Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

DEPARTMENT OF MECHANICAL ENGINEERING

INDEX

This to certify that Mr./Ms.____, of class ***Third Year*** Roll no ____________ Exam. Seat no. ____________________has performed the above mentioned ***17*** number of experiments in the ***Numerical Methods*** Subject laboratory in the ***Department of Mechanical Engineering*** at VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

Name: Pranav Prasanna Kulkarni
Roll No.: 352021
PRN: 221210421

Date: **01-12-202** In-charge Faculty Head of Department

(1) Bisection Method: Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Bisection Method
"""
import math as m
def ombbsm(fun,x1,x2,acc,maxitr):
  while fun(x1)*fun(x2) > 0:
      print("WRONG INTIAL GUESS")
      x1 = float(input("Enter the New Value of x1: “))
        x2 = float(input("Enter the New Value of x2: "))
    for itr in range(maxitr):
        x0 = (x1+x2)/2
        if fun(x0)*fun(x1) < 0:
            x1 =x1
            x2 =x0
        elif fun(x0)*fun(x1) > 0:
            x1 = x0
            x2 =x2
    print("The root of given function =",x0)
```

Input:

```
ombbsm(lambda x: m.sin(x) + m.cos(x),17,20,0.000001,3)
```

Output:

```
The root of given function = 18.125
```

Name: Pranav Prasanna Kulkarni
Roll No.: 352021
PRN: 22210421

(2) Newton Raphson Method:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Newton Raphson Method
"""
import math as m
def ombnrm(fun,dfun,ddfun,x1,acc,maxitr):
    while abs(fun(x1)*ddfun(x1)/(dfun(x1))**2) > 1:
        print("WRONG INITIAL GUESS")
        x1 = float(input("Enter the New Value of x1:"))
    for itr in range (maxitr):
        x0 = x1 - fun(x1)/dfun(x1)
        if abs(x1-x0) < acc:
            break
        x1 = x0
    print("The root is: ", x0)
```

Input:

```
ombnrm(lambda x: m.exp(x) - m.sin(x), #these are arranged according to the
function defined above        lambda x: m.exp(x) - m.cos(x), #for this
problem value of x1=-3 is accurate        lambda x: m.exp(x) + m.sin(x),
       5,0.00001,100)
```

Output:

```
Enter the New Value of x1:-3
The root is:  -3.1830630119333634
```

Name: Pranav Prasanna Kulkarni
Roll No.: 352021
PRN: 221210421

(3) Gauss Elimination Method:
Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Gauss Elimination Method
"""
import numpy as np
def ombgem(a,d):
    a = np.array(a, dtype = float)
    d = np.array(d, dtype = float)
    n = len(d)
    for i in range (0, n, 1):
        for k in range (i+1, n, 1):
            f = a[k,i] / a[i,i]
            for j in range (0, n):
                a[k,j] = a[k,j] - f * a[i,j]
                d[k] = d[k] - f * d[i]
    print(a)
    print(d)
    x = np.zeros(n)
    print(x)
    for i in range (n-1, -1, -1):
        temp = 0
        for j in range (i+1, n, 1):
            temp = temp + a[i,j] * x[j]
        x[i] = (d[i] - temp) / a[i,i]
    print("Answer = ", x)
```

Input:

```
ombgem(np.array([[4,1,2,3],[3,4,1,2],[2,3,4,1],[1,2,3,4]]),np.array([[40],[
40],[40],[40]]))
```

Output:

```
[[ 4.          1.          2.          3.        ]
 [ 0.          3.25       -0.5        -0.25      ]
 [ 0.          0.          3.38461538 -0.30769231]
 [ 0.          0.          0.          3.63636364]]
[[  40.        ]
 [ -80.        ]
 [ 206.15384615]
 [-502.37762238]]
[0. 0. 0. 0.]
Answer =  [  96.39160839  -27.8041958    48.34965035 -138.15384615]
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 22210421

(4) Gauss-Seidal method:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Gauss Seidal Method
"""
import numpy as np
a = np.array([[4,1,2],[1,3,1],[1,2,5]])
d = np.array([[16],[10],[12]])
a = np.array(a,dtype=float)
d = np.array(d,dtype=float)
n = len(d)
x = np.zeros(n)
maxitr = 10
for itr in range (maxitr):
    for i in range(0,n,1):
        temp = 0
        for j in range(0,n,1):
            if i!=j:
                temp = temp +
a[i,j]*x[j]
        x[i] = (d[i] - temp)/a[i,i]
print(x)
```

Output:

```
[2.99999999 2.00000001 1.]
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 221210421

Curve Fitting: Straight line :

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Straight line curve
"""
import numpy as np
def pra_slcf(x,y):
    x = x.astype(float)
    y = y.astype(float)
    a = np.array([[len(x),sum(x)],[sum(x),sum(x*x)]])
    d = np.array([[sum(y)],[sum(x*y)]])
    b= np.linalg.solve(a,d)
    print("Output")
    print("y = %.4f + (%.4f) * x" %(b[0],b[1]))
```

Input:

```
pra_slcf(np.array([6,7,7,8,8,8,9,9,10]),np.array([5,5,4,5,4,3,4,3,3]))
```

Output:

```
Output
y = 8.0000 + (-0.5000) * x
```

(6) Curve Fitting: Parabola:

Code:

```
"""
Name:Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Curve Fitting Parabola
"""
import numpy as np
def pra_plcf(x,y):
    x = x.astype(float)
    y = y.astype(float)
    a = np.array([[len(x),sum(x),sum(x*x)],
                  [sum(x),sum(x*x),sum(x*x*x)],
                  [sum(x*x),sum(x*x*x),sum(x*x*x*x)]])
    d = np.array([[sum(y)],
                  [sum(x*y)],
                  [sum(x*x*y)]])
    b = np.linalg.solve(a,d)
    print("y = %.4f + (%.4f) * x + (%.4f) * x ^ 2"%(b[0],b[1],b[2]))
```

Input:

```
pra_plcf(np.array([0,1,2,3,4]),np.array([1,1.8,1.3,2,6.3]))
```

Output:

```
y = 1.4914 + (-1.2629) * x + (0.5857) * x ^ 2
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 22210421

(7) Curve Fitting: Power equation Type 1:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Curve Fitting: Power equation Type 1
"""

def pra_cfpe(t,v):
    t = np.array(t,dtype=float)
    v = np.array(v,dtype=float)
    Y = np.log(v)
    X = np.log(t)
    A
np.array([[len(X),sum(X)],[sum(X),sum(X*X)]])
    C = np.array([[sum(Y)],[sum(X*Y)]])
    B = np.linalg.solve(A,C)
    a0 = B[0]
    a1 = B[1]
    alpha = np.exp(a0)
    beta = a1
    print("v = %4f * t ^ %4f"%(alpha,beta))
```

Input:

```
pra_cfpe(np.array([61,26,7,2.6]),np.array([350,400,50,600]))
```

Output:

```
v = 210.465386 * t ^ 0.074103
```

Name: Pranav Prasanna Kulkarni
Roll No.: 352021
PRN: 22210421

(8) Curve Fitting: Power equation Type 2:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Curve Fitting: Power equation Type 2
"""
import numpy as np
def pra_cfpe2(t,N):
    N = np.array(N,dtype=float)
    X = np.array(t,dtype=float)
    Y = np.log(N)
    A = np.array([[len(X),sum(X)],[sum(X),sum(X*X)]])
    C = np.array([[sum(Y)],[sum(X*Y)]])
    B = np.linalg.solve(A,C)
    a0 = B[0]
    a1 = B[1]
    alpha = np.exp(a0)
    beta = np.exp(a1)
    print("N = %4f * %4f ^ t"%(alpha,beta))
```

Input:

```
pra_cfpe2(np.array([0,1,2,3,4,5,6]),np.array([36,57,66,95,132,195,270]
))
```

Output:

```
N = 36.734479 * 1.388871 ^ t
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 22210421

(9) Curve Fitting: Exponential equation: Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Curve Fitting: Exponential equation
"""
import numpy as np
def pra_cfee(x,y):
    y = np.array(y,dtype=float)
    x = np.array(x,dtype=float)
    Y = np.log(y)
    A = np.array([[len(x),sum(x)],[sum(x),sum(x*x)]])
    C = np.array([[sum(Y)],[sum(x*Y)]])
    B = np.linalg.solve(A,C)
    a0 = B[0]
    a1 = B[1]
    a = np.exp(a0)
    b = a1
    print("y = %4f * e ^ (%4f * x)"%(a,b))
```

Input:

```
pra_cfee(np.array([0,1,2,3]),np.array([1.05,2.10,3.85,8.30]))
```

Output:

```
y = 1.043400 * e ^ (0.680853 * x)
```

Name: Pranav Prasanna Kulkarni
Roll No.: 352021
PRN: 22210421

(10) Lagrange’s Interpolation:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021

Program: Lagrange’s Interpolation
"""
import numpy as np
def pra_li(x,y,xr):
    x = x.astype(float)
    y = y.astype(float)
    n = len(x)
    yr = 0
    for i in range(n):
        L = 1
        for j in range(n):
            if i !=j:
                 L = L*(xr-x[j])/(x[i]-x[j])
        yr = yr + y[i] * L
    print("y at x = %.4f is equal to %.4f"%(xr,yr))
```

Input:

```
pra_li(np.array([5,7,11,13,17]),np.array([150,392,1452,2366,5202]),9)
```

Output:

```
y at x = 9.0000 is equal to 810.0000
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 22210421

(11) Newton’s Forward Difference
Interpolation: Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Newton’s Forward Difference Interpolation
"""
import numpy as np
import math as m
def pra_ndfi (x,y,xr):
    x = np.array(x,dtype=float)
    y = np.array(y,dtype=float)
    n = len(x)
    delta = np.zeros((n-1,n-1))
    for j in range (n-1):
        for i in range ((n-1)-j):
            if j == 0:
                delta [i,j] = y[i+1] - y[i]
            else:
                delta[i,j] = delta[i+1,j-1] - delta[i,j-1]
    h = x[1] - x[0]
    u = (xr - x[0])/h
    term = 0
    mult = 1
    for j in range (n-1):
        mult = mult * (u-j)
        term = term + delta[0,j] / m.factorial(j+1) * mult
    yr = y[0] + term
    print("Answer =",yr)
```

Input:

```
pra_ndfi(np.array([2003,2005,2005,2009]),np.array([43,40,45,47]),2004)
```

Output:

```
Answer = 39.8125
```

(12) Trapezoidal rule:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Trapezoidal Rule
"""
import math as m
def pra_tr(fun,x0,xn,n):
    h = (xn - x0) / n
    y0 = fun(x0)
    yn = fun(xn)
    yr = 0
    for i in range (1,n):
        yr = yr + fun(x0 + i * h)
    A = 1/2 * h * (y0 + yn + 2 * yr)
    print("Area = ", A)
```

Input:

```
pra_tr(lambda x: 1/(1+x**2),0,6,6)
```

Output:

```
Area =  1.4107985813868167
```

(13) Simpson’s 1/3rd Rule:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Simpson’s 1/3rd Rule
"""
import math as m
def pra_s13(fun,x0,xn,n):
    h = (xn - x0) / n
    y0 = fun(x0)
    yn = fun(xn)
    yodd = 0
    yeven = 0
    for i in range (1,n,2):
        yodd = yodd + fun(x0 + i * h)
    for j in range (2,n-1,2):
        yeven = yeven + fun(x0 + j * h)
    A = 1/3 * h * (y0 + yn + 4 * yodd + 2 * yeven)
    print("Area = ", A)
```

Input:

```
pra_s13(lambda x: 1/(1+x**2),0,6,6)
```

Output:

```
Area =  1.3661734132322367
```

(14) Simpson’s 3/8th Rule:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Simpson’s 3/8th Rule
"""
import math as m
def pra_s38(fun,x0,xn,n):
    h = (xn - x0) / n
    y0 = fun(x0)
    yn = fun(xn)
    ym3 = 0
    yr = 0
    for i in range (3,n-2,3):
        ym3 = ym3 + fun(x0 + i * h)
    for j in range (1,n,1):
        yr = yr + fun(x0 + j * h)
    yr = yr - ym3
    A = 3/8 * h * (y0 + yn + 3 * yr + 2 * ym3)
    print("Area = ", A)
```

Input:

```
pra_s38(lambda x: 1/(1+x**2),0,6,6)
```

Output:

```
Area =  1.3570808364926013
```

Name: Pranav Prasanna Kulkarni

Roll No.: 352021

PRN: 22210421

(15) Euler Method:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Euler’s Method
"""
import math as m
def pra_em(fun,x0,y0,xn,n):
    h = (xn - x0) / n
    for i in range (1,n+1):
        ynew = y0 + h *
fun(x0,y0)
        x0 = x0 + h
        y0 = ynew
    print("xn = ", xn," ; yn = ",
ynew)
```

Input:

```
pra_em(lambda x,y: x+y,0,1,1,5)
```

Output:

```
xn =  1  ; yn =  2.9766399999999997
```

(16) Runge-Kutta Methods- Second order:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Runge-kutta 2nd order method

"""
import math as m
def pra_rk2(fun,x0,y0,xn,n):
    h = (xn - x0) / n
    for i in range (1,n+1):
        k1 = h * fun(x0,y0)
        k2 = h * fun(x0+h,y0+k1)
        ynew = y0 + 1/2 *
(k1+k2)
        x0 = x0 + h
        y0 = ynew
    print("xn = ", xn," ; yn = ", ynew)
```

Input:

```
pra_rk2(lambda x,y: x+y,0,1,1,5)
```

Output:

```
xn =  1  ; yn =  3.4054163264000006
```

(17) Runge-Kutta Methods- Fourth order:

Code:

```
"""
Name: Pranav Prasanna Kulkarni
Roll No.: 352021
Program: Runge-kutta 4rd order method

"""
import math as m
def pra_rk4(fun,x0,y0,xn,n):
    h = (xn - x0) / n
    for i in range (1,n+1):
        k1 = h * fun(x0,y0)
        k2 = h * fun(x0+h/2,y0+k1/2)
        k3 = h * fun(x0+h/2,y0+k2/2)
        k4 = h * fun(x0+h,y0+k3)
        ynew = y0 + 1/6 * (k1+2*k2+2*k3+k4)
        x0 = x0 + h
        y0 = ynew
    print("xn = ", xn," ; yn = ", ynew)
```

Input:

```
pra_rk4(lambda x,y: x+y,0,1,1,5)
```

Output:

```
xn =  1  ; yn =  3.43650227321187
```