Python Assignment -6

Swapnil Basak

EE11B122, IIT Madras

Abstract

In this assignment, we will look at how to analyse “Linear Time-invariant Systems” with numerical tools in Python.

1 Spring Behaviour

Solve for the time response of a spring satisfying $x + x \ddot{=} 0$with $x(0) = 0.1$ and $x = 0$ for t going from $0 - 20$ seconds.

1.1 Code

1 ⟨*code* 1⟩≡

```
#! usr/bin/python
from scipy import *
from scipy import signal
from matplotlib.pyplot import *

# The transfer function of the first spring
den=poly1d([1,0,1])
num=poly1d([0.1,0])
spring=signal.lti(num,den)
time=linspace(-20,20,2000)
response=spring.impulse(T=time)[1]

# Plotting the behaviour
subplot(221)
title("Spring behaviour")
ylabel(r'$x-displacement$')
plot(time,response,'r+')
```

2 Coupled Spring behaviour

Solve for a coupled spring problem:

$$\ddot{x} + (x - y) = 0$$

$$\ddot{y} + 2(y - x) = 0$$

where the initial conditions are $x(0) = 0, \dot{x}(0) = \dot{y}(0) = y(0) = 0$

2.1 Python Code

2 $\langle$*runsim* 2$\rangle\equiv$

```
# Transfer function of the Coupled Spring - Spring 1
x_num=poly1d([0.5,0,1,])
x_den=poly1d([0.5,0,1.5,0])
spring=signal.lti(x_num,x_den)
response = spring.impulse(T=time)[1]

# Plotting the behaviour
subplot(223)
title("Coupled Spring behaviour - X")
xlabel(r'$time$')
ylabel(r'$x-displacement$')
plot(time,response,'r+')

# Transfer function of the Coupled Spring - Spring 2
y_num=poly1d([1,0])
y_den=poly1d([0.5,0,1.5,0,0])
spring=signal.lti(y_num,y_den)
response = spring.impulse(T=time)[1]

# Plotting the behaviour
subplot(224)
title("Coupled Spring behaviour - Y")
xlabel(r'$time$')
ylabel(r'$Y-displacement$')
plot(time,response,'r+')
```

3 Curcuit Analysis

Obtain the magnitude and phase response of the Transfer function of the given two-network with $L = 10^{-3}, R_1 = 10, R_2 = 100$

Also, plot the properties of the transfer function.

3.1 Python Code

3 ⟨*fft* 3⟩≡

```
# Components of the circuit
L=1e-3
R1=10
R2=100
omega=logspace(3,9,61).reshape((61,1))

# Transfer Function
H=1/((R1/R2)+1+(1j*omega*L/R2))

# Plotting the magnitude of H
figure(3)
subplot(211)
loglog(omega,abs(H),’ro’)
title("frequency response")
xlabel(r’$\omega$’)
ylabel(r’$|H|$’)

# Plotting the phase of H
subplot(212)
semilogx(omega,180*angle(H)/pi,’ro’)
xlabel(r"$\omega$")
ylabel(r"Arg$(H)$")

# Simlulating Step response
den=poly1d([1*L,R1+R2])
print "Pole(s) are ", roots(den)
sys=signal.lti(R2,den)
time=linspace(0,1e-4,1000)
step_response = sys.step(T=time)[1]

# Plotting the Step response
figure(5)
plot(time, step_response)
xlabel(r’$t$’)
title(’Step response’)
t=linspace(0,50,513)*1e-6;t=t[0:-1]
vi=array(cos(1e5*t))

# Analysing circuit in fft domain
Vi=fft(vi)
wmax=1/(t[1]-t[0])
frq=linspace(-0.5,0.5,513)*wmax
frq=frq[0:-1]
Vi=hstack([Vi[256:512],Vi[0:256]])
omega=2*pi*frq
print L, R1, R2
den=poly1d([1j*L,R1+R2])
H=100.0/den(omega)
Vo=array(Vi)*array(H)
Vo=hstack([Vo[256:512],Vo[0:256]])
vo=ifft(Vo)
```

```
# Plotting the Frequency Response
figure(6)
subplot(211)
plot(frq,abs(Vo),'k')
title("Output Frequency Response")
xlabel(r"$\omega$")
ylabel(r"$|Vo|$")
subplot(212)
plot(frq,angle(Vo)*180/pi,'k')
xlabel(r"$\omega$")
ylabel(r"$arg(Vo)$")

# Plotting the output and input
figure(7)
plot(t,vo,'g')
plot(t,vi,'b')
legend(('Input','Output'))
axhline(0, color='black')
xlabel("Time")
ylabel("Response")
title("Input and Outptut Signals")
show()
```

Plots

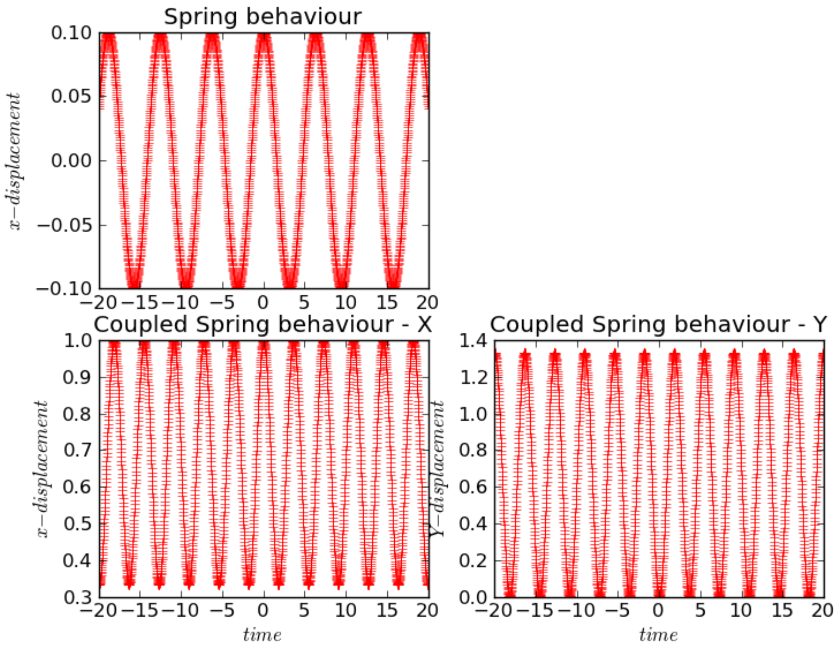

Figure 1: Spring

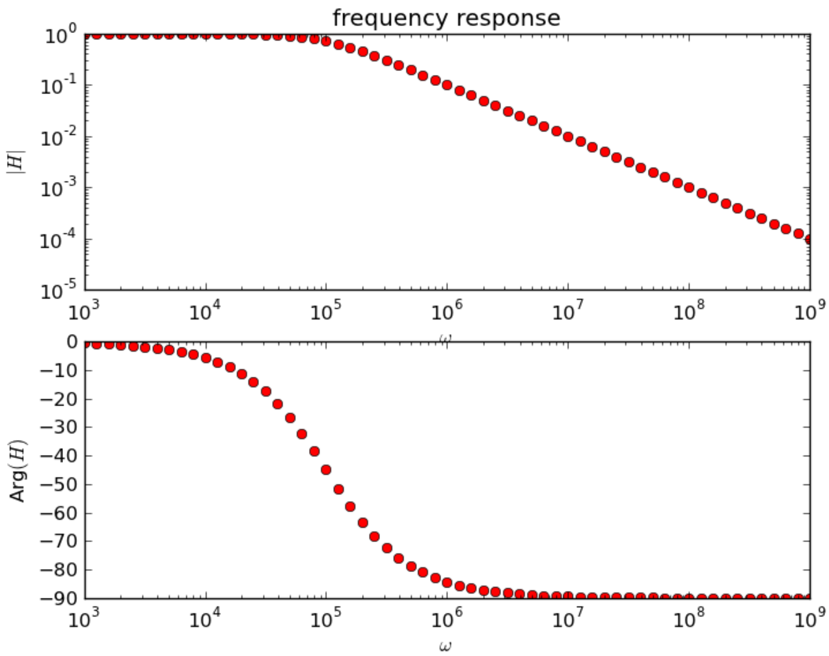

Figure 2: Transfer Function

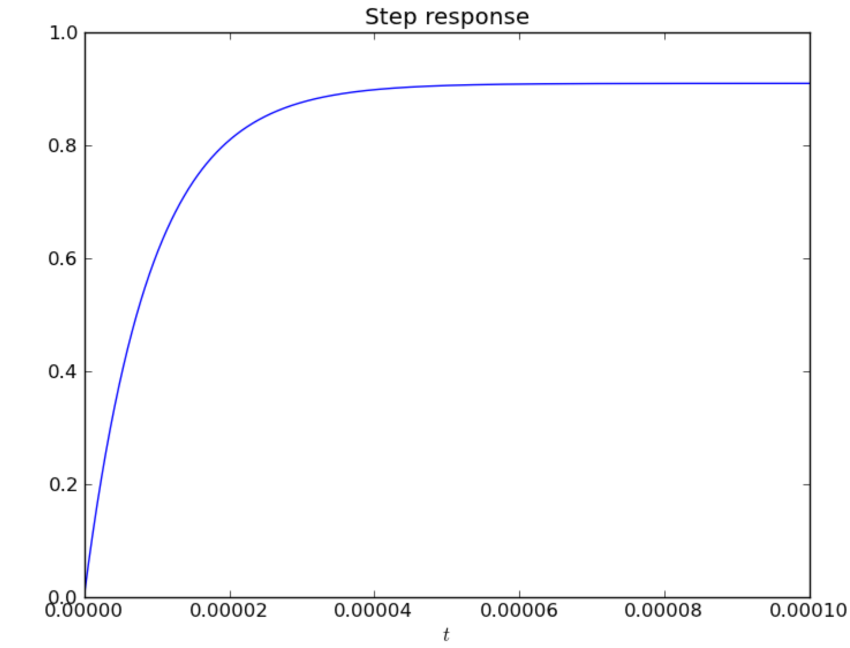

Figure 3: Step Response

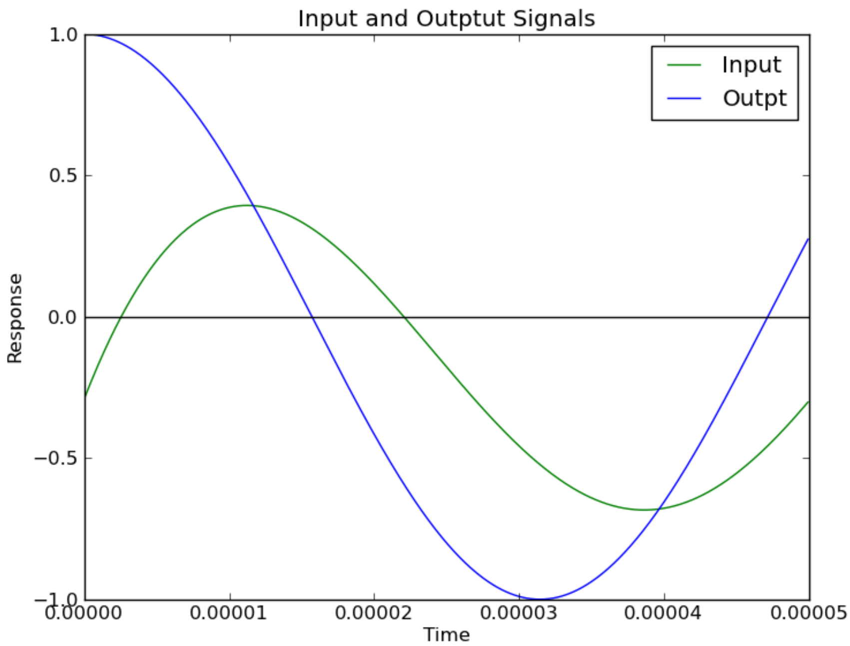

Figure 4: Input/Output voltage

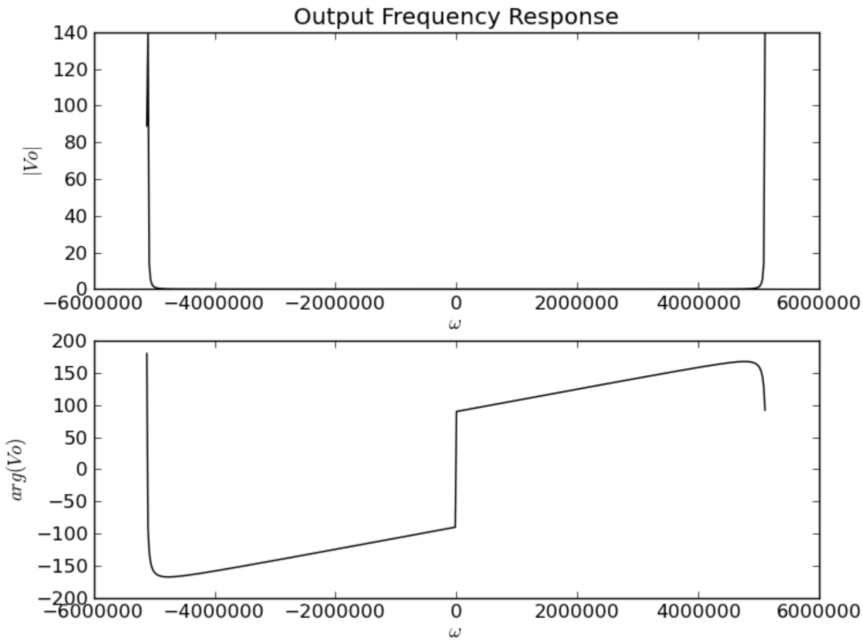

Figure 5: FFT an phase