UNIT 6 Euler's Method to Solve Ordinary Differential Equations

1. To Solve First Order Differential Equation by Euler Method and compare it with Exact Solution and Inbuilt Function.

```
# import libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

def f(y,x):         #Differential Equation to solve
    return np.exp(x)

def f_exact(y,x):   #Exact Equation
    return np.exp(x)

print("\n\n\tMehendi Hasan\n\n\t2230248\n\n")


#User Inputs
x0=float(input("Enter Initial value of X: "))
y0=float(input("Enter Value of Y at Initial value of X: "))
h=float(input("Enter Step Size: "))
b=float(input("Enter last value of interval: "))

x_values=np.arange(0,b+h,h)
y_values=np.zeros(len(x_values))
y_exact=np.zeros(len(x_values))
y_odeint=np.zeros(len(x_values))
x_values[0]=x0
y_values[0]=y0
for i in range(len(y_values)-1):
    y_values[i+1]=y_values[i] + h*f(y_values[i],x_values[i])
y_exact=np.array(f_exact(y_values,x_values))
y_odeint=odeint(f,y0,x_values)


#plot
plt.subplot(3,1,1)
plt.plot(x_values,y_values,label="F(x) Eulers",color="red")
plt.xlabel("X Points")
plt.ylabel("Y Points")
plt.grid()
plt.legend()
plt.subplot(3,1,2)
plt.plot(x_values,y_exact,label="F(x) Exact",color="blue")
plt.xlabel("X Points")
plt.ylabel("Y Points")
plt.grid()
plt.legend()
plt.subplot(3,1,3)
plt.plot(x_values,y_odeint,label="F(x) Odeint",color="green")
plt.xlabel("X Points")
plt.ylabel("Y Points")
plt.grid()
plt.legend()
plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Solve First Order Differential Equation by Euler Method and compare it with Exact Solution and Inbuilt Function.")
plt.show()
```

2. To Plot Newton's cooling law ODE by Euler method, Exact solution & Inbuilt solver.

```
#libraries
import matplotlib.pyplot as plt
import numpy as np
import scipy.integrate as  it

def f(T,t):    # Differential Equation of cooling
   return (-K)*(T-Ts)

print("\n\n\tMehendi Hasan\n\n\t2230248\n\n")
print("Newton's Law of Cooling\n\nTemperature is in Degree Celsius and time is in secons\n\n")

T0=int(input("Enter initial Temperature of Object: "))
Ts=int(input("Enter Surrounding temperature: "))
t=int(input("Enter time from t=0, at which temperature of Object to be calculated: "))
h=0.001    #step size
K=0.1    #cooling constant
x_points=np.arange(0,t+h,h)
T=np.zeros(len(x_points))
TExact=np.zeros(len(x_points))
T[0]=T0
TExact[0]=T0

for i in range(len(x_points)-1):
   T[i+1]= (T[i] + (h*(f(T[i],x_points))))
TExact = Ts + ((T0-Ts)*np.exp((-K)*(x_points)))
solOdeint=it.odeint(f,T0,x_points)

#ploting

plt.subplot(3,1,1)
plt.plot(x_points,T,label="Euler's Solution")
plt.grid()
plt.title("Euler's Method")
plt.xlabel("Time")
plt.ylabel("Temperature of Object")
plt.legend()
plt.subplot(3,1,2)
plt.plot(x_points,TExact,label='Exact Equation Solution')
plt.grid()
plt.title("Exact Equation")
plt.xlabel("Time")
plt.ylabel("Temperature of Object")
plt.subplot(3,1,3)
plt.plot(x_points,solOdeint,label='Odient Solution')
plt.grid()
plt.title("Odeient Solution")
plt.xlabel("Time")
plt.ylabel("Temperature of Object")
plt.legend()
plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Plot Newton's cooling law ODE by Euler method, Exact solution & Inbuilt solver")
plt.show()
```

3. To Plot Radioactive Decay ODE by Euler method, Exact solution & Inbuilt solver.

```
#importing libraries

import matplotlib.pyplot as plt

import numpy as np

import scipy.integrate as it


def diff_Equ(N,t):

   return (-1)*(K)*(N)

def Exact_Equ(N,t):

   return N0*(np.exp((-1)*(K)*(t)))


print("\n\n\tMehendi Hasan\n\n\t2230248\n\nRadioactive Decay \n\nTime is in Seconds\n")


#itaking input from user

N0=int(input("Enter Number of Parent Atoms at t=0: "))

t=int(input("Enter time instant at which Remaining of Parent Atoms to be calculated: "))

K=float(input("Enter Radioactive Decay constant value: "))      # Radioactive Decay Constant


h=0.001      # Step size for Euler method

t_array=np.arange(0,t+h,h)     #initializing time array(independent Variable)

Y_differential=np.zeros(len(t_array))      #Initializing array for values of dependent variable(Euler's metod)

Y_Exact=np.zeros(len(t_array))        # #Initializing array for values of dependent variable(Solution equation)

Y_differential[0] = Y_Exact[0] = N0       #Initial values of dependent variable Y at independent variable t=0

for i in range(len(t_array)-1):

     Y_differential[i+1]=Y_differential[i] + h*(diff_Equ(Y_differential[i],t_array[i]))

Y_Exact=Exact_Equ(Y_Exact,t_array)
# solution equation

solOdeint=it.odeint(diff_Equ,N0,t_array)
# odeint solution


#ploting all the values of dependent variable with respect to independent variable

plt.subplot(3,1,1)

plt.plot(t_array,Y_differential,color="green",label="Euler's Solution")

plt.grid()

plt.xlabel("Time (Second)")

plt.ylabel("No. of parent Atoms")

plt.legend()

plt.subplot(3,1,2)

plt.plot(t_array,Y_Exact,color="red",label='Exact Equation Solution')

plt.grid()

plt.xlabel("Time (Second)")

plt.ylabel("No. of parent Atoms")

plt.legend()

plt.subplot(3,1,3)

plt.plot(t_array,solOdeint,color="blue",label='Odeint Solution')

plt.grid()

plt.xlabel("Time (Second)")

plt.ylabel("No. of parent Atoms")

plt.legend()

plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Plot Radioactive Decay ODE by Euler method, Exact solution & Inbuilt solver.")

plt.show()
```

4. To Plot Charging and Discharging of a capacitor in RC circuit ODE with DC source by Euler Method, Exact solution, Inbuilt solver.

```
#importing libraries

import matplotlib.pyplot as plt

import numpy as np

import scipy.integrate as it


def diff_equ_charging(q,t):    # Differential Equation of Charging

   return ((C*E - q)/(R*C))


def Exact_equ_charging(t):     #Solution equation of Differential Equation of Charging

   return (C*E)*(1-(np.exp(((-1)*t)/(R*C))))


def diff_equ_discharging(q,t):  # Differential Equation of Discharging

   return ((-1)*q)/(R*C)


def Exact_equ_discharging(t):     #Solution equation of Differential Equation of Discharging

   return ((C*E)*(np.exp(((-1)*t)/(R*C))))


print("\n\n\tMehendi Hasan\n\n\t2230248\n\nRC Circuit Charging and Discharging of Capacitor\n\n")

print("Capacitance is in Farad, resistance is in ohm,time is in second,charge in coulomb,voltage in volts.\n\n")


#taking inputs from user for the terms envoled in equations

C=float(input("Enter Capacitance of Capacitor: "))

E=float(input("Enter EMF of Battery: "))

R=float(input("Enter Resistance of Resistor: "))

t=float(input("Enter time instant at which charge on capacitor to be calculated: "))


h=0.1      #Step size

Qmax=C*E       #max value of charge on capacitor

t_array=np.arange(0,t+h,h)      #initializing time array(independent Variable)


#Charging of Capacitor

Y_diff_charging=np.zeros(len(t_array))      #Initializing array for values of dependent variable(Euler's method)

Y_Exact_charging=np.zeros(len(t_array))     #Initializing array for values of dependent variable(Solution equation)

Y_Exact_charging[0] = Y_diff_charging[0] = 0    #Initial values of dependent variable Y at dependent variable t=0

for i in range(len(t_array)-1):             #updating values of dependent variable

   Y_diff_charging[i+1]=Y_diff_charging[i] + h*(diff_equ_charging(Y_diff_charging[i],t_array[i]))  #Euler's Method

Y_Exact_charging=Exact_equ_charging(t_array)  #Solution Equation

solOdeintCharging=it.odeint(diff_equ_charging,Y_diff_charging[0],t_array)      #Odeint solution


#Discharging of Capacitor

Y_diff_discharging=np.zeros(len(t_array))   #Initializing array for values of dependent variable(Euler's method)

Y_Exact_discharging=np.zeros(len(t_array))  #Initializing array for values of dependent variable(Solution equation)

Y_Exact_discharging[0] = Y_diff_discharging[0] = Qmax  #Initial values of dependent variable Y at independent variable t=0
```

```
for i in range(len(t_array)-1):    #updating values of dependent variable
    Y_diff_discharging[i+1]=Y_diff_discharging[i] + h*(diff_equ_discharging(Y_diff_discharging[i],t_array[i]))    #Euler's Method
Y_Exact_discharging=Exact_equ_discharging(t_array)    #Solution Equation
solOdeintDischarging=it.odeint(diff_equ_discharging,Y_diff_discharging[0],t_array)    #Odeint solution

#ploting all the values of dependent variable with respect to independent variable
plt.subplot(3,2,2)
plt.plot(t_array,Y_diff_charging,color='blue',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Euler's Solution of Charging")
plt.legend()
plt.subplot(3,2,4)
plt.plot(t_array,Y_Exact_charging,color='red',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Exact Equation of Charging")
plt.legend()
plt.subplot(3,2,1)
plt.plot(t_array,Y_diff_discharging,color='orange',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Euler's Solution of Discharging")
plt.legend()
plt.subplot(3,2,3)
plt.plot(t_array,Y_Exact_discharging,color='green',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Exact Equation of Discharging")
plt.legend()
plt.subplot(3,2,6)
plt.plot(t_array,solOdeintCharging,color='orange',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Odeint Solution of Charging")
plt.legend()
plt.subplot(3,2,5)
plt.plot(t_array,solOdeintDischarging,color='red',label="Charge")
plt.xlabel("Time")
plt.ylabel("Charge at Capacitor")
plt.title("Odeint Solution of Discharging")
plt.legend()
plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Plot Charging and Discharging of a capacitor in RC circuit ODE with DC source by Euler Method, Exact solution, Inbuilt solver")

plt.show()
```

5. To Plot Current in RC circuit and potential ODE with DC source by Euler Method, Exact solution, Inbuilt solver.

```
#importing libraries to be used

import matplotlib.pyplot as plt

import numpy as np

import scipy.integrate as it


class RC:                  # Created a class of RC Circuit which have multiple Functions

   def current(I,t):       # Current v/s time graph using Euler's Method

       return ((-1)*(I))/(R*C)

   def current_exact(t):       # Current v/s time graph by ploting the solution equation of ODE

       return I0*(np.exp((-1)*t/(R*C)))

   def Vr(Vr,t):              # Voltage across resistor v/s time graph using Euler's Method

       return -Vr/(R*C)

   def VrExact(t):         # Voltage across resistor v/s time graph by ploting the solution equation of ODE

       return    V*(np.exp((-t)/(R*C)))

   def Vc(Vc,t):              # Voltage across capacitor v/s time graph using Euler's Method

       return (1/(R*C))*(V-Vc)

   def VcExact(t):         # Voltage across capacitor v/s time graph by ploting the solution equation of ODE

       return  V*(1-(np.exp((-t)/(R*C))))

print("\n\n\tMehendi Hasan\n\n\t2230248\n\nRC Circuit\n\n")

print("Capacitance is in Farad, resistance is in ohm,time is in second,charge in coulomb,voltage in volts.\n\n")


# input constant values

R=float(input('Enter the value of resistance in ohms:'))    #resistance


C=float(input('Enter the value of capacitance in farads:')) # capacitance

V=float(input('Enter the value of EMF in volts:')) # EMF

T_fin=float(input('Enter time instant at which current to be measured:'))       # time instant

h=0.001     #step size

time_array=np.arange(0,T_fin+h,h)# X-coordinate (time)


#   Current v/s time

I0=V/R      #current in circuit at t=0

yPointsCurrent=np.zeros(len(time_array))   #initializing Y-coordinates as array of zeros of lenght time array(Euler method )

yPointsCurrentExact=np.zeros(len(time_array))     #initializing Y-coordinates as array of zeros of lenght time array(solution Equation)

yPointsCurrent[0]=I0                        # initializing Initial value for euler's method
```

```
for i in range(len(time_array)-1):      # updating the array of zeros with help of euler's method and solution equation

    yPointsCurrent[i+1]=yPointsCurrent[i]+h*RC.current(yPointsCurrent[i],time_array[i])      #Euler's Method

yPointsCurrentExact=RC.current_exact(time_array)      # Solution Equation

solOdeintYPointsCurrent=it.odeint(RC.current,I0,time_array)      #odeint solution


#   Voltage across resistor v/s time

Vr0=V

yPointsVr=np.zeros(len(time_array))      #initializing Y-coordinates as array of zeros of lenght time array(Euler method )

yPointsVrExact=np.zeros(len(time_array))      #initializing Y-coordinates as array of zeros of lenght time array(solution Equation)

yPointsVr[0]=Vr0      # initializing Initial value for euler's method

for i in range(len(time_array)-1):      # updating the array of zeros with help of euler's method and solution equation

    yPointsVr[i+1]=yPointsVr[i]+h*RC.Vr(yPointsVr[i],time_array[i])      #Euler's Method

yPointsVrExact=RC.VrExact(time_array)      # Solution Equation

solOdeintYPointsVr=it.odeint(RC.Vr,Vr0,time_array)      #odeint solution


#   Voltage across capacitor v/s time

Vc0=0

yPointsVc=np.zeros(len(time_array))      #initializing Y-coordinates as array of zeros of lenght time array(Euler method )

yPointsVcExact=np.zeros(len(time_array))      #initializing Y-coordinates as array of zeros of lenght time array(solution Equation)

yPointsVc[0]=Vc0      # initializing Initial value for euler's method

for i in range(len(time_array)-1):      # updating the array of zeros with help of euler's method and solution equation

    yPointsVc[i+1]=yPointsVc[i]+h*RC.Vc(yPointsVc[i],time_array[i])      #Euler's Method

yPointsVcExact=RC.VcExact(time_array)      # Solution Equation

solOdeintYPointsVc=it.odeint(RC.Vc,Vc0,time_array)      #odeint solution


# plot of I v/s t

plt.subplot(3,2,1)

plt.plot(time_array,yPointsCurrent,color='red',label="I")

plt.xlabel('Time(s)')

plt.ylabel('Current(amps)')

plt.title("Current v/s time Euler's")

plt.grid('true')

plt.legend()

plt.subplot(3,2,2)

plt.plot(time_array,yPointsCurrentExact, color='blue',label="I")

plt.xlabel('Time(s)')

plt.ylabel('Current(amps)')

plt.title("Current v/s time Solution Equation")
```

```
plt.grid('true')
plt.legend()

# plot of Vr v/s t
plt.subplot(3,2,3)
plt.plot(time_array,yPointsVr,color='red',label="Vr")
plt.plot(time_array,yPointsVc,color='blue',label="Vc")
plt.xlabel('Time(s)')
plt.ylabel('(volts)')
plt.title("Vr and Vc v/s time Eulers ")
plt.grid('true')
plt.legend()
plt.subplot(3,2,4)
plt.plot(time_array,yPointsVrExact, color='blue',label="Vr")
plt.plot(time_array,yPointsVcExact, color='red',label="Vc")
plt.xlabel('Time(s)')
plt.ylabel("(volts)")
plt.title("Vr and Vc v/s time Solution equation")
plt.grid('true')
plt.legend()

plt.subplot(3,2,5)
plt.plot(time_array,solOdeintYPointsVr, color='blue',label="Vr")
plt.plot(time_array,solOdeintYPointsVc, color='red',label="Vc")
plt.xlabel('Time(s)')
plt.ylabel("(volts)")
plt.title("Vr and Vc v/s time Odeint Solution")
plt.grid('true')
plt.legend()

plt.subplot(3,2,6)
plt.plot(time_array,solOdeintYPointsCurrent, color='blue',label="Current")
plt.xlabel('Time(s)')
plt.ylabel("(volts)")
plt.title("Current v/s time Odeint Solution equation")
plt.grid('true')
plt.legend()
plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Plot Current in RC circuit and potential ODE with DC source by Euler Method, Exact solution, Inbuilt solver.")
plt.show()
```

6. To Plot Current in RL circuit ODE with DC source by Euler Method, Exact solution, Inbuilt solver.

```
#importing libraries to be used

import matplotlib.pyplot as plt

import numpy as np

import scipy.integrate as it


def diffEquation(i,t):

   return (V/L)-((R*i)/L)

def solEquation(i,t):

   return (V/R)*(1-(np.exp((((-1)*R)*t)/L)))


print("\n\n\tMehendi Hasan\n\n\t2230248\n\nVariation of curent with time in RL Circuit \n\n")

print("Resistance is in ohm,time is in second,Inductance in henry,voltage in volts.\n\n")


L=float(input("Enter Inductance of Inductor: "))

V=float(input("Enter EMF of Battery: "))

R=float(input("Enter Resistance of Resistor: "))

t=float(input("Enter time instant at which Current through inductor to be calculated: "))


h=0.001     #step size

time_array=np.arange(0,t+h,h)

I0=0     #current in circuit at t=0

yPointsCurrent=np.zeros(len(time_array))

yPointsCurrentExact=np.zeros(len(time_array))

yPointsCurrent[0]=I0

yPointsCurrentExact[0]=I0

for i in range(len(time_array)-1):
yPointsCurrent[i+1]=yPointsCurrent[i]+h*diffEquation(yPointsCurrent[i],time_array[i])     #Euler's Method

yPointsCurrentExact=solEquation(yPointsCurrentExact,time_array)          # Solution Equation

solOdeintYPointsCurrent=it.odeint(diffEquation,I0,time_array)          #odeint solution


# plot of I v/s t

plt.subplot(1,3,1)

plt.plot(time_array,yPointsCurrent,color='red',label="I Euler")

plt.xlabel('Time(s)')

plt.ylabel('Current(amps)')

plt.title("Current v/s time Euler's")

plt.grid('true')

plt.legend()

plt.subplot(1,3,2)

plt.plot(time_array,yPointsCurrentExact, color='blue',label="I")

plt.xlabel('Time(s)')

plt.ylabel('Current(amps)')

plt.title("Current v/s time Solution Equation")

plt.grid('true')

plt.legend()

plt.subplot(1,3,3)

plt.plot(time_array,solOdeintYPointsCurrent, color='green',label="Current")

plt.xlabel('Time(s)')

plt.ylabel("(volts)")

plt.title("Current v/s time Odeint Solution equation")

plt.grid('true')

plt.suptitle("Mehendi Hasan B.SC.(H) Physics 2230248\nTo Plot Current in RL circuit ODE with DC source by Euler Method, Exact solution, Inbuilt solver.")

plt.legend()

plt.show()
```