**April 16, 2010**

**MATLAB Analysis of EMG Filtering Techniques**

**ECE 503 – Digital Signal Processing**

**Carlos Lazo**

# Consultations

None.

# Verification of Required Software

**MATLAB CODE for emg\_sim():**

% Carlos Lazo

% ECE 503 Final Project

% MATLAB Analysis of EMG Signal Filtering Techniques

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The function emg\_sim() serves to simulate an EMG signal %

% that has been sampled at 2048Hz. This signal will then %

% be used to help compare different filtering techniques. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The following are inputs to the function:

% t\_s = length of EMG signal, in seconds

% Rel\_Mag = relative magnitude of noise with respect to EMG

% The following are outputs to the function:

% EMG\_sigC = EMG signal without interference (clean)

% EMG\_sigN = EMG signal with 60Hz interference (noisy)

function [EMG\_sigC, EMG\_sigN] = emg\_sim(t\_s, Rel\_Mag)

% Define all static variables:

Fc = 150; % Cutoff Freq of lowpass filter, in Hz

Fn = 60; % Frequency of power line interference

Fs = 2048; % Sampling Frequency of EMG, in Hz

N\_filt = 4; % Order for lowpass filter

% The length of the vector should be:

% Length(EMG) = Sampling Frequency \* Time + Transient

n = floor(Fs \* t\_s) + (N\_filt - 1);

% !!!!!!!!!!!!!!!!!!!!!!!! %

% ! Function Declaration ! %

% !!!!!!!!!!!!!!!!!!!!!!!! %

% Seed the random number generator, and create a Gaussian

% random vector that will model the EMG signal.

randn ('state', sum (100\*clock));

EMG\_sig = randn(1, n);

% Create a 4th-order Butterworth lowpass filter, and pass

% the EMG signal through it. Cutoff frequency is at 150Hz.

% The resulting signal will be the "clean" EMG.

% Eliminate the startup transient from the lowpass filter.

[b, a] = butter(N\_filt, (Fc)/(Fs/2), 'low');

EMG\_sigC = filter(b, a, EMG\_sig);

EMG\_sigC = EMG\_sigC (N\_filt : n);

% Compute the RMS value of the "clean" EMG.

RMS = sqrt (mean (EMG\_sigC .^ 2));

% Create an appropriately-scaled signal to model the 60Hz

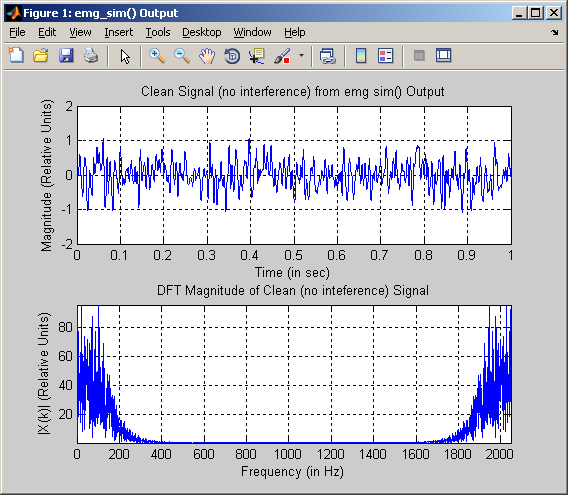
% power-line interference.

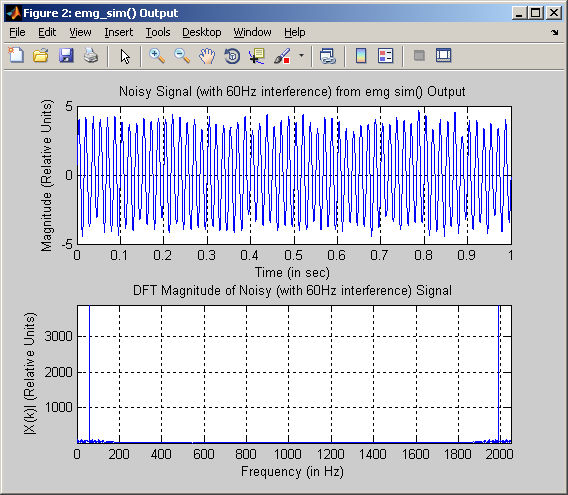
Noise\_sig = RMS \* Rel\_Mag \* sin(2 \* pi \* (Fn / Fs) \* (1:(n - N\_filt + 1)) );

% Add the power line interference into the clean signal.

EMG\_sigN = EMG\_sigC + Noise\_sig;

**Sample Output for emg\_sim():**





**MATLAB CODE for emg\_notch60():**

% Carlos Lazo

% ECE 503 Final Project

% MATLAB Analysis of EMG Signal Filtering Techniques

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The function emg\_notch60() will apply a notch-filtering %

% technique to the EMG signal with the 60Hz power-line %

% interference. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The following are inputs to the function:

% EMG\_sigN = EMG signal with interference that is being filtered

% N\_Filt = Order of the notch filter

% The following are outputs to the function:

% EMG\_sigF = EMG signal that has been filtered

function EMG\_sigF = emg\_notch60(EMG\_sigN, N\_Filt)

% Define all static variables:

Fn = 60; % Frequency of power line interference

Fs = 2048; % Sampling Frequency of EMG, in Hz

Fb = 5; % Define the overall bandwidth of the filter

% which will be 2\*Fb

Fb\_N = Fb / (Fs/2); % Normalize Fb in terms of Fs for filtering,

% then define the normalized range at where

% the stopband filter will effectively

% notch out the frequency of interest (Fn)

W\_n = [(((Fn)/(Fs/2)) - Fb\_N) (((Fn)/(Fs/2)) + Fb\_N)];

% !!!!!!!!!!!!!!!!!!!!!!!! %

% ! Function Declaration ! %

% !!!!!!!!!!!!!!!!!!!!!!!! %

% Use an IIR stopband Butterworth filter to clean the signal using a

% zero-phase linear time-invariant (LTI) filter. The output from the

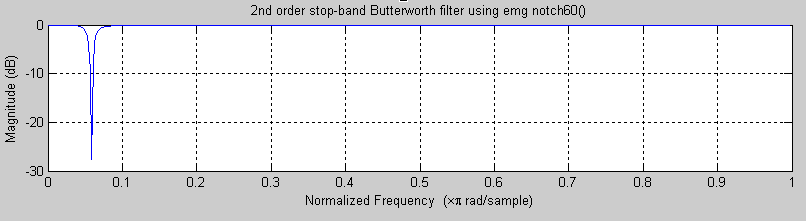
% filter will have the magnitude squared.

[b,a] = butter(N\_Filt/2, W\_n, 'stop');

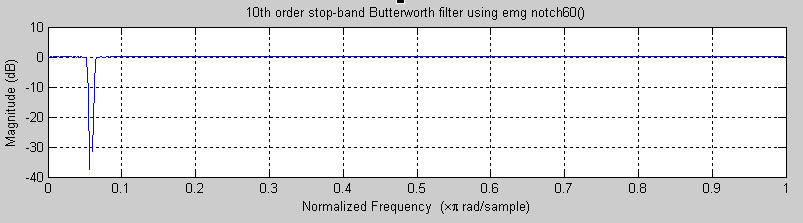
EMG\_sigF = filtfilt(b, a, EMG\_sigN);

**Filter magnitude responses from emg\_notch60():**

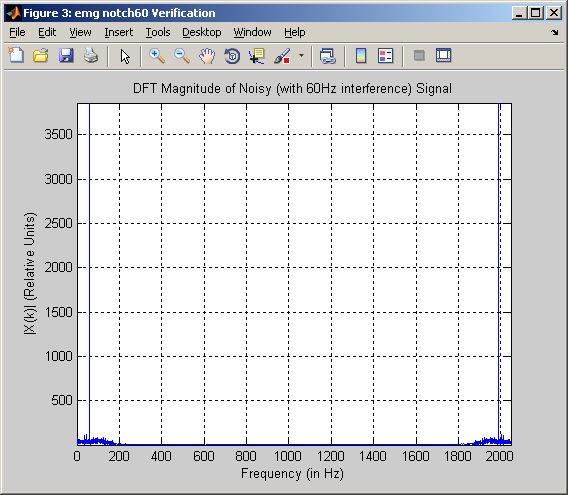
*Low Order Filter (N = 2):*

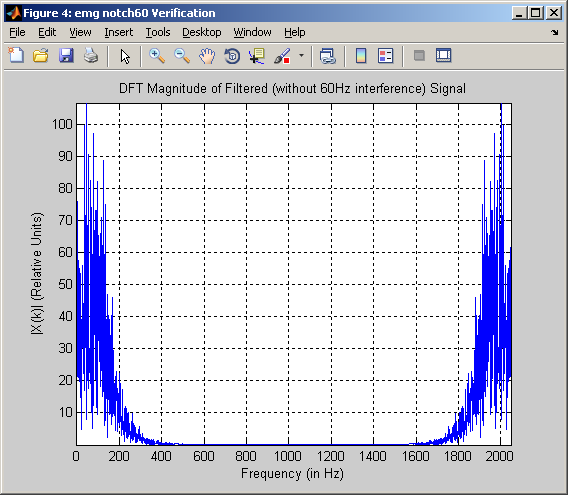


*High Order Filter (N = 10):*



**Filtering on Noisy signal using emg\_notch60():**





**MATLAB CODE for emg\_freq\_null():**

% Carlos Lazo

% ECE 503 Final Project

% MATLAB Analysis of EMG Signal Filtering Techniques

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The function emg\_freq\_null() will apply a DFT frequency- %

% nulling technique to the EMG signal with the 60Hz power- %

% line interference. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The following are inputs to the function:

% EMG\_sigN = EMG signal with interference that is being filtered

% The following are outputs to the function:

% EMG\_sigF = EMG signal that has been filtered

function EMG\_sigF = emg\_freq\_null(EMG\_sigN)

% Define all static variables:

Fn = 60; % Frequency of power line interference

Fs = 2048; % Sampling Frequency of EMG, in Hz

Fb = 5; % Define the overall bandwidth of the filter

% which will be 2\*Fb

N = length(EMG\_sigN); % Length of DFT of noisy EMG signal

% !!!!!!!!!!!!!!!!!!!!!!!! %

% ! Function Declaration ! %

% !!!!!!!!!!!!!!!!!!!!!!!! %

% Compute the N-length DFT of the noisy EMG signal:

X\_k = fft(EMG\_sigN, N);

% Compute the indeces at where the 60Hz signal is located.

% Add +1 to the index since 0Hz is included in the signal.

k1 = ((Fn \* N) / Fs) + 1;

k2 = N - k1 + 2;

% Compute the nulling-range based off the given bandwidth.

R = ((Fb \* N) / Fs) + 1;

% Null the frequencies around both positive and negative

% locations of the DFT.

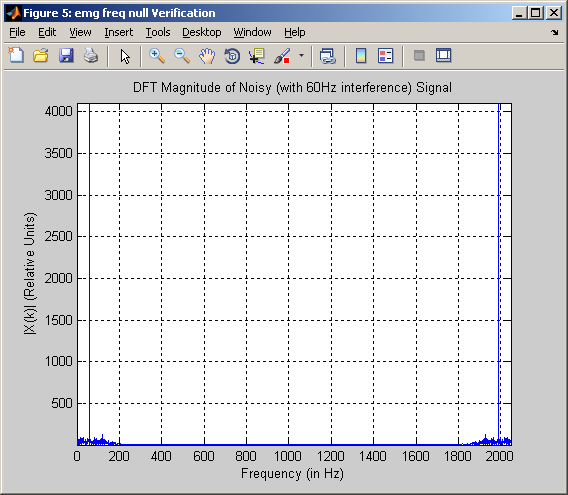
X\_k ( (k1 - R) : (k1 + R) ) = 0;

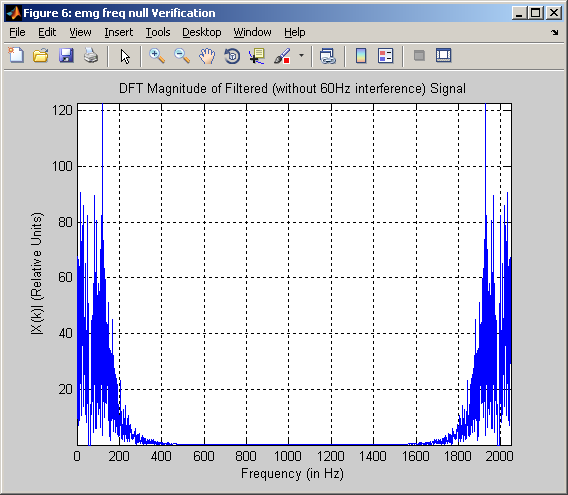
X\_k ( (k2 - R) : (k2 + R) ) = 0;

% Take the inverse DFT to get the filtered signal.

EMG\_sigF = ifft(X\_k);

**Filtering on Noisy signal using emg\_freq\_null():**





**MATLAB CODE for emg\_compare():**

% Carlos Lazo

% ECE 503 Final Project

% MATLAB Analysis of EMG Signal Filtering Techniques

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The function emg\_compare() serves to output a scalar representation %

% of how close the filtered, noisy signal is to the actual clean %

% EMG generated at the beginning of the simulation. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The following are inputs to the function:

% EMG\_sigC = EMG signal without interference (clean)

% EMG\_sigF = EMG signal with 60Hz interference removed (filtered)

% The following are outputs to the function:

% Rel\_Mag = relative magnitude of noise with respect to EMG

function perc\_diff = emg\_compare(EMG\_sigC, EMG\_sigF)

% Define all static variables:

RMS\_C = sqrt (mean (EMG\_sigC .^ 2));

RMS\_F = sqrt (mean (EMG\_sigF .^ 2));

% !!!!!!!!!!!!!!!!!!!!!!!! %

% ! Function Declaration ! %

% !!!!!!!!!!!!!!!!!!!!!!!! %

% With both of the RMS values of the signals computed,

% figure out the percent difference between the original

% and filtered versions:

perc\_diff = abs((RMS\_C - RMS\_F) / RMS\_C) \* 100;

# Analysis: Notch Filter and its Start-Up Transient

## Methods

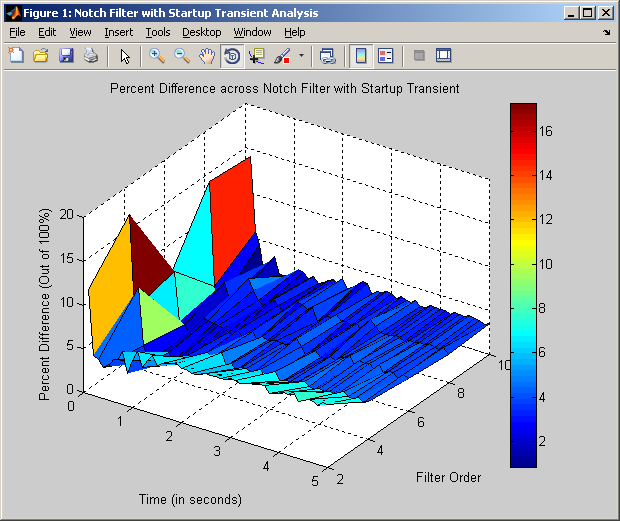
This section of the analysis deals primarily with understanding the effects of applying a notch filter centered around 60Hz to a noisy EMG signal with incorporated power line interference, without eliminating startup transients.

In order to effectively create a graphical simulation that looks across the signal duration and filter order variable spaces, while taking into account the randomness of the EMG signal, a surface map from each run of the simulation was created. Given the range from 100 ms to 5 s, a granularity of 100 ms was chosen. Having chosen to model the notch filter as an Infinite Impulse Response (IIR) Butterworth filter, filter orders from 2 to 10 were examined in increments of 2. A bandwidth of 10Hz was used for the analysis. Due to the randomness of the signal created each time *emg\_sim()* is called, a total of 10 different EMG signal pairs were created for each combination of signal duration and filter order. After using *emg\_compare()* on all 10 signal sets for each data pair, the average of all the comparisons was calculated, which represented the overall percent difference between the clean EMG and the filtered EMG for that specific signal duration & filter order.

The analysis was performed in MATLAB, using 3 nested for-loops to iterate across the variable spaces. Utilizing the *surf* command, a colored surface plot is generated to showcase the different magnitudes (percent difference) of the comparison function. The value of the signal duration and filter order that minimizes the difference between the clean signal and the filtered signal is also displayed, along with the average percent difference across all variable spaces.

## Results

**Notch Filter and its Start-Up Transient Simulation – RUN #1**



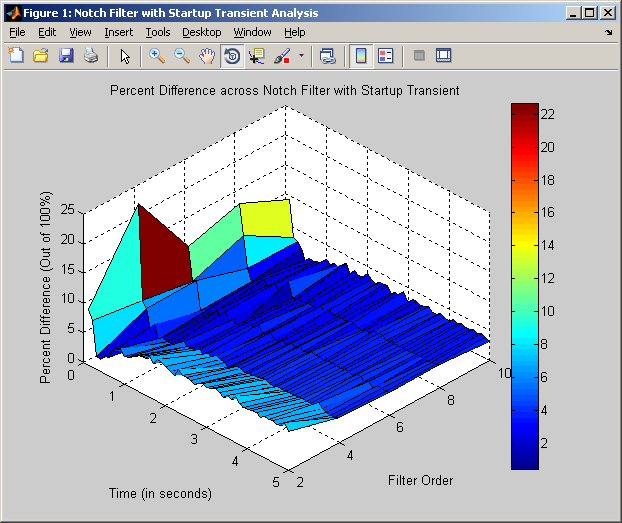
*Minimum difference* = 0.8390 %

*Signal Length of Minimum Difference* = 0.6000 sec

*Filter Order of Minimum Difference* = 6

*Average Difference* = 4.0955 %

**Notch Filter and its Start-Up Transient Simulation – RUN #2**



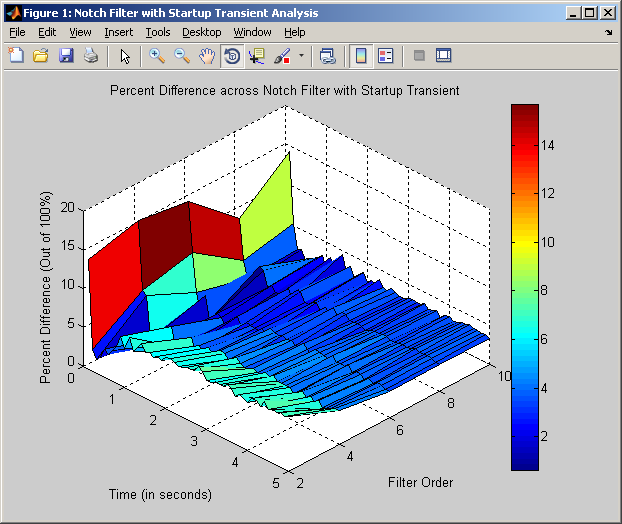
*Minimum difference* = 0.4199 %

*Signal Length of Minimum Difference* = 0.5000 sec

*Filter Order of Minimum Difference* = 10

*Average Difference* = 3.9499 %

**Notch Filter and its Start-Up Transient Simulation – RUN #3**



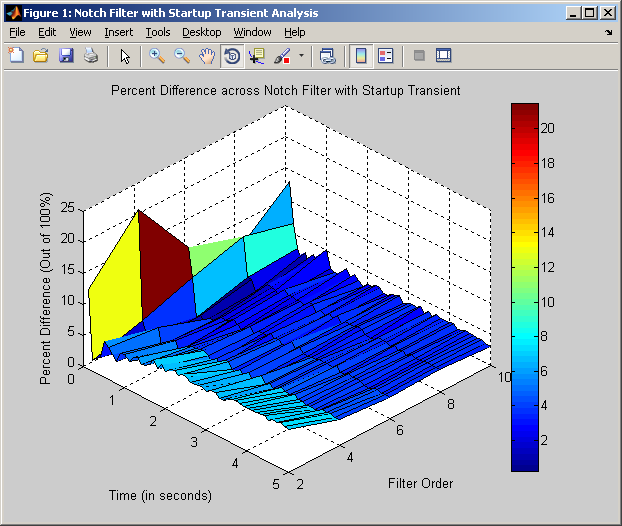
*Minimum difference* = 0.6347 %

*Signal Length of Minimum Difference* = 0.6000 sec

*Filter Order of Minimum Difference* = 6

*Average Difference* = 3.9804 %

**Notch Filter and its Start-Up Transient Simulation – RUN #4**



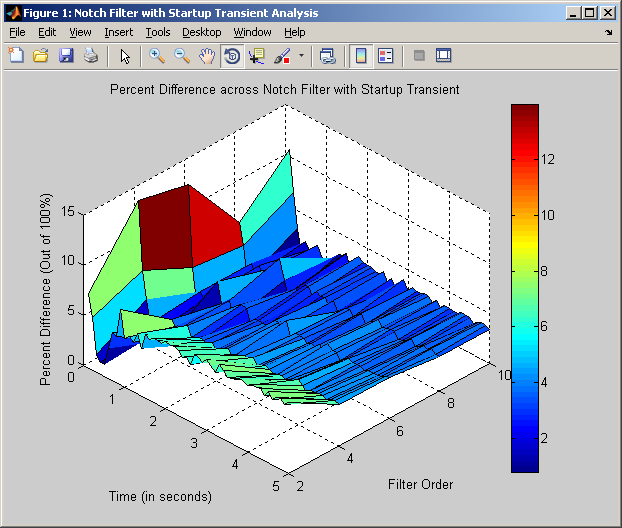
*Minimum difference* = 0.2441 %

*Signal Length of Minimum Difference* = 0.5000 sec

*Filter Order of Minimum Difference* = 6

*Average Difference* = 4.0090 %

**Notch Filter and its Start-Up Transient Simulation – RUN #5**



*Minimum difference* = 0.8003 %

*Signal Length of Minimum Difference* = 0.3000 sec

*Filter Order of Minimum Difference* = 8

*Average Difference* = 3.9425 %

## Discussion

In looking at the above simulations and at the overall percent differences seen between the clean EMG and the filtered EMG at the various signal durations and filter orders, observations can be made. Again, the filter implementation was a Butterworth IIR.

* All timing values that minimized the percent differences between the clean EMG and the filtered EMG were less than 1.0 seconds. This is less than 1/5 of the total variable space for signal length. In other words, the filtering was *most effective* when the signal duration was short.
* In looking at the graphs and the color scales, it is seen that the percent difference between the graphs is minimized as the filter order increases. This is to be expected – as the system grows in complexity (more filter coefficients), overall performance and attenuation at the cutoff frequency in question, 60 Hz, should improve.
* When the signal duration is small, the percent difference is the largest. This is due to the effect that the startup transient has on the smaller duration signals.
* The overall average difference across the simulations shown is 4.000%, implying that the filtered signal is incredibly close, in average RMS, to the original signal.

Overall, the 60Hz notch filtering technique yields an output that is approximately 4% different than the original, clean input. It is also important to note that the pairs of signal duration and filter order which match the original signal the most occur at smaller signal durations and higher filter orders.

# Analysis: Notch Filter WITHOUT Start-Up Transient

## Methods

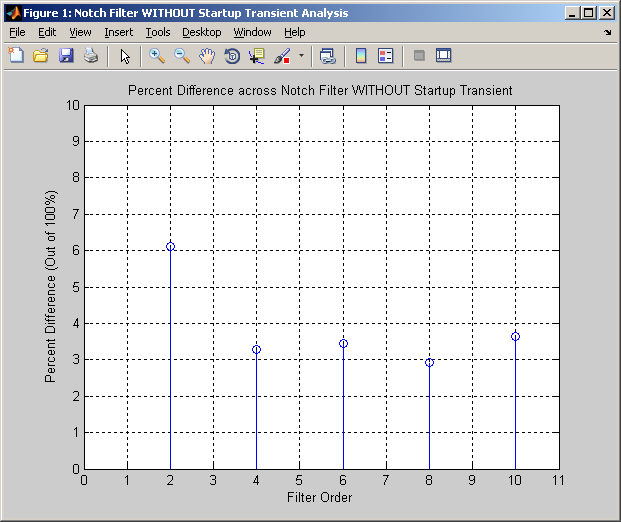
This section of the analysis deals primarily with understanding the effects of applying a notch filter centered around 60Hz to a noisy EMG signal with incorporated power line interference, while eliminating startup transients.

In order to effectively create a simulation that iterates across the order of the notch filter, while taking into account the randomness of the EMG signal, a stem plot was generated for each simulation. The plot maps the filter order vs. the percent difference seen for EMG signals of a defined duration, which have had the head and tail transients removed. A signal duration of 3 s was chosen, where exactly .5 s (500 ms) of samples were removed from the noisy signal before filtering. This means that the noisy signal was reduced to 2 s. Having chosen to model the notch filter as an Infinite Impulse Response (IIR) Butterworth filter, filter orders from 2 to 10 were examined in increments of 2. A bandwidth of 10Hz was used for the analysis. Due to the randomness of the signal created each time *emg\_sim()* is called, a total of 10 different EMG signal pairs were created for each filter order. After using *emg\_compare()* on all 10 signal sets for each data pair, the average of all the comparisons was calculated, which represented the overall percent difference between the clean EMG and the filtered EMG for that specific filter order.

The analysis was performed in MATLAB, using 2 nested for-loops to iterate across the variable spaces. Utilizing the *stem* command, a plot is generated to showcase the different magnitudes (percent difference) of the comparison function. The value of the filter order that minimizes the difference between the clean signal and the filtered signal is displayed, along with the average percent difference across all variable spaces.

## Results

**Notch Filter WITHOUT Start-Up Transient Simulation – RUN #1**

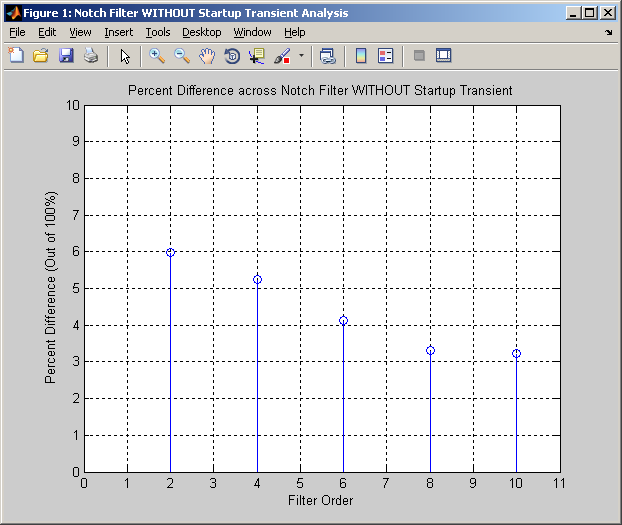


*Minimum difference* = 2.9223 %

*Filter Order of Minimum Difference* = 8

*Average Difference* = 3.8820 %

**Notch Filter WITHOUT Start-Up Transient Simulation – RUN #2**

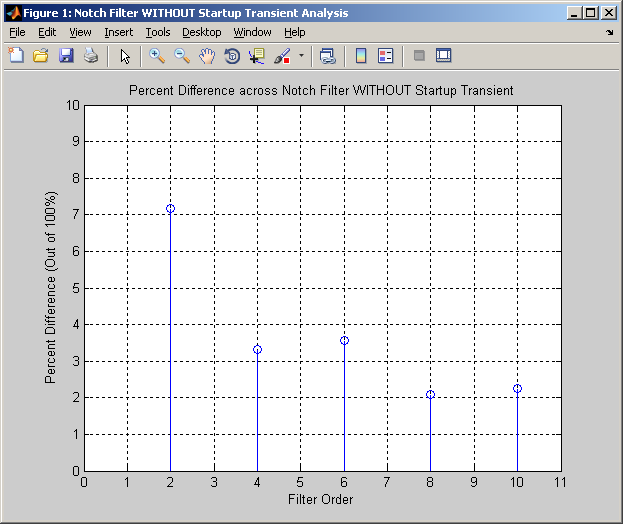


*Minimum difference* = 3.2364 %

*Filter Order of Minimum Difference* = 10

*Average Difference* = 4.3795 %

**Notch Filter WITHOUT Start-Up Transient Simulation – RUN #3**

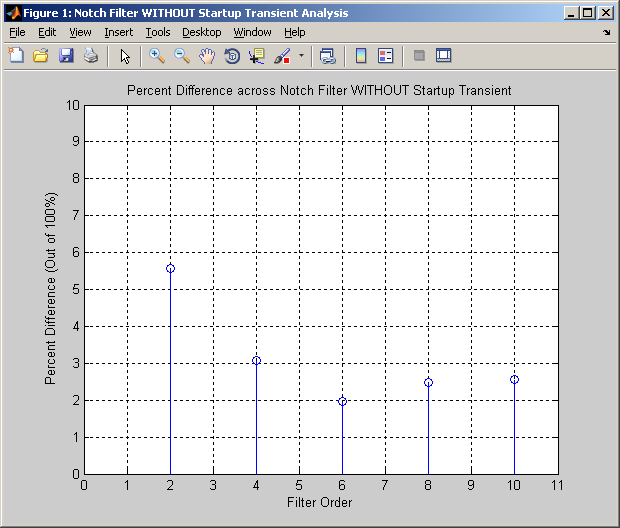


*Minimum difference* = 2.0983 %

*Filter Order of Minimum Difference* = 8

*Average Difference* = 3.6911 %

**Notch Filter WITHOUT Start-Up Transient Simulation – RUN #4**

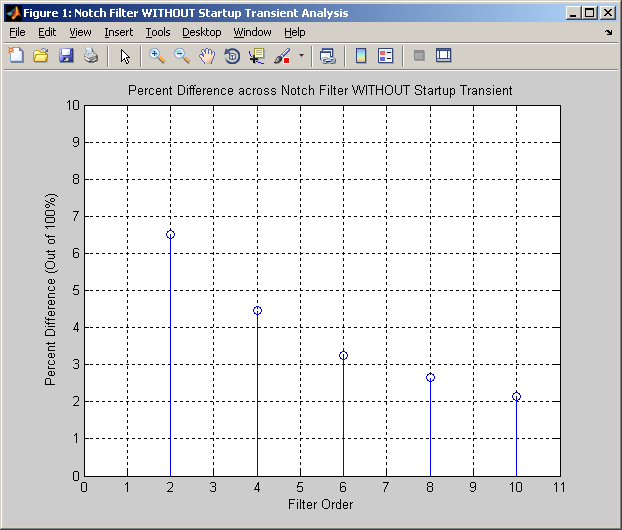


*Minimum difference* = 1.9730 %

*Filter Order of Minimum Difference* = 6

*Average Difference* = 3.1262 %

**Notch Filter WITHOUT Start-Up Transient Simulation – RUN #5**



*Minimum difference* = 2.1431 %

*Filter Order of Minimum Difference* = 10

*Average Difference* = 3.8108 %

## Discussion

In looking at the above simulations and at the overall percent differences seen between the clean EMG and the filtered EMG at the various filter orders, observations can be made. Again, the filter implementation was a Butterworth IIR.

* In looking at the graphs, it is seen that the percent difference between the graphs is minimized as the filter order increases. This is to be expected – as the system grows in complexity (more filter coefficients), overall performance and attenuation at the cutoff frequency in question, 60 Hz, should improve.
* The overall average difference across the simulations shown is 3.77%, implying that the filtered signal is incredibly close, in average RMS, to the original signal.
  + In comparison to Section 3, which included startup transients, eliminating startup transients further minimizes the difference between the original signal and the filtered signal, as expected. With the removal of the transients, the clean signal and the filtered signal match ~6% more than with the transients in place. IIR filters tend to be much lower in order than FIR filters, implying that the startup transient wouldn’t have a tremendous effect with average-sized signals. This analysis still shows that removing the transient does improve the overall matching of the filtered signal with the original signal.

Overall, the 60Hz notch filtering technique yields an output that is approximately 3.77 % different than the original, clean input. Removing the head and tail transients from the signal does, in fact, improve the overall signal match between the clean EMG and the filtered EMG.

# Analysis: Frequency Nulling

## Methods

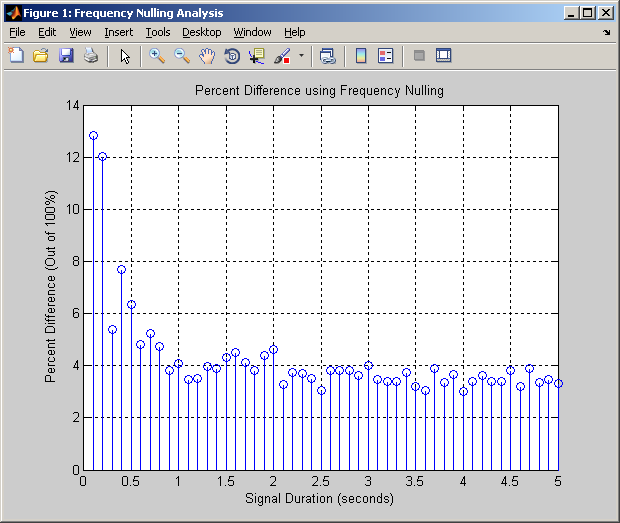
This section of the analysis deals primarily with understanding the effects of applying a frequency nulling technique centered around 60Hz to a noisy EMG signal with incorporated power line interference.

In order to effectively create a simulation that iterates across the duration of the signal, while taking into account the randomness of the EMG signal, a stem plot was generated for each simulation. The plot maps the duration of the signal vs. the percent difference seen between the clean and filtered EMG signal of that specific length. Given the range from 100 ms to 5 s, a granularity of 100 ms was chosen. A bandwidth of 10Hz was used for the analysis. Due to the randomness of the signal created each time *emg\_sim()* is called, a total of 10 different EMG signal pairs were created for each signal duration. After using *emg\_compare()* on all 10 signal sets for each data pair, the average of all the comparisons was calculated, which represented the overall percent difference between the clean EMG and the filtered EMG for that specific signal duration.

The analysis was performed in MATLAB, using 2 nested for-loops to iterate across the variable spaces. Utilizing the *stem* command, a plot is generated to showcase the different magnitudes (percent difference) of the comparison function. The value of the signal duration that minimizes the difference between the clean signal and the filtered signal is displayed, along with the average percent difference across all variable spaces.

## Results

**Frequency Nulling Simulation – RUN #1**

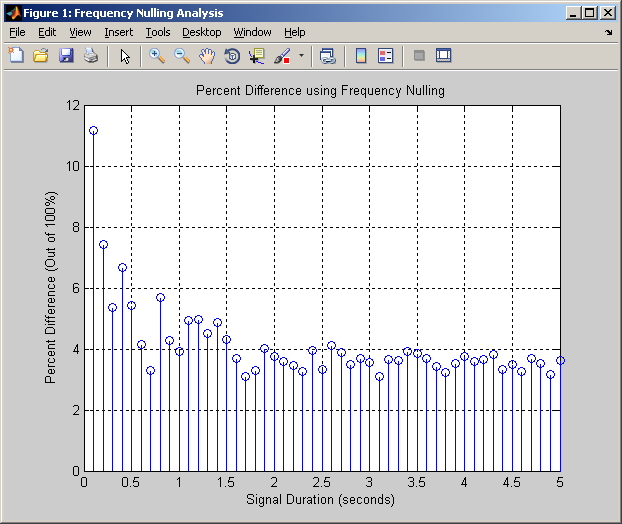


*Minimum difference* = 3.0004 %

*Signal Duration of Minimum Difference* = 4.0000

*Average Difference* = 4.2591 %

**Frequency Nulling Simulation – RUN #2**

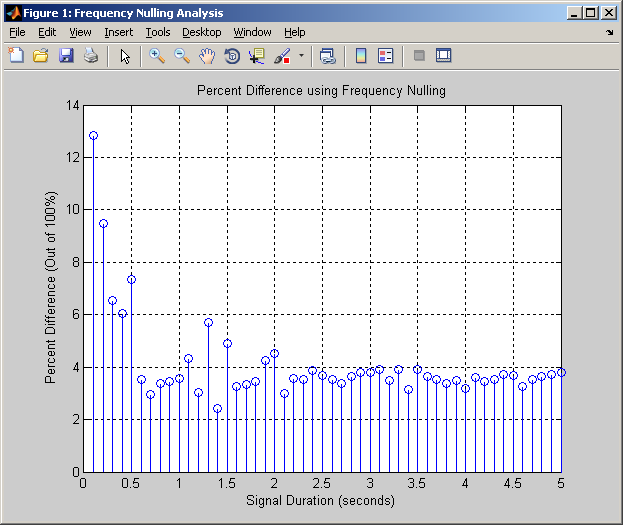


*Minimum difference* = 3.0944 %

*Signal Duration of Minimum Difference* = 1.7000

*Average Difference* = 4.1200 %

**Frequency Nulling Simulation – RUN #3**

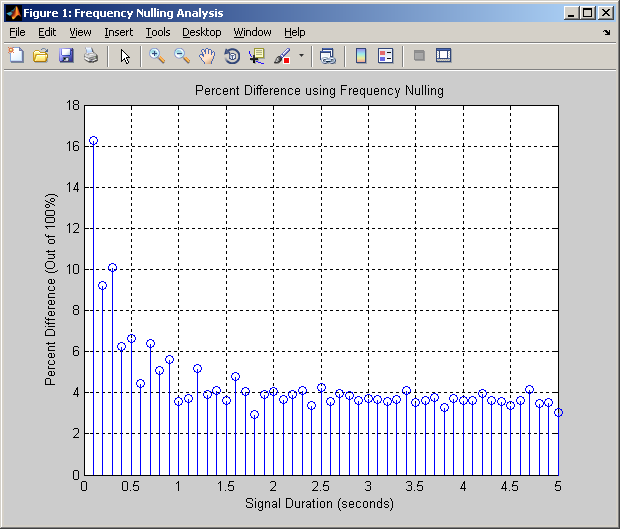


*Minimum difference* = 2.4256 %

*Signal Duration of Minimum Difference* = 1.4000

*Average Difference* = 4.1146 %

**Frequency Nulling Simulation – RUN #4**

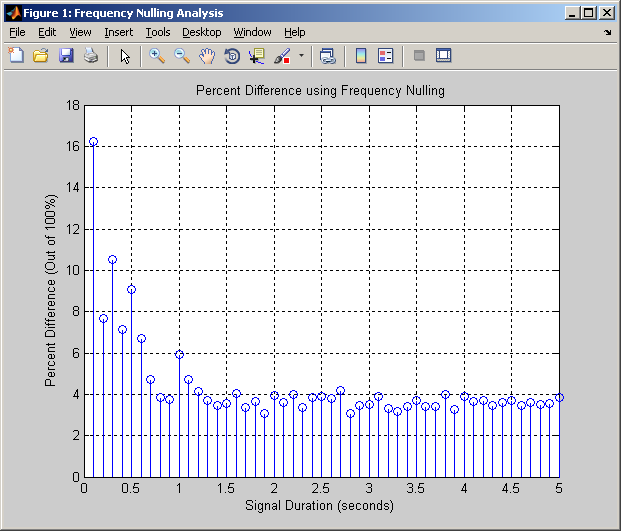


*Minimum difference* = 2.9229 %

*Signal Duration of Minimum Difference* = 1.8000

*Average Difference* = 4.4938 %

**Frequency Nulling Simulation – RUN #5**



*Minimum difference* = 3.0891 %

*Signal Duration of Minimum Difference* = 2.8000

*Average Difference* = 4.4307 %

## Discussion

In looking at the above simulations and at the overall percent differences between the clean and filtered EMG signals at the various signal durations, a few observations can be made:

* In looking at the graphs, it is seen that the percent difference between the graphs is minimized as the signal duration increases. This is to be expected – as the signal grows in size, the overall effect of the startup transient decreases. In other words, the larger the duration of the EMG, the less the startup transient will drive the output.
* In looking at the graphs, the signal duration for the minimum value has more variability than when doing notch filtering (with or without transients).
* When using frequency nulling, the smallest minimum difference seen is 2.4256%, based on the five simulations shown above. This is much higher than the minimum values seen with the notch filtering technique.
* The overall average difference across the simulations shown is 4.28%, implying that the filtered signal is incredibly close, in average RMS, to the original signal.
  + In comparison to Sections 3 & 4 (involving notch filtering), the frequency nulling technique generates a higher average difference overall.

Overall, the frequency nulling technique yields an output that is approximately 4.28 % different than the original, clean input. It performs poorly at extremely low signal lengths due to the effects of the startup transient, but does fairly well at filtering out the 60Hz noise at the higher signal duration times.

# Analysis: Relative Signal to Noise (SNR) Magnitude

## Methods

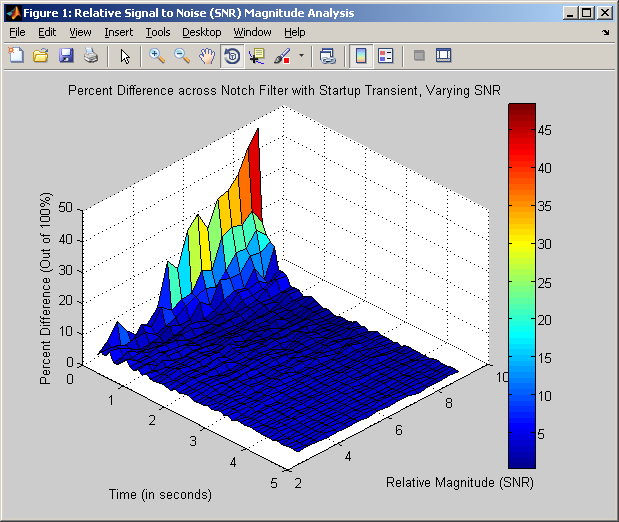
This section of the analysis deals primarily with understanding the effects of applying a notch filter centered around 60Hz to a noisy EMG signal with incorporated power line interference, without eliminating startup transients. In this case, the filter order will be fixed at N = 6, or half the range of the original analysis, and the relative magnitude will be changed.

In order to effectively create a graphical simulation that looks across the signal duration and relative magnitude variable spaces, while taking into account the randomness of the EMG signal, a surface map from each run of the simulation was created. Given the range from 100 ms to 5 s, a granularity of 100 ms was chosen. Relative magnitudes (SNRs) between 2 and 10 at a granularity of .5 were chosen for the analysis. A bandwidth of 10Hz was used for the analysis. Due to the randomness of the signal created each time *emg\_sim()* is called, a total of 10 different EMG signal pairs were created for each combination of signal duration and filter order. After using *emg\_compare()* on all 10 signal sets for each data pair, the average of all the comparisons was calculated, which represented the overall percent difference between the clean EMG and the filtered EMG for that specific signal duration & relative magnitude.

The analysis was performed in MATLAB, using 3 nested for-loops to iterate across the variable spaces. Utilizing the *surf* command, a colored surface plot is generated to showcase the different magnitudes (percent difference) of the comparison function. The value of the signal duration and relative magnitude that minimizes the difference between the clean signal and the filtered signal is also displayed, along with the average percent difference across all variable spaces.

## Results

**Relative Magnitude Simulation – RUN #1**



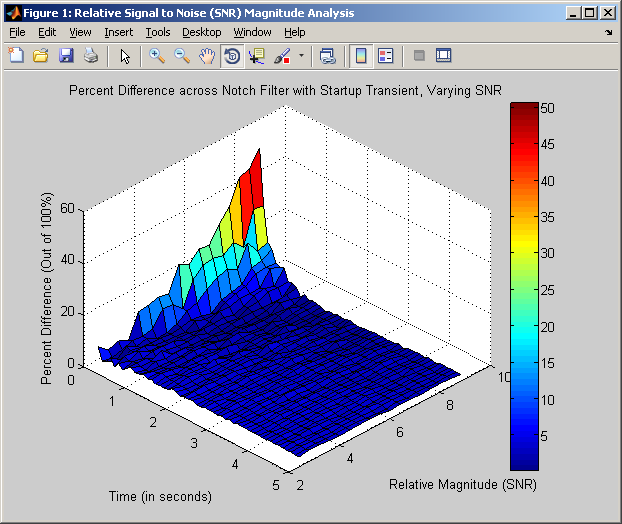
*Minimum difference* = 0.2669 %

*Signal Length of Minimum Difference* = 2.0000 sec

*Relative Magnitude of Minimum Difference* = 10

*Average Difference* = 3.5964 %

**Relative Magnitude Simulation – RUN #2**



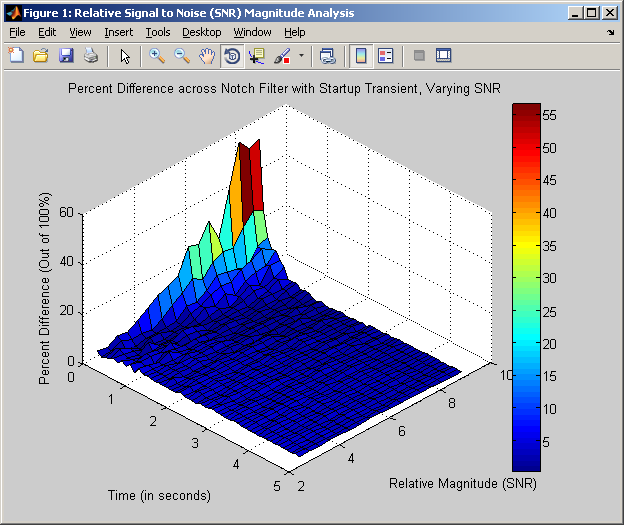
*Minimum difference* = 0.1761 %

*Signal Length of Minimum Difference* = 1.1000 sec

*Relative Magnitude of Minimum Difference* = 8.5

*Average Difference* = 3.6197 %

**Relative Magnitude Simulation – RUN #3**



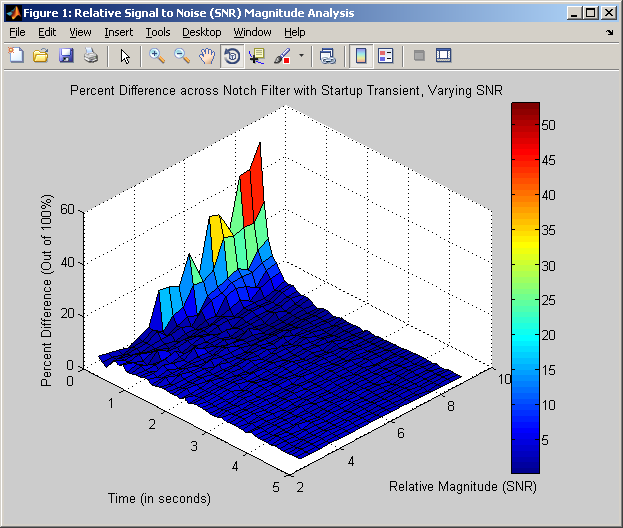
*Minimum difference* = 0.2820 %

*Signal Length of Minimum Difference* = 1.8000 sec

*Relative Magnitude of Minimum Difference* = 10

*Average Difference* = 3.6278 %

**Relative Magnitude Simulation – RUN #4**



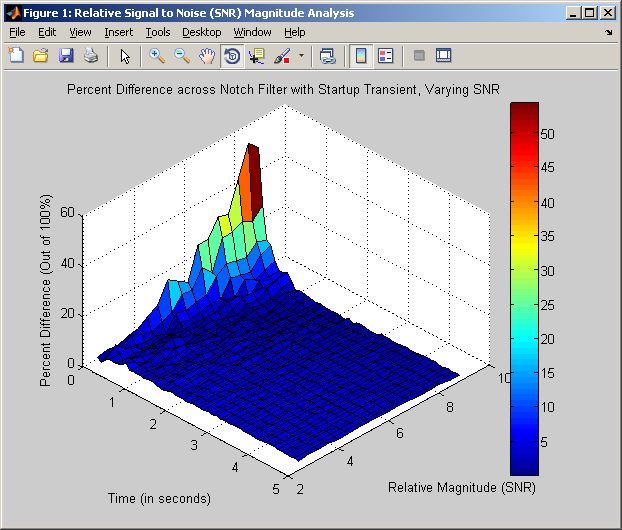
*Minimum difference* = 0.2625 %

*Signal Length of Minimum Difference* = 1.5000 sec

*Relative Magnitude of Minimum Difference* = 8.5

*Average Difference* = 3.5779 %

**Relative Magnitude Simulation – RUN #5**



*Minimum difference* = 0.0781 %

*Signal Length of Minimum Difference* = 0.3000 sec

*Relative Magnitude of Minimum Difference* = 5

*Average Difference* = 3.5960 %

## Discussion

In looking at the above simulations and at the overall percent differences seen between the clean EMG and the filtered EMG at the various signal durations and relative magnitudes, observations can be made. Again, the filter implementation was a Butterworth IIR.

* All timing values that minimized the percent differences between the clean EMG and the filtered EMG were in the 1st half of the variable spaced analyzed. In other words, the filtering at varying relative magnitudes, in general, was *most effective* when the signal duration was short.
* In looking at the graphs and the color scales, it is seen that the percent difference between the graphs is minimized as the relative magnitude increases. All the graphs above have their minimum percent differences at a relative magnitude at or above 5. At the higher end of relative magnitude variable space, almost the exact same amount of overall attenuation is seen, causing nearly identical percent differences.
* When the signal duration is small, the percent difference is the largest. This is due to the effect that the startup transient has on the smaller duration signals.
* The overall average difference across the simulations shown is 3.6037%, implying that the filtered signal is incredibly close, in average RMS, to the original signal.

Overall, varying the relative magnitude at different signal durations has little to no adverse effects on the filter performance. As the relative magnitude of the clean signal and its 60 Hz interference increase, the filter still does an adequate job at attenuation and essentially eliminating the noise. It is also important to note that the pairs of signal duration and relative magnitude which match the original signal the most occur at smaller signal durations and higher relative magnitudes.

# Discussion and Conclusion

Overall, the three filtering techniques end up having their advantages and disadvantages:

* Notch Filtering with Start-Up Transients
  + (+) : Consistent average differences across the different variables, along with a low threshold for overall RMS difference.
  + (–) : Filtering occurs best at lower signal durations with higher filter order, but yield high contributions from the filter start-up transients.
* Notch Filtering Without Start-Up Transients
  + (+) : Lowest average differences across the different variables, along with a low threshold for overall RMS difference.
  + (–) : Eliminating transients from the beginning and end of the sequences could potentially require more signal to be captured upfront, which may not always be possible based on the equipment at hand.
* Frequency Nulling
  + (+) : Technique is independent of filter order, and is instead dependant on the DFT bandwidth span (in Hz) that is to be nulled. This allows for greater flexibility in the frequency domain.
  + (–) : Highest average difference across the different variables, along with an relatively inconsistent average minimum signal duration.
* Further analysis has also shown that the relative magnitude, or Signal to Noise Ratio (SNR), of the interference with respect to the original, clean EMG has little effect on the overall performance of the filter and signal recovery effort.

In general, all techniques were able to achieve an RMS difference between the original signal and the filtered signal of less than 5%, which may or may not be good enough depending on the situation as it applies to the problem at hand.

The accuracy of the information presented in this analysis could be enhanced, and there are definite areas of improvement that can be analyzed that would help gain a much clearer and more defined understanding of which techniques would work better under certain situations:

* Running each of the simulations across many more than just 5 sets of data. In this way, the average difference and all associated outputs will be closer to the actual mean of the simulation vs. a small approximation of the mean.
* In these simulations, a bandwidth of 10Hz was chosen for the notch filter and the frequency nulling techniques. If the bandwidth for filtering was added as a variable, then the problem could be further optimized. A trade-space between signal duration, filtering bandwidth, and filter order could be performed.
* Modeling the filter as either an FIR / IIR, along with varying the different sorts of filters (Butterworth, Chebychev, etc.), would allow one to see if one type of filter works better than another. Linear phase implementations of FIR / IIR are different, and the transient play a significant role in the overall RMS error difference of the clean vs. filtered signals.

The simulations do an accurate job at representing a real-world case where digital signal processing plays an important role in filtering data used for the recording of skeletal electromyograms (EMGs). The randomness factor utilized in creating the simulated EMGs is the sort of outcome one would expect in an actual engineering application, and further adds to the validity of the modeling and analysis presented in this report.

# MATLAB Appendix

**Main Program Developed for ALL MATLAB Simulations:**

***NOTE:*** *Each code block was run ‘individually’ within MATLAB to produce output.*

% Carlos Lazo

% ECE 503 Final Project

% MATLAB Analysis of EMG Signal Filtering Techniques

%% 2. Verification of Required Software - emg\_sim()

clear all; close all; clc;

% This block of code will generate all graphs required for the

% second section of the Report.

% Define all static variables.

Fs = 2048;

t\_s = .1;

Rel\_Mag = 10;

% Generate clean and noisy signals.

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

% Setup plotting vectors and compute the DFTs of both sequences:

t = (1:floor(Fs\*t\_s)) / Fs;

X\_kC = fft(EMG\_sigC);

fC = ((0 : length(X\_kC) - 1) \* Fs / length(X\_kC));

X\_kN = fft(EMG\_sigN);

fN = ((0 : length(X\_kN) - 1) \* Fs / length(X\_kN));

% First, plot the clean signal and its DFT magnitude:

figure('Name', 'emg\_sim() Output');

subplot(2,1,1);

plot (t, EMG\_sigC);

xlabel('Time (in sec)');

ylabel('Magnitude (Relative Units)');

title ('Clean Signal (no interference) from emg sim() Output');

grid on;

subplot(2,1,2);

plot(fC, abs(X\_kC));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Clean (no inteference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kC)) max(abs(X\_kC))];

axis(lim);

% Second, plot the clean signal and its DFT magnitude:

figure('Name', 'emg\_sim() Output');

subplot(2,1,1);

plot (t, EMG\_sigN);

xlabel('Time (in sec)');

ylabel('Magnitude (Relative Units)');

title ('Noisy Signal (with 60Hz interference) from emg sim() Output');

grid on;

subplot(2,1,2);

plot(fN, abs(X\_kN));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Noisy (with 60Hz interference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kN)) max(abs(X\_kN))];

axis(lim);

%% 2. Verification of Required Software - emg\_notch()

clear all; close all; clc;

% For the emg\_notch60() analysis, copy the code that is used to make

% the filter from emg\_notch60() to show the frequency responses:

Fn = 60; % Frequency of power line interference

Fs = 2048; % Sampling Frequency of EMG, in Hz

Fb = 5; % Define the overall bandwidth of the filter

% which will be 2\*Fb

Fb\_N = Fb / (Fs/2); % Normalize Fb in terms of Fs for filtering,

% then define the normalized range at where

% the stopband filter will effectively

% notch out the frequency of interest (Fn)

W\_n = [(((Fn)/(Fs/2)) - Fb\_N) (((Fn)/(Fs/2)) + Fb\_N)];

N\_FiltL = 2;

N\_FiltH = 10;

% Develop lower and higher order Butterworth filters and plot

% the magnitude responses:

[bL,aL] = butter(N\_FiltL/2, W\_n, 'stop');

[bH,aH] = butter(N\_FiltH/2, W\_n, 'stop');

figure('Name', 'emg notch60() Filter Design');

freqz(bL,aL);

title('2nd order stop-band Butterworth filter using emg notch60()');

grid on;

figure('Name', 'emg notch60() Filter Design');

freqz(bH,aH);

title('10th order stop-band Butterworth filter using emg notch60()');

grid on;

% Define static variables:

Fs = 2048;

t\_s = 1;

Rel\_Mag = 10;

% Generate clean and noisy signals,

% then filter the signal through emg\_notch60():

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

EMG\_sigF = emg\_notch60(EMG\_sigN, N\_FiltH);

% Setup plotting vectors and compute the DFTs of both sequences:

t = (1:floor(Fs\*t\_s)) / Fs;

X\_kN = fft(EMG\_sigN);

fN = ((0 : length(X\_kN) - 1) \* Fs / length(X\_kN));

X\_kF = fft(EMG\_sigF);

fF = ((0 : length(X\_kF) - 1) \* Fs / length(X\_kF));

% First, plot the DFT magnitude of the noisy signal:

figure('Name', 'emg notch60 Verification');

plot(fN, abs(X\_kN));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Noisy (with 60Hz interference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kN)) max(abs(X\_kN))];

axis(lim);

% First, plot the DFT magnitude of the noisy signal after filtering:

figure('Name', 'emg notch60 Verification');

plot(fF, abs(X\_kF));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Filtered (without 60Hz interference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kF)) max(abs(X\_kF))];

axis(lim);

%% 2. Verification of Required Software - emg\_freq\_null()

% Define static variables:

Fs = 2048;

t\_s = 1;

Rel\_Mag = 10;

% Generate clean and noisy signals,

% then filter the signal through emg\_freq\_null():

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

EMG\_sigF = emg\_freq\_null(EMG\_sigN);

% Setup plotting vectors and compute the DFTs of both sequences:

t = (1:floor(Fs\*t\_s)) / Fs;

X\_kN = fft(EMG\_sigN);

fN = ((0 : length(X\_kN) - 1) \* Fs / length(X\_kN));

X\_kF = fft(EMG\_sigF);

fF = ((0 : length(X\_kF) - 1) \* Fs / length(X\_kF));

% First, plot the DFT magnitude of the noisy signal:

figure('Name', 'emg freq null Verification');

plot(fN, abs(X\_kN));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Noisy (with 60Hz interference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kN)) max(abs(X\_kN))];

axis(lim);

% First, plot the DFT magnitude of the noisy signal after filtering:

figure('Name', 'emg freq null Verification');

plot(fF, abs(X\_kF));

xlabel('Frequency (in Hz)');

ylabel('|X(k)| (Relative Units)');

title ('DFT Magnitude of Filtered (without 60Hz interference) Signal');

grid on;

lim = [0 Fs min(abs(X\_kF)) max(abs(X\_kF))];

axis(lim);

%% 3. Analysis: Notch Filter and its Start-Up Transient

clear all; close all; clc;

% Determine granularity for this simulation:

t\_g = .1; % Iterate every 100ms

N\_g = 2; % Increment by filter orders of 2

max\_r = 10; % Max # of repetitions for each t\_s sample

Rel\_Mag = 5; % Signal to Noise Ratio (SNR)

% Ranging from t\_s = 100ms - 5sec and from N = 2 to 10,

% develop a contour map of comparisons between the original signal

% and the filtered signal at the varying different combinations of

% time and filter order.

i = 1;

j = 1;

perc\_diff = zeros(i,j);

rep\_vals = zeros(1,max\_r);

for t\_s = .1 : t\_g : 5

for N\_filt = 2 : N\_g : 10

for r = 1 : max\_r

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

EMG\_sigF = emg\_notch60(EMG\_sigN, N\_filt);

rep\_vals(r) = emg\_compare(EMG\_sigC, EMG\_sigF);

end

perc\_diff(i,j) = mean(rep\_vals);

i = i + 1;

end

i = 1;

j = j + 1;

end

% Generate a surface plot of the values:

figure('Name', 'Notch Filter with Startup Transient Analysis');

surf(perc\_diff);

set(gca,'XTickLabel',(0 : 1 : 5));

set(gca,'YTickLabel',(2 : N\_g : 10));

colorbar;

xlabel ('Time (in seconds)');

ylabel ('Filter Order');

zlabel ('Percent Difference (Out of 100%)');

title ('Percent Difference across Notch Filter with Startup Transient');

[C,I] = min(perc\_diff(:));

% Find the smallest percent different between the original

% signal and the filtered signal, and find the corresponding

% time value and filter order.

R = mod(I,5);

if (R == 0)

R = 5;

end

C = ceil(I/5);

minVal\_ts = C \* t\_g;

minVal\_N = R \* N\_g;

minVal = perc\_diff(R,C);

avgDiff = mean(perc\_diff(:));

%% 4. Analysis: Notch Filter WITHOUT its Start-Up Transient

clear all; close all; clc;

% Determine granularity for this simulation:

Fs = 2048; % Sampling frequency, in Hz

t\_s = 3; % Analyze a signal of 3 seconds

N\_g = 2; % Increment by filter orders of 2

max\_r = 10; % Max # of repetitions for each filter iteration

Rel\_Mag = 5; % Signal to Noise Ratio (SNR)

% Ranging from N = 2 to 10 and through truncating the transients,

% develop a plot of comparisons between the original signal

% and the filtered signal at the varying different combinations of

% filter order.

i = 1;

perc\_diff = zeros(i);

rep\_vals = zeros(1,max\_r);

for N\_filt = 2 : N\_g : 10

for r = 1 : max\_r

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

% Truncate teh signal down to 2 seconds by discarding

% .5 seconds at the head and tail, respectively.

EMG\_sigN = EMG\_sigN( (1 + (Fs/2)) : ( (Fs \* t\_s) - (Fs/2)));

EMG\_sigF = emg\_notch60(EMG\_sigN, N\_filt);

rep\_vals(r) = emg\_compare(EMG\_sigC, EMG\_sigF);

end

perc\_diff(i) = mean(rep\_vals);

i = i + 1;

end

figure('Name', 'Notch Filter WITHOUT Startup Transient Analysis');

stem ( (2 : N\_g : 10), perc\_diff);

xlabel ('Filter Order');

ylabel ('Percent Difference (Out of 100%)');

title ('Percent Difference across Notch Filter WITHOUT Startup Transient');

axis ([0 11 0 10]);

grid on;

[C,I] = min(perc\_diff);

minVal\_N = I \* N\_g;

minVal = perc\_diff(I);

avgDiff = mean(perc\_diff);

%% 5. Analysis: Frequency Nulling

clear all; close all; clc;

% Determine granularity for this simulation:

t\_g = .1; % Iterate every 100ms

max\_r = 10; % Max # of repetitions for each filter iteration

Rel\_Mag = 5; % Signal to Noise Ratio (SNR)

% Ranging the signal duration from t\_s = .1 to 5 seconds,

% develop a plot of comparisons between the original signal

% and the filtered signal at the varying different combinations of

% filter order when using frequency nulling.

i = 1;

perc\_diff = zeros(i);

rep\_vals = zeros(1,max\_r);

for t\_s = .1 : t\_g : 5

for r = 1 : max\_r

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

EMG\_sigF = emg\_freq\_null(EMG\_sigN);

rep\_vals(r) = emg\_compare(EMG\_sigC, EMG\_sigF);

end

perc\_diff(i) = mean(rep\_vals);

i = i + 1;

end

figure('Name', 'Frequency Nulling Analysis');

stem ( (.1 : t\_g : 5), perc\_diff);

xlabel ('Signal Duration (seconds)');

ylabel ('Percent Difference (Out of 100%)');

title ('Percent Difference using Frequency Nulling');

%axis ([0 6 0 20]);

grid on;

[C,I] = min(perc\_diff);

minVal\_ts = I \* t\_g;

minVal = perc\_diff(I);

avgDiff = mean(perc\_diff);

%% 6. Analysis: Relative Signal to Noise (SNR) Magnitude

clear all; close all; clc;

% Determine granularity for this simulation:

r\_g = .5; % Increment the SNR by .5 each iteration

t\_g = .1; % Iterate every 100ms

N\_fix = 6; % Fix the filter order to be N = 6

max\_r = 10; % Max # of repetitions for each t\_s sample

% Ranging from t\_s = 100ms - 5sec and from N = 2 to 10,

% develop a contour map of comparisons between the original signal

% and the filtered signal at the varying different combinations of

% time and filter order.

i = 1;

j = 1;

perc\_diff = zeros(i,j);

rep\_vals = zeros(1,max\_r);

for t\_s = .1 : t\_g : 5

for Rel\_Mag = 2 : r\_g : 10

for r = 1 : max\_r

[EMG\_sigC, EMG\_sigN] = emg\_sim (t\_s, Rel\_Mag);

EMG\_sigF = emg\_notch60(EMG\_sigN, N\_fix);

rep\_vals(r) = emg\_compare(EMG\_sigC, EMG\_sigF);

end

perc\_diff(i,j) = mean(rep\_vals);

i = i + 1;

end

i = 1;

j = j + 1;

end

% Generate a surface plot of the values:

figure('Name', 'Relative Signal to Noise (SNR) Magnitude Analysis');

surf(perc\_diff);

set(gca,'XTickLabel',(0 : 1 : 5));

set(gca,'YTickLabel',(2 : 2 : 10));

colorbar;

xlabel ('Time (in seconds)');

ylabel ('Relative Magnitude (SNR)');

zlabel ('Percent Difference (Out of 100%)');

title ('Percent Difference across Notch Filter with Startup Transient, Varying SNR');

[C,I] = min(perc\_diff(:));

% Find the smallest percent different between the original

% signal and the filtered signal, and find the corresponding

% time value and filter order.

R = mod(I,17);

if (R == 0)

R = 17;

end

C = ceil(I/17);

minVal\_ts = C \* t\_g;

minVal\_r = 1.5 + R \* r\_g;

minVal = perc\_diff(R,C);

avgDiff = mean(perc\_diff(:));