Mathematical Physics III

Lab Assignment #8

Kabir Sethi

College Roll No	:	2020PHY1097
University Roll No	:	20068567031
Unique Paper Code	:	32221401
Paper Title	:	Mathematical physics III Lab
Course and Semester	:	B.Sc.(Hons.) Physics, Sem IV
Date of Submission	:	12 April 2022
Lab Report File Name	:	2020PHY1097_A8.pdf
Submitted to	:	Dr. Mamta Dahiya

Shri Guru Tegh Bahadur Khalsa College, University of Delhi
New Delhi-110007, India.

Contents

1 Theory

DATE : __/__/____
PAGE : _____

Theory

(a) Explain how will you transform the m^{th} order IVP

$$\frac{d^m y}{dx^m} = f(x, y, y', y'', \ldots y^{m-1})$$

with initial conditions

$y(x_0) = \alpha_1$

$y'(x_0) = \alpha_2$

$y''(x_0) = \alpha_3$

$y^m(x_0) = \alpha_{m-1}$

Sol. We have m^{th} order IVP

$$\frac{d^m y}{dx^m} = f(x, y, y' \ldots y^{m-1})$$

Let $x_1 = y \rightarrow x_1' = y' = x_2$

$x_2 = y' \Rightarrow x_2' = y'' = x_3$

$\vdots \quad \vdots$

$x_{m-1} = y^{m-2} \Rightarrow x_{m-1}' = y^{m-1} = x_m$

$x_m = y^{m-1} \rightarrow x_m' = y^m$

$x_m' = y^m \rightarrow \frac{d^m y}{dx^m} = f(x_0, x_1, x_2 \ldots x_m)$

DATE : __/__/__
PAGE : _____

So we got system of m-equations with initial condition

$$x_1(x_0) = \alpha_1$$
$$x_2(x_0) = \alpha_2$$
$$x_3(x_0) = \alpha_3$$
$$x_m(x_0) = \alpha_{m-1}$$

(b) We have

$$y'' - 2y^2 + 2y = e^{2x}\sin x \quad -(1)$$

first we'll convert it into 2 first order DE

let $\frac{dy}{dx} = u$ $\quad -(2)$

So

$$\frac{du}{dx} - 2u + 2y = e^{2x}\sin x$$

$$\frac{du}{dx} = e^{2x}\sin x + 2u - 2y \quad -(3)$$

$f_1(x, y, u) \Rightarrow \frac{dy}{dx} = u$

$f_2(x, y, u) = \frac{du}{dx} = e^{2x}\sin x + 2u - 2y$

Initial condition

$y(0) = -0.4 \qquad y'(0) = u(0) = -0.6$

for $0 \le x \le 1$

Step size $h = \frac{b-a}{N}$

$= \frac{1-0}{5} = 0.2$

DATE : __/__/__
PAGE : _____

RK2- method

$$y_{i+1} = y_i + \left(\frac{k_1 + k_2}{2}\right)$$

$k_1 = h * f(x_i, y_i)$ $\quad k_2 = h * f(x_i + h, y_i + k_1)$

Step 1

$k_1 = h * f(x_0, y_1, u_0)$

$= 0.2 * f_1(0, -0.4, -0.6)$

$= 0.2 * (-0.6) = -0.12$

$k_2 = h * f_2(x_0, y_0, u_0)$

$= 0.2 * f_2(0, -0.4, -0.6)$

$= 0.2 [0.8 - 1.2]$

$= -0.08$

$k_2 = h * f_1(x_0 + h, y_0 + k_1, u_0 + k_1)$

$= 0.2 * -0.68$

$= -0.136$

$k_2 = h f_2(x_0 + h, y_0 + k_1, u_0 + k_1)$

$0.2 * (-0.023620,9)$

$= -6.00472404$

DATE : __/__/____
PAGE : _____

So $y_{i+1} = y_i + \frac{k_1+k_2}{2}$

for f_1 $i=0$

$$y = f_0 + \frac{k_1+k_2}{2}$$

$$= -0.4 \left(\frac{-012 + (-0.131)}{2} \right)$$

$$= -0.520$$

for f_2

$$y_1 = y_0 + \left(\frac{k_1+k_2}{2} \right)$$

$$= -0.4 + \left(\frac{-0.08 - 0.00472404}{2} \right)$$

$$= 0.6423620$$

~~Step~~ we started with initial condition.
Now we have to fine the next term with
$y_1 = -0.528$ $y_1 = -0.64236202$

$x \rightarrow x+h \quad \Rightarrow 02$

DATE :__/__/__
PAGE :_____

$k_1 = h^x f_1(x_0+h, y_1, u_1)$

$h^x(-0.64236202)$

$= -0.1284724$

$k_1 = h^x f_2(x_0+h, y_1, u_1)$

$= h^x(+0.0676558)$

$= +0.01353116$

$k_2 = h^x f_1(x_0+2h, y_1+k_1, u_1+k_1)$

$= -0.12557617$

$k_2 = h^x f_2(x_0+2h, y_1+k_1, u_1+k_1)$

$= h^x(0.92194454)$

$= 0.18438891$

So $y_2 = y_1 + \left(\frac{k_1+k_2}{2}\right)$, $u_2 = u_1 + \left(\frac{k_1+k_2}{2}\right)$

$y_2 = -0.65511929$ $u_2 = -0.54340149$

Step3

$k_1 = h^x f_1(x_0+2h, y_2, u_2)$

$= h^x(-0.54340149) = -0.10868808$

$k_1 = h^x f_2(x_0+2h, y_2, u_2)$

$= h^x(1.0901020b) = 0.21802041$

$k_2 = h^x f_1(x_0+3h, y_2+k_1, u_2+k_1)$

$= h^x(-0.32538807) = -0.06507621$

DATE : __/__/____
PAGE : _____

$k_2 = h^* f_2(x_0+3h, y_2+k_1, u_2+k_1)$

$= h^* (2.75151605)$

$= 0.55030321$

Next terms

$y_2 = y_2 + \left(\frac{k_1+k_2}{2}\right)$ $\qquad u_2 = u_2 + \left(\frac{k_1+k_2}{2}\right)$

$y_3 = -0.74194951$ $\qquad u_2 = -0.15923460$

Step-4

$k_1 = h^* f_1(x_0+3h, y_2, u_2)$

$= -0.07184794$

$k_1 = h^* f_2(x_0+3h, y_3, u_3)$

$= 0.60807895$

$k_2 = h^* f_1(x_0+4h, y_3+k_1, u_3+k_1)$

$= h^* (0.44874920) = 0.08925488$

$k_2 = h^* f_2(x_0+4h, u_3+k_1+u_3+k_1)$

$= h^* (5.4987744) = 1.1996755$

Next term

$y_4 = y_3 + \left(\frac{k_1+k_2}{2}\right)$ $\qquad u_4 = u_3 + \left(\frac{k_1+k_2}{2}\right)$

$y_4 = -0.71304158$ $\qquad u_4 = 0.74461755$

DATE : / /
PAGE :

Step-5

$k_1 = h^* f_1 (x_4 + 4h, y_4 + 4u_4)$

$= h^* (0.79961744) = 0.14089231$

$k_1 = h^* f_2 (x_4 + 4h, y_4, u_4)$

$h^* (.46840675) = 2.29360123$

$k_2 = h^* f_1 (x_0 + 5h, y_4 + k_1, u_4 + k_1)$

$= 0.40765471$

$k_2 = h^* f_2 (x_0 + 5h, y_4 + k_1, u_4 + k_1)$

$= 2.28450201$

So

$y_5 = -0.43479495 \qquad u_5 = 2.53370918$

2 Programming

2.1 Module_A8_1097.py

```
import numpy as np

def euler(Func,IC,a,b,N):
    h=((b-a)/N)
    t=np.linspace(a,b,N+1)
    X=np.zeros([N+1,len(IC)])
    X[0]=IC
    for i in range(N):
        X[i+1] = X[i] + h*Func(t[i],X[i])
    return X,t

def RK2(Func,IC,a,b,N):
    h=((b-a)/N)
    t = np.linspace(a,b,N+1)
    X  = np.zeros([N+1, len(IC)])
    X[0] =IC
    for i in range(N):
        k1 =h* Func(t[i],X[i])
        k2 = h*Func( t[i] + h,X[i] +  k1)
        X[i+1] = X[i] +  (k1 +k2 )/2
    return X,t

def RK2_M(Func,IC,a,b,N):
    h=((b-a)/N)
    t = np.linspace(a,b,N+1)
    X  = np.zeros([N+1, len(IC)])
    X[0] =IC
    for i in range(N):
        k1 =h* Func(t[i],X[i])
        k2 = h*Func( t[i] + h/2,X[i] +  k1/2)
        X[i+1] = X[i] +  (k1 +k2 )/2
    return X,t

def RK4(Func,IC,a,b,N):
    h=((b-a)/N)
    t = np.linspace(a,b,N+1)
    X  = np.zeros([N+1, len(IC)])
    X[0] =IC
```

```
    for i in range(N):
        k1 =h* Func(t[i],X[i])
        k2 =h*Func(t[i] + h/2,X[i] +  k1/2)
        k3 =h*Func(t[i]+h/2,X[i] + k2/2)
        k4 =h*Func(t[i]+h,X[i]+k3)
        X[i+1] = X[i] +  (k1+2*k2+2*k3+k4)/6
    return X,t
```

2.2 tol_A8_1097.py

```
from MyIVP import euler
from MyIVP import RK2
from MyIVP import RK4
import numpy as np


def euler_tol(Func, IC ,a,b,N,N_max,tol):
    w=0
    Val=[]
    N_arr=[]
    count=0
    while N<=N_max:
        g=euler(Func, IC ,a,b,N)[0].T
        t=euler(Func, IC ,a,b,N)[1]
        Val.append(g)
        N_arr.append(N)
        if count>=1:
            J=[]
            K=[]
            for i in range(len(IC)):
                J.append(Val[-1][i][-1])
                K.append(Val[-2][i][-1])
            J=np.array(J)
            K=np.array(K)
            ff=[]
            for g1,g2 in zip(J,K):
                if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
                    err=abs(g1-g2)
                else:
                    err=abs((g2-g1)/g1)
                ff.append(err)
            if max(ff)<=tol:
```

```
                w=1
                break
            else:
                pass

        N=2*N
        count+=1
    if w==0:
        s=("N_max reached without achieving required tolerance")
    elif w==1:
      s="Given tolerance achieved with",N_arr[-1],"sub-intervals"

    return Val,Val[-1],N_arr[-1],N_arr,s,g,t

def RK2_tol(Func, IC ,a,b,N,N_max,tol):
    w=0
    Val=[]
    N_arr=[]
    count=0
    while N<=N_max:
        g=RK2(Func, IC ,a,b,N)[0].T
        t=RK2(Func, IC ,a,b,N)[1]
        Val.append(g)
        N_arr.append(N)
        if count>=1:
            J=[]
            K=[]
            for i in range(len(IC)):
                J.append(Val[-1][i][-1])
                K.append(Val[-2][i][-1])
            J=np.array(J)
            K=np.array(K)
            ff=[]
            for g1,g2 in zip(J,K):
                if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
                    err=abs(g1-g2)
                else:
                    err=abs((g2-g1)/g1)
                ff.append(err)
            if max(ff)<=tol:
                w=1
                break
```

```
            else:
                pass

        N=2*N
        count+=1
    if w==0:
        s=("N_max reached without achieving required tolerance")
    elif w==1:
      s="Given tolerance achieved with",N_arr[-1],"sub-intervals"

    return Val,Val[-1],N_arr[-1],N_arr,s,g,t

def RK4_tol(Func, IC ,a,b,N,N_max,tol):
    w=0
    Val=[]
    N_arr=[]
    count=0
    while N<=N_max:
        g=RK4(Func, IC ,a,b,N)[0].T
        t=RK4(Func, IC ,a,b,N)[1]
        Val.append(g)
        N_arr.append(N)
        if count>=1:
            J=[]
            K=[]
            for i in range(len(IC)):
                J.append(Val[-1][i][-1])
                K.append(Val[-2][i][-1])
            J=np.array(J)
            K=np.array(K)
            ff=[]
            for g1,g2 in zip(J,K):
                if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
                    err=abs(g1-g2)
                else:
                    err=abs((g2-g1)/g1)
                ff.append(err)
            if max(ff)<=tol:
                w=1
                break
            else:
                pass
```

```
117
118         N=2*N
119         count+=1
120     if w==0:
121         s=("N_max reached without achieving required tolerance")
122     elif w==1:
123       s="Given tolerance achieved with",N_arr[-1],"sub-intervals"
124
125     return Val,Val[-1],N_arr[-1],N_arr,s,g,t
```

2.3 1097_Main_A8.py

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from MyIVP_1097 import euler
4  from MyIVP_1097  import RK2
5  from MyIVP_1097  import RK4
6
7  def Func(x,Var):
8      y1,y2,y3=Var
9      f1=y2-y3+x
10     f2=3*x**2
11     f3=y2 + np.exp(-x)
12     return np.array([f1,f2,f3])
13
14 def output(Func,IC,a,b,N):
15     X=euler(Func,IC,a,b,N)
16     l1=X[0].T
17     X1=RK2(Func,IC,a,b,N)
18     l2=X1[0].T
19     X3=RK4(Func,IC,a,b,N)
20     l3=X3[0].T
21     t=X[1]
22
23     return l1,l2,l3,t
24
25 def table(data,head,Title):
26     print(Title)
27     line='_'*len(head)*12+'____'
28     for i in head:
29         print("{0:^12}".format(i),end=" ")
30     print("\n",line)
```

```
    for row in data:
        for val in row:
            print("{0:^12.2}".format(val), end=" ")
        print("\n")

IC=[1,1,-1]
a=0
b=1
N=100

Ana_Sol_1= lambda x : -0.05*x**5 + 0.25*x**4 + x +2 - np.exp(-x)
Ana_Sol_2= lambda x :  x**3 +1
Ana_Sol_3= lambda x :  0.25*x**4 + x - np.exp(-x)

X,Y,Z,t=output(Func,IC,a,b,N)

EUL_data=np.column_stack((t,X[0],X[1],X[2]))
RK2_data=np.column_stack((t,Y[0],Y[1],Y[2]))
RK4_data=np.column_stack((t,Z[0],Z[1],Z[2]))
heading=["x","Y1","Y2","Y3"]

table(EUL_data,heading,"\nEULER METHOD FOR N=100")
table(RK2_data,heading,"\nRK2 METHOD FOR N=100")
table(RK4_data,heading,"\nRK4 METHOD FOR N=100")

def graph(t,X,title):
    for i in range(len(X)):
        plt.plot(t,X[i],label="Y"+str(i))
    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("$y_i$")
    plt.legend()
    plt.grid()

def graph_ana(t,Analytic):
    for g,i in zip(Analytic,range(len(Analytic))):
        plt.plot(t,g(t),label="Analytic_Y"+str(i),ls="--",c="black")
    plt.legend()

Analytic=[Ana_Sol_1,Ana_Sol_2,Ana_Sol_3]

graph(t,X,"\nEULER METHOD FOR N=100")
```

```
graph_ana(t,Analytic)
plt.show()

graph(t,Y,"\nRK2 METHOD FOR N=100")
graph_ana(t,Analytic)
plt.show()

graph(t,Z,"\nRK4 METHOD FOR N=100")
graph_ana(t,Analytic)
plt.show()

N_arr=[]
E_err_1=[];E_err_2=[];E_err_3=[]
RK2_err_1=[];RK2_err_2=[];RK2_err_3=[]
RK4_err_1=[];RK4_err_2=[];RK4_err_3=[]

def plot3(X,t,N):
    for i in range(len(X)):
        plt.plot(t,X[i],label="$N={0}$".format(N))

for i in range(0,6,1):
    N=10**(i)
    N_arr.append(N)
    X,Y,Z,t=output(Func,IC,a,b,N)
    plot3(X,t,N)
    f1=Ana_Sol_1(t)
    f2=Ana_Sol_2(t)
    f3=Ana_Sol_3(t)
    E_err_1.append(max(np.array(f1)-np.array(X[0])))
    E_err_2.append(max(np.array(f2)-np.array(X[1])))
    E_err_3.append(max(np.array(f3)-np.array(X[2])))

graph_ana(t,Analytic)
plt.title("Euler Method for different N")
plt.xlabel("x")
plt.ylabel("$y_i$")
plt.legend()
plt.grid()
plt.show()

for i in range(0,6,1):
    N=10**(i)
```

```
    X,Y,Z,t=output(Func,IC,a,b,N)
    plot3(Y,t,N)
    f1=Ana_Sol_1(t)
    f2=Ana_Sol_2(t)
    f3=Ana_Sol_3(t)
    RK2_err_1.append(max(np.array(f1)-np.array(Y[0])))
    RK2_err_2.append(max(np.array(f2)-np.array(Y[1])))
    RK2_err_3.append(max(np.array(f3)-np.array(Y[2])))

graph_ana(t,Analytic)
plt.title("RK2 Method for different N")
plt.xlabel("x")
plt.ylabel("$y_i$")
plt.legend()
plt.grid()
plt.show()

for i in range(0,6,1):
    N=10**(i)
    X,Y,Z,t=output(Func,IC,a,b,N)
    plot3(Z,t,N)
    f1=Ana_Sol_1(t)
    f2=Ana_Sol_2(t)
    f3=Ana_Sol_3(t)
    RK4_err_1.append(max(np.array(f1)-np.array(Z[0])))
    RK4_err_2.append(max(np.array(f2)-np.array(Z[1])))
    RK4_err_3.append(max(np.array(f3)-np.array(Z[2])))

graph_ana(t,Analytic)
plt.title("RK4 Method for different N")
plt.xlabel("x")
plt.ylabel("$y_i$")
plt.legend()
plt.grid()
plt.show()

print("\n")
data=np.column_stack((N_arr,E_err_1,RK2_err_1,RK4_err_1,E_err_2,RK2_err_2,
   RK4_err_2,E_err_3,RK2_err_3,RK4_err_3))
head=["N","E_y1(Euler)","E_y1(RK2)","E_y1(RK4)","E_y2(Euler)","E_y2(RK2)",
   "E_y2(RK4)","E_y3(Euler)","E_y3(RK2)","E_y3(RK4)"]
table(data,head,"\nTable for N and E= max(|y_ana -y_num|) values for y0,y1
```

```
    and y2 for all three methods  ")

fig, (ax1,ax2,ax3) = plt.subplots(3,sharex=True)
fig.suptitle("log(E) vs log(N) plot",fontsize=15,c="r")
ax1.plot(N_arr,E_err_1,label="$y_{0}(Euler)$")
ax1.plot(N_arr,RK2_err_1,label="$y_{0}(RK2)$")
ax1.plot(N_arr,RK4_err_1,label="$y_{0}(RK4)$")
ax1.set(ylabel="log(E)= log(max(|y_ana -y_num|))",yscale="log",xscale="log
    ")
ax1.grid()
ax1.legend()

ax2.plot(N_arr,E_err_2,label="$y_{1}(Euler)$")
ax2.plot(N_arr,RK2_err_2,label="$y_{1}(RK2)$")
ax2.plot(N_arr,RK4_err_2,label="$y_{1}(RK4)$")
ax2.set(ylabel="log(E)= log(max(|y_ana -y_num|))",xscale="log",yscale="log
    ")
ax2.grid()
ax2.legend()

ax3.plot(N_arr,E_err_3,label="$y_{2}(Euler)$")
ax3.plot(N_arr,RK2_err_3,label="$y_{2}(RK2)$")
ax3.plot(N_arr,RK4_err_3,label="$y_{2}(RK4)$")
ax3.set(xlabel="log(N)",ylabel="log(E)= log(max(|y_ana -y_num|))",xscale="
    log",yscale="log")
ax3.grid()
ax3.legend()
plt.show()

x_f=[1,2.5,5.0,7.5,10]
for i in x_f:
    l1,l2,l3,t=output(Func,IC,0,i,100)
    for k in range(len(l1)):
        plt.plot(t,l1[k],label="Y at xf="+str(i))
    plt.title("euler")
    plt.xlabel("x")
    plt.ylabel("$y_i$")
    plt.legend()
    plt.grid()
plt.show()

for i in x_f:
```

```
    l1,l2,l3,t=output(Func,IC,0,i,100)
    for k in range(len(l1)):
        plt.plot(t,l2[k],label="Y at xf="+str(i))
    plt.title("RK2")
    plt.xlabel("x")
    plt.ylabel("$y_i$")
    plt.legend()
    plt.grid()
plt.show()

for i in x_f:
    l1,l2,l3,t=output(Func,IC,0,i,100)
    for k in range(len(l1)):
        plt.plot(t,l3[k],label="Y at xf="+str(i))
    plt.title("RK4")
    plt.xlabel("x")
    plt.ylabel("$y_i$")
    plt.legend()
    plt.grid()
plt.show()

def f3_b(x,var):
    y,u=var
    f1=u
    f2=np.exp(2*x)*np.sin(x) -2*y +2*u
    return np.array([f1,f2])

ic=[-0.4,-0.6]
z=RK2(f3_b,ic,0,1,5)
y=z[0].T

head=["x","y","y'"]
data_3b=np.column_stack((z[1],y[0],y[1]))
table(data_3b,head,"TABLE for output for q3(b)")

plt.plot(z[1],y[0],label="Y",marker=".")
plt.plot(z[1],y[1],label="Y'",marker=".")
plt.xlabel("x")
plt.title("Y and Y' for Q3(B)")
plt.legend()
plt.grid()
plt.show()
```

3 Results

3.1 Comparing the three methods for a specific N (N = 100)

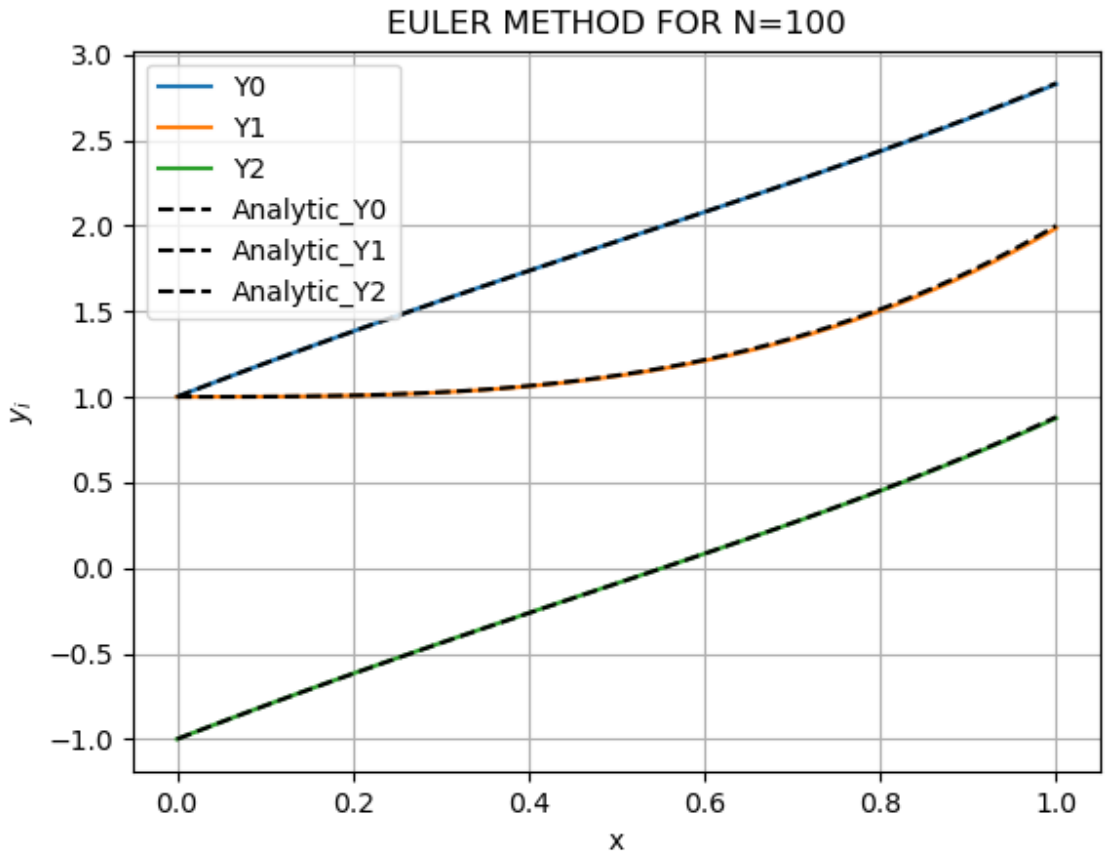

Figure 1: Euler Method for N = 100

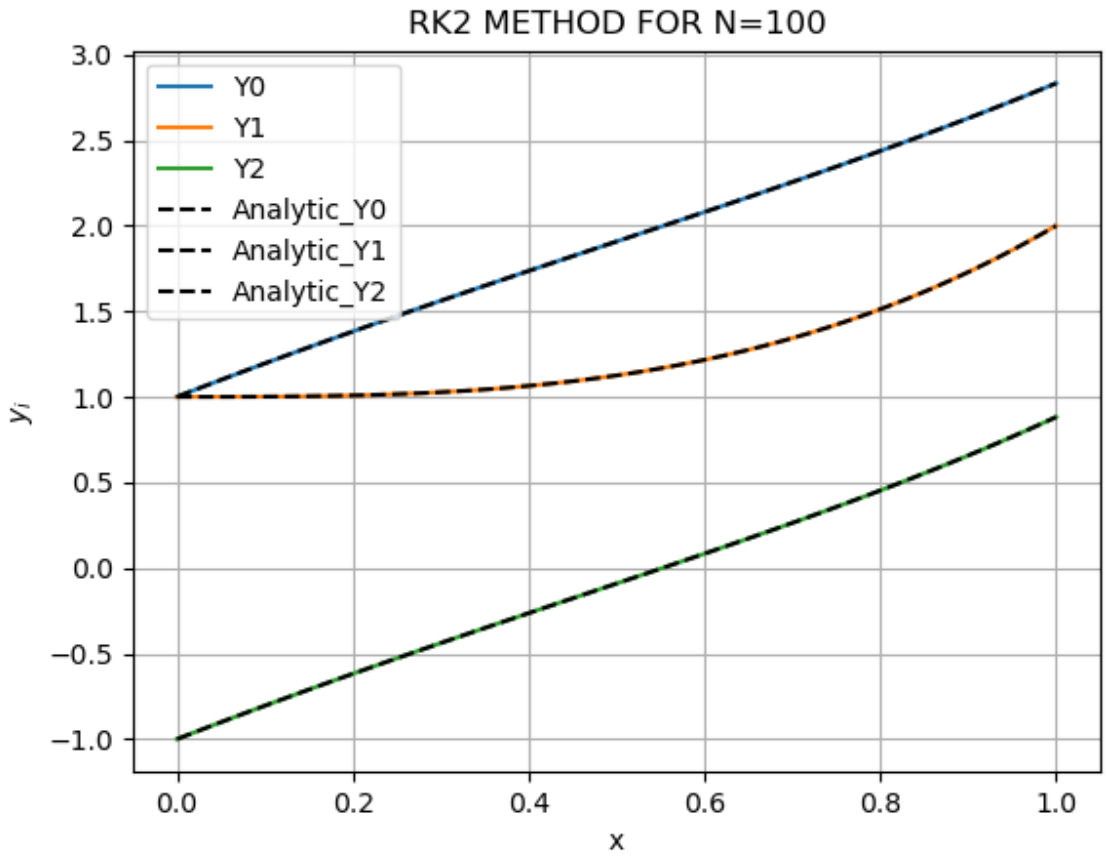

Figure 2: RK-2 Method for N = 100

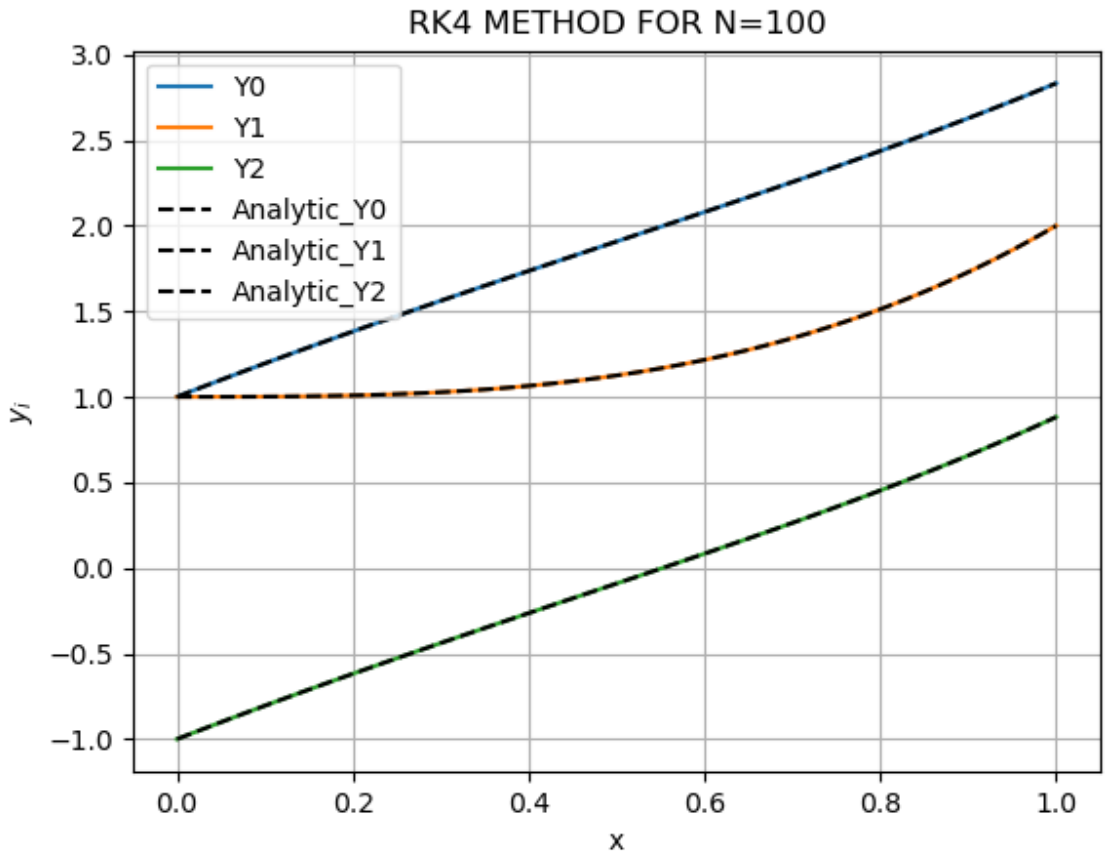

Figure 3: RK-4 Method for N = 100

3.2 Comparing the three methods for different N

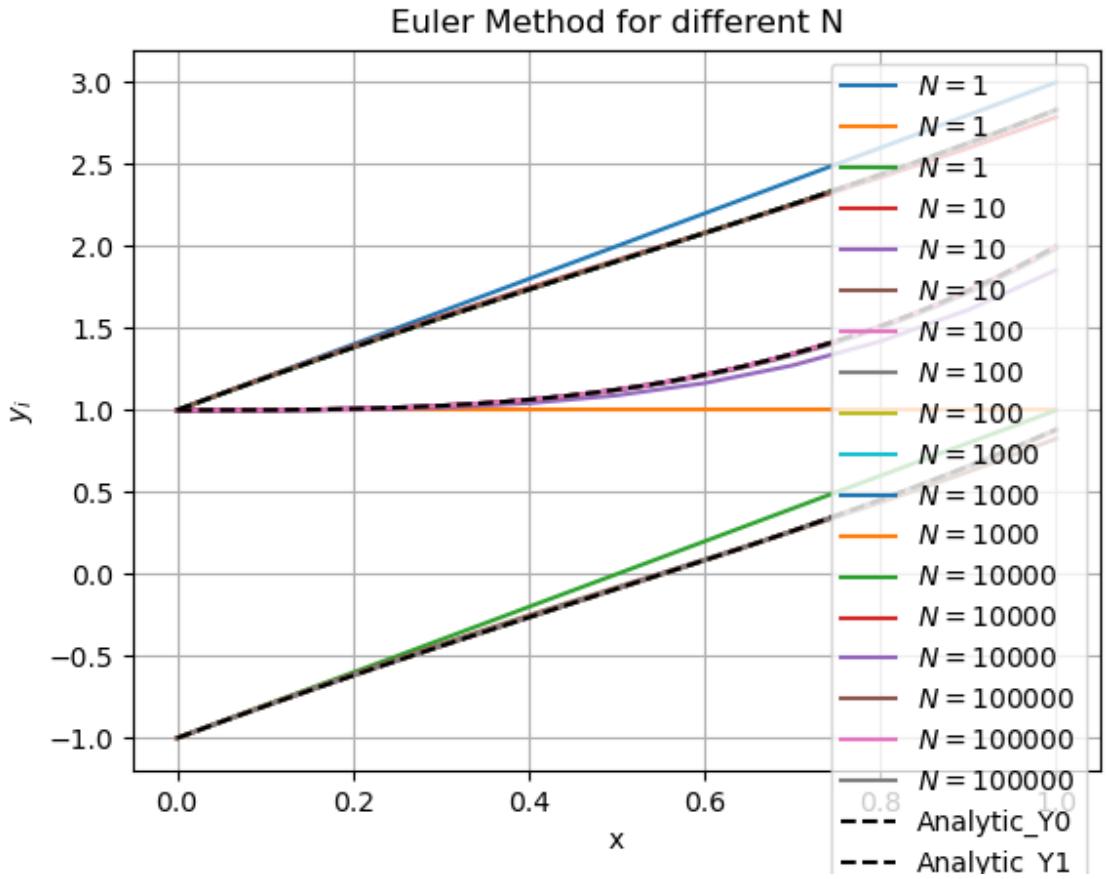

Figure 4: Euler Method for multiple N

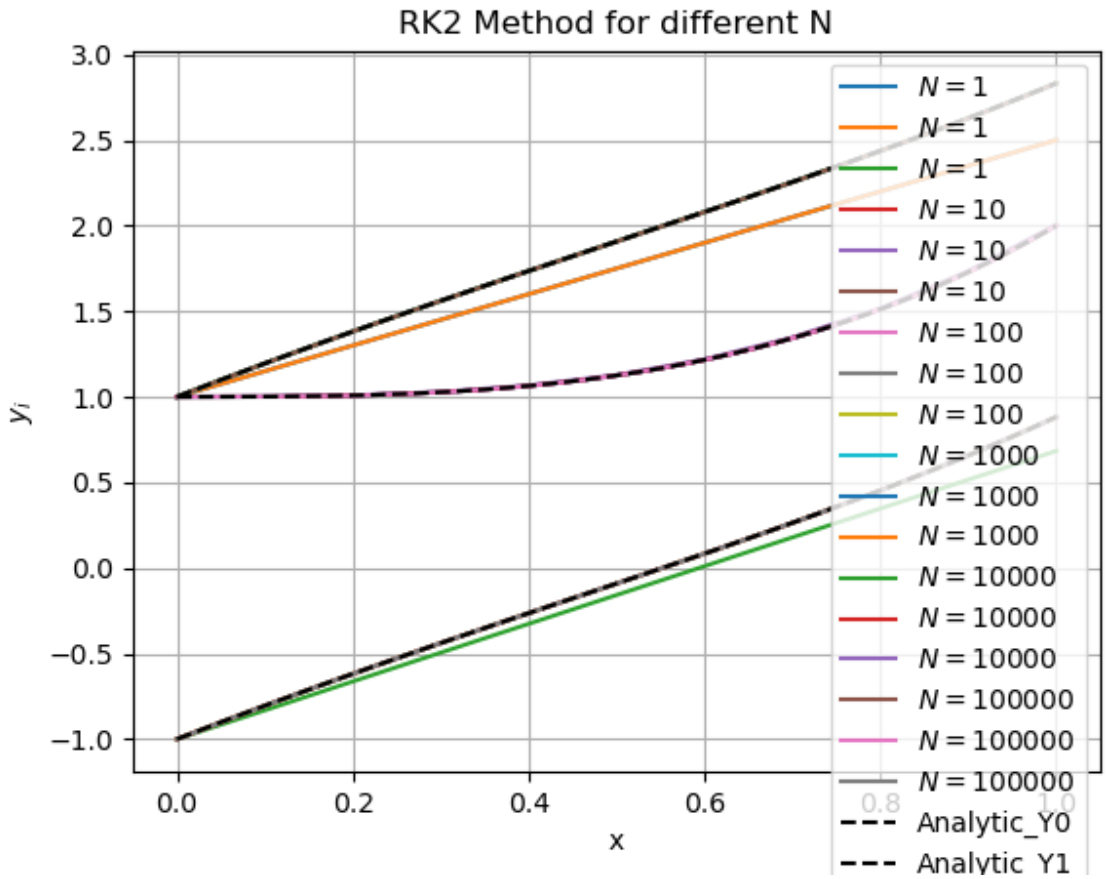

Figure 5: RK-2 Method for multiple N

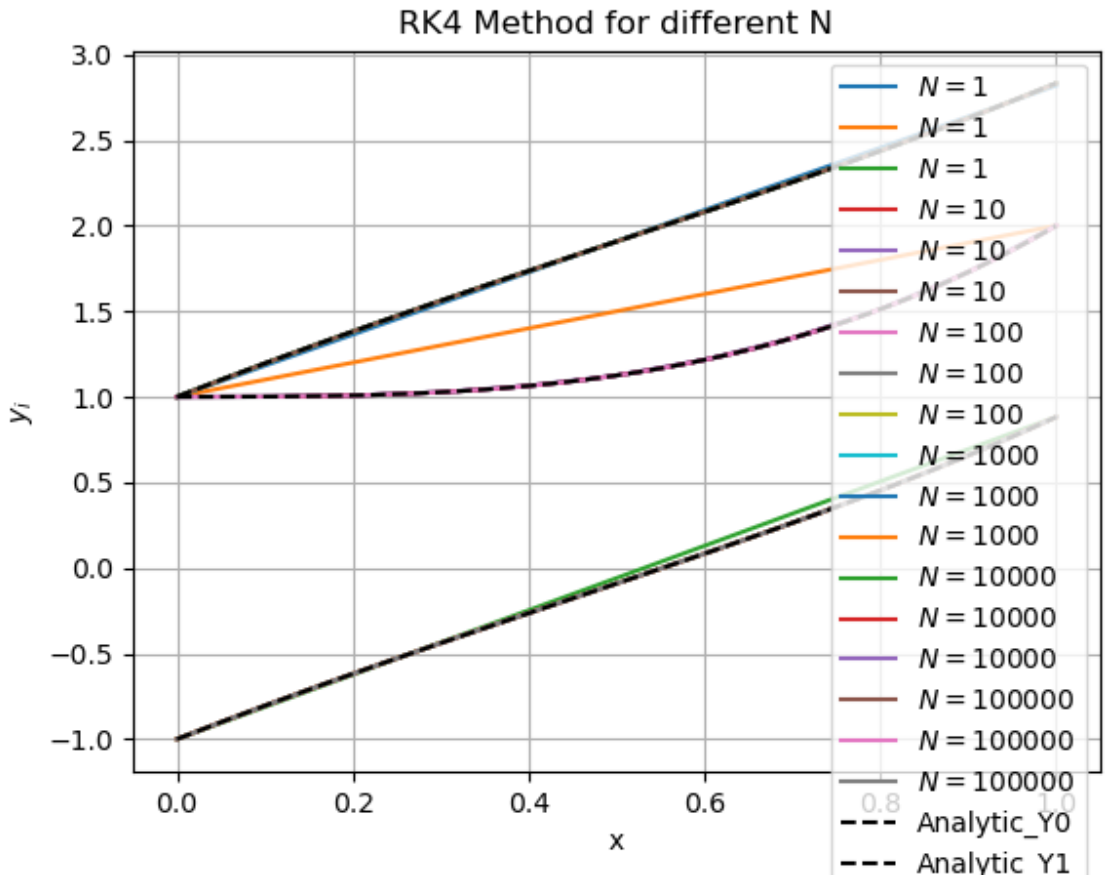

Figure 6: RK-4 Method for multiple N

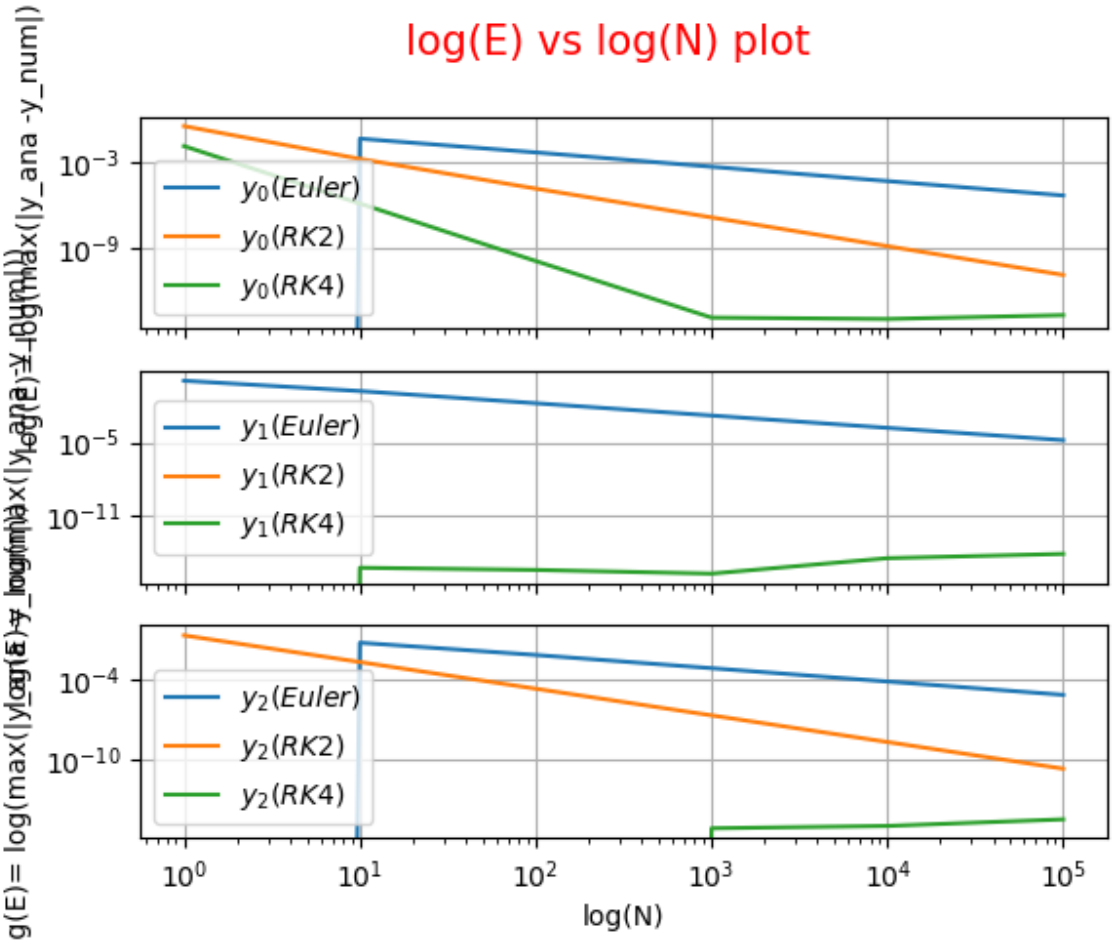

Figure 7: Log(E) vs log(N)

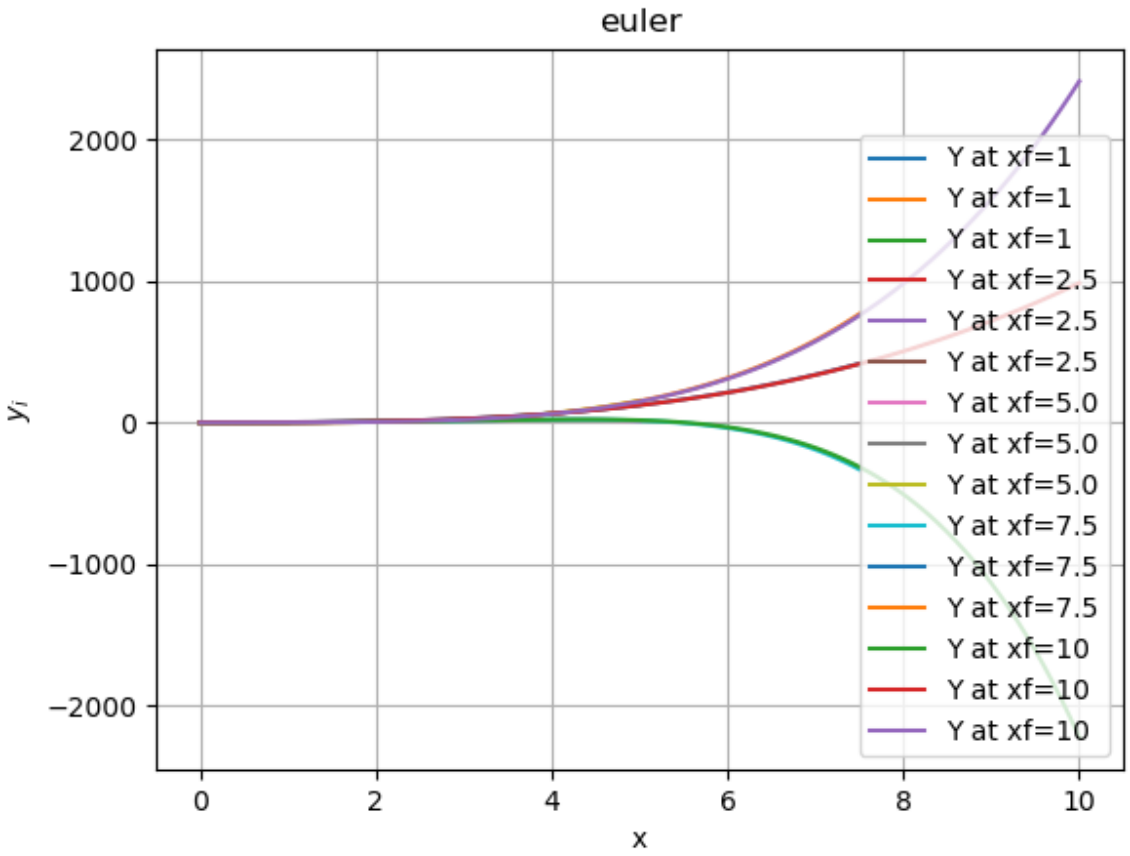

Figure 8: For Different xf : Euler

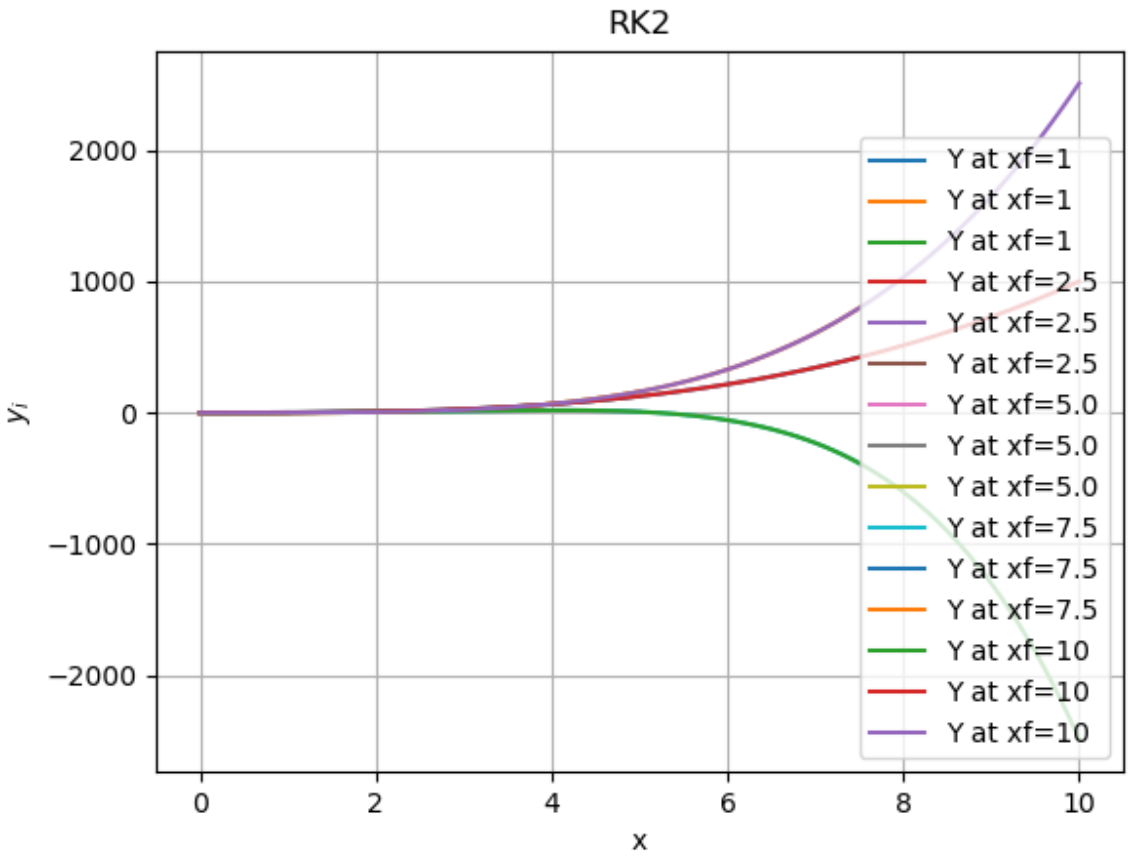

Figure 9: For Different xf : RK2

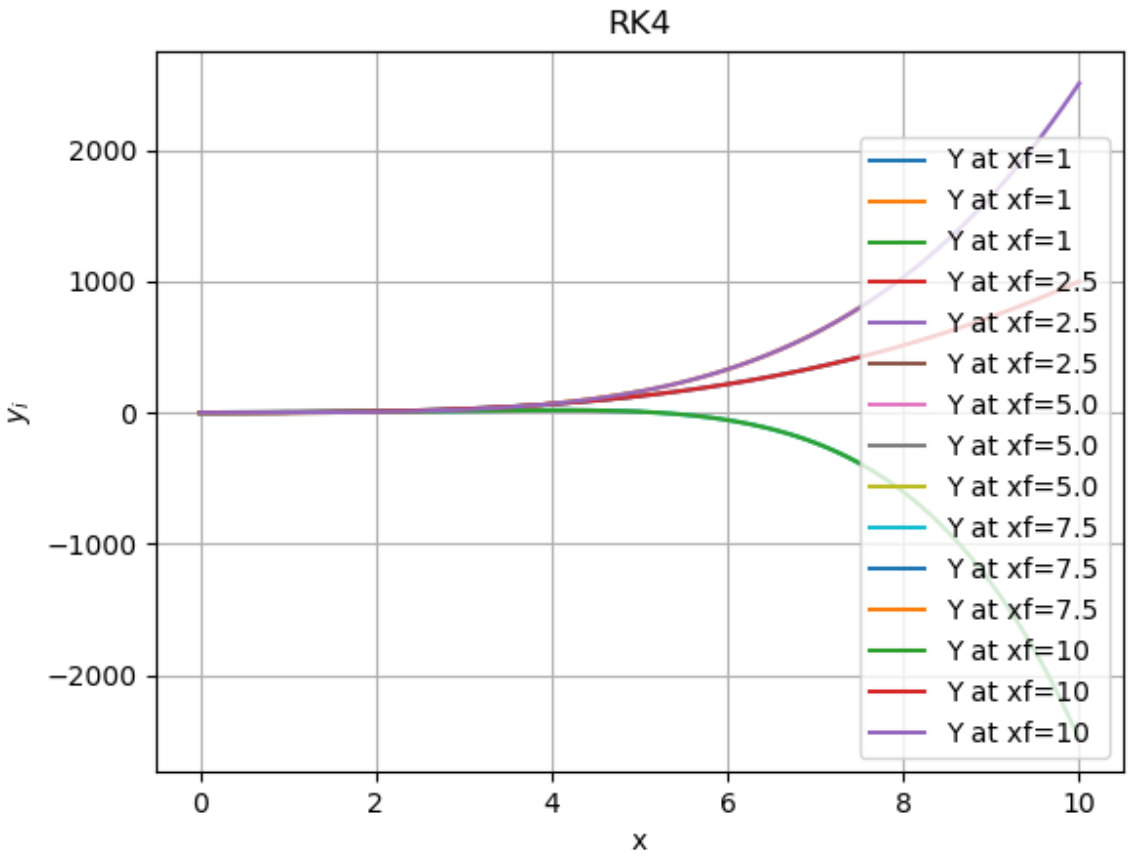

Figure 10: For Different xf : RK4

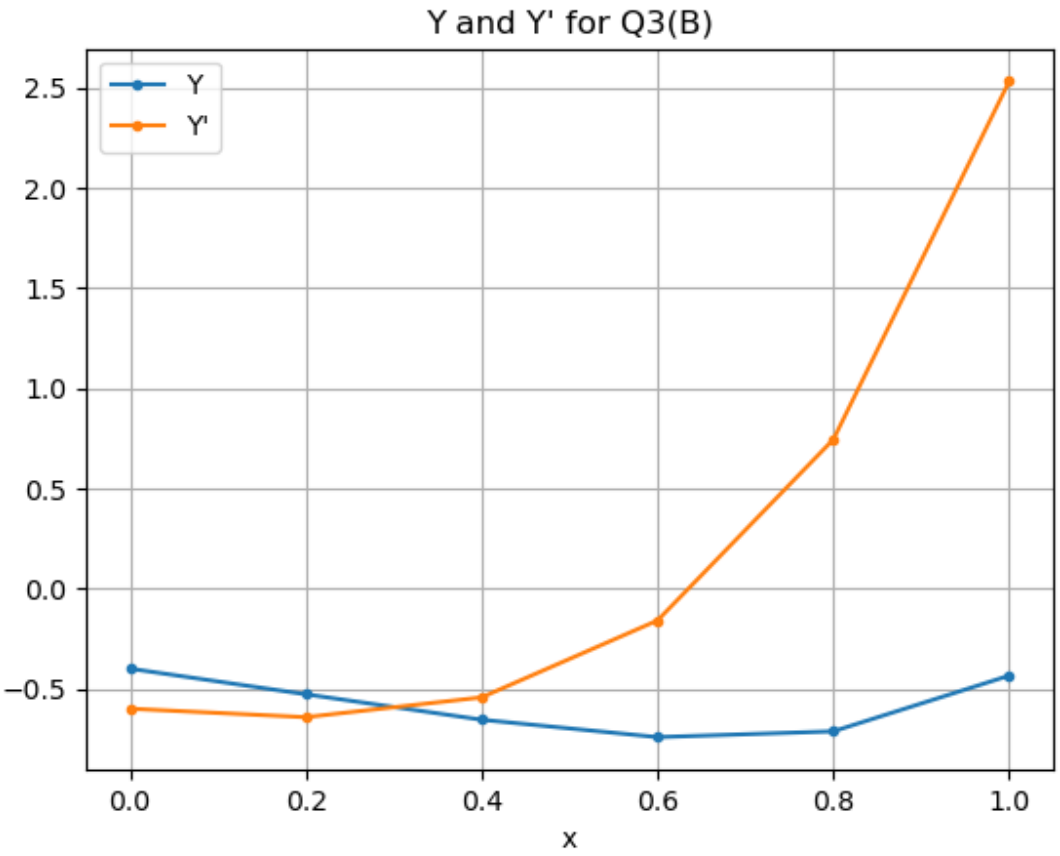

Figure 11: Q3(b)

3.3 Tabulated Data

EULER METHOD FOR N=100

x	Y1	Y2	Y3
0.0	1.0	1.0	-1.0
0.01	1.0	1.0	-0.98
0.02	1.0	1.0	-0.96
0.03	1.1	1.0	-0.94
0.04	1.1	1.0	-0.92
0.05	1.1	1.0	-0.9
0.06	1.1	1.0	-0.88
0.07	1.1	1.0	-0.86
0.08	1.2	1.0	-0.84
0.09	1.2	1.0	-0.82
0.1	1.2	1.0	-0.8
0.11	1.2	1.0	-0.79
0.12	1.2	1.0	-0.77
0.13	1.3	1.0	-0.75
0.14	1.3	1.0	-0.73

0.15	1.3	1.0	-0.71
0.16	1.3	1.0	-0.69
0.17	1.3	1.0	-0.67
0.18	1.3	1.0	-0.65
0.19	1.4	1.0	-0.64
0.2	1.4	1.0	-0.62
0.21	1.4	1.0	-0.6
0.22	1.4	1.0	-0.58
0.23	1.4	1.0	-0.56
0.24	1.5	1.0	-0.54
0.25	1.5	1.0	-0.53
0.26	1.5	1.0	-0.51
0.27	1.5	1.0	-0.49
0.28	1.5	1.0	-0.47
0.29	1.5	1.0	-0.46
0.3	1.6	1.0	-0.44
0.31	1.6	1.0	-0.42
0.32	1.6	1.0	-0.4

0.33	1.6	1.0	-0.38
0.34	1.6	1.0	-0.37
0.35	1.6	1.0	-0.35
0.36	1.7	1.0	-0.33
0.37	1.7	1.0	-0.31
0.38	1.7	1.1	-0.3
0.39	1.7	1.1	-0.28
0.4	1.7	1.1	-0.26
0.41	1.8	1.1	-0.25
0.42	1.8	1.1	-0.23
0.43	1.8	1.1	-0.21
0.44	1.8	1.1	-0.19
0.45	1.8	1.1	-0.18
0.46	1.8	1.1	-0.16
0.47	1.9	1.1	-0.14
0.48	1.9	1.1	-0.12
0.49	1.9	1.1	-0.11

0.5	1.9	1.1	-0.09
0.51	1.9	1.1	-0.073
0.52	1.9	1.1	-0.056
0.53	2.0	1.1	-0.038
0.54	2.0	1.2	-0.021
0.55	2.0	1.2	-0.0036
0.56	2.0	1.2	0.014
0.57	2.0	1.2	0.031
0.58	2.0	1.2	0.049
0.59	2.1	1.2	0.066
0.6	2.1	1.2	0.084
0.61	2.1	1.2	0.1
0.62	2.1	1.2	0.12
0.63	2.1	1.2	0.14
0.64	2.1	1.3	0.15
0.65	2.2	1.3	0.17
0.66	2.2	1.3	0.19
0.67	2.2	1.3	0.21

0.68	2.2	1.3	0.23
0.69	2.2	1.3	0.24
0.7	2.3	1.3	0.26
0.71	2.3	1.4	0.28
0.72	2.3	1.4	0.3
0.73	2.3	1.4	0.32
0.74	2.3	1.4	0.34
0.75	2.3	1.4	0.36
0.76	2.4	1.4	0.37
0.77	2.4	1.4	0.39
0.78	2.4	1.5	0.41
0.79	2.4	1.5	0.43
0.8	2.4	1.5	0.45
0.81	2.5	1.5	0.47
0.82	2.5	1.5	0.49
0.83	2.5	1.6	0.51
0.84	2.5	1.6	0.53

```
    0.85          2.5           1.6           0.55

    0.86          2.5           1.6           0.57

    0.87          2.6           1.6           0.59

    0.88          2.6           1.7           0.61

    0.89          2.6           1.7           0.63

    0.9           2.6           1.7           0.65

    0.91          2.6           1.7           0.67

    0.92          2.7           1.8           0.7

    0.93          2.7           1.8           0.72

    0.94          2.7           1.8           0.74

    0.95          2.7           1.8           0.76

    0.96          2.7           1.9           0.78

    0.97          2.8           1.9           0.81

    0.98          2.8           1.9           0.83

    0.99          2.8           2.0           0.85

    1.0           2.8           2.0           0.88


RK2 METHOD FOR N=100
     x             Y1            Y2            Y3
```

0.0	1.0	1.0	-1.0
0.01	1.0	1.0	-0.98
0.02	1.0	1.0	-0.96
0.03	1.1	1.0	-0.94
0.04	1.1	1.0	-0.92
0.05	1.1	1.0	-0.9
0.06	1.1	1.0	-0.88
0.07	1.1	1.0	-0.86
0.08	1.2	1.0	-0.84
0.09	1.2	1.0	-0.82
0.1	1.2	1.0	-0.8
0.11	1.2	1.0	-0.79
0.12	1.2	1.0	-0.77
0.13	1.3	1.0	-0.75
0.14	1.3	1.0	-0.73
0.15	1.3	1.0	-0.71
0.16	1.3	1.0	-0.69

0.17	1.3	1.0	-0.67
0.18	1.3	1.0	-0.66
0.19	1.4	1.0	-0.64
0.2	1.4	1.0	-0.62
0.21	1.4	1.0	-0.6
0.22	1.4	1.0	-0.58
0.23	1.4	1.0	-0.56
0.24	1.5	1.0	-0.55
0.25	1.5	1.0	-0.53
0.26	1.5	1.0	-0.51
0.27	1.5	1.0	-0.49
0.28	1.5	1.0	-0.47
0.29	1.5	1.0	-0.46
0.3	1.6	1.0	-0.44
0.31	1.6	1.0	-0.42
0.32	1.6	1.0	-0.4
0.33	1.6	1.0	-0.39
0.34	1.6	1.0	-0.37

0.35	1.6	1.0	-0.35
0.36	1.7	1.0	-0.33
0.37	1.7	1.1	-0.32
0.38	1.7	1.1	-0.3
0.39	1.7	1.1	-0.28
0.4	1.7	1.1	-0.26
0.41	1.8	1.1	-0.25
0.42	1.8	1.1	-0.23
0.43	1.8	1.1	-0.21
0.44	1.8	1.1	-0.19
0.45	1.8	1.1	-0.18
0.46	1.8	1.1	-0.16
0.47	1.9	1.1	-0.14
0.48	1.9	1.1	-0.13
0.49	1.9	1.1	-0.11
0.5	1.9	1.1	-0.091
0.51	1.9	1.1	-0.074

0.52	1.9	1.1	-0.056
0.53	2.0	1.1	-0.039
0.54	2.0	1.2	-0.021
0.55	2.0	1.2	-0.0041
0.56	2.0	1.2	0.013
0.57	2.0	1.2	0.031
0.58	2.0	1.2	0.048
0.59	2.1	1.2	0.066
0.6	2.1	1.2	0.084
0.61	2.1	1.2	0.1
0.62	2.1	1.2	0.12
0.63	2.1	1.3	0.14
0.64	2.1	1.3	0.15
0.65	2.2	1.3	0.17
0.66	2.2	1.3	0.19
0.67	2.2	1.3	0.21
0.68	2.2	1.3	0.23
0.69	2.2	1.3	0.25

0.7	2.3	1.3	0.26
0.71	2.3	1.4	0.28
0.72	2.3	1.4	0.3
0.73	2.3	1.4	0.32
0.74	2.3	1.4	0.34
0.75	2.3	1.4	0.36
0.76	2.4	1.4	0.38
0.77	2.4	1.5	0.39
0.78	2.4	1.5	0.41
0.79	2.4	1.5	0.43
0.8	2.4	1.5	0.45
0.81	2.5	1.5	0.47
0.82	2.5	1.6	0.49
0.83	2.5	1.6	0.51
0.84	2.5	1.6	0.53
0.85	2.5	1.6	0.55
0.86	2.6	1.6	0.57

```
    0.87          2.6           1.7           0.59

    0.88          2.6           1.7           0.62

    0.89          2.6           1.7           0.64

    0.9           2.6           1.7           0.66

    0.91          2.6           1.8           0.68

    0.92          2.7           1.8           0.7

    0.93          2.7           1.8           0.72

    0.94          2.7           1.8           0.74

    0.95          2.7           1.9           0.77

    0.96          2.7           1.9           0.79

    0.97          2.8           1.9           0.81

    0.98          2.8           1.9           0.84

    0.99          2.8           2.0           0.86

    1.0           2.8           2.0           0.88


RK4 METHOD FOR N=100
     x             Y1            Y2            Y3
 ---------------------------------------------------
    0.0           1.0           1.0           -1.0

    0.01          1.0           1.0          -0.98
```

0.02	1.0	1.0	-0.96
0.03	1.1	1.0	-0.94
0.04	1.1	1.0	-0.92
0.05	1.1	1.0	-0.9
0.06	1.1	1.0	-0.88
0.07	1.1	1.0	-0.86
0.08	1.2	1.0	-0.84
0.09	1.2	1.0	-0.82
0.1	1.2	1.0	-0.8
0.11	1.2	1.0	-0.79
0.12	1.2	1.0	-0.77
0.13	1.3	1.0	-0.75
0.14	1.3	1.0	-0.73
0.15	1.3	1.0	-0.71
0.16	1.3	1.0	-0.69
0.17	1.3	1.0	-0.67
0.18	1.3	1.0	-0.66

```
0.19        1.4         1.0         -0.64
0.2         1.4         1.0         -0.62
0.21        1.4         1.0          -0.6
0.22        1.4         1.0         -0.58
0.23        1.4         1.0         -0.56
0.24        1.5         1.0         -0.55
0.25        1.5         1.0         -0.53
0.26        1.5         1.0         -0.51
0.27        1.5         1.0         -0.49
0.28        1.5         1.0         -0.47
0.29        1.5         1.0         -0.46
0.3         1.6         1.0         -0.44
0.31        1.6         1.0         -0.42
0.32        1.6         1.0          -0.4
0.33        1.6         1.0         -0.39
0.34        1.6         1.0         -0.37
0.35        1.6         1.0         -0.35
0.36        1.7         1.0         -0.33
```

0.37	1.7	1.1	-0.32
0.38	1.7	1.1	-0.3
0.39	1.7	1.1	-0.28
0.4	1.7	1.1	-0.26
0.41	1.8	1.1	-0.25
0.42	1.8	1.1	-0.23
0.43	1.8	1.1	-0.21
0.44	1.8	1.1	-0.19
0.45	1.8	1.1	-0.18
0.46	1.8	1.1	-0.16
0.47	1.9	1.1	-0.14
0.48	1.9	1.1	-0.13
0.49	1.9	1.1	-0.11
0.5	1.9	1.1	-0.091
0.51	1.9	1.1	-0.074
0.52	1.9	1.1	-0.056
0.53	2.0	1.1	-0.039

0.54	2.0	1.2	-0.021
0.55	2.0	1.2	-0.0041
0.56	2.0	1.2	0.013
0.57	2.0	1.2	0.031
0.58	2.0	1.2	0.048
0.59	2.1	1.2	0.066
0.6	2.1	1.2	0.084
0.61	2.1	1.2	0.1
0.62	2.1	1.2	0.12
0.63	2.1	1.3	0.14
0.64	2.1	1.3	0.15
0.65	2.2	1.3	0.17
0.66	2.2	1.3	0.19
0.67	2.2	1.3	0.21
0.68	2.2	1.3	0.23
0.69	2.2	1.3	0.25
0.7	2.3	1.3	0.26
0.71	2.3	1.4	0.28

0.72	2.3	1.4	0.3
0.73	2.3	1.4	0.32
0.74	2.3	1.4	0.34
0.75	2.3	1.4	0.36
0.76	2.4	1.4	0.38
0.77	2.4	1.5	0.39
0.78	2.4	1.5	0.41
0.79	2.4	1.5	0.43
0.8	2.4	1.5	0.45
0.81	2.5	1.5	0.47
0.82	2.5	1.6	0.49
0.83	2.5	1.6	0.51
0.84	2.5	1.6	0.53
0.85	2.5	1.6	0.55
0.86	2.6	1.6	0.57
0.87	2.6	1.7	0.59
0.88	2.6	1.7	0.62

0.89	2.6	1.7	0.64
0.9	2.6	1.7	0.66
0.91	2.6	1.8	0.68
0.92	2.7	1.8	0.7
0.93	2.7	1.8	0.72
0.94	2.7	1.8	0.74
0.95	2.7	1.9	0.77
0.96	2.7	1.9	0.79
0.97	2.8	1.9	0.81
0.98	2.8	1.9	0.84
0.99	2.8	2.0	0.86
1.0	2.8	2.0	0.88

Table for N and E= max(|y_ana -y_num|) values for y0,y1 and y2 for all three methods

N	E_y1(Euler)	E_y1(RK2)	E_y1(RK4)	E_y2(Euler)	E_y2(RK2)	E_y2(RK4)
1.0	0.0	0.33	0.013	1.0	0.0	0.0
1e+01	0.044	0.0017	1.4e-06	0.14	0.0	6.7e-16
1e+02	0.0049	1.5e-05	1.4e-10	0.015	0.0	4.4e-16

1e+03	0.00049	1.5e-07	1.6e-14	0.0015	0.0	2.2e-16
1e+04	4.9e-05	1.5e-09	1.2e-14	0.00015	0.0	4e-15
1e+05	4.9e-06	1.5e-11	2.3e-14	1.5e-05	0.0	8.9e-15

TABLE for output for q3(b)

x	y	y’
0.0	-0.4	-0.6
0.2	-0.53	-0.64
0.4	-0.66	-0.54
0.6	-0.74	-0.16
0.8	-0.71	0.74
1.0	-0.43	2.5