

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	1
2	Programming	9
2.1	Module_A8_1097.py	9
2.2	tol_A8_1097.py	10
2.3	1097_Main_A8.py	13
3	Results	19
3.1	Comparing the three methods for a specific N (N = 100)	19
3.2	Comparing the three methods for different N	22
3.3	Tabulated Data	30

1 Theory

Theory

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

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

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

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

$$\text{let } x_1 = y \rightarrow x_1' = y' = x_2$$

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

$$\begin{matrix} \\ \\ \vdots \\ \vdots \\ \cdot \end{matrix}$$

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

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

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

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

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

First we'll convert it into 2 first order DE

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

so

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

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

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

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

Initial condition

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

$$f_{00} \quad 0 \leq x \leq 1$$

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

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

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

$$k_1 = h^* f(x_0, y_0, u_0)$$

$$= 0.2 \cdot f_1(0, -0.4, -0.6)$$

$$= 0.2 \cdot (-0.6) = -0.12$$

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

$$= 0.2 \cdot f_2(0, -0.4, -0.6)$$

$$= 0.2 \cdot [0.8 - 1.2]$$

$$= -0.08$$

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

$$= 0.2 \cdot -0.68$$

$$= -0.136$$

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

$$0.2 \cdot (-0.02362019)$$

$$= -0.00472404$$

DATE : ___ / ___ / ___
PAGE : ___

$$y_0 \quad y_{i+1} = y_i + \frac{k_1 + k_2}{2}$$

$f_0 \quad f_1 \quad i=0$

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

$$= -0.4 \left(\frac{-0.12 + (-0.13)}{2} \right)$$

$$= -0.528$$

$f_0 \quad f_2$

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

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

$$= 0.6423620$$

~~We started with initial condition.~~
Now we have to find the next term with
 $y_1 = -0.528 \quad u_1 = -0.6423620$

$$x \rightarrow x+h \Rightarrow \alpha^2$$

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

$$\begin{aligned} h^* &= (-0.64236202) \\ &= -0.1284724 \end{aligned}$$

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

$$\begin{aligned} &= h^* (0.06765578) \\ &= +0.002353116 \end{aligned}$$

$$\begin{aligned} k_2 &= h^* f_1(x_0 + 2h, y_1 + k_1, u_1 + k_1) \\ &= -0.125576617 \end{aligned}$$

$$\begin{aligned} k_2 &= h^* f_2(x_0 + 2h, y_1 + k_1, u_1 + k_1) \\ &= h^* (0.92194454) \\ &= 0.18438991 \end{aligned}$$

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

$$y_2 = -0.65511929 \quad u_2 = -0.54340149$$

Step 3

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

$$= h^* (-0.54340149) = -0.1086009$$

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

$$= h^* (1.09010206) = 0.21802041$$

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

$$= h^* (0.92538167) = -0.06567621$$

DATE: / /
PAGE: _____

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

$$= h^* (2.75151605) \\ = 0.55030921$$

Next term

$$y_3 = y_2 + \left(\frac{k_1 + k_2}{2} \right)$$

$$y_3 = -0.74194951$$

$$u_3 = u_2 + \left(\frac{k_1 + k_2}{2} \right)$$

$$u_3 = -0.15923960$$

Step-4

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

$$= -0.07184794$$

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

$$= 0.60807895$$

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

$$= h^* (0.44874920) = 0.08935488$$

$$k_2 = h^* f_2(x_0+4h, y_3+k_1, u_3+k_1)$$

$$= h^* (5.9987744) = 1.1996255$$

Next term

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

$$y_4 = -0.71304150$$

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

$$u_4 = 0.74461755$$

DATE : / /
PAGE :

Step-5

$$k_1 = h^* f_1(x_0+uh, y_4+y_u) \\ = h^* (0.79961744) = 0.140892381$$

$$k_1 = h^* f_2(x_0+4h, y_4, u_4) \\ h^* (6.46840675) = 7.29360123$$

$$k_2 = h^* f_1(x_0+5h, y_4+k_1, u_4+k_1) \\ = 0.40765971$$

$$k_2 = h^* f_2(x_0+5h, y_4+k_1, u_4+k_1) \\ = 2.78450201$$

So

$$y_5 = -0.43474995 \quad u_5 = 25.3370918$$

2 Programming

2.1 Module_A8_1097.py

```
1 import numpy as np
2
3 def euler(Func,IC,a,b,N):
4     h=((b-a)/N)
5     t=np.linspace(a,b,N+1)
6     X=np.zeros([N+1,len(IC)])
7     X[0]=IC
8     for i in range(N):
9         X[i+1] = X[i] + h*Func(t[i],X[i])
10    return X,t
11
12 def RK2(Func,IC,a,b,N):
13     h=((b-a)/N)
14     t = np.linspace(a,b,N+1)
15     X = np.zeros([N+1, len(IC)])
16     X[0] = IC
17     for i in range(N):
18         k1 =h* Func(t[i],X[i])
19         k2 = h*Func( t[i] + h,X[i] + k1)
20         X[i+1] = X[i] + (k1 +k2 )/2
21     return X,t
22
23 def RK2_M(Func,IC,a,b,N):
24     h=((b-a)/N)
25     t = np.linspace(a,b,N+1)
26     X = np.zeros([N+1, len(IC)])
27     X[0] = IC
28     for i in range(N):
29         k1 =h* Func(t[i],X[i])
30         k2 = h*Func( t[i] + h/2,X[i] + k1/2)
31         X[i+1] = X[i] + (k1 +k2 )/2
32     return X,t
33
34 def RK4(Func,IC,a,b,N):
35     h=((b-a)/N)
36     t = np.linspace(a,b,N+1)
37     X = np.zeros([N+1, len(IC)])
38     X[0] = IC
```

```

39     for i in range(N):
40         k1 = h* Func(t[i],X[i])
41         k2 = h*Func(t[i] + h/2,X[i] + k1/2)
42         k3 = h*Func(t[i]+h/2,X[i] + k2/2)
43         k4 = h*Func(t[i]+h,X[i]+k3)
44         X[i+1] = X[i] + (k1+2*k2+2*k3+k4)/6
45     return X,t

```

2.2 tol_A8_1097.py

```

1 from MyIVP import euler
2 from MyIVP import RK2
3 from MyIVP import RK4
4 import numpy as np
5
6
7 def euler_tol(Func , IC ,a,b,N,N_max,tol):
8     w=0
9     Val=[]
10    N_arr=[]
11    count=0
12    while N<=N_max:
13        g=euler(Func , IC ,a,b,N)[0].T
14        t=euler(Func , IC ,a,b,N)[1]
15        Val.append(g)
16        N_arr.append(N)
17        if count >=1:
18            J=[]
19            K=[]
20            for i in range(len(IC)):
21                J.append(Val[-1][i][-1])
22                K.append(Val[-2][i][-1])
23            J=np.array(J)
24            K=np.array(K)
25            ff=[]
26            for g1,g2 in zip(J,K):
27                if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
28                    err=abs(g1-g2)
29                else:
30                    err=abs((g2-g1)/g1)
31            ff.append(err)
32            if max(ff)<=tol:

```

```

33         w=1
34             break
35     else:
36         pass
37
38     N=2*N
39     count+=1
40 if w==0:
41     s=("N_max reached without achieving required tolerance")
42 elif w==1:
43     s="Given tolerance achieved with",N_arr[-1],"sub-intervals"
44
45 return Val,Val[-1],N_arr[-1],N_arr,s,g,t
46
47 def RK2_tol(Func , IC ,a,b,N,N_max,tol):
48     w=0
49     Val=[]
50     N_arr=[]
51     count=0
52     while N<=N_max:
53         g=RK2(Func , IC ,a,b,N)[0].T
54         t=RK2(Func , IC ,a,b,N)[1]
55         Val.append(g)
56         N_arr.append(N)
57         if count>=1:
58             J=[]
59             K=[]
60             for i in range(len(IC)):
61                 J.append(Val[-1][i][-1])
62                 K.append(Val[-2][i][-1])
63             J=np.array(J)
64             K=np.array(K)
65             ff=[]
66             for g1,g2 in zip(J,K):
67                 if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
68                     err=abs(g1-g2)
69                 else:
70                     err=abs((g2-g1)/g1)
71                 ff.append(err)
72             if max(ff)<=tol:
73                 w=1
74                 break

```

```

75         else:
76             pass
77
78     N=2*N
79     count+=1
80     if w==0:
81         s=("N_max reached without achieving required tolerance")
82     elif w==1:
83         s="Given tolerance achieved with",N_arr[-1],"sub-intervals"
84
85     return Val,Val[-1],N_arr[-1],N_arr,s,g,t
86
87 def RK4_tol(Func , IC ,a,b,N,N_max,tol):
88     w=0
89     Val=[]
90     N_arr=[]
91     count=0
92     while N<=N_max:
93         g=RK4(Func , IC ,a,b,N)[0].T
94         t=RK4(Func , IC ,a,b,N)[1]
95         Val.append(g)
96         N_arr.append(N)
97         if count>=1:
98             J=[]
99             K=[]
100            for i in range(len(IC)):
101                J.append(Val[-1][i][-1])
102                K.append(Val[-2][i][-1])
103            J=np.array(J)
104            K=np.array(K)
105            ff=[]
106            for g1,g2 in zip(J,K):
107                if abs(g1)<=0.1e-5 or abs(g2)<=0.1e-5:
108                    err=abs(g1-g2)
109                else:
110                    err=abs((g2-g1)/g1)
111                ff.append(err)
112            if max(ff)<=tol:
113                w=1
114                break
115            else:
116                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)

```

```

31     for row in data:
32         for val in row:
33             print("{0:^12.2}".format(val), end=" ")
34         print("\n")
35
36 IC=[1,1,-1]
37 a=0
38 b=1
39 N=100
40
41 Ana_Sol_1= lambda x : -0.05*x**5 + 0.25*x**4 + x +2 - np.exp(-x)
42 Ana_Sol_2= lambda x : x**3 +1
43 Ana_Sol_3= lambda x : 0.25*x**4 + x - np.exp(-x)
44
45 X,Y,Z,t=output(Func,IC,a,b,N)
46
47 EUL_data=np.column_stack((t,X[0],X[1],X[2]))
48 RK2_data=np.column_stack((t,Y[0],Y[1],Y[2]))
49 RK4_data=np.column_stack((t,Z[0],Z[1],Z[2]))
50 heading=["x","Y1","Y2","Y3"]
51
52 table(EUL_data,heading,"\\nEULER METHOD FOR N=100")
53 table(RK2_data,heading,"\\nRK2 METHOD FOR N=100")
54 table(RK4_data,heading,"\\nRK4 METHOD FOR N=100")
55
56 def graph(t,X,title):
57     for i in range(len(X)):
58         plt.plot(t,X[i],label="Y"+str(i))
59     plt.title(title)
60     plt.xlabel("x")
61     plt.ylabel("$y_i$")
62     plt.legend()
63     plt.grid()
64
65 def graph_ana(t,Analytic):
66     for g,i in zip(Analytic,range(len(Analytic))):
67         plt.plot(t,g(t),label="Analytic_Y"+str(i),ls="--",c="black")
68     plt.legend()
69
70 Analytic=[Ana_Sol_1,Ana_Sol_2,Ana_Sol_3]
71
72 graph(t,X,"\\nEULER METHOD FOR N=100")

```

```

73 graph_ana(t,Analytic)
74 plt.show()
75
76 graph(t,Y,"\\nRK2 METHOD FOR N=100")
77 graph_ana(t,Analytic)
78 plt.show()
79
80 graph(t,Z,"\\nRK4 METHOD FOR N=100")
81 graph_ana(t,Analytic)
82 plt.show()
83
84 N_arr=[]
85 E_err_1=[];E_err_2=[];E_err_3=[]
86 RK2_err_1=[];RK2_err_2=[];RK2_err_3=[]
87 RK4_err_1=[];RK4_err_2=[];RK4_err_3=[]
88
89 def plot3(X,t,N):
90     for i in range(len(X)):
91         plt.plot(t,X[i],label="$N={0}$".format(N))
92
93 for i in range(0,6,1):
94     N=10**i
95     N_arr.append(N)
96     X,Y,Z,t=output(Func,IC,a,b,N)
97     plot3(X,t,N)
98     f1=Ana_Sol_1(t)
99     f2=Ana_Sol_2(t)
100    f3=Ana_Sol_3(t)
101    E_err_1.append(max(np.array(f1)-np.array(X[0])))
102    E_err_2.append(max(np.array(f2)-np.array(X[1])))
103    E_err_3.append(max(np.array(f3)-np.array(X[2])))
104
105 graph_ana(t,Analytic)
106 plt.title("Euler Method for different N")
107 plt.xlabel("x")
108 plt.ylabel("$y_i$")
109 plt.legend()
110 plt.grid()
111 plt.show()
112
113 for i in range(0,6,1):
114     N=10**i

```

```

115 X,Y,Z,t=output(Func,IC,a,b,N)
116 plot3(Y,t,N)
117 f1=Ana_Sol_1(t)
118 f2=Ana_Sol_2(t)
119 f3=Ana_Sol_3(t)
120 RK2_err_1.append(max(np.array(f1)-np.array(Y[0])))
121 RK2_err_2.append(max(np.array(f2)-np.array(Y[1])))
122 RK2_err_3.append(max(np.array(f3)-np.array(Y[2])))
123
124 graph_ana(t,Analytic)
125 plt.title("RK2 Method for different N")
126 plt.xlabel("x")
127 plt.ylabel("$y_i$")
128 plt.legend()
129 plt.grid()
130 plt.show()
131
132 for i in range(0,6,1):
133     N=10**(i)
134     X,Y,Z,t=output(Func,IC,a,b,N)
135     plot3(Z,t,N)
136     f1=Ana_Sol_1(t)
137     f2=Ana_Sol_2(t)
138     f3=Ana_Sol_3(t)
139     RK4_err_1.append(max(np.array(f1)-np.array(Z[0])))
140     RK4_err_2.append(max(np.array(f2)-np.array(Z[1])))
141     RK4_err_3.append(max(np.array(f3)-np.array(Z[2])))
142
143 graph_ana(t,Analytic)
144 plt.title("RK4 Method for different N")
145 plt.xlabel("x")
146 plt.ylabel("$y_i$")
147 plt.legend()
148 plt.grid()
149 plt.show()
150
151 print("\n")
152 data=np.column_stack((N_arr,E_err_1,RK2_err_1,RK4_err_1,E_err_2,RK2_err_2,
153 RK4_err_2,E_err_3,RK2_err_3,RK4_err_3))
154 head=["N","E_y1(Euler)","E_y1(RK2)","E_y1(RK4)","E_y2(Euler)","E_y2(RK2)",
155 "E_y2(RK4)","E_y3(Euler)","E_y3(RK2)","E_y3(RK4)"]
156 table(data,head,"\\nTable for N and E= max(|y_ana - y_num|) values for y0,y1"

```

```

        and y2 for all three methods  ")

155
156 fig, (ax1,ax2,ax3) = plt.subplots(3,sharex=True)
157 fig.suptitle("log(E) vs log(N) plot",fontsize=15,c="r")
158 ax1.plot(N_arr,E_err_1,label="$y_{\{0\}}(\text{Euler})$")
159 ax1.plot(N_arr,RK2_err_1,label="$y_{\{0\}}(\text{RK2})$")
160 ax1.plot(N_arr,RK4_err_1,label="$y_{\{0\}}(\text{RK4})$")
161 ax1.set(ylabel="log(E)= log(max(|y_ana -y_num|))",yscale="log",xscale="log"
162     ")
163 ax1.grid()
164 ax1.legend()

165 ax2.plot(N_arr,E_err_2,label="$y_{\{1\}}(\text{Euler})$")
166 ax2.plot(N_arr,RK2_err_2,label="$y_{\{1\}}(\text{RK2})$")
167 ax2.plot(N_arr,RK4_err_2,label="$y_{\{1\}}(\text{RK4})$")
168 ax2.set(ylabel="log(E)= log(max(|y_ana -y_num|))",xscale="log",yscale="log"
169     ")
170 ax2.grid()
171 ax2.legend()

172 ax3.plot(N_arr,E_err_3,label="$y_{\{2\}}(\text{Euler})$")
173 ax3.plot(N_arr,RK2_err_3,label="$y_{\{2\}}(\text{RK2})$")
174 ax3.plot(N_arr,RK4_err_3,label="$y_{\{2\}}(\text{RK4})$")
175 ax3.set(xlabel="log(N)",ylabel="log(E)= log(max(|y_ana -y_num|))",xscale="
176     log",yscale="log")
177 ax3.grid()
178 ax3.legend()
179 plt.show()

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

191 for i in x_f:

```

```

193     l1,l2,l3,t=output(Func,IC,0,i,100)
194     for k in range(len(l1)):
195         plt.plot(t,l2[k],label="Y at xf="+str(i))
196     plt.title("RK2")
197     plt.xlabel("x")
198     plt.ylabel("$y_i$")
199     plt.legend()
200     plt.grid()
201 plt.show()
202
203 for i in x_f:
204     l1,l2,l3,t=output(Func,IC,0,i,100)
205     for k in range(len(l1)):
206         plt.plot(t,l3[k],label="Y at xf="+str(i))
207     plt.title("RK4")
208     plt.xlabel("x")
209     plt.ylabel("$y_i$")
210     plt.legend()
211     plt.grid()
212 plt.show()
213
214 def f3_b(x,var):
215     y,u=var
216     f1=u
217     f2=np.exp(2*x)*np.sin(x) -2*y +2*u
218     return np.array([f1,f2])
219
220 ic=[-0.4,-0.6]
221 z=RK2(f3_b,ic,0,1,5)
222 y=z[0].T
223
224 head=["x","y","y'"]
225 data_3b=np.column_stack((z[1],y[0],y[1]))
226 table(data_3b,head,"TABLE for output for q3(b)")
227
228 plt.plot(z[1],y[0],label="Y",marker=".")
229 plt.plot(z[1],y[1],label="Y'",marker=".")
230 plt.xlabel("x")
231 plt.title("Y and Y' for Q3(B)")
232 plt.legend()
233 plt.grid()
234 plt.show()

```

3 Results

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

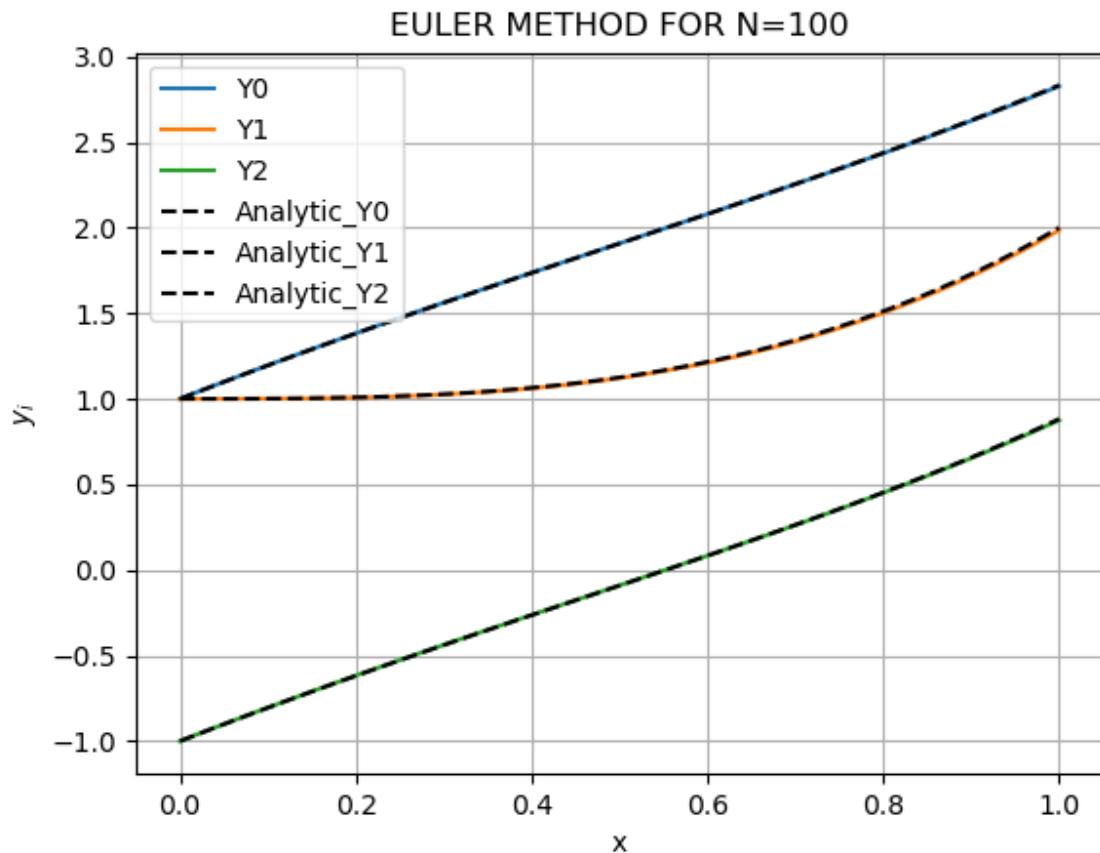


Figure 1: Euler Method for $N = 100$

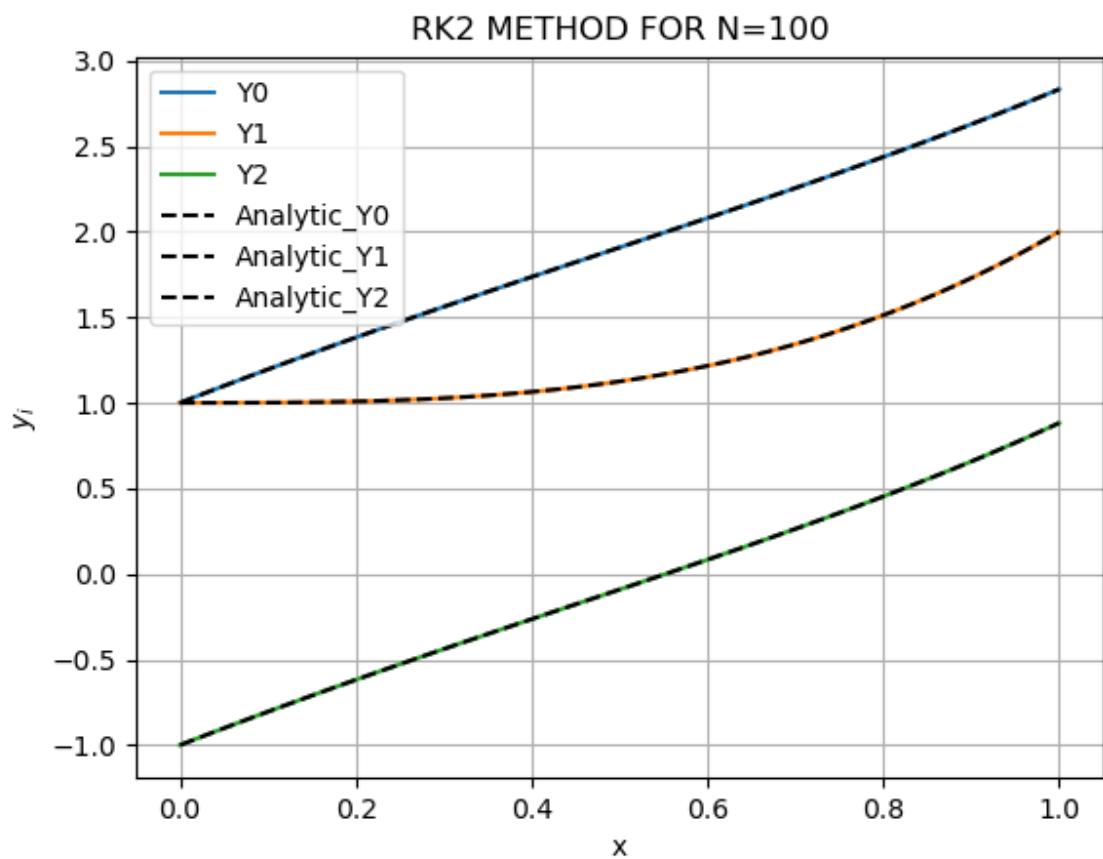


Figure 2: RK-2 Method for $N = 100$

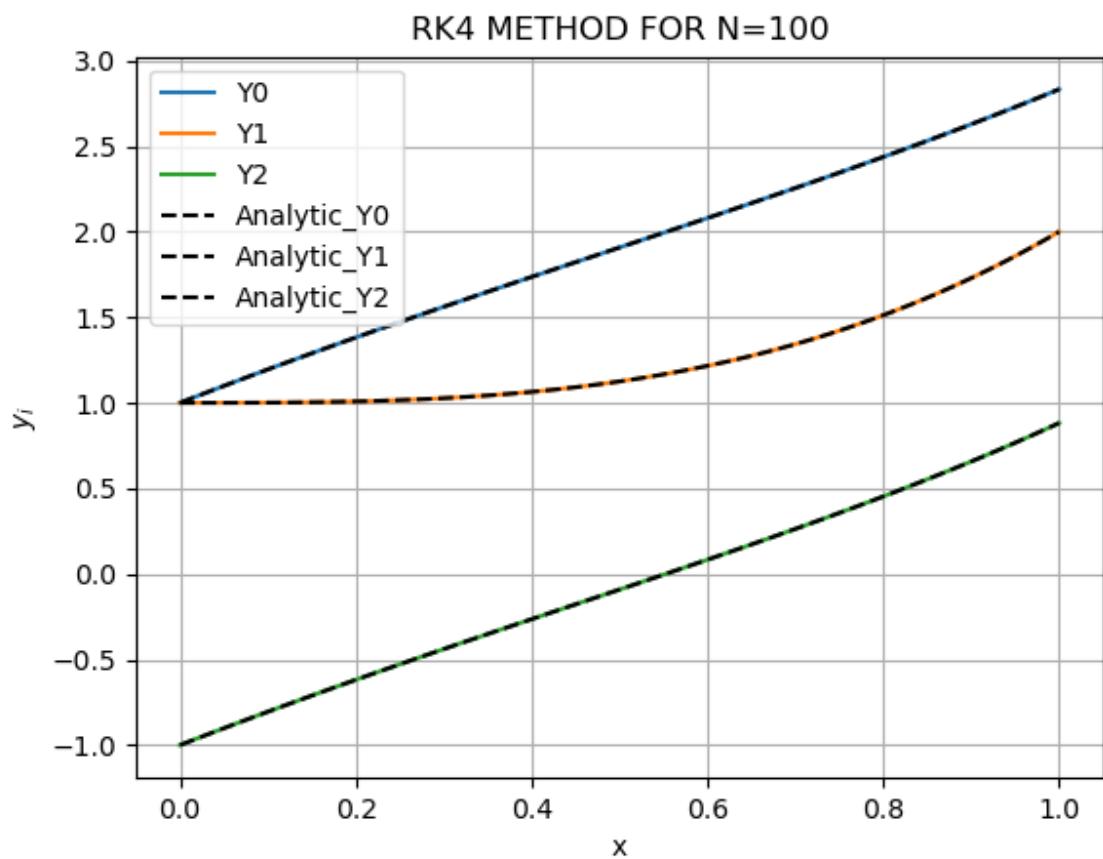


Figure 3: RK-4 Method for $N = 100$

3.2 Comparing the three methods for different N

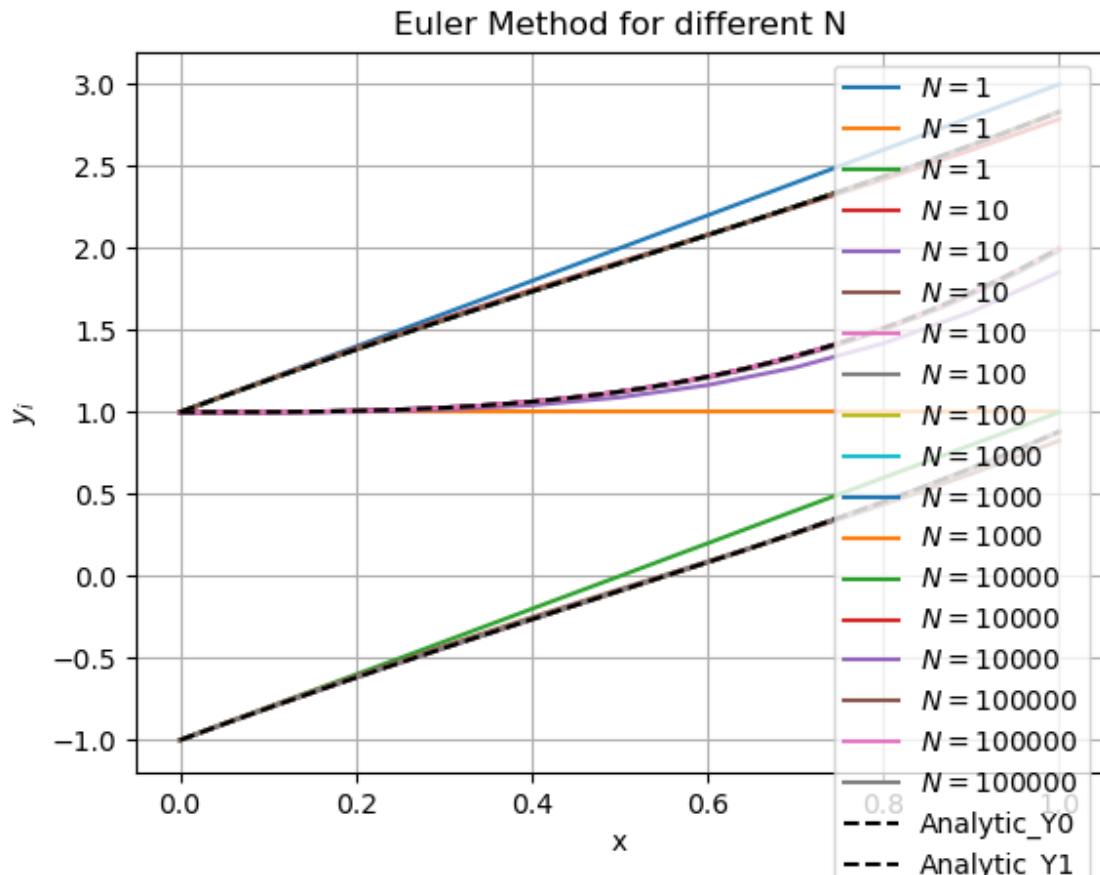


Figure 4: Euler Method for multiple N

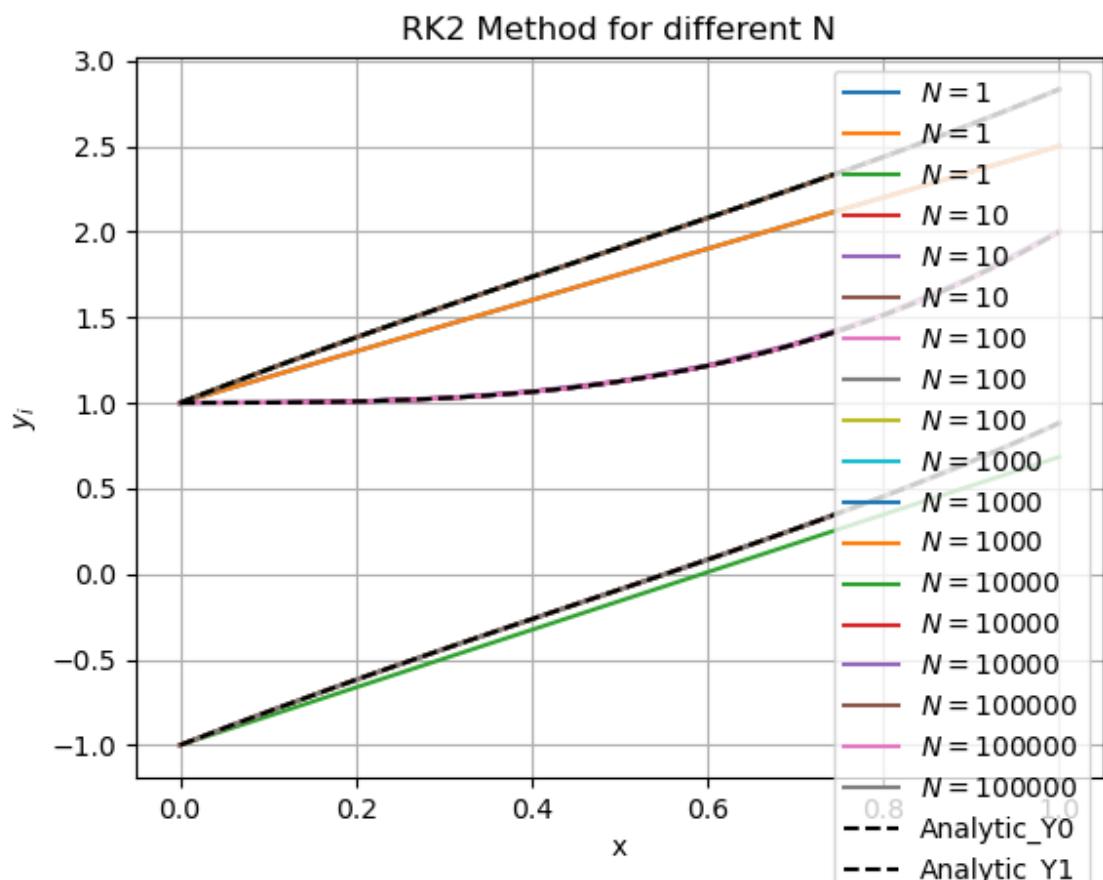


Figure 5: RK-2 Method for multiple N

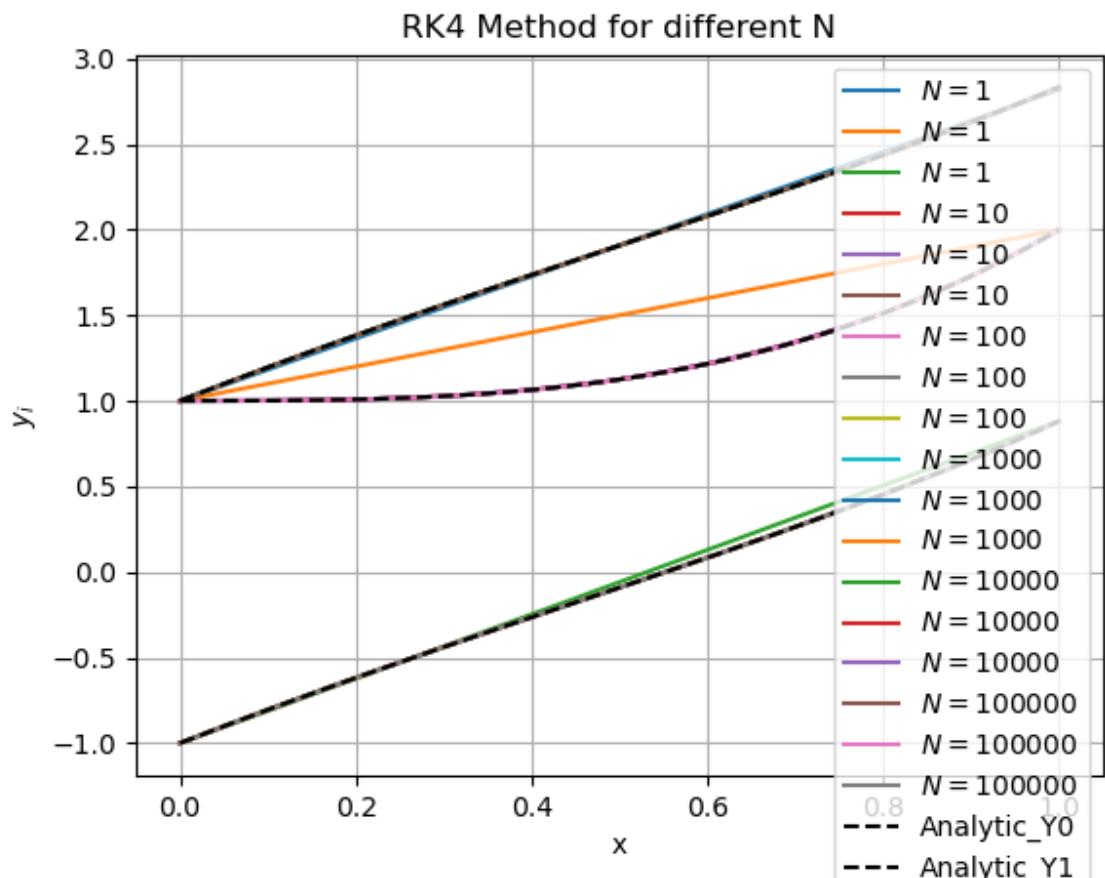


Figure 6: RK-4 Method for multiple N

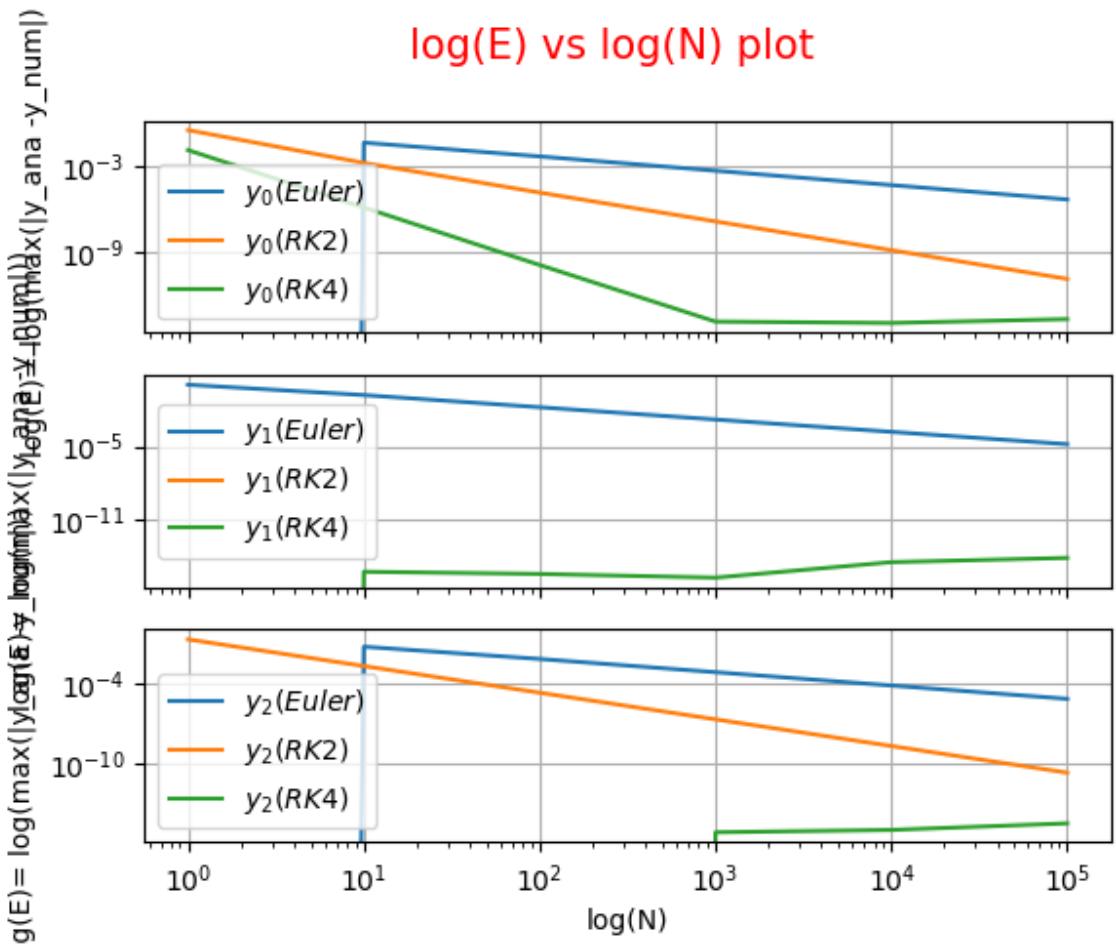


Figure 7: Log(E) vs log(N)

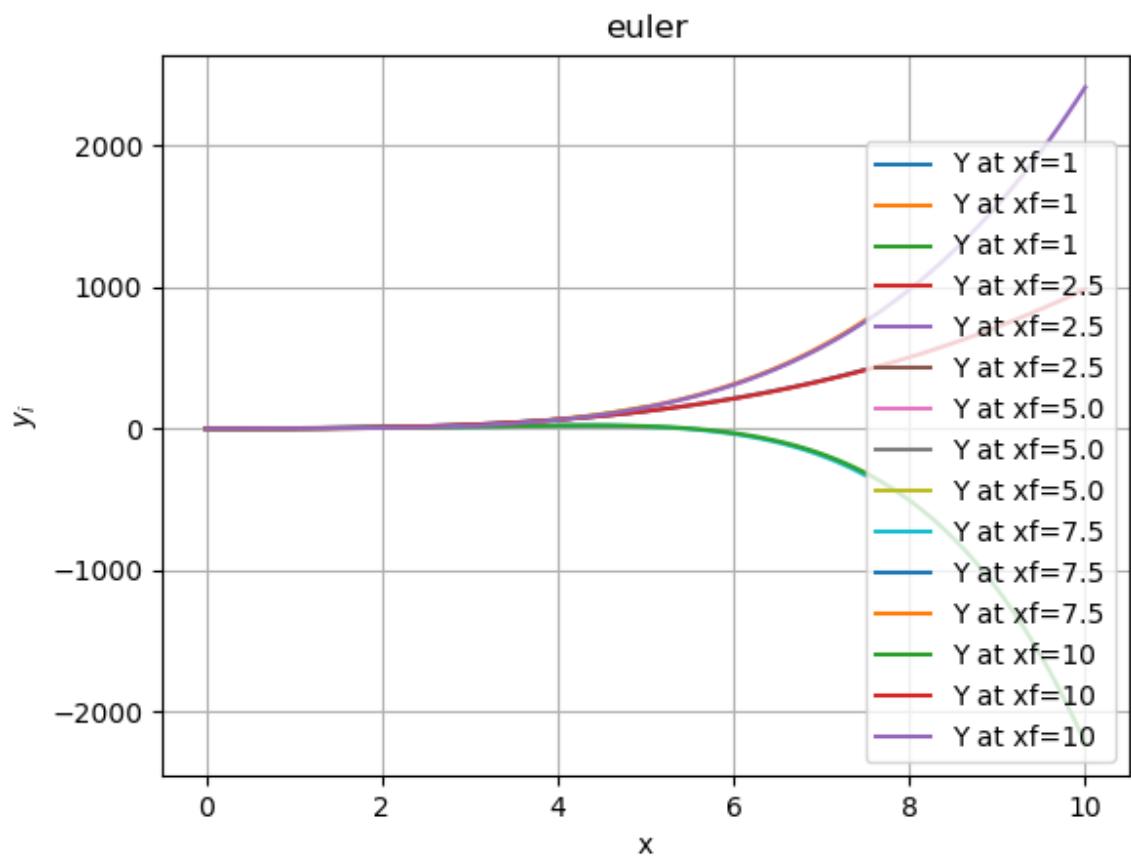


Figure 8: For Different xf : Euler

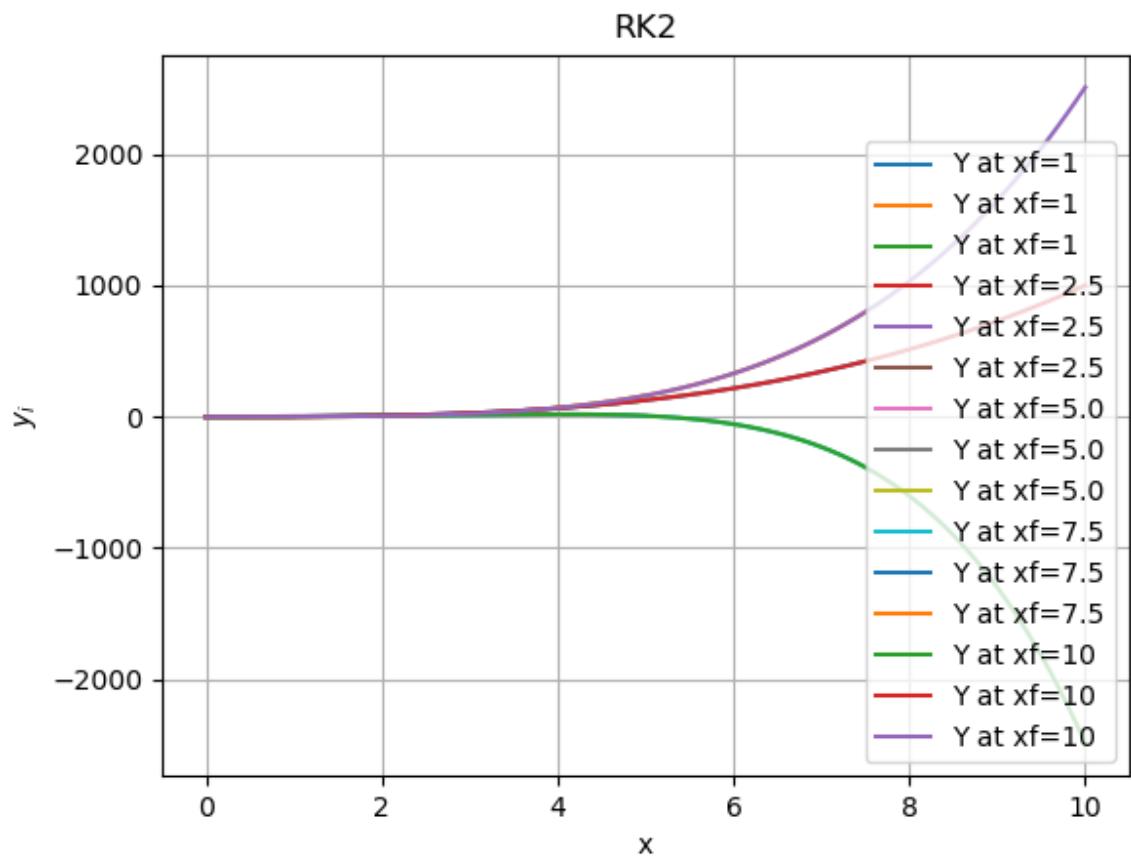


Figure 9: For Different xf : RK2

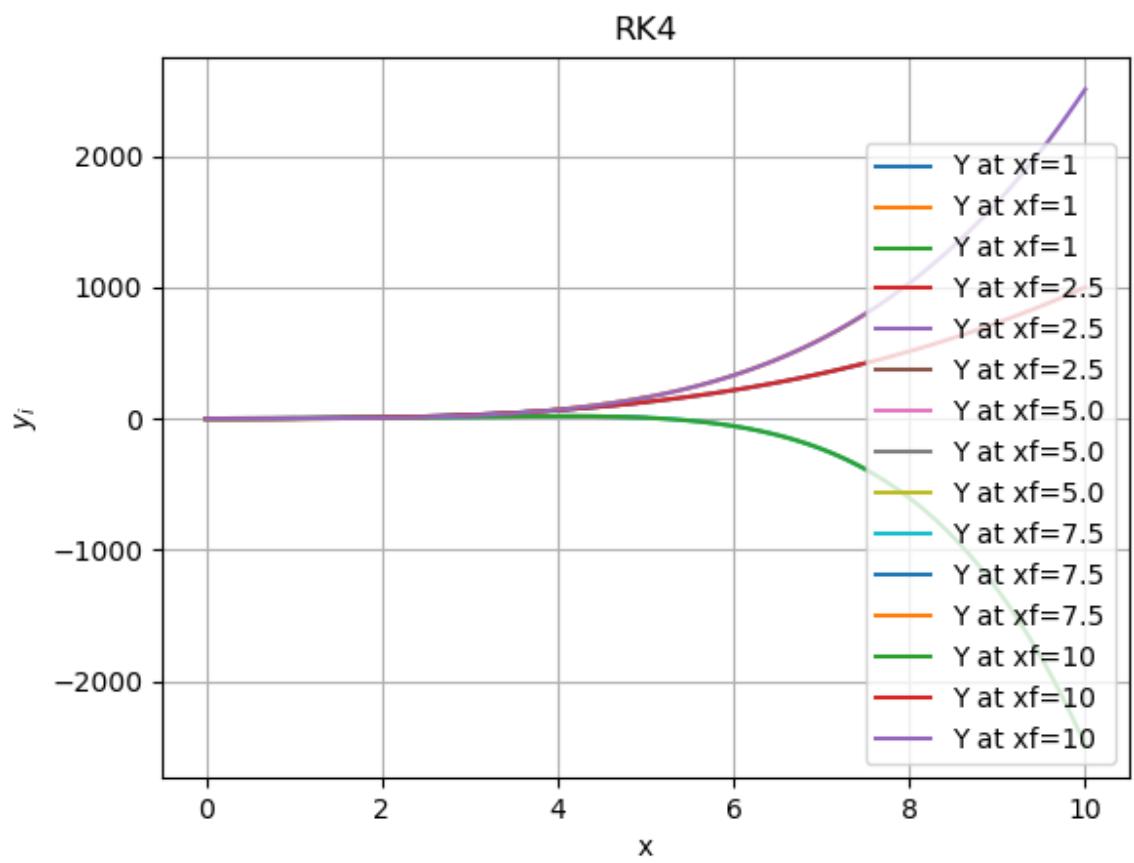


Figure 10: For Different xf : RK4

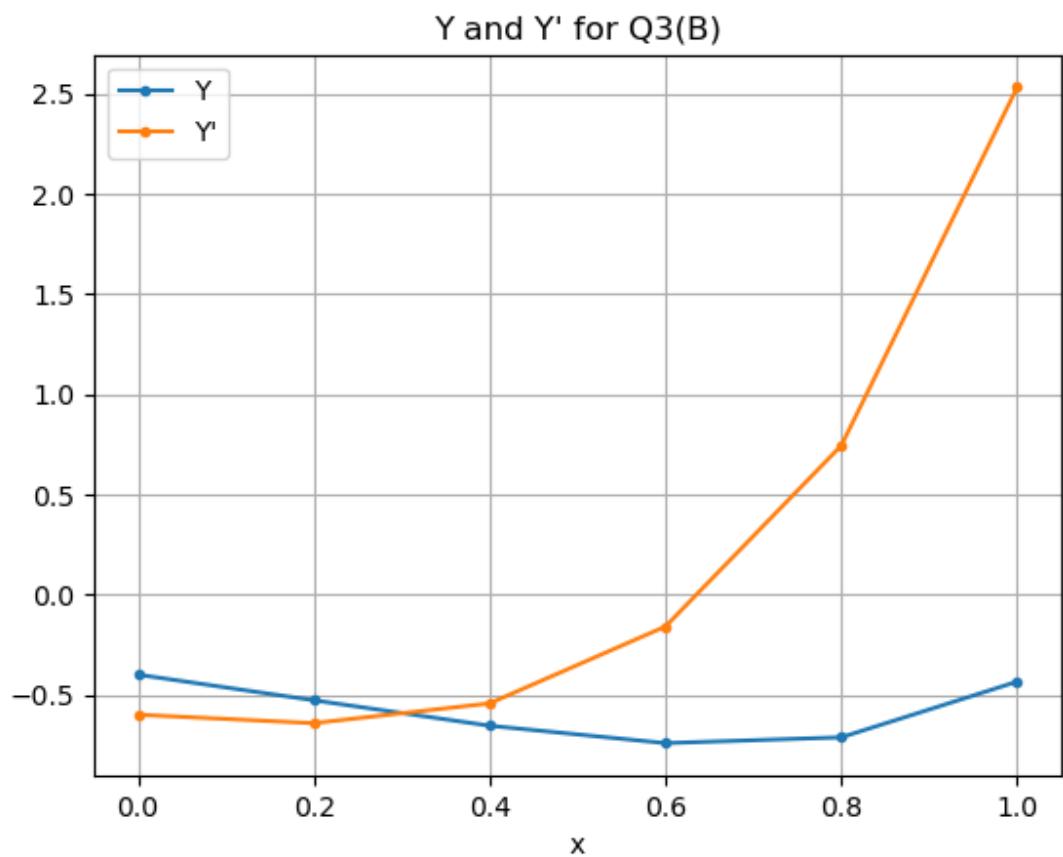


Figure 11: Q3(b)

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
<hr/>			
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= $\max(|y_{\text{ana}} - y_{\text{num}}|)$ 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