Lab 3 - Solution

October 17, 2019

1 Lab 3 - Filters and Z Transforms

1.1 SOLUTION

This week, the lab will work through plotting the impulse response and z transform of filters.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import IPython.display as ipd
        from scipy import signal

        %matplotlib inline
```

1.1.1 (a)

Find the transfer function H(z) for the filter corresponding to the following difference equation:

$$y[n] = 0.9y[n-1] + 1.8x[n]$$

Briefly show your work using Latex syntax.
Hint: Latex syntax should go between two pairs of dollar signs. Here's an example calculation.

$$x^2 - 4 = 0$$

$$(x-2)(x+2) = 0$$

$$x = 2, x = -2$$

Also, fractions can be made like this:

$$\frac{3x-4}{x+6}$$

1.1.2 Your Answer:

Take the z-transform of both sides to get:

$$Y(z) = 0.9z^{-1}Y(z) + 1.8X(z)$$

$$Y(z) - 0.9z^{-1}Y(z) = 1.8X(z)$$

$$Y(z)(1 - 0.9z^{-1}) = 1.8X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1.8}{1 - 0.9z^{-1}}$$

1.1.3 (b)

Find the causal impulse response h[n] for the above system.
Hint: You may refer to a z transform table.

1.1.4 Your Answer:

$$h[n] = 1.8(0.9)^n u[n]$$

1.1.5 (c)

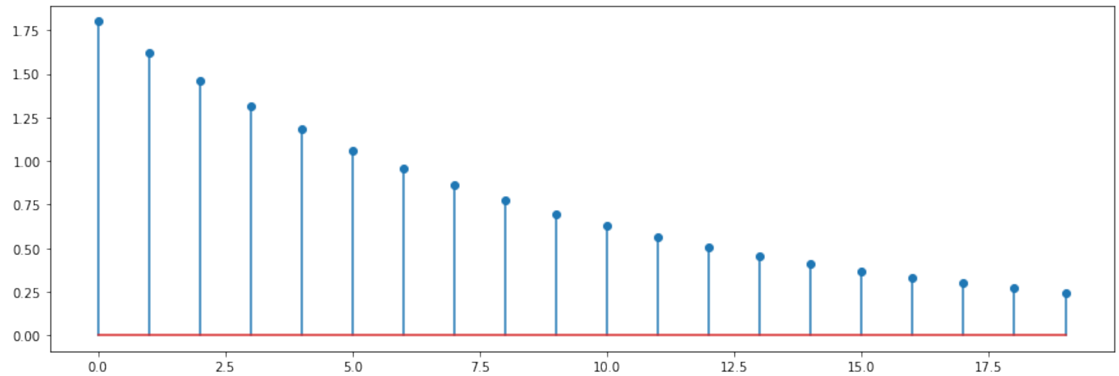
Plot the first 20 samples of the impulse response h[n]. What kind of filter is this?
Hint: the signal.dlti() function takes as input two lists, the transfer function's numerator coefficients and denominator coefficients.

1.1.6 Your Answer:

```
In [19]: ##### to do #####
         # determine the transfer function coefficients
         numH = [1.8, 0]
         denH = [1, -0.9]

         # compute the impulse response
         systemH = signal.dlti(numH, denH)
         n, h = signal.dimpulse(systemH, n=20)

         # plot the impulse response
         plt.figure(figsize = (15,5))
         plt.stem(n,np.squeeze(h));
```



1.1.7 (d)

Plot any poles and zeros of this filter on the z plane. Is the system stable?
Hint: You may consider using the following functions: - signal.ZerosPolesGain() - plt.scatter() Also, remember that if x is a complex number, x.real and x.imag are its components.

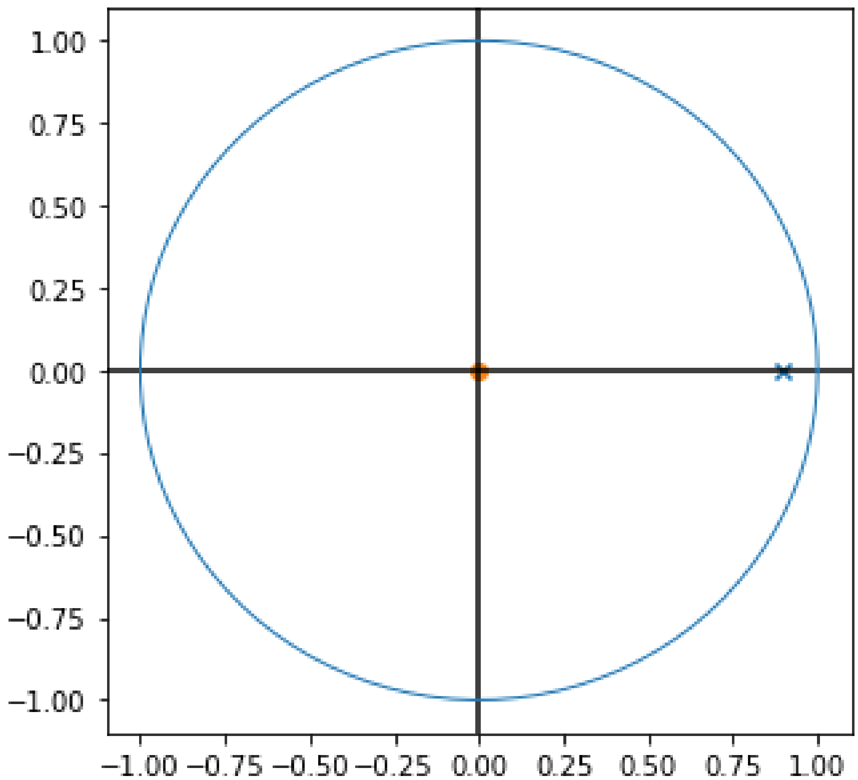
1.1.8 Your Answer:

In [20]:
```
# make plot with axes
fig,ax = plt.subplots(1,1,figsize=(5,5))
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')

# draw unit circle
t = np.linspace(0,np.pi*2,100)
plt.plot(np.cos(t), np.sin(t), linewidth=1)

##### to do #####
# plot the poles and zeros of system H
H = signal.ZerosPolesGain(systemH)

plt.scatter(H.poles.real, H.poles.imag, marker="x");
plt.scatter(H.zeros.real, H.zeros.imag, marker="o");
```



1.1.9 (e)

Condsider feeding the output of this system directly into another system, G(z). Find a transfer function for G(z) such that any input signal after going through H(z) and then G(z) will come out unchanged. In other words, G(z) cancels all effects of H(z).

1.1.10 Your Answer:

$$G(z) = \frac{1 - 0.9z^{-1}}{1.8}$$

1.1.11 (f)

Find the causal impulse response g[n] for G(z).

1.1.12 Your Answer:

$$g[n] = 0.\overline{5}\delta[n] - 0.5\delta[n-1]$$

1.1.13 (g)

Plot the first 20 samples of the impulse response h[n]. What kind of filter is this?

Hint: you may consider using the following functions: - signal.dlti() - signal.dimpulse() - plt.stem() - np.squeeze()

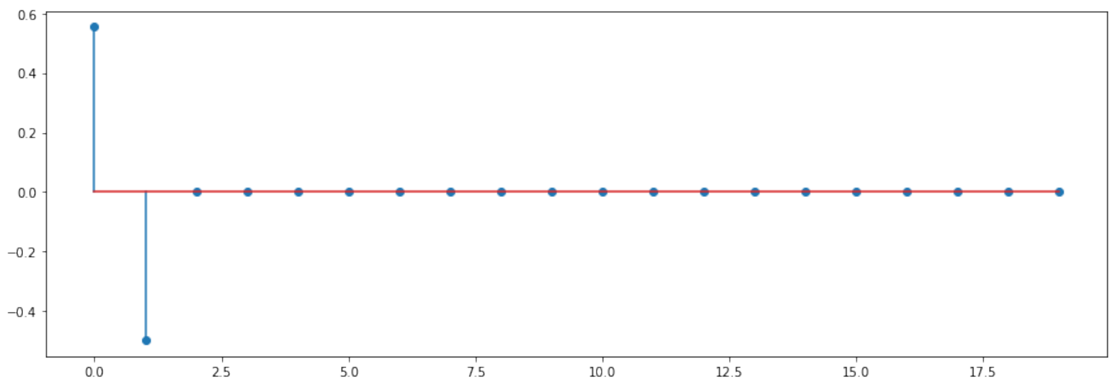
1.1.14 Your Answer:

```
In [21]: ##### to do #####

         # determine the transfer function coefficients
         numG = [1, -0.9]
         denG = [1.8, 0]

         # compute the impulse response
         systemG = signal.dlti(numG,denG)
         n, g = signal.dimpulse(systemG, n=20)

         # plot the impulse response
         plt.figure(figsize = (15,5))
         plt.stem(n,np.squeeze(g));
```



1.1.15 (h)

Plot any poles and zeros of this filter G(z) on the z plane. Is the system stable?

1.1.16 Your Answer:

```
In [22]: # make plot with axes
         fig,ax = plt.subplots(1,1,figsize=(5,5))
         ax.axhline(y=0, color='k')
         ax.axvline(x=0, color='k')

         # draw unit circle
```

```python
t = np.linspace(0,np.pi*2,100)
plt.plot(np.cos(t), np.sin(t), linewidth=1)

##### to do #####
# plot the poles and zeros of system G
G = signal.ZerosPolesGain(systemG)

plt.scatter(G.poles.real,G.poles.imag,marker="x");
plt.scatter(G.zeros.real,G.zeros.imag,marker="o");
```