Mathematical Physics Lab Practicals

SGTB Khalsa College, University of Delhi

Preetpal Singh(2020PHY1140)

University Roll No: 20068567043

Unique Paper Code: 32221301

Paper Title: Mathematical Physics Lab

Submitted to: Dr. Savinder Kaur

Sushil Kumar Singh

Programmes

Contents

1 Trapezoidal and Simpson Method **4**

2 Legendre Polynomial **9**

3 Lagrange Interpolation **16**

4 Radioactive Decay, RC Circuit and Stokes Law by Euler, RK2, RK4 Method **20**

5 2nd Order Coupled diffential equations using Euler, RK2, RK4 Method **28**

6 RK4 Method for Simulataneous Differential Equations **34**

7 Gauss Elimination Method **39**

8 Gauss Seidel Method **42**

Practical	**Submission**	**Page No.**
Trapezoidal and Simpson Method	Sep 20, 2021	4
Legendre Polynomial	Sep 21, 2021	9
Lagrange Interpolation	Sep 27, 2021	16
Radioactive Decay, RC Circuit and Stokes Law by Euler, RK2, RK4 Method	Oct 16, 2021	20
2nd Order Coupled diffential equations using Euler, RK2, RK4 Method	Oct 26, 2021	28
RK4 Method for Simulataneous Differential Equations	Nov 2, 2021	34
Gauss Elimination Method	Nov 16, 2021	39
Gauss Seidel Method	Nov 22, 2021	42

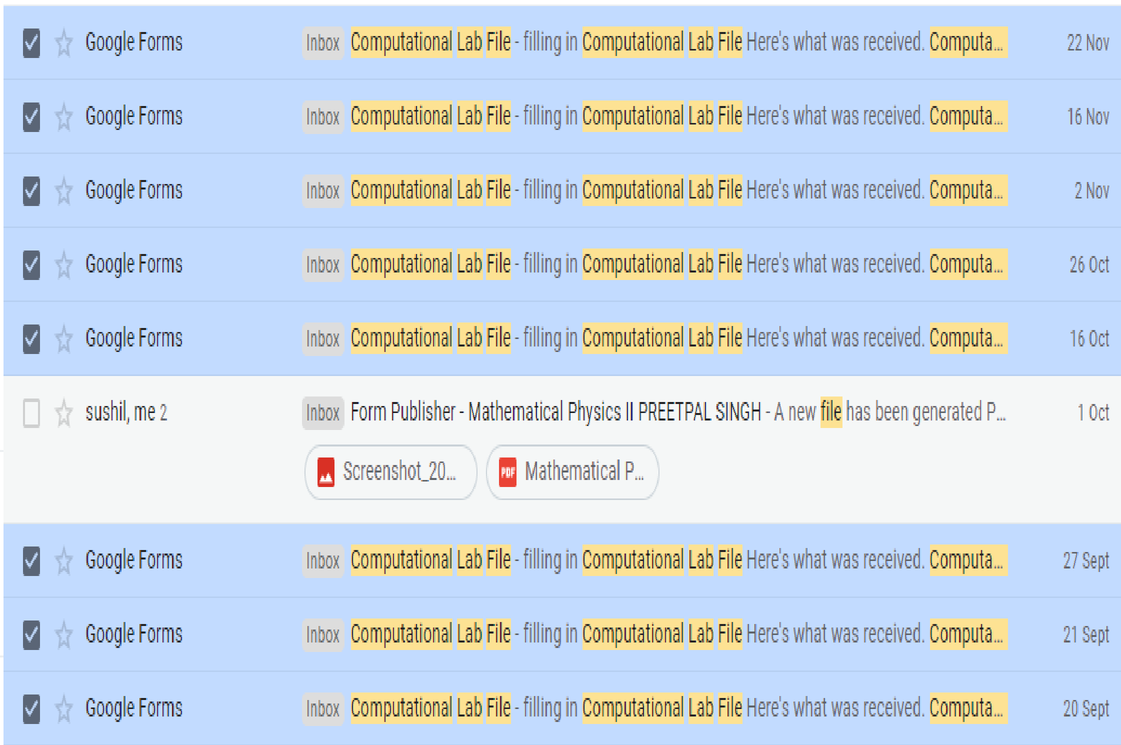


Figure 1: Submission of all Practicals

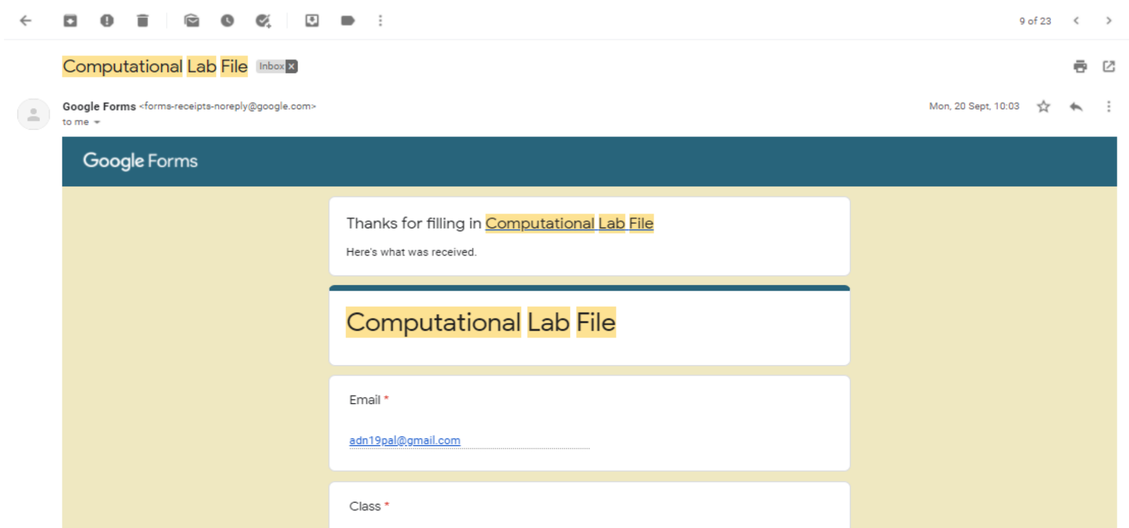


Figure 2: Trapezoidal and Simpson Method

1 Trapezoidal and Simpson Method

```
import numpy as np
import math
import matplotlib.pyplot as plt
import scipy.integrate as integrate
def trapeziodal(func,a,b,n):
    y=[]
    h=(b-a)/n
    for i in range(n+1):
        y.append(func(a+i*h))  #y at limit points
    trp=h*(func(a)+func(b))/2
    for j in range(1,len(y)-1):
        trp=trp+h*(y[j])
    return(trp)
def simpson(func,a,b,n):
    h=(b-a)/(2*n)
    simp=h*(func(a)+func(b))/3
    for i in range(1,2*n):
        if(i%2==0):
            simp=simp+2*h*func(a+i*h)/3
        elif(i%2==1):
            simp=simp+4*h*func(a+i*h)/3
    return(simp)


def graph(func,a,b,n):
    N=np.arange(1,n+1,1)    #Number of Subintervals
    H2=(b-a)/(2*N)
```

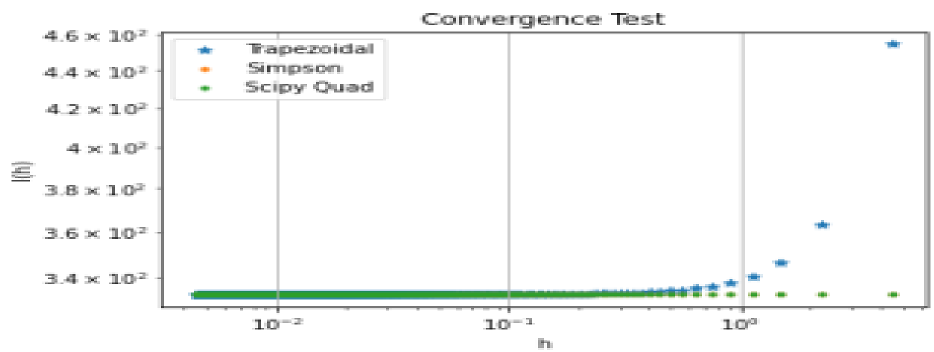
```
    It, Is, Iq =[], [], []
    for i in N:
        z=trapeziodal(func,a,b,i)
        It.append(z)
        z1=simpson(func,a,b,i)
        Is.append(z1)
        analytic = integrate.quad(func, a, b)
        Iq.append(analytic)
    plt.scatter(H2,It,label="Trapezoidal",marker="*")
    plt.scatter(H2,Is,label="Simpson",marker=".")
    plt.scatter(H2,Is,label="Scipy Quad",marker=".")
    plt.yscale("log")
    plt.xscale("log")
    plt.legend()
    plt.xlabel("h")
    plt.ylabel("I(h)")
    plt.title("Convergence Test")
    plt.grid(True)
    plt.show()


def Q2a():
    trp=trapeziodal(func,a,b,n)
    print("Integration Trapezoidal method",trp)

def Q2b():
    simp=simpson(func,a,b,n)
    print("Result by Simpson Method",float(simp))

def Q2c():

    analytic = integrate.quad(func, a, b)
    print("Analytic Solution = ", analytic[0])
    trap=trapeziodal(func,a,b,n)
    print("Solution by trapezoidal method: ",trap)
    err=abs(analytic[0]-trap)
    print("Truncation Error = ",float(err))

def Q2d():
    graph(func,a,b,n)


def Q3a():
    v=np.array([0.0, 0.5, 2.0, 4.05, 8.0, 12.5, 18.0, 24.5, 32.0, 40.5, 50.0])
    c=np.array([0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
    h=(c[-1]-c[0])/(len(c)-1)
    pwr_trap=h*(v[0]+v[-1])/2
    for j in range(1,len(v)-1):
        pwr_trap=pwr_trap+h*(v[j])
    print("Power using trapezoidal method: ", float(pwr_trap), "J")
    pwr_simp=h*(v[0]+v[-1])/3
    for k in range(1,(len(v)-1)):
        if(k%2==0):
            pwr_simp=pwr_simp+2*h*(v[k])/3
        elif(k%2==1):
            pwr_simp=pwr_simp+4*h*(v[k])/3
    print("Power using simpson method : ", float(pwr_simp), "J")
```

```python
def Q3b():
    h=(b-a)/n
    trp=trapeziodal(func,a,b,n)
    print("Integral is {:.8} using Trapezoidal Method".format(float(trp)))
    #Simpson
    simp=simpson(func,a,b,n)
    print("Integral is {:.8} using Simpson Method".format(float(simp)))
    graph(func,a,b,n)

if __name__ == "__main__":
    func=eval("lambda x:"+input("F: "))
    a=float(input("a = "))
    b=float(input("b = "))
    n=int(input("N = "))
    Q2a()
    Q2b()
    Q2c()
    Q2d()
    Q3a()
    Q3b()
```

Listing 1: Trapezoidal and Simpson Method

```
F: x**2
a = 1
b = 10
N = 1000
Integration Trapezoidal method 333.0001214999998

In [83]:  Q2b()

Result by Simpson Method 332.9999999999995

In [84]:  Q2c()

Analytic Solution =  333.0
Solution by trapezoidal method:  333.0001214999998
Truncation Error =  0.00012149999980692883

In [85]:  Q2d()
```

Figure 3: Trapezoidal and Simpson Method Output




```
[36]: Q3a()
```

```
Power using trapezoidal method:  16.705000000000002 J
Power using simpson method :  16.606666666666666 J
```

```
[37]: Q3b()
```

```
Integral is 333.00012 using Trapezoidal Method
Integral is 333.0 using Simpson Method
```

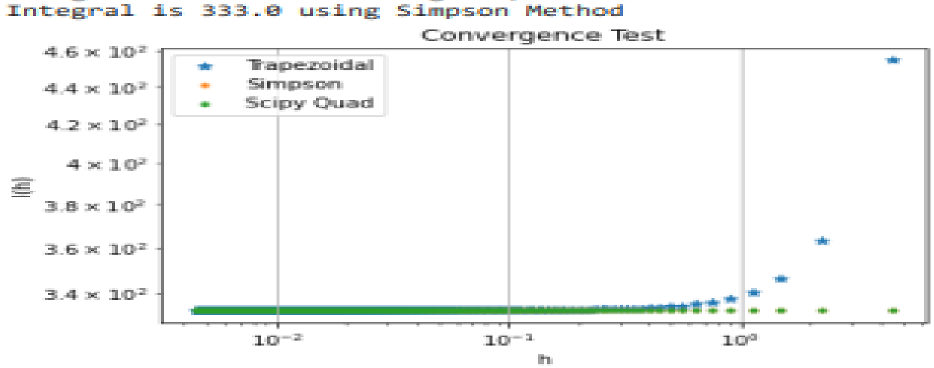


Figure 4: Trapezoidal and Simpson Method Output

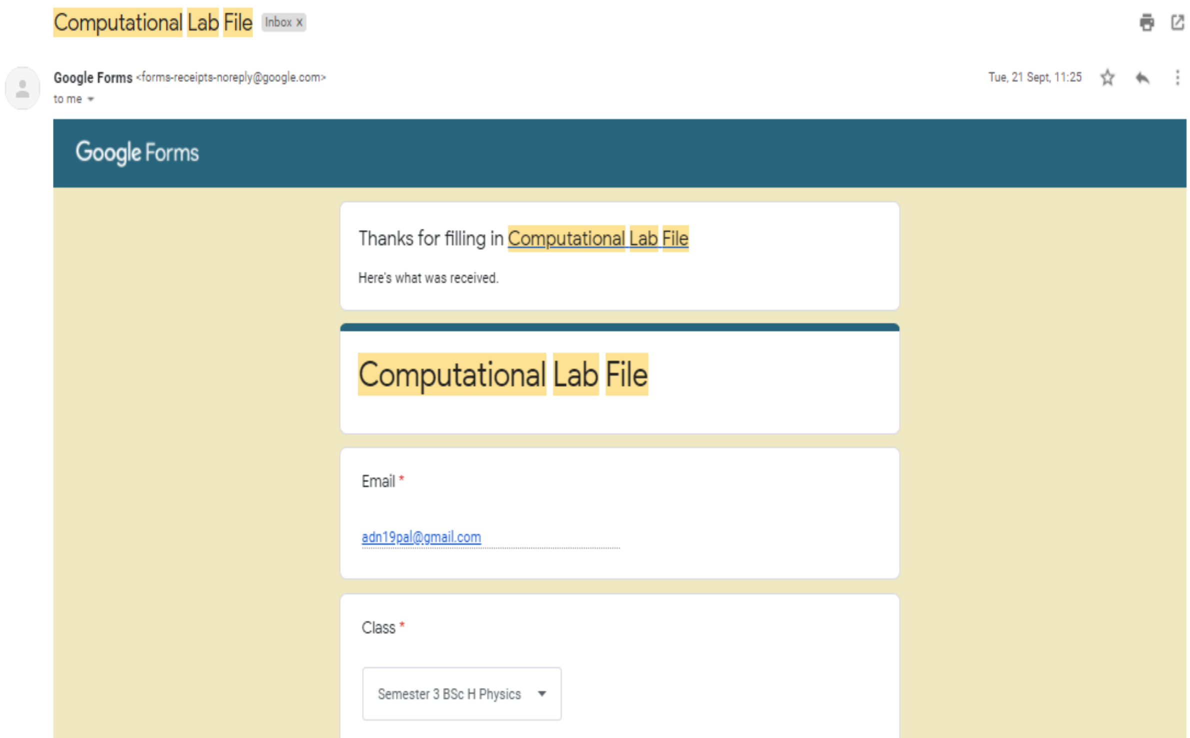


Figure 5: Legendre

2 Legendre Polynomial

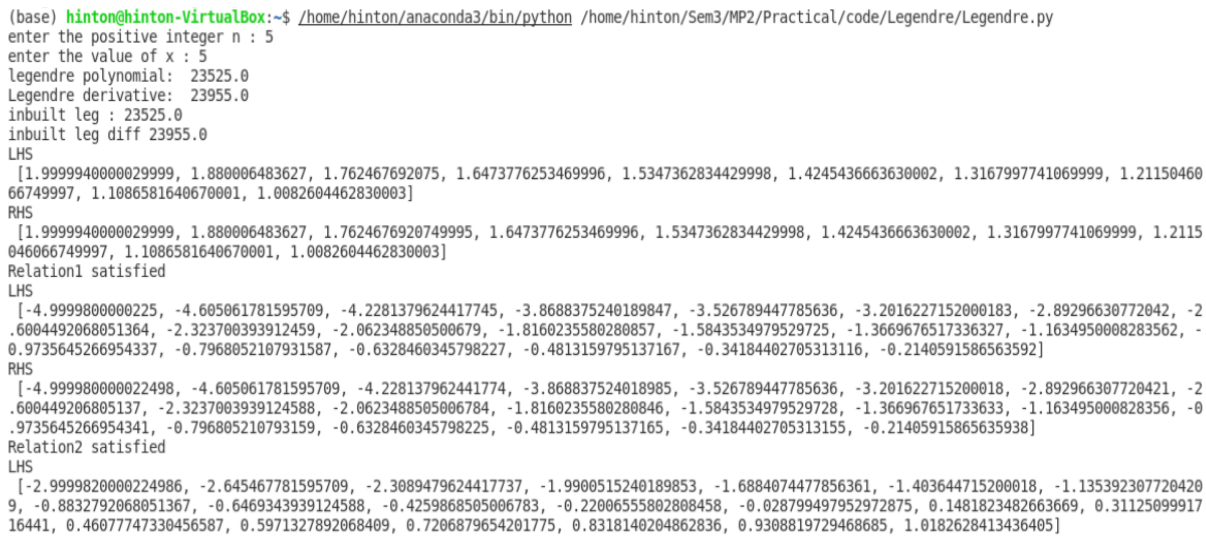
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import eval_legendre,legendre
from scipy.integrate import quad
import math
n = float(input("enter the positive integer n : "))
x = float(input("enter the value of x : "))
def gm(n):
    if n == 1:
        return 1
    elif n == 0.5:
        return np.sqrt(np.pi)
    else :
        return (n-1)*gm(n-1)
def leg(n,x,m=0,p=0):
    if (n %2) == 0 :
        m = int(n/2)
    else :
```

```
        m = int((n-1)/2)
    for i in range(m+1):
        p+= (((-1)**i) * (gm(2*n-2*i+1))* (x**(n-2*i)))/((2**n) * gm(i+1) * gm(n-
    i+1) * gm(n-2*i+1))
    return p

def leg_dif(n,x,m=0,p=0):
    m = 0
    if (n %2) == 0 :
        m = int(n/2)
    else :
        m = int((n-1)/2)
    p = 0
    for i in range(m+1):
        p+= ((n-2*i)*((-1)**i) * (gm(2*n-2*i+1))* (x**(n-2*i-1)))/((2**n) * gm(i
    +1) * gm(n-i+1) * gm(n-2*i+1))
    return p

print("legendre polynomial: ",leg(n,x,m=0,p=0))
print("Legendre derivative: ",leg_dif(n,x,m=0,p=0))
print('inbuilt leg :',eval_legendre(n,x))
print('inbuilt leg diff',np.polyval(legendre(n).deriv(),x))
xdata = np.linspace(-0.999999,0.999999,100)
p0,p1,p2,p3,p0_diff,p1_diff,p2_diff,p3_diff,=[],[],[],[],[],[],[],[]


for i in xdata:
    p0.append(leg(0 ,i))
    p1.append(leg(1 ,i))
    p2.append(leg(2, i))
    p3.append(leg(3, i))
    p0_diff.append(leg_dif(0, i))
    p1_diff.append(leg_dif(1, i))
    p2_diff.append(leg_dif(2, i))
    p3_diff.append(leg_dif(3, i))
def writeintofile(file1,xdata,p0,p1,p2):
    with open(file1,'w') as file :
        for i in range(len(xdata)):
            file.write(str(xdata[i])+' , '+str(p0[i])+' , '+str(p1[i])+' , '+str(
    p2[i])+'\n')
writeintofile('leg00.dat', xdata, p0, p1, p2)
writeintofile('leg01.dat', xdata, p1_diff, p2_diff, p3_diff)
#plotting graph
plt.title("PLOT OF xdata VS Pn ",c= 'b')
plt.plot(xdata,p0)
plt.plot(xdata,p1)
plt.plot(xdata,p2)
plt.legend(['p0','p1','p2'],loc = 'best')
plt.xlabel('xdata')
plt.ylabel('Pn')
plt.grid()
plt.show()
plt.title("PLOT OF xdata VS DIFF Pn")
plt.plot(xdata,p1)
plt.plot(xdata,p0_diff)
plt.plot(xdata,p2_diff)
plt.legend(['p1','p0 diff','p2 diff'],loc = 'best')
```

```
plt.xlabel('xdata')
plt.ylabel('Pn diff')
plt.grid()
plt.show()


n,LHS,RHS=2,[],[]
for i in range(len(xdata)):
    LHS.append(n*p2[i])
    RHS.append(xdata[i]*p2_diff[i] - p1_diff[i])
print("LHS \n",LHS[0:10])
print("RHS \n", RHS[0:10])
if np.allclose(LHS,RHS):
        print("Relation1 satisfied")
else:
    print("Relation1 not satisfied")
writeintofile('leg02.dat', xdata, p2, p2_diff, p1_diff)

# relation 2
n,LHS,RHS=2,[],[]
for i in range(len(xdata)):
    LHS.append((2*n+1)*xdata[i]*p2[i])
    RHS.append((n+1)*p3[i]+n*p1[i])
print("LHS \n",LHS[0:20])
print("RHS \n", RHS[0:20])
if np.allclose(LHS,RHS):
        print("Relation2 satisfied")
else:
    print("Relation2 not satisfied")
writeintofile('leg03.dat', xdata, p2, p1, p3)
#Relation 3
n,LHS,RHS=3,[],[]
for i in range(len(xdata)):
    LHS.append(n*p3[i])
    RHS.append((2*n-1)*xdata[i]*p2[i] - (n-1)*p1[i])
print("LHS \n",LHS[0:20])
print("RHS \n", RHS[0:20])
if np.allclose(LHS,RHS):
        print("Relation3 satisfied")
else:
    print("Relation3 not satisfied")
writeintofile('leg04.dat', xdata, p3, p2, p1)

#orthogonality
A=[] ; B =[]
for n in range(3):
    for m in range(3):
        if n == m:
           A.append(2/(2*n+1))
        else:
           A.append(0)
        f=legendre(n)*legendre(m)
        inte , err = quad(f, -1, 1)
        B.append(inte)
RHS = np.array(B).reshape(3,3)
LHS = np.array(A).reshape(3,3)
print("RHS = ",RHS)
```

```python
print("LHS = ",LHS)
if np.allclose(LHS,RHS):
   print("Orthogonality verified")
else:
    print("Orthogonality not verified")
```

Listing 2: Legendre Polynomial



```
(base) hinton@hinton-VirtualBox:~$ /home/hinton/anaconda3/bin/python /home/hinton/Sem3/MP2/Practical/code/Legendre/Legendre.py
enter the positive integer n : 5
enter the value of x : 5
legendre polynomial:  23525.0
Legendre derivative:  23955.0
inbuilt leg : 23525.0
inbuilt leg diff 23955.0
LHS
 [1.9999940000029999, 1.880006483627, 1.762467692075, 1.6473776253469996, 1.5347362834429998, 1.4245436663630002, 1.3167997741069999, 1.21150460
66749997, 1.1086581640670001, 1.0082604462830003]
RHS
 [1.9999940000029999, 1.880006483627, 1.7624676920749995, 1.6473776253469996, 1.5347362834429998, 1.4245436663630002, 1.3167997741069999, 1.2115
046066749997, 1.1086581640670001, 1.0082604462830003]
Relation1 satisfied
LHS
 [-4.9999800000225, -4.605061781595709, -4.2281379624417745, -3.8688375240189847, -3.526789447785636, -3.2016227152000183, -2.89296630772042, -2
.6004492068051364, -2.323700393912459, -2.062348850500679, -1.8160235580280857, -1.5843534979529725, -1.3669676517336327, -1.1634950008283562, -
0.9735645266954337, -0.7968052107931587, -0.6328460345798227, -0.4813159795137167, -0.34184402705313116, -0.2140591586563592]
RHS
 [-4.999980000022498, -4.605061781595709, -4.228137962441774, -3.868837524018985, -3.526789447785636, -3.201622715200018, -2.892966307720421, -2
.600449206805137, -2.3237003939124588, -2.0623488505006784, -1.8160235580280846, -1.5843534979529728, -1.366967651733633, -1.163495000828356, -0
.9735645266954341, -0.796805210793159, -0.6328460345798225, -0.4813159795137165, -0.34184402705313155, -0.21405915865635938]
Relation2 satisfied
LHS
 [-2.9999820000224986, -2.645467781595709, -2.3089479624417737, -1.9900515240189853, -1.6884074477856361, -1.403644715200018, -1.135392307720420
9, -0.8832792068051367, -0.6469343939124588, -0.4259868505006783, -0.22006555802808458, -0.028799497952972875, 0.1481823482663669, 0.31125099917
16441, 0.46077747330456587, 0.5971327892068409, 0.7206879654201775, 0.8318140204862836, 0.9308819729468685, 1.0182628413436405]
```

Figure 6: Legendre

```
 [1.9999940000029999, 1.880006483627, 1.7624676920749995, 1.6473776253469996, 1.5347362834429998, 1.4245436663630002, 1.3167997741069999, 1.2115
046066749997, 1.1086581640670001, 1.0082604462830003]
Relation1 satisfied
LHS
 [-4.9999800000225, -4.605061781595709, -4.2281379624417745, -3.8688375240189847, -3.526789447785636, -3.2016227152000183, -2.89296630772042, -2
.6004492068051364, -2.323700393912459, -2.062348850500679, -1.8160235580280857, -1.5843534979529725, -1.3669676517336327, -1.1634950008283562, -
0.9735645266954337, -0.7968052107931587, -0.6328460345798227, -0.4813159795137167, -0.34184402705313116, -0.2140591586563592]
RHS
 [-4.999980000022498, -4.605061781595709, -4.228137962441774, -3.868837524018985, -3.526789447785636, -3.201622715200018, -2.892966307720421, -2
.600449206805137, -2.3237003939124588, -2.0623488505006784, -1.8160235580280846, -1.5843534979529728, -1.366967651733633, -1.163495000828356, -0
.973564526695434l, -0.796805210793159, -0.6328460345798225, -0.4813159795137165, -0.34184402705313155, -0.21405915865635938]
Relation2 satisfied
LHS
 [-2.9999820000224986, -2.645467781595709, -2.3089479624417737, -1.9900515240189853, -1.6884074477856361, -1.403644715200018, -1.135392307720420
9, -0.8832792068051367, -0.6469343939124588, -0.4259868505006783, -0.22006555802808458, -0.028799497952972875, 0.1481823482663669, 0.31125099917
16441, 0.46077747330456587, 0.5971327892068409, 0.7206879654201775, 0.8318140204862836, 0.9308819729468685, 1.0182628413436405]
RHS
 [-2.9999820000225004, -2.645467781595709, -2.3089479624417746, -1.9900515240189849, -1.6884074477856361, -1.4036447152000182, -1.13539230772042
02, -0.8832792068051365, -0.6469343939124592, -0.42598685050067875, -0.2200655580280857, -0.028799497952972652, 0.14818234826636734, 0.311250999
1716439, 0.4607774733045663, 0.5971327892068412, 0.7206879654201773, 0.8318140204862834, 0.9308819729468689, 1.0182628413436408]
Relation3 satisfied
/home/hinton/anaconda3/lib/python3.8/site-packages/numpy/lib/polynomial.py:1329: FutureWarning: In the future extra properties will not be copie
d across when constructing one poly1d from another
  other = poly1d(other)
RHS =  [[2.00000000e+00 0.00000000e+00 5.55111512e-17]
 [0.00000000e+00 6.66666667e-01 0.00000000e+00]
 [5.55111512e-17 0.00000000e+00 4.00000000e-01]]
LHS =  [[2.         0.         0.        ]
 [0.         0.66666667 0.        ]
 [0.         0.         0.4       ]]
Orthogonality verified
```

Figure 7: Legendre

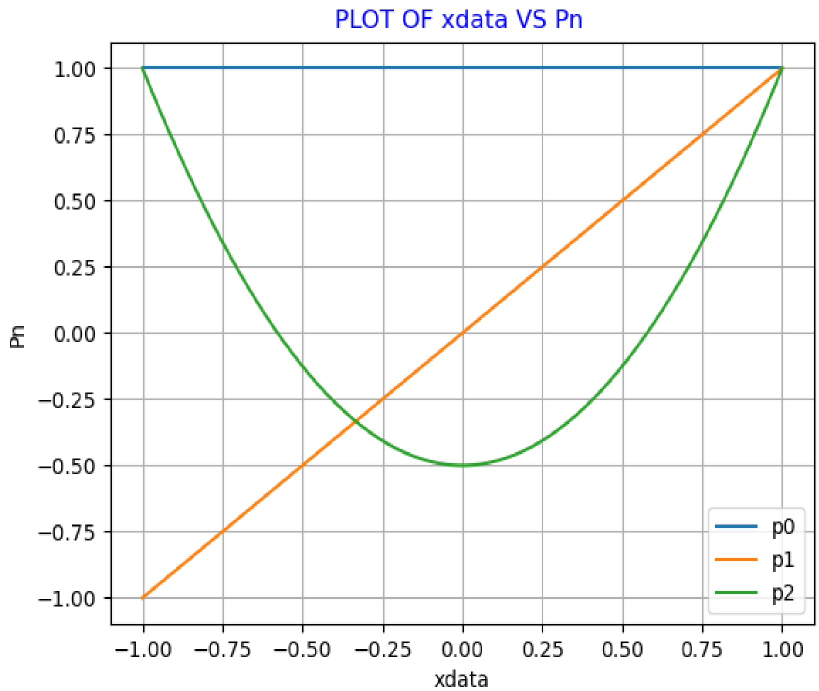


Figure 8: Legendre

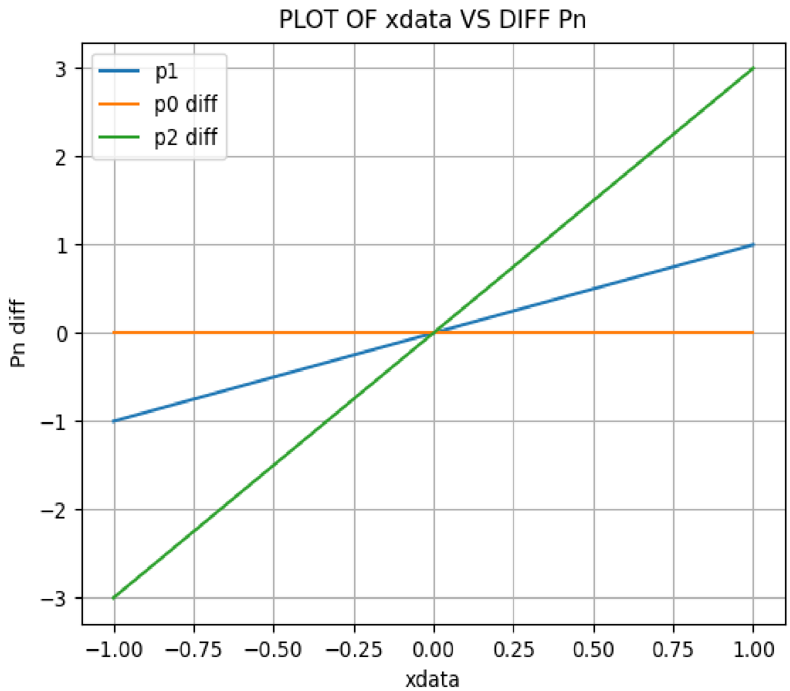


Figure 9: Legendre

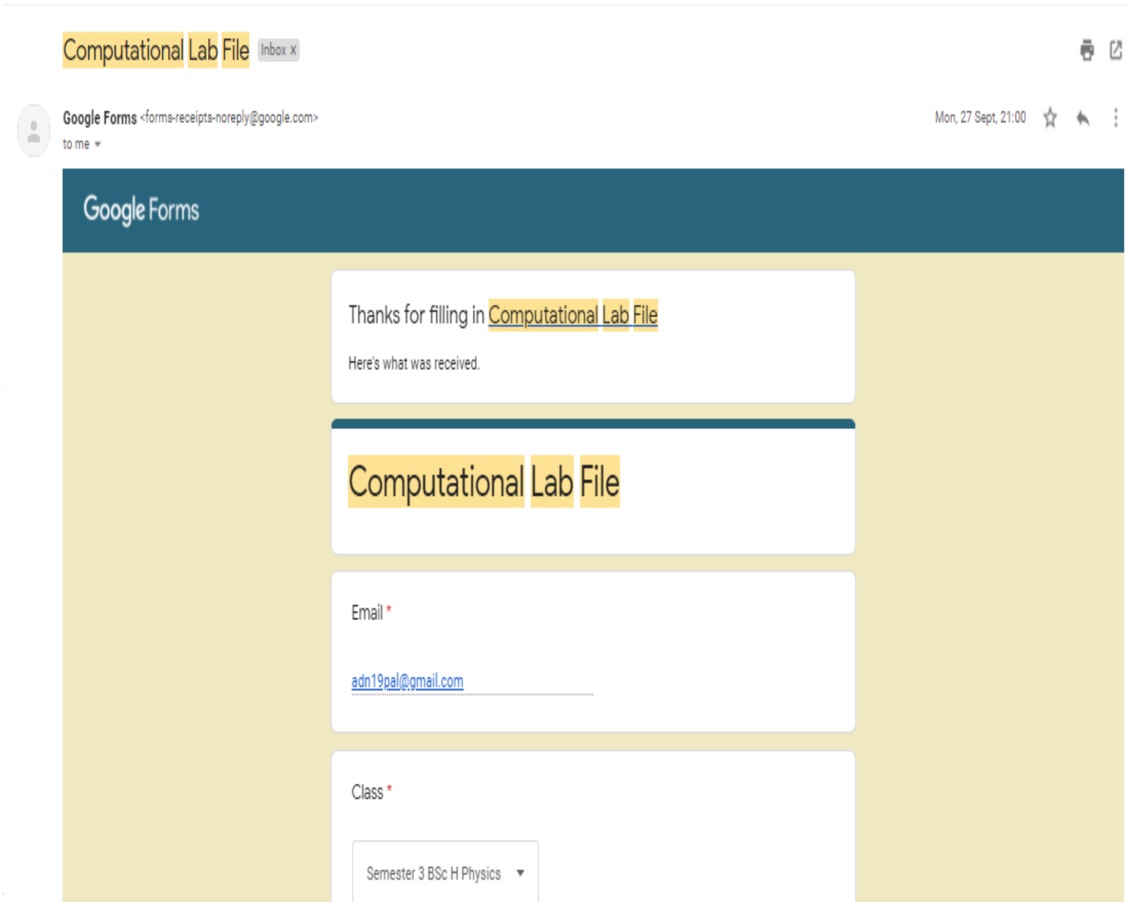


Figure 10: Lagrange Interpolation

3 Lagrange Interpolation

```
import numpy as np
from prettytable import PrettyTable
from scipy.interpolate import lagrange
import matplotlib.pyplot as plt
from sympy import Symbol
def Lagrange(x, L_x, L_y,y = 0):
  m = len(L_x)
  for i in range(0, m):
    prod_i = 1
    for j in range(m):
      if i==j:
        continue
```

```
      else:
        prod_i = prod_i*(x - L_x[j])/(L_x[i] - L_x[j])
    y = y + L_y[i]*prod_i
  return y
def Lagrange_inverse(y,L_x,L_y,x=0):
    return Lagrange(y,L_y,L_x,x=0)
def inbuilt(x, L_x, L_y):
    poly = lagrange(L_x, L_y)
    L=poly(x)
    return L

L1 = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8,
     3.0]
L2 = [1.0, 0.99, 0.96, 0.91, 0.85, 0.76, 0.67, 0.57, 0.46, 0.34, 0.22, 0.11,
     0.00, -0.10, -0.18, -0.26]
I = [2.81, 3.24, 3.80, 4.30, 4.37, 5.29, 6.03]
V = [0.5, 1.2, 2.1, 2.9, 3.6, 4.5, 5.7]
mytable = PrettyTable(["Problem" , "inbuilt", "my function","Relative Error"])
mytable.add_rows([["3(a)(i)",inbuilt(2.3, L1, L2),Lagrange(2.3, L1, L2),
                  (inbuilt(2.3, L1, L2)-Lagrange(2.3, L1, L2))/inbuilt(2.3, L1,
     L2)],
                 ["3(a)(ii)",inbuilt(0.5, L2,L1),Lagrange(0.5, L2, L1,y = 0),
                  (inbuilt(0.5, L2,L1)-Lagrange(0.5, L1, L2,y = 0))/inbuilt(0.5,
      L2,L1)],
                 ["3(b)",inbuilt(2.4, V,I),Lagrange(2.4, V,I),
                  (inbuilt(2.4, V,I)-Lagrange(2.4, V,I))/inbuilt(2.4, V,I)]])
bes_inb,bes_inb_inv,lag_bes,lag_bes_inv,lin_inb,lin_inb_inv,lin_lag,lin_lag_inv=
     [],[],[],[],[],[],[],[]
for i in xrange:
  bes_inb.append(inbuilt(i, L1, L2))
  lag_bes.append(Lagrange(i, L1, L2))
x1range = np.linspace(0.0,6.0,100,float)
for i in x1range:
  lin_inb.append(inbuilt(i, V,I))
  lin_lag.append(Lagrange(i, V,I))
x2range = np.linspace(1.0,-0.3,100,float)
for i in x2range:
    bes_inb_inv.append(inbuilt(i, L2,L1))
    lag_bes_inv.append(Lagrange(i, L2, L1,y = 0))
x3range = np.linspace(2.0,6.5,100,float)
for i in x3range:
    lin_inb_inv.append(inbuilt(i, I,V))
    lin_lag_inv.append(Lagrange(i,I,V,y = 0))
fig,axs=plt.subplots(2,2,figsize=(15,8))
ax11,ax12,ax21,ax22=axs[0][0],axs[0][1],axs[1][0],axs[1][1]
ax11.plot(xrange,lag_bes,label="lagrange interpolation"),ax11.plot(xrange,bes_inb
     ,label="inbuilt")
ax11.scatter(L1,L2,label="Given Data Points"),ax11.scatter(2.3,Lagrange(2.3, L1,
     L2),label="Interpolation Point",c='red')
ax11.set_title("Bessel Interpolation"),ax11.set_ylabel("f(x)")

ax12.plot(x2range,lag_bes_inv,label="Inverse lagrange interpolation"),ax12.plot(
     x2range,bes_inb_inv,label="inbuilt")
ax12.scatter(L2,L1,label="Given Data Points"),ax12.set_ylabel("x")
ax12.set_ylim([-0.5,3.5]), ax12.set_title("Bessel Inverse Interpolation")
ax21.plot(x1range,lin_lag,label="lagrange interpolation"),ax21.plot(x1range,
     lin_inb,label="inbuilt")
```

```python
ax21.scatter(V,I,label="Given Data Points"),ax21.scatter(2.4,Lagrange(2.4, V, I),
     label="Interpolation Point",c='red')
ax21.set_title("V vs I Interpolation"),ax21.set_ylabel("V")
ax22.plot(x3range,lin_inb_inv,label="Inverse lagrange interpolation"),ax22.plot(
     x3range,lin_lag_inv,label="inbuilt")
ax22.scatter(I,V,label="Given Data Points"),ax22.set_title("I vs V Inverse
     Interpolation")
ax22.set_ylim([-10,10]),ax22.set_ylabel("I")
ax11.legend(),ax11.grid(True),ax12.legend(),ax12.grid(True),ax21.legend(),ax21.
     grid(True),ax22.legend(),ax22.grid(True)
plt.show()
```

Listing 3: Lagrange Interpolation

Problem	inbuilt	my function	Relative Error
3(a)(i)	0.05503416527935989	0.05503375768661485	7.406176562730459e-06
3(a)(ii)	1.5373769473726524	1.5373769470625294	0.397845435865116
3(b)	4.046204673660418	4.046204673660199	5.399921874158954e-14

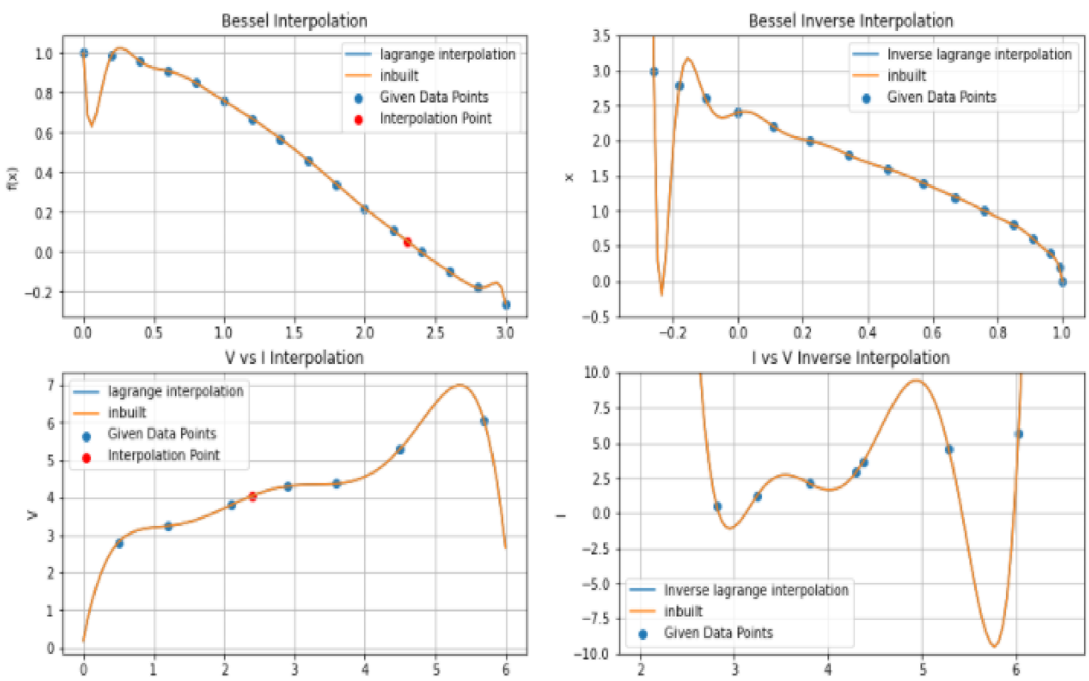


Figure 11: Lagrange Interpolation

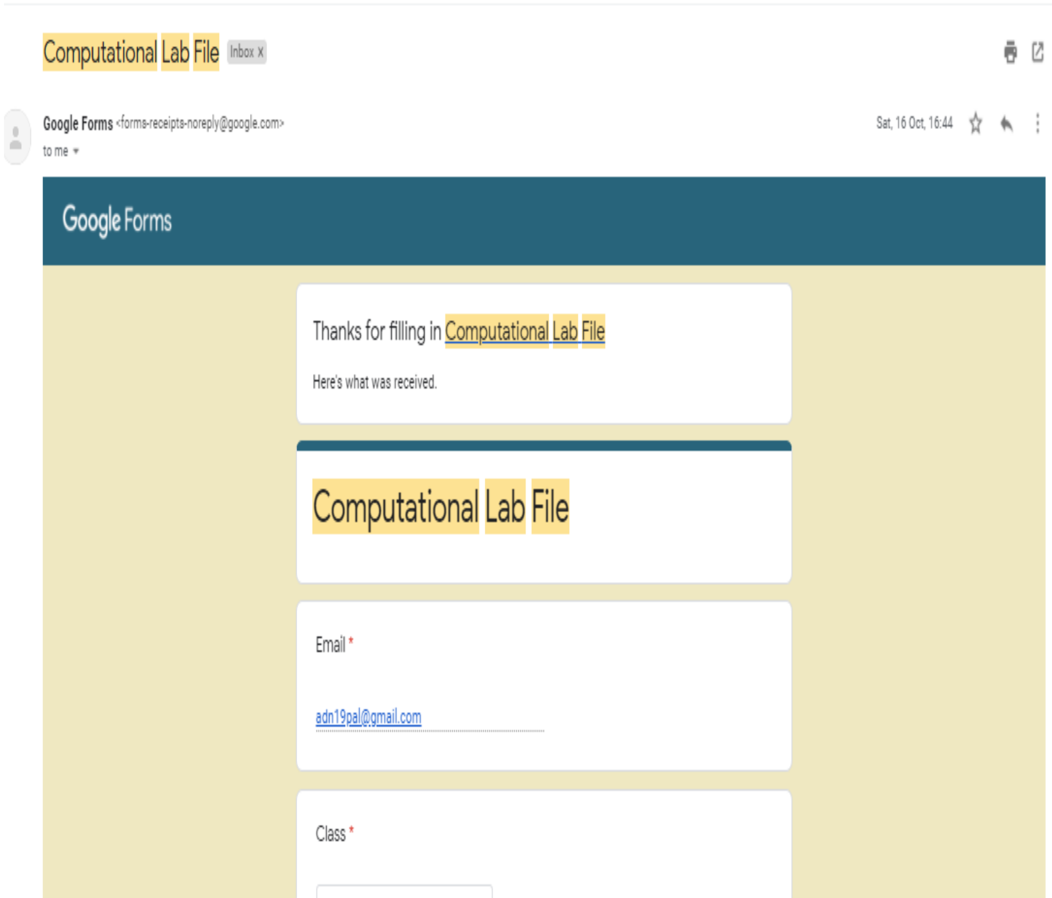


Figure 12: Radioactive Decay, RC Circuit and Stokes Law by Euler, RK2, RK4 Method

4 Radioactive Decay, RC Circuit and Stokes Law by Euler, RK2, RK4 Method

```
import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import texttable as tt
tab = tt.Texttable()
def euler(f,a,b,n,yinit):
    h=(b-a)/(n)
    xs = a+np.arange(n)*h
    ys=np.zeros(n)
```

```
    y = yinit
    for j,x in enumerate(xs):
        ys[j]=y
        y+=h*f(x,y)
    return xs, ys
def rk2(f,a,b,n,yinit):
    h=(b-a)/(n)
    xs = a+np.arange(n)*h
    ys=np.zeros(n)
    y = yinit
    for j,x in enumerate(xs):
        ys[j]=y
        k0 = h*f(x,y)
        y+=h*f(x+h/2,y+k0/2)
    return xs, ys
def rk4(f,a,b,n,yinit):
    h=(b-a)/(n)
    xs = a+np.arange(n)*h
    ys=np.zeros(n)
    y = yinit
    for j,x in enumerate(xs):
        ys[j]=y
        k0 = h*f(x,y)
        k1 = h*f(x+h/2,y+k0/2)
        k2 = h*f(x+h/2,y+k1/2)
        k3 = h*f(x+h,y+k2)
        y+=(k0+2*k1+2*k2+k3)/6
    return xs, ys
def Analytic(yinit,a,b,h,tau):
    xs,ys=[],[]
    p=np.arange(a,b,h)
    for t in p:
        D=yinit*np.exp(-1*t/tau)
        ys.append(D)
        xs.append(t)
    return xs,ys
def graph(xs_1,ys_1,xs_2,ys_2,xs_3,ys_3,xs_4,ys_4,xs_5,ys_5,xs_6,ys_6,xs_7,ys_7,
     xs_8,ys_8,xs_9,ys_9,xs_10,ys_10,title):
    fig,axs=plt.subplots(3,2,figsize=(15,15))
    fig.suptitle(title, fontsize=30)
    ax11,ax12,ax21,ax22,ax31,ax32=axs[0][0],axs[0][1],axs[1][0],axs[1][1],axs
     [2][0],axs[2][1]
    ax11.plot(xs_1,ys_1,'^', color='green',label="euler"),ax11.plot(xs_4,ys_4,'-'
     , color='black',label="rk2")
    ax11.plot(xs_7,ys_7,'>',color='black',label="rk4"),ax11.plot(xs_7,ys_7,'*',
     color='brown',label="Analytic")
    ax11.set_title("Analytic v/s Euler v/s rk2 v/s rk4"),ax11.set_ylabel("N"),
     ax11.set_xlabel("Time")
    ax12.plot(xs_1,ys_1,'^', color='green',label="h ={}".format(xs_1[1]-xs_1[0]))
     ,ax12.plot(xs_2,ys_2,'-', color='black',label="h ={}".format(xs_2[1]-xs_2
     [0]))
    ax12.plot(xs_3,ys_3,'>',color='black',label="h ={}".format(xs_3[1]-xs_3[0])),
     ax12.plot(xs_7,ys_7,'*',color='brown',label="Analytic")
    ax12.set_title("Euler for different stepsize(h)"),ax12.set_ylabel("N"),ax12.
     set_xlabel("Time")
    ax21.plot(xs_4,ys_4,'^', color='green',label="h ={}".format(xs_4[1]-xs_4[0]))
     ,ax21.plot(xs_5,ys_5,'-', color='black',label="h ={}".format(xs_5[1]-xs_5
```

```
     [0]))
    ax21.plot(xs_6,ys_6,'>',color='black',label="h ={}".format(xs_6[1]-xs_6[0])),
     ax21.plot(xs_7,ys_7,'*',color='brown',label="Analytic")
    ax21.set_title("rk2 for different stepsize(h)"),ax21.set_ylabel("N"),ax21.
     set_xlabel("Time")
    ax22.plot(xs_7,ys_7,'^', color='green',label="h ={}".format(xs_7[1]-xs_7[0]))
     ,ax22.plot(xs_8,ys_8,'-', color='black',label="h ={}".format(xs_8[1]-xs_8
     [0]))
    ax22.plot(xs_9,ys_9,'>',color='black',label="h ={}".format(xs_9[1]-xs_9[0])),
     ax22.plot(xs_10,ys_10,'*',color='brown',label="Analytic")
    ax22.set_title("rk4 for different stepsize(h)"),ax22.set_ylabel("N"),ax22.
     set_xlabel("Time")
    ax31.plot(xs_1,(ys_10-ys_1)/ys_10,'.', color='green',label="euler"),ax31.plot
     (xs_1,(ys_10-ys_4)/ys_10,'.', color='black',label="rk2")
    ax31.plot(xs_1,(ys_10-ys_7)/ys_10,'.', color='red',label="rk4")
    ax31.set_title("Error Plot at h = 0.4"),ax31.set_ylabel("Absolute Error"),
     ax31.set_xlabel("Time")
    ax32.plot(xs_1,(ys_1-ys_4)/ys_1,'.', color='green',label="euler-rk2"),ax32.
     plot(xs_1,(ys_7-ys_4)/ys_7,'.', color='black',label="rk4-rk2")
    ax32.plot(xs_1,(ys_1-ys_7)/ys_1,'.', color='red',label="euler-rk4")
    ax32.set_title("Comparative Error Plot at h = 0.4"),ax32.set_ylabel("Absolute
      Error"),ax32.set_xlabel("Time")
    ax11.legend(),ax11.grid(True),ax12.legend(),ax12.grid(True),ax21.legend(),
     ax21.grid(True),ax22.legend(),ax22.grid(True)
    ax31.legend(),ax31.grid(True),ax32.legend(),ax32.grid(True)
    plt.show()
def q3_a(a,yinit,t_half):
    b = 5*t_half
    tau=t_half/np.log(2)
    h = t_half/10
    n = int((b-a)/h)
    decay = lambda x, y: -1*y/tau
    xs_1, ys_1 = euler(decay,a,b,n,yinit)
    xs_2, ys_2 = euler(decay,a,b,2*n,yinit)
    xs_3, ys_3 = euler(decay,a,b,4*n,yinit)
    xs_4, ys_4 = rk2(decay,a,b,n,yinit)
    xs_5, ys_5 = rk2(decay,a,b,2*n,yinit)
    xs_6, ys_6 = rk2(decay,a,b,4*n,yinit)
    xs_7, ys_7 = rk4(decay,a,b,n,yinit)
    xs_8, ys_8 = rk4(decay,a,b,2*n,yinit)
    xs_9, ys_9 = rk4(decay,a,b,4*n,yinit)
    xs_10, ys_10 = Analytic(yinit,a,b,h,tau)
    print("Radioactive Decay", "h =", h)
    headings_1 = ["t" ,"Analytic","euler","rk2","rk4","Ab_error euler","Ab_error
     rk2","Ab_error rk4"]
    tab.header(headings_1)
    for row in zip(xs_1,ys_10,ys_1,ys_4,ys_7,(ys_10-ys_1)/ys_10,(ys_10-ys_4)/
     ys_10,(ys_10-ys_7)/ys_10):
        tab.add_row(row)
        tab.set_max_width(0)
        tab.set_precision(6)
    s = tab.draw()
    print(s)
    tab.reset()

    graph(xs_1,ys_1,xs_2,ys_2,xs_3,ys_3,xs_4,ys_4,xs_5,ys_5,xs_6,ys_6,xs_7,ys_7,
     xs_8,ys_8,xs_9,ys_9,xs_10,ys_10,"Radioactive Decay")
```

```
def q3_b(a,yinit,R,C):
    b = 5*R*C
    tau=R*C
    h = tau/10
    n = int((b-a)/h)
    rc = lambda x, y: -1*y/tau
    xs_1, ys_1 = euler(rc,a,b,n,yinit)
    xs_2, ys_2 = euler(rc,a,b,2*n,yinit)
    xs_3, ys_3 = euler(rc,a,b,4*n,yinit)

    xs_4, ys_4 = rk2(rc,a,b,n,yinit)
    xs_5, ys_5 = rk2(rc,a,b,2*n,yinit)
    xs_6, ys_6 = rk2(rc,a,b,4*n,yinit)

    xs_7, ys_7 = rk4(rc,a,b,n,yinit)
    xs_8, ys_8 = rk4(rc,a,b,2*n,yinit)
    xs_9, ys_9 = rk4(rc,a,b,4*n,yinit)
    xs_10, ys_10 = Analytic(yinit,a,b,h,tau)
    print("RC Circuit", "h =", h)
    headings_1 = ["t" ,"Analytic","euler","rk2","rk4","$\delta$","Ab_error rk2","
     Ab_error rk4"]
    tab.header(headings_1)
    for row in zip(xs_1,ys_10,ys_1,ys_4,ys_7,(ys_10-ys_1)/ys_10,(ys_10-ys_4)/
     ys_10,(ys_10-ys_7)/ys_10):
        tab.add_row(row)
        tab.set_max_width(0)
        tab.set_precision(6)
    s = tab.draw()
    print(s)
    tab.reset()

    graph(xs_1,ys_1,xs_2,ys_2,xs_3,ys_3,xs_4,ys_4,xs_5,ys_5,xs_6,ys_6,xs_7,ys_7,
     xs_8,ys_8,xs_9,ys_9,xs_10,ys_10,"RC Circuit")

def q3_c(a,yinit,eta,rad,m):
    tau=m/((np.pi)*6*rad*eta)
    b = 5*tau
    h = tau/10
    n = int((b-a)/h)
    stokes = lambda x, y: -1*y/tau
    xs_1, ys_1 = euler(stokes,a,b,n,yinit)
    xs_2, ys_2 = euler(stokes,a,b,2*n,yinit)
    xs_3, ys_3 = euler(stokes,a,b,4*n,yinit)
    xs_4, ys_4 = rk2(stokes,a,b,n,yinit)
    xs_5, ys_5 = rk2(stokes,a,b,2*n,yinit)
    xs_6, ys_6 = rk2(stokes,a,b,4*n,yinit)
    xs_7, ys_7 = rk4(stokes,a,b,n,yinit)
    xs_8, ys_8 = rk4(stokes,a,b,2*n,yinit)
    xs_9, ys_9 = rk4(stokes,a,b,4*n,yinit)
    xs_10, ys_10 = Analytic(yinit,a,b,h,tau)
    xs_10.pop(int(xs_10[-1]))
    ys_10.pop(int(ys_10[-1]))
    print("Stokes Law", "h =", h)
```

```
    headings_1 = ["t" ,"Analytic","euler","rk2","rk4","Ab_error euler","Ab_error
     rk2","Ab_error rk4"]
    tab.header(headings_1)
    for row in zip(xs_1,ys_10,ys_1,ys_4,ys_7,(ys_10-ys_1)/ys_10,(ys_10-ys_4)/
     ys_10,(ys_10-ys_7)/ys_10):
        tab.add_row(row)
        tab.set_max_width(0)
        tab.set_precision(6)
    s = tab.draw()
    print(s)
    tab.reset()
    graph(xs_1,ys_1,xs_2,ys_2,xs_3,ys_3,xs_4,ys_4,xs_5,ys_5,xs_6,ys_6,xs_7,ys_7,
     xs_8,ys_8,xs_9,ys_9,xs_10,ys_10,"Stokes Law")
if __name__ == "__main__":
    q3_a(0,20000,4)
    q3_b(0,10,1e3,1e-6)
    q3_c(0,10,10,0.2,200)
```

Listing 4: Radioactive Decay, RC Circuit and Stokes Law by Euler, RK2, and RK4 Method

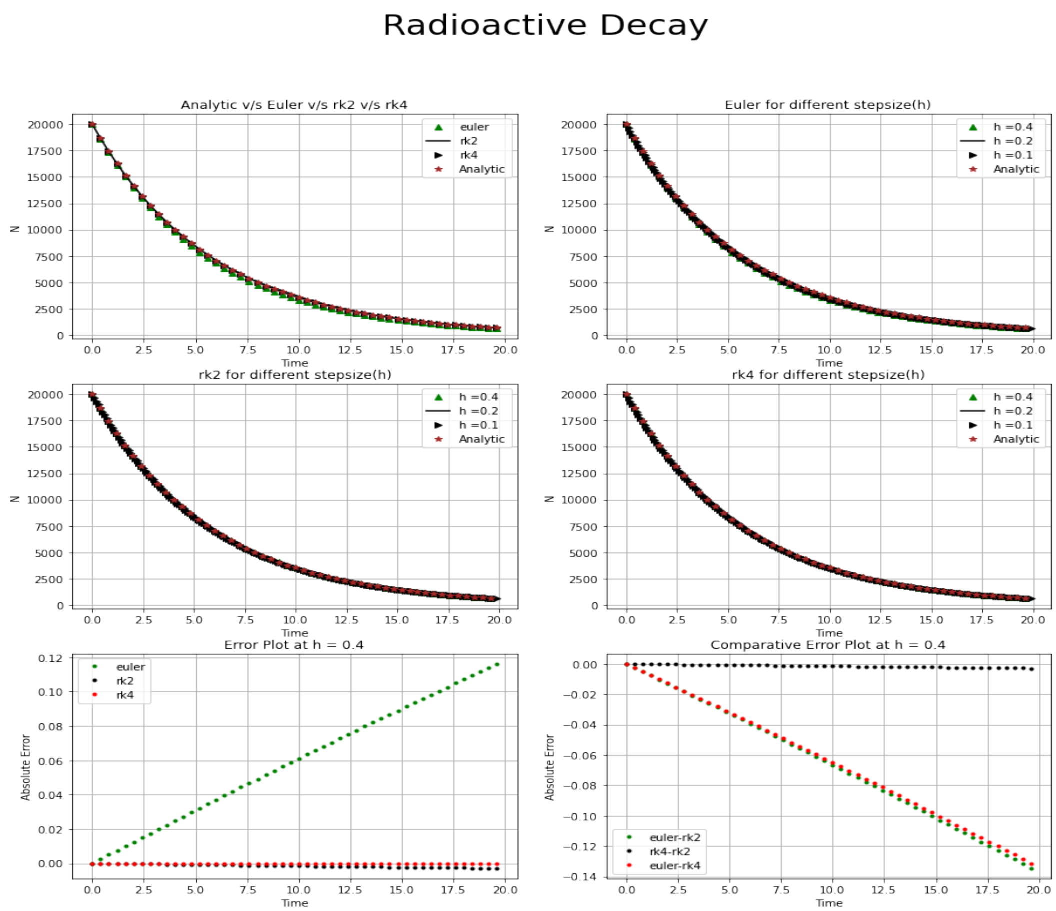


Figure 13: Radioactive Decay

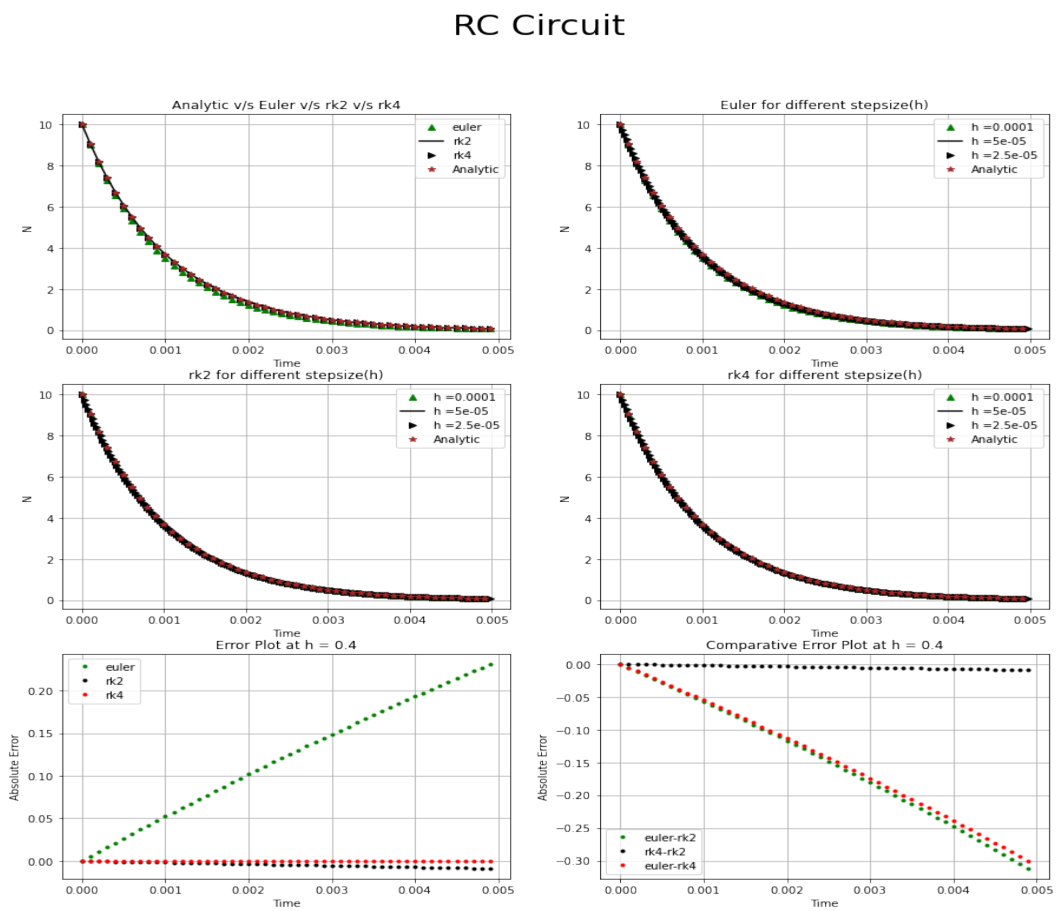


Figure 14: RC Circuit

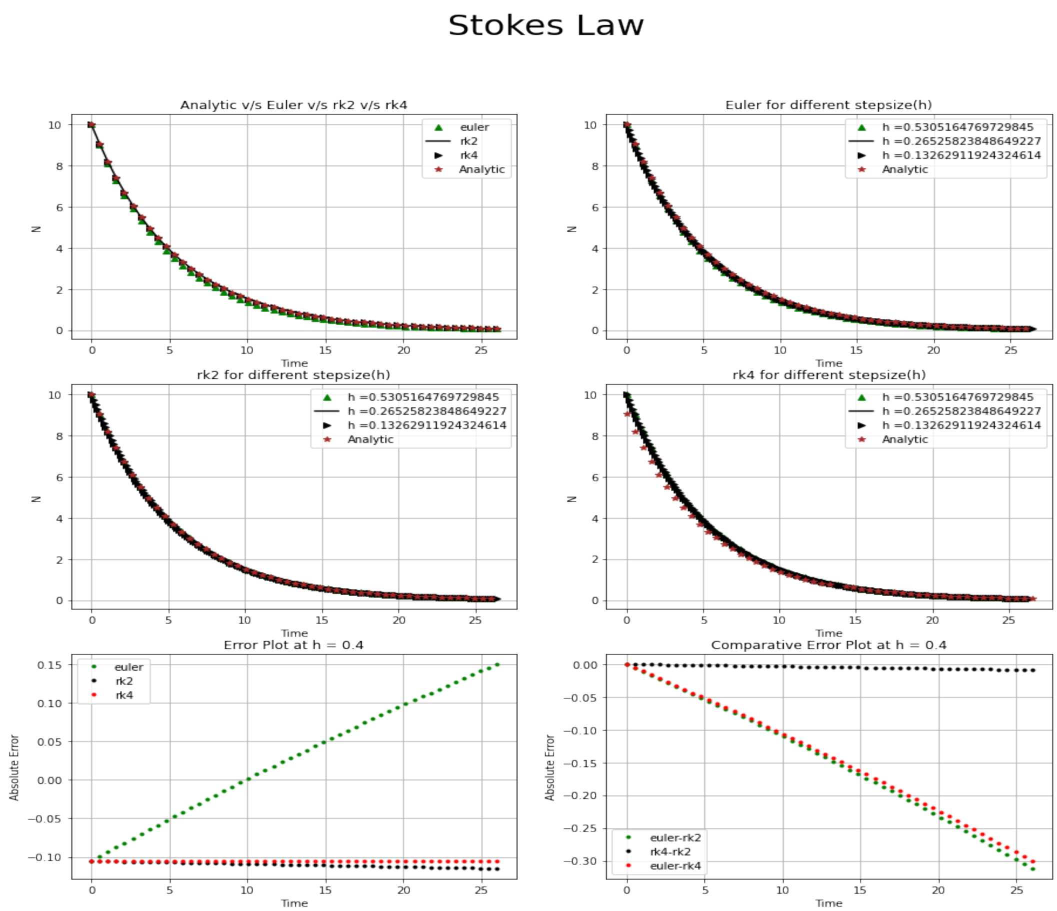


Figure 15: Stokes Law

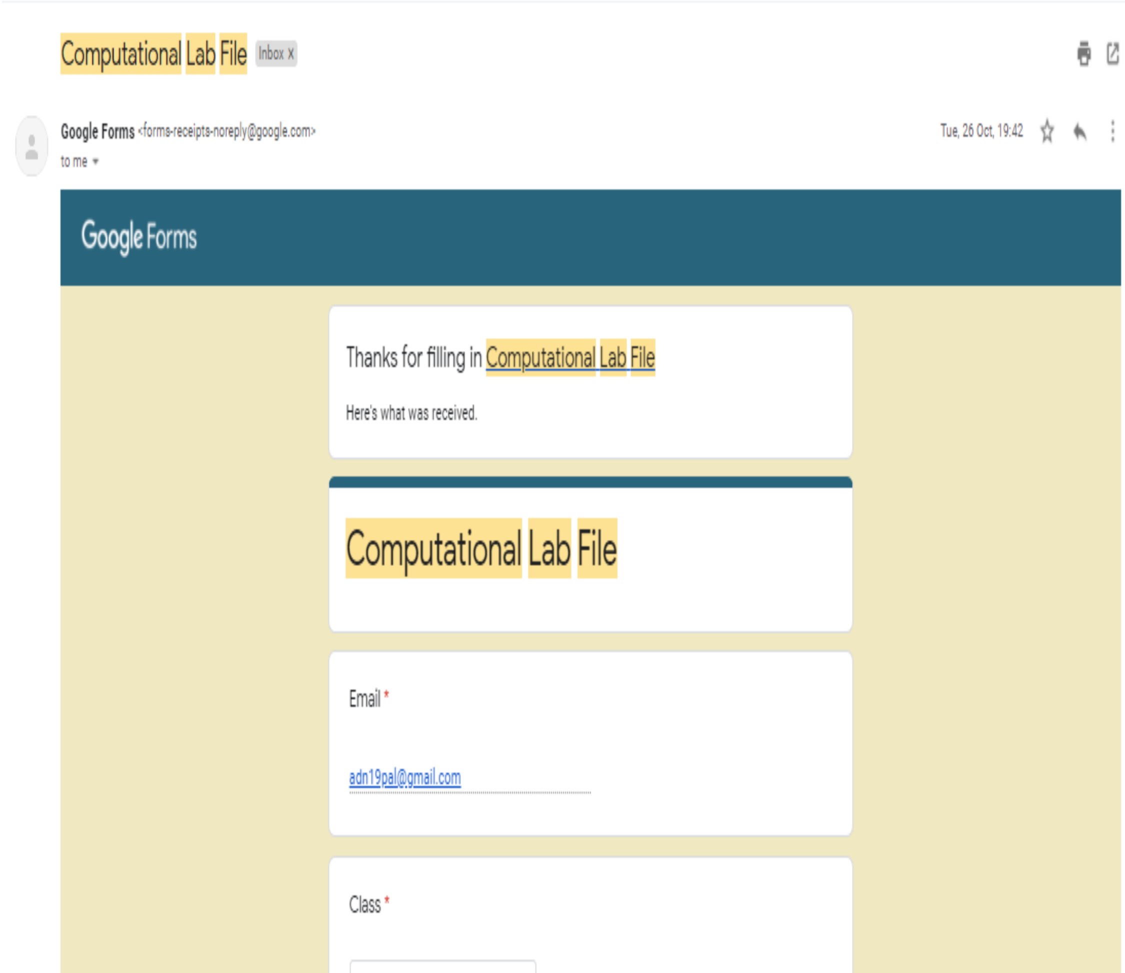


Figure 16: 2nd Order Coupled diffential equations using Euler, RK2, RK4 Method

5 2nd Order Coupled diffential equations using Euler, RK2, RK4 Method

```
import pandas as pd
#%matplotlib inline
import matplotlib.pyplot as plt
from scipy import integrate
import numpy as np
def simpe_harmonic(X, t, cons):  #simple harmonic oscillator
```

```
    k,m = cons
    x, y = X
    dotx = y
    dot2x = -k*x/m
    return np.array([dotx, dot2x])
def damped_harmonic(X, t, cons):  #damped oscillator
    k,m,b = cons
    x, y = X
    dotx = y
    doty = -k*x-b*dotx
    return np.array([dotx, doty])
def simple_pendulum(X, t, cons):  #Simple Pendulum
    g,L = cons
    x, y = X
    dotx = y
    doty = -g*x/L
    return np.array([dotx, doty])
def RK2(func, X0, t,cons):
    dt = t[1] - t[0]
    nt = len(t)
    X  = np.zeros([nt, len(X0)])
    X[0] = X0
    for i in range(nt-1):
        k1 =dt* func(X[i], t[i], cons)
        k2 = dt*func(X[i] +  k1, t[i] + dt, cons)
        X[i+1] = X[i] + (k1 +  k2 )/2
    return X, t


if __name__ == "__main__":
    k = 1; m = 1; cons=(k,m); t_p = 2*np.pi*np.sqrt(m/k); tmin=0; tmax=7*t_p; Nt
     = 1000; x0 = [2,0]
    t = np.linspace(tmin,tmax,Nt)
    Xrk2 = RK2(simpe_harmonic, x0, t, cons)
    x,y= Xrk2[0].T
    t=Xrk2[1]
    tdimless=t/t_p
    #Plotting

    plt.title('Simple Harmonic Oscillator', fontsize=20)
    plt.plot(tdimless,x,color='black',label="Displacement");plt.plot(tdimless,y,
     color='brown',label="Velocity")
    plt.grid(); plt.legend()
    plt.show()
#Damped Harmonic Oscillator
    k = 1; m = 1; t_p = 2*np.pi*np.sqrt(m/k); tmin=0; tmax=30*t_p; Nt = 1000; x0
     = [1,0]
    t = np.linspace(tmin,tmax,Nt)
    tdimless=t/t_p; dis=[];vel=[];time=[]
    ba=[0.15,2,5]
    for b in ba:
        cons=(m,k,b)
        Xrk2=RK2(damped_harmonic,x0,tdimless, cons)
        x,y= Xrk2[0].T
        tdimless=Xrk2[1]
        dis.append(x)
        vel.append(y)
```

```
    fig, axs = plt.subplots(3,figsize=(11,15))
    fig.suptitle('Damped Harmonic Oscillator', fontsize=20)
    axs[0].plot(tdimless,dis[0],label = "displacement")
    axs[0].set(xlabel="time ",title="Underdamped, b = 0.15")
    axs[0].plot(tdimless,vel[0],label = "velocity")
    axs[0].plot(tdimless,1/2*k*dis[0]**2)
    axs[0].grid(); axs[0].legend()
    axs[1].plot(tdimless,dis[1],label = "displacement")
    axs[1].set(xlabel="time ",title="Critically Damped, b = 2")
    axs[1].plot(tdimless,vel[1],label = "velocity")
    axs[1].plot(tdimless,1/2*k*dis[1]**2)
    axs[1].grid(); axs[1].legend()
    axs[2].plot(tdimless,dis[2],label = "displacement")
    axs[2].plot(tdimless,1/2*k*dis[2]**2)
    axs[2].set(xlabel="time ",title="Overdamped, b =5")
    axs[2].plot(tdimless,vel[2],label = "velocity")
    axs[2].grid(); axs[2].legend()
    plt.show()
#Simple pendulum
    g = 9.8; L = 1; cons=(g,L); t_p = 2*np.pi*np.sqrt(L/g); tmin=0; tmax=7*t_p;
     Nt = 1000; x0 = [2,0]
    t = np.linspace(tmin,tmax,Nt)
    Xrk2 = RK2(simple_pendulum, x0, t, cons)
    x,y= Xrk2[0].T
    t=Xrk2[1]
    tdimless=t/t_p
#Plotting
    plt.title('Simple Pendulum', fontsize=20)
    plt.plot(tdimless,x,color='black',label="Angular Displacement");plt.plot(
     tdimless,y,color='brown',label="Angular Velocity")
    plt.grid(); plt.legend()
    plt.show()
```

Listing 5: 2nd Order coupled diffential equations using Euler, RK2 and RK4 Method

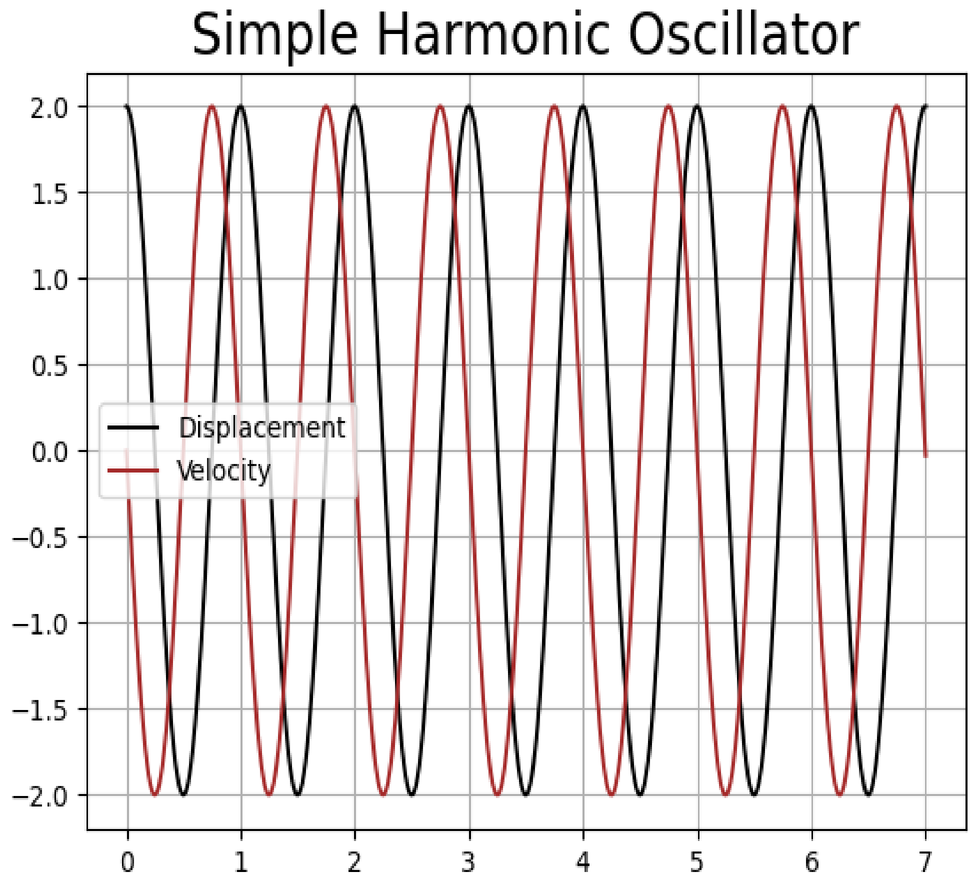


Figure 17: Simple Harmonic Oscillator

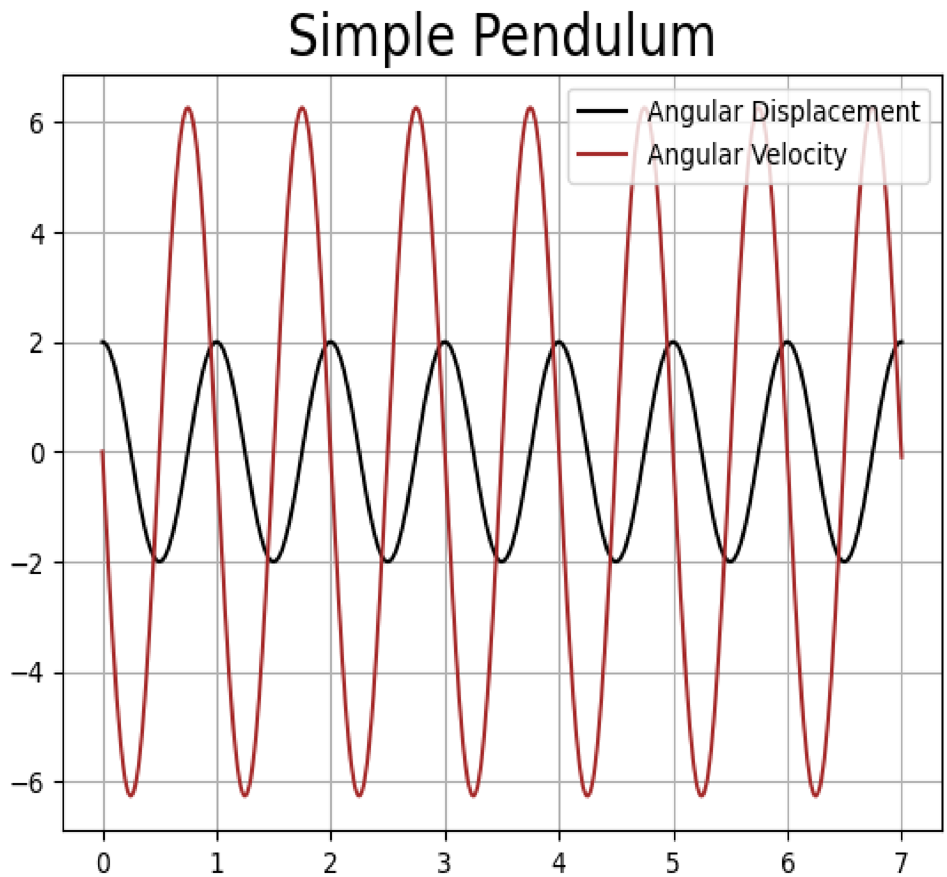


Figure 18: Simple Pendulum

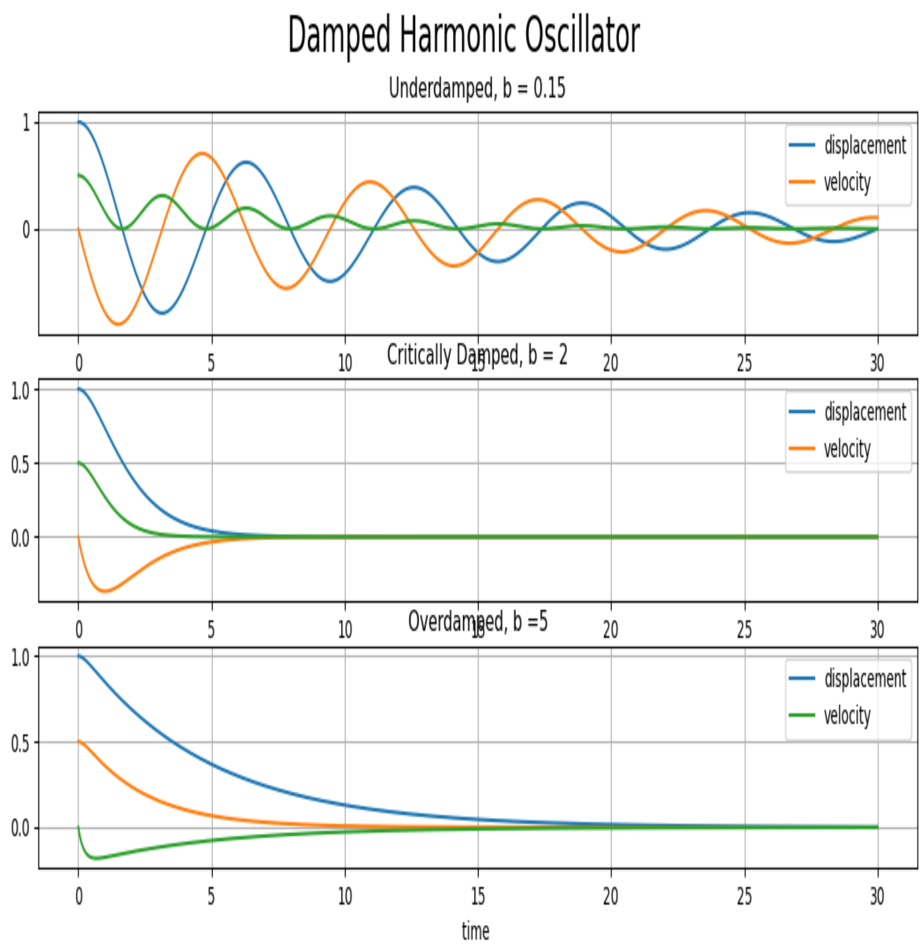


Figure 19: Damped Harmonic Oscillator

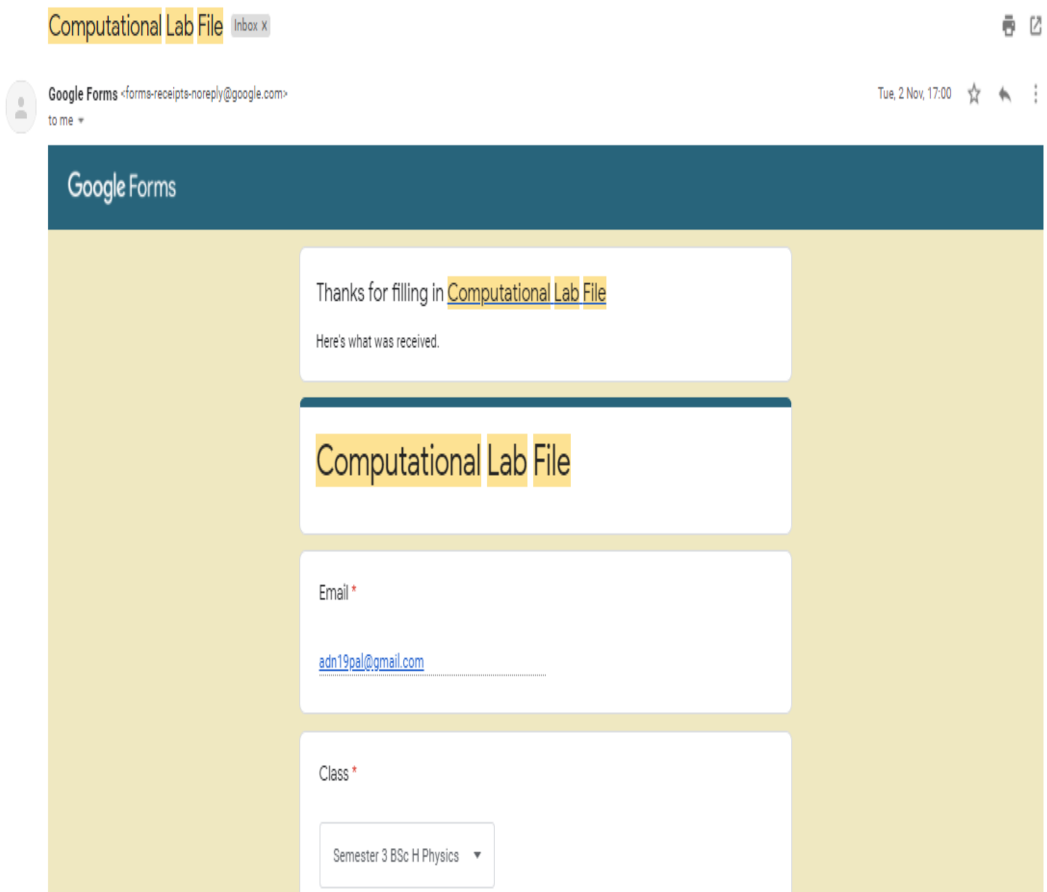


Figure 20: RK4 Method

6 RK4 Method for Simulataneous Differential Equations

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

def eqn(X, t):
    x, y = X
    dotx = y + x - x**3
    doty = -x
    return np.array([dotx, doty])
def RK4(func, X0, t):
```

```
    dt = t[1] - t[0]
    nt = len(t)
    X  = np.zeros([nt, len(X0)])
    X[0] = X0
    for i in range(nt-1):
        k1 = func(X[i], t[i])
        k2 = func(X[i] + dt/2. * k1, t[i] + dt/2.)
        k3 = func(X[i] + dt/2. * k2, t[i] + dt/2.)
        k4 = func(X[i] + dt    * k3, t[i] + dt)
        X[i+1] = X[i] + dt / 6. * (k1 + 2. * k2 + 2. * k3 + k4)
    return X

def graph(t,x,y,x1,y1,x2,y2,x3,y3,title):
    fig,axs=plt.subplots(2,2,figsize=(7,7))
    fig.suptitle(title, fontsize=30)
    ax11,ax12,ax21,ax22=axs[0][0],axs[0][1],axs[1][0],axs[1][1]

    ax11.scatter(t,x,color='black',label="x"),ax11.scatter(t,y,color='brown',
     label="y")
    ax11.set_title("x(0)=0 & y(0)=-1")
    ax12.scatter(t,x1,color='black',label="x"),ax12.scatter(t,y1,color='brown',
     label="y")
    ax12.set_title("x(0)=0 & y(0)=-2")
    ax21.scatter(t,x2,color='black',label="x"),ax21.scatter(t,y2,color='brown',
     label="y")
    ax21.set_title("x(0)=0 & y(0)=-3"),ax21.set_xlabel("Time")
    ax22.scatter(t,x3,color='black',label="x"),ax22.scatter(t,y3,color='brown',
     label="y")
    ax22.set_title("x(0)=0 & y(0)=-4"),ax22.set_xlabel("Time")

    ax11.legend(),ax11.grid(True),ax12.legend(),ax12.grid(True),ax21.legend(),
     ax21.grid(True),ax22.legend(),ax22.grid(True)
    plt.show()
def graph1(x,y,x1,y1,x2,y2,x3,y3,title):
    fig,axs=plt.subplots(2,2,figsize=(7,7))
    fig.suptitle(title, fontsize=30)
    ax11,ax12,ax21,ax22=axs[0][0],axs[0][1],axs[1][0],axs[1][1]
    ax11.scatter(x,y,color='black')
    ax11.set_title("1st condition");ax11.set_ylabel("y")
    ax12.scatter(x1,y1,color='black')
    ax12.set_title("2nd condition");
    ax21.scatter(x2,y2,color='black')
    ax21.set_title("3rd condition");ax21.set_xlabel("x");ax21.set_ylabel("y")
    ax22.scatter(x3,y3,color='black')
    ax22.set_title("4th condition");ax22.set_xlabel("x")
    ax11.legend(),ax11.grid(True),ax12.legend(),ax12.grid(True),ax21.legend(),
     ax21.grid(True),ax22.legend(),ax22.grid(True)
    plt.show()

if __name__ == "__main__":
    x0 = [0,0,0,0];y0 = [-1,-2,-3,-4];Nt = 100;tmax = 15
    t = np.linspace(0.,tmax, Nt)
    X0 = [x0[0], y0[0]]
    X1 = [x0[1], y0[1]]
    X2 = [x0[2], y0[2]]
    X3 = [x0[3], y0[3]]
    res = RK4(eqn, X0, t)
```

```
    res1 = RK4(eqn, X1, t)
    res2 = RK4(eqn, X2, t)
    res3 = RK4(eqn, X3, t)
    x, y = res.T;
    x1, y1 = res1.T
    x2, y2 = res2.T
    x3, y3 = res3.T
    graph(t,x,y,x1,y1,x2,y2,x3,y3,"RK4 Method")
    graph1(x,y,x1,y1,x2,y2,x3,y3,"x vs y")
    data = {"Time":t,"x_rk4":x2,"y_rk4":y2}
    print(pd.DataFrame(data))
```

Listing 6: RK4 Method for Simulataneous Differential Equations

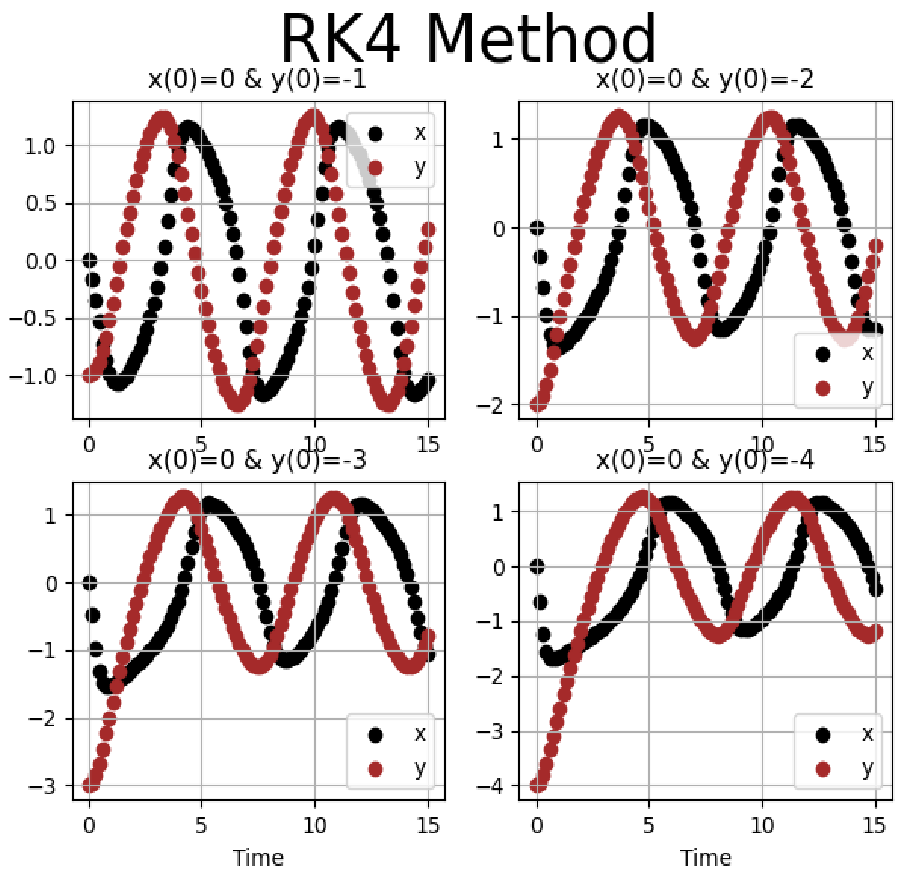


Figure 21: RK4 Method

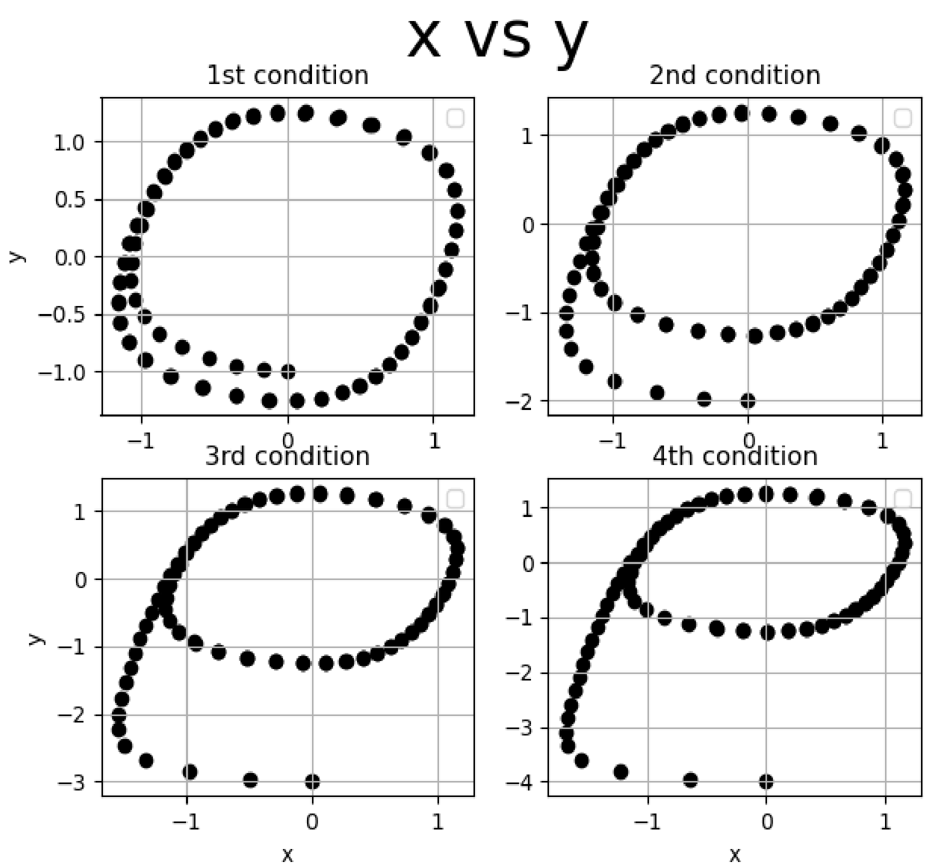


Figure 22: RK4 Method

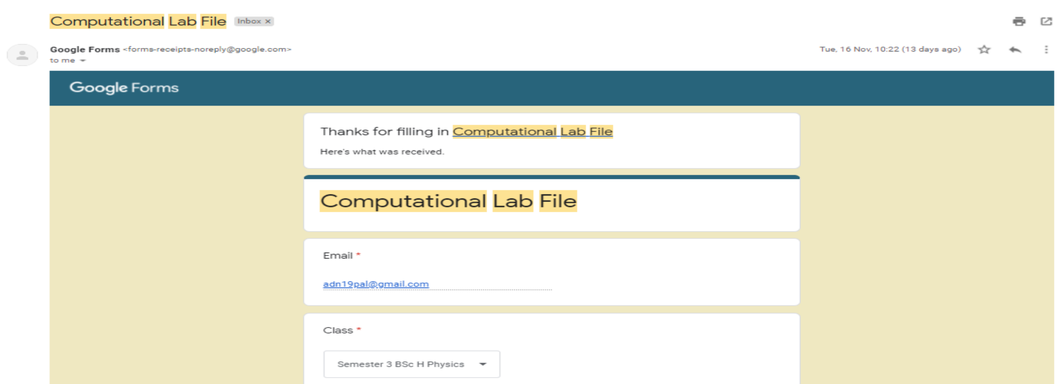


Figure 23: Gauss Elimination Method Output

7 Gauss Elimination Method

```
import numpy as np
def elimination(a,b):
    row, col = np.shape(a)
    if row == col:          #condition to check square matrix
        n = len(b)
        x  = np.zeros(n, float)
        for p in range(n-1):        #index the pivots
            if a[p,p] < 1e-12: #putting threshold condition to check pivot not to
 be zero
                for i in range(p+1,n):  #index elements under pivot
                    if np.fabs(a[i,p]) > np.fabs(a[p,p]):  #non zero rows under
 pivot element
                        a[[p,i]] = a[[i,p]]   #interchange rows and columns
                        b[[p,i]] = b[[i,p]]
                        break
            for i in range(p+1,n):
                if a[i,p] == 0:     #skip the row
                    continue
                factor = a[p,p]/a[i,p]
                for j in range(p,n):
                    a[i,j] = a[p,j] - a[i,j]*factor
                b[i] = b[p] - b[i]*factor     #elimination
        '''check rank of matrix to determine consistency'''
        rank_a = np.linalg.matrix_rank(a)
        aug_matrix = np.column_stack((a,b.T))
        aug_matrix_rank = np.linalg.matrix_rank(aug_matrix)
        if rank_a == aug_matrix_rank:
            if aug_matrix_rank == np.shape(b)[0]:
                print("System of equations has unique solution")
                print("Augmented Matrix \n" ,np.column_stack((a,b.T)))
                '''Back Substitution'''
                x[n-1] = b[n-1] / a[n-1,n-1]
                for i in range(n-2,-1, -1):
```

```python
                sum_ax = 0
                for j in range(i+1, n):
                    sum_ax += a[i, j]* x[j]
                x[i] = (b[i] - sum_ax) / a[i,i]

            print('The solution of the system: ')
            print(x)
            numpy_solution = np.linalg.solve(a, b)
            print("Solution by inbuilt function", numpy_solution)
        elif aug_matrix_rank < np.shape(b)[0]:
            print("system has infinitely many solutions")
    elif rank_a < aug_matrix_rank:
        print("System is inconsistent")
  else:
    print("Number of rows should be equal to number of columns")


if __name__ == "__main__":
    print("E.g. 1")
    a = np.array([[1,-2,1],
                [2,-5,4],
                [1,-4,6]], float)
    b = np.array([5,-3,10], float)
    elimination(a,b)

    print("E.g. 2")
    a = np.array([[1,-2,1,-1,1],
                [2,-5,4,1,-1],
                [1,-4,6,2,-1]], float)
    b = np.array([5,-3,10], float)
    elimination(a,b)
    print("E.g. 3")
    a = np.array([[1,-5,4],
                [1,-5,3],
                [2,-10,13]], float)
    b = np.array([3,6,5], float)
    elimination(a,b)

    print("E.g. 4")
    a = np.array([[12,-3],
                [-12,3]], float)
    b = np.array([6,-6], float)
    elimination(a,b)
```

Listing 7: Gauss Elimination

```
(base) hinton@hinton-VirtualBox:~$ /home/hinton/anaconda3/bin/python "/home/hinton/Sem3/MP2/Practical/code/Gauss Elimination/gauss_elim.py"
E.g. 1
System of equations has unique solution
Augmented Matrix
 [[ 1.   -2.    1.    5.  ]
 [ 0.    0.5  -1.    6.5 ]
 [ 0.    0.    0.25  7.75]]
The solution of the system:
[124.  75.  31.]
Solution by inbuilt function [124.  75.  31.]
E.g. 2
Number of rows should be equal to number of columns
E.g. 3
System is inconsistent
E.g. 4
system has infinitely many solutions
```

Figure 24: Gauss Elimination Method Output

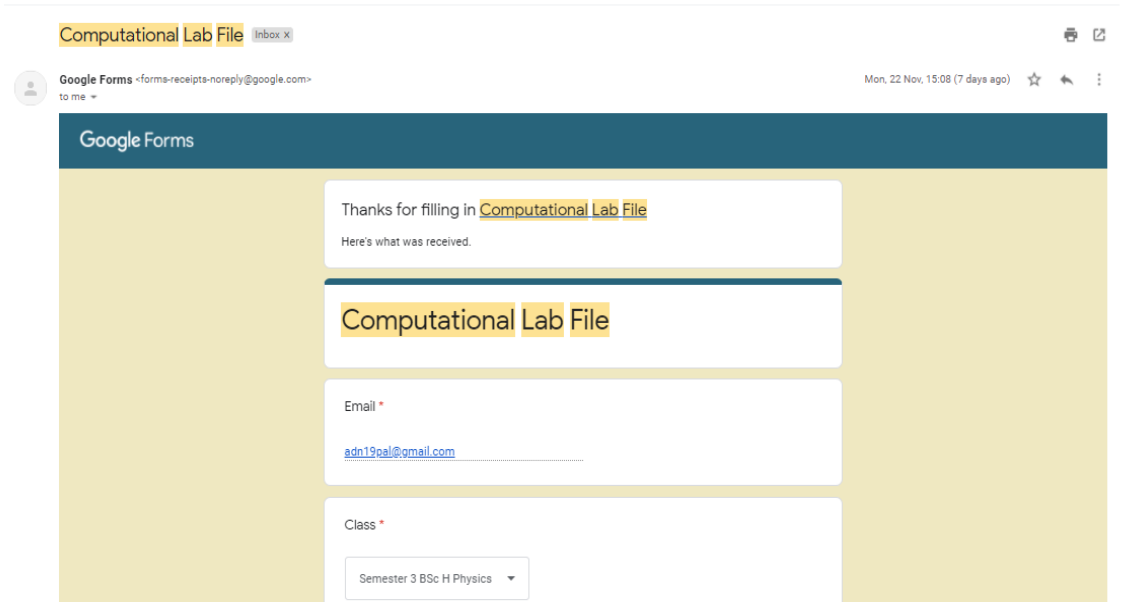


Figure 25: Gauss Seidel Method Output

8 Gauss Seidel Method

```
import numpy as np
from scipy.linalg import solve
def gaussseidel(A, B):
    row, col = np.shape(A)
    if row == col:
        n = 10000
        x = B/(np.diagonal(A))
        inbuilt = solve(A,B)
        for i in range(1, n):
            x_new = np.zeros_like(x)
            print(x)
            for i in range(row):
                x_new[i] = (B[i] - np.dot(A[i, :i], x_new[:i]) - np.dot(A[i, i + 1 :], x[i + 1 :])) / A[i, i]
            if np.allclose(x, x_new, rtol=1e-8):
                break
            else:
                x = x_new
        print("Solution: ",x)
        error = np.dot(A, x) - B
        print("Inbuilt Solution",solve(A,B))
        print("Error: ",error)
    else:
        print("Matrix is not square matrix")
```

```python
if __name__ == "__main__":
    A = np.array([[8, 3, -3], [-2, -8, 5], [3, 5, 10]])
    # initialize the RHS vector
    B = np.array([14,5,-8])
    # Find diagonal coefficients
    diag = np.diag(np.abs(A))
    # Find row sum without diagonal
    off_diag = np.sum(np.abs(A), axis=1) - diag
    if np.all(diag > off_diag):
        print('matrix is diagonally dominant')
    else:
        print('NOT diagonally dominant')
    gaussseidel(A,B)
```

Listing 8: Gauss Seidel Method

```
(base) hinton@hinton-VirtualBox:~$ /home/hinton/anaconda3/bin/python /home/hinton/Sem3/MP2/Practical/code/gaussseidel/gaussseidel.py
matrix is diagonally dominant
[ 1.75  -0.625 -0.8  ]
[ 1.684375   -1.54609375 -0.53226562]
[ 2.13018555 -1.4902124  -0.69394946]
[ 2.0485986  -1.57086806 -0.62914555]
[ 2.10314594 -1.54400245 -0.65894256]
[ 2.08189746 -1.55731346 -0.64591251]
[ 2.09177536 -1.55163916 -0.65171303]
[ 2.0874723  -1.55418872 -0.64914733]
[ 2.08939052 -1.55306471 -0.6502848 ]
[ 2.08854247 -1.55356362 -0.64978093]
[ 2.08891851 -1.55334271 -0.6500042 ]
[ 2.08875194 -1.55344061 -0.64990528]
[ 2.08882575 -1.55339724 -0.64994911]
[ 2.08879305 -1.55341645 -0.64992969]
[ 2.08880754 -1.55340794 -0.64993829]
[ 2.08880112 -1.55341171 -0.64993448]
[ 2.08880396 -1.55341004 -0.64993617]
[ 2.0888027  -1.55341078 -0.64993542]
[ 2.08880326 -1.55341045 -0.64993575]
[ 2.08880301 -1.5534106  -0.6499356 ]
[ 2.08880312 -1.55341053 -0.64993567]
[ 2.08880307 -1.55341056 -0.64993564]
Solution:  [ 2.08880307 -1.55341056 -0.64993564]
Inbuilt Solution [ 2.08880309 -1.55341055 -0.64993565]
Error:  [-1.72144553e-07  1.44201799e-07  8.88178420e-16]
```

Figure 26: Gauss Seidel Method Output