HW6 Rahul Deshmukh

November 30, 2017

```
In [1]: import numpy as np
        from sympy import *
        import matplotlib.pyplot as plt
```

Problem 1:
$y' = -5y$ and $y(0) = 1$

a) The ODE is stable as $\lambda < 0$

b) For euler's method to be stable we have the condition on the step size

$$h < \frac{-2}{\lambda}$$

and for this problem($\lambda = -5$) $h < \frac{-2}{-5}$ i.e. $h < 0.4$

Therefore Euler's Method is not stable for this ODE when using a step size of h = 0.5

c) Solving using Euler's Method

```
In [2]: def Euler_ODE(y0,h,A,a,b):
            N= int((b-a)/h)
            x= np.linspace(a,b,N+1)
            if np.size(y0)>1:
                p= np.zeros((np.size(y0),(N+1)))
                p[:,0]=np.copy(y0)
                for i in range(0,N):
                    t=a+i*h;
                    k1=A(p[:,i],t)
                    p[:,i+1]=p[:,i]+(h)*(k1)
            else:
                p= np.zeros((N+1))
                p[0]=np.copy(y0)
                for i in range(0,N):
                    t=a+i*h;
                    k1=A(p[i],t)
                    p[i+1]=p[i]+(h)*(k1)
            return(x,p)
```

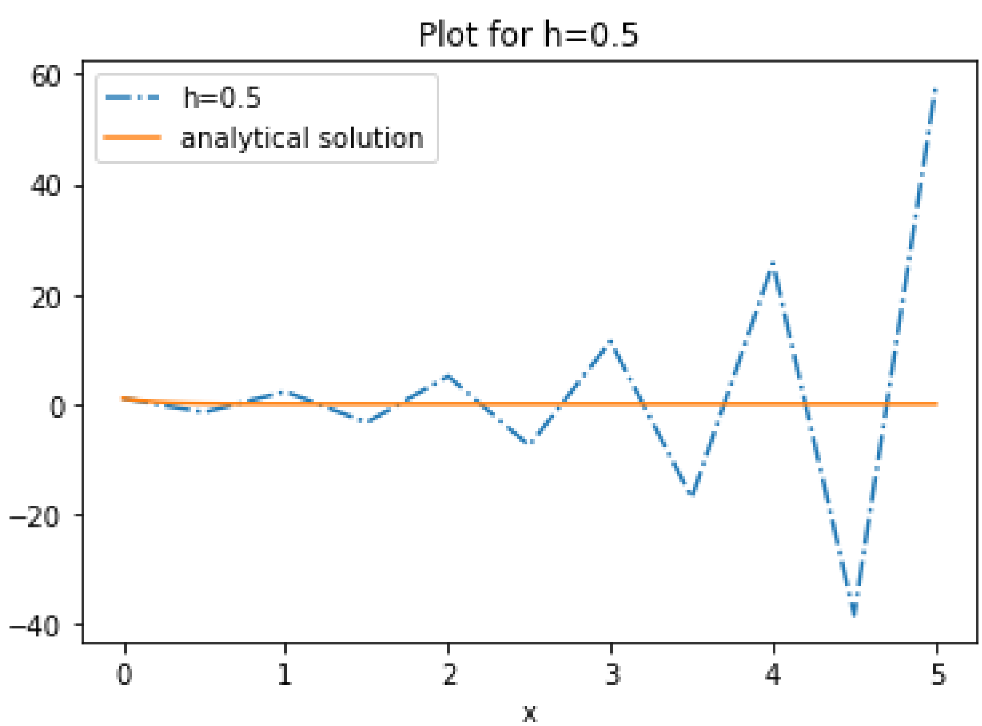
```
In [3]: a=0;b=5;y0=1
        h1=0.5;h2=0.1;
        def A(y,t):
            return(np.array([-5*y+0*t]))
        p1=Euler_ODE(y0,h1,A,a,b)
        p2=Euler_ODE(y0,h2,A,a,b)
```

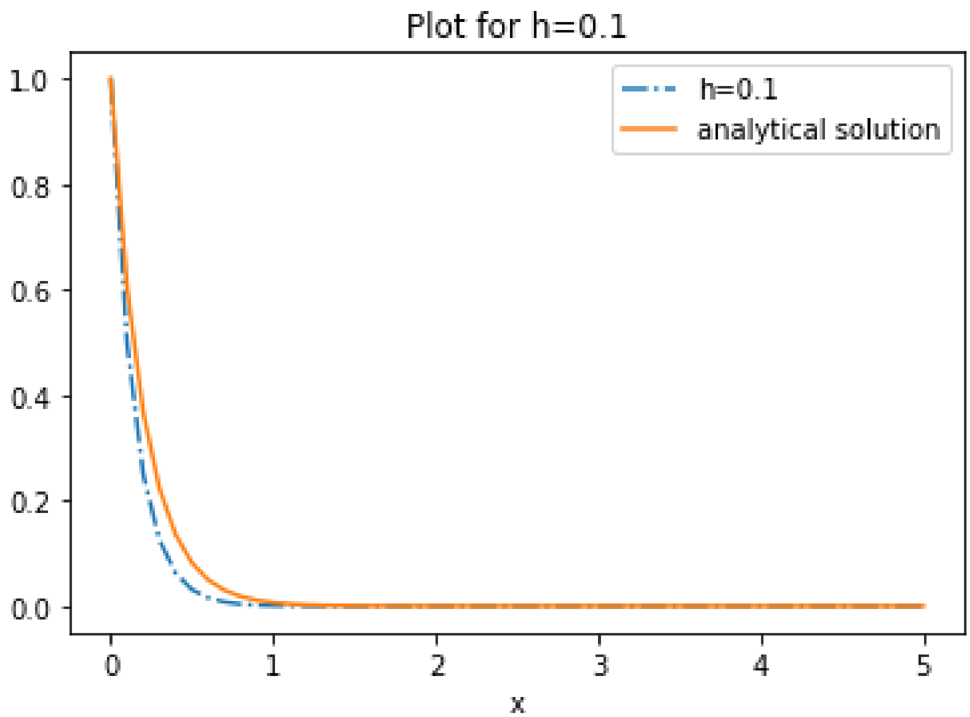
```
In [4]: x = symbols('x')
        y = Function('y')
        Y=dsolve(Derivative(y(x),x)+5*y(x),y(x))
        print('The analytical solution is: '+str(Y))
```

```
The analytical solution is: Eq(y(x), C1*exp(-5*x))
```

The analytical solution satisfying the initial value is $y(x) = e^{-5x}$

```
In [5]: a,=plt.plot(p1[0],p1[-1],'-.',label='h=0.5')
        c,= plt.plot(p2[0],np.e**(-5*p2[0]),label='analytical solution')
        plt.title('Plot for h=0.5')
        plt.legend(handles=[a,c])
        plt.xlabel('x')
        plt.show()
        b,=plt.plot(p2[0],p2[-1],'-.',label='h=0.1')
        c,= plt.plot(p2[0],np.e**(-5*p2[0]),label='analytical solution')
        plt.legend(handles=[b,c])
        plt.title('Plot for h=0.1')
        plt.xlabel('x')
        plt.show()
```



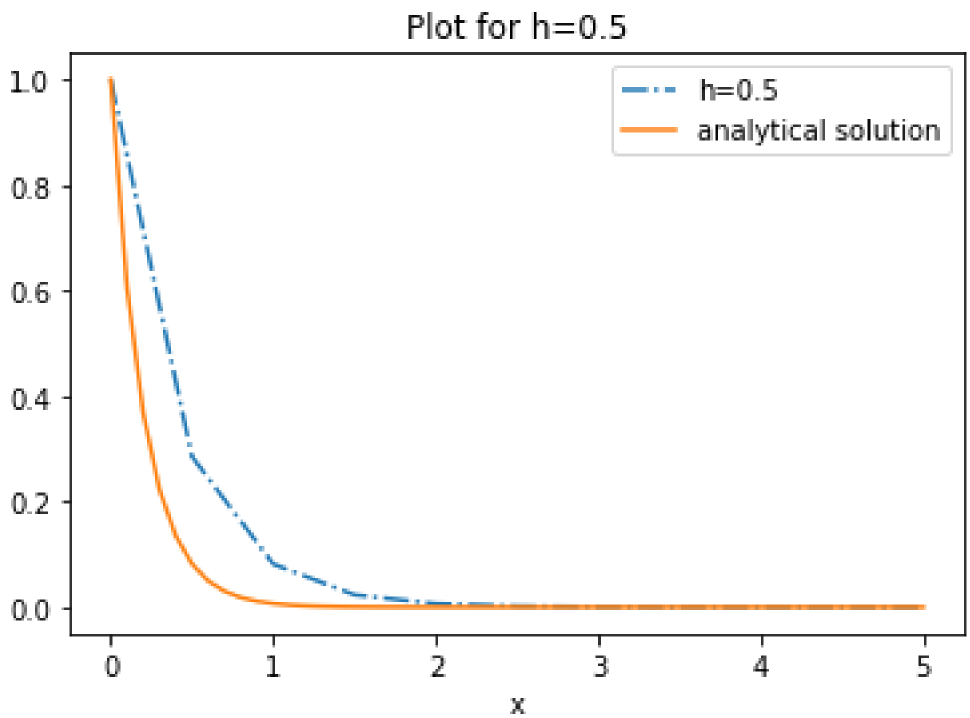


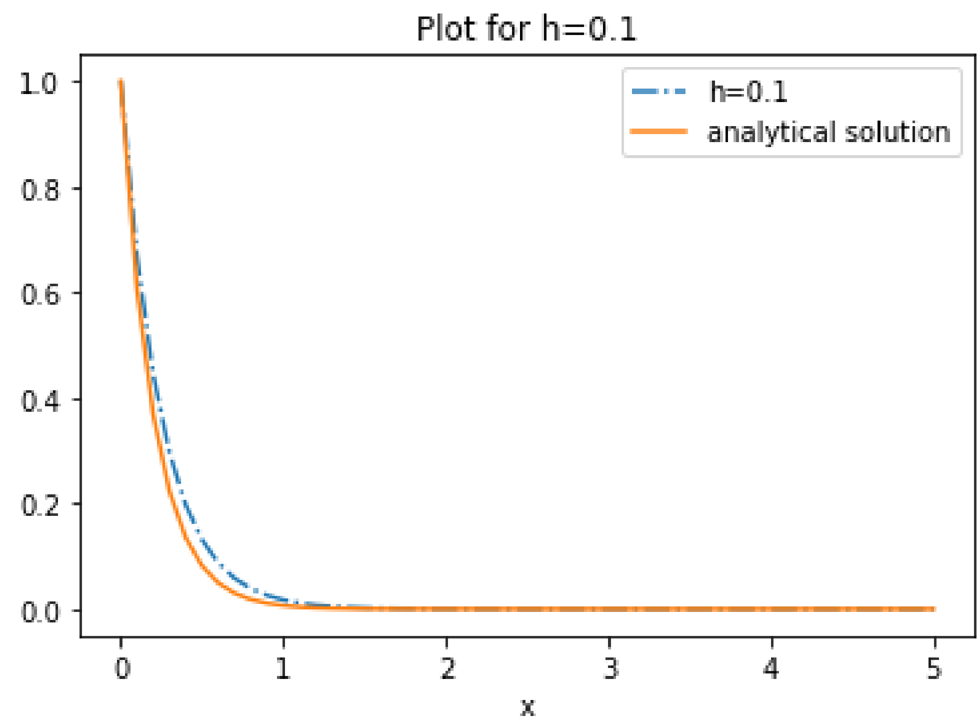
d) For Backward Eulers to be stable we have the condition on $h > 0$. Therefore for h = 0.5 the backward eulers method wil be stable

e) Solving using Backward Eulers Method

```
In [6]: def back_euler(y0,h,f,a,b): #function good for single ODE of the form y'=f(y,t) with y t
            N= int((b-a)/h)         #better to use Euler_ODE when we have higher power on y term
            x= np.linspace(a,b,N+1)
            p= np.zeros(N+1)
            p[0]=y0
            t = symbols('t')
            for i in range(0,N):
                s=solve(y-h*f.subs(t,a+i*h)-p[i],y) #back_euler can be messy as y can have multi
                p[i+1]=s[0]                         #and we dont have a selection criteria
            return(x,p)
        y = symbols('y')
        t = symbols('t')
        f=-5*y+0*t;
        y0=1;h1=0.5;h2=0.1
        a=0;b=5
        p1=back_euler(y0,h1,f,a,b)
        p2=back_euler(y0,h2,f,a,b)
```

In [7]:
```
a,=plt.plot(p1[0],p1[-1],'-.',label='h=0.5')
c,= plt.plot(p2[0],np.e**(-5*p2[0]),label='analytical solution')
plt.title('Plot for h=0.5')
plt.legend(handles=[a,c])
plt.xlabel('x')
plt.show()
b,=plt.plot(p2[0],p2[-1],'-.',label='h=0.1')
c,= plt.plot(p2[0],np.e**(-5*p2[0]),label='analytical solution')
plt.legend(handles=[b,c])
plt.title('Plot for h=0.1')
plt.xlabel('x')
plt.show()
```





Problem 2:

a) Analytical solution

Let us convert the Second order Differential equation to a ssytems of differential equations

$$y_1 = y$$
$$y_2 = y'$$
$$y_1' = y_2$$
$$y_2' = -8.96 * y_1 - 6 * y_2$$

and initial condition

$$y_1(0) = 1$$
$$y_2(0) = 5$$

$$\Rightarrow \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -8.96 & -6 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

In [8]:
```
M= Matrix([[0,1],[-8.96,-6]])
p,d = M.diagonalize()
print(d)
print(p)
c1 = symbols('c1')
c2 = symbols('c2')
c=solve([c1*p[0,0]+c2*p[0,1]-1,c1*p[1,0]+c2*p[1,1]-5],[c1,c2])
print(c)
```

```
Matrix([[-3.20000000000000, 0], [0, -2.80000000000000]])
Matrix([[-5.00000000000000, -5.00000000000000], [16.0000000000000, 14.0000000000000]])
{c1: 3.90000000000000, c2: -4.10000000000000}
```

The analytical solution is:

$$\begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 3.9 * \begin{bmatrix} -5 \\ 16 \end{bmatrix} * e^{-3.2x} - 4.1 * \begin{bmatrix} -5 \\ 14 \end{bmatrix} e^{-2.8x}$$

$$y = -19.5 * e^{-3.2x} + 20.5 * e^{-2.8x}$$

b) Solve using 4th order Runge-Kutta method

```
In [9]: def RK4_ODE(y0,h,A,a,b):
            N= int((b-a)/h)
            x= np.linspace(a,b,N+1)
            p= np.zeros((np.size(y0),(N+1)))
            p[:,0]=np.copy(y0)
            for i in range(0,N):
                t=a+i*h;
                k1=A(p[:,i],t)
                t=a+i*h+h/2;
                k2=A((p[:,i]+((h/2)*k1)),t)
                t = a+i*h+h/2;
                k3=A((p[:,i]+((h/2)*k2)),t)
                t=a+i*h+h;
                k4=A((p[:,i]+((h)*k3)),t)
                p[:,i+1]=p[:,i]+(h/6)*(k1+2*k2+2*k3+k4)
            return(x,p)
```

```
In [10]: y0=np.array([1,5])
         h=0.08;a=0;b=8;
         def A(y,t):
             return(np.array([0*y[0]+1*y[1]+0*t,-8.96*y[0]-6*y[1]+0*t]))
         sol = RK4_ODE(y0,h,A,a,b)
         a = sol[1]
         rk=a[0,:]
```

c) Solve using Euler's method

```
In [11]: y0=np.array([1,5])
         h=0.08;a=0;b=8;
         def A(y,t):
             return(np.array([0*y[0]+1*y[1]+0*t,-8.96*y[0]-6*y[1]+0*t]))
         sol = Euler_ODE(y0,h,A,a,b)
         a = sol[1]
         eu=a[0,:]
```

d) solve using Taylor series method of order two

```
In [12]: def Taylor_ODE(y0,h,F,dF,a,b):
             N= int((b-a)/h)
             x= np.linspace(a,b,N+1)
             if np.size(y0)>1:
                 p= np.zeros((np.size(y0),(N+1)))
                 p[:,0]=np.copy(y0)
                 for i in range(0,N):
                     t=a+i*h;
                     k1=F(p[:,i],t)
                     k2=dF(p[:,i],t)
                     p[:,i+1]=p[:,i]+(h)*(k1)+(h**2/2)*k2
             else:
                 p= np.zeros((N+1))
                 p[0]=np.copy(y0)
                 for i in range(0,N):
                     t=a+i*h;
                     k1=A(p[i],t)
                     k2=dF(p[i],t)
                     p[i+1]=p[i]+(h)*(k1)+(h**2/2)*k2
             return(x,p)
```

Our system of equations is:

$$\underline{Y}' = \underline{F} = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} 0 * y_1 + 1 * y_2 \\ -8.96 * y_1 - 6 * y_2 \end{bmatrix}$$

For taylor series of order two we need to find $\underline{F}'$

$$\underline{F}' = \begin{bmatrix} y_1'' \\ y_2'' \end{bmatrix} = \begin{bmatrix} 1 * y_2' \\ -8.96 * y_1' - 6 * y_2' \end{bmatrix} = \begin{bmatrix} -8.96 * y_1 - 6 * y_2 \\ -8.96 * y_2 - 6 * (-8.96 * y_1 - 6 * y_2) \end{bmatrix}$$

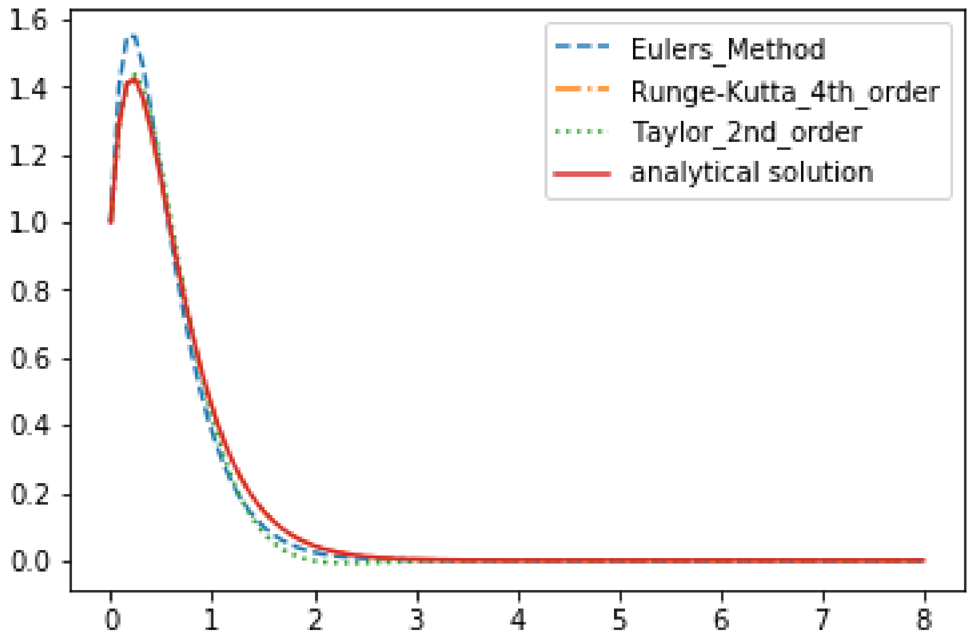
$$\Rightarrow \underline{F}' = \begin{bmatrix} -8.96 * y_1 - 6 * y_2 \\ 53.76 * y_1 + 44.96 * y_2 \end{bmatrix}$$

```
In [13]: y0=np.array([1,5])
         h=0.08;a=0;b=8;
         def F(y,t):
             return(np.array([0*y[0]+1*y[1]+0*t,-8.96*y[0]-6*y[1]+0*t]))
         def dF(y,t):
             return(np.array([-8.96*y[0]-6*y[1]+0*t,53.76*y[0]+44.96*y[1]+0*t]))
         sol = Taylor_ODE(y0,h,F,dF,a,b)
         a = sol[1]
         ty=a[0,:]
```

d) Plot

```
In [14]: def ana(t):
             return(-19.5*np.e**(-3.2*t)+20.5*np.e**(-2.8*t))
```

```
peu,=plt.plot(sol[0],eu,'--',label='Eulers_Method')
prk,=plt.plot(sol[0],rk,'-.',label='Runge-Kutta_4th_order')
pty,=plt.plot(sol[0],ty,':',label='Taylor_2nd_order')
pana,=plt.plot(sol[0],ana(sol[0]),label='analytical solution')
plt.legend(handles=[peu,prk,pty,pana])
plt.show()
```

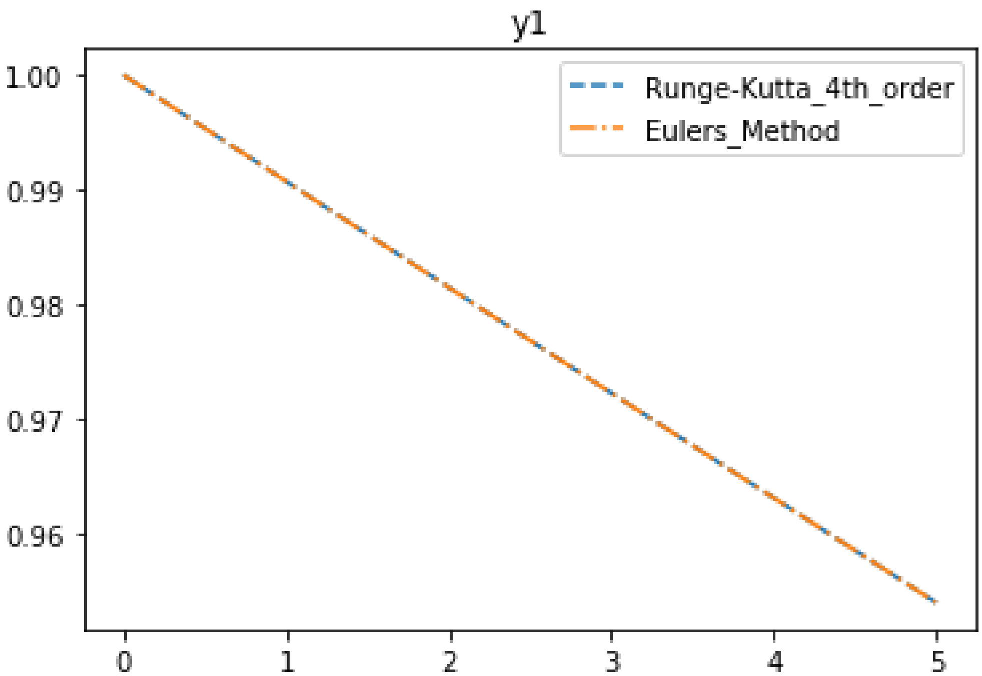


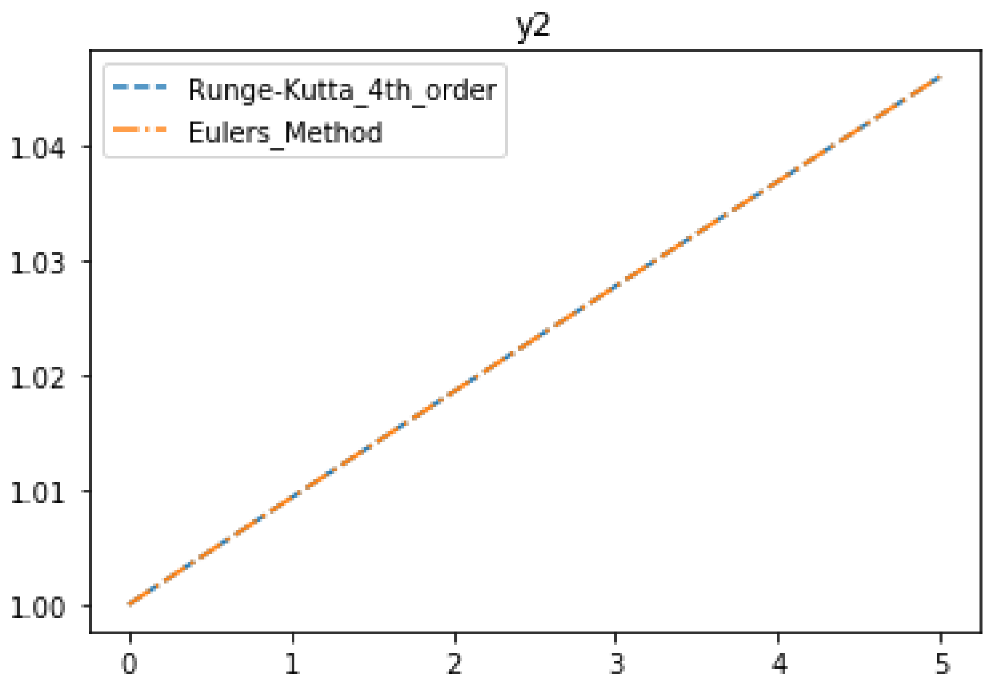
Problem 3:

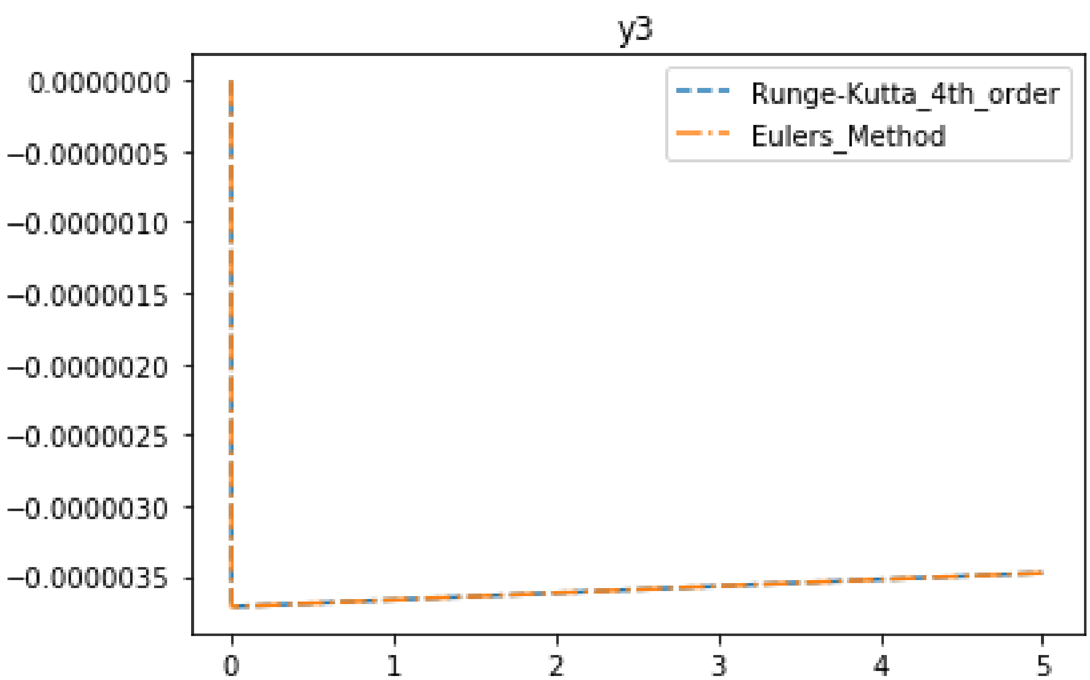
```
In [15]: def F(y,t):
             return(np.array([-0.013*y[0]-1000*y[0]*y[2]+0*t,-2500*y[1]*y[2]+0*t,
                              -0.013*y[0]-1000*y[0]*y[2]-2500*y[1]*y[2]+0*t]))
         y0=np.array([1,1,0])
         h=0.0001;a=0;b=5;
         sol1 = RK4_ODE(y0,h,F,a,b)
         m = sol1[1]
         sol2=Euler_ODE(y0,h,F,a,b)
         n = sol2[1]
```

```
In [16]: rk=m[0,:]
         eu=n[0,:]
         prk,=plt.plot(sol1[0],rk,'--',label='Runge-Kutta_4th_order')
         peu,=plt.plot(sol2[0],eu,'-.',label='Eulers_Method')
         plt.legend(handles=[prk,peu])
         plt.title('y1')
         plt.show()
```

```
rk=m[1,:]
eu=n[1,:]
prk,=plt.plot(sol1[0],rk,'--',label='Runge-Kutta_4th_order')
peu,=plt.plot(sol2[0],eu,'-.',label='Eulers_Method')
plt.legend(handles=[prk,peu])
plt.title('y2')
plt.show()
rk=m[2,:]
eu=n[2,:]
prk,=plt.plot(sol1[0],rk,'--',label='Runge-Kutta_4th_order')
peu,=plt.plot(sol2[0],eu,'-.',label='Eulers_Method')
plt.legend(handles=[prk,peu])
plt.title('y3')
plt.show()
```




y2
Runge-Kutta_4th_order
Eulers_Method
1.04
1.03
1.02
1.01
1.00
0
1
2
3
4
5


y3
Runge-Kutta_4th_order
Eulers_Method
0.0000000
−0.0000005
−0.0000010
−0.0000015
−0.0000020
−0.0000025
−0.0000030
−0.0000035
0
1
2
3
4
5

Problem 4:

For a Second order Runge-Kutta Method we have:

$$Q(z) = 1 + z + \frac{1}{2}z^2$$

so absolute stability requires:

$$\left|1 + (h\lambda) + \frac{1}{2}(h\lambda)^2\right| < 1$$

lets solve this inequality

```
In [17]: x = symbols('x')
         q=1+x+(1/2)*x**2
         print(solve_univariate_inequality(q<1, x))
```

```
And(-2.0 < x, x < 0.0)
```

Therefore for Second order Runge-Kutta Method we have the condition the following condition:

$$-2 < (h\lambda) < 0$$

We have $0 < h < \frac{2}{|\lambda|}$ for $\lambda < 0$

a)

```
In [18]: def RK2_ODE(y0,h,A,a,b):
             N= int((b-a)/h)
             x= np.linspace(a,b,N+1)
             if np.size(y0)>1:
                 p= np.zeros((np.size(y0),(N+1)))
                 p[:,0]=np.copy(y0)
                 for i in range(0,N):
                     t=a+i*h;
                     k1=A(p[:,i],t)
                     t=a+i*h+h;
                     k2=A((p[:,i]+((h)*k1)),t)
                     p[:,i+1]=p[:,i]+(h/2)*(k1+k2)
             else:
                 p= np.zeros((N+1))
                 p[0]=np.copy(y0)
                 for i in range(0,N):
                     t=a+i*h;
                     k1=A(p[i],t)
                     t=a+i*h+h;
                     k2=A((p[i]+((h)*k1)),t)
                     p[i+1]=p[i]+(h/2)*(k1+k2)
             return(x,p)
```

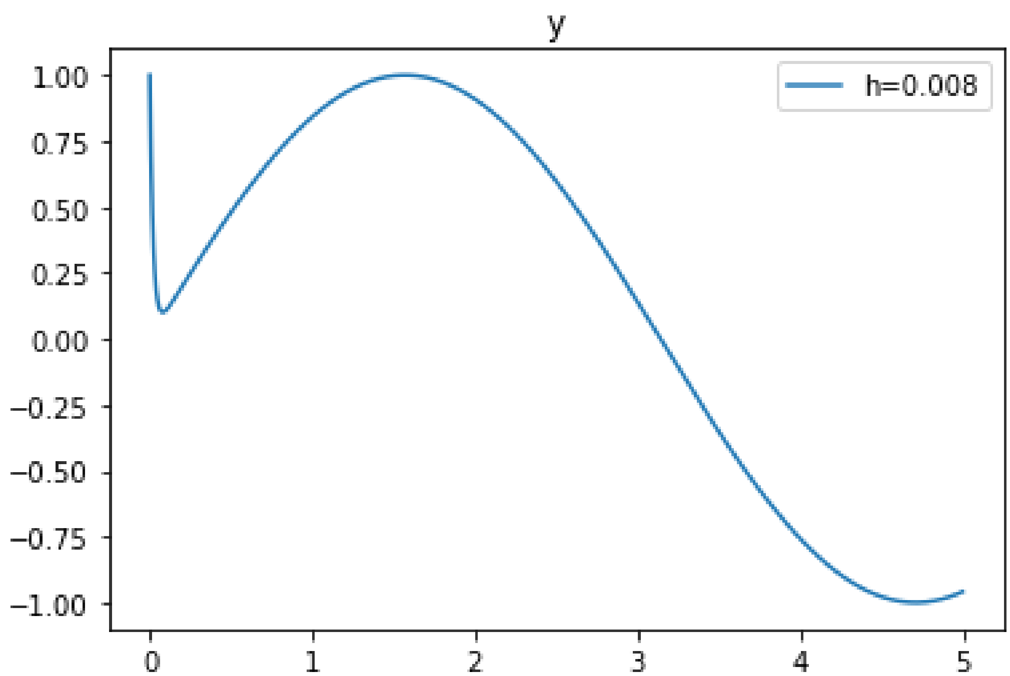
i)

Finding eigenvalues of homogenous system and for this Problem $\lambda = -200$ and $0 < h < 2/|\lambda|$ for $\lambda < 0$

$\Rightarrow h < 2/200 = 0.01$

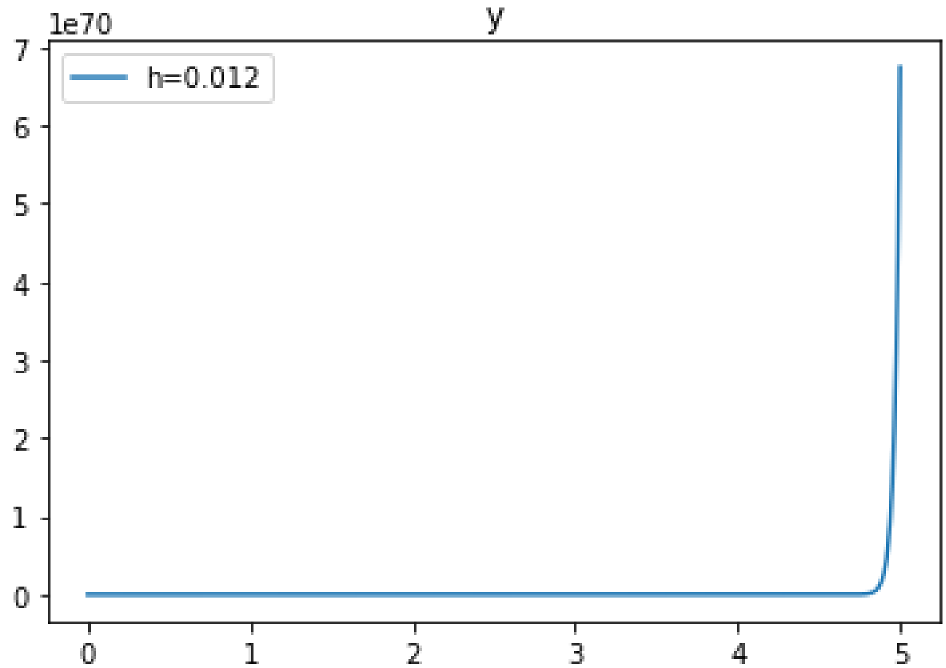
ii)

In [19]:
```
def F(y,t):
    return(-200*y+200*np.sin(t)+np.cos(t))
y0=1;a=0;b=5;h=0.008
sol=RK2_ODE(y0,h,F,a,b)
rk = sol[1]
prk,=plt.plot(sol[0],rk,label='h=0.008')
plt.legend(handles=[prk])
plt.title('y')
plt.show()
```



iii)

In [20]:
```
y0=1;a=0;b=5;h=0.012
sol=RK2_ODE(y0,h,F,a,b)
rk= sol[1]
prk,=plt.plot(sol[0],rk,label='h=0.012')
plt.legend(handles=[prk])
plt.title('y')
plt.show()
```



b)

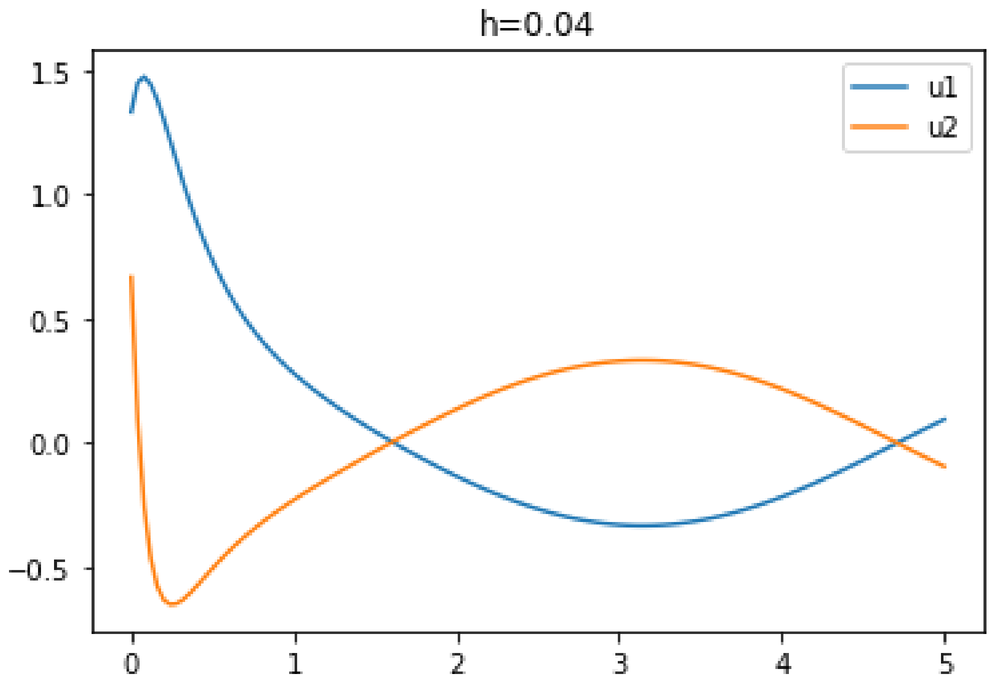
i) Finding eigenvalues of homogenous system and $0 < h < 2/|\lambda|$ for $\lambda < 0$

```
In [21]: M=Matrix([[9,24],[-24,-51]])
         p,v=M.diagonalize()
         print(v)

Matrix([[-39, 0], [0, -3]])
```

Therefore $h < 2/39 \simeq 0.05128$

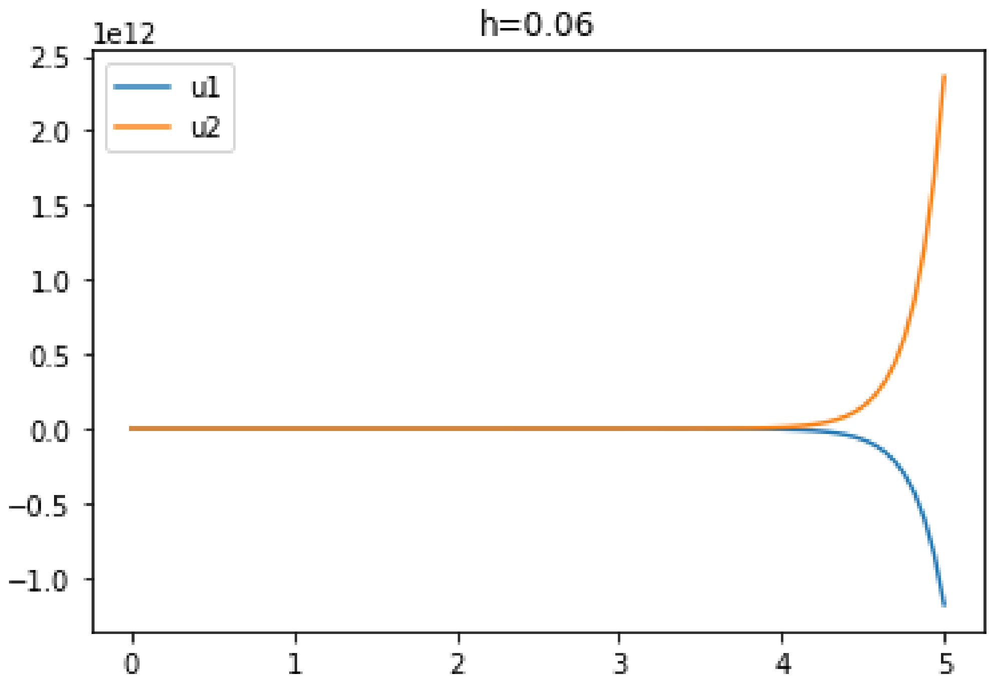
ii)

```
In [22]: def F(y,t):
             return(np.array([9*y[0]+24*y[1]+5*np.cos(t)-(1/3)*np.sin(t),
                              -24*y[0]-51*y[1]-9*np.cos(t)+(1/3)*sin(t)]))
         y0= np.array([(4/3),(2/3)])
         a=0;b=5;h=0.04
         sol=RK2_ODE(y0,h,F,a,b)
         rk = sol[1]

In [23]: u1,=plt.plot(sol[0],rk[0,:],label='u1')
         u2,=plt.plot(sol[0],rk[1,:],label='u2')
         plt.legend(handles=[u1,u2])
         plt.title('h=0.04')
         plt.show()
```



iii)

```
In [24]: a=0;b=5;h=0.06
         sol=RK2_ODE(y0,h,F,a,b)
         rk = sol[1]
         u1,=plt.plot(sol[0],rk[0,:],label='u1')
         u2,=plt.plot(sol[0],rk[1,:],label='u2')
         plt.legend(handles=[u1,u2])
         plt.title('h=0.06')
         plt.show()
```



c)

i) Finding eigenvalues of homogenous system and $0 < h < 2/|\lambda|$ for $\lambda < 0$

```
In [25]: M=Matrix([[-20,-19],[-19,-20]])
         p,v=M.diagonalize()
         print(v)
```

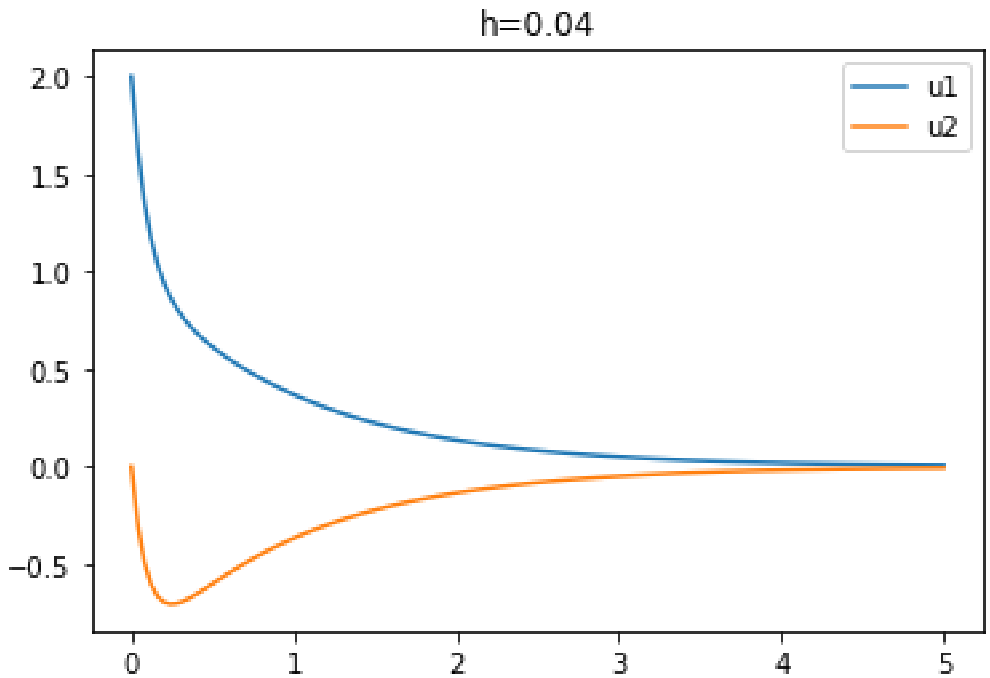
```
Matrix([[-39, 0], [0, -1]])
```

Therefore $h < 2/39 \simeq 0.05128$

ii)

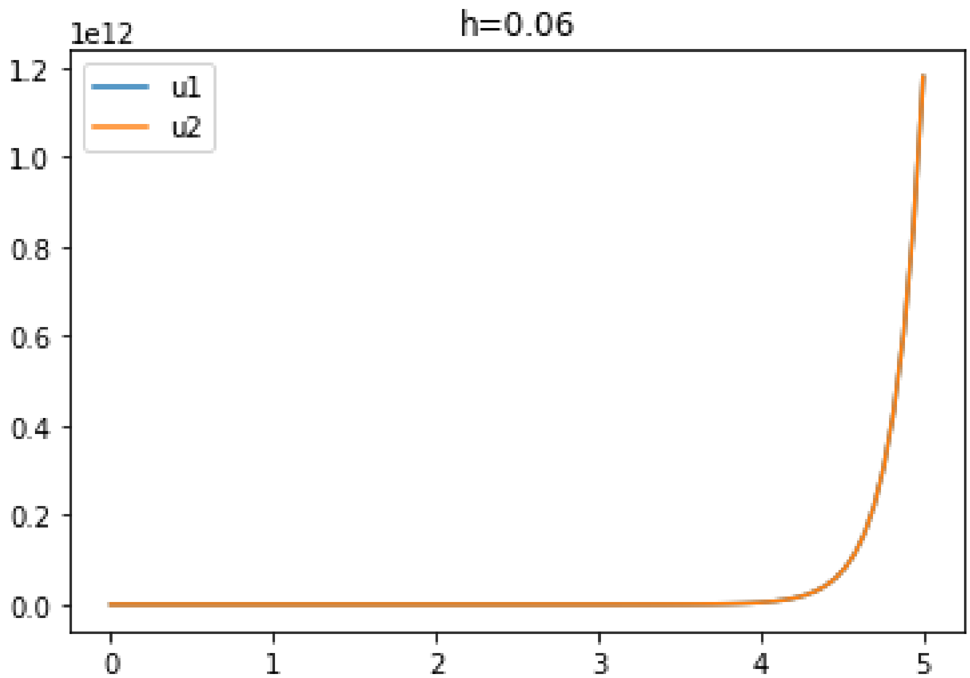
```
In [26]: def F(y,t):
             return(np.array([-20*y[0]-19*y[1],
                              -19*y[0]-20*y[1]]))
         y0= np.array([2,0])
         a=0;b=5;h=0.04
         sol=RK2_ODE(y0,h,F,a,b)
         rk = sol[1]
```

```
In [27]: u1,=plt.plot(sol[0],rk[0,:],label='u1')
         u2,=plt.plot(sol[0],rk[1,:],label='u2')
         plt.legend(handles=[u1,u2])
         plt.title('h=0.04')
         plt.show()
```



iii)

```
In [28]: a=0;b=5;h=0.06
         sol=RK2_ODE(y0,h,F,a,b)
         rk = sol[1]
         u1,=plt.plot(sol[0],rk[0,:],label='u1')
         u2,=plt.plot(sol[0],rk[1,:],label='u2')
         plt.legend(handles=[u1,u2])
         plt.title('h=0.06')
         plt.show()
```


h=0.06
1e12
1.2
1.0
0.8
0.6
0.4
0.2
0.0
u1
u2
0
1
2
3
4
5