Taylor Rembos
EEL 3135
Lab 7 Part 1

Lab 7: Poles and Zeros – Vowel Synthesis

Lab Part 1

1.1 Modeling Vowel Production

In other words, all of the A_k are equal in the Fourier series of the glottal source signal. First, create a one-second-long glottal source signal, containing 7 nonzero harmonics (A -7 through A7), with fundamental frequency 150Hz. What is the minimum sampling frequency necessary to avoid aliasing?

```
>> f = 150; % freq in Hz
>> ff = 30*f*2;
>> t = 0:1/ff:1; % one second long
>> glottal = 0;
>> for i = 1:30
glottal = cos(2*pi*150*t*i) + glottal;
end
```

Min sampling frequency is 9000Hz ((150*pi / 2*pi) *2 * 30).

1.2 A Function to Play a Vocal Note

Write a MATLAB function (similar to your key2note function from Lab 3) that takes in a key number and a duration, to produce a glottal source signal of given duration with fundamental frequency corresponding to the desired note. Use a sampling frequency of 8000Hz:
function xx = glottalkey2note(keynum, dur)

glottalkey2note.m function

```
function [ xx ] = glottalkey2note( keynum, dur )
%1.2 Lab 7 Part 1
%   Takes in keynum and dur to produce glottal source sig
fs = 8000; % sampling freq
tt = (1/fs):(1/fs):dur; % for duration
freq = 220*(2^(keynum-49)/12); % human voice formula
xx = zeros(size(tt));
for i = 0:30
xx = real(exp(i*j*2*pi*freq*tt)) + xx; %output notes
end
```

1.3 Synthesize a Song – Mary Had a Bleating Lamb

Use glottalkey2note()to write a script, play_glottallamb.m, that plays a series of notes.
Use the following skeleton code to write your script:

```
% --------------play_lamb.m--------------- %
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
% NOTES: C D E F G
% Key #40 is middle-C
mary.durations = 0.25 * ones(1,length(mary.keys));
fs = 8000;
xx = zeros(1, sum(mary.durations)*fs + length(mary.keys));
n1 = 1;
for kk = 1:length(mary.keys)
        keynum = mary.keys(kk);
        tone = % <------- Fill in this line
        n2 = n1 + length(tone) - 1;
        xx(n1:n2) = xx(n1:n2) + tone; %<------- Insert the note
        n1 = n2 + 1;
end
soundsc(xx, fs)
```

Generate the sound and play it for a TA. Your TA will check you off, and you'll move on to the next section. This check-off isn't for credit – it's just to make sure you're on track. Plot the frequency-time spectrogram of Mary using one of plotspec, specgram, or spectrogram (or any other Matlab program you can think of).

play_glottallamb.m script

```
% --------------play_lamb.m--------------- %
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
% NOTES: C D E F G
% Key #40 is middle-C
mary.durations = 0.25 * ones(1,length(mary.keys));
fs = 8000;
xy = zeros(1, sum(mary.durations)*fs + length(mary.keys));
n1 = 1;
for kk = 1:length(mary.keys)
    keynum = mary.keys(kk);
    tone = glottalkey2note(keynum, mary.durations(kk));% <-------
glottal function
    n2 = n1 + length(tone) - 1;
    xy(n1:n2) = xy(n1:n2) + tone;
    n1 = n2 + 1;
end

soundsc(xy, fs)
plotspec(xy, 1/fs)
audiowrite('glottallamb.wav',xy,fs);
```

1.4 Synthesize a Song – Baaaaaaaach Taaaalks

Now, use your glottalkey2note() to synthesize your Bach Fugue from Lab 3. Submit a .wav file of your new Bach Fugue.

playSong.m script

```
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 = 8000;
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); %%%
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
```

1.5 Pole-Zero Plots

This is a two-dimensional plot of the -plane that shows the unit circle, the real and imaginary axes, and the position of the system's poles and zeros. Zeros are typically marked with an 'o', while poles are indicated with an 'x'. Sometimes, a location has multiple poles and zeros. In this case, a number is marked next to that location to indicate how many poles or zeros exist there. The above figure, for instance, shows four zeros (two conjugate pairs), two "trivial" poles at the origin, and one other conjugate pair of poles. Where in the complex plane can zeros and poles be placed to have the strongest influence on the magnitude response of the filter?

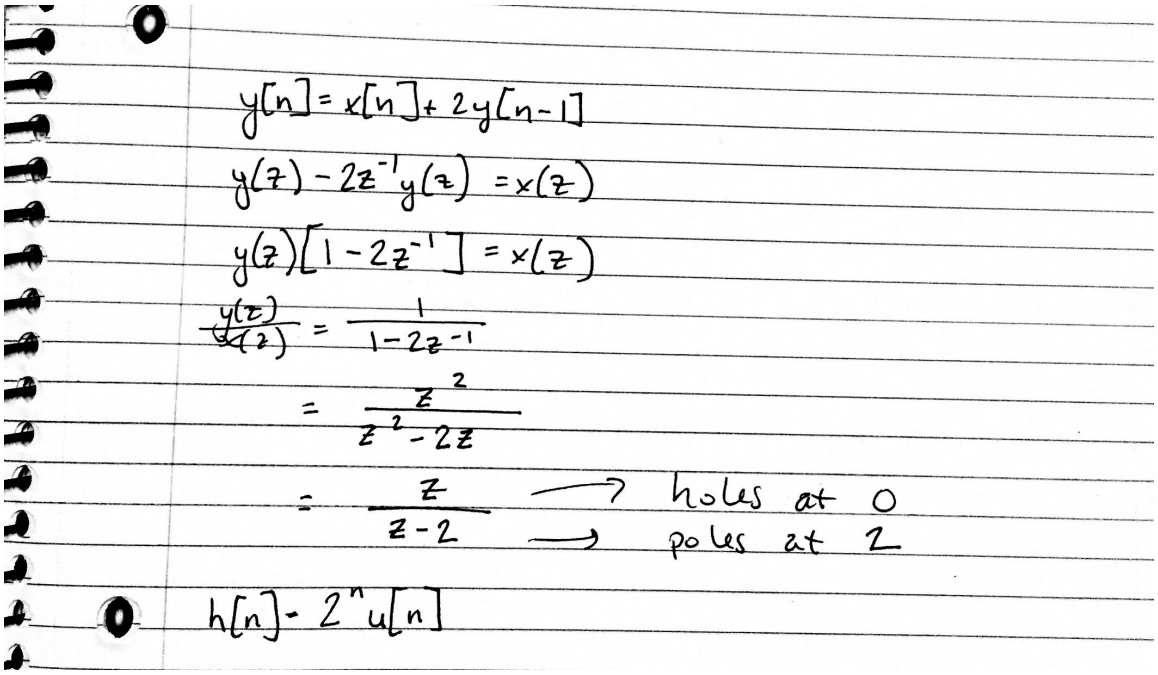
The places in the complex plane where zeros and poles are placed to have the strongest influence on the magnitude response of the filter are with the holes closer to each other on the unit circle and the poles closer to the origin.

1.6 Poles and Stability

Consider the filter:

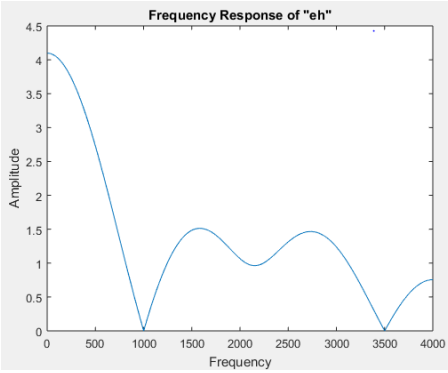
$y[n] = x[n] + 2y[n-1]$

What are the poles and zeros of this filter? What is this filter's impulse response?



1.7 Vowel Creation

a) The magnitude response plot below has sampling frequency of 8000Hz. There are six non- trivial zeros (including conjugates). Submit a table of the zeros location in normalized radiant frequency.



b) This filter represents the 'eh' sound. Based on the poles/zeros location in part (a), using the GUI, create an FIR filter with six nontrivial zeros that matches the following magnitude response. What are the filter coefficients b and a?

c) Submit a screenshot of your GUI (magnitude response + poles/zeros plots of the filter that you create). Your magnitude response plot must be in normalized radiant frequency.

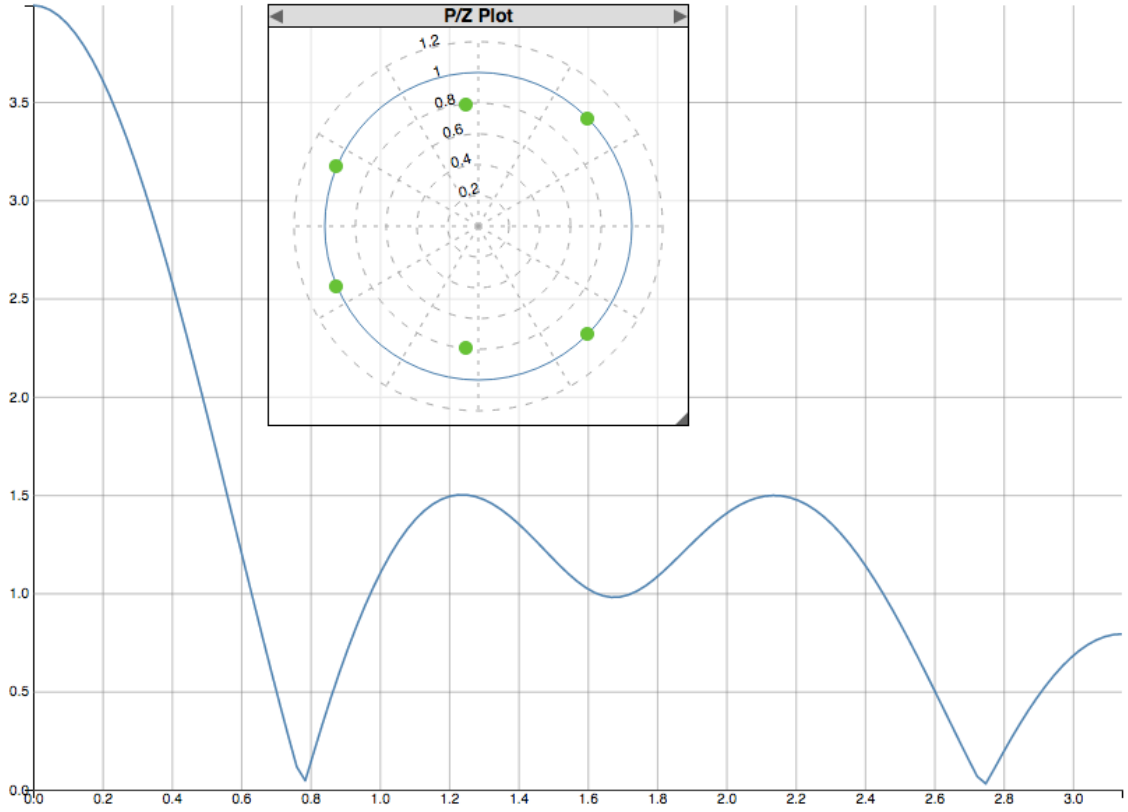
d) Filter a glottal source signal, with fundamental frequency 150 Hz and sampling frequency 8000 Hz, through the 'eh' filter. Play this sound for your TA, who will check you off (alternately, submit as a *.wav* file).

Note that the filter(b,a,xx) command will prove useful here.

a & b)

B = [1, 0.6, 0.07901, 0.57877, 0.67674, 0.42063, 0.63562]
A = [1]
zeros = [-0.08182-0.79091i, -0.08182+0.79091i, -0.92727-0.39091i, -0.92727+0.39091i, 0.70909-0.70000i, 0.70909+0.70000i]
poles = []

c)



d)

```
>> B = [1, 0.6, 0.07901, 0.57877, 0.67674, 0.42063, 0.63562];
>> A = [1];
>> f = 150;
>> ff = 30*f*2;
>> t = 0:(1/ff):1;
>> glottal = 0;
>> for i = 1:30
glottal = cos(2*pi*150*t*i) + glottal;
end
>> sound = filter(B,A,glottal);
>> soundsc(glottal, fs);
>> glottal = glottal/(max(abs(glottal)));
```

```
>> audiowrite('glottal.wav',glottal,fs)
>> [eh,w] = freqz(B,A, 150);
>> plot(w,abs(eh))
```