Chapter X

Numerical Integration

The numerical solution of the integral

$$A = \int_{x=a}^{b} f(x)dx$$

will be dealt with using two methods:

- Trapezoidal Rule.
- Simpson's 1/3 Rule.

10.1 Integration by Trapezoidal Rule

Since the result of integration is the area bounded by $f(x)$ and the x axis from $x=a$ to $x=b$ (see Fig.10.1) the problem can be approximated numerically by dividing the region into small segments each of width Δx. The area of each segment can then be approximated by a trapezoid by approximating each segment of the curve by a straight line. Thus

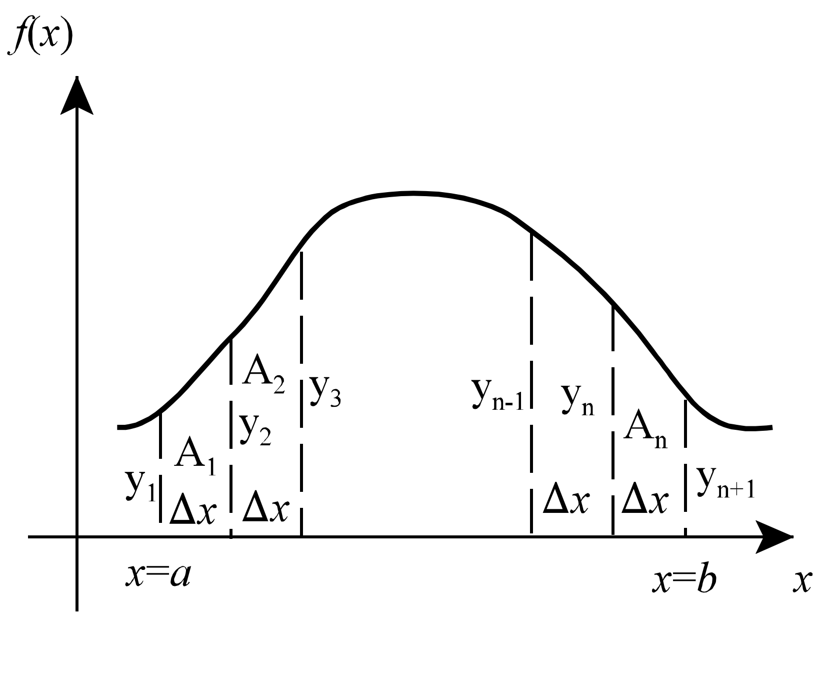

Figure 10.1 Numerical integration.

$$A_1 = \Delta x\left(\frac{y_1 + y_2}{2}\right)$$

$$A_2 = \Delta x\left(\frac{y_2 + y_3}{2}\right)$$

$$\vdots$$

$$A_n = \Delta x\left(\frac{y_n + y_{n+1}}{2}\right)$$

Hence:

$$A = \int_a^b f(x)dx \cong A_1 + A_2 + \cdots + A_n$$

$$= \frac{\Delta x}{2}(y_1 + 2y_2 + 2y_3 + \cdots + 2y_n + y_{n+1})$$

or

$$A \cong \frac{\Delta x}{2}\left(f(a) + f(b) + 2\sum_{i=2}^{n} f(x_i)\right)$$

where $x_i = x_{i-1} + \Delta x$; $x_1 = a$

A C++ program for numerical integration using the Trapezoidal rule follows.

```
#include <iostream.h>
#include <conio.h>
#include <math.h>

class Integration {
   private:
      double A; //area under the curve
```

```
      double xmin, xmax; //limits of integration
      int numberOfPoints;
      double (*f)(double x);  //function to be integrated
   public:
      Integration( double (*F)(double x), double a, double b, int n)
         {
         f=F;
         xmin=a;
         xmax=b;
         numberOfPoints=n;
         }

      double Trapezoidal();
        };

double Integration::Trapezoidal()
  {
 double dx=(xmax-xmin)/double(numberOfPoints);
 double sum=0.0;
 double x=xmin+dx;

 for(int i=1; i< numberOfPoints; i++)
   {
   sum+=(*f)(x);
   x+=dx;
   }
 A=(*f)(xmin)+(*f)(xmax)+2.0*sum;
 A*=dx/double(2.0);
 return A;
   }

 double FUN(double x)  //User supplied function
   {
   return x*sqrt(8.0-x*x*x);
   }

//----------------------------------------------------------------------

int main()
{
Integration I(FUN,0,2,8);
cout << I.Trapezoidal() << endl;
```

```
getch();
 return 1;
  }
```

10.2 Simpson's 1/3 Rule

With Simpson's approach a quadratic is used to approximate two segments of the curve instead of using straight line segments.

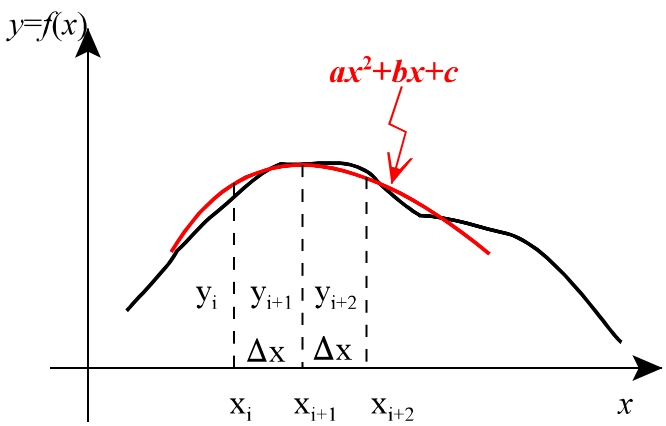

Figure 10.2 Simpson's Integration method

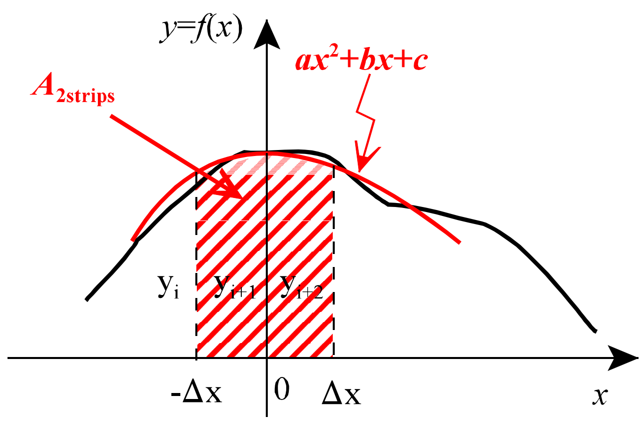

Figure 10.3 Centering the two-strips around the origin.

Figures 10.2 and 10.3 shows two strips under the curve bounded by a quadratic and the *x*-axis. In Fig. 10.2 the two strips are centered about the x-axis. This tends to simplify the derivation of the area, $A_{2\text{-strips}}$, under the quadratic and of width $2\Delta x$.

From Fig. 10.3

$$
\begin{aligned}
A_{2-strips} &= \int_{-\Delta x}^{\Delta x} (ax^2 + bx + c)dx \\
&= \left[\frac{ax^3}{3} + \frac{bx^2}{2} + cx \right]_{-\Delta x}^{\Delta x} \\
&= 2a\frac{(\Delta x)^3}{3} + 2c\Delta x
\end{aligned}
$$

The constants *a* and *c* can be determined from the fact that the points $(-\Delta x, y_i)$, $(0, y_{i+1})$, $(\Delta x, y_{i+2})$ lie on the curve. Hence,

$$
\begin{aligned}
y_i &= a(-\Delta x)^2 + b(-\Delta x) + c \\
y_{i+1} &= c \\
y_{i+2} &= a(\Delta x)^2 + b(\Delta x) + c
\end{aligned}
$$

Solving the above three equations for *a* and *c* we get

$$
\begin{aligned}
a &= \frac{y_i - 2y_{i+1} + y_{i+2}}{2\Delta x} \\
c &= y_{i+1}
\end{aligned}
$$

Substituting these two equations in the $A_{2\text{-strips}}$ equation derived above we get

$$
A_{2-strips} = \frac{\Delta x}{3}(y_i + 4y_{i+1} + y_{i+2})
$$

For the general case of *n* equal width strips (**n** is even) we can write (see Fig.10.4):

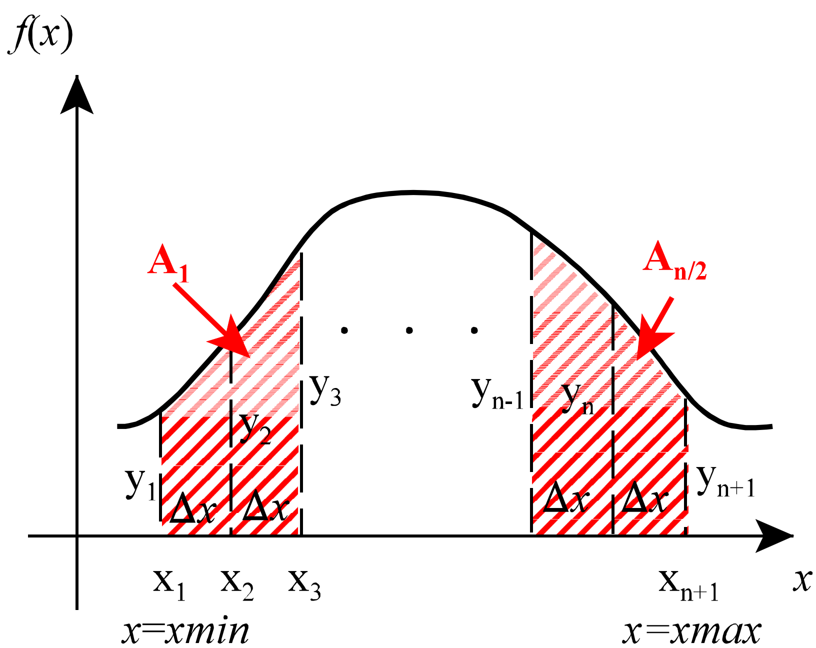

Figure 10.4 Numerical integration.

$$\Delta x = \frac{x_{\max} - x_{\min}}{n}$$

and

$$A_1 = \frac{\Delta x}{3}(y_1 + 4y_2 + y_3)$$

$$A_2 = \frac{\Delta x}{3}(y_3 + 4y_4 + y_5)$$

$$\vdots$$

$$A_{\frac{n}{2}} = \frac{\Delta x}{3}(y_{n-1} + 4y_{n-1} + y_{n+1})$$

or we can write:

$$\int_{x_{\min}}^{x_{\max}} f(x)dx \cong \frac{\Delta x}{3}\left(y_1 + 4y_2 + 2y_3 + 4y_4 + 2y_5 + \ldots + 2y_{n-1} + 4y_n + y_{n+1}\right)$$

$$= \frac{\Delta x}{3}\left(f(x_{\min}) + f(x_{\max}) + 4\sum_{\substack{i=2,4,\ldots \\ n-even}}^{n} y_i + \sum_{\substack{i=3,5,\ldots \\ n-odd}}^{n-1} y_i \right)$$

where $y_i = f\left(x_{\min} + (i-1)\Delta x\right)$

We can now add to the class Integration developed after the Trapezoid rule the Simpson's 1/3 rule as follows:

```
#include <iostream.h>
#include <conio.h>
#include <math.h>

class Integration {
   private:
      double A; //area under the curve
      double xmin, xmax; //limits of integration
      int numberOfPoints;
      double (*f)(double x);  //function to be integrated
   public:
      Integration( double (*F)(double x), double a, double b, int n)
         {
         f=F;
         xmin=a;
         xmax=b;
         numberOfPoints=n;
         }

      double Trapezoidal();
      double Simpson();
        };

double Integration::Trapezoidal()
  {
 double dx=(xmax-xmin)/double(numberOfPoints);
```

```
 double sum=0.0;
 double x=xmin+dx;

 for(int i=1; i< numberOfPoints; i++)
   {
   sum+=(*f)(x);
   x+=dx;
   }
 A=(*f)(xmin)+(*f)(xmax)+2.0*sum;
 A*=dx/double(2.0);
 return A;
   }
```

double Integration::Simpson()

```
  {
 double dx=(xmax-xmin)/double(numberOfPoints);
 double sum1=0, sum2=0.0;
 double x=xmin+dx;

 for(int i=1; i< numberOfPoints; i++)
   {
  if(i%2!=0)
   sum1+=(*f)(x);
  else
   sum2+=(*f)(x);
  x+=dx;
   }
 A=(*f)(xmin)+(*f)(xmax)+4.0*sum1+2.0*sum2;
 A*=dx/double(3.0);
 return A;
   }
```

```
 double FUN(double x)  //User supplied function
   {
   return x*sqrt(8.0-x*x*x);
   }

//------------------------------------------------------------------------

int main()
{
Integration I(FUN,0,2,8);
cout << I.Trapezoidal() << endl;
```

```
cout << I.Simpson() << endl;
getch();
 return 1;
   }
```

Example. Use the trapezoidal and Simpson's rule with 8 strips to calculate the integral.

$$I = \int_0^2 t\sqrt{8 - t^3}\,dt$$

Solution.

$$\Delta t = \frac{2 - 0}{8} = 0.25$$

t	0	0.25	0.5	0.75	1	1.25	1.50	1.75	2
y_i (i=1,2,...,9)	0	0.706	1.403	2.065	2.646	3.0738	3.226	2.844	0

Trapezoidal rule:

$$A = \frac{\Delta t}{3}\left(y_1 + y_9 + 2\sum_{i=2}^{9} y_9 \right) = 3.9909$$

Simpson's rule:

$$A = \frac{\Delta t}{3}\left(y_1 + 4(y_2 + y_4 + y_6 + y_8) + 2(y_3 + y_5 + y_7) + y_9\right)$$
$$= 4.1087$$

Example. Use Simpson's rule with n=32 to evaluate the following integrals by a computer program

B. $$\int_0^\infty e^{-x}\sin^2 x\,dx$$

C. $$\int_0^\infty \frac{1}{(1 + x^2)\left(1 + \frac{x^2}{2}\right)}dx$$

D. $\int_{-1}^{\infty} xe^{-x}dx$

E. $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-x^2/2}dx$

There are two methods of calculating any of the above integrals

a. Variable substitution that transforms the integrals to ones with bounded limits.

b. Setting the limits based on the arithmetic precision of the variable type or machine you are using.

2. Since $\sin^2 x$ is bounded between 0 and 1 and e^{-x} decreases monotonically with x then the upper limit can be set based on the equation:

$$e^{-x} <= \varepsilon$$

$$\text{we will select } \varepsilon = 10^{-7}$$

or $\sqrt{7\ln(10)} \approx 4$

The same approach can be followed for the rest of the integrals.

3. $\frac{1}{x^2 . \frac{x^2}{2}} = 10^{-7}$

or $x = \sqrt[4]{2\times 10^{-7}} \cong 67 \leftarrow$ upper limit

4. $xe^{-x} = 10^{-7}$

As a first approximation try:

$e^{-x} = 10^{-7}$

or $x \approx 16$

Try $x = 18$

$18e^{-18} = 1.07\times 10^{-7}$

5. $e^{-x^2/2} = 10^{-7}$

$x = \sqrt{14\ln(10)} \approx 2.4$

The class Integration is utilized with the following functions and main() program:

```
 double FUNA(double x)
  {
  return exp(-x)*sin(x)*sin(x);
  }

 double FUNB(double x)
  {
  return 1.0/((1+x*x)*(1+x*x/2.0));
  }

 double FUNC(double x)
  {
  return x*exp(-x);
  }

 double FUND(double x)
  {
  return exp(-x*x/2.0);
  }

//------------------------------------------------------------------------

int main()
{
Integration IA(FUNA,0,4,32), IB(FUNB,0,67,32), IC(FUNC,-1,18,32),
    ID(FUND,-2.4,2.4,32);

cout << "A.  " << IA.Simpson() << endl;
cout << "B.  " << IB.Simpson() << endl;
cout << "C.  " << IC.Simpson() << endl;
cout << "D.  " << (ID.Simpson()/sqrt(2.0*3.14159)) << endl;

getch();
 return 1;
  }
```

The results are as follows:

A. 0.38696
B. 0.872791
C. -0.0070575
D. 0.983605

Problems

1. Evaluate the following integrals using Simpson's 1/3rd rule:

 a. $\int_0^1 (1+x^2)^{3/2}\,dx$ (Exact answer 1.567951962…)

 b. $\int_0^\infty xe^{-x}dx$ (Exact answer $=1.0$)

2. Determine the area enclosed by an ellipse $\left(\frac{x}{5}\right)^2+\left(\frac{x}{3}\right)^2=1$ using the trapezoidal rule.

3. Develop a formulation for evaluating the double integral using the Simpson's 1/3rd rule

$$I=\int_a^b\int_c^d f(x,y)dxdy$$

4. Develop a C++ class for evaluating double integrals.

5. Evaluate the following integral using the class developed in problem 4:

$$I=\int_0^2\int_0^1 (x^2y+5)dxdy$$

using 8 intervals in both directions.