Lab Week-13 (Exact and Numerical Sol ODE)

November 15, 2023

[1]:
```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
sp.init_printing()
```

1 Analytical solutions to Differential equations

1.1 Solving the Differential equation $\frac{dT(t)}{dt} = -k(T(t) - T_a)$ with initial condition $T(0) = T_0$

1.1.1 Defining Ordinary Differential Equation (ODE)

[2]:
```
t,k,Ta,To=sp.symbols("t,k,T_a,T_o")
T=sp.Function("T")
ode=T(t).diff(t)+k*(T(t)-Ta)
sp.Eq(ode,0)
```

[2]:
$$k\left(-T_a + T(t)\right) + \frac{d}{dt}T(t) = 0$$

1.1.2 Finding the general solution to ODE

[3]:
```
gen_sol=sp.dsolve(ode)
gen_sol
```

[3]:
$$T(t) = C_1 e^{-kt} + T_a$$

[4]:
```
gen_sol.lhs
```

[4]:
$$T(t)$$

[5]:
```
gen_sol.rhs
```

[5]:
$$C_1 e^{-kt} + T_a$$

1.1.3 Finding the particular solution of Initial-Value Problem (IVP)

[6]:
```
part_sol=sp.dsolve(ode,T(t),ics={T(0):To})
part_sol
```

[6]: $T(t) = T_a + (-T_a + T_o)\, e^{-kt}$

[7]:
```
value={t:0.5}
part_sol.subs(value)
```

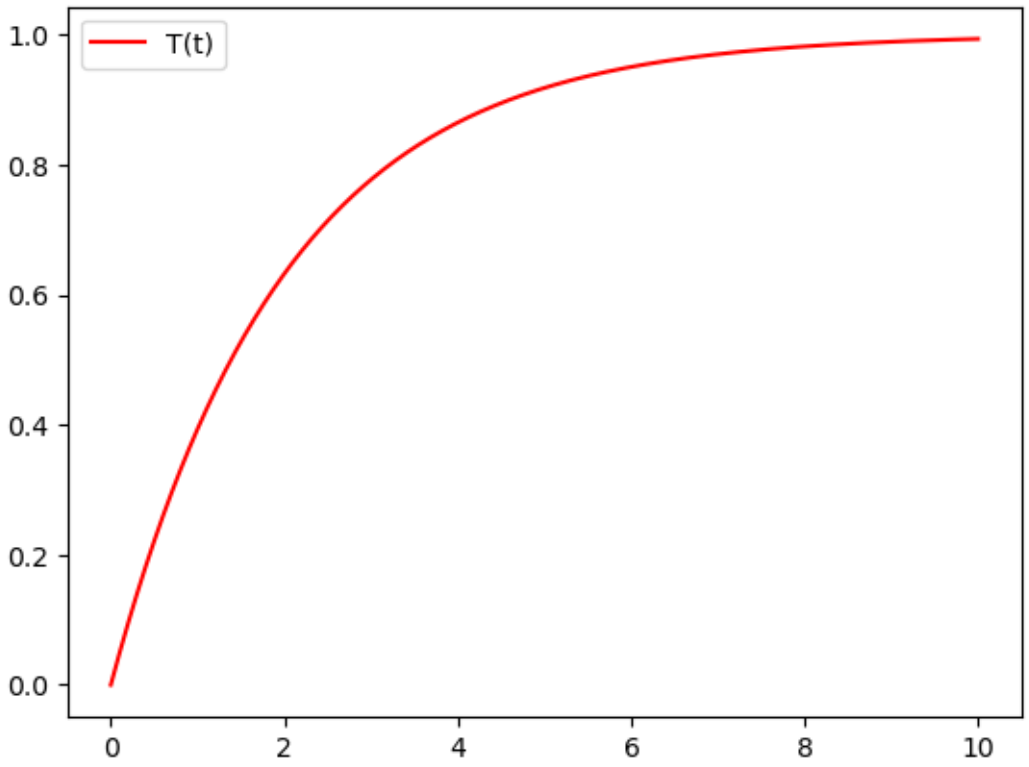
[7]: $T(0.5) = T_a + (-T_a + T_o)\, e^{-0.5k}$

1.1.4 Plotting the particular solution

[8]:
```
time=np.linspace(0,10,1000)
sol=sp.lambdify(t,part_sol.rhs.subs({k:0.5,To:0,Ta:1}),modules="numpy")
#sol(time)
```

[9]:
```
fig,ax=plt.subplots()
ax.plot(time,sol(time),color="red",label="T(t)")
ax.legend()
```

[9]:
```
<matplotlib.legend.Legend at 0x2148363c4d0>
```



1.2 Solving first order differential system

$$\frac{dx}{dt} = y, \frac{dy}{dt} = -x$$

with initial condition $x(0) = 0, y(0) = 1$

1.2.1 Defining the systems of first order ODE

[10]:
```
from sympy.solvers.ode.systems import dsolve_system
t=sp.symbols("t")
X=sp.Function("X")
Y=sp.Function("Y")
ode1=Y(t).diff(t)+X(t)
ode2=X(t).diff(t)-Y(t)
eqs=[sp.Eq(ode1,0),sp.Eq(ode2,0)]
eqs
```

[10]:
$$\left[X(t) + \frac{d}{dt}Y(t) = 0,\ -Y(t) + \frac{d}{dt}X(t) = 0\right]$$

1.2.2 Finding the general solution to given system of ODEs

[11]:
```
gen_sys_sol=sp.dsolve(eqs,[X(t),Y(t)])
gen_sys_sol
```

[11]:
$$[X(t) = C_1 \sin(t) + C_2 \cos(t),\ Y(t) = C_1 \cos(t) - C_2 \sin(t)]$$

1.2.3 Finding the particular solution to system of ODEs

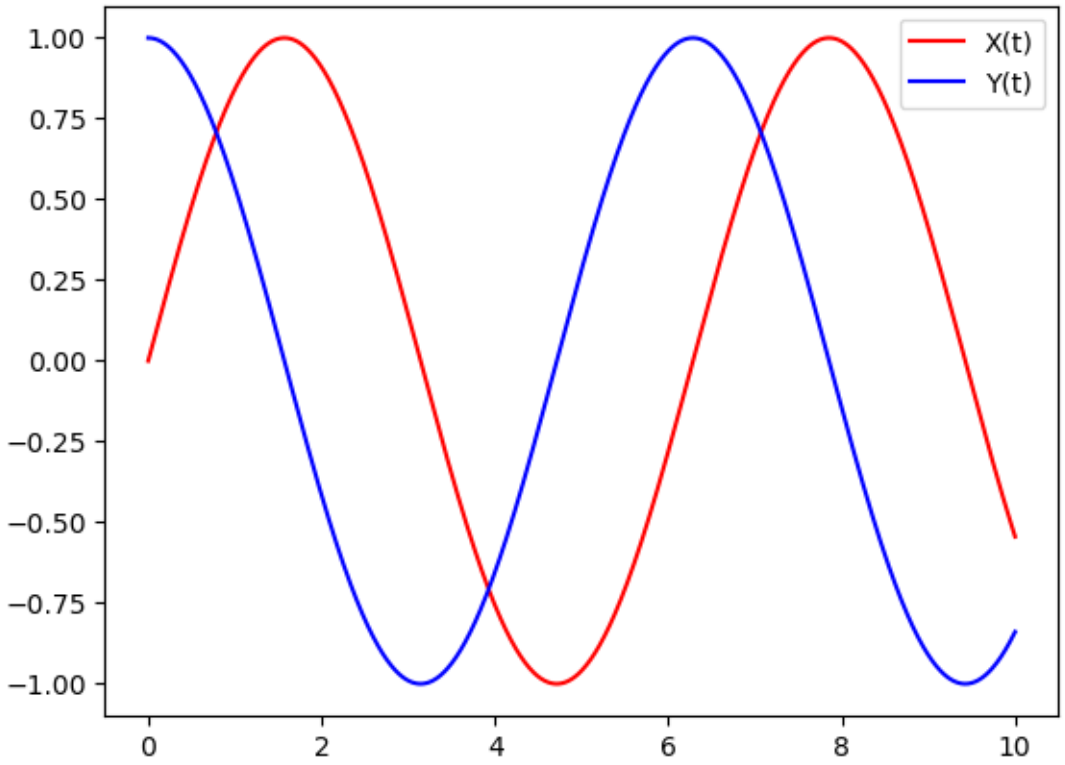
[12]:
```
part_sys_sol=sp.dsolve(eqs,[X(t),Y(t)],ics={X(0):0,Y(0):1})
part_sys_sol
```

[12]:
$$[X(t) = \sin(t),\ Y(t) = \cos(t)]$$

1.2.4 Plotting the solutions of given system of ODEs

[16]:
```
sol_X=sp.lambdify(t,part_sys_sol[0].rhs,modules="numpy")
sol_Y=sp.lambdify(t,part_sys_sol[1].rhs,modules="numpy")
fig,ax1=plt.subplots()
ax1.plot(time,sol_X(time),color="red",label="X(t)")
ax1.plot(time,sol_Y(time),color="blue",label="Y(t)")
ax1.legend()
```

[16]:
```
<matplotlib.legend.Legend at 0x214838595d0>
```



2 Numerical Solutions to Differential equations

2.0.1 Function for Euler method

[18]:
```
def rk1(f,time,initial,h): #define the function
    index=np.arange(time[0],time[1]+h,h) #divide the interval
    n=len(index) # how many points here
    F=np.zeros(n) #where we store the solution
    F[0]=initial #if y is solution then the first entry of F is y(t_0)
    for i in range(1,n):
        values={t:index[i-1],y:F[i-1]}
        F[i]=F[i-1]+h*f.subs(values) #y_n=y_(n-1)+h(f(x_(n-1),y_(n-1)))
    return F
```

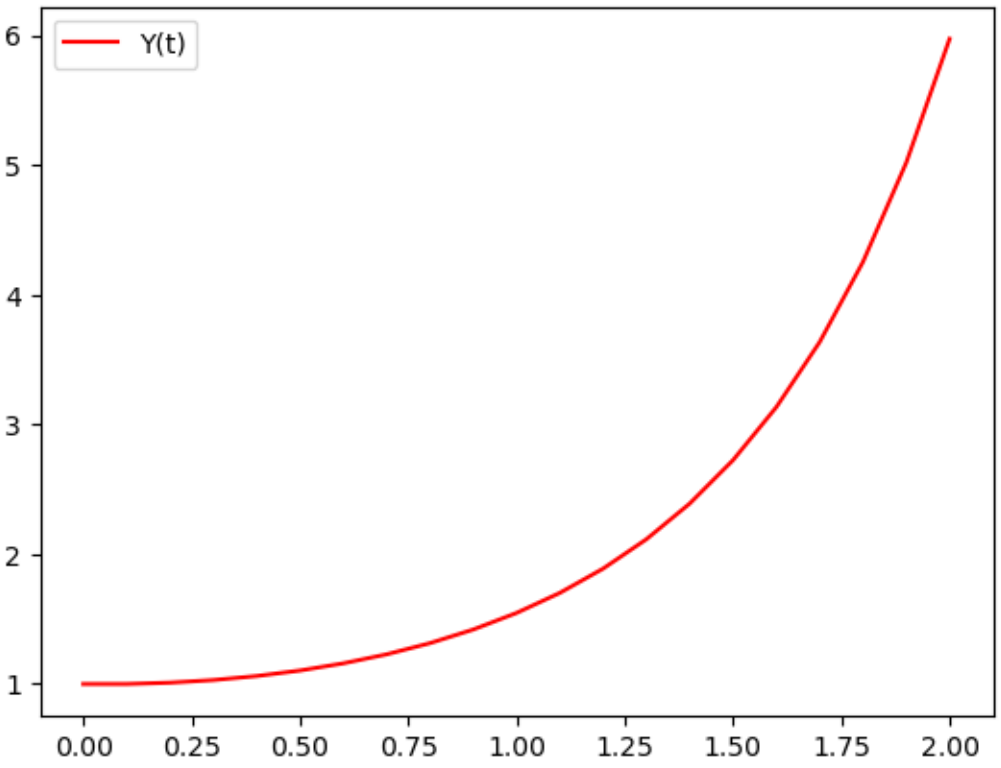
[19]:
```
y,t=sp.symbols("y t") #define the variable
```

2.0.2 Consider the differential equation $y' = ty$, $y(0) = 1$ and step size=0.1 over the interval $[0, 2]$

[20]:
```
f=t*y  #y'=f(t,y)
time=[0,2] #intravl(time)
initial=1 #y(t_0)
h=0.1  #step size
Euler_sol=rk1(f,time,initial,h) #here, we call the function
```

[21]:
```
time=np.arange(time[0],time[1]+h,h)
fig,ax=plt.subplots()
ax.plot(time,Euler_sol,color="red",label="Y(t)")
ax.legend()
```

[21]:
```
<matplotlib.legend.Legend at 0x214841044d0>
```



2.0.3 Function for Euler method to be implemented on system of ODEs

[22]:
```
def rk1_sys(f,time,initial,h):
    index=np.arange(time[0],time[1]+h,h)
    n=len(index)
    F=np.zeros((len(initial),n))
```

```
    F[0,0]=initial[0]
    F[1,0]=initial[1]
    for i in range(1,n):
        values={t:index[i-1],x:F[0,i-1],y:F[1,i-1]}
        F[0:,i]=F[0:,i-1]+h*f.subs(values)[0]
        F[1:,i]=F[1:,i-1]+h*f.subs(values)[1]
    return F
```

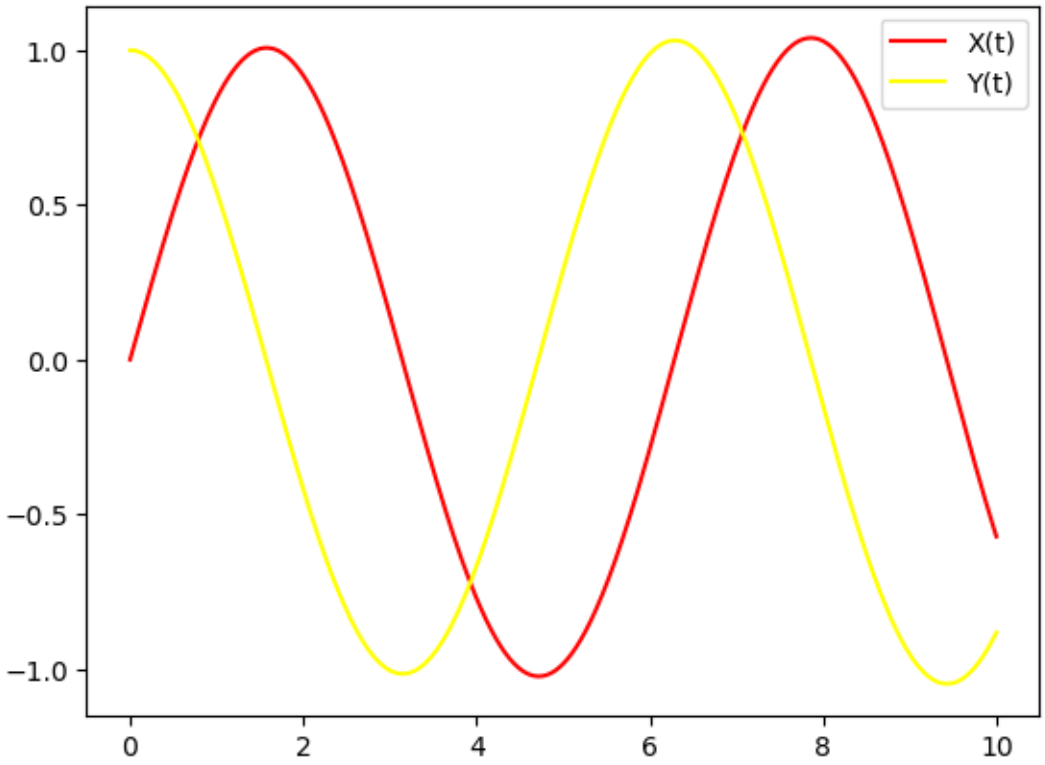
[23]:
```
x,y,t=sp.symbols("x y t")
```

2.0.4 Consider the differential equation $x' = y$,$y' = -x$,and $x(0) = 1, y(0) = 1$ and step size=0.01 over the interval $[0, 10]$

[24]:
```
f=sp.Matrix([y,-x])
time=[0,10]
initial=[0,1]
h=0.01
Euler_sol=rk1_sys(f,time,initial,h)
time=np.arange(time[0],time[1]+h,h)
```

[25]:
```
fig,ax=plt.subplots()
ax.plot(time,Euler_sol[0],color="red",label="X(t)")
ax.plot(time,Euler_sol[1],color="yellow",label="Y(t)")
#ax.plot(Euler_sol[0],Euler_sol[1],color="blue",label="X(t) and Y(t)")
ax.legend()
```

[25]:
```
<matplotlib.legend.Legend at 0x214840ace50>
```


X(t)
Y(t)
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10

[ ]: