Mathematics-II for CV/ME stream Lab (MVJ22MATC21/M21)

TABLE OF CONTENTS

LAB 1: Programme to comp... gravity

1.1 Objectives:

Use python

1. to evaluate double integration.
2. to compute area and volume.
3. to calculate center of gravity of 2D object.

Syntax for the commands used:

1. Data pretty printer in Python:
   pprint ()
2. integrate:
   integrate (function ,(variable , min_limit , max_limit))

1.2 Double and triple integration

Example 1:

Evaluate the integral $\int_0^1 \int_0^x (x^2 + y^2)dydx$

```
from sympy import *
x ,y , z= symbols ('x y z')
w1= integrate ( x ** 2+y ** 2 ,( y ,0 , x ) ,(x ,0 , 1 ) )
print ( w1 )
```

Example 2:

Evaluate the integral $\int_0^3 \int_0^{3-x} \int_0^{3-x-y} (xyz)dxdydz$

```
from sympy import *
x= Symbol ('x')
y= Symbol ('y')
z= Symbol ('z')
w2= integrate (( x*y*z ) ,(z ,0 , 3-x-y ) ,(y ,0 , 3-x ) ,(x ,0 , 3 ) )
print ( w2 )
```

Example 3:
Prove that $\int\int(x^2+y^2)dydx = \int\int(x^2+y^2)dxdy$

```
from sympy import *
x= Symbol ('x')
y= Symbol ('y')
z= Symbol ('z')
w3= integrate ( x ** 2+y ** 2 ,y , x )
pprint ( w3 )
w4= integrate ( x ** 2+y ** 2 ,x , y )
pprint ( w4 )
```

1.3 Area and Volume

Area of the region R in the cartesian form is $\iint_R dxdy$

Example 4:

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

```
from sympy import *
x= Symbol ('x')
y= Symbol ('y')
#a= Symbol ('a ')
#b= Symbol ('b ')
a=4
b=6
w3=4* integrate (1 ,( y ,0 ,( b/a )* sqrt ( a ** 2-x ** 2 )),(x ,0 , a ))
print ( w3 )
```

1.4 Area of the region R in the polar form is $\iint_R rdrd\theta$

Example 5:
Find the area of the cardioid r = a(1 + cosθ) by double integration

```
from sympy import *
```

```
r= Symbol ('r')
t= Symbol ('t')
a= Symbol ('a')
#a=4
w3=2* integrate (r ,( r ,0 , a*( 1+cos ( t ) ) ) ,(t ,0 ,pi) )
pprint ( w3 )
```

1.5 Volume of a solid is given by $\iiint_v dxdydz$

Example 6:
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= Symbol ('x')
y= Symbol ('y')
z= Symbol ('z')
a= Symbol ('a')
b= Symbol ('b')
c= Symbol ('c')
w2= integrate (1 ,( z ,0 , c*( 1-x/a-y/b ) ) ,(y ,0 , b*( 1-x/a ) ) ,(x ,0 , a ) )
print ( w2 )
```

1.6 Center of Gravity

Example 7:
Find the center of gravity of cardioid . Plot the graph of cardioid and mark the center of gravity.

```
import numpy as np
import matplotlib . pyplot as plt
import math from sympy import *
r= Symbol ('r')
t= Symbol ('t')
a= Symbol ('a')
```

```
I1= integrate (cos ( t )*r ** 2 ,( r ,0 , a*( 1+cos ( t ) ) ) ,(t ,-pi ,pi) )
I2= integrate (r ,( r ,0 , a*( 1+cos ( t ) ) ) ,(t ,-pi ,pi) )
I=I1/I2
print ( I )
I=I . subs (a , 5 )
plt . axes ( projection = 'polar ')
a=5
rad = np . arange (0 , ( 2 * np .pi) , 0 . 01 )
# plotting the cardioid
for i in rad :
    r = a + ( a*np .cos( i ) )
    plt . polar (i ,r ,'g.')
    plt . polar (0 ,I ,'r.')
    plt . show ()
```

LAB 2: Evaluation of improper integrals, Beta and Gamma functions

2.1Objectives:

Use python

1. to evaluate improper integrals using Beta function.
2. to evaluate improper integrals using Gamma function.

Syntax for the commands used:

1. gamma
   math . gamma (x)
   Parameters :
   x : The number whose gamma value needs to be computed.
2. beta
   math . beta (x , y)
   Parameters :
   x ,y: The numbers whose beta value needs to be computed.

Example 1:
Evaluate $\int_0^{\infty} e^{-x} dx$

```
from sympy import *
x= symbols ('x')
w1= integrate (exp (-x ) ,(x ,0 , float ('inf') ) )
print ( simplify ( w1 ) )
```

Gamma function:

Gamma function is $\Gamma(n) = \int_0^{\infty} e^{-x} x^{n-1} dx$

Example 2:
Evaluate $\Gamma(5)$ by using definition

```
from sympy import *
x= symbols ('x')
w1= integrate (exp (-x )*x ** 4 ,( x ,0 , float ('inf') ) )
print ( simplify ( w1 ) )
```

Example 3:
Find Beta(3,5), Gamma(5)

```
# beta and gamma functions
from sympy import beta , gamma
m= input ('m :') ;
n= input ('n :') ;
m= float ( m ) ; n
= float ( n ) ;
s= beta (m , n ) ;
t= gamma ( n )
print ('gamma (',n ,') is %3.3f '%t )
print ('Beta (',m ,n ,') is %3.3f '%s )
```

Example 5:
Calculate Beta(5/2,7/2) and Gamma(5/2).

```
# beta and gamma functions
# If the number is a fraction give it in decimals . Eg 5/2=2.5
from sympy import beta , gamma
m= float ( input ('m : ') );
n= float ( input ('n :') );
s= beta (m , n );
t= gamma ( n )
print ('gamma (',n ,') is %3.3f'%t )
print ('Beta (',m ,n,') is %3.3f'%s )
```

Example 6:
Verify that Beta(m, n) = Gamma(m)Gamma(n)/Gamma(m + n) for m=5 and n=7

```
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 (abs ( s-t )<=0 . 00001 ):
        print ('beta and gamma are related ')
else :
        print ('given values are wrong ')
```

LAB 3: Finding gradient, divergent, curl and their geometrical interpretation and Verification of Green's theorem.

3.1 Objectives:

Use python

1. to find the gradient of a given scalar function.

2. to find find divergence and curl of a vector function.
3. to evaluate integrals using Green's theorem.

1. To find gradient of $\emptyset = x^2y + 2xz - 4$.

```
#To find gradient of scalar point function.
from sympy.vector import *
from sympy import symbols
N= CoordSys3D ('N')                          # Setting the coordinate system
x ,y , z = symbols ('x y z')
A=N.x **2*N.y+2*N.x*N.z-4                    # Variables x, y, z to be used with coordinate
system N
delop =Del()                                 #Del operator
display(delop ( A ) )                        #Del operator applied to A
gradA = gradient ( A )                       # Gradient function is used
print ( f"\n Gradient of {A} is \n")
display(gradA )
```

2. To find divergence of $\vec{F} = x^2yz\,\hat{\imath} + y^2zx\,\hat{\jmath} + z^2xy\,\hat{k}$.

```
#To find divergence of a vector point function
from sympy.vector import *
from sympy import symbols
N= CoordSys3D ('N')
x ,y , z= symbols ('x y z')
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
delop =Del ()
divA = delop.dot (A)
display(divA )
print( f"\n Divergence of {A} is \n")
display(divergence(A))
```

3. To find curl of $\vec{F} = x^2yz\,\hat{\imath} + y^2zx\,\hat{\jmath} + z^2xy\,\hat{k}$.

```
#To find curl of a vector point function
```

```
from sympy . vector import *
from sympy import symbols
N= CoordSys3D ('N')
x ,y , z= symbols ('x y z')
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
delop =Del ()
curlA = delop.cross(A)
display(curlA)
print(f"\n Curl of {A} is \n")
display(curl(A))
```

LAB 4: Verification of Green's theorem

1.1 Objectives:

Use python

1. to evaluate integrals using Green's theorem.

1.2 Green's theorem

Statement of Green's theorem in the plane:

If $P(x, y)$ and $Q(x, y)$ be two continuous functions having continuous partial derivatives in a region R of the xy-plane, bounded by a simple closed curve C, then

$$\oint(Pdx + Qdy) = \int\int_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dxdy.$$

1. **Using Green's theorem, evaluate $\oint[(x + 2y)dx + (x - 2y)dy]$, where c is the region bounded by coordinate axes and the line x = 1 and y = 1.**

```
from sympy import *
var ('x,y')
p=x+2*y q=x-2*y
f= diff(q ,x)- diff (p,y)
soln=integrate (f ,[x ,0 , 1],[y ,0 , 1])
print("I=", soln)
```

2. **Using Green's theorem, evaluate $\oint[(xy+y^2)dx+x^2dy]$, where c is the closed curve bounded by $y=x$ and $y=x^2$.**

```
from sympy import *
var ('x,y')
p=x*y+y ** 2
q=x ** 2
f= diff (q,x)- diff (p,y)
soln = integrate (f,[y,x ** 2,x],[x,0,1])
print ("I=",soln )
```

LAB 5: Solution of Lagrange's linear partial differential equations

1.1 Objectives:

Use python

1. to solve linear Partial Differential Equations of first order

1. **Solve the PDE, $xp + yq = z$, where $z = f(x,y)$**

```
from sympy . solvers .pde import pdsolve
from sympy import Function , Eq ,cot , classify_pde , pprint
from sympy .abc import x, y, a
f = Function ('f')
z = f(x, y)
zx = z. diff (x)
zy = z. diff (y)
# Solve xp+yq=z
eq = Eq(x*zx+y*zy , z)
pprint (eq)
print ("\n")
soln = pdsolve (eq ,z)
pprint ( soln )
```

2. Solve the PDE $2p + 3q = 1$, where $p = \frac{\partial z}{\partial x}$ and $q = \frac{\partial z}{\partial y}$

```
from sympy . solvers .pde import pdsolve
from sympy import Function , Eq ,cot , classify_pde , pprint
from sympy .abc import x, y, a
f = Function ('f')
z = f(x, y)
zx = z. diff (x)
zy = z. diff (y)
# Solve 2p+3q=1
eq = Eq(2*zx+3*zy , 1)
pprint (eq)
print ("\n")
soln = pdsolve (eq ,z)
pprint ( soln )
```

3. Solve the PDE $x^2p + y^2q = (x + y)z$, where $p = \frac{\partial z}{\partial x}$ and $q = \frac{\partial z}{\partial y}$

```
from sympy . solvers .pde import pdsolve
from sympy import Function , Eq ,cot , classify_pde , pprint
from sympy .abc import x, y, a
f = Function ('f')
z = f(x, y)
zx = z. diff (x)
zy = z. diff (y)
# Solve x^2p+y^2q =(x+y)z
eq=Eq(x ** 2*zx+y ** 2*zy ,(x+y)*z)
pprint (eq)
print ("\n")
soln = pdsolve (eq ,z)
pprint ( soln )
```

LAB 6: Solution of algebraic and transcendental equation by Regula-Falsi and Newton-Raphson method

6.1 Objectives:

Use python

1. to solve algebraic and transcendental equation by Regula-Falsi method.
2. to solve algebraic and transcendental equation by Newton-Raphson method.

6.2 Regula-Falsi method to solve a transcendental equation.

1. **Obtain a root of the equation $x^3 - 2x - 5 = 0$ between 2 and 3 by regula-falsi method. Perform 5 iterations.**

```
# Regula Falsi method
from sympy import *
x= Symbol ('x')
g = input ('Enter the function ')                    #%x^3-2*x-5; % function
f= lambdify (x, g)
a= float (input ('Enter a value:'))                  #2
b= float (input ('Enter b value:'))                  # 3
N=int (input ('Enter number of iterations:'))              #5
for i in range (1, N+1):
  c = (a*f(b)-b*f(a))/ (f(b)-f(a))
  if((f(a)*f(c)) <0):
        b=c
  else:
        a=c
  print ('iteration %d \t the root %0.3f\t function value %0.3f\n'% (i, c, f(c)));
```

Using tolerance value, we can write the same program as follows:

1. **Obtain a root of the equation $x^3 - 2x - 5 = 0$ between 2 and 3 by regula-falsi method. Correct to 3 decimal places.**

```
# Regula Falsi method
from sympy import *
```

```
x= Symbol ('x')
g = input ('Enter the function ')                #%x^3-2*x-5; % function
f= lambdify (x, g)
a= float (input ('Enter a value:'))              #2
b= float (input ('Enter b value:'))              # 3
N=float (input ('Enter tolerance:'))             # 0.0001
x=a;
c=b;
i=0
while (abs(x-c)>=N):
    x=c
    c = (a*f (b)-b*f (a)) / (f (b)-f (a))
    if ((f (a)*f (c)) <0):
        b=c
    else:
        a=c
        i=i+1
    print ('iteration %d \t the root %0.3f\t function value %0.3f\n'% (i, c, f(c)));
print ('final value of the root is %0.5f' %c)
```

6.3 Newton-Raphson method to solve a transcendental equation.

1. **Find a root of the equation $3x = \cos x + 1$, near 1, by Newton Raphson method. Perform 5 iterations.**

```
from sympy import *
x= Symbol ('x') g = input ('Enter the function ')      #%3x -cos(x)-1; % function
f= lambdify (x, g)
dg = diff (g);
df= lambdify x, dg)
x0= float (input ('Enter the initial approximation '));
n= int (input ('Enter the number of iterations '));    # x0=1
for i in range (1, n+1):                               #n=5;
    x1= (x0 - (f (x0)/df (x0)))
    print ('iteration %d \t the root %0.3f \t function value %0.3f \n'% (i, x1, f (x1)));   x0=x1
```