Bioen 316 AC
Homework 7
June 6, 2019

Skyler Hallinan
hallisky@uw.edu
1732227

Problem 1

In the past few weeks, you have analyzed a second-order band-pass RLC circuit to apply before sampling, applied an FIR filter to extract EEG beta waves, and seen MATLAB's IIR filter design tool. For this homework, use MATLAB to compare the behavior of these three filter types, all with cutoff frequencies of 0.5 and 50 Hz. For the digital filters, choose a sampling frequency that can handle 120 Hz noise, i.e. the first harmonic of 60 Hz.

Answer:
Our comparison of the three filtering methods is shown in this plot:

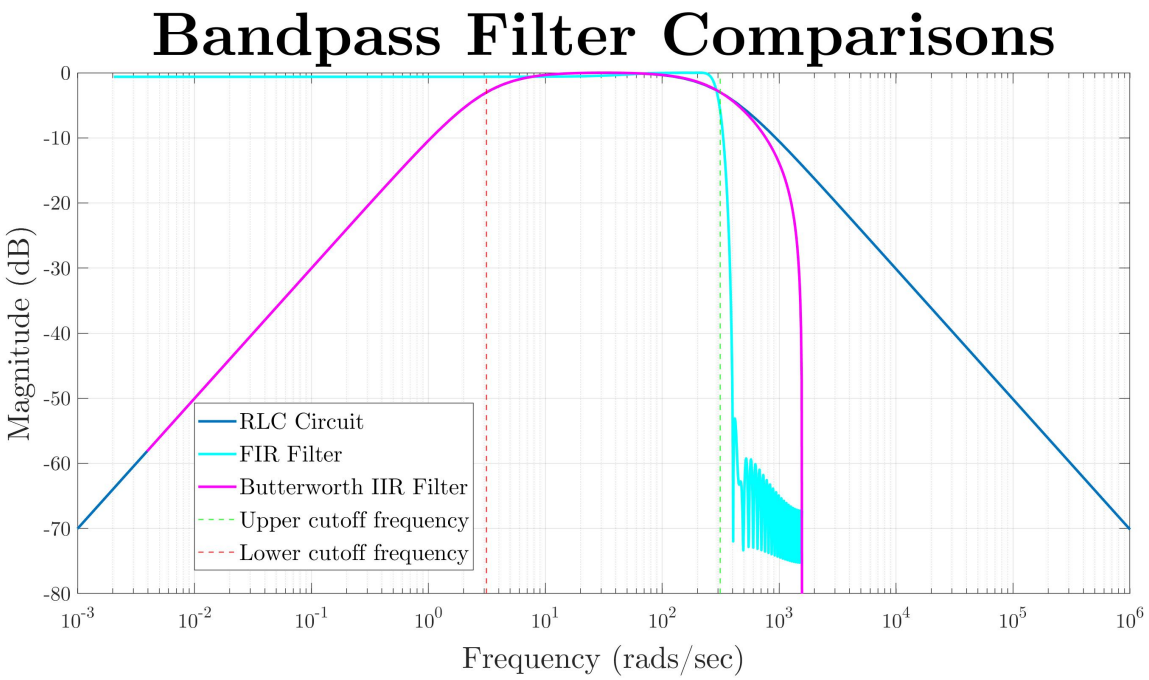

Figure 1: Bandpass filter comparisons of three varying methods

We conducted the filtering in radians per second (ω). We calculated our low pass and high pass cutoff frequencies of the band pass filter, ω_{low} and ω_{high}, to be π and 100π respectively. We also had the following equations:

$$\omega_{hi} = +\alpha + \sqrt{\alpha^2 + \omega_0^2}, \quad \omega_{low} = -\alpha + \sqrt{\alpha^2 + \omega_0^2}$$

We calculated our α to be $\frac{99}{2}\pi$, from subtracting our equations relating our ω_{low} and ω_{high} to their ω_0 and simplifying for alpha; $\alpha = \frac{\omega high - \omega low}{2} = \frac{99}{2}\pi$. Finally, we calculated our ω_0 to be 10 π, found from substituting our calculated values into their equations and isolating for ω_0.

We first created an RLC filter using the general form, $G(\omega) = \frac{2\alpha j\omega}{(j\omega)^2 + 2\alpha j\omega + \omega_0^2}$, and inputted a variety

Skyler Hallinan
hallisky@uw.edu
1732227

of frequencies to create the Bode plot. Then, we created a FIR filter as in the previous homework, centering it at $\frac{101\pi}{2}$ and applying a Hamming smoothing window. We used a sampling frequency of 500 Hz, as this was higher than 2 times the 120 Hz electrical noise that was present, so it could properly handle it. By Nyquist's Theorem, we can avoid aliasing of this frequency by choosing a sampling frequency that is sufficiently high (above 2 times that frequency), hence we chose 500 Hz. 500 Hz is more than two times greater than 120, so we do not have to worry about aliasing occurring. We also used this when we created our last filter in MATLAB, the 2nd-order Butterworth filter.

We see that the RLC filter performs well, giving clear bandpass filtering of our desired frequencies, with attenuation on the sides of higher or lower undesired frequencies. The Butterworth IIR filter also performs relatively well, but it has a noticeably steeper decline after the upper frequency cutoff than does the RLC filter. In addition, the RLC filter and Butterworth filter align almost perfectly on the left side (for lower frequencies). Although it seems like the RLC circuit extends more on the left side, it is just because we used different number of points/ranges for the two filters. Finally, we see that the FIR filter does not do a good job at attenuating lower frequencies. There is a horizontal line to the left of the lower cutoff frequency at a relatively low magnitude dB scale. This means the FIR filter is not attenuating low frequencies as it should be doing. However, it does filter higher frequencies (above the upper cutoff) more successfully, albeit with some oscillatory behavior.

See the code for more annotation.

Skyler Hallinan
hallisky@uw.edu
1732227

Appendix A

MATLAB code

```
% Skyler Hallinan
% 6/1/19
% Homework 6: Part B

clear all; close all; clc;

% Changing our default fonts
set (0, 'defaultAxesFontName', 'CMU Serif')
set (0, 'defaultTextFontName', 'CMU Serif')

%% Setting the lower and upper cutoff frequencies (in radians)
whigh = 50*2* pi; % Converting lower cutoff to radians
wlow = 0.5*2*pi; % Converting upper cutoff to radians
alpha = (whigh - wlow)/2; % Calcuating the alpha value from whigh
   and wlow

% Below: Calculaing w0 from alpha, whigh, and wlow.
w0 = sqrt(((whigh +wlow)/2)^2-alpha^2);
% Alternative calculation methods of w0:
% w01 = sqrt((whigh - alpha)^2-alpha^2);
% w02 = sqrt((wlow + alpha)^2-alpha^2);

%% RLC Filter Calculation

% Creating frequency axis (on a log scale) for our filter bode plot.
    Note
% that this frequency axis is in rads/sec (omega).
omegas = logspace(-3,6,100);

% Calculating RLC circuit gain (From formula)
gain = abs((2*alpha.*omegas*i)./((i.*omegas).^2+2*alpha.*omegas*i+w0
   ^2));

% Semilogarthmic (bode plot) in dB scale of our omega frequency axis
    and
% circuit gain (Bode plot)
semilogx(omegas,db(gain), 'linewidth', 2)
set(gca, 'fontsize', 16);
ylabel("Magnitude (dB)", 'fontsize', 24); xlabel("Frequency (rads/
   sec)", 'fontsize', 24);
title("Bandpass Filter Comparisons", 'fontsize' , 48);
```

Skyler Hallinan
hallisky@uw.edu
1732227

```
grid on;

hold on;

%% FIR Filter

% Calculating the 3rd zero of our sinc function, for truncation
   limits
tlim = (3*pi)/alpha;
fs = 500; % Sampling frequency
ts = 1/fs; %
% Creaing t vector to input into our sinc function with desired
   smapling
% frequency as specified above
t = -tlim:ts:tlim-ts;

% Setting any t == 0 indeces to a small number to remove NaN values
t(t == 0) = 1e-9;

% Inputting our FIR filter with smoothing
filter = (cos((wlow+whigh)/2 *t).*sin(alpha*t).*(0.54+0.46*cos(alpha
   /3 * t))./(pi*t);

% Calculating length of filter for frequency domain
nlen = length(filter);

% Creating the frequency domain for fftshifted data
fftdomain = (-nlen/2:1/25600:nlen/2)*(fs/nlen)*2*pi;

% Calculating gain of filter (in frequency domain) and removing it
   by
% dividing it from the filter (and re-ffting)
fft_FIR = fftshift(abs(fft(filter)));
gain = max(fft_FIR);
filter = filter/gain;

% Plotting the fftdomain along with the db scale of the fftshifted
% magnitude spectrum, on a semilog plot
semilogx(fftdomain, db(abs(fftshift(fft(filter, 1536001)))), 'c', '
   linewidth', 2)
%%

% Calculating the nyquist frequency (half our fs, then multiply by 2
    pi
```

Skyler Hallinan
hallisky@uw.edu
1732227

```
% since we convert to rads/sec
nyq  = fs*pi;

% Implementing the butterworth filter with an order of 2 (1*2 = 2)
   and with
% cutoff frequencies relative to calculated nyquist frequency
[b,a] = butter(1, [wlow/nyq, whigh/nyq]);

% Converting butterworth coefficients into plotting variable and
   axes, with
% 400000 points, and paremeter fs*2*pi
[h f] = freqz(b,a, 400000, fs*2*pi);

% Plotting butterworth filter on same plot, db scale, and with the
   axes
% generated in previous step
semilogx(f, db(h), 'm', 'linewidth', 2);

%% Graphing edits
% Setting bounds on graph for graphing
xlim([min(omegas), max(omegas)]);
ylim([-80 0])

% Adding pass band lines
xline(whigh, 'g--','linewidth', 1)
xline(wlow, 'r--','linewidth', 1)

% Adding legend
legend("RLC Circuit", "FIR Filter", "Butterworth IIR Filter", ...
    'Upper cutoff frequency', 'Lower cutoff frequency', 'location',
      'best', 'fontsize', 18);
```