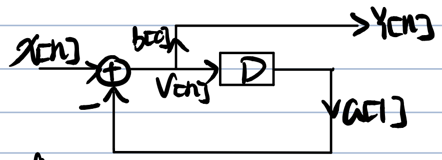
Assignment: EE599 Homework 1.
Name: Zifan Wang
ID: 9565587296

Problem 2. Using filters in Python

1. a). First order (L=1) AR filter



Difference Equation:

$$y[n] = b[0] \cdot x[n] + a[1] \cdot y[n-1]$$

z-transform:

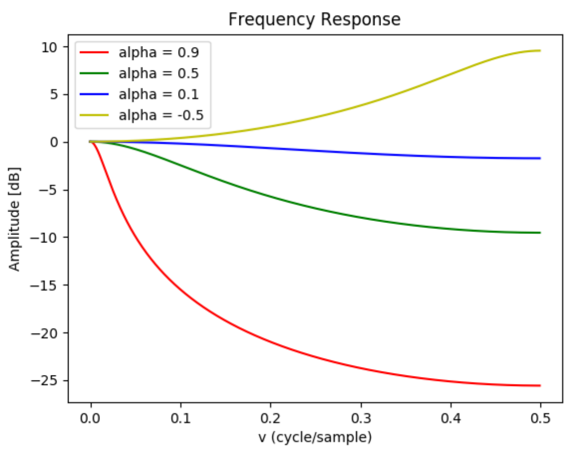
$$y[n] - a[1] \cdot y[n-1] = b[0] \cdot x[n]$$

$$Y[z] - a[1] \cdot Y[z] \cdot z^{-k=1} = b[0]\, z^{-k=0}$$

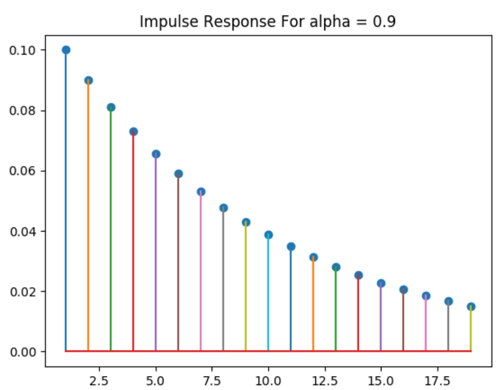
$$(1 - a[1] \cdot z^{-1}) \cdot Y[z] = b[0]$$

$$Y[z] = \frac{b[0]}{1 - a[1] \cdot z^{-1}}, \quad b[0] = (1-\alpha), \quad a[1] = \alpha$$

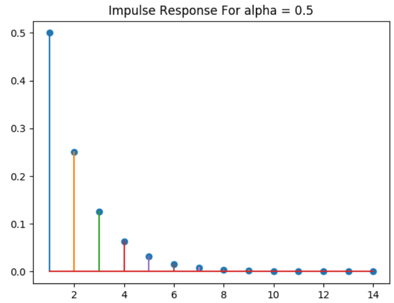
b).



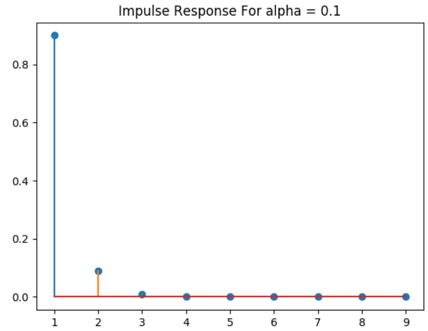
c). $\alpha = 0.9$: $y[0] = 1 - 0.9 = 0.1 \longrightarrow 0.1 \times 20\% = 0.02 \longrightarrow n = 16$



$\alpha = 0.5$: $\quad y[0] = 1 - 0.5 = 0.5 \longrightarrow 0.5 \times 20\% = 0.1 \longrightarrow n=3$



$\alpha = 0.1$: $\quad y[0] = 1 - 0.1 = 0.9 \longrightarrow 0.9 \times 20\% = 0.18 \longrightarrow n=1$



2. Design filter.

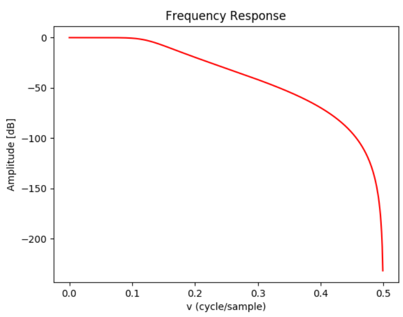
a).

```
± |HW1-Problem2 ?:3 ✗| → python3 AR_Filter.pyNumerator:  [0.01020948 0.04083792 0.06125688 0.04083792 0.01020948]
Denominator:  [ 1.         -1.96842779  1.73586071 -0.72447083  0.1203896 ]
```

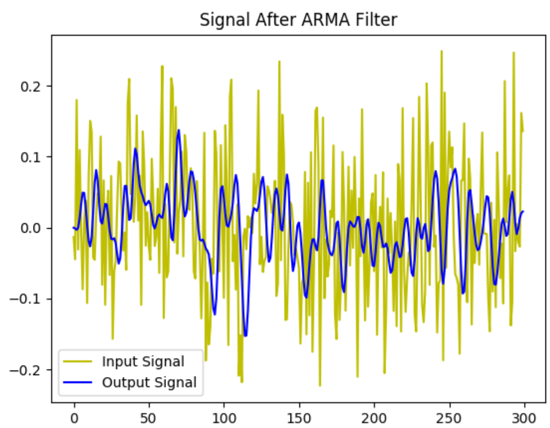
$$H[z] = \frac{0.01021 + 0.0408 z^{-1} + 0.0613 z^{-2} + 0.0408 z^{-3} + 0.01021 z^{-4}}{1 - 1.968 z^{-1} + 1.736 z^{-2} - 0.7245 z^{-3} + 0.1204 z^{-4}}$$

$\therefore$ $b[0] = 0.01021$, $b[1] = 0.0408$, $b[2] = 0.0613 +$ $b[4] = 0.0408$, $b[5] = 0.01021$

$a[1] = -1.968$, $a[2] = 1.736$, $a[3] = -0.7245$, $a[4] = 0.1204$



b).



Code:

AR_Filter.py

```python
# Assignment: Homework1-Problem2-Filter
# Author: Zifan Wang
# Object: Simulation of first order AR Filter

import numpy as np
from scipy import signal
import matplotlib.pyplot as plt


# Fisrt Order AR Filter: y[n] = (1-a) - a*y[n-1]
# Experiment a with four vals.
Alpha = [0.9,0.5,0.1,-0.5]

# Plot frequency response for the AR filter
W = []
H = []
for alpha in Alpha:
    b = 1-alpha
    a = np.array([1,-alpha])
    w,h = signal.freqz(b,a,fs=1)
    W.append(w)
    H.append(h)

fig = plt.figure()
plt.title('Frequency Response')
plt.plot(W[0],20*np.log10(abs(H[0])),'r',label = 'alpha = 0.9') # alpha = 0.9
plt.plot(W[1],20*np.log10(abs(H[1])),'g',label = 'alpha = 0.5') # alpha = 0.5
plt.plot(W[2],20*np.log10(abs(H[2])),'b',label = 'alpha = 0.1') # alpha = 0.1
plt.plot(W[3],20*np.log10(abs(H[3])),'y',label = 'alpha = -0.5') # alpha = -0.5
legend = plt.legend()
plt.ylabel('Amplitude [dB]')
plt.xlabel('v (cycle/sample)')
plt.show()

# Plot Impulse Response for the AR filter
num = [0.1]
den = [1,-0.9]
dlti = signal.dlti(num,den,dt=0.1)
tout,yout = signal.dimpulse(dlti,n=20)
toutMod = tout[1:]*10
# print(toutMod)
dataMod = yout[0]
dataMod = dataMod[1:,:]
# print(dataMod)
# print(tout.shape)
# print(yout[0])
fig = plt.figure()
plt.title('Impulse Response For alpha = 0.9')
markerline, stemlines, baseline = plt.stem(toutMod, dataMod, '-')
plt.show()

num = [0.5]
den = [1,-0.5]
dlti = signal.dlti(num,den,dt=0.1)
tout,yout = signal.dimpulse(dlti,n=15)
toutMod = tout[1:]*10
# print(toutMod)
dataMod = yout[0]
dataMod = dataMod[1:,:]
# print(dataMod)
# print(tout.shape)
# print(yout[0])
fig = plt.figure()
plt.title('Impulse Response For alpha = 0.5')
markerline, stemlines, baseline = plt.stem(toutMod, dataMod, '-')
plt.show()

num = [0.9]
den = [1,-0.1]
dlti = signal.dlti(num,den,dt=0.1)
tout,yout = signal.dimpulse(dlti,n=10)
toutMod = tout[1:]*10
# print(toutMod)
dataMod = yout[0]
dataMod = dataMod[1:,:]
# print(dataMod)
# print(tout.shape)
# print(yout[0])
fig = plt.figure()
plt.title('Impulse Response For alpha = 0.1')
markerline, stemlines, baseline = plt.stem(toutMod, dataMod, '-')
plt.show()

# Design an L = 4 Butterworth filter with bandwidth of v0 = 0.25.
[b,a] = signal.butter(4,0.25,btype='low',analog=False)
print('Numerator: ',b)
print('Denominator: ',a)
# Frequency response
w,h = signal.freqz(b,a,fs=1)
fig = plt.figure()
plt.title('Frequency Response')
plt.plot(w,20*np.log10(abs(h)),'r')
plt.ylabel('Amplitude [dB]')
plt.xlabel('v (cycle/sample)')
plt.show()

# Play arround with this filter
# Create a numpy array of length 300 comprising iid realizations of a standard normal distribution.
inputSignal = np.random.normal(0,0.1,300)
outputSignal = signal.lfilter(b,a,inputSignal)
fig = plt.figure()
plt.title('Signal After ARMA Filter')
plt.plot(inputSignal,'y', label='Input Signal')
plt.plot(outputSignal,'b', label='Output Signal')
plt.legend()
plt.show()
```