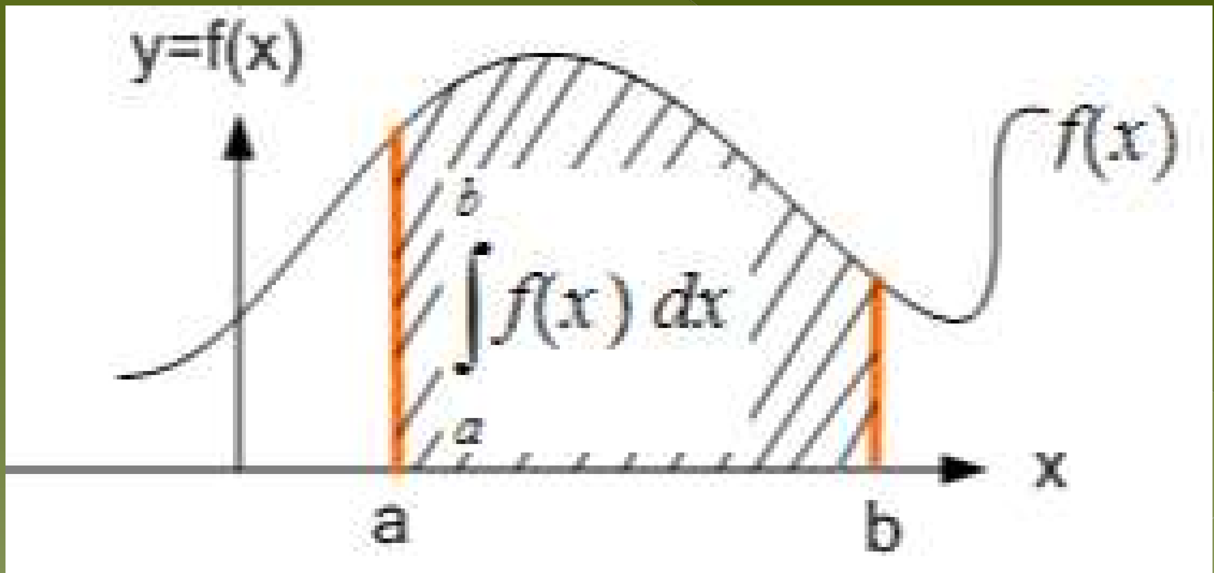
NUMERICAL INTEGRATION

Definition:

Numerical integration is equivalent to the area under the function curve with in the integration limit.



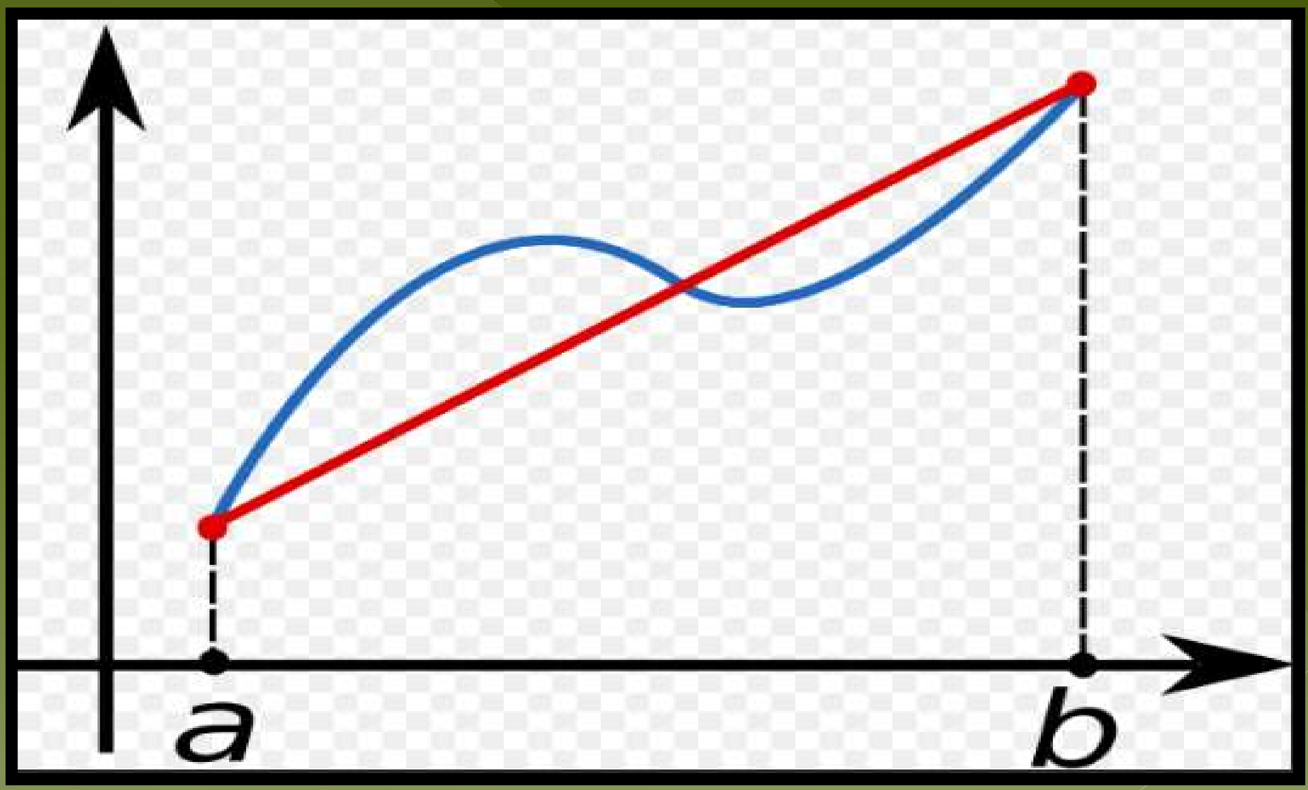
Types Of Integration Method

There are 3 types of integration method:

- Trapezoidal Rule
- Simpson's 1/3 Rule
- Gaussian Quadrature

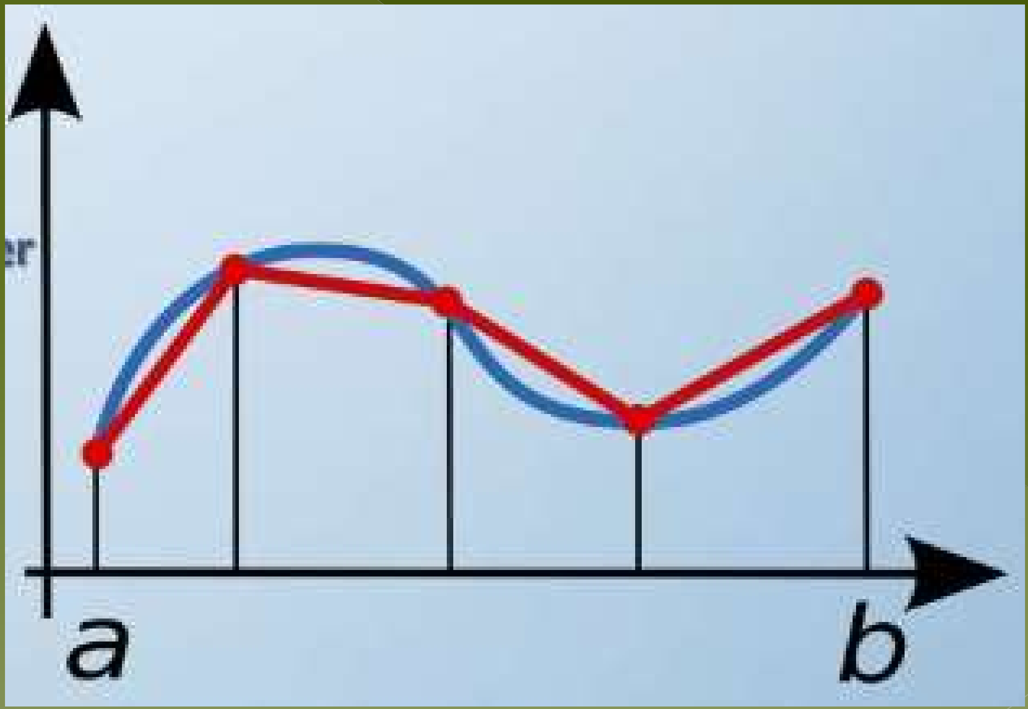
Trapezoidal Rule

In this method the area under the curve is approximated by a Trapezoid.

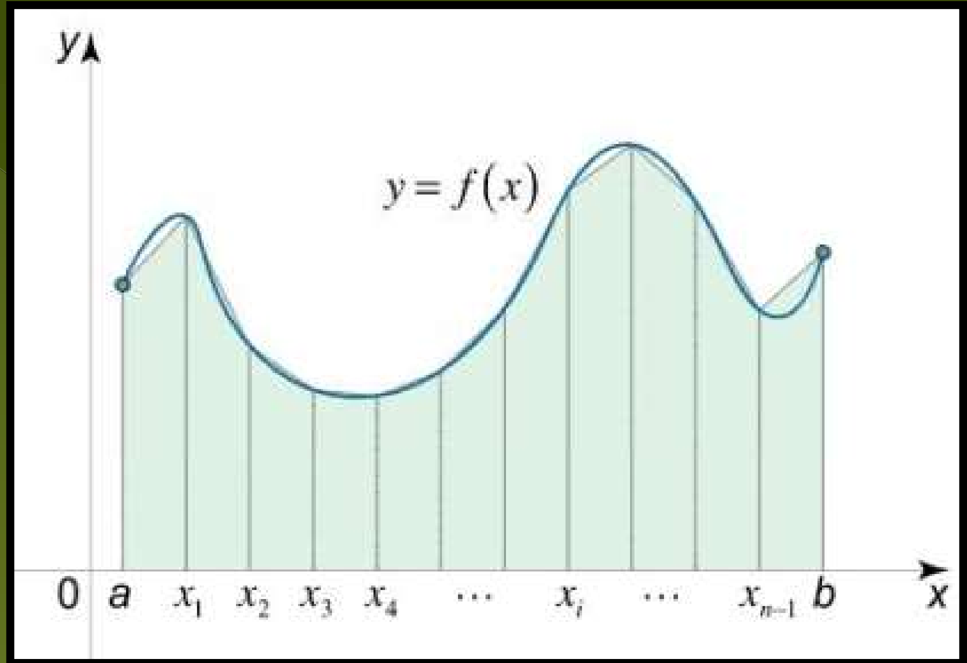


Trapezoidal Rule

To reduce the error more Trapezoid are to be used for better fit to the curvature of the graph.



Trapezoidal Rule: Formula



$$\int_a^b f(x)\,dx \approx T_n = \frac{\Delta x}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right],$$

where $\Delta x = \frac{b-a}{n}$ and $x_i = a + i\Delta x$.

Python Code To Find: $\int_0^1 e^x dx$

```
import math
def f(x):return (math.exp(x))
a = input('Enter the lower limit:  ')
b = input('Enter the upper limit:  ')
s=(f(a)+f(b))
k=100
h=float(b-a)/k
for i in range(1,k):
        x=a+i*h
        s=s+2*f(x)

print 'Integration =',s*h/2
```

OUTPUT

Enter the lower limit: 0
Enter the upper limit: 1
Integration = 1.71829614745

Trapezoidal Rule: With Accuracy

```
import math

def f(x):return math.exp(x)

a = float(input('Enter the lower limit:  '))
b = float(input('Enter the upper limit:  '))
s1=f(a)+f(b)
s1_old,sum=0.0,0.0
k=30

while abs(s1_old-s1)> 0.0001:
        s1_old=sum
        h=float(b-a)/k
        for i in range(1,k):
                x=a+i*h
                s1=s1+2*f(x)

        s1=h*s1/2
        sum=s1
        k=k+10

print ('Integration =',s1)
```

OUTPUT(With Accuracy)

Enter the lower limit: 0
Enter the upper limit: 1
Integration = 1.71524809993

Trapezoidal Rule Using Integration function from Numpy and Scipy Module

Using Numpy Module:

- import numpy as np
- a=float(input("Enter the lower limit:"))
- b=float(input("Enter the upper limit:"))
- n=int(input("Enter the number of interval:"))
- x=np.linspace(a,b,n)
- y=np.sqrt(x+1)
- I=np.trapz(y,x)
- print("The value of integration:",I)

Enter the lower limit:0
Enter the upper limit:1
Enter the number of interval:200
The value of integration:
1.2189511083266238

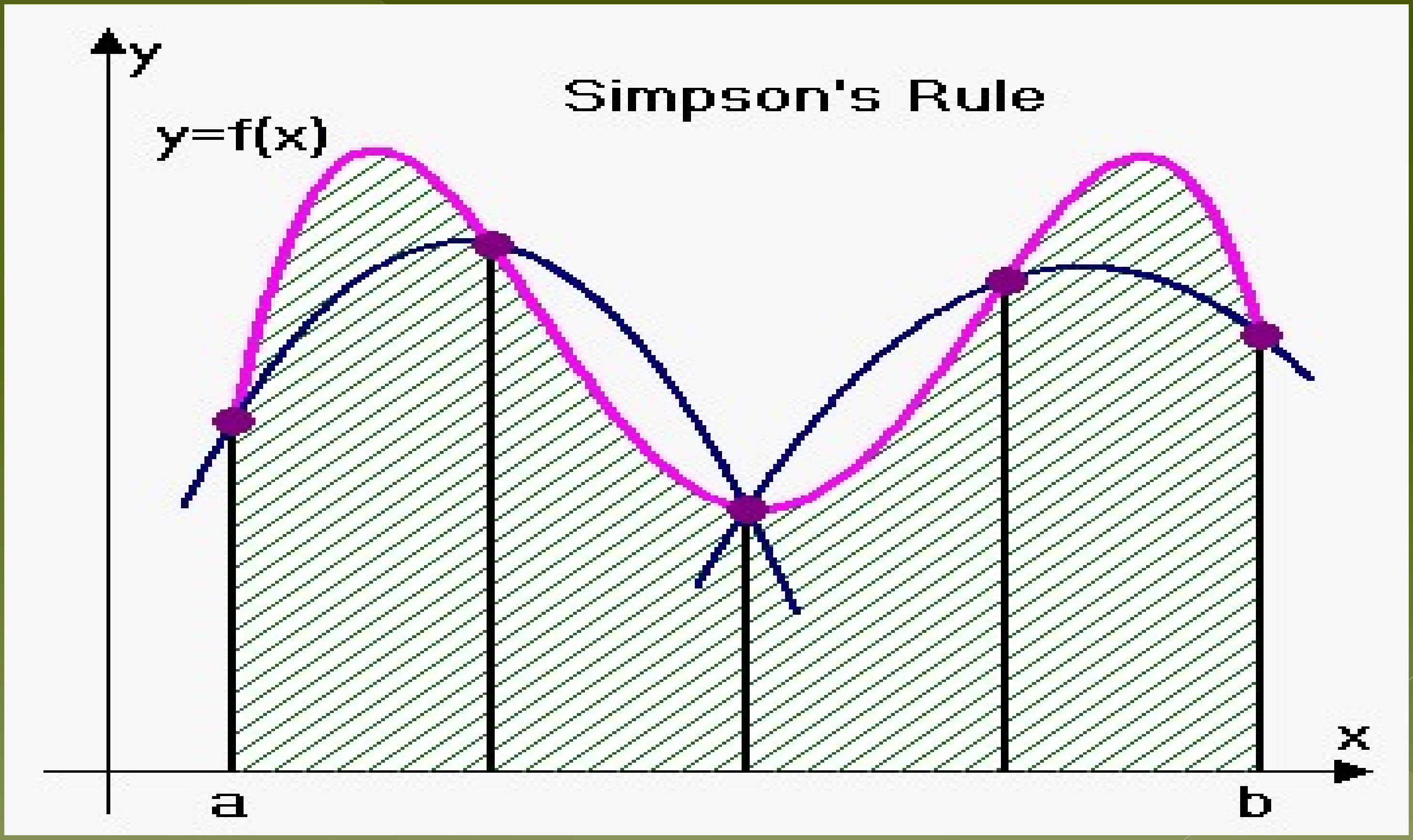
Using Scipy Module:

```
import numpy as np
import scipy
from scipy import integrate
a=float(input("Enter the lower limit:"))
b=float(input("Enter the upper limit:"))
n=int(input("Enter the number of interval:"))
x=np.linspace(a,b,n)
y=np.sqrt(x+1)
I=scipy.integrate.trapz(y,x)
print("The value of integration:",I)
```

Enter the lower limit:0
Enter the upper limit:1
Enter the number of interval:100
The value of integration:
1.2189501713346835

Simpson's 1/3 Rule

In this method the area under the curve is approximated by a Parabola.



Simpson's1/3 Rule: Formula

$$\int_a^b f(x)dx = \frac{h}{3}[f(x_0) + f(x_n) + 2\{f(x_2) + f(x_4) + \cdots\} + 4\{f(x_1) + f(x_3) + \cdots\}]$$

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

Python Code To Find: $\int_0^1 (x^2+1)\,dx$

```
def f(x):return (x**2+1)
a = float(input('Enter the lower limit:  '))
b = float(input('Enter the upper limit:  '))
n = int(input('Enter the no of intervals:  '))
s1=f(a)+f(b)
h=float(b-a)/n
d=4
for i in range(1,n):
        x=a+i*h
        s1=s1+d*f(x)
        d=6-d

s1=h*s1/3
print ('Integration =',s1)
```

OUTPUT

Enter the lower limit: 0

Enter the upper limit: 1

Enter the no of intervals: 500

Integration = 1.33333333333

Simpson's 1/3 Rule: With Accuracy

```
def f(x):return (x**2+1)
a = float(input('Enter the lower limit:  '))
b = float(input('Enter the upper limit:  '))
s1=f(a)+f(b)
s1_old,sum=0.0,0.0
k=10
while abs(s1_old-s1)> 0.0001:
        s1_old=sum
        h=float(b-a)/k
        d=4
        for i in range(1,k):
            x=a+i*h
            s1=s1+d*f(x)
            d=6-d

        s1=h/3*s1
        sum=s1
        k=k+10

print ('Integration =',s1,'\nNo of steps:',k)
```

OUTPUT(With Accuracy)

Enter the lower limit: 0
Enter the upper limit: 1
Integration = 1.33110802021
No of steps: 260

Integration with pi(π) limit: $\int_0^{\pi/2} \sqrt{sinx}\, dx$

```
import math
def f(x):return math.sqrt(math.sin(x))
a = 0
b = (math.pi)/2
s=0.5*(f(a)+f(b))
k=100
h=float(b-a)/k
for i in range(1,k):
        x=a+i*h
        s=s+f(x)

print ('Integration =',s*h)

#Output:Integration = 1.1977309685212676
```

Simpson's 1/3 Rule Using Integration function from Scipy Module

Using Scipy Module:

- import numpy as np
- import scipy
- from scipy import integrate
- a=float(input("Enter the lower limit:"))
- b=float(input("Enter the upper limit:"))
- n=int(input("Enter the number of interval:"))
- x=np.linspace(a,b,n)
- y=np.sqrt(x**2+1)
- I=scipy.integrate.simps(y,x)
- print("The value of integration:",I)

Enter the lower limit:1
Enter the upper limit:3.5
Enter the number of interval:250
The value of integration:
6.205162829953411

N-point Gauss-Quadrature formula for integration

Quadrature:

- **Quadrature is a method in which the points where the function is evaluated are choosen, so that the formula is exact for polynomials of as high a degree as possible.**

- **The most basic method of quadreture is Gaussian Quadrature**

Formula of Gaussian Quadrature

$$\int_{-1}^{+1} f(x)dx \approx \sum_{i=1}^{n} W_i f(x_i)$$

$$\int_{-1}^{1} f(x)\, dx \approx W_1 f(x_1) + W_2 f(x_2) + W_3 f(x_3) + \cdots$$

- This integration formula is exact for polynomials of degree (2n-1)

Formula of Gaussian Quadrature

Now if the limits of the integration is [a,b] i.e

$$\int_a^b f(x')dx'$$

- Then if we put $x' = \frac{(b-a)x}{2} + \frac{(a+b)}{2}$
- We will get,

$$x = -1 \; at \; x' = a \quad and \; x = 1 \; at \; x' = b$$

$$while \; dx = \frac{(b-a)}{2} dx$$

Formula of Gaussian Quadrature

$$\int_a^b f(x')dx' \approx \int_{-1}^{1} \frac{(b-a)}{2} . f\left(\frac{(b-a)x}{2} + \frac{a+b}{2}\right) dx$$

Python Code Of Gaussian Quadrature

```
import numpy as np

def  f(x):
        return (x**2+1)
a=float(input("Enter the lower limit:"))
b=float(input("Enter the upper limit:"))
n=int(input("Enter the n:"))

x,w=np.polynomial.legendre.leggauss(n)
I=0.0
x_prime=np.zeros(n)
for i in range(n):
            x_prime[i]=x_prime[i]+(b-a)*x[i]/2 + (a+b)/2
            I=I+(b-a)*(w[i]*f(x_prime[i]))/2

print("Integration=",I)
```

OUTPUT

Enter the lower limit:2
Enter the upper limit:3
Enter the n:6
Integration= 7.333333333333335

Gauss Quadrature in Scipy

```
from scipy.integrate import quad
import numpy as np
def f(x):
        return 1/(x**2+1)
I=quad(f,-np.inf,np.inf)
print("Integration=",I)
```

OUTPUT:

integration: (3.141592653589793, 5.155583041103855e-10)