KSSEM

K S SCHOOL OF ENGINEERING AND MANAGEMENT

K.S. GROUP OF INSTITUTIONS

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT

15, Mallasandra, Near Vajarahalli, Off. Kanakapura Road, Bengaluru - 560 109.

PRACTICAL RECORD BOOK

Name : Sneha. K.S

Class : I Sem

Lab : Mathematics Lab

USN : 1 K 23 CS 105

K.S. GROUP OF INSTITUTIONS

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT

15, Mallasandra, Near Vajarahalli, Off. Kanakapura Road, Bengaluru - 560 109.

KSSEM
K.S SCHOOL OF ENGINEERING AND MANAGEMENT

Laboratory Certificate

This is to certify that

Mr. / Ms. Sneha. K. S

has satisfactorily completed the course of experiment in Maths lab *Laboratory. Code* BMATS201

Prescribed by Visvesvaraya Technological University, Belagavi for the 2nd *Semester B.E.* CSE *branch in this college during in the academic year 20*23*- 20*24*.*

Name of the Candidate : Sneha. K. S

USN : 1KG23CS105 Lab (with code) BMATS201

Marks	
Maximum	Obtained
15	15

Signature of the Teacher 8/6

Date : 8/6/24

Head of the Department

Dr. C. VASUDEV
Professor & HOD
Department of Applied Science
K.S. School of Engineering & Management
Bangalore - 560 109

Particulars of the Experiments Performed

CONTENTS

5/5

output :-

f(38) = 180.24

f(85) = 289.75

DATE 15/03/2024 EXP. NO.

EXPT. TITLE : Interpolation / Extrapolation using newton's forward and Newton's Backward interpolation formula.

Lab 7 :- Interpolation / Extrapolation using Newton's forward and Newton's Backward Interpolation formula.

objectives :- use python

* To Interpolate using Newton's forward interpolation formula.
* To Interpolate using Newton's Backward interpolation formula.

1. A) Given f(40) = 184, f(50) = 204, f(60) = 226, f(70) = 250, f(80) = 276, f(90) = 304 find f(38) and f(85) using Newton's forward and Newton's Backward interpolation

```
import numpy as np
x = np.array([40, 50, 60, 70, 80, 90])
y = np.array([184, 204, 226, 250, 276, 304])
coefficients = np.polyfit(x, y, deg = len(x)-1)
def interpolate(x_val):
    return np.polyval(coefficients, x_val)
x_val_1 = 38
x_val_2 = 85
f_val_1 = interpolate(x_val_1)
f_val_2 = interpolate(x_val_2)
print("f(38) = ", f_val_1)
print("f(85) = ", f_val_2)
```

output:-

f(105)= 8666.00

DATE ..... EXP. NO. .....

EXPT. TITLE :

PAGE NO. 02

B) Given f(80) = 5026, f(85)= 5674, f(90)= 6362, f(95)= 7088, f(100)= 7854. Find f(105) using newton's Backward interpolation method.

```
import numpy as np
x= np. array ([80,85,90,95,100])
y= np. array ([[ 5026, 5674, 6362, 7088, 7854]])
Coefficients = np. polyfit ( x, y, deg = len (x)-1)
def interpolate (x_val):
    return np. polyval (coefficients, x_val)
x_val = 105
f_val = interpolate (x_val)
print (" f(105) =", f_val)
```

DATE 22-03-24
EXP. NO.
EXPT. TITLE Lab 6 :- Solution of algebraic and transcendental Equation by regula-Falsi and Newton-Raphson method.

Lab 6 :- Solution of algebraic and transcendental Equation by regula-Falsi and Newton-Raphson method.

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.

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.

```
from sympy import *
x = Symbol('x')
g = input('enter the function')
f = lambdify(x, g)
a = float(input('enter a value :'))
b = float(input('enter b value :'))
N = int(input('enter number of iterations :'))

for i in range(1, N+1):
    c = (a * f(b) - b * f(a))/(f(b) - f(a))
    if((f(a) * f(c) < 0)):
```

output :-

```
enter the function  x **3-2*x-5
enter a value : 2
enter b value : 3
enter number of iterations : 5
```

iteration 1	the root 2.059	function value -0.391
iteration 2	the root 2.081	function value -0.147
iteration 3	the root 2.090	function value -0.055
iteration 4	the root 2.093	function value -0.020
iteration 5	the root 2.094	function value -0.007.

output :-

```
enter the function  3*x-cos(x)-1
enter the initial approximation : 1
enter the number of iteration : 5
```

iteration 1	the root 0.620	function value 0.046
iteration 2	the root 0.607	function value 0.000
iteration 3	the root 0.607	function value 0.000
iteration 4	the root 0.607	function value 0.000
iteration 5	the root 0.607	function value -0.000

DATE ..... EXP. NO. ..... EXPT. TITLE : PAGE NO 04

```
        b = c
    else :
        a = c
    print('iteration %d \t the root %.3f \t function value %.3f \n' %(i, c, f(c)));
```

Newton - Raphson Method to solve a transcendatal equation.

1] Find a root of the equation $3x = \cos x + 1$, near 1, by newton raphson method. perform 5 iteration.

```
from sympy import *
x = symbol('x')
g = input('enter the 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 :'))
for i in range(1, n+1):
    x1 = (x0-((f(x0)/df(x0)))
    print('iteration %d \t the root %.3f \t function value %.3f \n' %(i, x1, f(x1)));
    x0 = x1
```

DATE 23/03/24
EXP. NO. 3
EXPT. TITLE: Computation of area under Curve using trapazoidal rule and Simpson's 1/3th rule and Simpson's 3/8th rule.

Lab 8:- Computation of area under curve using trapazoidal Rule and Simpson's $1/3^{th}$ rule and Simpson's $3/8^{th}$ rule.

8.1 objectives

use python

1] To find the area under the curve represented by a function using trapazoidal rule.

2] To find the area under the curve represented by a function using Simpson's $1/3^{rd}$ and Simpson's $3/8^{th}$ rule.

1] Solve $\int_0^1 \frac{1}{1+x^2} dx$ by Simpson's $1/3^{rd}$ rule by taking 4 equal parts.

```
def f(x):
    return 1/(1+x**2)
def simpsons_1_3_rule(f,a,b,n):
    h=(b-a)/n
    sum_odd=0
    sum_even=0
    for i in range(1,n,2):
        sum_odd+=f(a+i*h)
    for i in range(2,n,2):
        sum_even+=f(a+i*h)
    integral=(h/3)*(f(a)+4*sum_odd+2*sum_even+f(b))
    return integral.
a=0
b=1
```

output:-

integration of 1/(1+x^2) from 0 to 1 using Simpson's 1/3rd rule 4 equal parts

result: 0.7853921568627455


```
n=4
integral = Simpsons_1_3_rule(f,a,b,n)
print("integration of 1/(1+x^2) from 0 to 1 using simpson's 1/3rd rule with 4 equal parts")
print("Result:", integral)
```

2] Solve $\int_0^3 \frac{1}{(1+x)^2} dx$ by Simpsons 3/8th rule by taking 3 equal parts.

```
def f(x):
    return 1/(1+x)**2
def simpson_3_8_rule(f,a,b,n):
    h=(b-a)/n
    sum_term1=0
    sum_term2=0
    for i in range(1,n):
        x=a+i*h
        if i%3==0:
            sum_term2+=f(x)
        else:
            sum_term1+=f(x)
    integral=(3*h/8)*(f(a)+3*sum_term1+2*sum_term2+f(b))
    return integral
a=0
b=3
n=3
integral=Simpsons_3_8_rule(f,a,b,n)
print("Integration of 1/(1+x)^2 from 0 to 3 using simpson's
```

output:-

Integration of $1/(1+x^2)$ from 0 to 3 using simpsons 3/8rd rule with 3 equal parts

Result : 0.8046874999999999

output:-

x = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

y = [1.0, 0.99, 0.962, 0.917, 0.862, 0.8, 0.735, 0.671, 0.552, 0.5]

The value of the integral is 0.785.

DATE ..... EXP. NO. .....

EXPT. TITLE :

PAGE NO. 07

3/8th rule with 3 equal parts :")

print("result :", integral)

3] Solve by trapezoidal rule.

```
def trap (x0, x1, n):
    def f(x):
        return 1/(1+x**2)
    h = (x1-x0)/n
    x = []
    y = []
    trap = 0
    for i in range (n+1):
        x.append(round(x0+i*h),3))
        y.append(round(f(x0+i*h),3)
        if i==0 or i==n:
            trap = trap + f(x0+i*h)/2*h
        else:
            trap = trap + f(x0+i*h)*h
    print('x=', x)
    print('y =', y)
    print('the value of the integral is', round(trap,4))
trap(0,1,10)
```

23/3/24

output :-

Taylors Series is: 4*x**3/3 + 2**x**2 + x + 1

y(0.1) = 1.1113

y(0.2) = 1.2507.

DATE 05/04/2024

EXP. NO. 9

EXPT. TITLE lab 9:- Solution of ordinary differential equation of first order and first degree by taylor's series and modified ~~euler's method~~.

lab 9 :- Solution of ODE of ordinary differential equation of first order and first degree by taylor's series method and modified Euler's method.

Objectives 9.1 :-

- To solve ODE by taylor's series method
- To solve ODE by modified Euler's method.

1] From taylor's series method, find $y(0.1) = 1$ considering up to 4th degree term for $\frac{dy}{dx} = x^2 + y^2$, at $x = 0.1, 0.2$ given $x_0 = 0$, $y_0 = 1$ upto the third degree terms.

```
from sympy import *
x,y = symbols('x,y')
x0, x1, x2 = 0, .1, .2
yd0 = 1
yd1 = x**2 + y**2
yd2 = 2*x + 2*y * yd1
yd3 = 2+2*y*yd2 + yd1*2*yd1
yd1 = yd1.subs({x:0, y:1})
yd2 = yd2.subs({x:0, y:1})
yd3 = yd3.subs({x:0, y:1})
ts = yd0+(x-x0)*yd1 + (x-x0)**2 * yd2/2 + (x-x0)**3 * yd3/6
tsx1 = ts.subs(x, x1)
tsx2 = ts.subs(x, x2)
print('Taylors series is', ts)
print('y(', x1,') =', round(tsx1, 4))
print('y(', x2,') =', round(tsx2, 4))
```

Output :-

y(0.1) = 0.9

The 1 th improved value of y(0.1) = 0.9145

The 2 th improved value of y(0.1) = 0.9132

The 3 th improved value of y(0.1) = 0.9133

y(0.2) = 0.8399

The 1 th improved value of y(0.2) = 0.8563

the 2 th improved value of y(0.2) = 0.8549

The 3 th improved value of y(0.2) = 0.8551

DATE ..... EXP. NO. .....

EXPT. TITLE :

PAGE NO. 09

2] Modified Euler's method to Solve $\frac{dy}{dx} = x - y * x^2$ at x = 0.0(0.1)0.2 given $x_0 = 0$, $y_0 = 1$

```
=> def euler (x0, x1, y0, h):
       def f(x,y):
           return x - y * x**2
       n = round((x1 - x0)/h)
       for i in range (1, n+1):
           y1 = y0 + h * f(x0, y0)
           print('y(', round(x0 + h, 3), ') = ', round(y1, 4))
           for j in range (1, 4):
               y1 = y0 + h/2 * (f(x0, y0) + f(x0 + i * h, y1))
               print('the', j, 'th improved value of y
               (', round(x0+h, 3), ') = ', round(y1, 4))
           x0, y0 = x0 + h, y1
euler(0.0, 0.2, 1, 0.1)
```

DATE 12-04-2024 | EXPT. TITLE: Lab10:- Solution of ODE of first order and first degree by using Runge Kutta Method of 4th order and Milne's PC Method. | PAGE NO. 10

EXP. NO. 5

Lab 10 :- Solution of ODE of first order and first degree by using Runge Kutta Method of 4th order and Milne's PC Method.

Objectives 10.1

* To write python program to solve first order and first degree ODE's using Runge kutta Method and Milne's PC method.

1] Runge-Kutta fourth order method to Solve $\frac{dy}{dx} = x + y^2$ at

$x = 0.2(0.2)0.4$ given $x_0 = 0$, $y_0 = 1$

~~def~~

```
def rk(x0, x1, y0, h):
    def f(x,y):
        return x+y**2
    n = round((x1 - x0)/h)
    for i in range(1, n+1):
        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)
        k=(k1 + 2 * k2 + 2 * k3 + k4)/6
        print('k1 =', round(k1,4), 'k2 =', round(k2,4),
        'k3 =', round(k3,4), 'k4 =', round(k4,4), 'k =',
        round(k,4))
        y1 = y0+k
        print('y (', round(x0+h,3),')= ', round(y1,4))
        x0, y0 = x0+h, y1
```

output :-

k1 = 0.2 k2 = 0.262 k3 = 0.2758 k4 = 0.3655 k = 0.2735

y(0.2) = 1.2735

k1 = 0.3644 k2 = 0.4838 k3 = 0.5193 k4 = 0.7229 k = 0.5156

y(0.4) = 1.7891

DATE .....
EXP. NO. .....

EXPT. TITLE :

RK(0, 0.4, 1, 0.2)

Milne's predicator - Corrector Method.

2] given $\frac{dy}{dx} = x^2 + \frac{y}{2}$, $y(1) = 2$, $y(1.1) = 2.2156$, $y(1.2) = 2.4649$, $y(1.3) = 2.75114$ find $y(1.4)$ using Milne's PC method. Use the Corrector formula thrice.

⇒
```
x0 = 1
y0 = 2
y1 = 2.2156
y2 = 2.4649
y3 = 2.75114
h = 0.1
x1 = x0+h
x2 = x1+h
x3 = x2+h
x4 = x3+h
def f(x,y):
    return x**2+(y/2)
y10 = f(x0, y0)
y11 = f(x1, y1)
y12 = f(x2, y2)
y13 = f(x3, y3)
y4p = y0+(4+h/3)*(2*y11 - y12+2*y13)
print('predicted value of y4 is %.3f' % y4P)
y14 = f(x4, y4P)
for i in range (1,4):
    y4 = y2+(h/3)*(y14+4*y13+y12);
```

output :-

predicted value of y_4 is 3.079
Corrected value of y_4 after iteration 1 is 3.07940
Corrected value of y_4 after iteration 2 is 3.07940
Corrected value of y_4 after iteration 3 is 3.07940

→ print('Corrected value of y_4 after %d iteration %d is \t %3.5f \t', %(i, y_4))

$y_{14} = f(x_4, y_4)$;

output:-

$\left(\frac{\partial}{\partial x_n}(x_n^2 y_n + 2x_n z_n - 4)\right)\hat{i}_n + \left(\frac{\partial}{\partial y_n}(x_n^2 y_n + 2x_n z_n - 4)\right)\hat{j}_n$

$+ \left(\frac{\partial}{\partial z_n}(x_n^2 y_n + 2x_n z_n - 4)\right)\hat{k}_n.$

gradient of N.x ** 2 * N.y + 2 * N.x * N.z - 4 is

$(2x_n y_n + 2z_n)\hat{i}_n + (x_n^2)\hat{j}_n + (2x_n)\hat{k}_n.$

DATE 11/05/24 EXP. NO. 6

EXPT. TITLE: Lab 3:- Finding Gradient, divergent & Curl and their geometrical interpolation.

PAGE NO. 13

Lab 3:- Finding Gradient, divergent & Curl and their geometrical interpolation.

Objective:- Use python

1] To find the gradient, divergence, Curl of a given Vector field.

1. Find the gradient of $\phi = x^2y + 2xz - 4$

⇒
```
from sympy.vector import *
from sympy import symbols
N = CoordSys3D('N')
x,y,z = symbols('x,y,z')
A = N.x**2 * N.y + 2*N.x * N.z - 4
delop = Del()
display(delop(A))
gradA = gradient(A)
print("\n gradient of {A} in \n")
display(gradA)
```

2. To find the divergence of $\vec{A} = x^2yz\,\hat{i} + y^2zx\,\hat{j} + z^2xy\,\hat{k}$

⇒
```
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()
display(delop(A))
divA = delop.dot(A)
```

output :-

$\left(\frac{\partial}{\partial x_n}(x_n^2 y_n z_n)\hat{i}_n + (x_n y_n^2 z_n)\hat{j}_n + (x_n y_n z_n^2)\hat{k}_n\right)$ in

$+\left(\frac{\partial}{\partial y_n}\left((x^2_n y_n z_n)\hat{i}_n + (x y_n^2 z_n^2)\hat{j}_n + (x_n y_n z_n^2)\hat{k}_n\right)\right) j_n$

$+\left(\frac{\partial}{\partial z_n}\left((x_n^2 y_n z_n)\hat{i}_n + (x_n y_n^2 z_n)\hat{j}_n + (x_n y_n z_n^2)\hat{k}_n\right)\hat{k}_n\right)$

divergence of N.x**2 *N.y *N.z * N.i + N.x * N.y **2 *N.z * N.j + N.x *N.y * N.z **2 *N.k is

$6 x_n y_n z_n$.

output :-

Curl of N.x **2 x N.y * N.z * N.i + N.y **2 *N.z * N.i * N.j + N.x *N.y * N.z **2 + N.k is

$(-x_n y_n^2 + x_n z_n^2)\hat{i}_n + (x_n^2 y_n - y_n z_n^2)\hat{j}_n + (-x_n^2 z_n + y_n^2 z_n)\hat{k}_n$

DATE ..... EXP. NO. .....

EXPT. TITLE :

PAGE NO. 14

```
print(f"\n divergence of {A} is \n")
display(divergence(A))
```

3. To find the Curl of $\vec{A} = x^2yz\,i + y^2zx\,j + z^2xy\,k$

```
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(Curl(A))
print(f"\n Curl of {A} in \n")
display(curl(A))
```

output :-

$\frac{1}{3}$.

output :-

$\frac{81}{80}$.

DATE 17/05/2024 | EXPT TITLE: lab1 :- programme to Compute area, Volume. | PAGE NO 15

EXP NO. 7

lab 1 :- programme to Compute area, Volume

1.1 objectives :-

use python

1. to evaluate double integration
2. to Compute area and volume.

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

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

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

2. Evaluate the integral $\int_0^3 \int_0^{3-x} \int_0^{3-x-y} (xyz)\,dz\,dy\,dx$

```
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)
```

output :-

$$\frac{x^3 \cdot y}{3} + \frac{x \cdot y^3}{3}$$

$$\frac{x^3 \cdot y}{3} + \frac{x \cdot y^3}{3}$$

output :-

24.0 * pi

DATE ..... EXP. NO. .....

EXPT. TITLE :

PAGE NO. 16

3. prove that $\int\int (x^2+y^2)\,dy\,dx = \int\int (x^2+y^2)\,dx\,dy$

⇒

```
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)
```

1.2 Area and Volume

Area of the region R in the Cartesian form is $\int_R \int dx\,dy$

4. Find the area of an ellipse by double integration.

$$A = 4\int_0^a \int_0^{(b/a)\sqrt{a^2-x^2}} dy\,dx$$

⇒

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

output:-

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

DATE ..... EXP. NO. .....

EXPT. TITLE :

1.4 Area of the region R in the polar form is $\iint_R r\,dr\,d\theta$.

5. Find the area of the cardioid $r=a(1+\cos t)$ by double integration.

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

output :-

1

DATE 17/05/2024
EXP. NO. 8.
EXPT. TITLE lab 2 :- Evaluation of improper integrals, Beta and Gamma functions.

lab 2 :- Evaluation of improper integrals, Beta and gamma functions.

2.1 objectives :-

use python

1. to evaluate improper integrals using Beta functions.
2. to evaluate 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 number whose beta value needs to be computed.

3. Note :- We can evaluate improper integral involving infinity by using inf.

1. Evaluate $\int_{n}^{\infty} e^{-x}\,dx$

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

Output:-
24.

output :-
m: 3
n: 5
gamma(5.0) is 24.000
Beta (3.0 a) is 0.010.

output :-
m: 2.5
n: 3.5
gamma (3.5) is 3.323
beta (2.5 3.5) is 0.037.

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))
```

3. Find Beta (3,5), Gamma(5)

```
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 %.3f' %t)
print ('Beta (', m,n,') is %.3f' %s)
```

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

```
from sympy import beta, gamma
m = float (input ('m:'));
n = float (input ('n:'));
s = beta(m, n);
t = gamma(n)
print ('gamma(',n,') is %.3f' %t)
print ('Beta (',m,n,') is %.3f' %s)
```