Lab 4 Part 1

Contents

1.1 - Modeling Vowel Production

```
clear
```

Create a one second long glottal source signal, containing 7 non-zero harmonics (Fourier series k values -7 through 7), with fundamental frequency 150Hz. Use a sampling frequency of 8000 samples/s. Note: this is not to be played for a TA, just for you to hear how the glottal signal sounds.

```
freq = 150;
A = 1;
phi = 0;

fs = 8000;
tt = 0:1/fs:1;
xx = zeros(1,length(tt));
harm = 7;

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

audiowrite("glottal.wav", xx/max(xx), fs)
```

Question: What is the minimum sampling frequency necessary to avoid aliasing?

The sampling frequency must be strictly greater than 2100 Hz. This is because the largest frequency in the harmonic is equal to 7 * 150 = 1050 Hz and the sampling frequency must be strictly greater than double this value (2100 Hz).

1.2 - A Function to Play a Vocal Note

Write a function, named glottal_key_to_note (similar to your key_to_note function from Lab 1, See the Appendix) that takes in a key number, a duration, and a number of harmonics to produce a glottal source signal of given duration with fundamental frequency corresponding to the desired note. Use a sampling frequency of 8000 Hz (remember that a note with harmonics is a sum of tones with frequencies given by a fundamental frequency f times the harmonic number k).

```
type glottal_key_to_note
```

```
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
```

Question: If the keynum provided is 49, for 7 harmonics, what is the minimum sampling frequency required to prevent aliasing?

The result stored in ans below is the minimum required frequency to prevent aliasing.

```
max_freq = 220 * (   2^( (49-49)/12 )   ) * 7;

2 * max_freq + 1
```

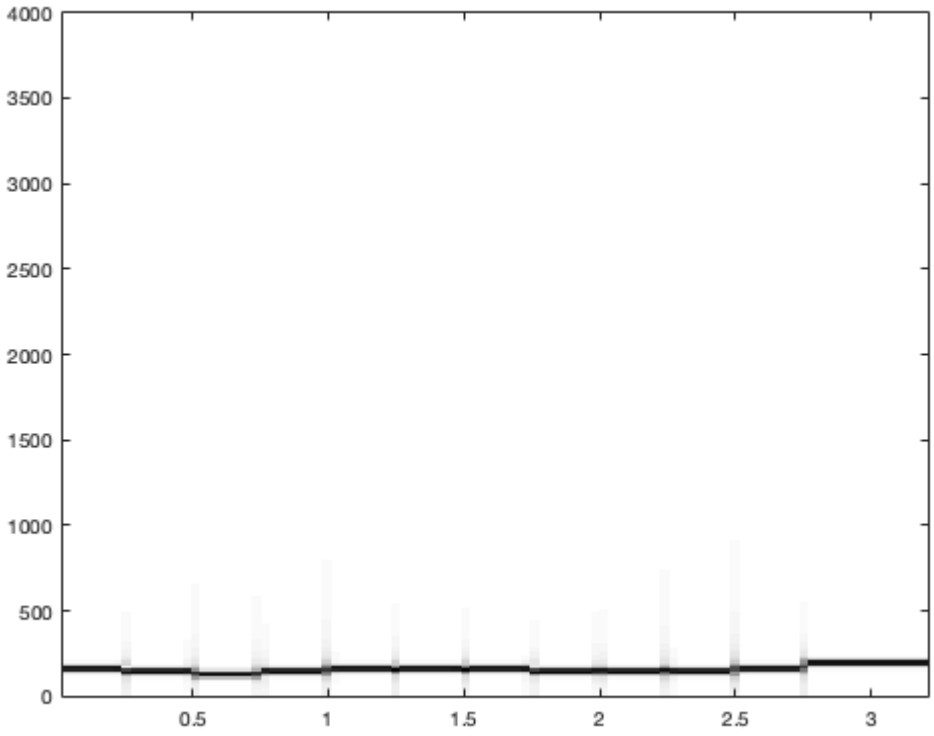
```
ans =

        3081
```

1.3 - Synthesize a Song - Mary had a Bleating Lamb

Use glottal_key_to_note to write a script that plays the Mary song with note durations of 0.25 seconds.

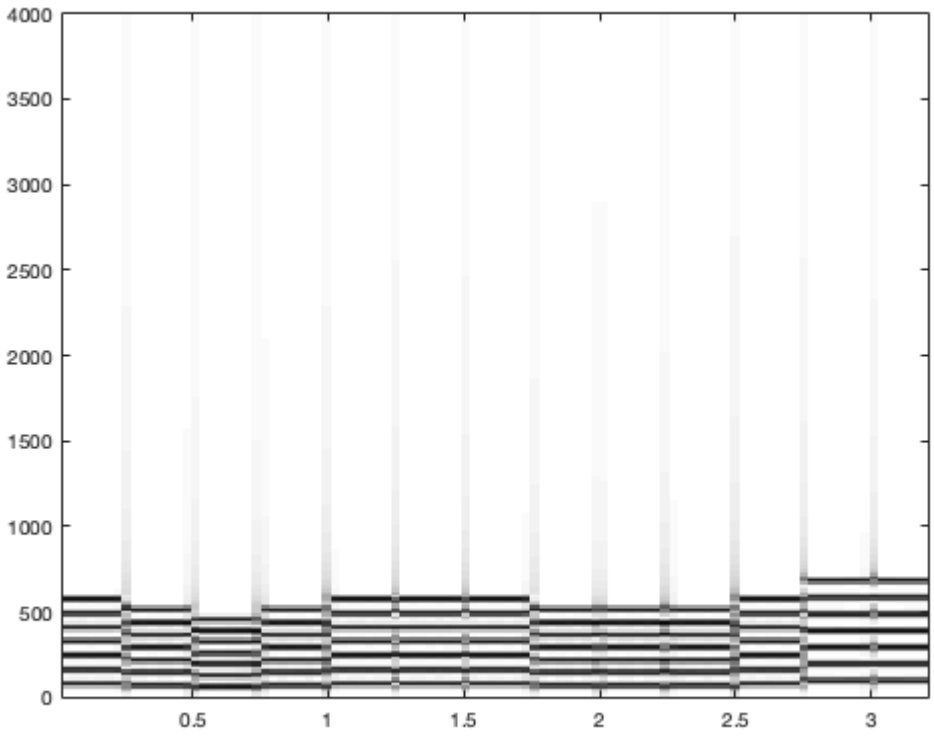
```
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
mary.durations = 0.25 * ones(1,length(mary.keys));

fs = 8000;
xx = zeros(1, sum(mary.durations)*fs);
n1 = 1;
A = 1;
phi = 0;

for kk = 1:length(mary.keys)
    keynum = mary.keys(kk);
    dur = mary.durations(kk);
    freq = 220 * (   2^( (keynum-49)/12 )   );
    tt = 0:(1/fs):dur-1/fs;
```

```
    tone = real( A*exp(j*2*pi*freq*tt)*exp(j*phi) );
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

audiowrite("mary.wav", xx/max(xx), fs)
plotspec(xx,fs,512);
```



Create a script that plays the notes with seven harmonics (k=7).

```
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
mary.durations = 0.25 * ones(1,length(mary.keys));

fs = 8000;
xx = zeros(1, sum(mary.durations)*fs);
n1 = 1;

for kk = 1:length(mary.keys)
    tone = glottal_key_to_note(mary.keys(kk), mary.durations(kk), 7, fs);
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

audiowrite("mary-harmonic.wav", xx/max(xx), fs)
plotspec(xx,fs,512);
```



Plot the frequency-time spectrogram of Mary for both one harmonic and seven harmonics and compare.

You can see that there is much more information contained within the frequency-time spectogram of the harmonic version when compared to the version that had no harmonics. This is a result of the harmonic version being a sum of sinusoids with varying frequencies corresponding to the fourier series terms).

1.4 - Scale Creation

1.4.1 - Scale Matrix

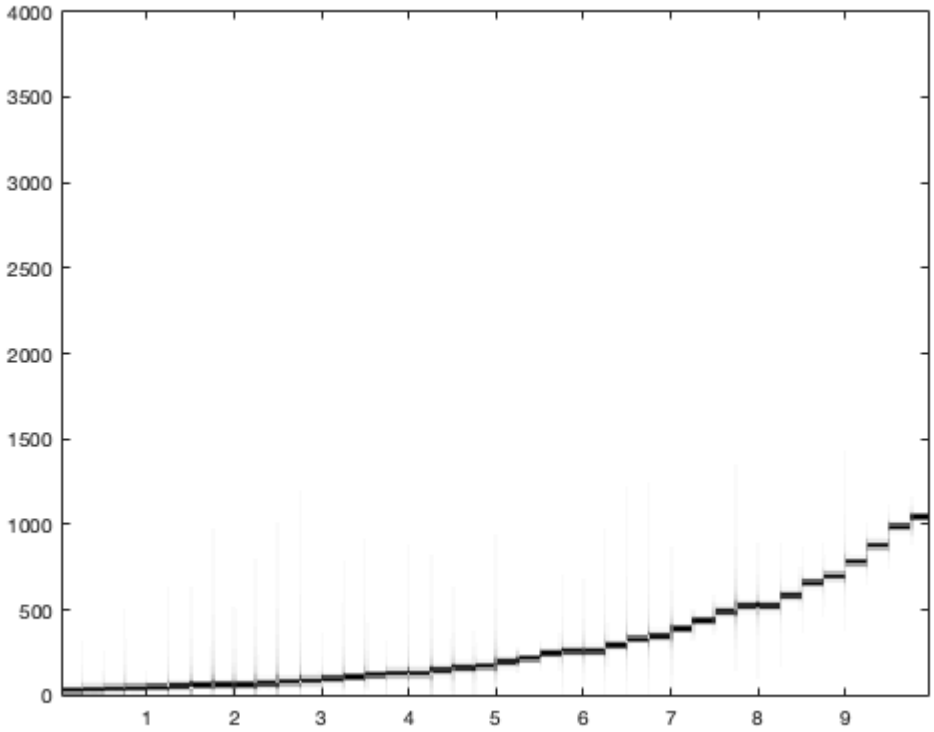
```
% Using the code provided, generate a 7x8 matrix of the 8 notes in octaves
% 1-7
baseKey = 4;
offsets = [0,2,4,5,7,9,11,12];
keynums = zeros(7,length(offsets));

for i = 1:7 %For loop for octaves 1-7
    keynums(i,:) = baseKey + 12*(i-1) + offsets;
end
```

1.4.2 - Creating the Sounds

Make one scale with 1 harmonic.

```
durations = 0.25 * ones(1,length(keynums));
octaves = 3:1:7;
```

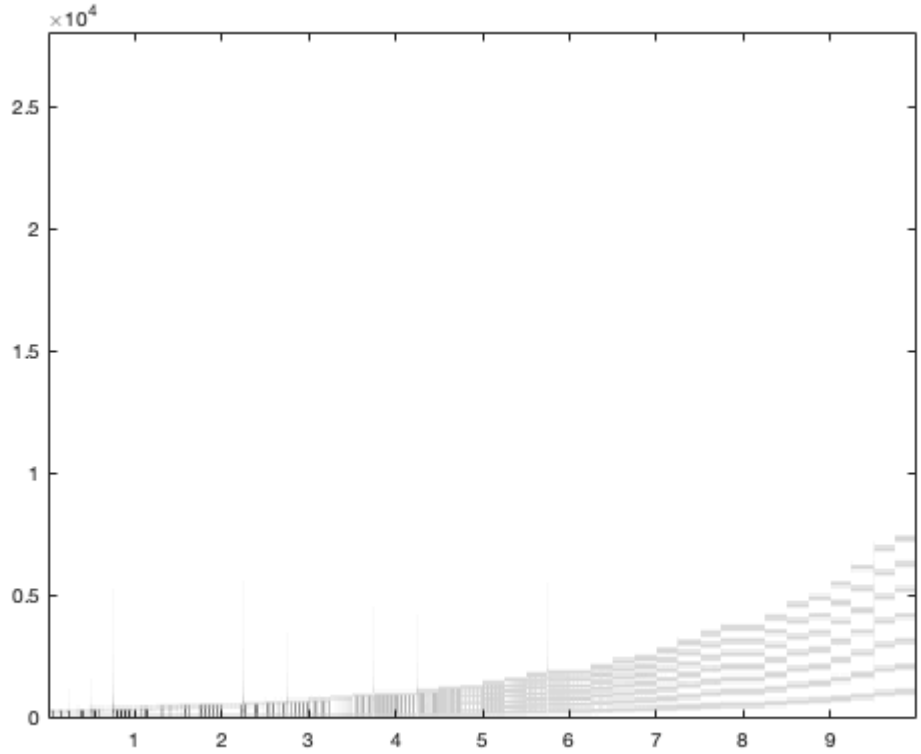
```
oneharmfs = 8000;
oneharm = zeros(1, sum(durations)*oneharmfs*length(octaves));

n1 = 1;
for octave = octaves
    for kk = 1:length(keynums)
        tone = glottal_key_to_note(keynums(octave,kk), durations(kk), 1, oneharmfs);
        n2 = n1 + length(tone) - 1;
        oneharm(n1:n2) = oneharm(n1:n2) + tone;
        n1 = n2 + 1;
    end
end

audiowrite("musical-scale-1.wav", oneharm/max(oneharm), oneharmfs)
plotspec(oneharm,oneharmfs,512);
```



Make another set with seven harmonics.

```
durations = 0.25 * ones(1,length(keynums));
octaves = 3:1:7;

sevenharmfs = 56000;
sevenharm = zeros(1, sum(durations)*sevenharmfs*length(octaves));

n1 = 1;
for octave = octaves
```

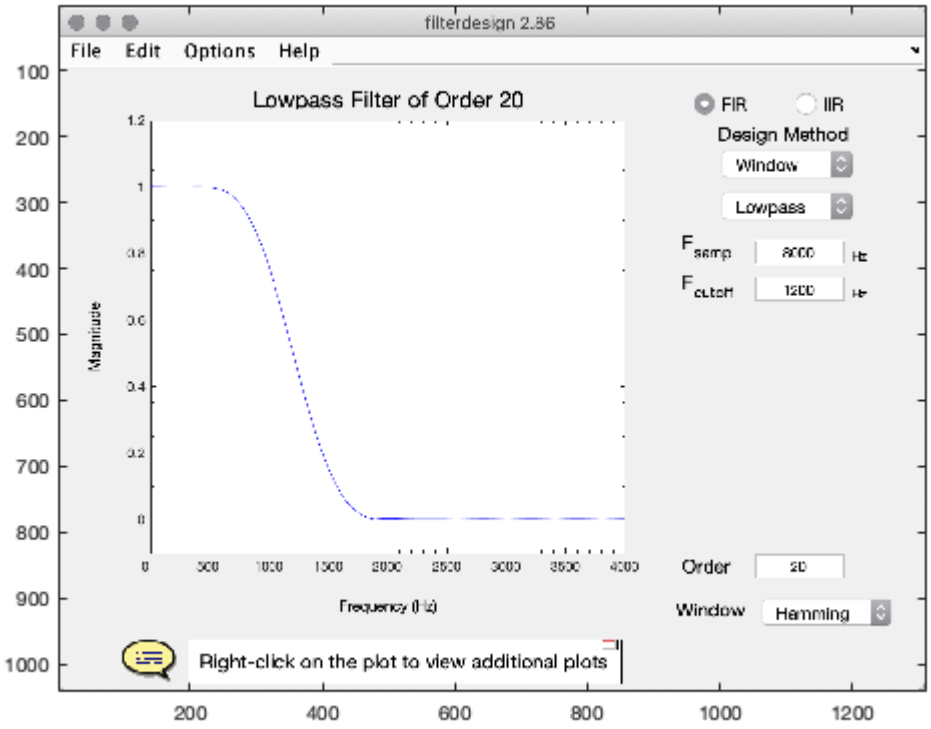
```
    for kk = 1:length(keynums)
        tone = glottal_key_to_note(keynums(octave,kk), durations(kk), 7, sevenharmfs);
        n2 = n1 + length(tone) - 1;
        sevenharm(n1:n2) = sevenharm(n1:n2) + tone;
        n1 = n2 + 1;
    end
end

audiowrite("musical-scale-7.wav", sevenharm/max(sevenharm), sevenharmfs)
plotspec(sevenharm,sevenharmfs,512);
```



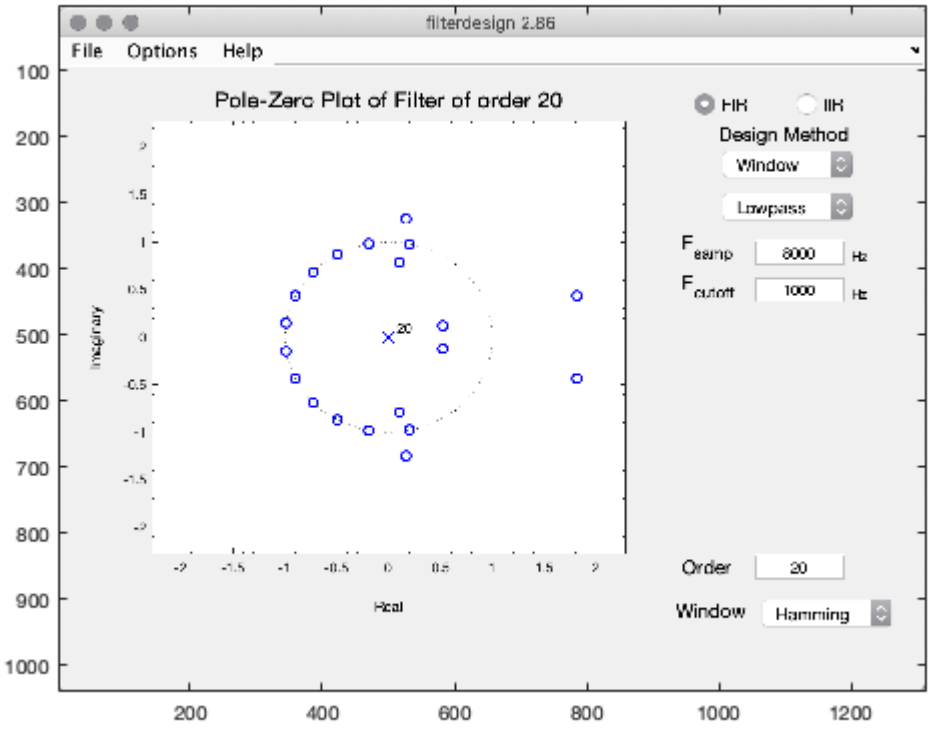
1.5 - Intro to MATLAB Filter Designer

1.5.1

```
img = imread("4.1.1.5.1 magnitude.png");
image(img);
```



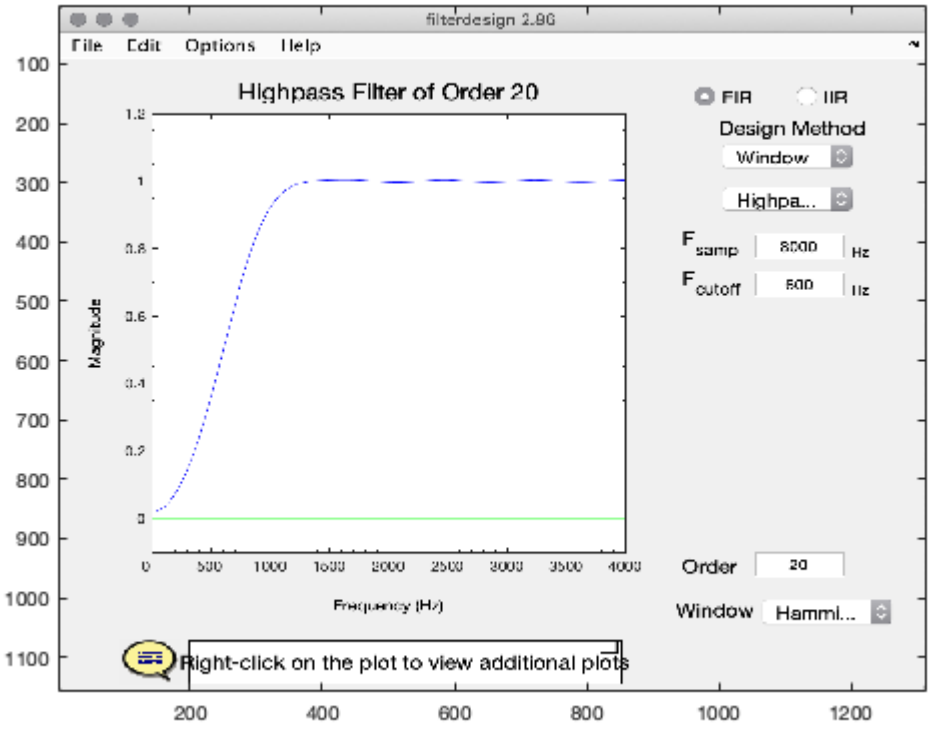

```
img = imread("4.1.1.5.1 pole-zero.png");
image(img);
```



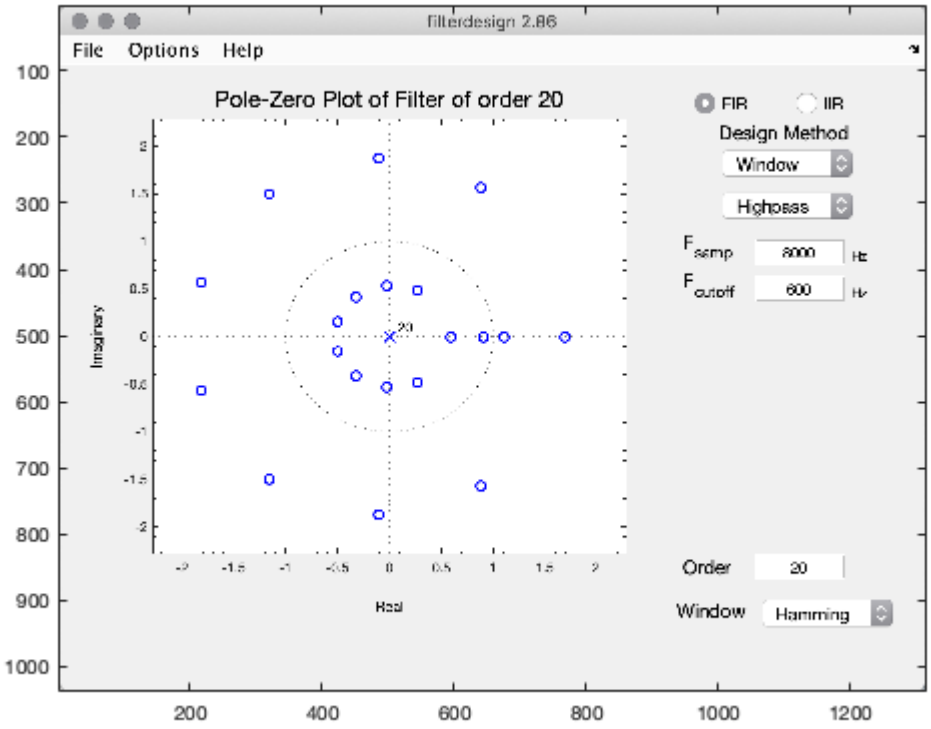

```
blow = [9.35561239804650e-19,0.00293324576743476,0.00635175091214551, ...
    0.00378865096369201,-0.0124062151114183,-0.0343774677078494, ...
    -0.0319070986574590,0.0265705535690567,0.138067569737800, ...
    0.251720344085349,0.300000000000000,0.251720344085349, ...
    0.138067569737800,0.0265705535690568,-0.0319070986574590, ...
    -0.0343774677078494,-0.0124062151114183,0.00378865096369202, ...
    0.00635175091214551,0.00293324576743476,9.35561239804650e-19];
alow = 1;

yylow = filter(blow,alow,oneharm);
audiowrite("1.5.1.wav", yylow/max(yylow), oneharmfs);
```

1.5.2

```
img = imread("4.1.1.5.2 magnitude.png");
image(img);
```



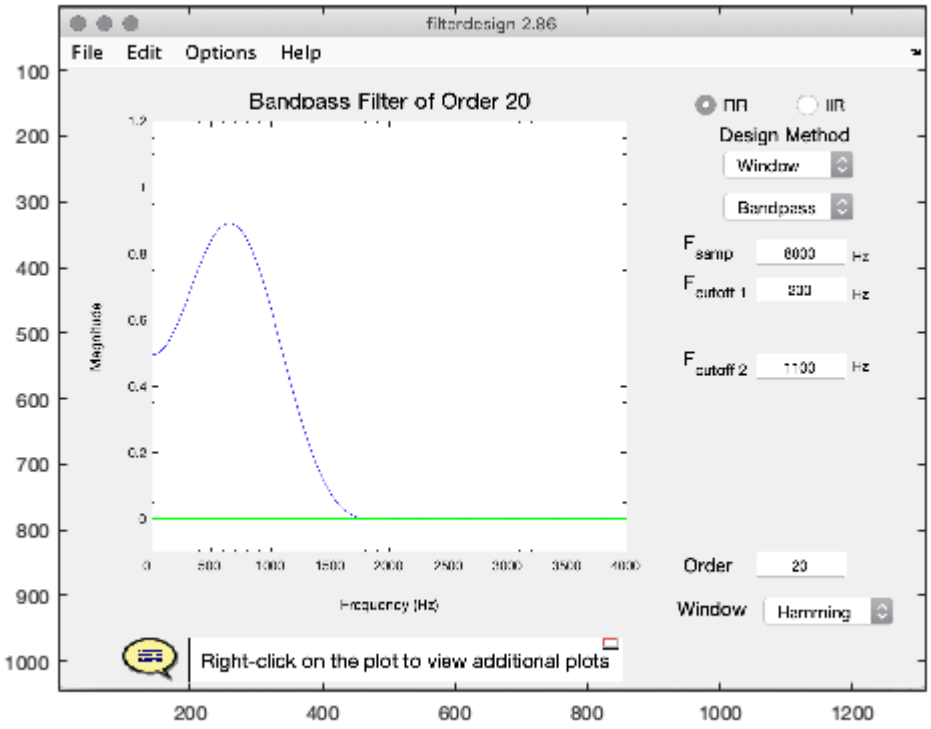

```
img = imread("4.1.1.5.2 pole-zero.png");
image(img);
```



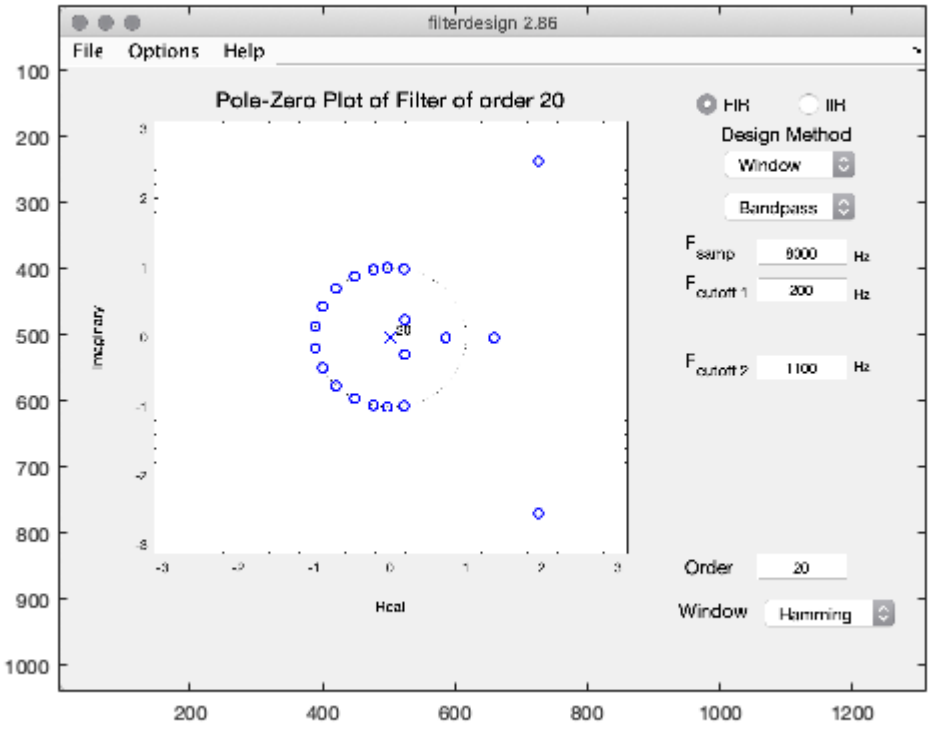

```
bhigh = [0.00254647908947032,0.00323051448115936,0.00392559795177907, ...
    0.00191793848726065,-0.00652233326771518,-0.0243085405362418, ...
    -0.0516267701101649,-0.0849255103797730,-0.117447289804661, ...
    -0.141256173356668,0.850000000000000,-0.141256173356668, ...
    -0.117447289804661,-0.0849255103797730,-0.0516267701101649, ...
    -0.0243085405362419,-0.00652233326771518,0.00191793848726065, ...
    0.00392559795177907,0.00323051448115936,0.00254647908947032];
ahigh = 1;

yyhigh = filter(bhigh,ahigh,oneharm);
audiowrite("1.5.2.wav", yyhigh/max(yyhigh), oneharmfs);
```

1.5.3

```
img = imread("4.1.1.5.3 magnitude.png");
image(img);
```




```
img = imread("4.1.1.5.3 pole-zero.png");
image(img);
```




```
bband = [-0.000745846457156114,3.34614788662254e-05,-0.00242615296036643, ...
    -0.0137861535346343,-0.0358819099041843,-0.0560691793310916, ...
    -0.0486816531191399,0.00589060551200377,0.0985246426661597, ...
    0.187921780377578,0.225000000000000,0.187921780377578, ...
    0.0985246426661597,0.00589060551200377,-0.0486816531191399, ...
    -0.0560691793310916,-0.0358819099041843,-0.0137861535346343, ...
    -0.00242615296036644,3.34614788662254e-05,-0.000745846457156114];
aband = 1;

yyband = filter(bband,aband,oneharm);
audiowrite("1.5.3.wav", yyband/max(yyband), oneharmfs);
```

1.5.4 - Comment on how the pole/zero plots differ between the above filters.

The lowpass and the bandpass have very similar pole-zero plots. This is because the bandpass filter was designed to filter out higher freauencies with a bandpass that is near zero. The zeros seen for this filter all exist on the unit circle around/after pi/2. This reduces the higher frequencies.

The highpass filter, in contrast, has no zeros on the unit circle, only ones near it. There are zeros nearly all the way around the circle, but more are concentrated around w = 0, and this makes sense because it reduces lower frequencies.

1.5.5 - Generate the scale vector with k = 7. Run the scale through each of the above filters, and comment on the differences.

```
yyband = filter(bband,aband,sevenharm);
```

```
audiowrite("1.5.5 band.wav", yyband/max(yyband), sevenharmfs);

yylow = filter(blow,alow,sevenharm);
audiowrite("1.5.5 low.wav", yylow/max(yylow), sevenharmfs);

yyhigh = filter(bhigh,ahigh,sevenharm);
audiowrite("1.5.5 high.wav", yyhigh/max(yyhigh), sevenharmfs);
```

For me, it was much easier to tell what frequencies were reduced and which ones were kept when listening to this version of the output filters. It was very similar to the previous outputs though.

Published with MATLAB® R2019a