Numerical Methods of Integration

Richa Sharma

Department of Physics & Astrophysics
University of Delhi

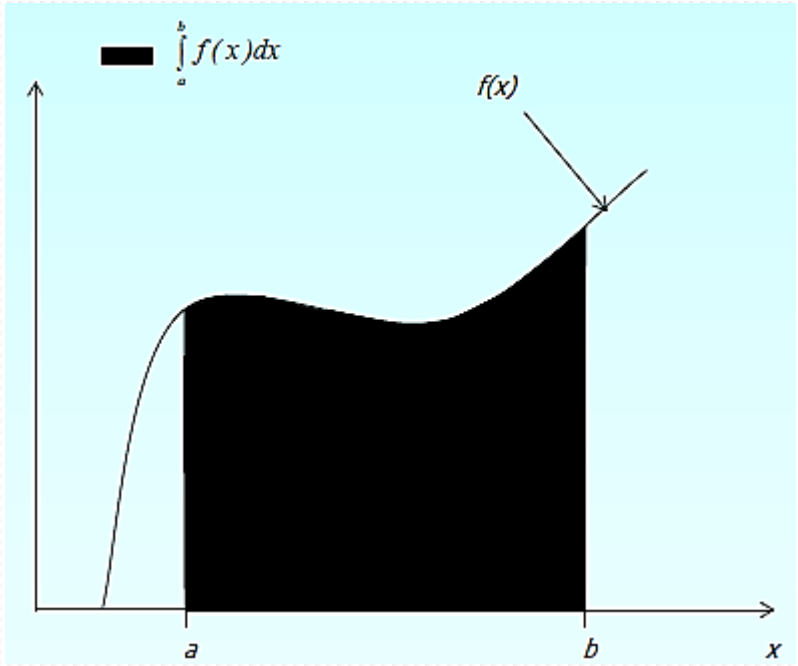
Outline :

- **Integartion**
- **Different methods of Numerical Integration :**
  - **Uniformly-spaced samples**
    - Newton–Cotes formulas
  - **Non-uniformly spaced samples**
    - Gaussian Quadrature Formulas
- **Program Code**

What is Integration?

The process of measuring the area under a curve.

$$I = \int_a^b f(x)dx$$



Where:
f(x) is the integrand
a= lower limit of integration
b= upper limit of integration

Numerical Integration :

- constitutes a broad family of algorithms for calculating the numerical value of a integral.
- The integrand *f(x)* may be known only at certain points, such as obtained by sampling.
- A formula for the integrand may be known, but it may be difficult or impossible to find an antiderivative .
- It may be possible to find an antiderivative symbolically, but it may be easier to compute a numerical approximation than to compute the antiderivative.

- The methods that are based on equally spaced data points: these are Newton-cotes formulas: the mid-point rule, the trapezoid rule and simpson rule.

- The methods that are based on data points which are not equally spaced:these are Gaussian quadrature formulas.

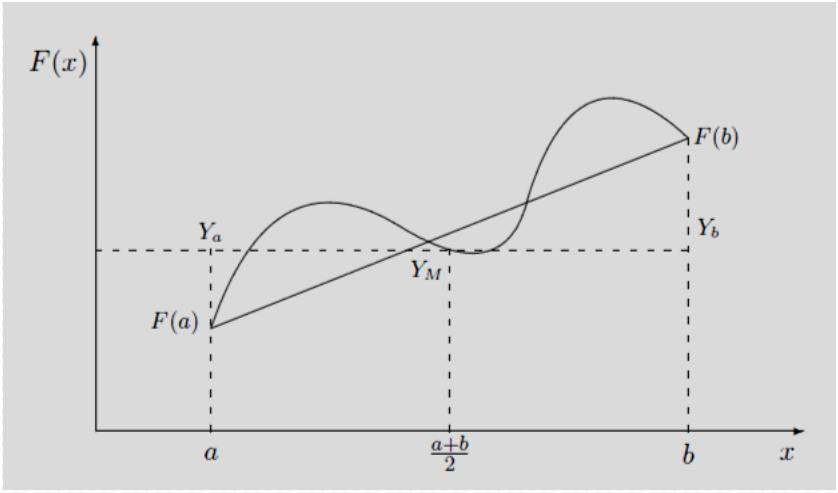
Newton–Cotes formulas

- derived from interpolating polynomials
- evaluate the integrand at equally-spaced points.
- the formulas are exact for polynomials of degree less than or equal to n.

➢ **Types of Newton–Cotes formulas**

- Mid-Point rule
- Trapezoidal Rule
- Simpson Rule

Mid-point Rule:



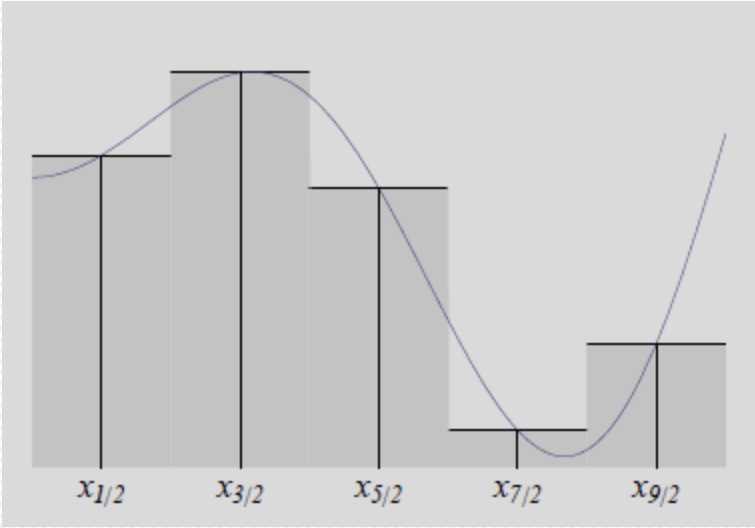
- compute the area of the rectangle formed by the four points *(a,0),(0,b),(a,f(a+b)/2)) and (b,f(a+b)/2))*

- such that such that the approximate integral is given by

$$\int_a^b F(x)\mathrm{d}x = (b-a)F\left(\frac{a+b}{2}\right)$$

- this rule does not make any use of the end points.

Composite Mid-point Rule:



- the interval [a ,b] can be break into smaller intervals and compute the approximation on each sub interval.
- Sub-intervals of size h = $(b-a)/n,$
- n+1 data points

$$x_i = a + \left(i - \tfrac{1}{2}\right) h \text{ with } i = 1, \ldots, n.$$

- Value of integral is :

$$\int_a^b F(x)\mathrm{d}x = h\sum_{i=1}^{n} f(x_i)$$

Error in Mid-point Rule

$$\int_a^b f(x)\,dx - f(a_{1/2})(b-a), \quad a_{1/2} = (a+b)/2.$$

expanding both f (x) and f (a_1/2) about the the left endpoint a, and then integrating the taylor expansion we get error

$$E_m = \frac{7M}{24}(b-a)^3,$$

$$\text{where } M = |f''(x)|$$

If we reduce the size of the interval to half its width, the error in the mid-point method will be reduced by a factor of 8.

Trapezoidal Rule :

Newton-Cotes Formula that states if one can approximate the integrand as an n^{th} order polynomial...

$$I = \int_a^b f(x)dx \qquad \text{where} \qquad f(x) \approx f_n(x)$$

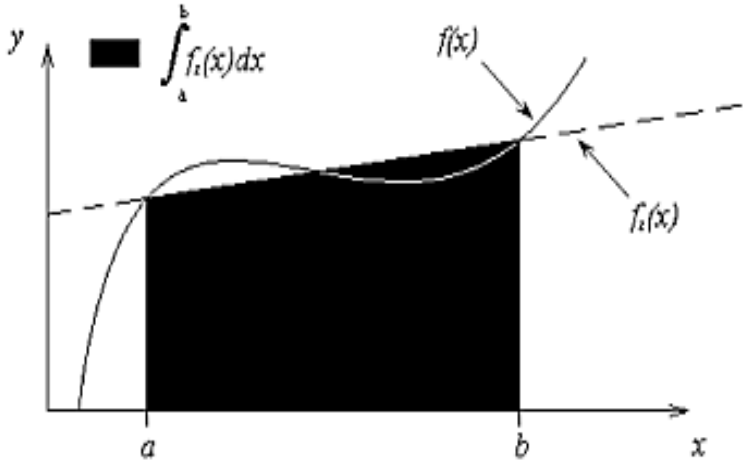
and $$f_n(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} + a_n x^n$$

Then the integral of that function is approximated by the integral of that n^{th} order polynomial.

$$\int_a^b f(x) \approx \int_a^b f_n(x)$$

Trapezoidal Rule assumes n=1, that is, the area under the linear polynomial,

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



Proof :

$$\int_a^b f(x)dx \approx Area\ of\ trapezoid$$

$$= \frac{1}{2}(Sum\ of\ parallel\ sides)(height)$$

$$= \frac{1}{2}(f(b)+f(a))(b-a)$$

$$= (b-a)\left[\frac{f(a)+f(b)}{2}\right]$$

Composite Trapezoidal Rule :

The integral *I* can be broken into *h* integrals as:

$$\int_a^b f(x)dx = \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \ldots + \int_{a+(n-2)h}^{a+(n-1)h} f(x)dx + \int_{a+(n-1)h}^{b} f(x)dx$$

Applying Trapezoidal rule on each segment gives:

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

Error in Trapezoidal Rule

$$E_t = \frac{5M}{12}(b-a)^3,$$

where $M = |f''(x)|$.

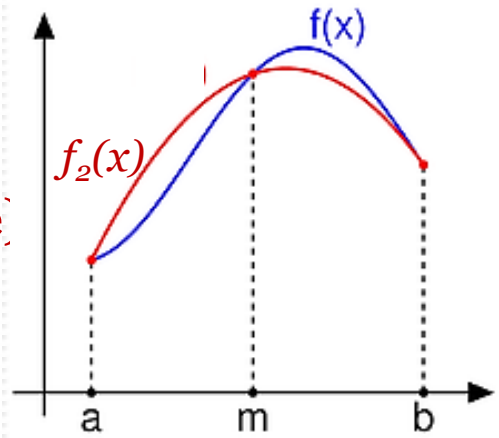
- even though the trapezoid rule uses two values of *f* , *the error estimate is slightly larger than the* estimate for the midpoint method.

- the exponent on (*b-a), which tells us how quickly the error goes to 0 when the interval width* is reduced, and from this point of view the two methods are the same.

Simpson Rule :

- **Simpson's 1/3 Rule :**

derived by approximating the integrand $f(x)$ (**in blue**) by a second order polynomial



$$f_2(x) = a_0 + a_1 x + a_2 x^2$$ **(in red).**

$$I = \int_a^b f(x)dx \approx \int_a^b f_2(x)dx$$

Choose

$$(a, f(a)), \left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right), \text{ and } (b, f(b))$$

as the three points of the function to evaluate a_0, a_1 and a_2.

Then

$$I \approx \int_a^b f_2(x)dx$$

$$= \int_a^b \left(a_0 + a_1 x + a_2 x^2\right)dx$$

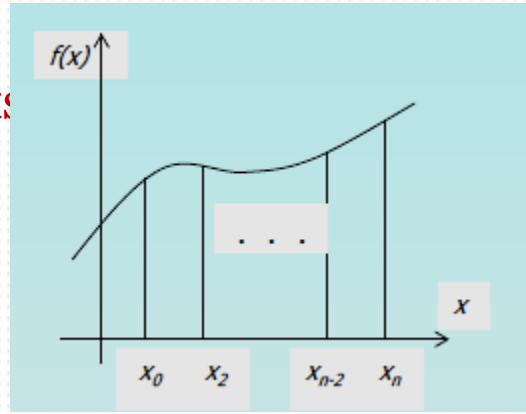
$$= \left[a_0 x + a_1 \frac{x^2}{2} + a_2 \frac{x^3}{3}\right]_a^b$$

Substituting values of a_0, a_1, a_2 give

$$\int_a^b f_2(x)dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Because the above form has 1/3 in its formula, it is called Simpson's 1/3rd Rule.

Composite Simpson's 1/3rd Rule :



- one can subdivide the interval [a, b] into n segments and apply Simpson's 1/3rd Rule repeatedly over every two segments.
- the segment width h,

$$h = \frac{b-a}{n} \qquad \int_a^b f(x)dx = \int_{x_0}^{x_n} f(x)dx$$

$x_o = a$, $x_n = b$

$$\int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots..$$

$$\ldots. + \int_{x_{n-4}}^{x_{n-2}} f(x)dx + \int_{x_{n-2}}^{x_n} f(x)dx$$

Apply Simpson's 1/3rd Rule over each interval,

$$\int_a^b f(x)dx = \frac{h}{3}[f(x_0)+4\{f(x_1)+f(x_3)+\ldots+f(x_{n-1})\}+\ldots]$$

$$\ldots+2\{f(x_2)+f(x_4)+\ldots+f(x_{n-2})\}+f(x_n)\}]$$

$$= \frac{h}{3}\left[f(x_0)+4\sum_{\substack{i=1 \\ i=odd}}^{n-1} f(x_i)+2\sum_{\substack{i=2 \\ i=even}}^{n-2} f(x_i)+f(x_n) \right]$$

Rule is :

$$\int_a^b f(x)dx = \frac{b-a}{3n}\left[f(x_0)+4\sum_{\substack{i=1 \\ i=odd}}^{n-1} f(x_i)+2\sum_{\substack{i=2 \\ i=even}}^{n-2} f(x_i)+f(x_n) \right]$$

Error in Simpson Rule :

The error in Simpson's 1/3rd Rule is given as

$$E_t = -\frac{h^5}{90} f^{(4)}(\xi)$$

Since the error term is proportional to the fourth derivative of *f* at ξ, this shows that Simpson's rule provides exact results for any polynomial *f* of degree three or less,

⇒Integrates a cubic exactly:

$$f^{(4)}(\xi) = 0$$

- The methods we presented so far were defined over finite domains, but it will be often the case that we will be dealing with problems in which the domain of integration is infinite.

- We will now investigate how we can transform the problem to be able to use standard methods to compute the integrals.

Gaussian Quadrature

- Gaussian Quadrature & Optimal Nodes
- Using Legendre Polynomials to Derive Gaussian Quadrature Formulae
- Gaussian Quadrature on Arbitrary Intervals

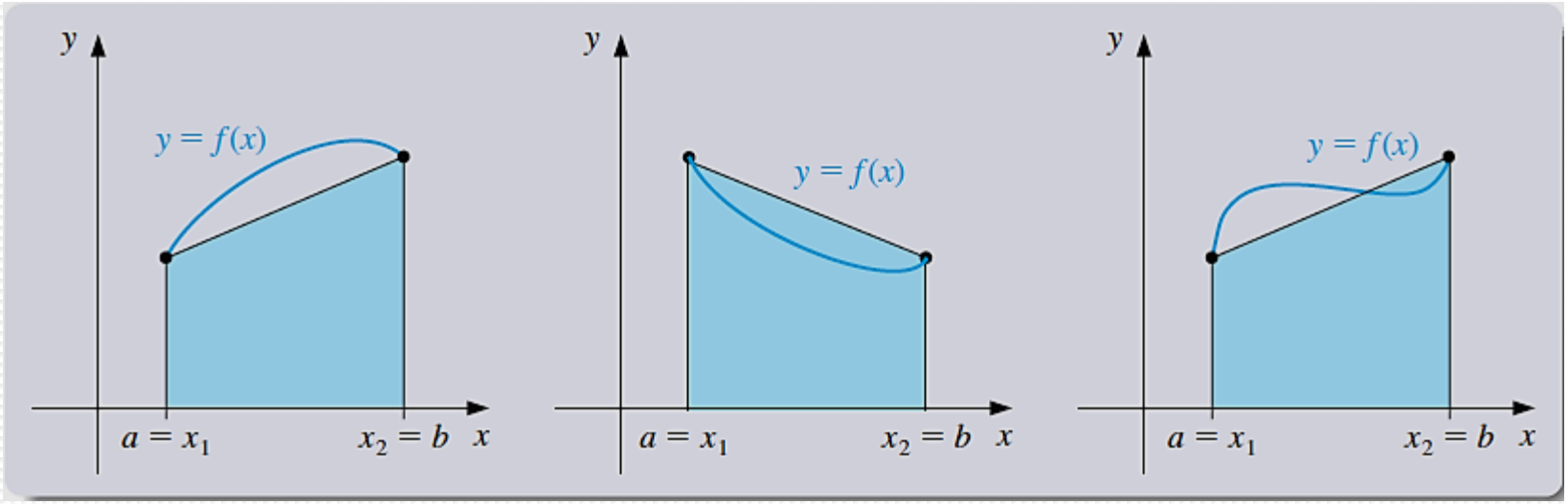
Gaussian Quadrature: Contrast with Newton-Cotes

- The Newton-Cotes formulas were derived by integrating interpolating polynomials.
- The error term in the interpolating polynomial of degree n involves the $(n + 1)$st derivative of the function being approximated, . . .
- so a Newton-Cotes formula is exact when approximating the integral of any polynomial of degree less than or equal to n.
- All the Newton-Cotes formulas use values of the function at equally-spaced points.
- This restriction is convenient when the formulas are combined to form the composite rules which we considered earlier, . . .

But in Gaussian Quadrature

- we may find sets of weights and abscissas that make the formulas exact for integrands that are composed of some real function multiplied by a polynomial
- gives us a huge advantage in calculating integrals numerically.

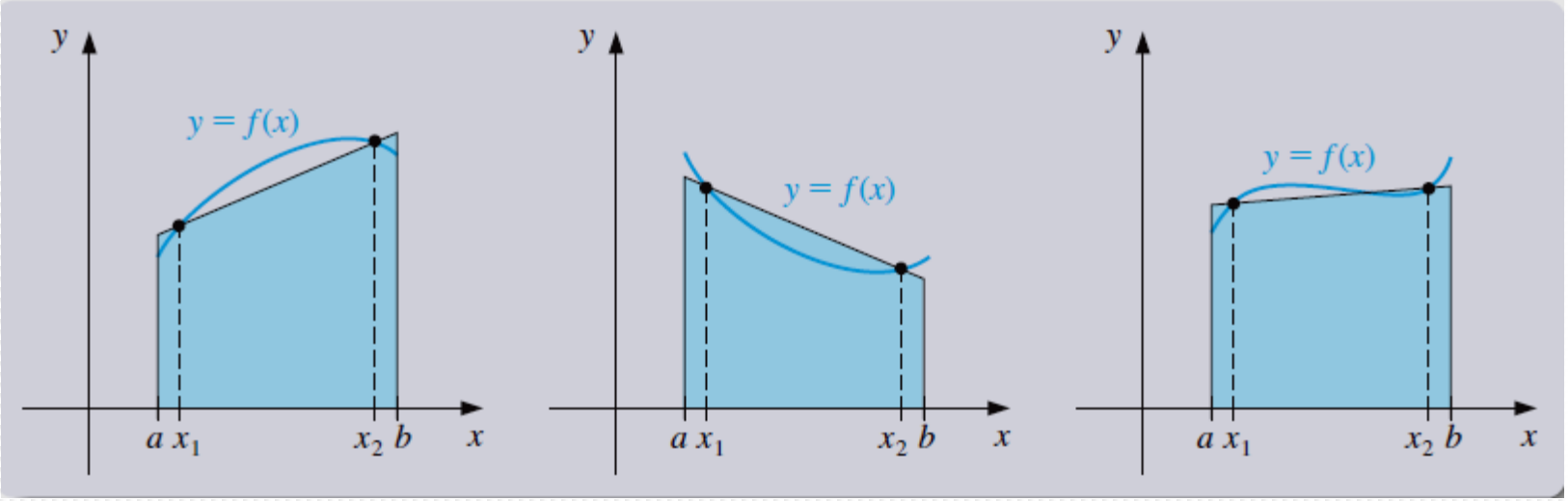
Consider, for example, the Trapezoidal rule applied to determine the integrals of the functions whose graphs are as shown.



It approximates the integral of the function by integrating the linear function that joins the endpoints of the graph of the function.

Gaussian Integration: Optimal integration points

But this is not likely the best line for approximating the integral. Lines such as those shown below would likely give much better approximations in most cases.



Gaussian quadrature chooses the points for evaluation in an optimal, rather than equally-spaced, way.

Gaussian Quadrature: Introduction

Choice of Integration Nodes :

- The nodes $x_1, x_2, \ldots, x_n$ in the interval $[a, b]$ and coefficients $c_1, c_2, \ldots, c_n$, are chosen to minimize the expected error obtained in the approximation

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} c_i f(x_i).$$

- assume that the best choice of these values produces the exact result for the largest class of polynomials, . . .
- The coefficients $c_1, c_2, \ldots, c_n$, in the approximation formula are arbitrary, and the nodes $x_1, x_2, \ldots, x_n$ are restricted only by the fact that they must lie in $[a, b]$, the interval of integration.
- This gives us $2n$ parameters to choose.

- The way that this is done is through viewing the integrand as being composed of some weighting function W(x) multiplied by some polynomial P(x) so that

$$f(x)=W(x)P(x)$$

- Instead of using simple polynomials to interpolate the function, quadrature use the set of polynomials that are orthogonal over the interval with weighting function W(x).

- With this choice of interpolating polynomial, we find that if we evaluate P(x) at the zeroes (x_i) of the interpolating polynomial of desired order, and multiply each evaluation by a weighting factor (w_i)

- we can obtain a result that is exact up to twice the order of the interpolating polynomial!

Gaussian quadrature method based on the polynomials p_m as follows

Let $x_0, x_1, \ldots, x_n$ be the roots of p_{n+1}.
Let l_i the *ith* Lagrange interpolating polynomial for these roots, i.e. l_i is the unique polynomial of degree $\leq n$. Then

$$\int_a^b f(x)w(x)\,dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

where the weights w_i are given by

$$w_i = \int_a^b \ell_i(x)w(x).$$

Gaussian Quadrature: Illustration ($n = 2$)

Example: Formula when $n = 2$ on $[-1, 1]$

Suppose we want to determine c_1, c_2, x_1, and x_2 so that the integration formula

$$\int_{-1}^{1} f(x)\,dx \approx c_1 f(x_1) + c_2 f(x_2)$$

gives the exact result whenever $f(x)$ is a polynomial of degree $2(2) - 1 = 3$ or less, that is, when

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$$

for some collection of constants, a_0, a_1, a_2, and a_3.

Because

$$\int (a_0 + a_1 x + a_2 x^2 + a_3 x^3)\, dx$$

$$= a_0 \int 1\, dx + a_1 \int x\, dx + a_2 \int x^2\, dx + a_3 \int x^3\, dx$$

this is equivalent to showing that the formula gives exact results when $f(x)$ is 1, x, x^2, and x^3.

Hence, we need c_1, c_2, x_1, and x_2, so that

$$\begin{aligned}
c_1 \cdot 1 + c_2 \cdot 1 &= \int_{-1}^{1} 1\, dx = 2 \\
c_1 \cdot x_1 + c_2 \cdot x_2 &= \int_{-1}^{1} x\, dx = 0 \\
c_1 \cdot x_1^2 + c_2 \cdot x_2^2 &= \int_{-1}^{1} x^2\, dx = \frac{2}{3} \\
c_1 \cdot x_1^3 + c_2 \cdot x_2^3 &= \int_{-1}^{1} x^3\, dx = 0
\end{aligned}$$

A little algebra shows that this system of equations has the unique solution

$$c_1 = 1, \quad c_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3} \quad \text{and} \quad x_2 = \frac{\sqrt{3}}{3},$$

which gives the approximation formula

$$\int_{-1}^{1} f(x)\,dx \approx f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

This formula has degree of precision 3, that is, it produces the exact result for every polynomial of degree 3 or less.

Using Legendre Polynomials to Derive Gaussian Quadrature Formulae

Gaussian Quadrature: Legendre Polynomials

An Alternative Method of Derivation

- We will consider an approach which generates more easily the nodes and coefficients for formulas that give exact results for higher-degree polynomials.
- This will be achieved using a particular set of orthogonal polynomials (functions with the property that a particular definite integral of the product of any two of them is 0).

- This set is the is the Legendre polynomials, a collection $\{P_0(x), P_1(x), \ldots, P_n(x), \ldots, \}$ with properties:
  - **(1)** For each n, $P_n(x)$ is a monic polynomial of degree n.
  - **(2)** $\int_{-1}^{1} P(x)P_n(x)\,dx = 0$ whenever $P(x)$ is a polynomial of degree less than n.

The first few Legendre Polynomials

$$P_0(x) = 1, \qquad P_1(x) = x, \qquad P_2(x) = x^2 - \frac{1}{3}$$

$$P_3(x) = x^3 - \frac{3}{5}x \qquad \text{and} \qquad P_4(x) = x^4 - \frac{6}{7}x^2 + \frac{3}{35}$$

- The roots of these polynomials are distinct, lie in the interval (–1, 1), have a symmetry with respect to the origin, and, most importantly,
- they are the correct choice for determining the parameters that give us the nodes and coefficients for our quadrature method

The nodes $x_1, x_2, \ldots, x_n$ needed to produce an integral approximation formula that gives exact results for any polynomial of degree less than $2n$ are the roots of the nth-degree Legendre polynomial.

Theorem

Suppose that $x_1, x_2, \ldots, x_n$ are the roots of the nth Legendre polynomial $P_n(x)$ and that for each $i = 1, 2, \ldots, n$, the numbers c_i are defined by

$$c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \, dx$$

If $P(x)$ is any polynomial of degree less than $2n$, then

$$\int_{-1}^{1} P(x) \, dx = \sum_{i=1}^{n} c_i P(x_i)$$

Proof :

- Let us first consider the situation for a polynomial $P(x)$ of degree less than n.
- Re-write $P(x)$ in terms of $(n - 1)$st Lagrange coefficient polynomials with nodes at the roots of the nth Legendre polynomial $Pn(x)$.
- Since $P(x)$ is of degree less than n, the nth derivative of $P(x)$ is 0, and this representation of is exact. So

Therefore

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

and

$$\int_{-1}^{1} P(x)\, dx = \int_{-1}^{1} \left[\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} P(x_i) \right] dx$$

$$= \sum_{i=1}^{n} \left[\int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}\, dx \right] P(x_i) = \sum_{i=1}^{n} c_i P(x_i)$$

Hence the result is true for polynomials of degree less than n.

- Now consider a polynomial $P(x)$ of degree at least n but less than $2n$.
- Divide $P(x)$ by the nth Legendre polynomial $P_n(x)$.
- This gives two polynomials $Q(x)$ and $R(x)$, each of degree less than n, with
$$P(x) = Q(x)P_n(x) + R(x)$$
- Note that x_i is a root of $P_n(x)$ for each $i = 1, 2, \ldots, n$, so we have
$$P(x_i) = Q(x_i)P_n(x_i) + R(x_i) = R(x_i)$$
- We now invoke the unique power of the Legendre polynomials.

First, the degree of the polynomial $Q(x)$ is less than n, so (by the Legendre orthogonality property),

$$\int_{-1}^{1} Q(x)P_n(x)\,dx = 0$$

Then, since $R(x)$ is a polynomial of degree less than n, the opening argument implies that

$$\int_{-1}^{1} R(x)\,dx = \sum_{i=1}^{n} c_i R(x_i)$$

Putting these facts together verifies that the formula is exact for the polynomial $P(x)$:

$$\begin{aligned}\int_{-1}^{1} P(x)\,dx &= \int_{-1}^{1} [Q(x)P_n(x) + R(x)]\,dx \\ &= \int_{-1}^{1} R(x)\,dx \\ &= \sum_{i=1}^{n} c_i R(x_i) \\ &= \sum_{i=1}^{n} c_i P(x_i)\end{aligned}$$

The constants c_i needed for the quadrature rule can be generated from the equation given in the theorem:

$$c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx$$

but both these constants and the roots of the Legendre polynomials are extensively tabulated.

The following table lists these values for n = 2, 3, 4, and 5.

n	Roots $r_{n,i}$	Coefficients $c_{n,i}$
2	0.5773502692	1.0000000000
	−0.5773502692	1.0000000000
3	0.7745966692	0.5555555556
	0.0000000000	0.8888888889
	−0.7745966692	0.5555555556
4	0.8611363116	0.3478548451
	0.3399810436	0.6521451549
	−0.3399810436	0.6521451549
	−0.8611363116	0.3478548451
5	0.9061798459	0.2369268850
	0.5384693101	0.4786286705
	0.0000000000	0.5688888889
	−0.5384693101	0.4786286705
	−0.9061798459	0.2369268850

Gaussian Quadrature on Arbitrary Intervals

Transform the Interval of Integration from [a,b] to [−1, 1]

An integral $\int_a^b f(x)\,dx$ over an arbitrary $[a, b]$ can be transformed into an integral over $[-1, 1]$ by using the change of variables ▸ See Diagram:

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

This permits Gaussian quadrature to be applied to any interval $[a, b]$, because

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f\left(\frac{(b - a)t + (b + a)}{2}\right)\frac{(b - a)}{2}\,dt$$

Gauss–Laguerre quadrature :

- numerical analysis **Gauss–Laguerre quadrature** is an extension of Gaussian quadrature method for approximating the value of integrals of the following kind:

$$\int_0^{+\infty} e^{-x} f(x)\,dx.$$

- In this case

$$\int_0^{+\infty} e^{-x} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

- To integrate the function

$$\int_0^{\infty} f(x)\,dx = \int_0^{\infty} f(x)\,e^{x}e^{-x}dx = \int_0^{\infty} g(x)\,e^{-x}dx$$

where x_i is the i-th root of Laguerre polynomial $L_n(x)$ and the weight w_i is given by

$$w_i = \frac{x_i}{(n+1)^2[L_{n+1}(x_i)]^2}.$$

Gauss–Hermite quadrature

- method for approximating the value of integrals of the following kind:

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx.$$

- In this case

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where n is the number of sample points to use for the approximation.

The x_i are the roots of the Hermite polynomial $H_n(x)$ ($i = 1,2,\ldots,n$) and the associated weights w_i are given by

$$w_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}.$$

Program code

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

double f(double x)
{
double y;
//y = (arctan(x))/(pow(x,2));
y= log(1+pow(x,2));
//y=1;
//y= 1/(x+3);
return (y);
}

double g_quad(double a,double b, int n1)
{
double x[20],w[20],z[20],s3,ans,s;
int i;
/* x[0] = 0.33998; x[1] =0.8611363 ;x[2] =-0.33998 ;x[3] =-0.8611363 ;
```

```
w[0] = 0.652145;  w[1] = 0.347854;w[2] =0.652145 ;w[3] = 0.347854;
*/
w[0] = 0.362683783378620;w[1] = 0.362683783378620;
w[2] =0.313706645877887 3 ;w[3] = 0.313706645877887 3;
w[4] = 0.222381034453374 5;w[5] = 0.222381034453374 5;
w[6] =0.101228536290376 3 ;w[7] = 0.101228536290376 3;
x[0] = -0.183434642495649 8;x[1] = 0.183434642495649 8;
x[2] = -0.525532409916329 0;x[3] = 0.525532409916329 0;
x[4] =-0.796666477413626 7 ;x[5] = 0.796666477413626 7;
x[6] = -0.960289856497536 3;x[7] = 0.960289856497536 3;
s3=0.0;
for(i=0; i<n1; i++)
{
z[i] = (((b+a)/2) + ((b-a)/2)*x[i]);
s=z[i];
s3 = s3 + (w[i]*f(z[i]));
}
ans = ((b-a)/2.0)*s3;
return(ans);
}
```

```
double g_lag(int n1)
{
double x[20],w[20],z[20],s4;
int i;
s4 =0.0;
x[0] = 0.32254; x[1] = 1.74757;x[2] = 4.53662;x[3] = 9.39507;
w[0] =0.60315 ;w[1] = 0.35741;w[2] =0.038887 ;w[3] = 0.0005392;

for(i=0; i<n1; i++)
{
s4 = s4 + (w[i] * f(x[i]));
}
return(s4);
}

double g_her(int n1)
{
double x[20],w[20],z[20],s7;
int i;
s7=0.0;
```

```
x[0] =0.38118 ;  x[1] =1.1571 ;x[2] = 1.9816;x[3] =2.93063 ;
x[4] =-0.3811 ;   x[5] =-1.15719 ; x[6] = -1.98165; x[7] = -2.930;
w[0] = 0.66114;   w[1] =0.2078 ;w[2] = 0.01707;w[3] = 0.000109;
w[4] =0.66114 ;w[5] =0.2078 ;w[6] = 0.01707;w[7] = 0.000109;
for(i=0; i<n1; i++)
{
s7 = s7+ (w[i]*f(x[i]));
}
return(s7);
}

main()
{
clrscr();
double a,b,s1,s2,s5,s6,y1,y2,h,r;
int n,n1,i,c;

s1= 0.0;
s2=0.0;
```

```
cout<<"Enter     1:  For finite limits      \n";
cout<<"          2:  Limits from zero to infinity    \n";
cout<<"          3:  Infinite limits       \n";

cout<<"Enter Choice \n";
cin>>c;

switch(c)
{
case 1:

cout<<"\n";
cout<<"Enter values of a , b and n \n";
cin>>a>>b>>n;
cout<<" Enter value of n for gauss quadrature :\n";
cin>>n1;

h=(b-a)/n;
for(i=1; i<=(n-1);i++)
{
s1=s1+f(a + (i*h));
}
```

```
r = (h*(f(a) + f(b) + (2*s1)))/2;
cout<<"\n";
cout<<"Using trapezoidal rule :  I = "<<r<<"\n";

//----simpson rule----------
for(i=1; i<=(n/2); i++)
{
y1 = y1 + f(a + (((2*i) - 1)*h));
}
for(i=1; i<=(n/2)-1; i++)
{
y2 = y2 + f(a + (2*i*h));
}
r = (h*(f(a) + f(b) + (4*y1) + (2*y2)))/3;
cout<<"\n";
cout<<"Using Simpson Rule  :   I = "<<r<<"\n";
//------end-------------

//----------Gauss Quadrature-----------------

s2 = g_quad(a,b,n1);
cout<<"Using Gauss Quadrature  :  I = "<<s2<<"\n";
```

```
cout<<"\n";
//---------end-----------------

break;

case 2:

//---------Gauss Laguerre-------------

cout<<" Enter value of n for gauss laguerre :\n";
cin>>n1;

s5 =  g_lag(n1);
cout<<"Using Gauss Laguerre  :  I = "<<s5<<"\n";
cout<<"\n";
//-------end-----------------------
break;

//----------------Gauss hermite-----------
case 3 :
cout<<" Enter value of n for gauss hermite :\n";
cin>>n1;
```

```
s6 = g_her(n1);
cout<<"Using Gauss Hermite  :  I = "<<s6<<"\n";
cout<<"\n";
//----------end---------------
break ;

}
getch();
}
```

Thank You