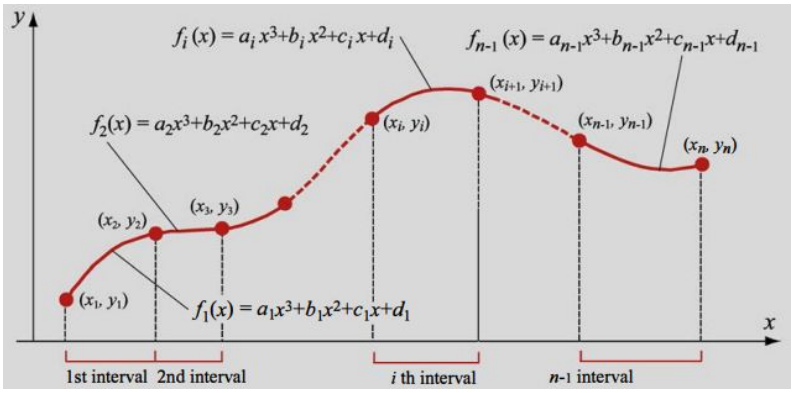
(*c*) The plot on the right shows the data points and the polynomials. The plot clearly shows that the first spline is a straight line (constant slope).



Cubic Splines

The equation of the polynomial in the i^{th} interval between points x_i and x_{i+1} is:
$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$

Overall, there are $n-1$ such equations (corresponding to $n-1$) intervals.
Since each equation has four coefficients, we need $4(n-1)$ equations (in terms of coefficients) in order to solve for these.
Like with the use of quadratic equations, we impose $4(n-1)$ conditions on the equations.



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})

   $a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = y_i \qquad for\ i = 1, 2, ..., n-1$
   $a_i x_{i+1}^3 + b_i x_{i+1}^2 + c_i x_{i+1} + d_i = y_i \qquad for\ i = 1, 2, ..., n-1$

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

2. At the interior knots (there are $n-2$ of them), the **slopes** (first derivative) **from adjacent intervals are equal**.

   In general, the first derivative of the i^{th} polynomial is:
   $f'(x) = \frac{df}{dx} = 3a_i x^2 + 2b_i x + c_i$

   Equating the successive first derivatives at all of the interior points gives:
   $3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_i = 3a_i x_i^2 + 2b_i x_i + c_i \ for\ i = 2, 3, ..., n-1$

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

4. At the interior points (knots), the **second derivatives** of the polynomials **from adjacent intervals are equal**. The rate of change of the slopes must be equal.

   The second derivative of the polynomial in the i^{th} interval is:
   $f''(x) = \frac{d^2 f}{dx^2} = 6a_i x + 2b_i$

   Equating the successive first derivatives at all of the interior points gives:
   $6a_i x_{i-1} + 2b_{i-1} = 6a_i x_i + 2b_i \qquad for\ i = 2, 3, ..., n-1$

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

5. 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 coefficients.
   $6a_1 x_1 + 2b_1 = 6a_{n-1} x_n + 2b_{n-1}$

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

Cubic Splines based on Lagrange Form Polynomials

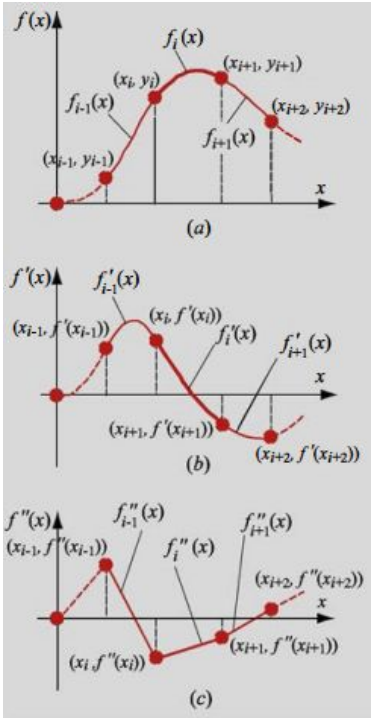
The derivation of cubic splines using the Lagrange form starts with the **second derivative** of the polynomial. The second derivative of a cubic polynomial is a **linear function**.

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 knot are known:

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



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

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

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

$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \qquad for\ i = 1, 2, ..., n-2$ **(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:

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

This is a system of $n-2$ equations with n unknowns ($f_i''(x_i)$ and $f_{i+1}''(x_{i+1})$).
As before we enforce the condition that the second derivatives at the endpoints are equal to zero.

$f''(x_1) = 0$ and $f''(x_n) = 0$ **(6.85)**

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

The process of solving for $f_i''(x_i)$ and $f_{i+1}''(x_{i+1})$ can be **simplified** by defining:

1. h_i as the **length** of the i^{th} interval
2. a_i as the **second derivative** of the polynomial at the point x_i

$h_i = x_{i+1} - x_i$
$a_i = f''(x_i)$

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

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

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

$$h_i a_i + 2(h_i + h_{i+1}) a_{i+1} + h_{i+1} a_{i+2} = 6\left[\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}\right] \qquad for\ i = 1, 2, ..., n-2$$

a_i and a_n are zero - natural splines.

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

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

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

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

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

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

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

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

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

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

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 a_3 a_4

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

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

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

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

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

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

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

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

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

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

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

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

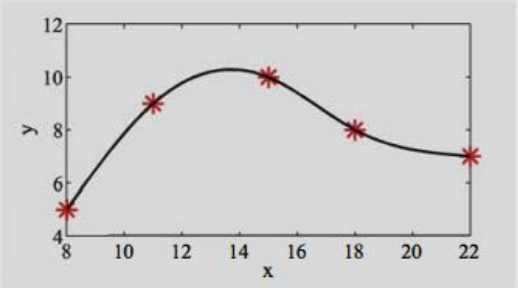
$$f_4(x) = 0.0105(22 - x)^3 + 1.832(22 - x) + 1.75(x - 18) \quad \text{for } 18 \le x \le 22$$

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

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

$$f_2(x) = 10.11$$

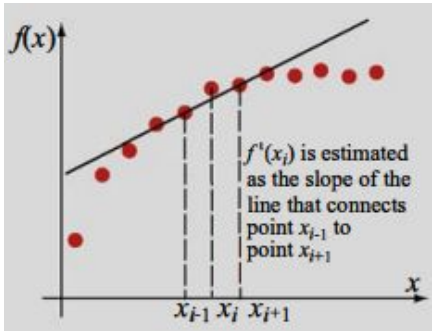
(*c*) The plot on the right shows the data points and the polynomial.



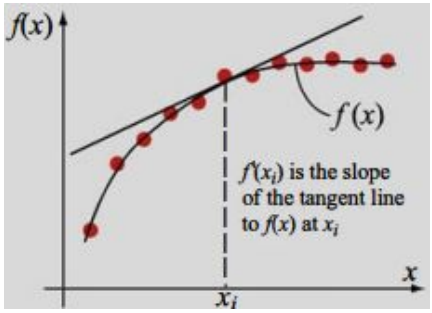
08 Numerical Differentiation

Numerical differentiation is needed where the function in question cannot be differentiated analytically, cannot be differentiated easily and / or there are many differentiations to be performed.

The simplest approach is to use the **finite difference approximation** for the derivative.
This approach calculates the derivative $f'(x_i)$ by obtaining the slope of the line containing $f'(x_{i-1})$ and $f'(x_{i+1})$ which are experimental data points or values of a discretised continuous function.



Another approach is to obtain an analytic expression for $f(x)$ by **curve-fitting** and differentiating the resulting analytic expression.

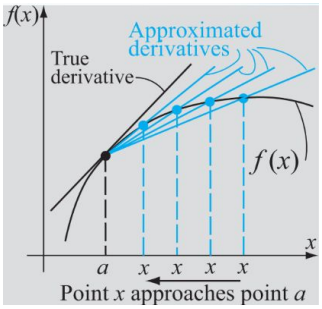


Oftentimes there is a lot of scatter in experimental data which means that there can be large inaccuracies in differentiation using the finite difference method. This can be remedied using **higher-order** finite difference approximations.

Finite Difference Approximation of the Derivative

The derivative $f'(x)$ of a function $f(x)$ at the point $x = a$ is defined by:

$$\frac{df(x)}{dx}\Big|_{x=a} = f'(a) = \lim_{x \to a} \frac{f(x)-f(a)}{x-a}$$



This is the slope of the tangent to the function $f(x)$ at the point $x = a$. The derivative is obtained by choosing a point x close to a and calculating the slope of the line containing these two points.

The closer x is to a the more accurate the result.

In finite difference approximations of the derivative, values of the function at different points in the neighbourhood of $x = a$ are used for estimating the slope.

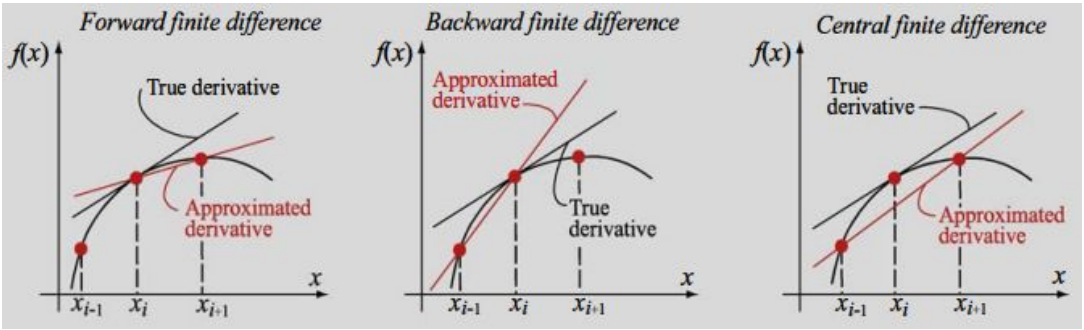
Three such fundamental variants exist:

1. The **forward-difference** formula:
$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}$$

2. The **backward-difference** formula:
$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}}$$

3. The **central-difference** formula:
$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_{i+1})-f(x_{i-1})}{x_{i+1}-x_{i-1}}$$

If the function is **slowly varying**, then the **central-difference formula** is usually the most accurate. That being said, all are accurate if the spacings between the points are made very small.



Example 8-1: Comparing numerical and analytical differentiation.

Consider the function $f(x) = x^3$. Calculate its first derivative at point $x = 3$ numerically with the forward, backward, and central finite difference formulas and using:

(*a*) Points $x = 2$, $x = 3$, and $x = 4$.

(*b*) Points $x = 2.75$, $x = 3$, and $x = 3.25$.

Compare the results with the exact (analytical) derivative.

SOLUTION

***Analytical differentiation*:** The derivative of the function is $f'(x) = 3x^2$, and the value of the derivative at $x = 3$ is $f'(3) = 3 \cdot 3^2 = 27$.

Numerical differentiation

(*a*) The points used for numerical differentiation are:

x:	2	3	4
$f(x)$:	8	27	64

Using Eqs. (8.5) through (8.7), the derivatives using the forward, backward, and central finite difference formulas are:

Forward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(4)-f(3)}{4-3} = \frac{64-27}{1} = 37 \qquad error = \left|\frac{37-27}{27}\cdot 100\right| = 37.04\,\%$$

Backward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3)-f(2)}{3-2} = \frac{27-8}{1} = 19 \qquad error = \left|\frac{19-27}{27}\cdot 100\right| = 29.63\,\%$$

Central finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(4)-f(2)}{4-2} = \frac{64-8}{2} = 28 \qquad error = \left|\frac{28-27}{27}\cdot 100\right| = 3.704\,\%$$

(*b*) The points used for numerical differentiation are:

x:	2.75	3	3.25
$f(x)$:	2.75^3	3^3	3.25^3

Using Eqs. (8.5) through (8.7), the derivatives using the forward, backward, and central finite difference formulas are:

Forward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3.25)-f(3)}{3.25-3} = \frac{3.25^3-27}{0.25} = 29.3125 \qquad error = \left|\frac{29.3125-27}{27}\right|\cdot 100 = 8.565\ \%$$

Backward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3)-f(2.75)}{3-2.75} = \frac{27-2.75^3}{0.25} = 24.8125 \qquad error = \left|\frac{24.8125-27}{27}\right|\cdot 100 = 8.102\ \%$$

Central finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3.25)-f(2.75)}{3.25-2.75} = \frac{3.25^3-2.75^3}{0.5} = 27.0625 \qquad error = \left|\frac{27.0625-27}{27}\right|\cdot 100 = 0.2315\ \%$$

The results show that the central finite difference formula gives a more accurate approximation. This will be discussed further in the next section. In addition, smaller separation between the points gives a significantly more accurate approximation.

Finite Difference Formulae using the Taylor Series Expansion

The forward, backward and central-difference formulae, as well as others, can be obtained using the Taylor series expansion of $f(x)$.
The advantage of using the Taylor series expansion is that an estimate of the error in the approximation can be obtained.

The Two-Point Forward-Difference Formula

Assuming that points are evenly-spaced, we can write:
$O(h^2)$

This can be written:
$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(\zeta)}{2!}h^2$
where $h = x_{i+1} - x_i$ and $\zeta \in]x_i, x_{i+1}[$

Solving for $f'(x_i)$ yields:
$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(\zeta)}{2!}h$

An approximate value of the derivative $f'(x_i)$ can be calculated if the second term of the RHS is ignored. Ignoring this term introduces a truncation error which is said to be in the 'order of h', written $O(h)$, since it is proportional to h.
$TruncationError = \frac{f''(\zeta)}{2!}h = O(h)$

Using this notation, we can write the forward-difference formula as:
$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$

The Two-Point Backward-Difference Formula

Using Taylor's theorem we can write:

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 + \ldots$$

As before, this reduces to:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \frac{f''(\zeta)}{2!}h$$

This in turn gives:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$$

The Two-Point Central-Difference Formula

We write:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(\zeta_1)}{3!}h^3$$

and

$$f(x_{i-1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(\zeta_2)}{3!}h^3$$

The spacings between points are equal ($h = x_{i+1} - x_i = x_i - x_{i-1}$).
Subtracting the latter from the former equation gives:

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + \frac{f'''(\zeta_1)}{3!}h^3 + \frac{f'''(\zeta_2)}{3!}h^3$$

Solving for $f'(x_i)$ gives:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)$$

This gives us the noteworthy result that the central-difference formula is more accurate than the forward and backward-difference formula, because the order of the truncation error is $O(h^2)$ rather than $O(h)$.

At the **endpoints** it is not possible to use the central difference formula for the derivative. Once can use the forward and backward-difference formulae, but the truncation error is larger.
It would therefore be useful if we could obtain a different formula for the endpoints, that has a smaller truncation error of $O(h^2)$.

We derive the **three-point** forward and backward-difference formulae for the endpoints.

Letting the endpoint be x_i then:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(\zeta_1)}{3!}h^3 \qquad \textbf{(8.21)}$$

$$f(x_{i+2}) = f(x_i) + f'(x_i)2h + \frac{f''(x_i)}{2!}(2h)^2 + \frac{f'''(\zeta_2)}{3!}(2h)^3 \qquad \textbf{(8.22)}$$

where $\zeta_1 \in]x_i, x_{i+1}[$ and $\zeta_2 \in]x_i, x_{i+2}[$

Multiplying equation (8.21) by 4, and subtracting it from (8.22) gives:

$$4f(x_{i+1}) - f(x_{i+2}) = 3f(x_i) + 2f'(x_i)h + \frac{4f'''(\zeta_1)}{3!}h^3 - \frac{f'''(\zeta_2)}{3!}h^3$$

Solving for $f'(x_i)$ gives:

$$f'(x_i) = \frac{-3f(x_i)+4f(x_{i+1})-f(x_{i+2})}{2h} + O(h^2)$$

This is our desired three-point forward-difference formula for use at one of the endpoints with a truncation error of $O(h^2)$.

In likewise fashion, a three-point backward-difference formula can be derived for use at the other endpoint:

$$f'(x_i) = \frac{f(x_{i-2})-4f(x_{i-1})+3f(x_i)}{2h} + O(h^2)$$

Example 8-3: Comparing numerical and analytical differentiation.

Consider the function $f(x) = x^3$. Calculate the first derivative at point $x = 3$ numerically with the three-point forward difference formula, using:

(*a*) Points $x = 3$, $x = 4$, and $x = 5$.

(*b*) Points $x = 3$, $x = 3.25$, and $x = 3.5$.

Compare the results with the exact value of the derivative, obtained analytically.

SOLUTION

***Analytical differentiation*:** The derivative of the function is $f'(x) = 3x^2$, and the value of the derivative at $x = 3$ is $f'(3) = 3 \cdot 3^2 = 27$.

Numerical differentiation

(*a*) The points used for numerical differentiation are:

x:	3	4	5
$f(x)$:	27	64	125

Using Eq. (8.24), the derivative using the three-point forward difference formula is:

$$f'(3) = \frac{-3f(3)+4f(4)-f(5)}{2 \cdot 1} = \frac{-3 \cdot 27 + 4 \cdot 64 - 125}{2} = 25 \qquad error = \left|\frac{25-27}{27}\right| \cdot 100 = 7.41 \ \%$$

(*b*) The points used for numerical differentiation are:

x:	3	3.25	3.5
$f(x)$:	27	3.25^3	3.5^3

Using Eq. (8.24), the derivative using the three points forward finite difference formula is:

$$f'(3) = \frac{-3f(3)+4f(3.25)-f(3.5)}{2 \cdot 0.25} = \frac{-3 \cdot 27 + 4 \cdot 3.25^3 - 3.5^3}{0.5} = 26.875$$

$$error = \left|\frac{26.875-27}{27}\right| \cdot 100 = 0.46 \ \%$$

The results show that the three-point forward difference formula gives a much more accurate value for the first derivative than the two-point forward finite difference formula in Example 8-1. For $h = 1$ the error reduces from 37.04% to 7.4%, and for $h = 0.25$ the error reduces from 8.57% to 0.46%.

The Three-Point Central-Difference Formula for the Second Derivative

We write:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f''(x_i)}{3!}h^3 + \frac{f^{(4)}(\zeta_1)}{4!}h^4$$

and

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f''(x_i)}{3!}h^3 + \frac{f^{(4)}(\zeta_2)}{4!}h^4$$

where $\zeta_1 \in]x_i, x_{i+1}[$ and $\zeta_2 \in]x_i, x_{i+2}[$

Adding these equations gives:

$$f(x_{i+1}) + f(x_{i-1}) = 2f(x_i) + 2\frac{f''(x_i)}{2!}h^2 + \frac{f^{(4)}(\zeta_1)}{4!}h^4 + \frac{f^{(4)}(\zeta_2)}{4!}h^4$$

Solving for $f''(x_i)$ gives:

$$f''(x_i) = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2} + O(h^2)$$

This is a three-point central-difference formula that provides an estimate of the **second derivative** at the point x_i in terms of the value of the function at that point x_i, the previous point x_{i-1} and then next point x_{i+1} with a truncation error of $O(h^2)$.

The same procedure can be used to obtain higher-order formulae involving more points.

The Three-Point Forward and Backward-Difference Formulae for the Second Derivative

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(\zeta_1)}{3!}h^3 \qquad \textbf{(8.21 again)}$$

$$f(x_{i+2}) = f(x_i) + f'(x_i)2h + \frac{f''(x_i)}{2!}(2h)^2 + \frac{f'''(\zeta_2)}{3!}(2h)^3 \qquad \textbf{(8.22 again)}$$

The three-point forward-difference formula for the second derivative can be obtained by multiplying equation (8.21) by 2, and then subtracting it from (8.22), which eliminates $f'(x_i)$.

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i+1}) + f(x_{i+2})}{h^2} + O(h)$$

Likewise, a three-point backward-difference formula for the second derivative can be obtained:

$$f''(x_i) = \frac{f(x_{i-2}) - 2f(x_{i-1}) + f(x_i)}{h^2} + O(h)$$

Example 8-4: Comparing numerical and analytical differentiation.

Consider the function $f(x)=\frac{2^x}{x}$. Calculate the second derivative at $x=2$ numerically with the three-point central difference formula using:

(*a*) Points $x=1.8$, $x=2$, and $x=2.2$.

(*b*) Points $x=1.9$, $x=2$, and $x=2.1$.

Compare the results with the exact (analytical) derivative.

SOLUTION

***Analytical differentiation*:** The second derivative of the function $f(x)=\frac{2^x}{x}$ is:

$$f''(x)=\frac{2^x[\ln(2)]^2}{x}-\frac{2\cdot 2^x\ln(2)}{x^2}+\frac{2\cdot 2^x}{x^3}$$

and the value of the derivative at $x=2$ is $f''(2)=0.574617$.

Numerical differentiation

(*a*) The numerical differentiation is done by substituting the values of the points $x=1.8$, $x=2$, and $x=2.2$ in Eq. (8.29). The operations are done with MATLAB, in the Command Window:

```
>> xa = [1.8 2 2.2];
>> ya = 2.^xa./xa;
>> df = (ya(1) - 2*ya(2) + ya(3))/0.2^2
df =
   0.57748177389232
```

(*b*) The numerical differentiation is done by substituting the values of the points $x=1.9$, $x=2$, and $x=2.1$ in Eq. (8.29). The operations are done with MATLAB, in the Command Window:

```
>> xb = [1.9 2 2.1];
>> yb = 2.^xb./xb;
>> dfb = (yb(1) - 2*yb(2) + yb(3))/0.1^2
dfb =
   0.57532441566441
```

Error in part (*a*): $error=\frac{0.577482-0.574617}{0.574617}\cdot 100=0.4986\ \%$

Error in part (*b*): $error=\frac{0.575324-0.574617}{0.574617}\cdot 100=0.1230\ \%$

The results show that the three-point central difference formula gives a quite accurate approximation for the value of the second derivative.

Differentiation Formulae using Lagrange Polynomials

Differentiation formulae can also be derived using Lagrange Polynomials.

For the first derivative, the two-point central, three-point forward and three-point backward-difference formulae are obtained by considering three points (x_i, y_i), (x_{i+1}, y_{i+1}) and (x_{i+2}, y_{i+2}).

The polynomial, in Lagrange form, that passes through these points is given by:

$$f(x) = \frac{(x-x_{i+1})(x-x_{i+2})}{(x_i-x_{i+1})(x_i-x_{i+2})}y_i + \frac{(x-x_i)(x-x_{i+2})}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}y_{i+1} + \frac{(x-x_i)(x-x_{i+1})}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}y_{i+2}$$

Differentiating this equation gives:

$$f'(x) = \frac{2x-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})}y_i + \frac{2x-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}y_{i+1} + \frac{2x-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}y_{i+2}$$

The first derivative at any one of the three points (x_i, y_i), (x_{i+1}, y_{i+1}) and (x_{i+2}, y_{i+2}) is obtained by substituting the corresponding value of x into this equation.

The following are the three-point forward-difference, two-point central-difference and three-point backward-difference formulae respectively:

$$f'(x_i) = \frac{2x-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})}y_i + \frac{x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}y_{i+1} + \frac{x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}y_{i+2}$$

$$f'(x_{i+1}) = \frac{x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})}y_i + \frac{2x_{i+1}-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}y_{i+1} + \frac{x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}y_{i+2}$$

$$f'(x_{i+2}) = \frac{x_{i+2}-x_{i+1}}{(x_i-x_{i+1})(x_i-x_{i+2})}y_i + \frac{x_{i+2}-x_i}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}y_{i+1} + \frac{2x_{i+2}-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}y_{i+2}$$

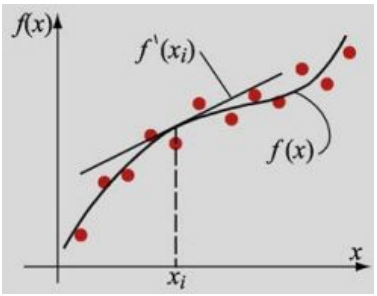
The advantage of these formulae is that points **do not have to be evenly-spaced**, however there is **no estimate for the magnitude of the error**.

Differentiation using Curve-Fitting

Differentiation may also be performed by finding an analytic function that approximates the data. The value of the derivative at a point can then be found by **differentiating analytically**.

This procedure is preferred if there is a lot of scatter in the data. Curve-fitting smoothes out the scatter, and so enhances the accuracy of the estimate of the derivative.

However, unlike using the finite-difference formulae using the Taylor series expansion, there is **no error estimate**.



Example 8-5: Using Richardson's extrapolation in differentiation.

Use Richardson's extrapolation with the results in Example 8-4 to calculate a more accurate approximation for the derivative of the function $f(x) = \frac{2^x}{x}$ at the point $x = 2$.

Compare the results with the exact (analytical) derivative.

SOLUTION

In Example 8-4 two approximations of the derivative of the function at $x = 2$ were calculated using the central difference formula in which the error is $O(h^2)$. In one approximation $h = 0.2$, and in the other $h = 0.1$. The results from Example 8-4 are:

for $h = 0.2$, $f''(2) = 0.577482$. The error in this approximation is 0.5016 %.

for $h = 0.1$, $f''(2) = 0.575324$. The error in this approximation is 0.126 %.

Richardson's extrapolation can be used by substituting these results in Eq. (8.45) (or Eq. (8.53)):

$$D = \frac{1}{3}\left(4D\left(\frac{h}{2}\right) - D(h)\right) + O(h^4) = \frac{1}{3}(4 \cdot 0.575324 - 0.577481) = 0.574605$$

The error now is $error = \frac{0.574605 - 0.5746}{0.5746} \cdot 100 = 0.00087$ %

This result shows that a much more accurate approximation is obtained by using Richardson's extrapolation.

We have seen with finite-difference formulae derived using the Taylor series expansion that truncation errors of various orders of the spacing h were obtained. Since for a discrete set of data points h is fixed, it is not possible to reduce the error by reducing h. We can however, use a difference formula with higher-order truncation error to get a greater accuracy.

On the other hand, if the function to be differentiated is continuous, then we can reduce h to get greater accuracy. However, there is also a round-off error resulting from the finite precision of the computer being used. This means that even though h can be made vanishingly small, the round-off error remains or can even grow as h is made smaller and smaller.

Example 8-6: Comparing numerical and analytical differentiation.

Consider the function $f(x) = e^x$. Write an expression for the first derivative of the function at $x = 0$ using the two-point central difference formula in Eq. (8.20). Investigate the effect that the spacing, h, between the points has on the truncation and round-off errors.

SOLUTION

The two-point central difference formula in Eq. (8.20) is:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - 2\,\frac{f'''(\xi)}{3!}h^2$$

where ξ is a value of x between x_{i-1} and x_{i+1}

The points used for calculating the derivative of $f(x) = e^x$ at $x = 0$ are $x_{i-1} = -h$ and $x_{i+1} = h$. Substituting these points in the formula gives:

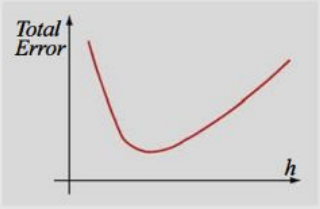
$$f'(0) = \frac{e^h - e^{-h}}{2h} - 2\,\frac{f'''(\xi)}{3!}h^2 \qquad (8.55)$$

When the computer calculates the values of e^h and e^{-h}, a round-off error is introduced, since the computer has finite precision. Consequently, the terms e^h and e^{-h} in Eq. (8.55) are replaced by $e^h + R_1$ and $e^{-h} + R_2$ where now e^h and e^{-h} are the exact values, and R_1 and R_2 are the round-off errors:

$$f'(0) = \frac{e^h + R_1 - e^{-h} - R_2}{2h} - 2\,\frac{f'''(\xi)}{3!}h^2 = \frac{e^h - e^{-h}}{2h} + \frac{R_1 - R_2}{2h} - 2\,\frac{f'''(\xi)}{3!}h^2 \qquad (8.56)$$

In Eq. (8.56) the last term on the right-hand side is the truncation error. In this term, the value of $f'''(\xi)$ is not known, but it is bounded. This means that as h decreases the truncation error decreases.

The round-off error is $(R_1 - R_2)/(2h)$. As h decreases the round-off error increases. The total error is the sum of the truncation error and round-off error. Its behavior is shown schematically in the figure on the right. As h decreases, the total error initially decreases, but after a certain value (which depends on the precision of the computer used) the total error increases as h decreases further.



Numerical Partial Differentiation

The finite-difference formulae obtained earlier can also be used to obtain **partial derivatives**.

For a function $f(x,y)$, the partial derivatives with respect to x and y at that point (a,b) are:

$$\frac{\partial f(x,y)}{\partial x}\Big|_{x=a,\,y=b} = \lim_{x\to a}\frac{f(x,b)-f(a,b)}{x-a}$$

$$\frac{\partial f(x,y)}{\partial y}\Big|_{x=a,\,y=b} = \lim_{y\to b}\frac{f(a,y)-f(a,b)}{y-b}$$

This means that the finite-difference formulae for function of a single variable can be extended to calculating the partial derivatives with respect to a particular variable of functions of two or more independent variables, by holding the other variables constant.

An approximation for the partial derivative at the point (x_i,y_i) with the two-point **forward-difference** formula is:

$$\frac{\partial f(x,y)}{\partial x}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_{i+1},y_i)-f(x_i,y_i)}{h_x}$$

$$\frac{\partial f(x,y)}{\partial y}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_i,y_{i+1})-f(x_i,y_i)}{h_y}$$

In the same way, the two-point **backward-difference** formula is:

$$\frac{\partial f(x,y)}{\partial x}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_i,y_i)-f(x_{i-1},y_i)}{h_x} \qquad \frac{\partial f(x,y)}{\partial y}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_i,y_i)-f(x_i,y_{i-1})}{h_y}$$

and the two-point **central-difference** formula is:

$$\frac{\partial f(x,y)}{\partial x}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_{i+1},y_i)-f(x_{i-1},y_i)}{2h_x} \qquad \frac{\partial f(x,y)}{\partial y}\Big|_{x=x_i,\,y=y_i} = \lim_{x\to a}\frac{f(x_i,y_{i+1})-f(x_i,y_{i-1})}{2h_y}$$

The **second partial derivatives** with the **three-point central-difference** formulae are:

$$\frac{\partial^2 f}{\partial x^2}\Big|_{x=x_i,\,y=y_i} = \frac{f(x_{i-1},y_i)-2f(x_i,y_i)+f(x_{i+1},y_i)}{h_x^2}$$

$$\frac{\partial^2 f}{\partial y^2}\Big|_{x=x_i,\,y=y_i} = \frac{f(x_i,y_{i-1})-2f(x_i,y_i)+f(x_i,y_{i+1})}{h_y^2}$$

The second-order partial derivative can also be mixed:

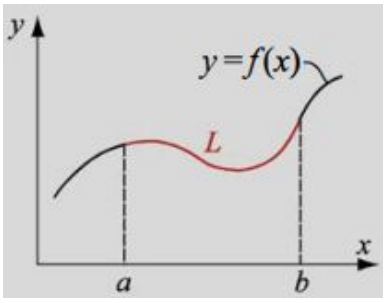
$$\frac{\delta^2 f}{\delta x \delta y} = \frac{\delta}{\delta y}\left(\frac{\delta f}{\delta x}\right) == \frac{\delta}{\delta x}\left(\frac{\delta f}{\delta y}\right)$$

A finite-difference formula for the **mixed derivative** can be obtained by using the **first-order** finite-difference formulae for **partial derivatives**.

09 Numerical Integration

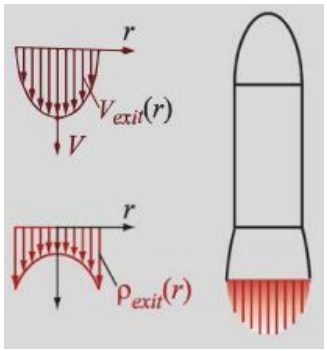
The length of a curve between points a and b is given by:

$$L = \int_a^b \sqrt{1 + (f'(x))^2}\, dx$$



Another example is in calculating the thrust of a rocket which is expressed as a function of the exit-velocity of the exhaust and the density of the exhaust over the exhaust aperture.

$$T = \int_0^R 2\pi\rho(r) V_{exit}^2(r) r\, dr$$



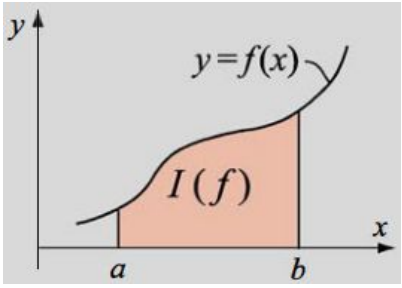
The general form of a **definite integral** (the antiderivative) is:

$$I(f) = \int_a^b f(x)\, dx \qquad \textbf{(9.1)}$$

where:

- $f(x)$, called the **integrand**, is a function of the independent variable x
- a and b are the **limits of integration**

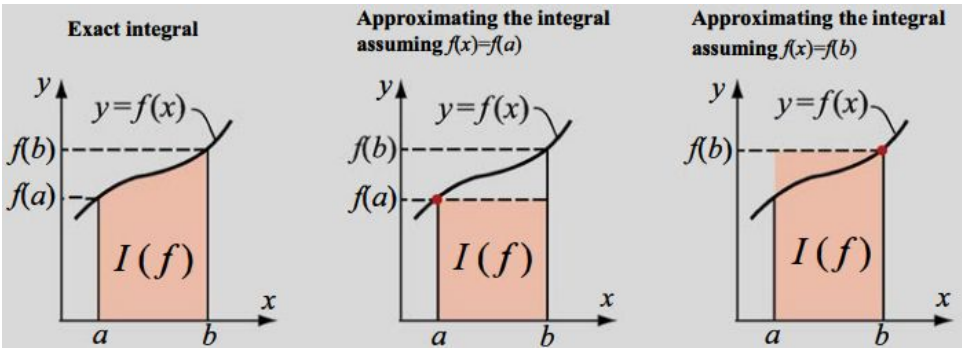
Graphically, the value of the integral corresponds to the **area under the curve** between a and b when we integrate over the x-axis.



The integrand is an analytical function or a set of discrete points (tabulated data). Numerical integration is needed when the integrand is discrete and / or where analytical integration is difficult, numerous or impossible.

The Rectangle Method

The simplest approximation for $\int_a^b f(x)dx$ is to take $f(x)$ over the interval $x \in [a,b]$ as a **constant** equal to the value of $f(x)$ at either one of the endpoints.

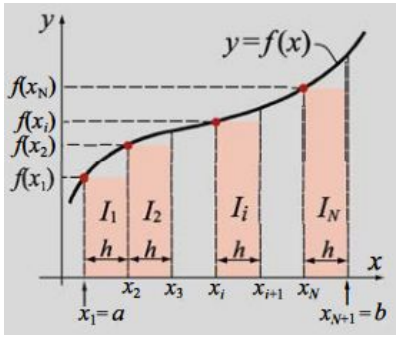


The integral is then:

$$I(f) = \int_a^b f(a)dx = f(a)(b-a) \qquad \text{or} \qquad I(f) = \int_a^b f(b)dx = f(b)(b-a)$$

A more accurate method is to use the **composite rectangle method**:

- The domain $[a,b]$ is divided into N subintervals
- The integrals over each of the subintervals are evaluated using the rectangle method
- The results are **summed** to get the integral over the **complete domain**



Then:

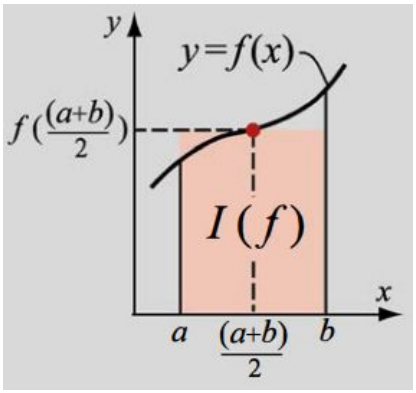
$$I(f) = \int_a^b f(x)dx \approx f(x_1)(x_2-x_1) + f(x_2)(x_3-x_2) + \ldots + f(x_n)(x_{n+1}-x_n) = \sum_{i=1}^{n} \left[f(x_i)(x_{i+1}-x_i)\right]$$

or when the subintervals have the **same width** h:

$$I(f) = \int_a^b f(x)dx \approx h \sum_{i=1}^{n} f(x_i)$$

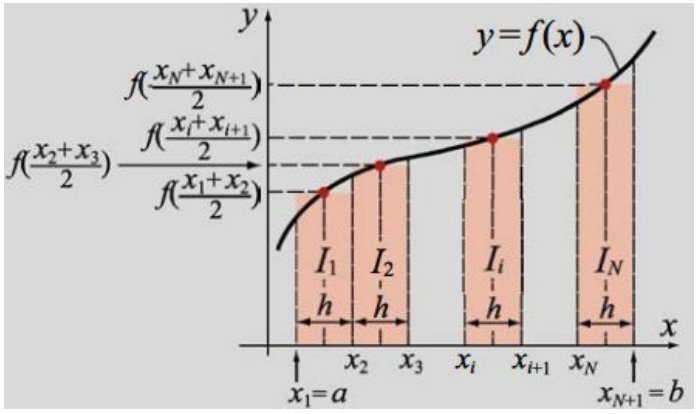
The Midpoint Method

An **improvement** to the rectangle method is the midpoint method.
Instead of approximating the integrand by the values of the function at $x = a$ or $x = b$, the value of the integrand at the **middle** of the interval $f(\frac{a+b}{2})$ is used:



$$I(f) = \int_a^b f(x)dx \approx \int_a^b f(\tfrac{a+b}{2})dx = f(\tfrac{a+b}{2})(b-a)$$

Again, this can be improved upon by dividing the domain into N subintervals and performing a **composite integration**:



Mathematically, this is:

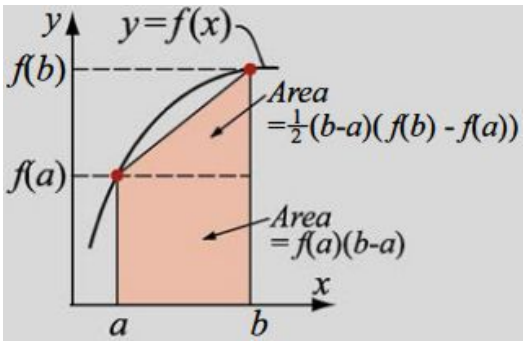
$$I(f) = \int_a^b f(x)dx \approx f(\tfrac{x_1+x_2}{2})(x_2-x_1) + f(\tfrac{x_2+x_3}{2})(x_3-x_2) + \ldots + f(\tfrac{x_n+x_{n+1}}{2})(x_{n+1}-x_n) = \sum_{i=1}^{n} \left[f(\tfrac{x_i+x_{i+1}}{2})(x_{i+1}-x_i) \right]$$

If each of the subintervals have the same length h we can write:

$$I(f) = \int_a^b f(x)dx \approx h \sum_{i=1}^{n} f(\tfrac{x_i+x_{i+1}}{2})$$

The Trapezoidal Method

A refinement over the simple rectangle and midpoint method is to use a **linear function** to approximate the integrand over the interval of integration.



$$I(f) = \int_a^b f(x)dx \qquad \textbf{(9.1 again)}$$

The equation of a line connecting two points with x-ordinates a and b is:

$$f(x) \approx f(a) + (x-a)f[a,b] = f(a) + (x-a)\frac{[f(b)-f(a)]}{b-a} \qquad \textbf{(9.8)}$$

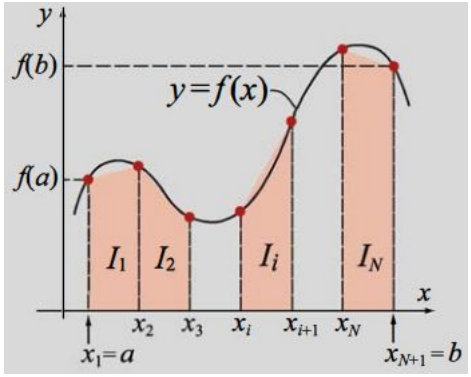
Substituting equation (9.8) into (9.1) and integrating analytically gives:

$$I(f) = \int_a^b \left(f(a) + (x-a)\frac{[f(b)-f(a)]}{b-a} \right) dx$$
$$= f(a)(b-a) + \tfrac{1}{2}[f(b)-f(a)](b-a)$$

Simplifying the result gives the Trapezoidal rule:

$$I(f) \approx \frac{[f(b)-f(a)]}{2}(b-a)$$

Again, a more accurate result can be performed by composite integration yielding the **composite trapezoidal method**.



Mathematically:

$$I(f) = \int_a^b f(x)dx = \int_{x_1=a}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx + \ldots + \int_{x_n}^{x_{n+1}} f(x)dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx$$

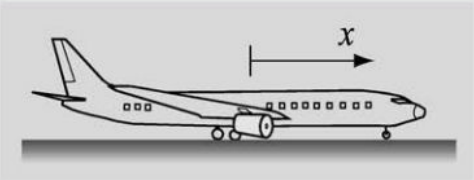
Applying the Trapezoidal method to each subinterval yields:

$$I(f) = \int_a^b f(x)dx \approx \frac{1}{2} \sum_{i=1}^{n} \left[f(x_i) + f(x_{i+1}) \right] (x_{i+1} - x_i)$$

With equally-sized subintervals this can be reduced to:

$$I(f) \approx \frac{h}{2} \left[f(a) + f(b) \right] + h \sum_{i=1}^{n} f(x_i)$$

Example 9-1: Distance traveled by a decelerating airplane.



A Boeing 737-200 airplane of mass $m = 97000$ kg lands at a speed of 93 m/s (about 181 knots) and applies its thrust reversers at $t = 0$. The force F that is applied to the airplane, as it decelerates, is given by $F = -5v^2 - 570000$, where v is the airplane's velocity. Using Newton's second law of motion and flow dynamics, the relationship between the velocity and the position x of the airplane can be written as:

$$mv\frac{dv}{dx} = -5v^2 - 570000$$

where x is the distance measured from the location of the jet at $t = 0$.

Determine how far the airplane travels before its speed is reduced to 40 m/s (about 78 knots) by using the composite trapezoidal method to evaluate the integral resulting from the governing differential equation.

SOLUTION

Even though the governing equation is an ODE, it can be expressed as an integral in this case. This is done by separating the variables such that the speed v appears on one side of the equation and x appears on the other.

$$\frac{97000vdv}{(-5v^2 - 570000)} = dx$$

Next, both sides are integrated. For x the limits of integration are from 0 to an arbitrary location x, and for v the limits are from 93 m/s to 40 m/s.

$$\int_0^x dx = -\int_{93}^{40} \frac{97000v}{(5v^2 + 570000)}dv = \int_{40}^{93} \frac{97000v}{(5v^2 + 570000)}dv \qquad (9.14)$$

The objective of this example is to show how the definite integral on the right-hand side of the equation can be determined numerically using the composite trapezoidal method. In this problem, however, the integration can also be carried out analytically. For comparison, the integration is done both ways.

Analytical Integration

The integration can be carried out analytically by using substitution. By substituting $z = 5u^2 + 570000$, the integration can be performed to obtain the value $x = 574.1494$ m.

Numerical Integration

To carry out the numerical integration, the following user-defined function, named `trapezoidal`, is created.

Program 9-1: Function file, integration trapezoidal method.

```
function I = trapezoidal(Fun,a,b,N)
% trapezoidal numerically integrate using the composite trapezoidal method.
% Input Variables:
% Fun  Name for the function to be integrated.
% (Fun is assumed to be written with element-by-element calculations.)
% a  Lower limit of integration.
% b  Upper limit of integration.
% N  Number of subintervals.
% Output Variable:
% I  Value of the integral.

h = (b-a)/N;
x = a:h:b;
F = Fun(x);

I=h*(F(1)+F(N+1))/2+h*sum(F(2:N));
```

Calculate the width h of the subintervals.

Create a vector x with the coordinates of the subintervals.

Create a vector F with the values of the integrand at each point x.

Calculate the value of the integral according to Eq. (9.13).

The function `trapezoidal` is used next in the Command Window to determine the value of the integral in Eq. (9.14). To examine the effect of the number of subintervals on the result, the function is used three times using $N = 10, 100$, and 1000. The display in the Command Window is:

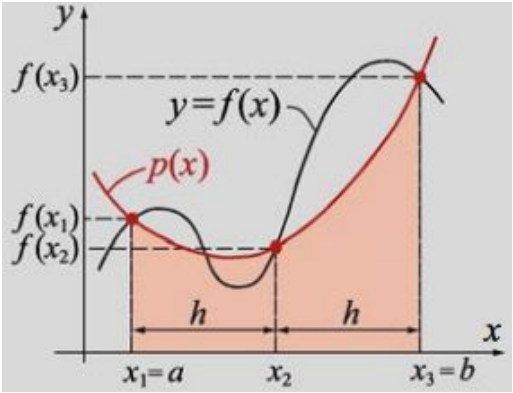
```
>> format long g
>> Vel = @ (v) 97000*v./(5*v.^2+570000);
>> distance = trapezoidal(Vel,40,93,10)
distance =
        574.085485133712
>> distance = trapezoidal(Vel,40,93,100)
distance =
        574.148773931409
>> distance = trapezoidal(Vel,40,93,1000)
distance =
        574.149406775129
```

Define an anonymous function for the integrand. Note element-by-element calculations.

As expected, the results show that the integral is evaluated more accurately as the number of subintervals is increased. When $N = 1000$, the answer is the same as that calculated analytically to four decimal places.

Simpson's Methods

With Simpson's methods, the integrand is approximated by a **quadratic or cubic polynomial**.



Simpson's 1/3 Method

In this method, a quadratic polynomial is used to approximate the integrand.
Three points from the domain $[a, b]$ are used.
These are the endpoints $x_1 = a$, $x_3 = b$ and midpoint $x_2 = \frac{a+b}{2}$.

The polynomial can be expressed in Newton's form (it could be expressed in standard or Lagrange form):

$$p(x) = \alpha + \beta(x - x_1) + \gamma(x - x_1)(x - x_2) \qquad \textbf{(9.15)}$$

where α, β and γ are unknown constants evaluated by applying the conditions that the polynomial passes through the points $p(x_1) = f(x_1)$, $p(x_2) = f(x_2)$ and $p(x_3) = f(x_3)$.

These conditions yield:

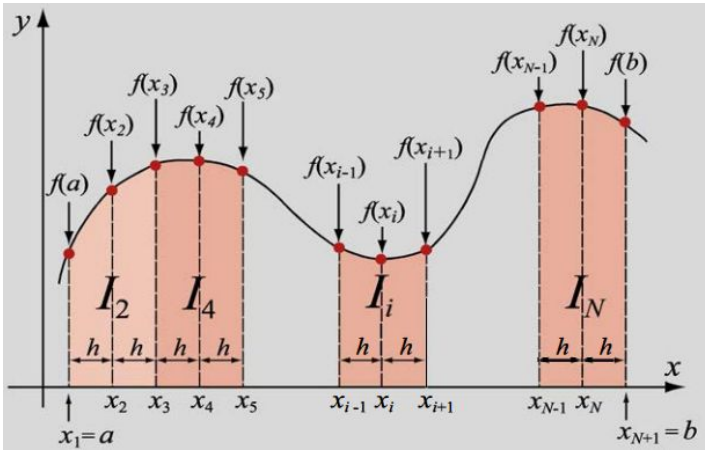
$$\alpha = f(x_1)$$
$$\beta = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$
$$\gamma = \frac{f(x_3) - f(x_2) + f(x_1)}{2h^2}$$

where $h = \frac{b-a}{2}$.

Substituting these values for constants back into equation (9.15) and integrating $p(x)$ over the interval $[a, b]$ gives:

$$I = \int_{x_1}^{x_3} f(x)dx \approx \int_{x_1}^{x_3} p(x)dx = \frac{h}{3}\left[f(x_1) + 4f(x_2) + f(x_3)\right] = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \qquad \textbf{(9.16)}$$

Simpson's methods can be extended to a **composite form** by dividing the interval $[a, b]$ into N subintervals of length h.



Since Simpson's 1/3 method is applied to two subintervals at a time, it is necessary that the total number of subintervals is **even**. The integral over the whole interval is then written:

$$I(f) = \int_a^b f(x)dx = \int_{x_1=a}^{x_3} f(x)dx + \int_{x_3}^{x_5} f(x)dx + \ldots + \int_{x_{N-1}}^{x_{N+1}=b} f(x)dx = \sum_{i=2,4,6}^{N} \int_{x_{i-1}}^{x_{i+1}} f(x)dx \quad \textbf{(9.17)}$$

Rewriting equation (9.16) for two adjacent subintervals of equal length gives:

$$I_i(f) = \int_{x_{i-1}}^{x_{i+1}} f(x)dx \approx \frac{h}{3}\left[f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right] \quad \textbf{(9.18)}$$

where $h = x_{i+1} - x_i = x_i - x_{i-1}$

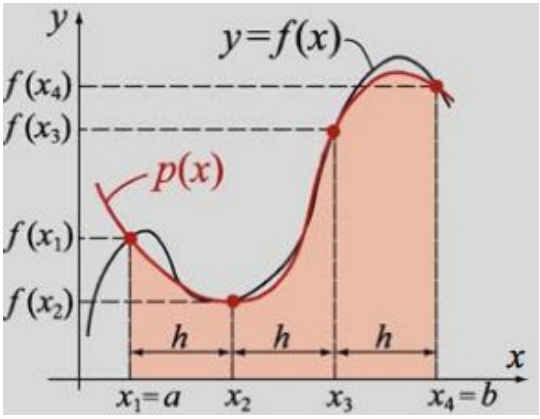
Substituting equation (9.18) into (9.17) for each of the integrals and collecting similar terms gives the **composite Simpson's 1/3 formula**:

$$I(f) \approx \frac{h}{3}\left[f(a) + 4\sum_{i=2,4,6}^{N} f(x_i) + 2\sum_{j=3,5,7}^{N-1} f(x_j) + f(b)\right]$$

where $h = \frac{b-a}{N}$

Simpson's 3/8 Method

In this method, a **cubic polynomial** is used to approximate the integrand.
A third-order polynomial can be determined using **four points**.
These are the endpoints $x_1 = a$, $x_4 = b$ and the two points x_2 and x_3 that divide the interval into three equal sections.

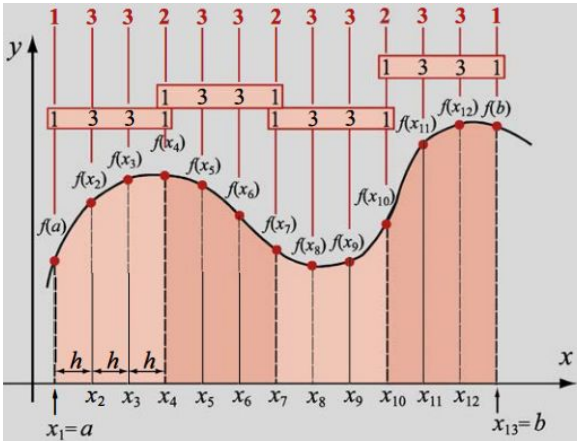


The polynomial can be written in the form:
$p(x) = c_3x^3 + c_2x^2 + c_1x + c_0$
where c_0 and c_3 are determined from the conditions that the polynomial passes through the points $p(x_1) = f(x_1),\ ...,\ p(x_4) = f(x_4)$

Once these constants are determined, the polynomial can be easily integrated to give:

$$I = \int_a^b f(x)dx \approx \int_a^b p(x)dx = \tfrac{3}{8}h\left[f(a) + 3f(x_2) + 3f(x_3) + f(b)\right] \qquad \textbf{(9.20)}$$

A **composite form** of Simpson's 3/8 method can be achieved by dividing the interval $[a, b]$ into N subintervals. In general, these can have arbitrary width but we concern ourselves with **equal spacings** here.



The integration in each group of three intervals is done by applying equation (9.20).
The integral over the whole domain is obtained by summing the integrals over the subintervals.

In general (where N is divisible by 3):

$$I(f) \approx \frac{3h}{8}\left[f(a) + 3\sum_{i=2,5,8}^{N-1}\left[f(x_i) + f(x_{i+1})\right] + 2\sum_{j=4,7,10}^{N-2} f(x_j) + f(b)\right]$$

Gauss Quadrature

The general form of Gauss quadrature is a weighted sum:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} C_i f(x_i) \qquad \textbf{(9.23)}$$

The coefficients C_i are the weights and x_i are specific points in the interval known as **Gauss points**.

We obtain weights C_i and abscissae x_i that make equation (9.23) exact for polynomials $x^0, x^1, x^2, \ldots$. The hope is that because these will yield exact results for polynomials, they can then be used for other functions that are well-approximated by polynomials.

Beginning with the domain $[-1, 1]$ the form of Gauss quadrature is:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} C_i f(x_i) \qquad \textbf{(9.24)}$$

We now force equation (9.24) to be exact for the cases where $f(x) = 1, x, x^2, x^3, \ldots$
The number of cases that have to be considered depends on n.
For example, when $n = 2$ we need to determine four values:

$$\int_{-1}^{1} f(x)dx \approx C_1 f(x_1) + C_2 f(x_2) \qquad \textbf{(9.25)}$$

The four constants C_1, C_2, x_1 and x_2 are determined by enforcing equation (9.25) to be exact when applied to the following four cases:

1. $f(x) = 1$ $\quad \int_{-1}^{1} (1)dx \approx 2 = C_1 + C_2$
2. $f(x) = x$ $\quad \int_{-1}^{1} x dx \approx 0 = C_1 x_1 + C_2 x_2$
3. $f(x) = x^2$ $\quad \int_{-1}^{1} x^2 dx \approx \frac{2}{3} = C_1 x_1^2 + C_2 x_2^2$
4. $f(x) = x^3$ $\quad \int_{-1}^{1} x^3 dx \approx 0 = C_1 x_1^3 + C_2 x_2^3$

This is a system of four equations with four unknowns so we can solve for C_1, C_2, x_1 and x_2. However, since the equations are nonlinear, multiple solutions can exist.

A particular solution can be obtained by imposing the condition that x_1 and x_2 are symmetrically located about $x = 0$ i.e. $x_1 = -x_2$. This requirement implies that $C_1 = C_2$.
Solving gives:

$C_1 = C_2 = 1$

$x_1 = -\frac{1}{\sqrt{3}}$

$x_2 = \frac{1}{\sqrt{3}}$

$$\int_{-1}^{1} f(x)dx \approx C_1 f(x_1) + C_2 f(x_2) \quad \textbf{(9.25 again)}$$

Substituting back into equation (9.25) gives (for $n = 2$):

$$\int_{-1}^{1} f(x)dx \approx f(-\tfrac{1}{\sqrt{3}}) + f(\tfrac{1}{\sqrt{3}})$$

This equation is exact for $f(x) = 1$, $f(x) = x$, $f(x) = x^2$ and $f(x) = x^3$.

Example:

Let $f(x) = \cos x$

Exact solution:

$$\int_{-1}^{1} \cos x\, dx = \sin x \Big|_{-1}^{1} = \sin(1) - \sin(-1) = 1.6829417$$

The approximate value using Gauss quadrature is:

$\cos(-\frac{1}{\sqrt{3}}) + \cos(\frac{1}{\sqrt{3}}) = 1.67582366$

The error is within 4.5% of the exact solution.

The accuracy of Gauss quadrature is increased by using a higher value of n.
For $n = 3$, the equation has the form:

$$\int_{-1}^{1} f(x)dx \approx C_1 f(x_1) + C_2 f(x_2) + C_3 f(x_3) \qquad \textbf{(9.28)}$$

In this case, six values have to be determined: C_1, C_2, C_3 and x_1, x_2, x_3.
The values are determined by enforcing equation (9.28) to be exact when
$f(x) = 1,\ f(x) = x,\ f(x) = x^2,\ \ldots,\ f(x) = x^5$

This results in a set of six equations with six unknowns. The values that are determined are given in the following table for this and a number of points up to $n = 6$.

The general formula to use in conjunction with the table for Gauss quadrature is:

$$\int_{-1}^{1} f(x)dx \approx C_1 f(x_1) + C_2 f(x_2) + \ldots + C_n f(x_n)$$

n (Number of points)	Coefficients C_i (weights)	Gauss points x_i
2	$C_1 = 1$ $C_2 = 1$	$x_1 = -0.57735027$ $x_2 = 0.57735027$
3	$C_1 = 0.5555556$ $C_2 = 0.8888889$ $C_3 = 0.5555556$	$x_1 = -0.77459667$ $x_2 = 0$ $x_3 = 0.77459667$
4	$C_1 = 0.3478548$ $C_2 = 0.6521452$ $C_3 = 0.6521452$ $C_4 = 0.3478548$	$x_1 = -0.86113631$ $x_2 = -0.33998104$ $x_3 = 0.33998104$ $x_4 = 0.86113631$
5	$C_1 = 0.2369269$ $C_2 = 0.4786287$ $C_3 = 0.5688889$ $C_4 = 0.4786287$ $C_5 = 0.2369269$	$x_1 = -0.90617985$ $x_2 = -0.53846931$ $x_3 = 0$ $x_4 = 0.53846931$ $x_5 = 0.90617985$
6	$C_1 = 0.1713245$ $C_2 = 0.3607616$ $C_3 = 0.4679139$ $C_4 = 0.4679139$ $C_5 = 0.3607616$ $C_6 = 0.1713245$	$x_1 = -0.93246951$ $x_2 = -0.66120938$ $x_3 = -0.23861919$ $x_4 = 0.23861919$ $x_5 = 0.66120938$ $x_6 = 0.93246951$

Using the same example as previous:

The approximate value using three-point Gauss quadrature is:

$$0.5555556cos(-0.77459667) + 0.8888889cos(0) + 0.5555556cos(0.77459667) = 1.68285982$$

This is almost an exact result.

When the domain of integration is not $[-1, 1]$, then a change of variables is necessary.

In other words, the integral $\int_a^b f(x)dx \rightarrow \int_{-1}^{1} g(t)dt$.

The change in variable is:

$x = \frac{1}{2}[t(b-a) + a + b]$ and $dx = \frac{1}{2}(b-a)dt$

The Gauss quadrature is then performed using the transformed integral.

Example 9-2: Evaluation of a single definite integral using fourth-order Gauss quadrature.

Evaluate $\int_0^3 e^{-x^2} dx$ using four-point Gauss quadrature.

SOLUTION

Step 1: Since the limits of integration are $[0, 3]$, the integral has to be transformed to the form $\int_{-1}^{1} f(t)dt$. In the present problem $a = 0$ and $b = 3$. Substituting these values in Eq. (9.31) gives:

$$x = \frac{1}{2}[t(b-a) + a + b] = \frac{1}{2}[t(3-0) + 0 + 3] = \frac{3}{2}(t+1) \quad \text{and} \quad dx = \frac{1}{2}(b-a)dt = \frac{1}{2}(3-0)dt = \frac{3}{2}dt$$

Substituting these values in the integral gives:

$$I = \int_0^3 e^{-x^2} dx = \int_{-1}^{1} f(t)dt = \int_{-1}^{1} \frac{3}{2} e^{-\left[\frac{3}{2}(t+1)\right]^2} dt$$

Step 2: Use four-point Gauss quadrature to evaluate the integral. From Eq. (9.30), and using Table 9-1:

$$I = \int_{-1}^{1} f(t)dt \approx C_1 f(t_1) + C_2 f(t_2) + C_3 f(t_3) + C_4 f(t_4) = 0.3478548 \cdot f(-0.86113631)$$
$$+ 0.6521452 \cdot f(-0.33998104) + 0.6521452 \cdot f(0.33998104) + 0.3478548 \cdot f(0.86113631)$$

Evaluating $f(t) = \frac{3}{2} e^{-\left[\frac{3}{2}(t+1)\right]^2}$ gives:

$$I = 0.3478548 \frac{3}{2} e^{-\left[\frac{3}{2}((-0.86113631)+1)\right]^2} + 0.6521452 \frac{3}{2} e^{-\left[\frac{3}{2}((-0.33998104)+1)\right]^2}$$
$$+ 0.6521452 \frac{3}{2} e^{-\left[\frac{3}{2}(0.33998104+1)\right]^2} + 0.3478548 \frac{3}{2} e^{-\left[\frac{3}{2}(0.86113631+1)\right]^2} = 0.8841359$$

The exact value of the integral (when carried out analytically) is 0.8862073. The error is only about 1%.

Improper Integrals

So far we have considered integrals of well-behaved functions with finite limits. Situations arise however where functions are not well-behaved (discontinuous or singular) and / or have infinite limits.

Integrals with Singularities

If the integral $\int_a^b f(x)dx$ is singular at $x = c \in [a, b]$ then the integration can be performed over two intervals:

1. $[a, c]$
2. $[c, b]$

Mathematically, integrals that have a singularity at one of the endpoints may or may not have a finite value.

For example:

The function $\frac{1}{\sqrt{x}}$ is singular at $x = 0$, but the integral of the function over $[0, 2]$ equals 2.

That is: $\int_0^1 \frac{1}{\sqrt{x}}dx = 2$

An estimate of the integral can be obtained using any of the methods described here that do not use endpoints e.g. the composite midpoint method or Gaussian quadrature.
Otherwise, we can choose points very close to the endpoints that do not include the singularity, and then integrate over this domain.

On the other hand, $\frac{1}{\sqrt{x}}$ does not have a finite value.

Integrals with Unbounded Limits

Again, intervals with infinite limits can converge (have a finite value) or may diverge (have an infinite value). In the case of the former, the integration can be performed over a finite domain where the function has values that are **not** close to zero.

This is a typical situation where the integrand has a finite value over a relatively small domain, and close to zero everywhere else.

An example of this is the Gaussian PDF in statistics: $\int_{-\infty}^b \frac{1}{\sqrt{2\pi}} e^{(\frac{-x^2}{2})}dx$

The computation of the integral stops where the successive iterations yield only a small (tolerable) change in the result.