Taylor Rembos
EEL 3135
Lab 7 Part 2

Lab 7: Poles and Zeros – Vowel Synthesis

**Lab Part 2**

**2.1 Four More Vowel Sounds**

Now, using as many poles and zeros as you'd like, use the GUI and the poles-and-zeros fundamentals that you've learned to mimic the following frequency responses to create the vowels "ee", "ah", "oh", and "oo".
All the modeled magnitude response plots below have sampling frequency 10000Hz. You need to convert the important "peaks" and "dips" location to normalized radiant frequency to put into your GUI.
For each vowel:

· Submitscreenshotsofyour GUI(magnituderesponseandpoles/zerosplot)
· Submitthefiltercoefficients $a_k$ and $b_k$ aswellasthepoleandzerolocationsasvectors.
· Make sure to either get each vowel sound checked off by a TA or submit .wav files.

Hint: if vowel sounds are off, consider increasing the number of harmonics when creating the glottal source signal. The vowels sound more distinguishable if they are played one after another.

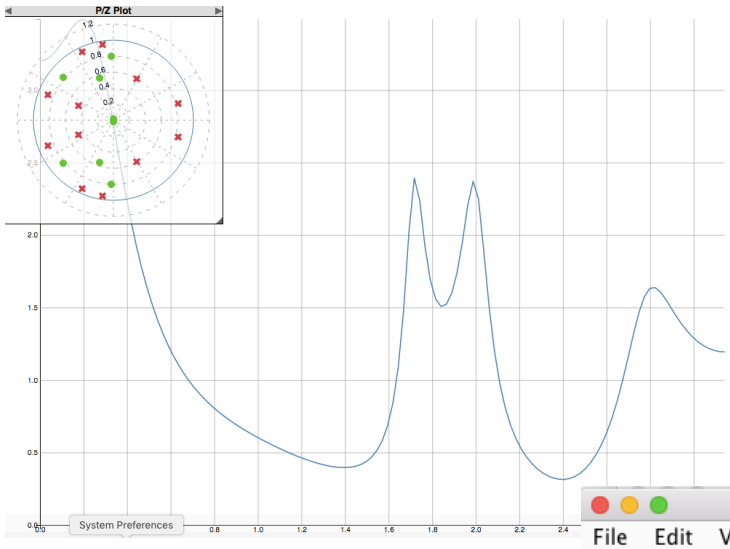
```
sound = sound/(max (abs (sound))); %% anti clipping xx soundsc(sound,fs);
audiowrite('Bach_trumpet.wav',sound,fs);
```

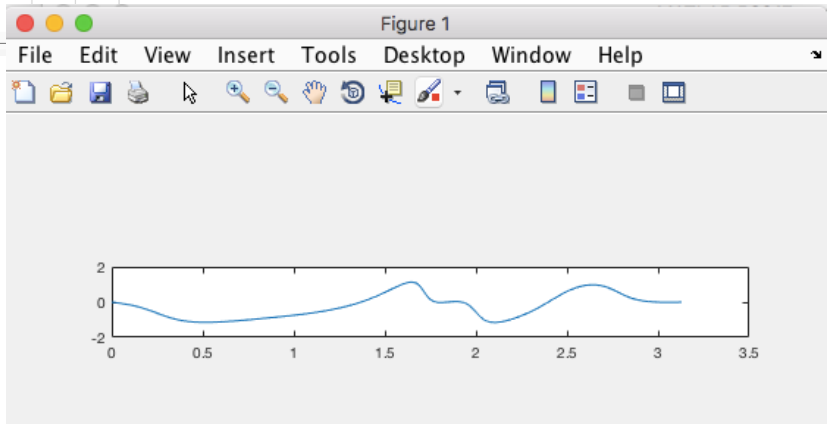
"ee"

B = [1, 1.65455, 2.15074, 1.72484, 1.15570, 0.41024, 0.13474, 0.00014, 0.00004]
A = [1, 1.36364, 1.22992, 0.14221, -0.89218, -0.86587, -0.26072, 0.15974, 0.33859, 0.10198, 0.10887, 0.11116, 0.03422]
zeros = [-0.02727+0.80000i, -0.02727-0.80000i, -0.62727+0.53636i, -0.62727-0.53636i, 0.01818i, -0.01818i, -0.17273+0.52727i, -0.17273-0.52727i]
poles = [0.80909+0.20909i, 0.80909-0.20909i, -0.39091+0.85455i, -0.39091-0.85455i, 0.29091-0.51818i, 0.29091+0.51818i, -0.13636+0.94545i, -0.13636-0.94545i, -0.43636+0.18182i, -0.43636-0.18182i, -0.81818-0.31818i, -0.81818+0.31818i]

```
>> f = 150;
>> ees = 0;
>> ff = 30*f*2;
>> t = 0:1/ff:1;
>> for i = 1:30
ees = cos(2*pi*150*t*i) + ees;
```

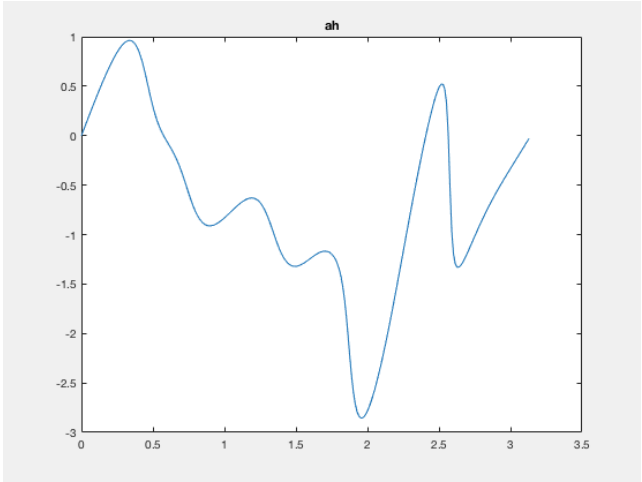
```
end

>> bk = [1, 1.65455, 2.15074, 1.72484, 1.15570, 0.41024, 0.13474, 0.00014, 0.00004];
>> ak = [1, 1.36364, 1.22992, 0.14221, -0.89218, -0.86587, -0.26072, 0.15974, 0.33859,
0.10198, 0.10887, 0.11116, 0.03422];
>> soundEE = filter(bk,ak,ees);
>> soundsc(soundEE, 10000)
>> [H,w] = freqz(bk,ak,256);
>> audiowrite('ee.wav', soundEE, 10000)
Warning: Data clipped when writing file.
> In audiowrite>clipInputData (line 396)
  In audiowrite (line 176)
>> subplot(5,1,2)
>> title('ee');
>> plot(w,angle(H))
>> soundEE = soundEE/(max (abs (soundEE)));
>> soundsc(soundEE, 10000)
>> audiowrite('ee.wav', soundEE, 10000)
```

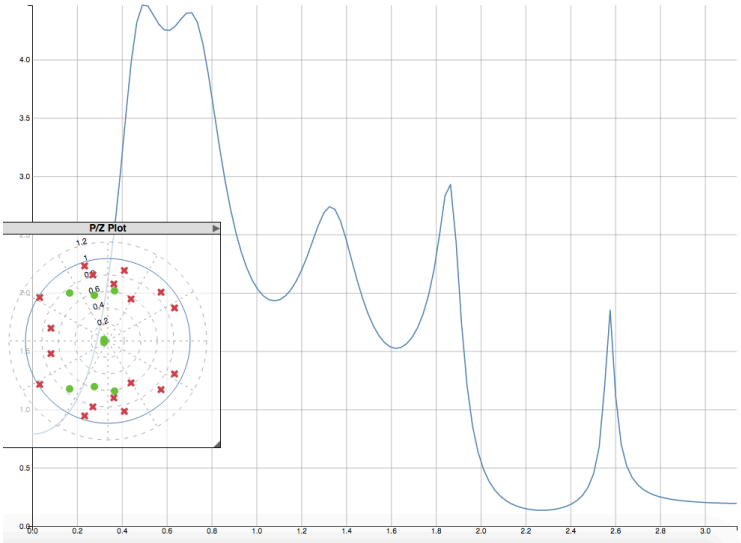


“ah”
B = [1, 1.18182, 1.46521, 0.89660, 0.62784, 0.20747, 0.08532, 0.00672, 0.00017]
A = [1, -0.05455, 1.27562, 0.01957, 0.85173, 0.29224, 0.88121, 0.11580, 0.99986, 0.04734, 0.44555, 0.37042, 0.08613, 0.34621, 0.09579, 0.07014, 0.09327, -0.01199, 0.02310]
zeros = [-0.46364+0.58182i, -0.46364-0.58182i, -0.04545+0.01818i, -0.04545-0.01818i, 0.08182+0.60909i, 0.08182-0.60909i, -0.16364-0.55455i, -0.16364+0.55455i]
poles = [-0.82727+0.52727i, -0.82727-0.52727i, 0.80909+0.40000i, 0.80909-0.40000i, -0.69091-0.15455i, -0.69091+0.15455i, 0.64545+0.59091i, 0.64545-0.59091i, 0.20000+0.85455i, 0.20000-0.85455i, 0.07273-0.69091i, 0.07273+0.69091i, 0.28182+0.50909i, 0.28182-0.50909i, -0.28182+0.90909i, -0.28182-0.90909i, -0.18182+0.80000i, -0.18182-0.80000i]

```
>> bk = [1, 1.18182, 1.46521, 0.89660, 0.62784, 0.20747, 0.08532, 0.00672, 0.00017];
>> ak = [1, -0.05455, 1.27562, 0.01957, 0.85173, 0.29224, 0.88121, 0.11580, 0.99986, 0.04734, 0.44555, 0.37042, 0.08613, 0.34621, 0.09579, 0.07014, 0.09327, -0.01199, 0.02310];
>> soundAH = filter(bk,ak,ees);
>> soundAH = soundAH/(max (abs (soundAH)));
>> soundsc(soundAH, 10000)
>> audiowrite('ah.wav', soundAH, 10000)
>> [H,w] = freqz(bk,ak,256);
>> plot(w,angle(H))
>> title('ah')
```

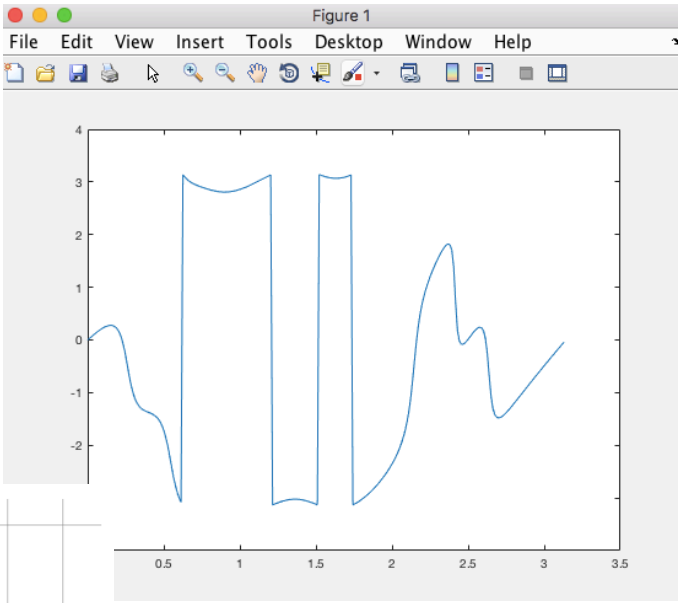


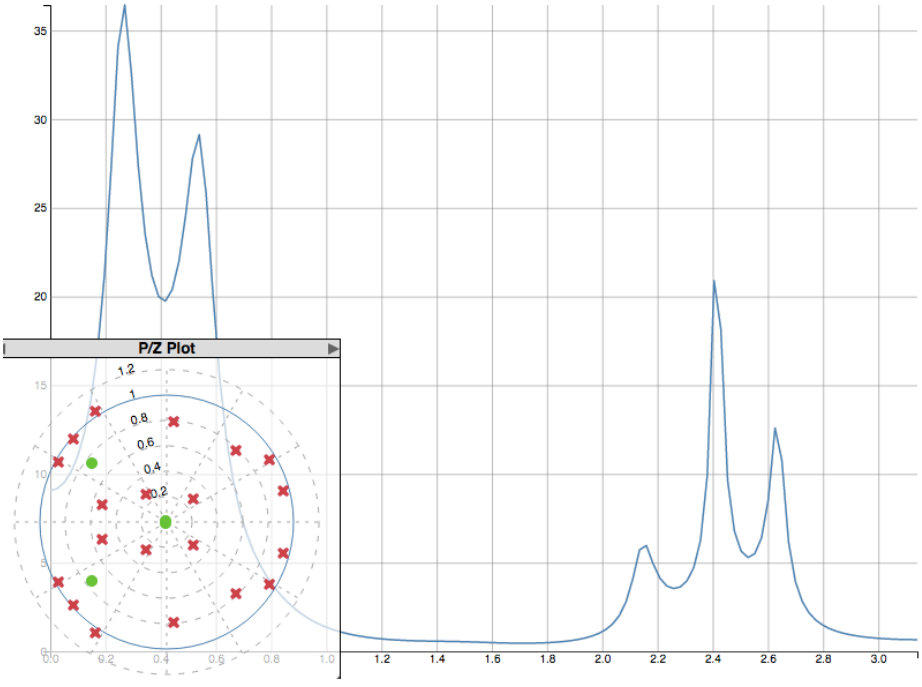



“oh”

B = [1, 1.43636, 3.74256, 3.59668, 1.68561, 0.03005, 0.00027]
A = [1, 0.58182, -0.79149, -1.59400, 0.73366, 1.44407, 0.37692, -1.02533, -0.06723, 0.06560, 0.14626, 0.15824, 0.24815, -0.22618, -0.04458, 0.19642, 0.07097, -0.00589, -0.00030, 0.00059, 0.00050]
zeros = [-0.11818+1.69091i, -0.11818-1.69091i, -0.00909-0.00909i, -0.00909+0.00909i, -0.59091+0.46364i, -0.59091-0.46364i]
poles = [-0.50909+0.13636i, -0.50909-0.13636i, 0.05455-0.79091i, 0.05455+0.79091i, -0.16364+0.21818i, -0.16364-0.21818i, -0.56364+0.87273i, -0.56364-0.87273i, 0.54545-0.56364i, 0.54545+0.56364i, 0.20909+0.18182i, 0.20909-0.18182i, -0.73636-0.65455i, -0.73636+0.65455i, 0.80909-0.49091i, 0.80909+0.49091i, 0.91818-0.24545i, 0.91818+0.24545i, -0.85455+0.47273i, -0.85455-0.47273i]

```
>> bk = [1, 1.43636, 3.74256, 3.59668, 1.68561, 0.03005, 0.00027];
>> ak = [1, 0.58182, -0.79149, -1.59400, 0.73366, 1.44407, 0.37692, -1.02533, -0.06723, 0.06560, 0.14626, 0.15824, 0.24815, -0.22618, -0.04458, 0.19642, 0.07097, -0.00589, -0.00030, 0.00059, 0.00050];
>> soundOH = filter(bk,ak,ees);
>> soundOH = filter(bk,ak,ees);
>> soundsc(soundOH, 10000)
>> [H,w] = freqz(bk,ak,256);
>> plot(w,angle(H))
>> plot(w,angle(H))
>> soundOH = soundOH/(max (abs (soundOH)));
>> audiowrite('oh.wav', soundOH, 10000)
```

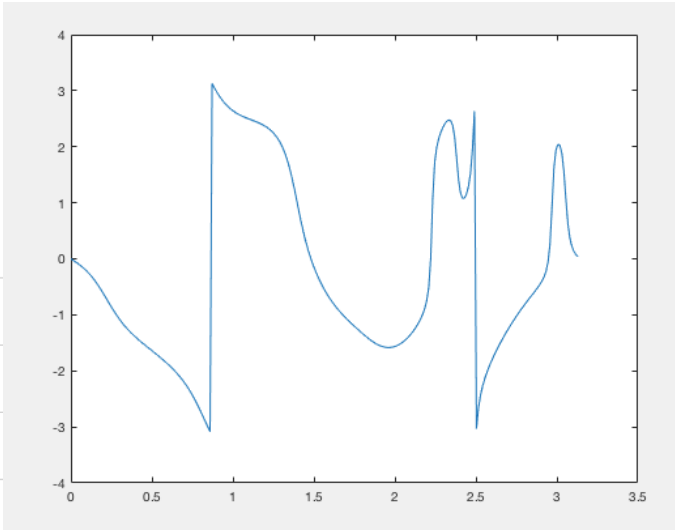


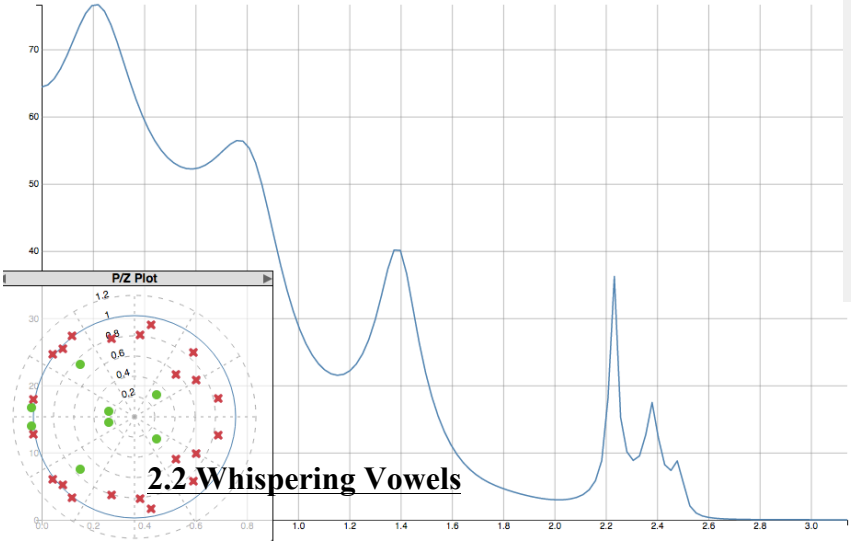



“oo”

B = [1, 6.38182, 19.90785, 42.41705, 66.03837, 71.34558, 49.75800, 20.37036, 3.98405, 0.45732, 0.45842, 0.19520, 0.02370]
A = [1, 1.43636, 1.17719, 0.20719, 0.25017, -0.06524, -0.16083, 0.13999, 0.13666, -0.12765, -0.32539, 0.14596, 0.17171, 0.15303, 0.19034, -0.11531, 0.10767, -0.10307, 0.18177, -0.09699, 0.12644, -0.08140, 0.03486]
zeros = [-1.01818+0.09091i, -1.01818-0.09091i, -0.11818+1.69091i, -0.11818-1.69091i, -0.25455+0.05455i, -0.25455-0.05455i, 0.21818+0.21818i, 0.21818-0.21818i, -1.48182-0.06364i, -1.48182+0.06364i, -0.53636+0.51818i, -0.53636-0.51818i]
poles = [0.05455+0.80909i, 0.05455-0.80909i, -0.70909+0.67273i, -0.70909-0.67273i, 0.82727-0.18182i, 0.82727+0.18182i, 0.58182-0.63636i, 0.58182+0.63636i, -0.61818-0.80000i, -0.61818+0.80000i, 0.16364-0.90909i, 0.16364+0.90909i, -0.80909-0.61818i, -0.80909+0.61818i, 0.60909+0.36364i, 0.60909-0.36364i, -1+0.17273i, -1-0.17273i, 0.40909+0.41818i, 0.40909-0.41818i, -0.22727+0.77273i, -0.22727-0.77273i]

```
>> bk = [1, 6.38182, 19.90785, 42.41705, 66.03837, 71.34558, 49.75800, 20.37036, 3.98405, 0.45732, 0.45842, 0.19520, 0.02370];
>> ak = [1, 1.43636, 1.17719, 0.20719, 0.25017, -0.06524, -0.16083, 0.13999, 0.13666, -0.12765, -0.32539, 0.14596, 0.17171, 0.15303, 0.19034, -0.11531, 0.10767, -0.10307, 0.18177, -0.09699, 0.12644, -0.08140, 0.03486];
>> soundOO = filter(bk,ak,ees);
>> soundOO = soundOO/(max (abs (soundOO)));
>> soundsc(soundOO, 10000)
>> [H,w] = freqz(bk,ak,256);
>> plot(w,angle(H))
>> soundOO = soundOO/(max (abs (soundOO)));
>> audiowrite('oo.wav', soundOH, 10000)
```




## 2.2 Whispering Vowels

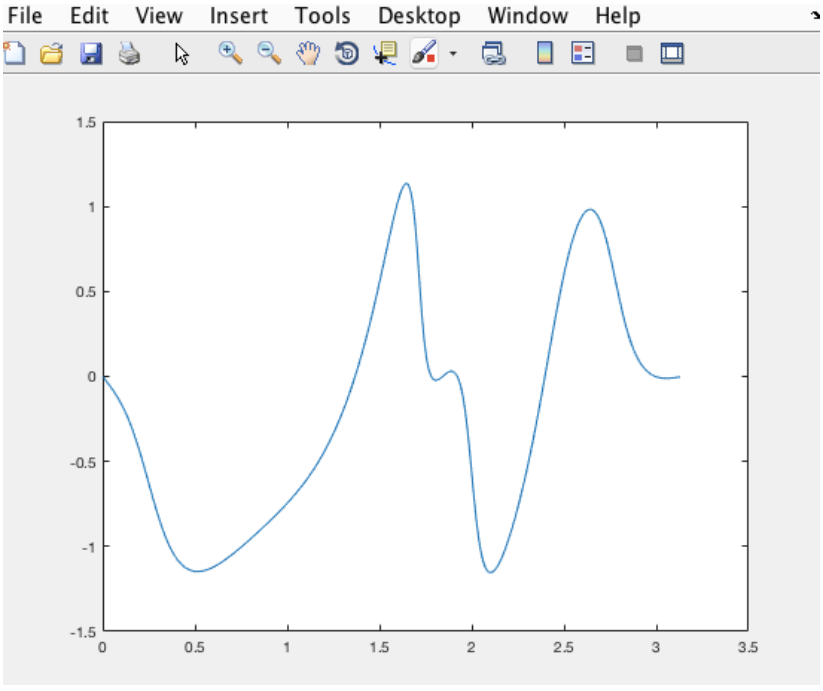
Use MATLAB's randn() command, which creates random numbers between 0 and 1, to create a vector with the same length as your glottal source signal vector, containing random numbers between zero and one. This signal is called "white noise", because it is broadband in frequency.
Filter this white noise through your five vowel filters – "eh", "ee", "ah", "oh", and "oo" – to create five whispered vowel sounds. Play them for your TA to get checked off.

Ee_whisper.wav
Ah_whisper.wav
Oh_whisper.wav
Oo_whisper.wav

```
>> f = 150;
>> ees = 0;
>> ff = 30*f*2;
>> t = 0:1/ff:1;
>> for i = 1:30
ees = cos(2*pi*150*t*i) + ees;
end
```

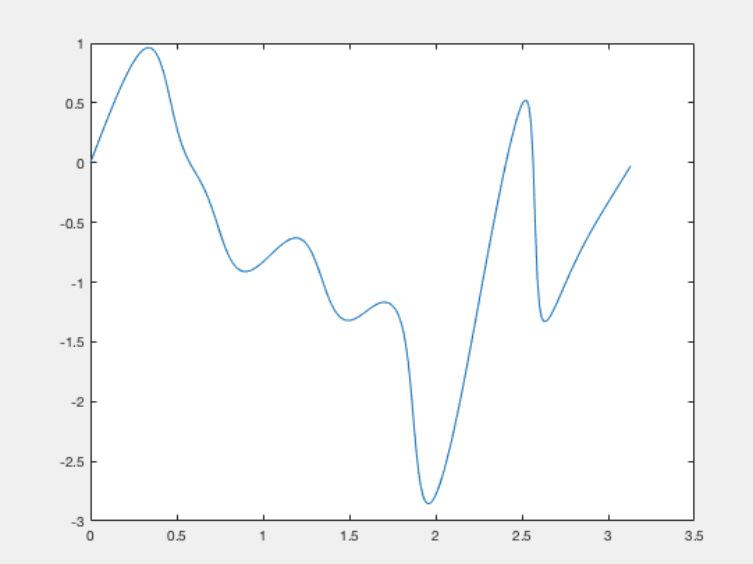
"ee"

```
>> ees = rand(1,1000);
>> bk = [1, 1.65455, 2.15074, 1.72484, 1.15570, 0.41024, 0.13474, 0.00014, 0.00004];
>> ak = [1, 1.36364, 1.22992, 0.14221, -0.89218, -0.86587, -0.26072, 0.15974, 0.33859,
0.10198, 0.10887, 0.11116, 0.03422];
>> soundEE = filter(bk,ak,ees);
>> soundsc(soundEE, 10000);
>> [H,w] = freqz(bk,ak,256);
>> soundEE = soundEE/(max (abs (soundEE)));
>> audiowrite('ee_whisper.wav',soundEE,10000)
>> plot(w,angle(H))
```



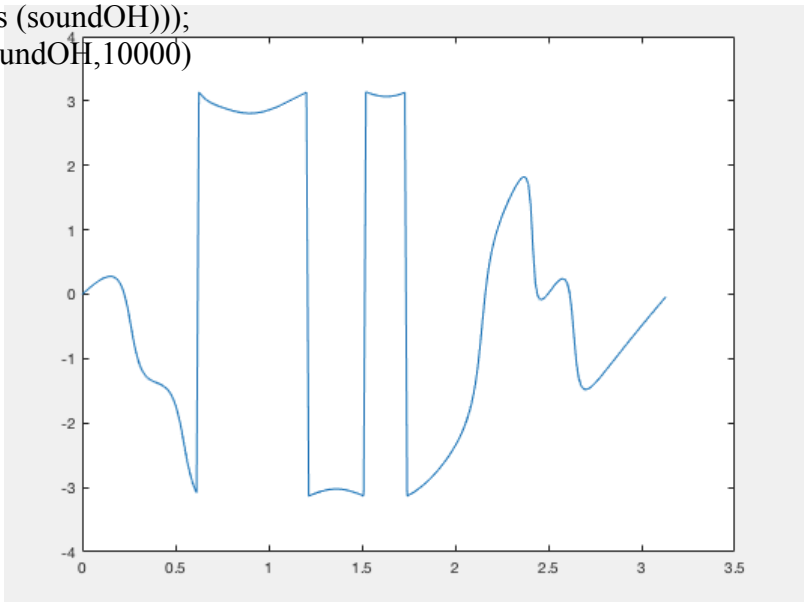
“ah”

```
>> bk = [1, 1.18182, 1.46521, 0.89660, 0.62784, 0.20747, 0.08532, 0.00672, 0.00017];
>> ak = [1, -0.05455, 1.27562, 0.01957, 0.85173, 0.29224, 0.88121, 0.11580, 0.99986,
0.04734, 0.44555, 0.37042, 0.08613, 0.34621, 0.09579, 0.07014, 0.09327, -0.01199,
0.02310];
>> soundAH = filter(bk,ak,ees);
>> soundsc(soundAH, 10000);
>> [H,w] = freqz(bk,ak,256);
>> soundAH = soundAH/(max (abs (soundAH)));
>> audiowrite('ah_whisper.wav',soundAH,10000)
>> plot(w,angle(H))
```



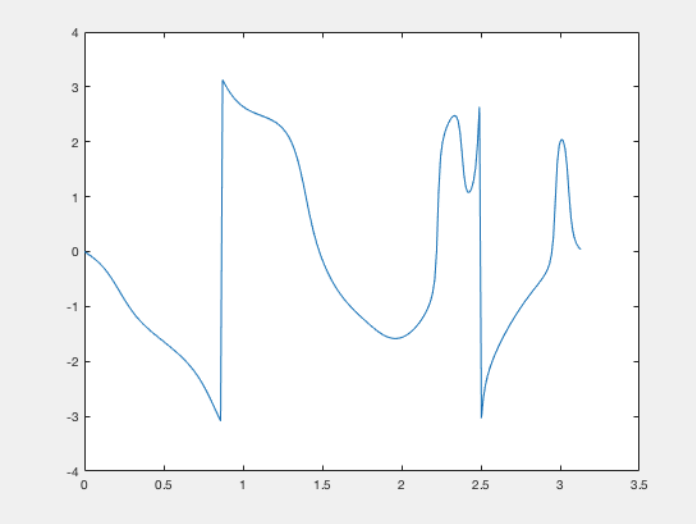

“oh”

```
>> bk = [1, 1.43636, 3.74256, 3.59668, 1.68561, 0.03005, 0.00027];
>> ak = [1, 0.58182, -0.79149, -1.59400, 0.73366, 1.44407, 0.37692, -1.02533, -0.06723,
0.06560, 0.14626, 0.15824, 0.24815, -0.22618, -0.04458, 0.19642, 0.07097, -0.00589, -
0.00030, 0.00059, 0.00050];
>> soundOH = filter(bk,ak,ees);
>> soundsc(soundOH, 10000);
>> [H,w] = freqz(bk,ak,256);
>> soundOH = soundOH/(max (abs (soundOH)));
>> audiowrite('oh_whisper.wav',soundOH,10000)
>> plot(w,angle(H))
```



“oo”

```
>> bk = [1, 6.38182, 19.90785, 42.41705, 66.03837, 71.34558, 49.75800, 20.37036,
3.98405, 0.45732, 0.45842, 0.19520, 0.02370];
>> ak = [1, 1.43636, 1.17719, 0.20719, 0.25017, -0.06524, -0.16083, 0.13999, 0.13666, -
0.12765, -0.32539, 0.14596, 0.17171, 0.15303, 0.19034, -0.11531, 0.10767, -0.10307,
0.18177, -0.09699, 0.12644, -0.08140, 0.03486];
>> soundOO = filter(bk,ak,ees);
>> soundsc(soundOO, 10000);
>> [H,w] = freqz(bk,ak,256);
>> soundOO = soundOO/(max (abs (soundOO)));
>> audiowrite('oo_whisper.wav',soundOO,10000)
>> plot(w,angle(H))
```



## 2.3 Reconstruction of a Voweled Fugue

We could define three separate arrays for one melody, stored as part ofthe structuremelody:

melody.noteNumbers

melody.durations
melody.startPulses
= [40 42 44 45 47 49 51 52];
= [1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5];
= [1 3 5 7 9 11 13 15];
We now introduce a new array for that melody:

melody.vowels = [‘eh’ ‘ee’ ‘ah’ ‘oh’ ‘oo’ ‘oh’ ‘ah’ ‘ee’];

The vowels array is an array of characters representing the vowels to be played at each note, two characters at a time. It’ll actually be stored in this form:

melody.vowels = [‘eheeahohooohahee’]; The vowel matching a particular note n, will be specified by melody.vowels((2*n- 1):(2*n)). MATLAB has a few functions, such as the strcmp() function, that can be used to determine the current vowel. Load the barukh_fugue.mat file, containing the theVoices structure. You’ll notice that it now has four fields: noteNumbers, durations, startPulses, and vowels.

Play the correctly-voweled, correctly-timed, all-three-voices-added-together version of the Barukh Fugue. Submit this as a .wav file.

playSong.m

```
function song = playSong(theVoices)
% PLAYSONG Produce a sinusoidal waveform containing the combination
ofthe different notes in theVoices
% usage: song = playSong ()
% song = the output sinusoidal waveform
    load('bach_fugue.mat')
    fs = 25000;
    spp = 0.25 %%% seconds per pulse, theVoices is measured in pulses
with 4 pulsesper beat

    % Create a vector of zeros with length equal to the total number
ofsamples in the entire song
    song =
zeros(1,4*spp*fs*(theVoices(2).durations(length(theVoices(2).durations)
) + (theVoices(2).startPulses(length(theVoices(2).durations))))); %%%
vector of zeros

    % Then add in the notes
    for j = 1:2
        for i = 1:length(theVoices) % Cycle through each set of notes
        % Convert data arrays to appropriate units
            for j = 1:length(theVoices(i).noteNumbers) % Cycle
througheach note in aset
                keynum = theVoices(i).noteNumbers(j);
                note =
glottalkey2note(keynum,theVoices(i).durations(j).*spp,theVoices(i).vowe
ls((2*j-1):(2*j))); %%% create sinusoid of correct length to represent
a single note
                locstart = theVoices(i).startPulses(j)*(fs/4)+1; %%%
index of where note starts
                locend = locstart + length(note) - 1; %%%
                % index of where note ends
                song(locstart:locend) = song(locstart:locend) + note;
            end
        end
    end
    soundsc(song, fs)
    song = song/(max (abs(song))) %% anti clipping xx soundsc(song,fs);
    audiowrite('Bach_new.wav',song,fs)
    % Use audiowrite() to generate WAV file
end
```

**glottalkey2note.m**

```
function [ x ] = glottalkey2note(keynum, dur, vowels)
    % KEY2NOTE Produce a sinusoidal waveform corresponding to a given
piano key number
    % usage: xx = key2note (X, keynum, dur)
    % xx = the output sinusoidal waveform
    % X = complex amplitude for the sinusoid, X = A*exp(j*phi).
    % keynum = the piano keyboard number of the desired note
    % dur = the duration (in seconds) of the output note
    load('bach_fugue.mat');
    fs = 25000;
    tt = (1/fs):(1/fs):dur;
    freq = 220*(2^((keynum-49)/12));%<=============== fill in this line
    x = zeros(size(tt));
    for i = 0:30
        x = real(exp(j*i*2*pi*freq*tt)) + x; %<=============== fill in
this line
    end

    if strcmp(vowels, 'eh')
        B_eh = [ 1.15,   0.536667,  0.0111726,   0.487026, 0.826319,
0.390572,   0.639617 ];
        A_eh = [ 1 ];
        x = filter(B_eh,A_eh,x);
    end

    if strcmp(vowels, 'ee')
        B_ee = [1, 1.65455, 2.15074, 1.72484, 1.15570, 0.41024,
0.13474, 0.00014, 0.00004];
        A_ee = [1, 1.36364, 1.22992, 0.14221, -0.89218, -0.86587, -
0.26072, 0.15974, 0.33859, 0.10198, 0.10887, 0.11116, 0.03422];
        x = filter(B_ee,A_ee,x);
    end

    if strcmp(vowels, 'ah')
        B_ah = 3*[1, 1.18182, 1.46521, 0.89660, 0.62784, 0.20747,
0.08532, 0.00672, 0.00017];
        A_ah = [1, -0.05455, 1.27562, 0.01957, 0.85173, 0.29224,
0.88121, 0.11580, 0.99986, 0.04734, 0.44555, 0.37042, 0.08613, 0.34621,
0.09579, 0.07014, 0.09327, -0.01199, 0.02310];
        x = filter(B_ah,A_ah,x);
    end

    if strcmp(vowels, 'oh')
        B_oh = 10*[1, 1.43636, 3.74256, 3.59668, 1.68561, 0.03005,
0.00027];
        A_oh = [1, 0.58182, -0.79149, -1.59400, 0.73366, 1.44407,
0.37692, -1.02533, -0.06723, 0.06560, 0.14626, 0.15824, 0.24815, -
0.22618, -0.04458, 0.19642, 0.07097, -0.00589, -0.00030, 0.00059,
0.00050];
        x = filter(B_oh,A_oh,x);
    end

    if strcmp(vowels, 'oo')
        B_oo = 7*[1, 6.38182, 19.90785, 42.41705, 66.03837, 71.34558,
49.75800, 20.37036, 3.98405, 0.45732, 0.45842, 0.19520, 0.02370];
        A_oo = [1, 1.43636, 1.17719, 0.20719, 0.25017, -0.06524, -
```

```
0.16083, 0.13999, 0.13666, -0.12765, -0.32539, 0.14596, 0.17171,
0.15303, 0.19034, -0.11531, 0.10767, -0.10307, 0.18177, -0.09699,
0.12644, -0.08140, 0.03486];
        x = filter(B_oo,A_oo,x);
    end
end
```

(error I couldn't not solve would not let me save to .wav file successfully, sorry)