Lab 4 Part 2

Contents

2.1 - Pole-Zero Plots

```
clear
```

2.1.1

Question: Where in the complex plane can zeros and poles be placed to have the strongest influence on the magnitude response of the filter?

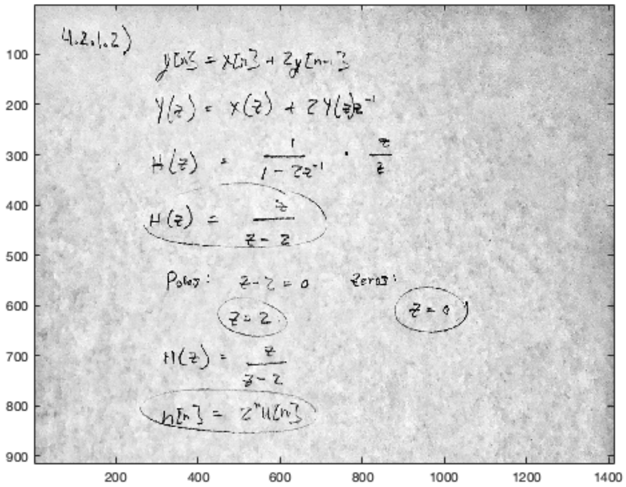
Zeros and poles can be placed near the unit circle (meaning they have a magnitude of ~1) to have the strongest influence on the magnitude response of the filter.

2.1.2

Question: What are the poles and zeros of this filter?

Question: What is this filter's impulse response?

```
img = imread("4.2.1.2.jpg");
image(img);
```



4.2.1.2)

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

$Y(z) = X(z) + 2Y(z)z^{-1}$

$H(z) = \frac{1}{1-2z^{-1}} \cdot \frac{z}{z}$

$\boxed{H(z) = \frac{z}{z-2}}$

Poles: $z-2=0$ Zeros:

$\boxed{z=2}$ $\boxed{z=0}$

$H(z) = \frac{z}{z-2}$

$\boxed{h[n] = 2^n u[n]}$

100
200
300
400
500
600
700
800
900
200
400
600
800
1000
1200
1400

2.2 - Vowel Creation

```
clf
```

2.2.1

eh

```
Ehzeros = [-0.08276-0.78621i, -0.08276+0.78621i, -0.92388+0.38268i, ...
    -0.92388-0.38268i, 0.70711+0.70711i, 0.70711-0.70711i, 0];
Ehpoles = [];

Ehzeros = angle(Ehzeros);
array2table(Ehzeros)
```

```
ans =

  1×7 table

    Ehzeros1    Ehzeros2    Ehzeros3    Ehzeros4    Ehzeros5    Ehzeros6    Ehzeros7
    ________    ________    ________    ________    ________    ________    ________

    -1.6757      1.6757      2.7489     -2.7489      0.7854     -0.7854        0
```

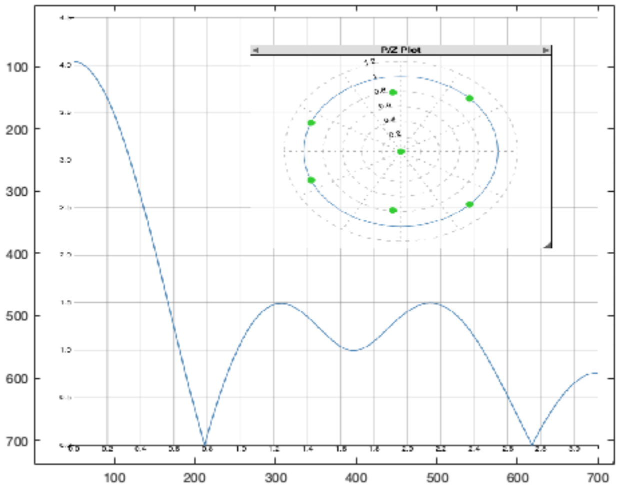
2.2.2

```
EhB = [1, 0.59906, 0.08360, 0.60303, 0.68858, 0.43649, 0.62498, 0] %#ok<NOPTS>
EhA = [1] %#ok<NBRAK,NOPTS>
```

```
EhB =

  Columns 1 through 7

    1.0000    0.5991    0.0836    0.6030    0.6886    0.4365    0.6250

  Column 8

         0


EhA =

     1
```

2.2.3

```
img = imread("4.2.2.3.png");
image(img);
```



2.2.4

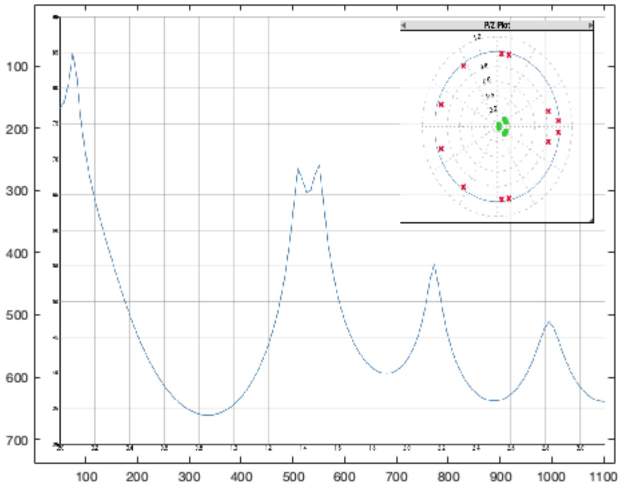
```
type glottal_key_to_note

fs = 8000;
dur = .20;
glottal = glottal_key_to_note(42, dur, 50, fs);

eh = filter(EhB,EhA,glottal);
audiowrite("eh.wav", eh/max(eh), fs);
```

```
function [xx] = glottal_key_to_note(keynum, dur, harm, fs)
    %GLOTTAL_KEY_TO_NOTE

    freq = 110 * (   2^( (keynum-49)/12 )   );
    A = 1;
    phi = 0;

    tt = 0:(1/fs):dur-1/fs;
    xx = zeros(1,length(tt));

    for k = [-harm:-1 1:harm]
        xx = xx + real( A*exp(j*2*pi*freq*k*tt)*exp(j*phi) );
    end

end
```

2.3 Four More Vowel Sounds

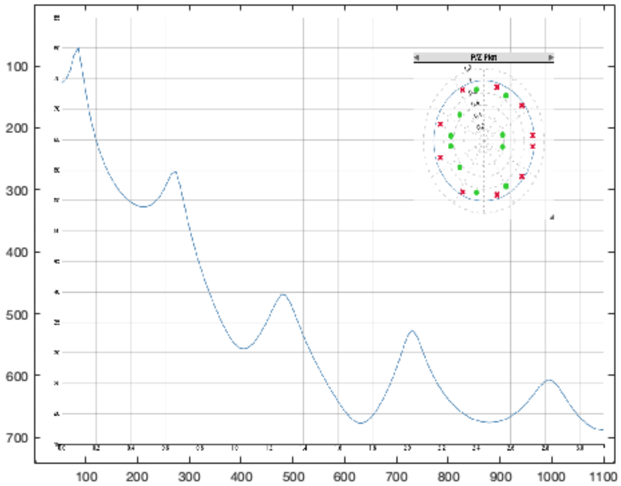
ee

```
img = imread("2.3 ee.png");
image(img);

EeB = [200, -112.94259, 27.76224, -3.61118, 0.25026, -0.00788, 0.00011];
EeA = [1, -1.26228, 0.68492, -0.90067, -0.23604, 0.20558, 0.47245, 0.53774, ...
    -0.08982, -0.40057, 0.39162, -0.92361, 0.53604];
Eezeros = [0.11626-0.09550i, 0.11626+0.09550i, 0.02491+0.02076i, ...
        0.02491-0.02076i, 0.14118+0.06228i, 0.14118-0.06228i];
Eepoles = [-0.89689-0.29481i, -0.89689+0.29481i, 0.82215-0.20346i, ...
        0.82215+0.20346i, 0.19100-0.95917i, 0.19100+0.95917i, ...
        0.07474-0.97578i, 0.07474+0.97578i, -0.53979-0.80969i, ...
        -0.53979+0.80969i, 0.97993+0.07889i, 0.97993-0.07889i];

ee = filter(EeB,EeA,glottal);
audiowrite("ee.wav", ee/max(ee), fs);
```



oo

```
img = imread("2.3 oo.png");
image(img);

OoB = [55, 51.43840, 51.39310, 48.07304, 30.54631, 23.39079, 10.37301, ...
    -3.26610, -5.37189, -0.43537, 0.85523];
OoA = [1, -1.40287, 0.43513, 0.10564, -0.15593, 0.16557, -0.29405, ...
    0.25794, 0.08829, -0.71436, 0.56551];
Oozeros = [-0.65330-0.08596i, -0.65330+0.08596i, -0.14441-0.85616i, ...
    -0.14441+0.85616i, 0.37135-0.09971i, 0.37135+0.09971i, ...
    0.44011-0.75301i, 0.44011+0.75301i, -0.48138-0.43668i, ...
    -0.48138+0.43668i];
Oopoles = [-0.86648-0.27851i, -0.86648+0.27851i, 0.75645-0.58797i, ...
    0.75645+0.58797i, -0.41948-0.84928i, -0.41948+0.84928i, ...
    0.96963-0.09284i, 0.96963+0.09284i, 0.26132-0.89742i, ...
    0.26132+0.89742i];

oo = filter(OoB,OoA,glottal);
audiowrite("oo.wav", oo/max(oo), fs);
```

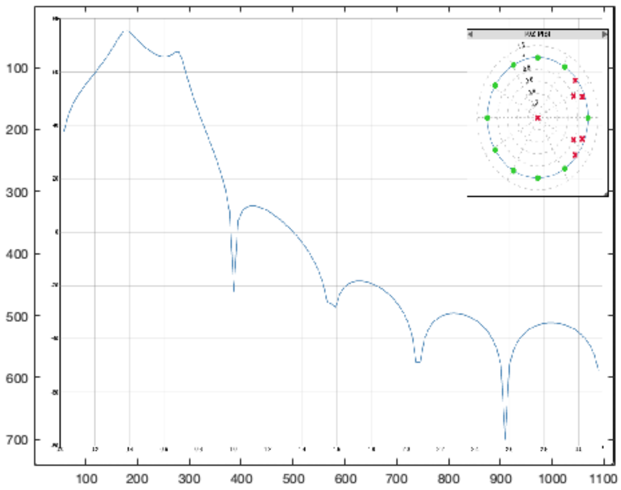


oh

```
img = imread("2.3 oh.png");
image(img);

OhB = [1, 1.58051, 1.78483, 1.41740, 0.78483, 0, -0.78483, -1.41740, ...
        -1.78483, -1.58051, -1];
OhA = [1, -4.68698, 9.77351, -11.49104, 8.01180, -3.13776, 0.54346, 0];
Ohzeros = [1, 1i, -1i, 0.53583+0.84433i, 0.53583-0.84433i, ...
    -0.48175+0.87631i, -0.48175-0.87631i, -0.84433+0.53583i, ...
    -0.84433-0.53583i, -1];
Ohpoles = [0.88329+0.34972i, 0.88329-0.34972i, 0.74740+0.61830i, ...
    0.74740-0.61830i, 0, 0.71281+0.36319i, 0.71281-0.36319i];
```

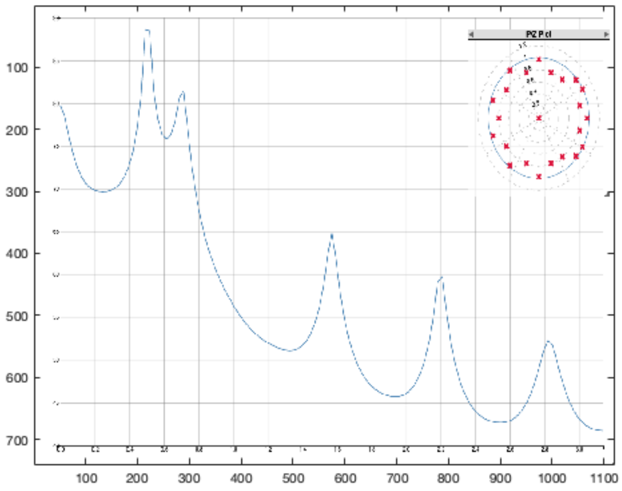
```
oh = filter(OhB,OhA,glottal);
audiowrite("oh.wav", oh/max(oh), fs);
```

```
Warning: Data clipped when writing file.
```



ah

```
img = imread("2.3 ah.png");
image(img);

AhB = [675];
AhA = [1, -1.71198, 0.92521, -0.08710, 0.18647, -0.02439, -0.07753, ...
    -0.09882, -0.17966, 0.11212, 0.63108, -0.96745, 0.48493, -0.06256, ...
    -0.06501, -0.01741, 0.00826, 0.02030, 0.02860, -0.00497, -0.07143, ...
    0.10441, -0.06347, 0];
Ahzeros = [];
Ahpoles = [0.86754+0.47694i, 0.86754-0.47694i, 0.74151+0.63331i, ...
    0.74151-0.63331i, 0.97515i, -0.97515i, -0.57318+0.78891i, ...
    -0.57318-0.78891i, -0.90388+0.29369i, -0.90388-0.29369i, 0.96030, 0, ...
    0.81506+0.20927i, 0.81506-0.20927i, -0.24474+0.75324i, ...
    -0.24474-0.75324i, -0.64074+0.46553i, -0.64074-0.46553i, ...
    0.24474+0.75324i, 0.24474-0.75324i, -0.792, 0.46553+0.64074i, ...
    0.46553-0.64074i];

ah = filter(AhB,AhA,glottal);
audiowrite("ah.wav", ah/max(ah), fs);
```



2.4 - Whispering Vowels

```
type whisper
whisp = whisper(dur, fs);
```

```
function [xx] = whisper(dur, fs)
    %WHISPER
    xx = randn(1,floor(fs * dur));
    xx = xx - min(xx);
    xx = xx / max(xx);
end
```

```
eh = filter(EhB,EhA,whisp);
ee = filter(EeB,EeA,whisp);
oo = filter(OoB,OoA,whisp);
oh = filter(OhB,OhA,whisp);
ah = filter(AhB,AhA,whisp);

audiowrite("eh-whisp.wav", eh/max(eh), fs);
audiowrite("ee-whisp.wav", ee/max(ee), fs);
audiowrite("oo-whisp.wav", oo/max(oo), fs);
audiowrite("oh-whisp.wav", oh/max(oh), fs);
audiowrite("ah-whisp.wav", ah/max(ah), fs);
```

2.5 - Reconstruction of a Voweled Fugue

```
load Chris_fugue.mat

A = containers.Map;
A('eh') = EhA;
A('ee') = EeA;
A('oo') = OoA;
A('oh') = OhA;
A('ah') = AhA;

B = containers.Map;
B('eh') = EhB;
B('ee') = EeB;
B('oo') = OoB;
B('oh') = OhB;
B('ah') = AhB;

type singSong
type ADSR

[song, fs] = singSong(theVoices, A, B);
audiowrite("chris_fugue.wav", song/max(song), fs);
```

```
function [song, fs] = singSong(theVoices, A, B)
    % SINGSONG

    fs = 10000;
    bpm = 120;
    bps = bpm / 60;
    spb = 1 / bps;
    spp = spb / 4; %seconds per pulse, theVoices is measured in pulses with 4 pulses per beat

    harm = 30;

    % determine how long we need to make the song by looking at the last
    % note in each of theVoices
    maxIndex = 1;
    for i = 1:length(theVoices)
        lastNoteNumber = length(theVoices(i).noteNumbers);
        lastNote = glottal_key_to_note(theVoices(i).noteNumbers(lastNoteNumber), theVoices(i).durations(lastNoteNumber)*spp, harm, fs);
        currentIndex = spp*fs*theVoices(i).startPulses(lastNoteNumber) + length(lastNote) - 1;
        if currentIndex > maxIndex
            maxIndex = currentIndex;
        end
    end

    song = zeros(1, ceil(maxIndex)); %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)
        theVoices(i).vowels = regexp(theVoices(i).vowels, sprintf('\\w{1,%d}', 2), 'match'); % split the vowel string into a cell array where each ele
        for j = 1:length(theVoices(i).noteNumbers)
            keynum = theVoices(i).noteNumbers(j); % get the ith jth keynum
            dur = theVoices(i).durations(j)*spp; % get the ith jth duration
            note = ADSR(glottal_key_to_note(keynum, dur, harm, fs)); % apply envelope to glottal sound
            vowel = theVoices(i).vowels{j}; % get the ith jth vowel
            a = A(vowel); % get the filter coefficients for the ith jth vowel
            b = B(vowel);
            note = filter(b,a,note); % apply the vowel filter
            note = note ./ max(note); % normalize volume
            locstart = spp*fs*theVoices(i).startPulses(j); % Index of where note starts
            locend = locstart + length(note) - 1; % index of where note ends
            % floor is included here to remove the warning "Integer operands are required for colon operator when used as index."
            song(floor(locstart):floor(locend)) = song(floor(locstart):floor(locend)) + note;
        end
    end
end

function tone = ADSR(tone)
    %{
    ADSR: Apply an envelope to a particular note.
    Input Args:
    -tone: the tone to apply the ADSR envelope to.
    Output:
    -tone: the tone that was supplied as a paramater, with the ADSR
    envelope.
    Usage:
    envelopedTone = ADSR(tone);
    %}

    A = linspace(0.0, 0.9, (length(tone)*0.25)); % rise 25%
    D = linspace(0.9, 0.7, (length(tone)*0.05)); % drop 5%
    S = linspace(0.7, 0.7, (length(tone)*0.40)); % maintain 40%
    R = linspace(0.7, 0.0, (length(tone)*0.30)); % drop 30%
    ADSR = [A D S R];

    x = zeros(size(tone));
    x(1:length(ADSR)) = ADSR;
    tone = tone .* x;
end
```

Published with MATLAB® R2019a