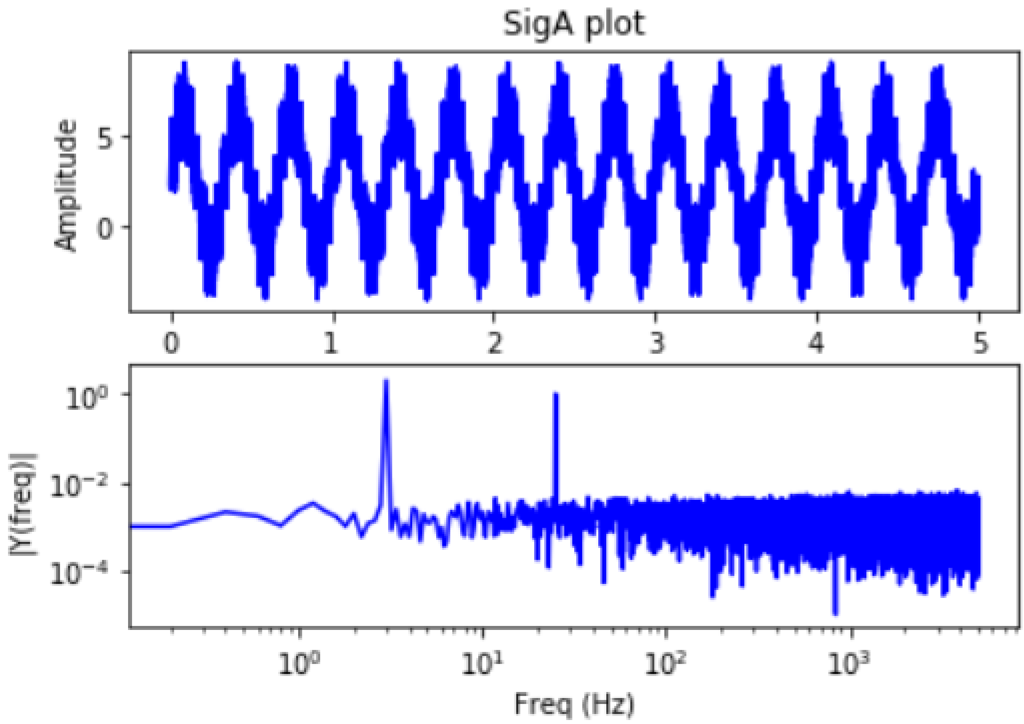
Evan Johns
ME 433

Note: Wasn't sure what signal was wanted for some of these plots. For example for sigB it appears to be the combination of 2 sin waves, the plot was smoothed so that both were visible. For sigD I used values which kept the oscillations and just smoothed them out. Code is also in the separate jupyter notebook.

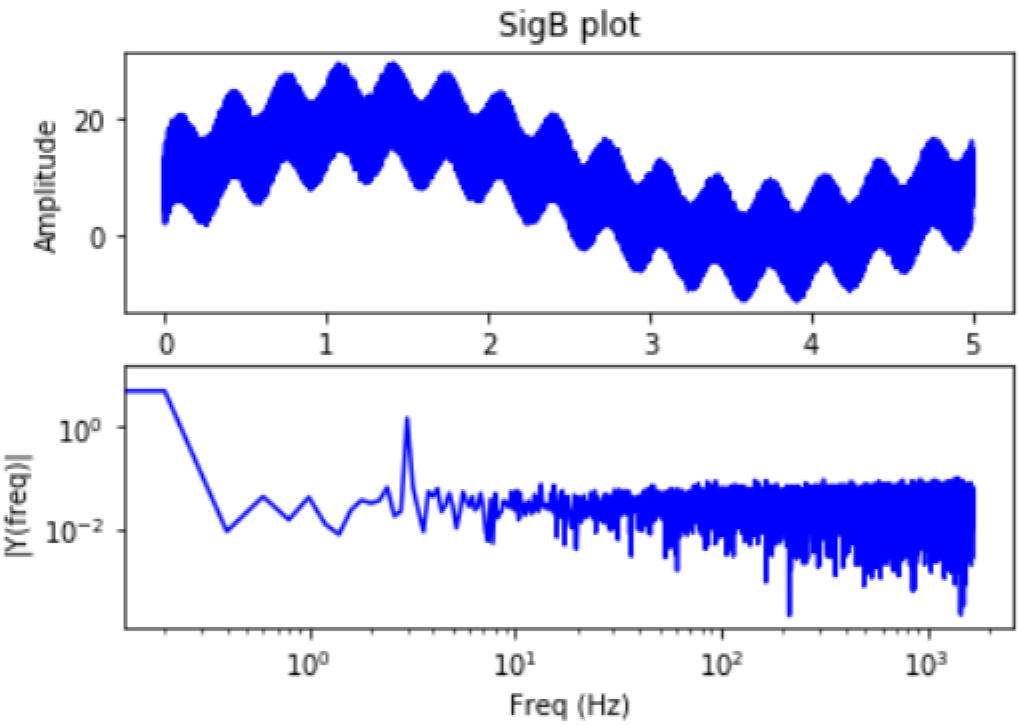
Question 4:

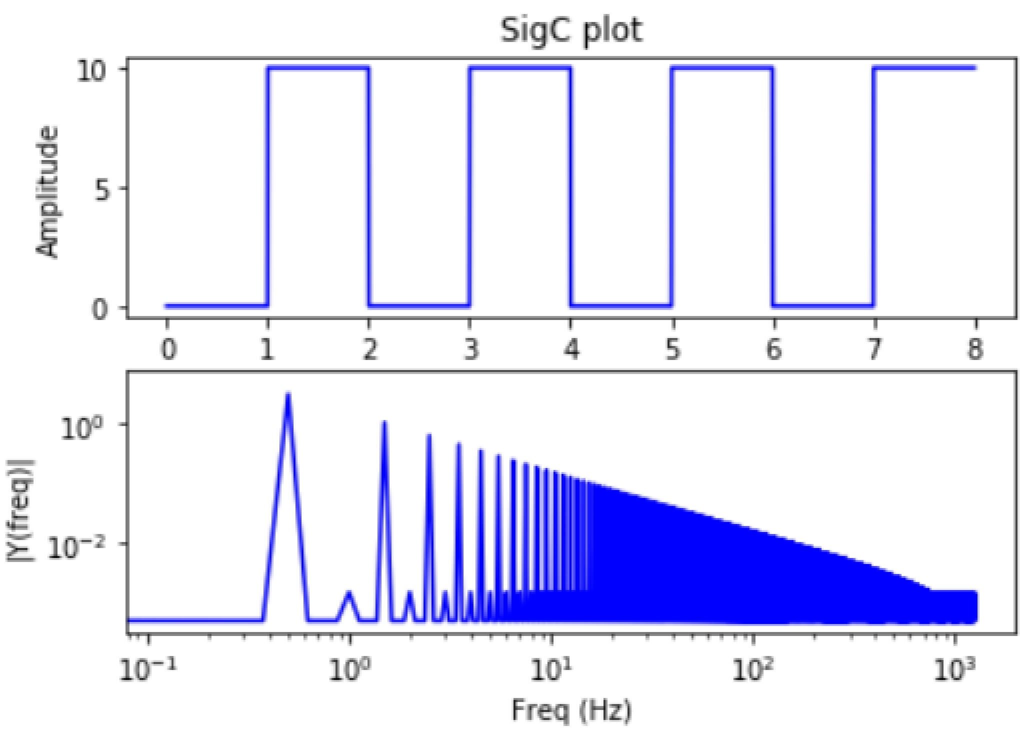
```
#question 4
import matplotlib.pyplot as plt
import numpy as np
import csv

t = [] # column 0
data1 = [] # column 1
data2 = [] # column 2

with open('sigA.csv') as f:
    # open the csv file
    reader = csv.reader(f)
    for row in reader:
        # read the rows 1 one by one
        t.append(float(row[0])) # leftmost column
        data1.append(float(row[1])) # second column
        #data2.append(float(row[2])) # third column
endtime=t[-1]
print(endtime)
nsamples=len(data1)
print(nsamples)
sam_rate=nsamples/endtime
sam_rate=round(sam_rate)
print(sam_rate)

Fs = sam_rate # sample rate
Ts = 1.0/Fs; # sampling interval
ts = np.arange(0,t[-1],Ts) # time vector
y = data1 # the data to make the fft from
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(int(n/2))] # one side frequency range
Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(int(n/2))]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(t,y,'b')
ax1.set_xlabel('Time')
ax1.set_ylabel('Amplitude')
ax2.loglog(frq,abs(Y),'b') # plotting the fft
ax2.set_xlabel('Freq (Hz)')
ax2.set_ylabel('|Y(freq)|')
```

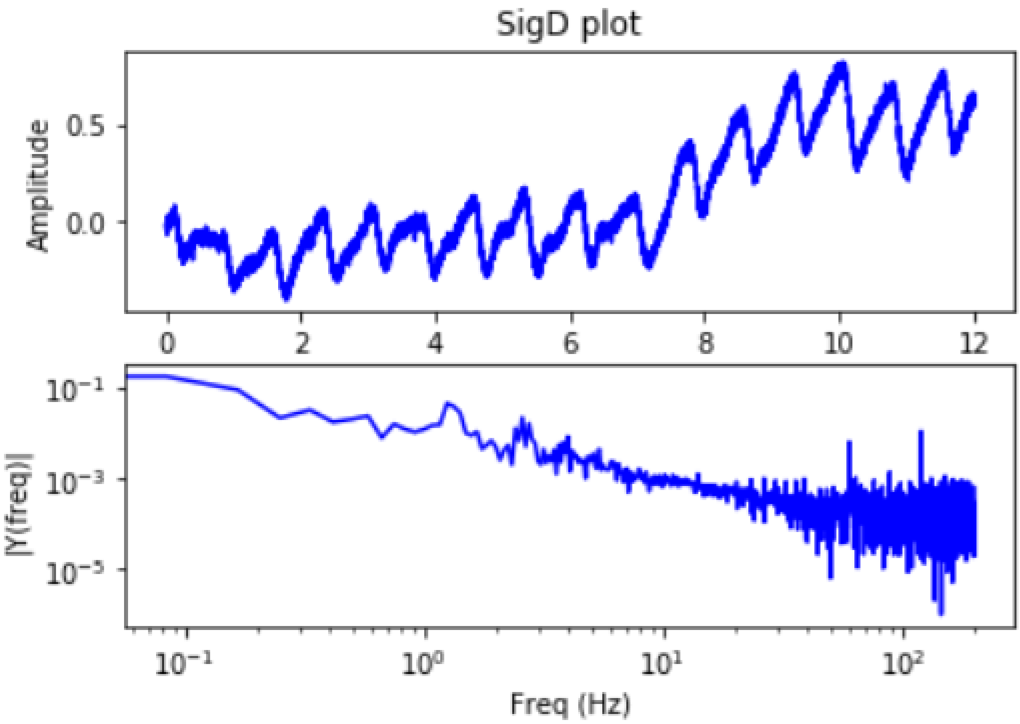
```
plt.show()
```



SigA plot
Amplitude
5
0
0
1
2
3
4
5
10^0
10^{-2}
10^{-4}
|Y(freq)|
10^0
10^1
10^2
10^3
Freq (Hz)


SigB plot
Amplitude
20
0
0
1
2
3
4
5
|Y(freq)|
10^0
10^{-2}
10^0
10^1
10^2
10^3
Freq (Hz)


SigC plot
Amplitude
10
5
0
0
1
2
3
4
5
6
7
8
|Y(freq)|
10^0
10^{-2}
10^{-1}
10^0
10^1
10^2
10^3
Freq (Hz)

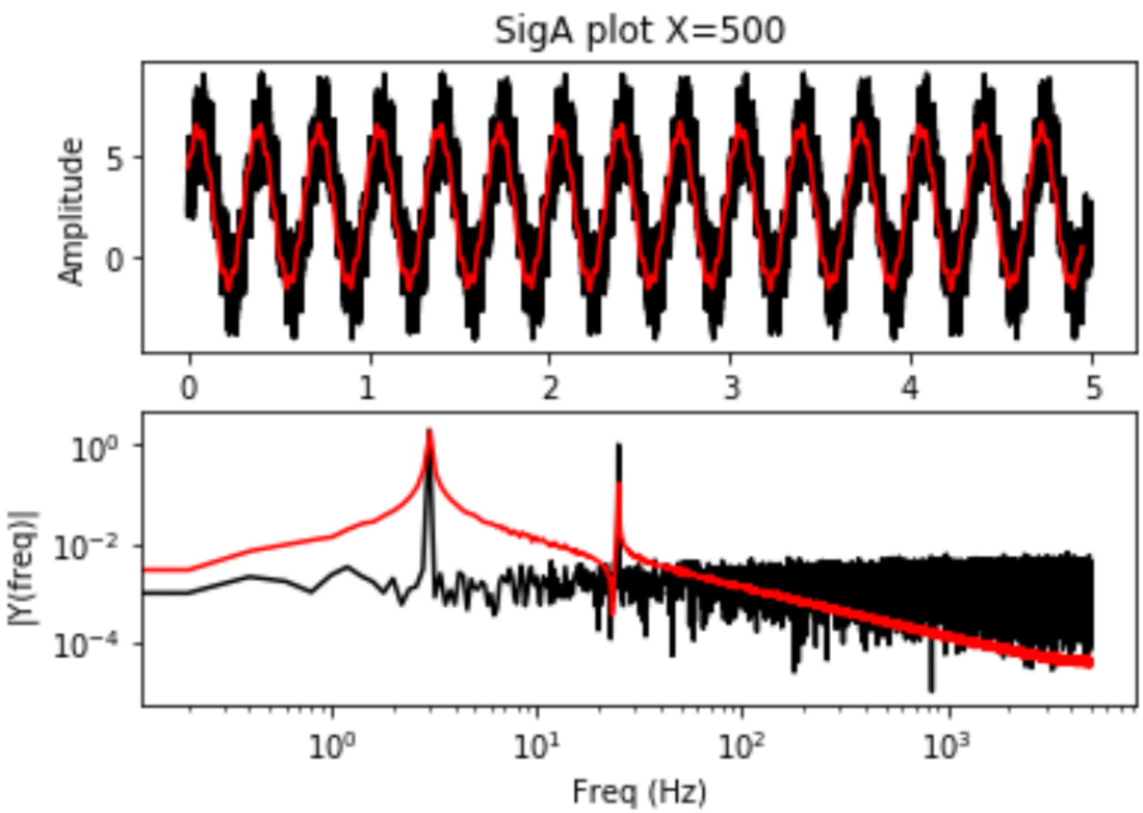


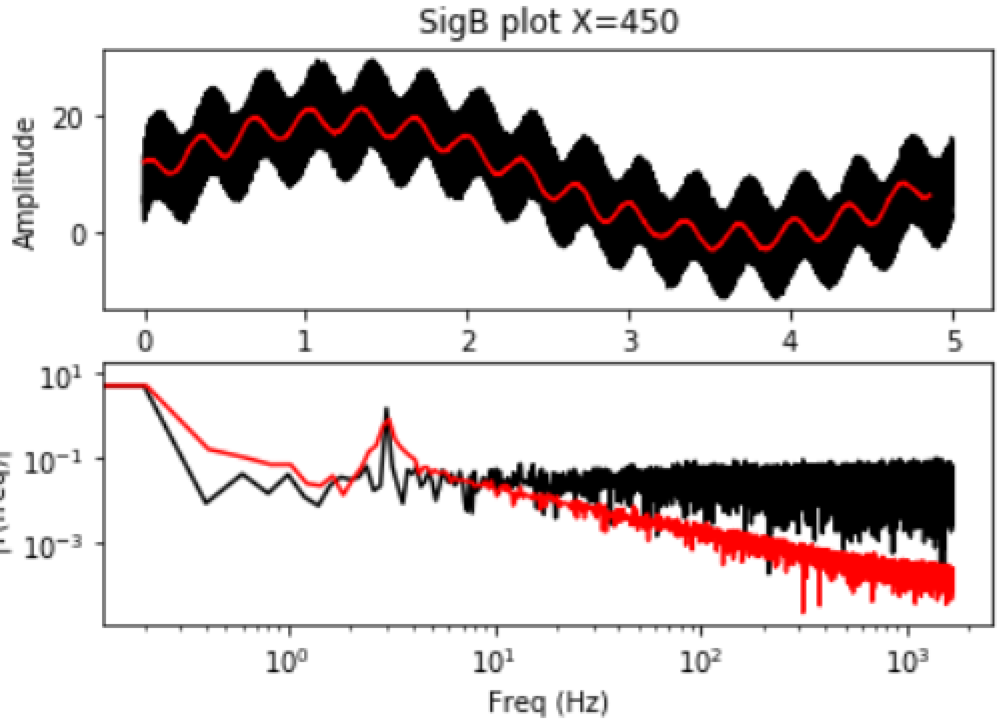
Question 5 MAF Transform:

```
#question 5
import matplotlib.pyplot as plt
import numpy as np
import csv

t = [] # column 0
data1 = [] # column 1
data2 = [] # column 2
X=500;
newdata=[]#the averaged vector
newtime=[]#the shortened time vector
tempdata=0;#the temporary averaged value

with open('sigC.csv') as f:
    # open the csv file
    reader = csv.reader(f)
    for row in reader:
        # read the rows 1 one by one
        t.append(float(row[0])) # leftmost column
        data1.append(float(row[1])) # second column
        #data2.append(float(row[2])) # third column

    for i in range(0,(len(data1)-X)):
        tempdata=0
        for ii in range(0,X):
            tempdata=tempdata+data1[i+ii]
        newdata.append(tempdata/X)
        newtime.append(t[i])
```

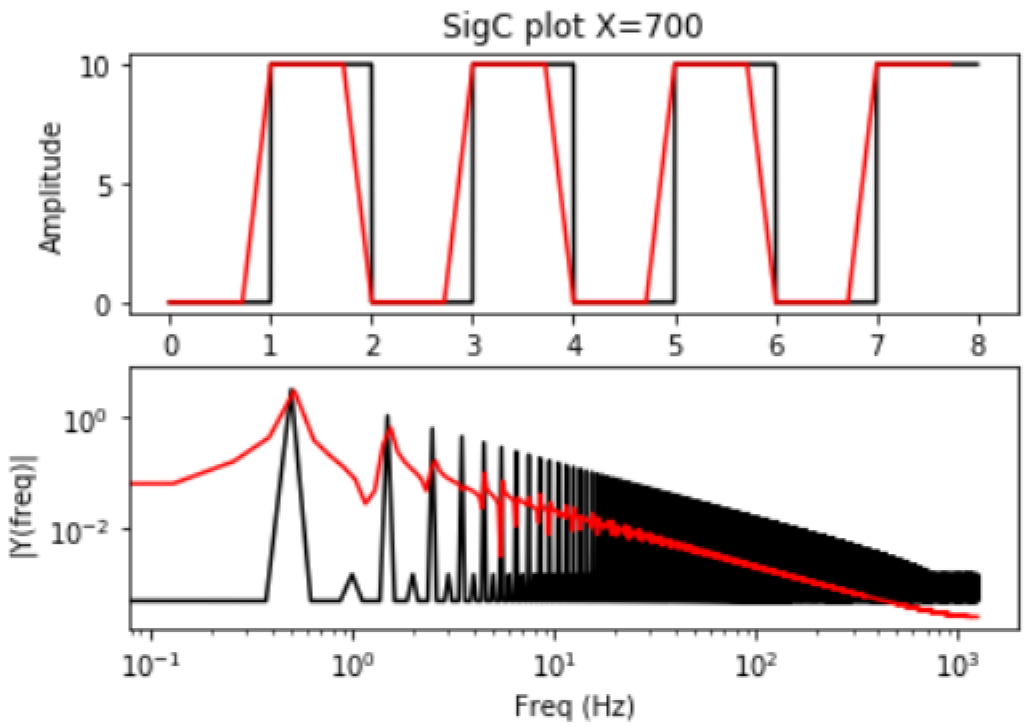
```
#unfiltered data
endtime=t[-1]
nsamples=len(data1)
sam_rate=nsamples/endtime
sam_rate=round(sam_rate)


Fs = sam_rate # sample rate
Ts = 1.0/Fs; # sampling interval
ts = np.arange(0,t[-1],Ts) # time vector
y = data1 # the data to make the fft from
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs

frq = k/T # two sides frequency range
frq = frq[range(int(n/2))] # one side frequency range
Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(int(n/2))]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(t,y,color="black")
ax1.set_xlabel('Time')
ax1.set_ylabel('Amplitude')
ax1.set_title('Unfiltered sample')
ax2.loglog(frq,abs(Y),color="black") # plotting the fft
ax2.set_xlabel('Freq (Hz)')
ax2.set_ylabel('|Y(freq)|')
plt.show()
y2=y
Y2=Y
frq2=frq
endtime=newtime[-1]
nsamples=len(newdata)
sam_rate=nsamples/endtime
sam_rate=round(sam_rate)

Fs = sam_rate # sample rate
Ts = 1.0/Fs; # sampling interval
ts = np.arange(0,newtime[-1],Ts) # time vector
y = newdata # the data to make the fft from
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(int(n/2))] # one side frequency range
Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(int(n/2))]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(t,y2,color="black")
ax1.plot(newtime,y,'r-')
ax1.set_xlabel('Time')
ax1.set_ylabel('Amplitude')
ax1.set_title('Filtered sample X='+str(X))
ax2.loglog(frq2,abs(Y2),color="black") # plotting the fft
```

```
ax2.loglog(frq,abs(Y),'r-') # plotting the fft
ax2.set_xlabel('Freq (Hz)')
ax2.set_ylabel('|Y(freq)|')
plt.show()
```

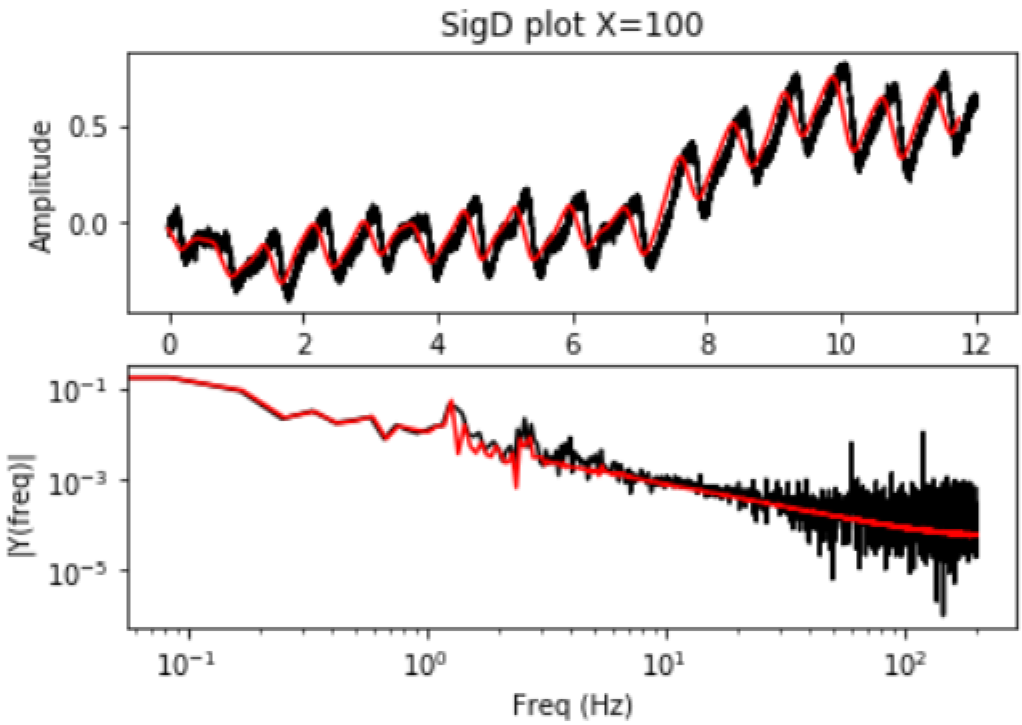




SigB plot X=450
Amplitude
|Y(freq)|
Freq (Hz)



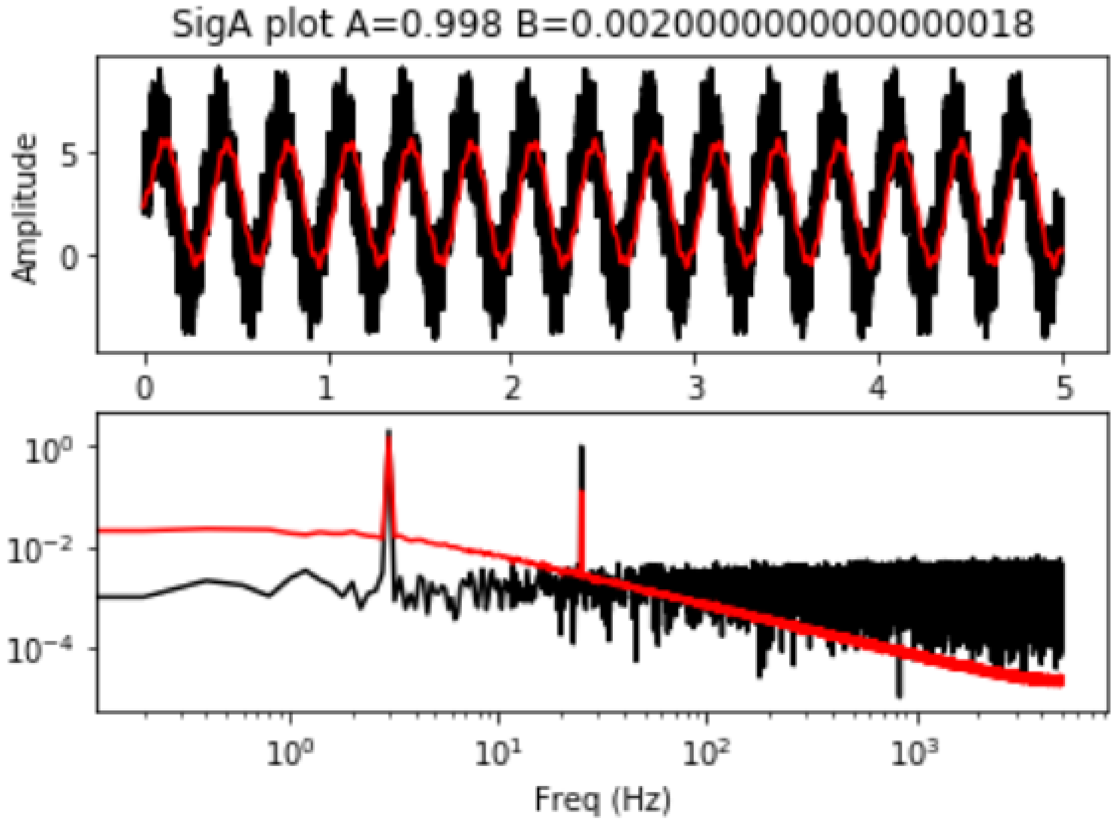
SigC plot X=700
Amplitude
|Y(freq)|
Freq (Hz)

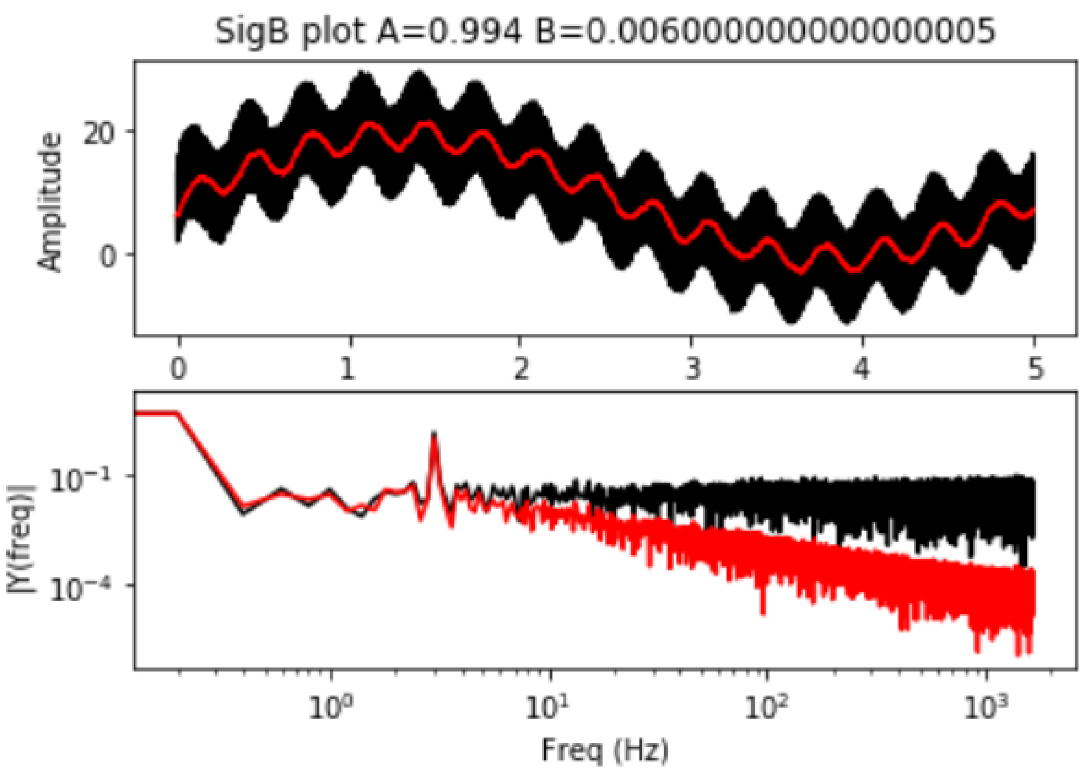


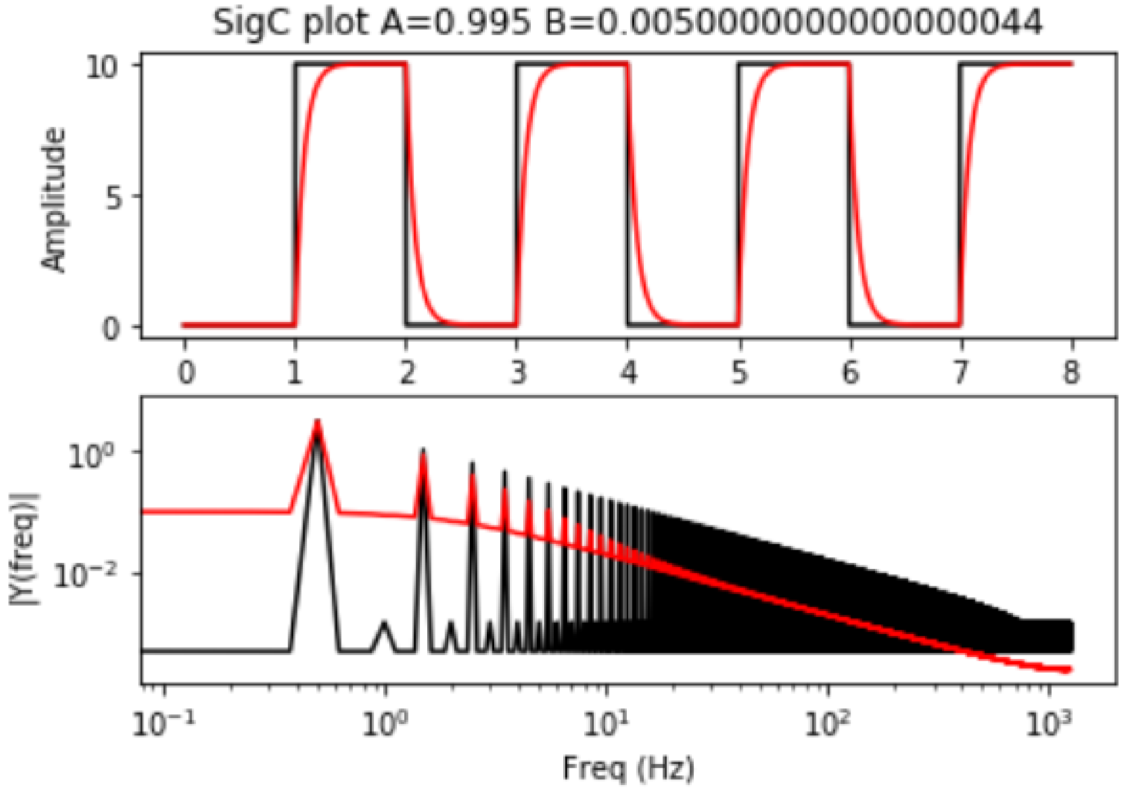
Question 6: IIR filter

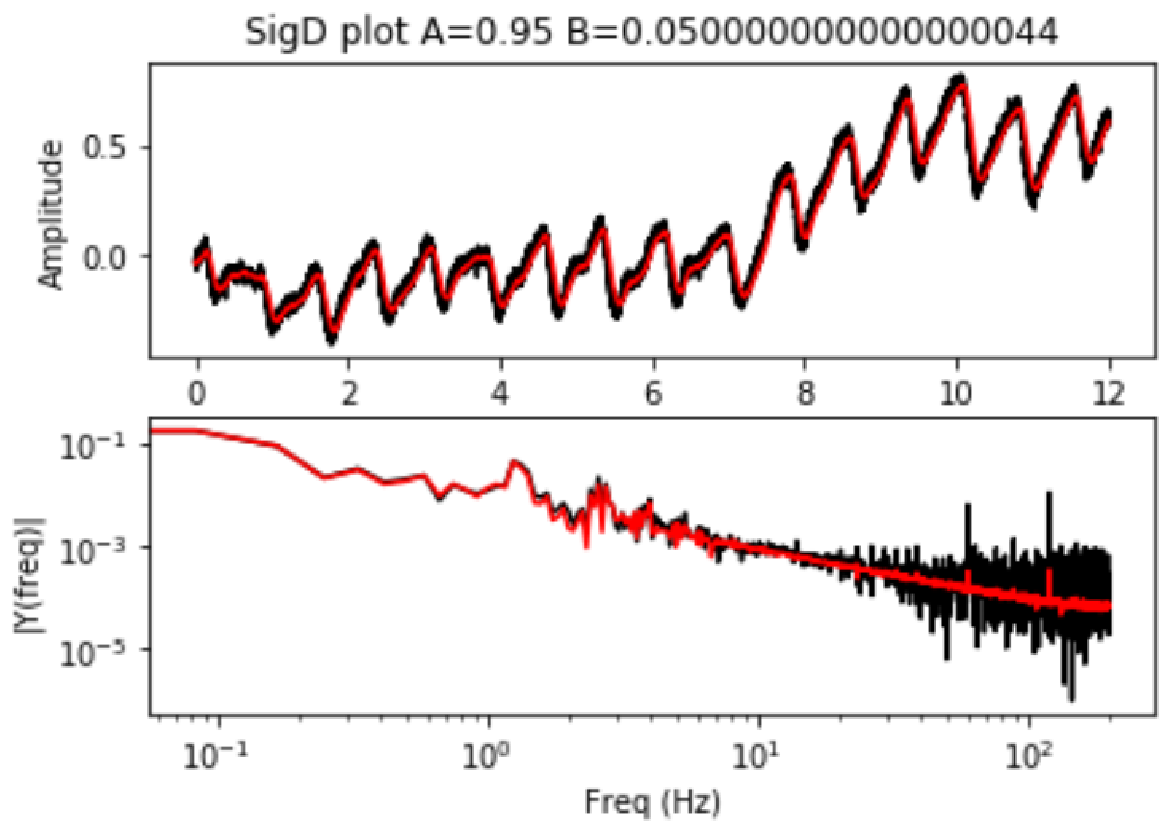
```
#question 6
import matplotlib.pyplot as plt
import numpy as np
import csv

t = [] # column 0
data1 = [] # column 1
data2 = [] # column 2
A=.95
B=1-A
with open('sigD.csv') as f:
    # open the csv file
    reader = csv.reader(f)
    for row in reader:
        # read the rows 1 one by one
        t.append(float(row[0])) # leftmost column
        data1.append(float(row[1])) # second column
        #data2.append(float(row[2])) # third column
endtime=t[-1]
nsamples=len(data1)
sam_rate=nsamples/endtime
sam_rate=round(sam_rate)



newdata=[]#the new vector
new_avg=data1[0]
for i in range(0,len(data1)):
    new_avg=A*new_avg+B*data1[i]
```

```
    newdata.append(new_avg)


Fs = sam_rate # sample rate
Ts = 1.0/Fs; # sampling interval
ts = np.arange(0,t[-1],Ts) # time vector
y = data1 # the data to make the fft from
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(int(n/2))] # one side frequency range
Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(int(n/2))]

Fs = sam_rate # sample rate
Ts = 1.0/Fs; # sampling interval
ts = np.arange(0,t[-1],Ts) # time vector
y = newdata # the data to make the fft from
n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq2 = k/T # two sides frequency range
frq2= frq[range(int(n/2))] # one side frequency range
Y2 = np.fft.fft(y)/n # fft computing and normalization
Y2 = Y2[range(int(n/2))]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(t,data1,color="black")
ax1.plot(t,newdata,color="red")
ax1.set_xlabel('Time')
ax1.set_ylabel('Amplitude')
ax1.set_title('SigD plot A='+str(A)+" B="+str(B))
ax2.loglog(frq,abs(Y),color="black") # plotting the fft
ax2.loglog(frq2,abs(Y2),color="red") # plotting the fft
ax2.set_xlabel('Freq (Hz)')
ax2.set_ylabel('|Y(freq)|')
plt.show()
```


SigA plot A=0.998 B=0.0020000000000000018
Amplitude
|Y(freq)|
Freq (Hz)


SigB plot A=0.994 B=0.0060000000000000005
Amplitude
|Y(freq)|
Freq (Hz)


SigC plot A=0.995 B=0.005000000000000044
Amplitude
10
5
0
0 1 2 3 4 5 6 7 8
|Y(freq)|
10^{0}
10^{-2}
10^{-1} 10^{0} 10^{1} 10^{2} 10^{3}
Freq (Hz)


SigD plot A=0.95 B=0.050000000000000044
Amplitude
0.5
0.0
0 2 4 6 8 10 12
|Y(freq)|
10^{-1}
10^{-3}
10^{-5}
10^{-1} 10^{0} 10^{1} 10^{2}
Freq (Hz)