

Syllabus:

1. Solve first order differential equation by Euler's method
2. Solve first order differential equation by Improved Euler's method
3. Solve first order coupled equations by Euler's method
4. Solve second order coupled equations by Euler's method

Euler's method

Euler's method numerically approximates solutions of first-order ordinary differential equations (ODEs) with a given initial value.

$$\frac{dy(t)}{dt} = f(t, y(t))$$

initial value: $y(t_0) = y_0$

To get a numeric solution, we replace the derivative on the LHS with a finite difference approximation

$$\frac{dy(t)}{dt} \approx \frac{y(t+h) - y(t)}{h} \quad \text{h is taken small for better result}$$

then
$$y(t+h) \approx y(t) + h \frac{dy(t)}{dt}$$

or,
$$y(t+h) \approx y(t) + hf(t, y(t))$$

The iterative solution rule is then:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

where h is called step size, the most relevant parameter for accuracy of the solution.

Example

$$\frac{dy}{dx} = x + y + xy, \text{ initial cond: } x_0 = 0, y_0 = 1 \rightarrow y(0) = 1$$

Use Euler's Method with a step size of $h=0.025$ to find approximate values

of the solution at $y = 0.05, 0.075, 0.1, 0.4$ and Compare them to the exact values of the solution at these points.

Algorithm:

Define function : $f(x, y) = x + y + xy$

Initial condition : $x_0 = 0, y_0 = 1$

Step 1 : $h = 0.025$

Initial cond/values		$f(x_0, y_0) = x_0 + y_0 + x_0y_0$	$y_{(x_0+h)} = y_0 + fh$
x_0	y_0		
0	1	$0+1+0=1$	$1+1*.025=1.025$
0.025	1.025	$0.025+1.025+1.025*.025= 1.075$	$1.025+1.075*.025=1.051$
0.050	1.051	$.050+1.051+.050*1.051=1.153$	$1.051+1.153*.025=1.079$
0.075	1.079	$.075+1.079+.075*1.079=1.234$	$1.079+1.234*.025=1.109$
0.1	1.109		

Ex:1 / Euler1

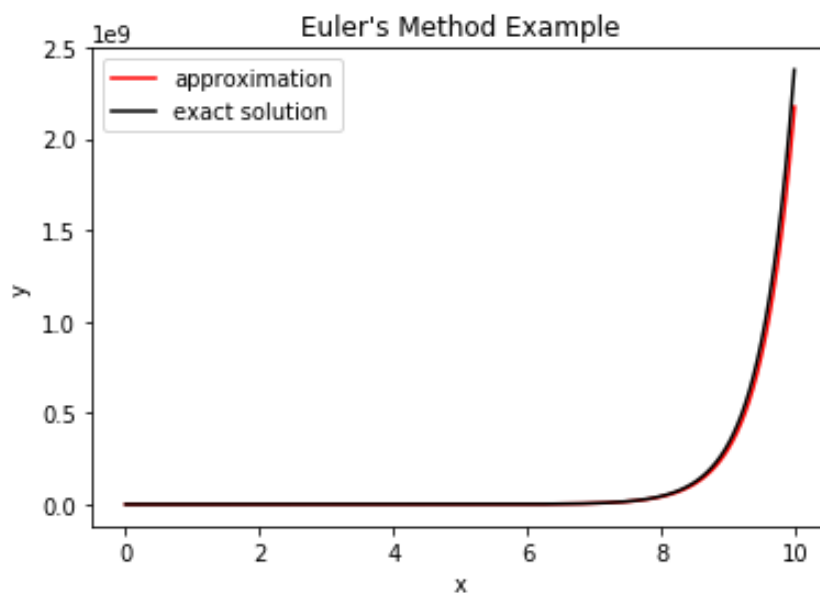
$dy / dx = 2y$ calculate y with initial cond. at $x=0, y=5$

```
import numpy as np
import matplotlib.pyplot as plt
def f(pr,x,y):
    #parameter set None from main
    return 2*y
x = 0.0
y = 5.0
dx=0.005
X = 10
# Function for euler formula
#def euler(f, pr, x, y, h, X ):
xx,yy,ddydx=[],[],[]
while abs(x) < abs(X):
    dydx=f(None,x,y)
    xx.append(x); yy.append(y); ddydx.append(dydx)
```

```

y+=dydx*dx
x+=dx
#plot
plt.plot(xx,yy,'r', label='approximation' )
x1=np.arange(0,10,0.01)
plt.plot(x1,5.0*np.exp(2*x1),'k-',label='exact solution')
plt.title( "Euler's Method Example" )
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
# Printing approximation
print('h = "%.3f"% h)
print('Approximate solution at X=10 is"' %f" %y)

```



$h = 0.005$

Approximate solution at X=10 is 2196431025.250482

Improved Euler method:

$$y_{n+1} = y_n + \frac{h}{2} [f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n))]$$

Ex:2 / ImEuler1

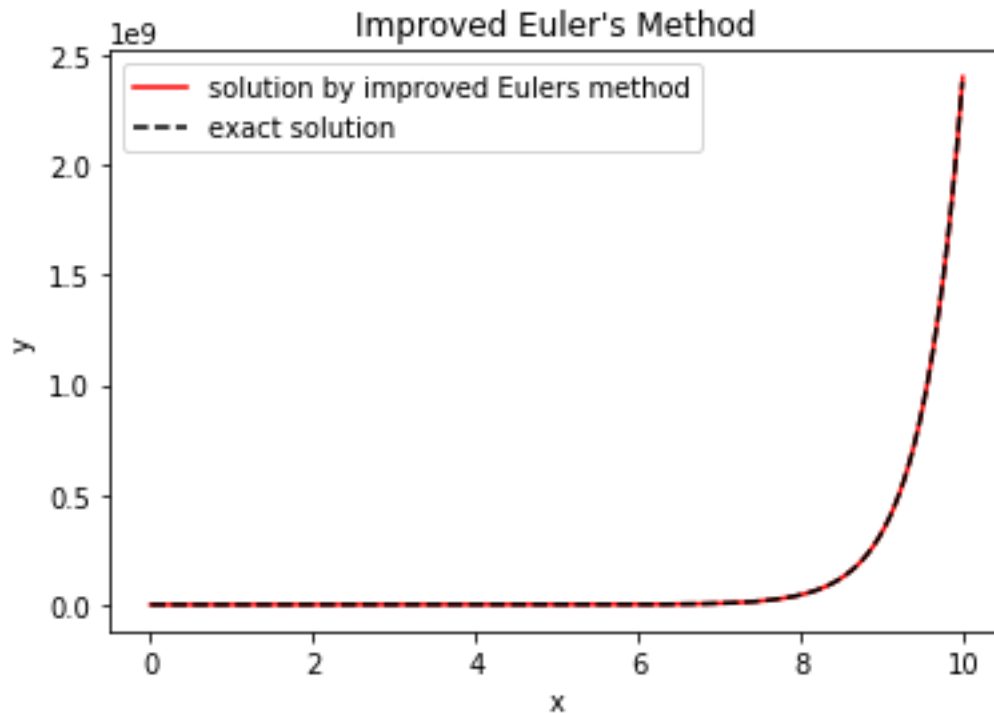
$dy / dx = 2y$ calculate y with initial cond at $x=0, y=5$

#improved Euler method

```
import numpy as np
import matplotlib.pyplot as plt
def f(x,y):
    #parameter set None from main
    return 2*y
x = 0.0
y = 5.0
dx=0.005
X = 10
xx,yy,ddydx=[],[],[]
while abs(x) < abs(X):
    dydx=0.5*(f(x,y)+f(x+dx,y+dx*f(x,y)))
    xx.append(x); yy.append(y); ddydx.append(dydx)
    y+=dydx*dx
    x+=dx

#plot
plt.plot(xx,yy,'r', label='solution by improved Eulers method' )
x1=np.arange(0.0,10.0,0.01)
plt.plot(x1,5.0*np.exp(2*x1),'k--',label='exact solution')
plt.title( "Improved Euler's Method" )
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()

# Printing approximation
print('h = "%.3f"% dx)
print('Approximate solution at X=10 is" %f" %y)
```



$h = 0.005$

Approximate solution at $X=10$ is 2425023541.445030

Ex:3 /EULER2

$dy / dx = x-t^2+1$, $0 \leq t \leq 2$, calculate y with initial cond at $x=0.5$, $t=0$

#For reference, the analytical solution is $x=(t+1)^2-0.5\exp(t)$

#calculate error= $\varepsilon = \sum_1^N |x(t_i) - x_i|$

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def f(pr,t,x):
```

```
    #parameter set None from main
```

```
    return x-t**2+1
```

```
t = 0.0
```

```
x = 0.5
```

```
h=(2.0-0.0)/1000.0
```

```
T = 2.0
```

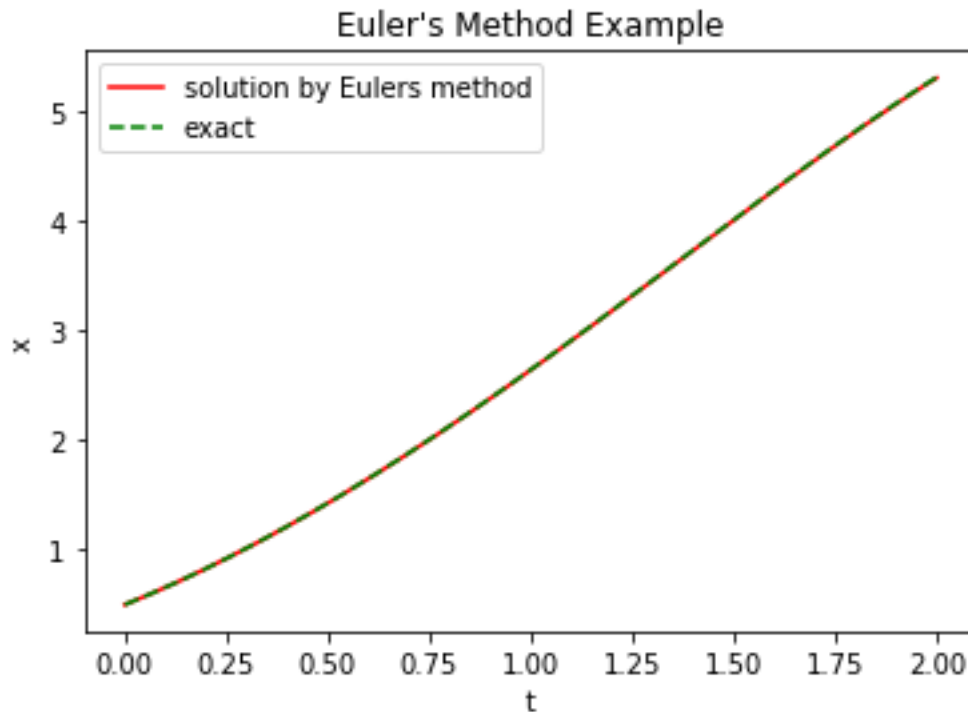
```
tt,xx,ddxdt=[],[],[]
```

```
while abs(t) < abs(T):
```

```
dxdt=f(None,t,x)
tt.append(t); xx.append(x); ddxdt.append(dxdt)
x+=dxdt*h
t+=h

#plot
plt.plot(tt,xx,'r', label='solution by Eulers method' )
t1=np.arange(0.0,2.0,0.001)
plt.plot(t1,(t1+1.0)**2-0.5*np.exp(t1),'g--',label='exact')
plt.title( "Euler's Method Example" )
plt.xlabel('t')
plt.ylabel('x')
plt.legend()
plt.show()

# Printing approximation
print('h = "%.3f"% h)
print('Approximate solution at X=2 is" %f" %x)
```



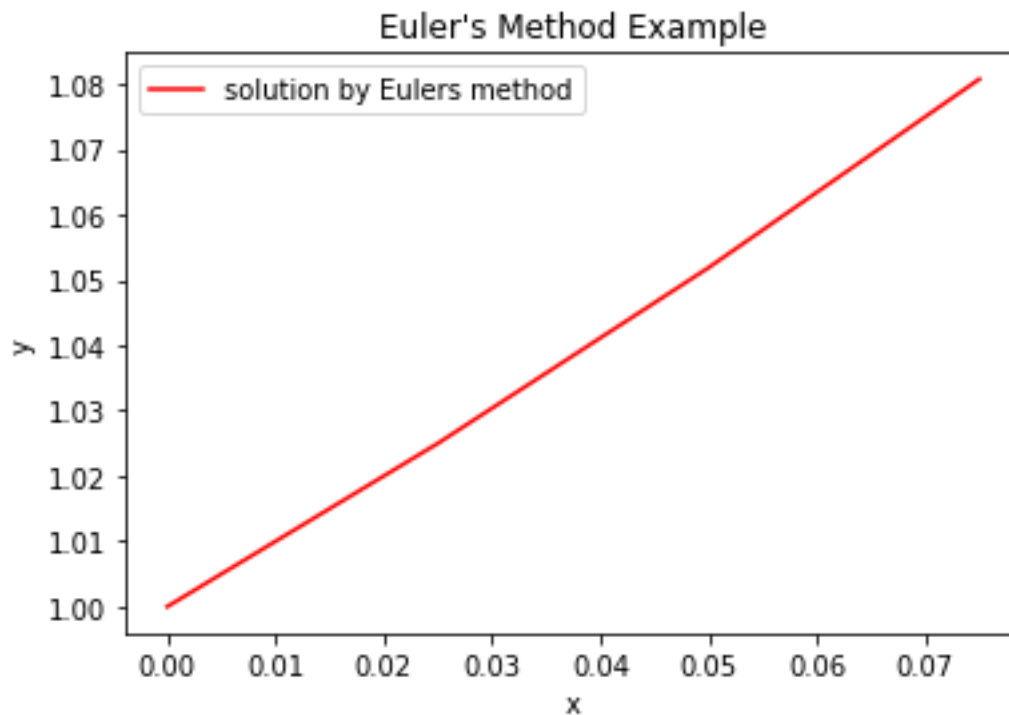
$h = 0.002$

Approximate solution at X=2 is 5.300095

Ex:4 / Euler3**#dy / dx =x+y+xy calculate y at x=0.1 with initial cond at x=0, y=1**

```
import numpy as np
import matplotlib.pyplot as plt

def f(pr,x,y):
    #parameter set None from main
    return x+y+x*y
x = 0.0
y = 1.0
h=0.025
X = 0.1
xx,yy,ddydx=[],[],[]
while abs(x) < abs(X):
    dydx=f(None,x,y)
    xx.append(x); yy.append(y); ddydx.append(dydx)
    y+=dydx*h
    x+=h
#plot
plt.plot(xx,yy,'r', label='solution by Eulers method' )
#x1=np.arange(0.0,1.0,0.01)
#plt.plot(x1,,'k-',label='exact solution')
plt.title( "Euler's Method Example" )
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
# Printing approximation
print('h = "%.3f"% h)
print('Approximate solution at X=0.1 is" %f" %y)
```



$h = 0.025$

Approximate solution at $X=0.1$ is 1.111673

Ex :5 / Euler4

**#dq / dt = -q/RC calculate q at t=10 with initial cond at t=0,
q=CE**

discharging of capacitor

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def f(RC,t,q):
```

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

```
R,C =100,1.5e-2
```

```
RC=R*C
```

```
E= 10.0
```

```
t = 0.0
```

```
q = E*C
```

```
h=0.01
```

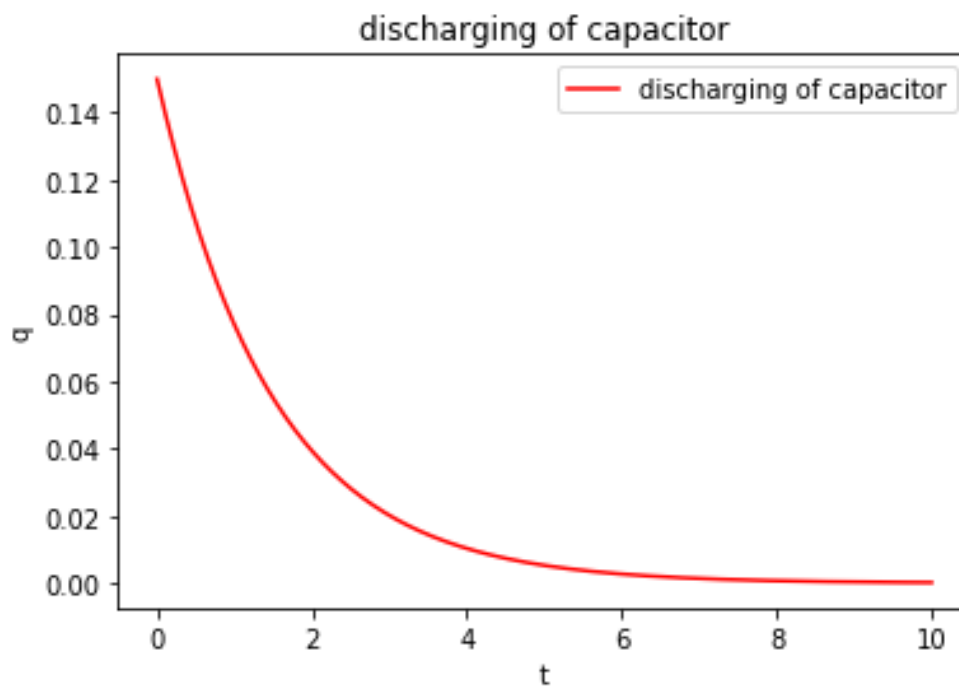
```
T = 10
```



```
tt,qq,ddqdt=[],[],[]
while abs(t) < abs(T):
    dqdt=f(RC,t,q)
    tt.append(t); qq.append(q); ddqdt.append(dqdt)
    q+=dqdt*h
    t+=h

#plot
plt.plot(tt,qq,'r', label='discharging of capacitor' )
plt.title( "discharging of capacitor" )
plt.xlabel('t')
plt.ylabel('q')
plt.legend()
plt.show()

# Printing approximation
print('h = "%.3f"% h)
print('Approximate solution at T=10 is"' %f" %q)
```



h = 0.010

Approximate solution at T=10 is 0.000185

Ex :6/ Euler5

#dq / dt =-q/RC calculate q at t=10 with initial cond at t=0, q=CE/discharging a capacitor

#dq / dt =E/R-q/RC calculate q at t=10 with initial cond at t=0, q=CE/charging a capacitor

```
import numpy as np
import matplotlib.pyplot as plt

def f(RC,t,q):
    return -q/(R*C)

def f2(RC,t,q):
    return (E/R)-q/(R*C)

R,C =100,1.5e-2
E= 10.0
RC=R*C
t = 0.0
q = 0.0
#for charging
h1=0.01
T1 = 10
tt,qq,ddqdt=[],[],[]
while abs(t) < abs(T1):
    dqdt=f2(RC,t,q)
    tt.append(t); qq.append(q); ddqdt.append(dqdt)
    q+=dqdt*h1
    t+=h1
plt.plot(tt,qq,'b--', label='charging of a capacitor' )
print('h1 = ""%.3f"" h1)
print('Approximate value of charge at T=10 during CHARGING is"" %f" %q)

#for discharging
h=0.01
```

T = 10

t = 0.0

q = E*C

```
tt,qq,ddqdt=[],[],[]
```

```
while abs(t) < abs(T):
```

```
    dqdt=f(RC,t,q)
```

```
    tt.append(t); qq.append(q); ddqdt.append(dqdt)
```

```
    q+=dqdt*h
```

```
    t+=h
```

```
plt.plot(tt,qq,'r', label='discharging of a capacitor' )
```

```
print('h = "%.3f"% h)
```

```
print('Approximate value of charge at T=10 during DISCHARGING is" %f" %q)
```

```
#plot
```

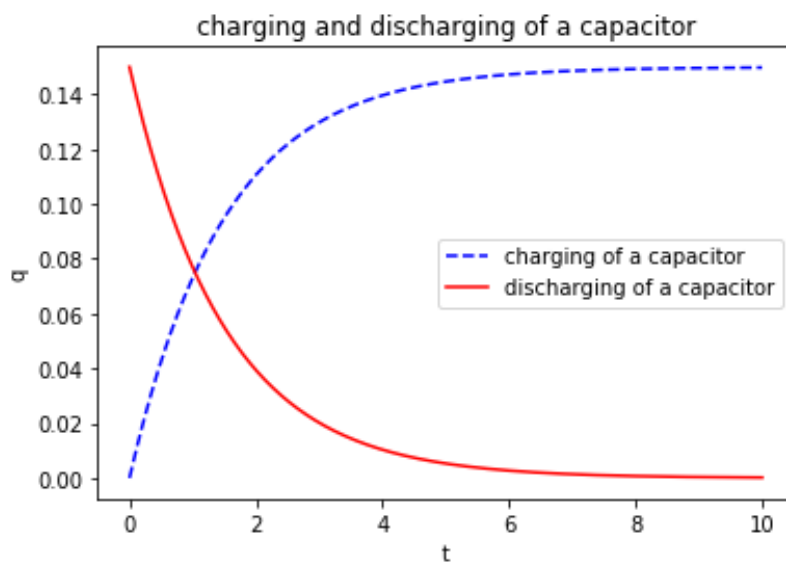
```
plt.title( "charging and discharging of a capacitor" )
```

```
plt.xlabel('t')
```

```
plt.ylabel('q')
```

```
plt.legend()
```

```
plt.show()
```



h1 = 0.010

Approximate value of charge at T=10 during CHARGING is 0.149815

h = 0.010

Approximate value of charge at T=10 during DISCHARGING is **0.000185**

Euler Method for coupled First order differential equations

$$\frac{dx}{dt} = f_1(t, x, y)$$

$$\frac{dy}{dt} = f_2(t, x, y)$$

x, y : dependent variable, t: independent variable.

Solve for x and y , given at t=0, x=x₀ and y=y₀?

Choose h.

Start from initial cond t=0 and use these eqs to get final result.

$$x(t + h) = x(t) + hf_1(t, x, y)$$

$$y(t + h) = y(t) + hf_2(t, x, y)$$

Ex:7 / Eulerprojectile

#A particle is projected with initial velocity u making an angle theta= pi/3

with the horizontal.dx/dt=u cos(theta), dy/dt= u sin(theta)-gt,

initial condition: at t=0, x=0,y=0. Take u=1, g=1

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def f1(ux,t,x):
```

```
    return ux
```

```
def f2(uy,t,y):
```

```
    return uy-g*t
```

```
u = 1.0
th = np.pi/3.0
ux = u*np.cos(th)
uy = u*np.sin(th)

t = 0.0
x = 0.0
y = 0.0
g = 1.0

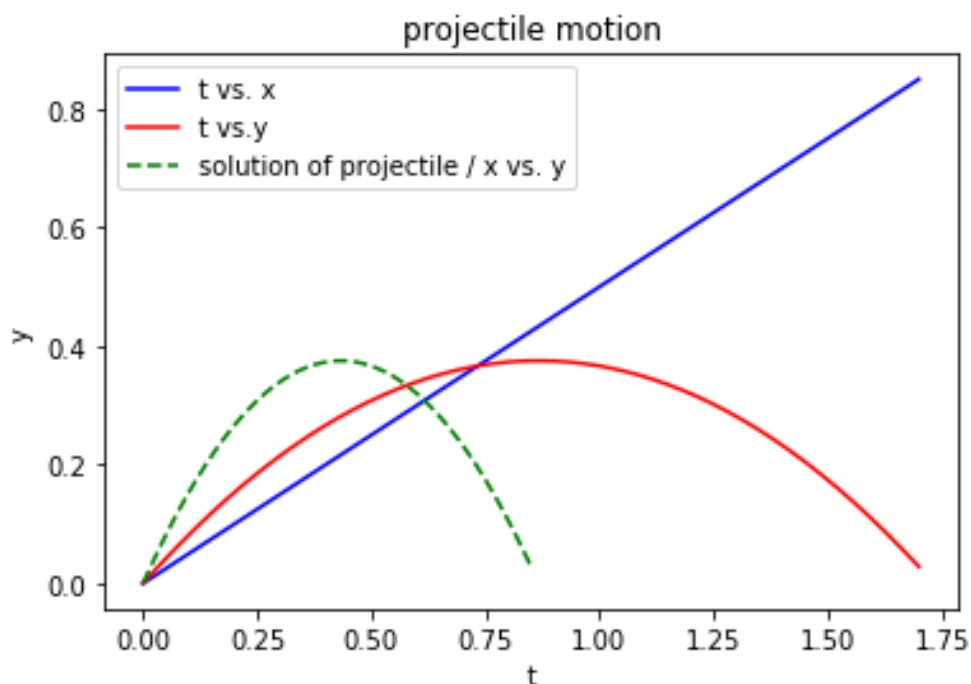
dt=0.001
T = 1.7

xx,tt,ddxdt=[],[],[]
while abs(t) < abs(T):
    dxdt=f1(ux,t,x)
    tt.append(t); xx.append(x); ddxdt.append(dxdt)
    x+=dxdt*dt
    t+=dt

t = 0.0
yy,tt,ddydt=[],[],[]
while abs(t) < abs(T):
    dydt=f2(uy,t,y)
    tt.append(t); yy.append(y); ddydt.append(dydt)
    y+=dydt*dt
    t+=dt

#plot
plt.plot(tt,xx,'b', label='t vs. x')
```

```
plt.plot(tt,yy,'r', label='t vs.y' )
plt.plot(xx,yy,'g--', label='solution of projectile / x vs. y')
plt.title( "projectile motion" )
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.show()
```



EX:8 /LCR

LCR circuit $dI_1/dt=6-4I_1+3I_2$ and $dI_2/dt=3.6-2.4I_1+1.6I_2$

#initial conditions at $t=0$, $I_1=I_2=0$

#exact solutions: $I_1=-3.375\exp(-2t)+1.875\exp(-0.4t)+1.5$

$I_2=-2.25\exp(-2t)+2.25\exp(-0.4t)$

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def f1(pr1,t,i1,i2):
    return 6.0-4.0*i1+3.0*i2

def f2(pr2,t,i1,i2):
    return 3.6-2.4*i1+1.6*i2

t=0.0
i1=0.0
i2=0.0

dt=0.1
T=10.0

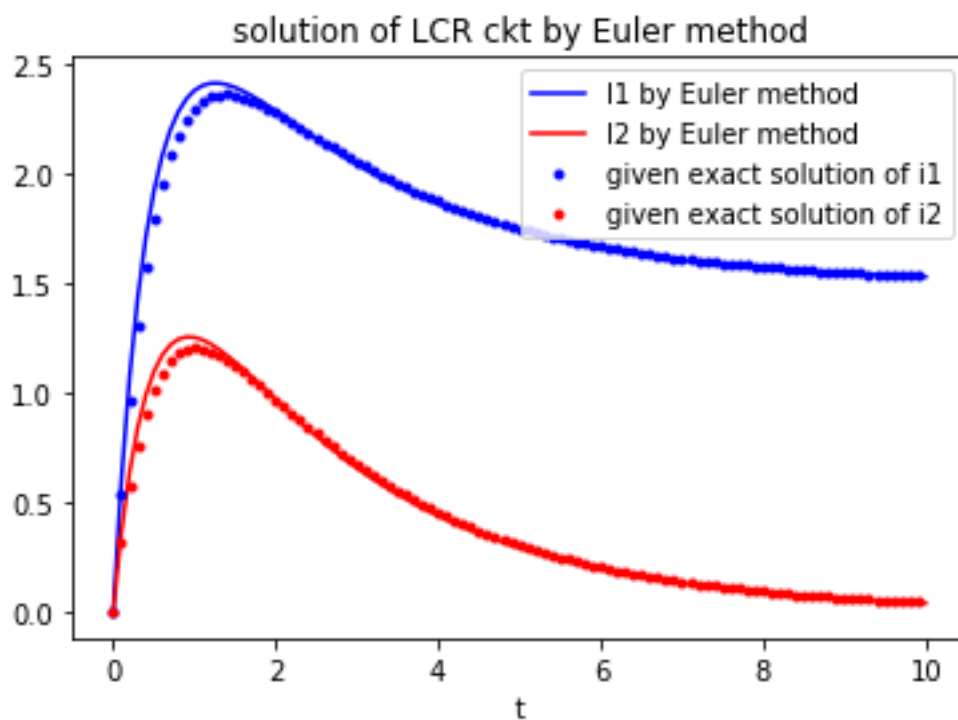
tt = []
ii1, ddi1dt = [], []
ii2, ddi2dt = [], []

while abs(t)<abs(T):
    di1dt = f1(None,t,i1,i2)
    tt.append(t)
    ii1.append(i1); ddi1dt.append(di1dt)
    di2dt = f2(None,t,i1,i2)
    ii2.append(i2); ddi2dt.append(di2dt)
    i1+=di1dt*dt
    i2+=di2dt*dt
    t+=dt
    print("%.2f"%i1,"%.2f"%i2,"%.1f"%t)

plt.plot(tt,ii1,'b',label='I1 by Euler method')
plt.plot(tt,ii2,'r',label='I2 by Euler method')
```

```
plt.title('solution of LCR ckt by Euler method')
plt.xlabel('t')
plt.legend()

tt=np.arange(0.0,10,0.1)
i1=-3.375*np.exp(-2*tt)+1.875*np.exp(-0.4*tt)+1.5
i2=-2.25*np.exp(-2*tt)+2.25*np.exp(-0.4*tt)
plt.plot(tt,i1,'b.',label='given exact solution of i1')
plt.plot(tt,i2,'r.',label='given exact solution of i2')
plt.legend()
plt.show()
```



HW.

1. Solve for half wave rectifier and full wave rectifier
2. Try to calculate Error

Solve 2nd order differential equation by Euler method

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right)$$

First split it into 2 equations.

$$\frac{dy'}{dx} = f(x, y, y') \quad \text{and} \quad \frac{dy}{dx} = y' \quad \text{and then solve } y' \quad \text{and then for } y.$$

d2x/dt2=a, Newton's law of motion

import numpy as np

import matplotlib.pyplot as plt

Ex : 9 / newton's Law

#Solve d2x/dt2=a, where a=1.0

```
def f(a,t,x,dxdt):
```

```
    return a
```

```
a=1.0
```

```
t,x,dxdt=0.0,0.0,0.0
```

```
dt=0.1
```

```
T=2.0
```

```
tt, xx, ddxdt, dd2xdt2=[], [], [], []
```

```
while abs(t)< abs(T):
```

```
    d2xdt2 = f(a,t,x,dxdt)
```

```
    tt.append(t); xx.append(x); ddxdt.append(dxdt); dd2xdt2.append(d2xdt2)
```

```
    dxdt+=d2xdt2*dt
```

```
    x+=dxdt*dt
```

```
    t+=dt
```

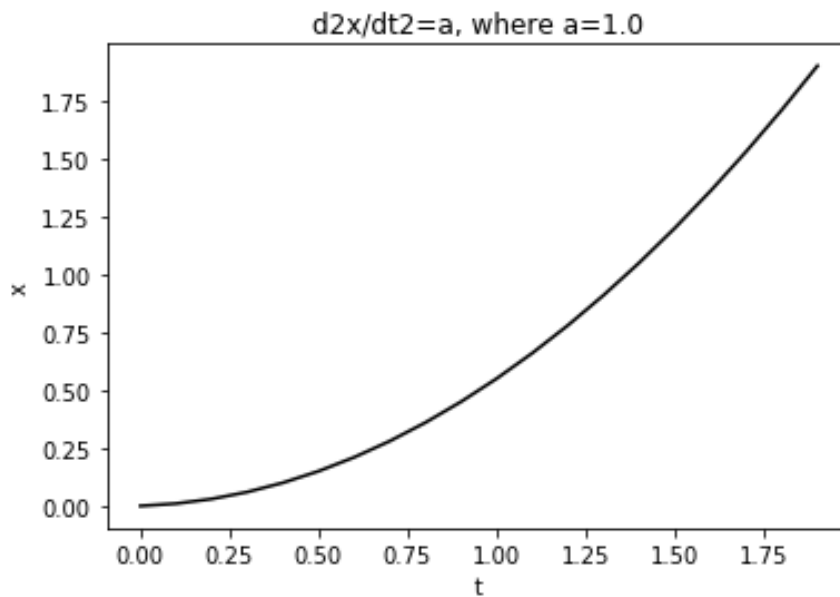
```
plt.plot(tt,xx, 'k')
```

```
plt.title('d2x/dt2=a, where a=1.0')
```

```
plt.xlabel('t')
```

```
plt.ylabel('x')
```

```
plt.show()
```



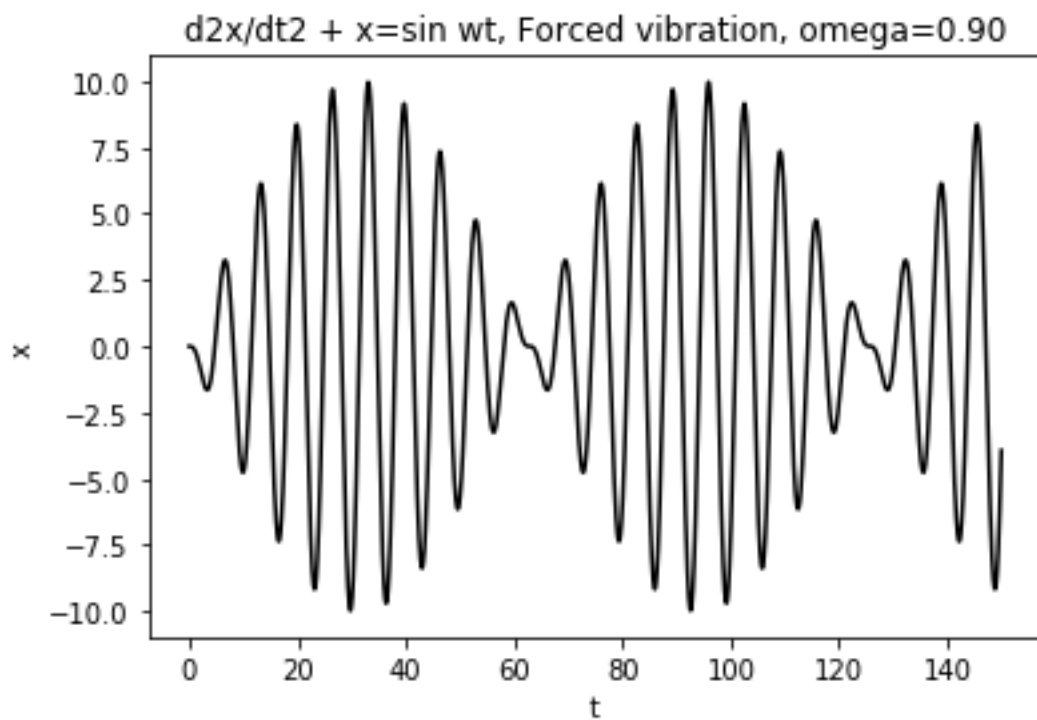
Ex:10 / Forcedvibration

$d^2x/dt^2 = -x - \sin wt$, forced vibration, $m=1$, $k=1$

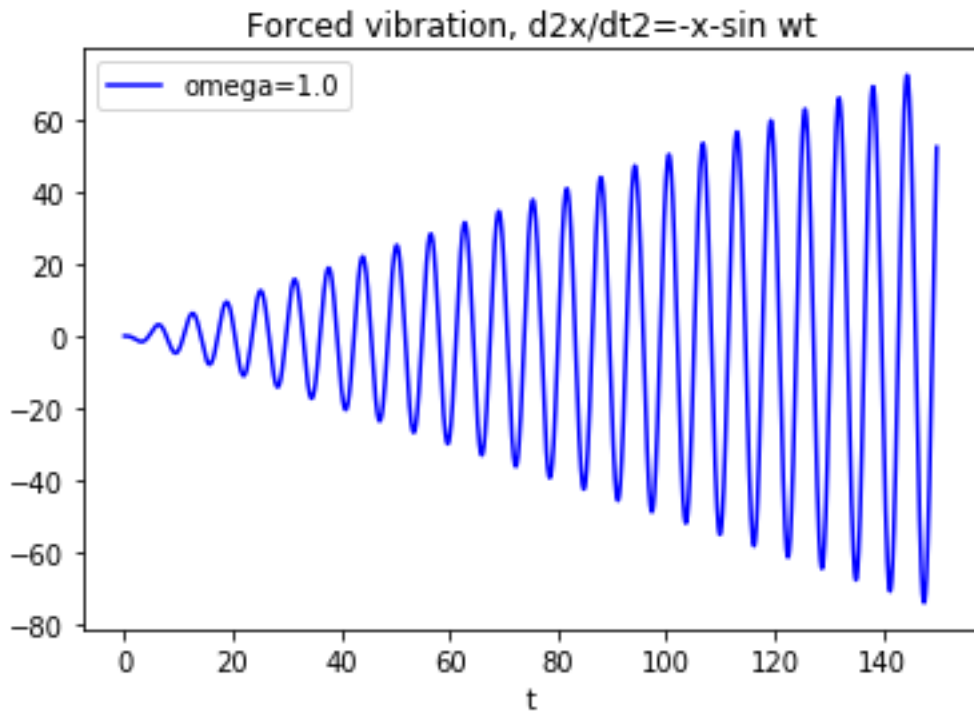
```
import numpy as np
import matplotlib.pyplot as plt
#to solve dx'/dt=-x-sinwt, x'=dx/dt
def f(omega,t,x,dxdt):
    return -x-np.sin(omega*t)

t,x,dxdt=0.0,0.0,0.0
dt=0.01
T=150.0
tt, xx, ddxdt = [], [], []
dd2xdt2=[]
omega=0.9
while abs(t)< abs(T):
    d2xdt2 = f(omega,t,x,dxdt)
    tt.append(t); xx.append(x); ddxdt.append(dxdt); dd2xdt2.append(d2xdt2)
```

```
dxdt+=d2xdt2*dt
x+=dxdt*dt
t+=dt
plt.plot(tt,xx,'b', label='omega=0.9' )
plt.title( "Forced vibration, d2x/dt2=-x-sin wt" )
plt.xlabel('t')
plt.legend()
plt.show()
```



Change omega to 1.0



Change omega to 1.25

