Assignment 6

Pruthvi Raj R G,EE17B114

March 12, 2019

In this assignment, we will look at how to analyse Linear Time-invariant Systems with numerical tools in Python. LTI systems are what Electrical Engineers spend most of their time thinking about - linear circuit analysis or communication channels for example. In this assignment we will use mostly mechanical examples,

1 Assignment Questions

1.1 Problem 1

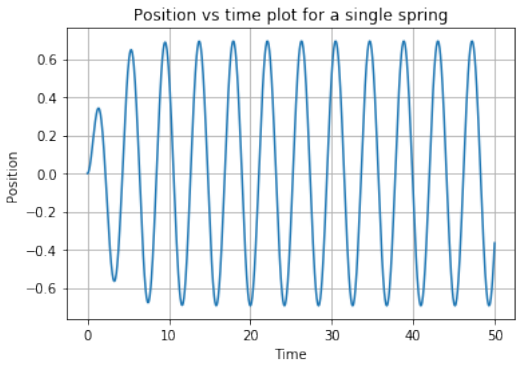
The Laplace transform of f(t) = $\cos(1.5t)e^{0.5t}\ u_0(t)$ is given by

$$F(s) = \frac{s+0.5}{(s+0.5)^2+2.25}$$

We solve for the time response of a spring like system satisfying:

$$\frac{dx^2}{dt^2} + 2.25x = f(t)$$

Also, initial conditions are intial rest.



We notice that the steady state response is a decaying sinusoid function. This makes intuitive sense, according to Bode plot inerpretations. Also,

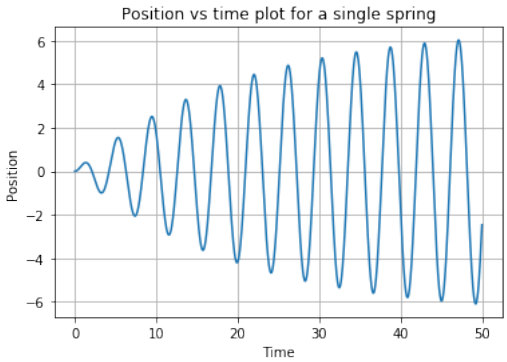
note that the decay is hard to observe, unless the timespan is broadened.

The code:

```
num = np.poly1d([1,0.5])
den = np.polymul([1,1,2.5],[1,0,2.25])
X1 = sp.lti(num,den)
t = np.linspace(0.0,50.0,1001)
t,x1 = sp.impulse(X1,None,t)
plt.plot(t,x1)
plt.xlabel('Time')
plt.ylabel('Position')
plt.title('Position vs time plot for a single spring')
plt.grid()
plt.show()
```

1.2 Problem 2

Now we do the same thing with a smaller decay,f(t) = $\cos(1.5t)e^{0.05t}$ u_0(t)



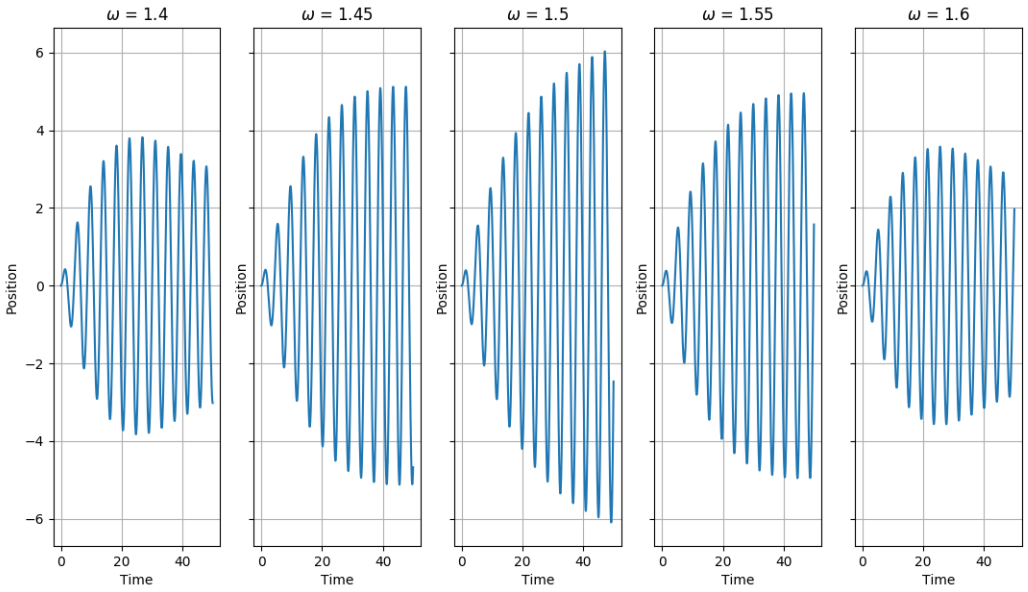
The code is as follows:

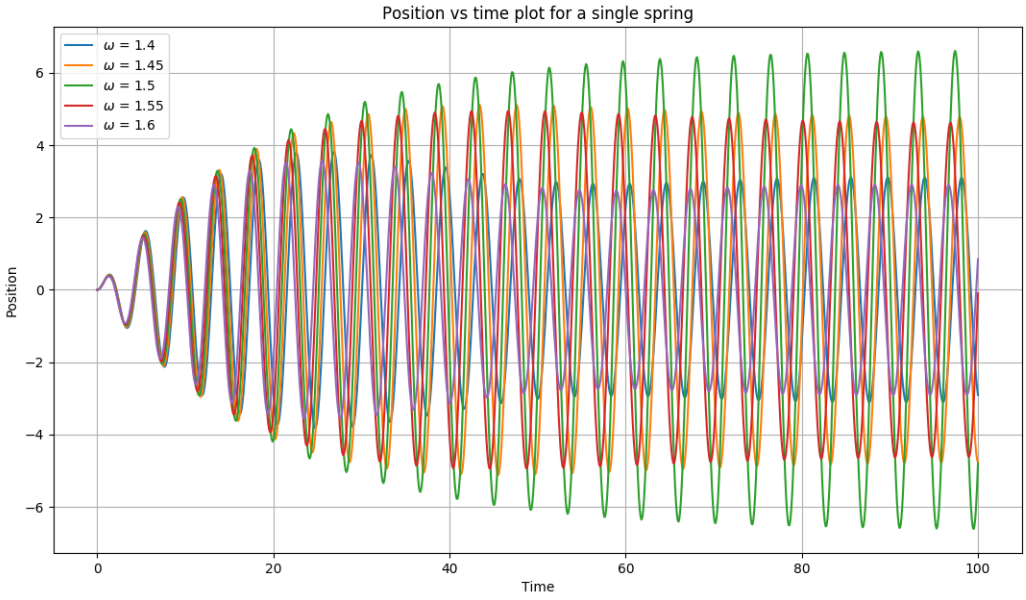
```
num = np.poly1d([1, 0.05])
den = np.polymul([1, 0.1, 2.2525],[1,0,2.25])
X2 = sp.lti(num,den)
t = np.linspace(0.0,50.0,1001)
t,x2 = sp.impulse(X2,None,t)
plt.plot(t,x2)   # Output is plotted
plt.xlabel('Time')
plt.ylabel('Position')
plt.title('Position vs time plot for a single spring')
plt.grid()
plt.show()
```

Here, the amplitude (at moderate time) is much larger owing to a much smaller decay.

1.3 Problem 3

Now, we try to vary the frequency of the cos from 1.4 to 1.6 in steps of 0.05 keeping the exponent as exp(0.05t) and plot the resulting responses. We notice that resonance occurs at 1.5 radians/sec, which is the natural frequency of this under-damped spring mass system..





Code for getting Calculating the error:

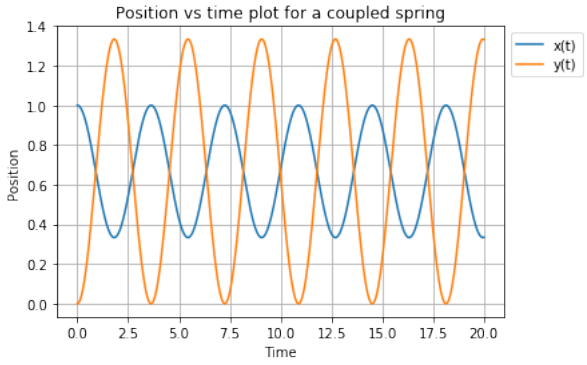
```
fig, axes = plt.subplots(1, 5, figsize=(20, 7), sharey = True)
i = 0
for w in np.linspace(1.4,1.6,5):
    num = np.poly1d([1])
    den = np.poly1d([1,0,2.25])
    H = sp.lti(num,den)
    t = np.linspace(0.0,50.0,1001)
    f = np.cos(w*t)*np.exp(-0.05*t)
    t,y,svec=sp.lsim(H,f,t)
    axes[i].plot(t,y)
    axes[i].set_xlabel('Time')
    axes[i].set_ylabel('Position')
    axes[i].set_title('$\omega$ = ' + str(w))
    axes[i].grid()
    i = i+1
# plt.legend(('$\omega$ = 1.4','$\omega$ = 1.45',
'$\omega$ = 1.5','$\omega$ = 1.55','$\omega$ = 1.6'))
plt.show()
```

1.4 Problem 4

Now,We try to solve for the coupled differential system:

$$\frac{dx^2}{dt^2} + \text{xy} = 0$$
$$\frac{dy^2}{dt^2} + 2\text{xy} = 0$$

The solution for these can be easily obtained by substituting the values of y from the first equation into the second and solving in s-domain.We repeat the same for the other.



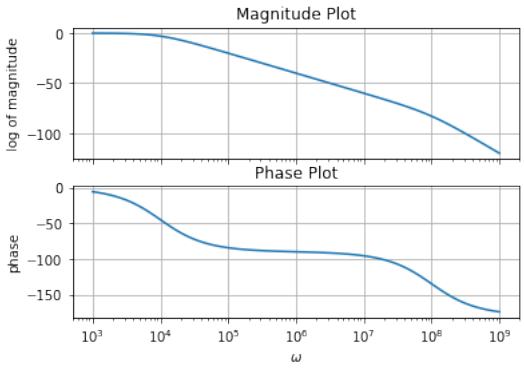
Code for finding the above solutions:

```
num = np.poly1d([1, 0, 2, 0])   # Numerator of X(s)
den = np.poly1d([1, 0, 3, 0, 0])  # Denominator of X(s)
X = sp.lti(num,den)              # X(s) calculated
t,x = sp.impulse(X,None,T=np.linspace(0,20,1001))  # Time range for x plot

num = np.poly1d([2, 0]) # Numerator of Y(s)
den = np.poly1d([1,0,3,0,0])    # Denominator of Y(s)
Y = sp.lti(num,den)         # Y(s) calculated
t,y = sp.impulse(Y,None,T=np.linspace(0,20,1001))  # Time range for y plot
```

1.5 Problem 5

Here,We obtain the magnitude and phase response of the steady state transfer function of a low-pass network, a second order network.



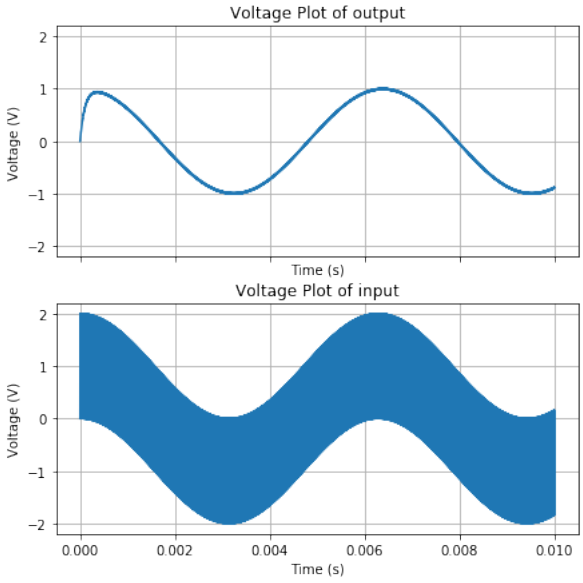

```
num = np.poly1d([1])
den = np.poly1d([10**(-12),10**(-4),1])
H = sp.lti(num,den)
w,S,phi=H.bode()
```

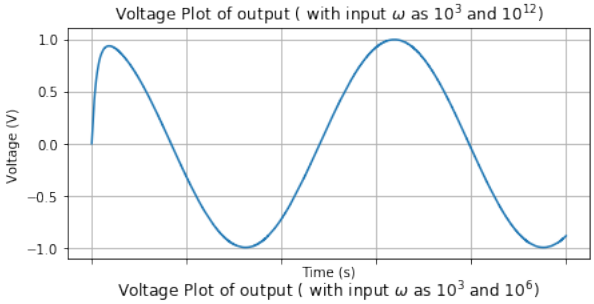
1.6 Problem 6

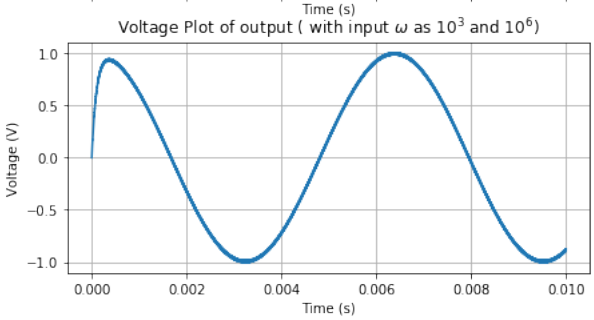
With regard to the previous two port, we obtain the output for:

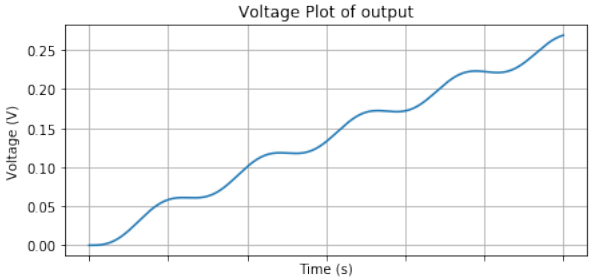
$$v_i(t) = cos(10^{3t}u(t)cos(10^{6t})u(t)$$

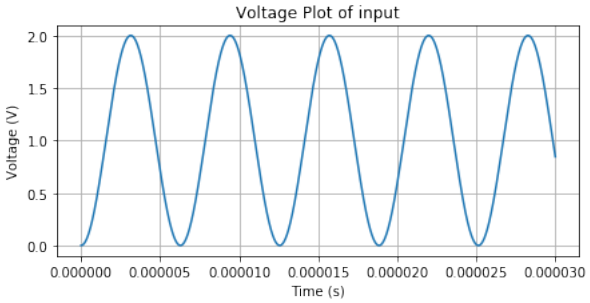
Obtain the output voltage $v_0(t)$ by defining the transfer function as a system and obtaining the output using signal.lsim.

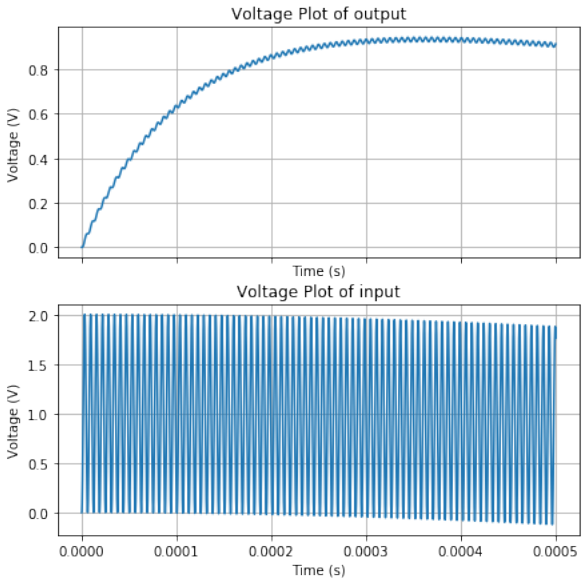



Voltage Plot of output (with input ω as 10^3 and 10^{12})
Voltage (V)
Time (s)


Voltage Plot of output (with input ω as 10^3 and 10^6)
Voltage (V)
Time (s)


Voltage Plot of output
Voltage (V)
Time (s)


Voltage Plot of input
Voltage (V)
Time (s)




```
t = np.linspace(0,10*(10**(-3)),100001)
v_i = np.cos((10**3)*t) - np.cos((10**6)*t)
time,v_0,vsec = sp.lsim(H,v_i,t)

fig, axes = plt.subplots(2, 1, figsize=(7, 7), sharex = True)
axes[0].plot(time,v_0)
axes[0].set_ylabel('Voltage (V)')
axes[0].set_title('Voltage Plot of output')
axes[0].set_xlabel('Time (s)')
axes[0].grid()

axes[1].plot(time,v_i)
axes[1].set_ylabel('Voltage (V)')
axes[1].set_title('Voltage Plot of input')
axes[1].set_xlabel('Time (s)')
axes[1].grid()
plt.show()
```

1.7 Results and Conclusion

Analysing LTI systems and approximate linear behaviour models are crucial in many aspects of engineering. Here, we have explored the usage of built-in python libraries under numpys poly- nomial library, scipys signal processing and linear simulation modules. This allows for convinient analysis of many systems, including spring systems, electrical fil- ters, coupled bodies,...etc. We have covered a few as a part of this assignment.