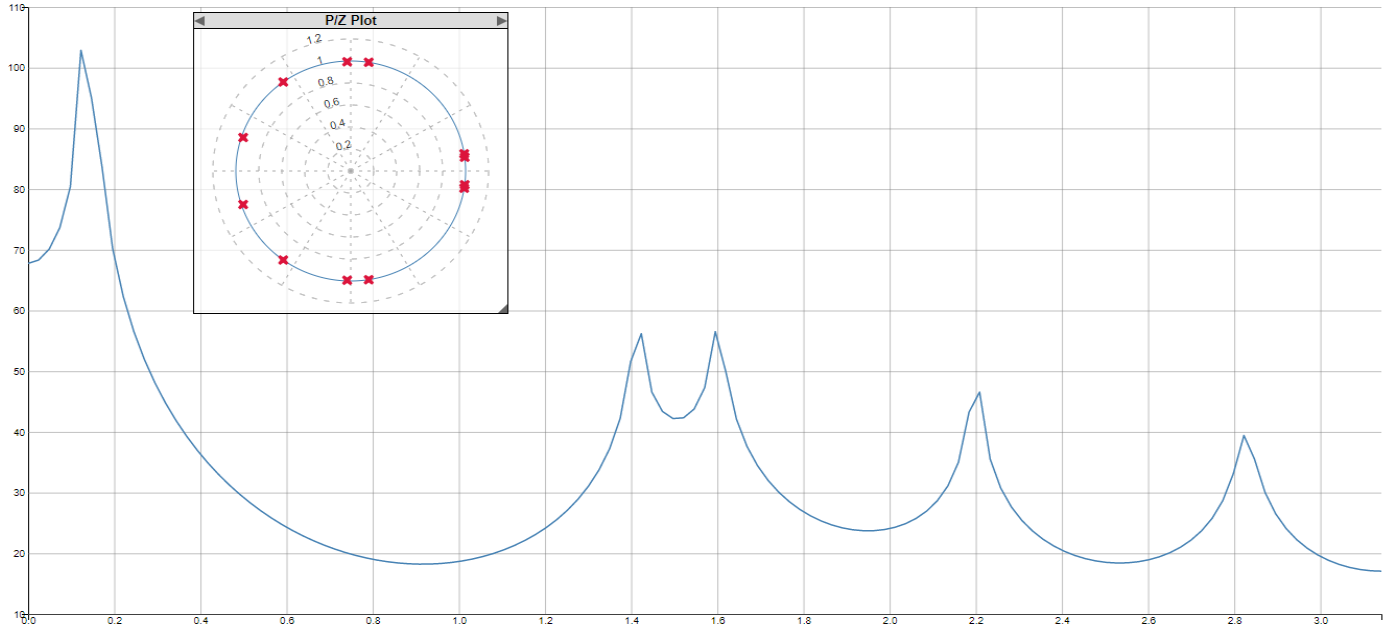
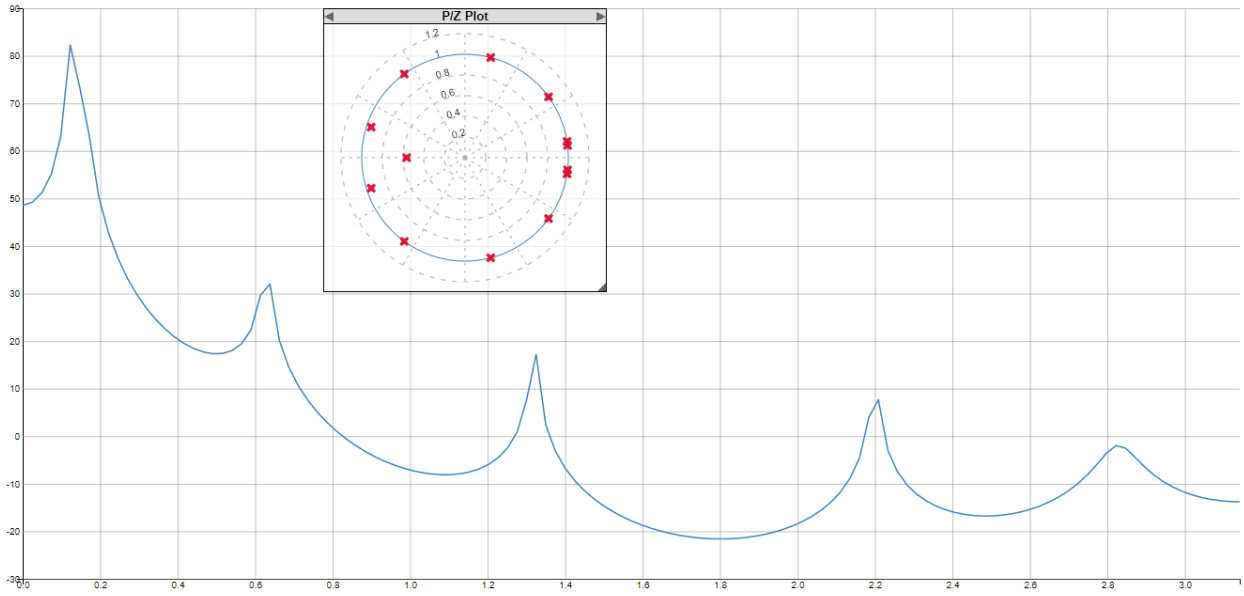
```
diary on
format compact
%Johnny Li
%EEL3135 Fall 2018
%Lab 7 Part 2

%2.1

%Mimic the following frequency responses to create the vowels ee, ah, oh,
%and oo.
%ee
```



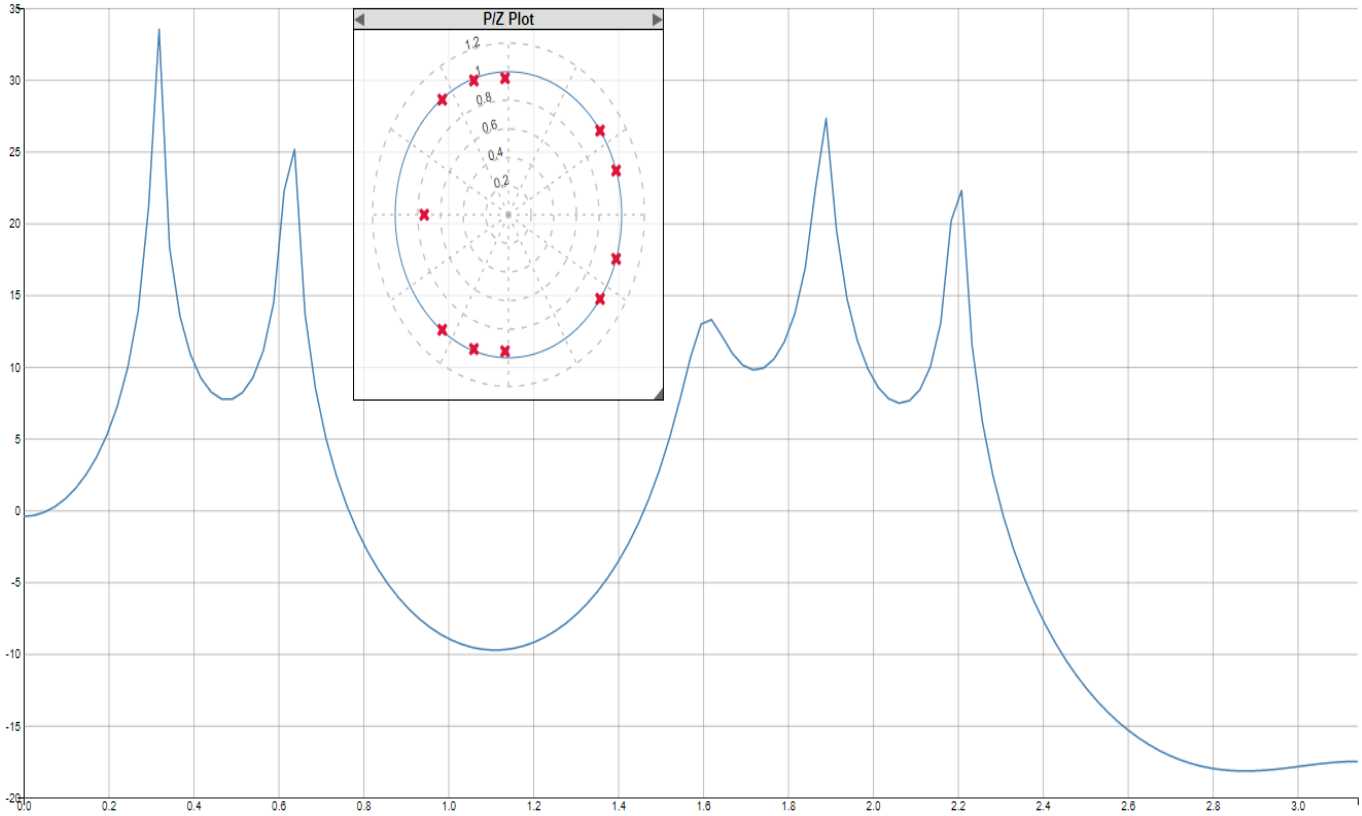

```
B = [ 1, 0.01401]
A = [1, -1.16094, 0.21350, -0.95847, -0.16852, 0.59457, 0.98926, 0.63387, -
0.29644, -0.85340, 0.12108, -1.05481, 0.95669]
zeros = []
poles = [0.98769+0.15643i, 0.98769-0.15643i, 0.15643+0.98769i, 0.15643-
0.98769i, -0.03119+0.99251i, -0.03119-0.99251i, -0.58779+0.80902i, -0.58779-
0.80902i, -0.93679+0.30438i, -0.93679-0.30438i, 0.99211+0.12533i, 0.99211-
0.12533i]

%Given Sample frequency
fs = 10000;
%Given Frequency
f=150;
%Time Interval, 1 second long
tt=0:1/fs:1;
%Storage/Initial value
glottal=0;
%Loop harmonics
for i=1:30
    %Function
    glottal=cos(2*pi*f*tt*i)+glottal;
end
```

```
%Filter coefficients from GUI
B = [ 1, 0.01401];
A = [1, -1.16094, 0.21350, -0.95847, -0.16852, 0.59457, 0.98926, 0.63387, -
0.29644, -0.85340, 0.12108, -1.05481, 0.95669];

%Filter
s= filter(B,A,glottal);

%For clipping
s=s/(max(abs(s)));

%Create autofile
audiowrite('ee.wav',s,fs);

%oo
```



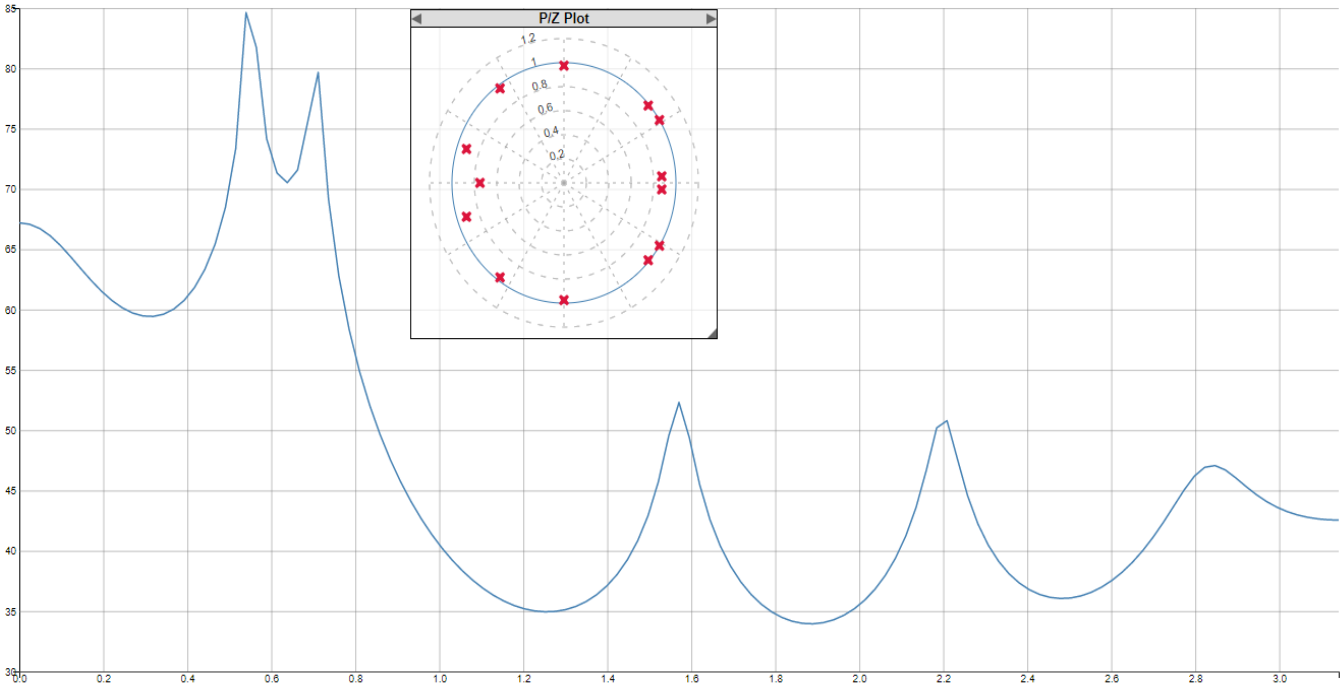

```
B = [1]
A = [1, -2.51947, 1.23002, 1.44335, -1.07590, -0.81458, 0.96112, 0.54177, -
1.57301, -0.14742, 2.49859, -1.46894, -0.58709, 0.51529]
zeros = []
poles = [0.98769+0.15643i, 0.98769-0.15643i, 0.80902+0.58779i, 0.80902-
0.58779i, 0.24869+0.96858i, 0.24869-0.96858i, -0.58779+0.80902i, -0.58779-
0.80902i, -0.90826+0.29511i, -0.90826-0.29511i, -0.565, 0.99288+0.11910i,
0.99288-0.11910i]

%Given Sample frequency
fs = 10000;
%Given Frequency
f=150;
%Time Interval, 1 second long
tt=0:1/fs:1;
%Storage/Initial value
glottal=0;
%Loop harmonics
for i=1:20
    %Function
```

```
    glottal=cos(2*pi*f*tt*i)+glottal;
end

%Filter coefficients from GUI
B = [1];
A = [1, -2.51947, 1.23002, 1.44335, -1.07590, -0.81458, 0.96112, 0.54177, -
1.57301, -0.14742, 2.49859, -1.46894, -0.58709, 0.51529];

%Filter
s= filter(B,A,glottal);

%For clipping
s=s/(max(abs(s)));

%Create autofile
audiowrite('oo.wav',s,fs);

%oh
```




```
B = [1]
A = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762, -
0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232]
zeros = []
poles = [0.98769+0.15643i, 0.98769-0.15643i, 0.15643+0.98769i, 0.15643-
0.98769i, -0.03119+0.99251i, -0.03119-0.99251i, -0.58779+0.80902i, -0.58779-
0.80902i, -0.90826+0.29511i, -0.90826-0.29511i, 0.99002+0.14090i, 0.99002-
0.14090i]

%Given Sample frequency
```

```
fs = 10000;
%Given Frequency
f=150;
%Time Interval, 1 second long
tt=0:1/fs:1;
%Storage/Initial value
glottal=0;
%Loop harmonics
for i=1:20
    %Function
    glottal=cos(2*pi*f*tt*i)+glottal;
end

%Filter coefficients from GUI
B = [1];
A = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762, -
0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232];

%Filter
s= filter(B,A,glottal);

%For clipping
s=s/(max(abs(s)));

%Create autofile
audiowrite('oo.wav',s,fs);

%ah
B = [1]
A = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762, -
0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232]
zeros = []
poles = [0.87498+0.00550i, 0.87498-0.00550i, 0.85264+0.52250i, 0.85264-
0.52250i, 0.75280+0.64295i, 0.75280-0.64295i, 0.975i, -0.975i, -
0.57074+0.78556i, -0.57074-0.78556i, -0.87022+0.28275i, -0.87022-0.28275i, -
0.75]
```




```
%Given Sample frequency
fs = 10000;
%Given Frequency
f=150;
%Time Interval, 1 second long
tt=0:1/fs:1;
%Storage/Initial value
glottal=0;
%Loop harmonics
for i=1:20
    %Function
    glottal=cos(2*pi*f*tt*i)+glottal;
end

%Filter coefficients from GUI
B = [1];
A = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762, -
0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232];

%Filter
s= filter(B,A,glottal);

%For clipping
s=s/(max(abs(s)));

%Create autofile
audiowrite('ah.wav',s,fs);

%2.2

type soundg
```

```
%SOUNDG Mimic the following frequency responses to create the vowels ee, ah,
%oh, and oo.
%Filter this white noise through your five vowel filters (eh, ee, ah, oh,
%and oo) to create five whispered vowel sounds.
%Script for 2.2.

%Given Sample frequency
fs = 10000;
%Given Frequency
f=150;
%Time Interval, 1 second long
tt=0:1/fs:1;
%Storage/Initial value
glottal=0;
%Loop harmonics
for i=1:20
    %Function
    glottal=cos(2*pi*f*tt*i)+glottal;
end

%Random filter noise
n = randn(1,10000);

%Filter coefficients from GUI
%eh
B1 = [1, 0.61536, 0.20421, 0.64225, 0.62603, 0.50200, 0.73851];
A1 = (1);
%ee
B2 = [ 1, 0.01401];
A2 = [1, -1.16094, 0.21350, -0.95847, -0.16852, 0.59457, 0.98926, 0.63387, -
0.29644, -0.85340, 0.12108, -1.05481, 0.95669];
%oo
B3 = [1];
A3 = [1, -2.51947, 1.23002, 1.44335, -1.07590, -0.81458, 0.96112, 0.54177, -
1.57301, -0.14742, 2.49859, -1.46894, -0.58709, 0.51529];
%oh
B4 = [1];
A4 = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762,
-0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232];
%ah
B5 = [1];
A5 = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762,
-0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232];

%Filter
s1= filter(B1,A1,n); %eh
s2= filter(B2,A2,n); %ee
s3= filter(B3,A3,n); %oo
s4= filter(B4,A4,n); %oh
s5= filter(B5,A5,n); %oh

%For clipping
s1=s1/(max(abs(s1)));
s2=s2/(max(abs(s2)));
s3=s3/(max(abs(s3)));
```

```
s4=s4/(max(abs(s4)));
s5=s5/(max(abs(s5)));

%Signal
s = [s1 s2 s3 s4 s5];
s=s/(max(abs(s)));

%Create autofile
audiowrite('ehw.wav',s1,fs);
audiowrite('eew.wav',s2,fs);
audiowrite('oow.wav',s3,fs);
audiowrite('ohw.wav',s4,fs);
audiowrite('ahw.wav',s5,fs);
audiowrite('sw.wav',s,fs);

%2.3

%Play the correctly-voweled, correctly-timed, all-three-voices-added-
%together version of the Barukh Fugue.
%Load Barukh fugue
load('barukh_fugue.mat')

type play_songg

function song = play_songg(theVoices)
%PLAY_SONG Construct the three voices in the Barukh Fugue and add them
%together. Add the five voices, from better_fugue.mat, together to produce
%the Better Fugue. Synthesisze the bach_fugue.
%Play the correctly-voweled, correctly-timed, all-three-voices-added-
%together version of the Barukh Fugue.
%Function based on given code 2.3.

%{
    PLAYSONG: Produce a sinusoidal waveform containing the combination of
    the different notes in theVoices
    Input Args:
        theVoices: structure contains noteNumbers, durations, and
        startpulses vectors for multiple voices of a song.
    Output:
        song: vector that represents discrete-time version of a musical
        waveform
    Usage:
        song = playSong()
%}
%load barukh_fugue.mat
%Define variables
%Frequency
fs = 8000;
%Beat per minute->beats per second->second per beats->second per pulse
%Given Code
beats_per_minute  = 120;
beats_per_second  = beats_per_minute / 60;
seconds_per_beat  = 1 / beats_per_second;
%spp = seconds_per_beat / 4;
%seconds per pulse, theVoices is measured in pulses with 4 pulses per beat
```

```
%Set spp to 0.15 for better fugue
spp=0.15;

%Length of voices
numV=length(theVoices);
%Length of notes
numN=length(theVoices(numV).noteNumbers);
%Final start pulse
fsp=theVoices(numV).startPulses(numN);
%Final durations
fd=theVoices(numV).durations(numN);

%Get Max value in theVoices
M=0;
for a=1:numV
    for b=1:length(theVoices(a).durations)
        d=theVoices(a).durations(a);
        st=theVoices(a).startPulses(b);
        if M<(d+st)
            M=d+st+1;
        end
    end
end

%Longest value in better
%song = zeros(1,ceil(M*spp*fs));
song = zeros(1,ceil((fsp+fd)*spp*fs));
%Create a vector of zeros with length equal to the total number of samples
%in the entire song

%Then add in the notes
for i = 1:length(theVoices)
    for j = 1:length(theVoices(i).noteNumbers)
        %set sound
        g = theVoices(i).vowels;
        %Set note
        note =
glottal_key_to_note(theVoices(i).noteNumbers(j),theVoices(i).durations(j)*spp
,g(2*j-1:2*j));


         %Create sinusoid of correct length to represent a single note
         locstart = theVoices(i).startPulses(j)*spp*fs;
         %Index of where note starts
         locend = locstart+length(note)-1;
         %Index of where note ends
         song(locstart:locend) = song(locstart:locend) + note;
    end
%For clipping
song=song/(max(abs(song)));
end
%Create autofile
audiowrite('Barukh_Fugueg.wav',song,fs);
end
```

```
type glottal_key_to_note

function xx = glottal_key_to_note(keynum, dur, sound)
%GLOTTAL_KEY_TO_NOTE Function created for lab7.1.
%Takes in a key number anda duration, to produce a glottal source signal of
%given duration with fundamental frequency corresponding to the desired %
%note.

%{
    KEY_TO_NOTE: Produce a sinusoidal waveform corresponding to a given
    piano key number.
    Input Args:
        X: amplitude (default = 1)
        keynum: number of the note on piano keyboard
        dur: duration of the note (in seconds)
    Output:
        xx: sinusoidal waveform of the note
%}

%Code Taken from key to note
%Smaple frequency
fs = 8000;
%Time interval
tt = 0:(1/fs):dur-1/fs;
%Given frequency function
freq = 110*2^((keynum-49)/12);

%Storage/Initial value
xx=zeros(size(tt));
%Loop harmonics
for i=1:20
    %Sinusoidal function
    xx = real(exp(i*j*2*pi*freq*tt))+xx;
end

%Filter coefficients from GUI
%eh
if sound == "eh"
B1 = [1, 0.61536, 0.20421, 0.64225, 0.62603, 0.50200, 0.73851];
A1 = (1);
xx = filter(B1,A1,xx);
end
%ee
if sound == "ee"
B2 = [ 1, 0.01401];
A2 = [1, -1.16094, 0.21350, -0.95847, -0.16852, 0.59457, 0.98926, 0.63387, -
0.29644, -0.85340, 0.12108, -1.05481, 0.95669];
xx = filter(B2,A2,xx);
end
%oo
if sound == "oo"
B3 = [1];
A3 = [1, -2.51947, 1.23002, 1.44335, -1.07590, -0.81458, 0.96112, 0.54177, -
1.57301, -0.14742, 2.49859, -1.46894, -0.58709, 0.51529];
xx = filter(B3,A3,xx);
```

```
end
%oh
if sound == "oh"
B3 = [1];
A3 = [1, -2.51947, 1.23002, 1.44335, -1.07590, -0.81458, 0.96112, 0.54177, -
1.57301, -0.14742, 2.49859, -1.46894, -0.58709, 0.51529];
xx = filter(B4,A4,xx);
end
%ah
if sound == "ah"
B5 = [1];
A5 = [1, -1.32894, -0.20644, 0.86079, 0.19974, -0.02983, -0.38534, -0.20762,
-0.19947, 0.30462, 0.87338, -0.83883, -0.38190, 0.42232];
xx = filter(B5,A5,xx);
end

end
```