

Source Codes

Arun Prasaad Gunasekaran

August 11, 2020

Hello!

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  2 07:34:25 2015

@author: arun
"""

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

def dampfunc(x, t):
    F0 = 50.0
    omg = 1.0
    m = 2.0
    c = 1.0
    k = 20
    x0 = x[0] # Displacement
    x1 = x[1] # Velocity
    x2 = F0*np.sin(omg*t)/m - c*x1/m - k*x0/m # Acceleration
    return x1, x2

F0 = 50.0
omg = 1.0
m = 2.0
c = 1.0
k = 20

init = [0.0, 0.0] # Initial conditions
t = np.linspace(0.0, 10.0, 101)

x = odeint(dampfunc, init, t) # This is the vital part!

mass = x

acc = F0*np.sin(omg*t)/m - c*x[:,1]/m - k*x[:,0]/m

plt.plot(t, x[:,0], 'b', label='$x$ (m)$')
plt.plot(t, x[:,1], 'r', label='$\dot{x}$ (m/s)$')
plt.plot(t, acc, 'g', label='$\ddot{x}$ (m/s^2)$')
plt.axis([0, 10, -10, 10])
plt.xticks(np.linspace(0,10,11))
plt.yticks(np.linspace(-10,10,11))
plt.grid('on')
plt.legend(framealpha=0.5)
plt.title('Solution to a Mass Spring Damper System')
plt.suptitle('Session 3, Demos')
plt.xlabel('Time in s')
```

```
plt.ylabel('Magnitudes')
plt.savefig('Mass_Spring_Damper.jpg', format='jpg', dpi=200)
```

```
1 print "Hello World!"
# -*- coding: utf-8 -*-
3 """
Created on Wed Sep  2 07:34:25 2015
5
@author: arun
7 """
9 import numpy as np
from scipy.integrate import odeint
11 import matplotlib.pyplot as plt
13 def dampfunc(x, t):
    F0 = 50.0
15     omg = 1.0
    m = 2.0
17     c = 1.0
    k = 20
19     x0 = x[0] # Displacement
    x1 = x[1] # Velocity
21     x2 = F0*np.sin(omg*t)/m - c*x1/m - k*x0/m # Acceleration
    return x1, x2
23
F0 = 50.0
25 omg = 1.0
m = 2.0
27 c = 1.0
k = 20
29
init = [0.0, 0.0] # Initial conditions
31 t = np.linspace(0.0, 10.0, 101)
33
x = odeint(dampfunc, init, t) # This is the vital part!
35
mass = x
37
acc = F0*np.sin(omg*t)/m - c*x[:,1]/m - k*x[:,0]/m
39
plt.plot(t, x[:,0], 'b', label='$x$ (m)$')
plt.plot(t, x[:,1], 'r', label='$\dot{x}$ (m/s)$')
41 plt.plot(t, acc, 'g', label='$\ddot{x}$ (m/s^2)$')
plt.axis([0, 10, -10, 10])
43 plt.xticks(np.linspace(0,10,11))
plt.yticks(np.linspace(-10,10,11))
45 plt.grid('on')
plt.legend(framealpha=0.5)
47 plt.title('Solution to a Mass Spring Damper System')
plt.suptitle('Session 3, Demos')
49 plt.xlabel('Time in s')
plt.ylabel('Magnitudes')
51 plt.savefig('Mass_Spring_Damper.jpg', format='jpg', dpi=200)
```

```
1 # -*- coding: utf-8 -*-
"""
3 Created on Wed Sep  2 07:34:25 2015
5
@author: arun
7 """
9 import numpy as np
from scipy.integrate import odeint
11 import matplotlib.pyplot as plt
13 def dampfunc(x, t):
    F0 = 50.0
15     omg = 1.0
    m = 2.0
17     c = 1.0
    k = 20
    x0 = x[0] # Displacement
```

```

19     x1 = x[1] # Velocity
    x2 = F0*np.sin(omg*t)/m - c*x1/m - k*x0/m # Acceleration
21     return x1, x2

23 F0 = 50.0
    omg = 1.0
25 m = 2.0
    c = 1.0
27 k = 20

29 init = [0.0, 0.0] # Initial conditions
    t = np.linspace(0.0, 10.0, 101)
31
    x = odeint(dampfunc, init, t) # This is the vital part!
33
    mass = x
35
    acc = F0*np.sin(omg*t)/m - c*x[:,1]/m - k*x[:,0]/m
37
    plt.plot(t, x[:,0], 'b', label='$x$ (m)$')
39    plt.plot(t, x[:,1], 'r', label='$\dot{x}$ (m/s)$')
    plt.plot(t, acc, 'g', label='$\ddot{x}$ (m/s^2)$')
41    plt.axis([0, 10, -10, 10])
    plt.xticks(np.linspace(0,10,11))
43    plt.yticks(np.linspace(-10,10,11))
    plt.grid('on')
45    plt.legend(framealpha=0.5)
    plt.title('Solution to a Mass Spring Damper System')
47    plt.suptitle('Session 3, Demos')
    plt.xlabel('Time in s')
49    plt.ylabel('Magnitudes')
    plt.savefig('Mass-Spring-Damper.jpg', format='jpg', dpi=200)

```

ode_solve1.py