VIII) Computation of area under the curve using Trapezoidal, Simpson's $\frac{1}{3}^{rd}$ and Simpsons $\frac{3}{8}^{th}$ rule

1 a) Evaluate $\int_0^5 \frac{1}{1+x^2}\, dx$ using Trapezoidal Rule

In [4]:
```
def y(x):
    return 1/(1+x**2)

x0= float(input("Enter lower limit of integration:"))
xn= float(input("Enter upper limit of integration:"))
n= int(input("Enter number of sub intervals:"))

def trapezoidal(x0, xn, n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range(1, n):
        k = x0 + i * h
        sum = sum + 2 * y(k)

    integration = sum *h/2 # Finding final integration value
    return integration

print("Integration result by Trapezoidal method is: %0.4f" %trapezoidal(x0, xn, n))
```

```
Enter lower limit of integration:0
Enter upper limit of integration:5
Enter number of sub intervals:10
Integration result by Trapezoidal method is: 1.3731
```

2 b) Evaluate $\int_0^1 \frac{x^2}{1+x^3}\, dx$ using Simpson's $\frac{1}{3}^{rd}$ rule. Take n=6.

In [10]:

```
def y(x):
    return x**2/(1+x**3)

x0= float(input("Enter lower limit of integration:"))
xn= float(input("Enter upper limit of integration:"))
n= int(input("Enter number of sub intervals:"))

def simpson13(x0,xn,n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range (1,n):
        k = x0 + i * h
        if i%2 == 0:
            sum = sum + 2 * y(k)
        else:
            sum = sum + 4 * y(k)

    integration = sum * h * (1/3) # Finding final integration value
    return integration

result = simpson13(x0,xn,n )
print (" Integration result by Simpson 's 1/3 method is: %0.5f" %result)
```

```
Enter lower limit of integration:0
Enter upper limit of integration:1
Enter number of sub intervals:6
 Integration result by Simpson 's 1/3 method is: 0.23106
```

Finding gradient ,divergence and curl

To find gradient of phi=x^2y+2xz-4

In [15]:
```
from sympy.vector import*
from sympy import symbols
x,y,z=symbols('x,y,z')
N=CoordSys3D('N')
A=N.x**2*N.y+2*N.x*N.z-4
print("\n Gradient is:")
display(gradient(A))
```

```
 Gradient is:
```

$$\left(2\mathbf{x_N}\mathbf{y_N} + 2\mathbf{z_N}\right)\hat{\mathbf{i}}_\mathbf{N} + \left(\mathbf{x_N}^2\right)\hat{\mathbf{j}}_\mathbf{N} + \left(2\mathbf{x_N}\right)\hat{\mathbf{k}}_\mathbf{N}$$

To find divergence of F=x^2yzî +Y^2zxĵ +z^2xyk̂

In [20]:
```
from sympy.vector import*
from sympy import symbols
x,y,z=symbols('x,y,z')
N=CoordSys3D('N')
A=N.x**2*N.y*N.z*N.i+N.y**2*N.z*N.x*N.j+N.z**2*N.x*N.y*N.k
print("\n Divergence is:")
display(divergence(A))
```

```
 Divergence is:
```

$$6\mathbf{x_N}\mathbf{y_N}\mathbf{z_N}$$

To find curl of F=x^2yzî +Y^2zxĵ +z^2xyk̂

In [21]:
```
from sympy.vector import*
from sympy import symbols
x,y,z=symbols('x,y,z')
N=CoordSys3D('N')
A=N.x**2*N.y*N.z*N.i+N.y**2*N.z*N.x*N.j+N.z**2*N.x*N.y*N.k
print("\n curl is:")
display(curl(A))
```

```
 curl is:
```

$$\left(-\mathbf{x_N}\mathbf{y_N}^2 + \mathbf{x_N}\mathbf{z_N}^2\right)\hat{\mathbf{i}}_\mathbf{N} + \left(\mathbf{x_N}^2\mathbf{y_N} - \mathbf{y_N}\mathbf{z_N}^2\right)\hat{\mathbf{j}}_\mathbf{N} + \left(-\mathbf{x_N}^2\mathbf{z_N} + \mathbf{y_N}^2\mathbf{z_N}\right)\hat{\mathbf{k}}_\mathbf{N}$$

X) Solution of ODE of first order and first degree.

1) Runge-Kutta method

1a) Apply the Runge Kutta method to find the solution of $\frac{dy}{dx} = 1 + \frac{y}{x}$ at $y(2)$ taking $h = 0.2$. Given that $y(1) = 2$.

In [1]:
```
from sympy import *
import numpy as np
def RungeKutta(g,x0,h,y0 ,xn):
    x,y= symbols('x,y')
    f=lambdify([x,y],g)
    xt=x0+h
    Y=[y0]
    while xt<=xn:
        k1=h*f(x0 ,y0)
        k2=h*f(x0+h/2, y0+k1/2)
        k3=h*f(x0+h/2, y0+k2/2)
        k4=h*f(x0+h, y0+k3)
        y1=y0+(1/6)*(k1+2*k2+2*k3+k4)
        Y.append(y1)
        x0=xt
        y0=y1
        xt=xt+h
    return np.round(Y,2)
RungeKutta('1+(y/x)',1,0.2,2,2)
```

Out[1]:
```
array([2.  , 2.62, 3.27, 3.95, 4.66, 5.39])
```

2a) Solve $\frac{dy}{dx} = x^2 + (y/2)$ at $y(1.4)$.

Given that $y(1) = 2$, $y(1.1) = 2.2156$, $y(1.2) = 2.4649$, $y(1.3) = 2.7514$.

In [6]:
```
x0=1
h=0.1
x1=x0+h
x2=x1+h
x3=x2+h
x4=x3+h

y0=2
y1=2.2156
y2=2.4649
y3=2.7514


def f(x,y):
    return x ** 2+(y/2)

f0 =f(x0,y0)
f1 =f(x1,y1)
f2 =f(x2,y2)
f3 =f(x3,y3)
y4p =y0+(4*h/3)*(2*f1-f2+2*f3)
print ('predicted value of y4 is %3.3f '%y4p)
f4 =f(x4 ,y4p )
for i in range (1,4):
    y4c=y2+(h/3)*(f2+4*f3 +f4)
    print ('corrected value of y4 after \t iteration %d is \t %3.5f\t '%(i,y4c))
    f4=f(x4 ,y4c)
```

```
predicted value of y4 is 3.079
corrected value of y4 after 	 iteration 1 is 	 3.07940
corrected value of y4 after 	 iteration 2 is 	 3.07940
corrected value of y4 after 	 iteration 3 is 	 3.07940
```

II) Evaluation of β and Γ functions

1a) Evaluate $\int_0^\infty e^{-x}\ dx$

```
from sympy import *
x= Symbol('x')
u=integrate(exp(-x),(x,0,oo))
print(u)
```

```
1
```

1b) Evaluate $\int_0^\infty e^{-t} cos(2t)\ dt$

```
from sympy import integrate
t= Symbol('t')
u=integrate(exp(-t)*cos(2*t),(t,0,oo))
print(u)
```

```
1/5
```

1c) Evaluate $\Gamma(5)$ using the definition $\Gamma(5) = \int_0^\infty e^{-x} x^4\ dx$

```
from sympy import *
x= symbols('x')
Gamma= integrate(exp(-x)*x**4,(x,0,float('inf')))
print(simplify(Gamma))
```

```
24
```

2a) Find $\beta(3, 5)$ and $\Gamma(5)$

In [6]:
```
from sympy import beta, gamma
m=float(3)
n=float(5)
beta3_5= beta(m,n)
Gamma5=gamma(5)
print("Beta(3,5)=",beta3_5)
print("Gamma(5)=", Gamma5)
```

```
Beta(3,5)= 0.00952380952380952
Gamma(5)= 24
```

2b) Find $\beta(\frac{5}{2}, \frac{7}{2})$ and $\Gamma(\frac{5}{2})$

In [7]:
```
from sympy import beta, gamma
m=float(5/2)
n=float(7/2)
beta_value= beta(m,n)
gamma_value=gamma(m)
print("Beta(5/2,7/2)=",beta_value)
print("Gamma(5/2)=",gamma_value)
```

```
Beta(5/2,7/2)= 0.0368155389092554
Gamma(5/2)= 1.32934038817914
```

2c) Verify Beta and Gamma relationship

In [8]:
```
from sympy import beta , gamma
m=5
n=7
m= float(m)
n= float(n)
s= beta(m,n)
t=(gamma(m)*gamma(n))/gamma(m+n)
print(s,t)
if(s==t):
    print('Beta and Gamma are related.')
else:
    print ('Given values are wrong.')
```

```
0.000432900432900433 0.000432900432900433
Beta and Gamma are related.
```

1 d) Evaluate $\int_0^{log2} \int_0^x \int_0^{x+log(y)} e^{x+y+z} \, dz \, dy \, dx$

```
from sympy import *
x,y,z= symbols ('x y z')
V= integrate(exp(x+y+z),(z,0,x+log(y)) ,(y,0,x) ,(x,0,log(2)))
print(V)
```

```
-19/9 + 8*log(2)/3
```

2 a) Find the area of an ellipse by double integration $A = 4 \int_0^a \int_0^{\frac{b}{a}\sqrt{a^2-x^2}} \, dy \, dx$

```
from sympy import *
x,y= symbols('x y')
a=4
b=6
w=4*integrate(1,(y,0,(b/a)*sqrt(a**2-x**2)),(x,0,a))
print(w)
```

```
24.0*pi
```

2 b) Find the area of positive quadrant of the circle $x^2 + y^2 = 16$

```
from sympy import *
x,y= symbols('x y')
w=integrate(1,(y,0,sqrt(16-x**2)),(x,0,4))
print(w)
```

```
4*pi
```

2c) Find the area of the cardioid $r = a(1 + cos\theta)$ by double integration.

```
from sympy import *
r= Symbol ('r')
t= Symbol ('t')
a= Symbol ('a')
Area=2*integrate(r,(r,0,a*(1+cos(t))),(t,0,pi))
display(Area)
```

$$\frac{3\pi a^2}{2}$$

2 a) Evaluate $\int_0^5 \frac{1}{1+x^2}\, dx$ using Simpson's $\frac{1}{3}^{rd}$ rule.

In [3]:
```
def y(x):
    return 1/(1+x**2)

x0= float(input("Enter lower limit of integration:"))
xn= float(input("Enter upper limit of integration:"))
n= int(input("Enter number of sub intervals:"))

def simpson13(x0,xn,n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range (1,n):
        k = x0 + i * h
        if i%2 == 0:
            sum = sum + 2 * y(k)
        else:
            sum = sum + 4 * y(k)

    integration = sum * h * (1/3) # Finding final integration value
    return integration

result = simpson13(x0,xn,n )
print (" Integration result by Simpson 's 1/3 method is: %0.6f" % result)
```

```
Enter lower limit of integration:0
Enter upper limit of integration:5
Enter number of sub intervals:10
 Integration result by Simpson 's 1/3 method is: 1.371454
```

I) Program to compute Area, Surface Area and Volume

1 a) Evaluate the integral $\int_0^1 \int_0^x (x^2 + y^2)\, dy\, dx$

```
from sympy import *
x,y=symbols ('x y')
w= integrate (x**2+y**2,(y,0,x) ,(x,0,1))
print(w)
```

```
1/3
```

1 b) Evaluate the integral $\int_0^1 \int_0^x (x + y)\, dy\, dx$

```
from sympy import *
x,y= symbols('x y')
I= integrate (x+y,(y,0,x),(x,0,1))
print(I)
```

```
1/2
```

1 c) Evaluate $\int_0^3 \int_0^{3-x} \int_0^{3-x-y} xyz\, dz\, dy\, dx$

```
from sympy import *
x,y,z= symbols('x y z')
V= integrate ((x*y*z),(z,0,3-x-y),(y,0,3-x),(x,0,3))
print(V)
```

```
81/80
```

3 b) Evaluate $\int_0^{0.6} e^{-x^2}\,dx$ using Simpson's $\frac{3}{8}^{th}$ rule by taking seven ordinates.

In [11]:
```
from sympy import *
def y(x):
    return exp(-x**2)

x0= float(input("Enter lower limit of integration:"))
xn= float(input("Enter upper limit of integration:"))
n= int(input("Enter number of sub intervals:"))

def simpson38(x0,xn,n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range (1,n):
        k = x0 + i * h
        if i%3 == 0:
            sum = sum + 2 * y(k)
        else:
            sum = sum + 3 * y(k)

        integration = sum * h * (3/8) # Finding final integration value
    return integration

result = simpson38(x0,xn,n)
print("Integration result by Simpson's 3/8 th method is: %0.4f" %result)
```

```
Enter lower limit of integration:0
Enter upper limit of integration:0.6
Enter number of sub intervals:6
Integration result by Simpson's 3/8 th method is: 0.5352
```

3 a) Evaluate $\int_0^5 \frac{1}{1+x^2}\, dx$ using Simpson's $\frac{3}{8}^{th}$ rule

In [9]:

```
def y(x):
    return 1/(1+x**2)

x0= float(input("Enter lower limit of integration:"))
xn= float(input("Enter upper limit of integration:"))
n= int(input("Enter number of sub intervals:"))

def simpson38(x0,xn,n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range (1,n):
        k = x0 + i * h
        if i%3 == 0:
            sum = sum + 2 * y(k)
        else:
            sum = sum + 3 * y(k)

        integration = sum * h * (3/8) # Finding final integration value
    return integration

result = simpson38(x0,xn,n)
print (" Integration result by Simpson's 3/8 th method is: %0.5f" %result)
```

```
Enter lower limit of integration:0
Enter upper limit of integration:5
Enter number of sub intervals:10
 Integration result by Simpson's 3/8 th method is: 1.36212
```

computing the inner product and orthogonality

1) find the inner product of the vectors (2,1,5,4) and (3,4,7,8)

In [1]:
```
import numpy as np
A=np.array([2,1,5,4])
B=np.array([3,4,7,8])
output=np.dot(A,B)
print("Inner product of the vectors is",output)
```

```
Inner product of the vectors is 77
```

2) verify the vectors (2,1,5,4) and (3,4,7,8) are orthogonal

In [2]:
```
import numpy as np
A=np.array([2,1,5,4])
B=np.array([3,4,7,8])
output=np.dot(A,B)
print("Inner product of the vectors is",output)
if output==0:
    print("Given vectors are orthogonal")
else:
    print("Given vectors are not orthogonal")
```

```
Inner product of the vectors is 77
Given vectors are not orthogonal
```

3 a) Find the volume of the tetrahedron bounded by the planes $x = 0, y = 0$ and $z = 0$, $\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1$

```
]: from sympy import *
x,y,z,a,b,c= symbols('x y z a b c')
Volume= integrate (1,(z,0,c*(1-x/a-y/b)),(y,0,b*(1-x/a)),(x,0,a))
display(Volume)
```

$$\frac{abc}{6}$$

3 b) Find the volume of the tetrahedron bounded by the planes $x = 0, y = 0$ and $z = 0$, $\frac{x}{2} + \frac{y}{3} + \frac{z}{4} = 1$

```
]: from sympy import *
x,y,z= symbols('x y z')
Volume= integrate (1 ,(z,0,4*(1-x/2-y/3)) ,(y,0,3*(1-x/2)) ,(x,0,2))
print(Volume)
```

4

2) find the dimension of the subspace spanned by the vectors (1,2,3),(2,3,1) and (3,1,2)

In [6]:
```
import numpy as np
V=np.array([[1,2,3],[2,3,1],[3,1,2]])
dimension=np.linalg.matrix_rank(V)
print("Dimension of the subspace spanned by the vector is",dimension)
```

```
Dimension of the subspace spanned by the vector is 3
```

1 b) Evaluate $\int_0^\pi sin^2 x \; dx$ using Trapezoidal Rule. Take n=6

In [5]:
```
from sympy import *
def y(x):
    return sin(x)**2

x0= 0
xn= pi
n= 6

def trapezoidal(x0, xn, n):
    h =(xn - x0)/n
    sum =y(x0) + y(xn)

    for i in range(1, n):
        k = x0 + i * h
        sum = sum + 2 * y(k)

    integration = sum * h / 2 # Finding final integration value
    return integration

print("Integration result by Trapezoidal method is: %0.4f" %trapezoidal(x0, xn, n))
```

```
Integration result by Trapezoidal method is: 1.5708
```