Table of Contents


```
function OneD_signal_Filter_Ex

% This program was written to serve as an introductory tutorial on
 matlab
% and signal processing.  Some of the functions covered in the
 tutorial are
% fft & ifft, xcorr and audio capture. Here 1D cross correlation
 (xcorr) is
% used to aling in time various recordings. The fast Fourier transform
 and
% its inverse, fft & ifft repsectively, are used to switch signals in
 and
% out of the time and frequency domains. The effects of quantization
 level
% and ideal filters (rect function in the frequecny domain) are also
% explored in this tutorial on both recorded audio data and a dirac
 delta
% function (impulse).
%
%*************************************************************************
% Inputs: none
%
%*************************************************************************
% Outputs: none
%
%*************************************************************************
% Written by:
% Cameron Rodriguez
% Copyright © Cameron Rodriguez 2016
% cdrodriguez@g.ucla.edu
% Last Modified 2016/10/18
%
%*************************************************************************
%
% see also: audiorecorder, play, fft, ifft, xcorr,
%
%*************************************************************************
```

Lets Get Some Data

```
% Display the name of the input device to be used
    info = audiodevinfo;
    disp(info.input(1).Name)

% Recording parameters
    recLenght = 5; % Recording lenght in Sec
    Fsample = 8000; % Sampleing Frequency

% Set up recording objects for 8, 16, & 24 bit depths
    %Create a Recording Object with a Fsample of 8kHz and a 8 bit
 depth
    recObj08 = audiorecorder(Fsample, 8,1,info.input(1).ID);
    %Create a Recording Object with a Fsample of 8kHz and a 16 bit
 depth
    recObj16 = audiorecorder(Fsample,16,1,info.input(1).ID);
    %Create a Recording Object with a Fsample of 8kHz and a 24 bit
 depth
    recObj24 = audiorecorder(Fsample,24,1,info.input(1).ID);

% Record the audio at various bit depths simultaineously
    disp('Start speaking') % display "Start speaking" in the command
 window
        % Start the Recordings
            record(recObj08); record(recObj16); record(recObj24);
        % Wait for the recording to finish
            pause(recLenght);
        % Stop the Recordings
            stop(recObj08); stop(recObj16); stop(recObj24);
    disp('Recording Complete') % "Recording Complete" in the command
 window

% Wait 2 Second before begining play back of the recordings
    pause(2);

% play the 8b it recording
    play(recObj08); pause(recLenght);

% Wait 1 Sec
    pause(1);

% play the 16 bit recording
    play(recObj16); pause(recLenght);

% Wait 1 Sec
    pause(1);

% play the 24 bit recording
    play(recObj24); pause(recLenght);
```

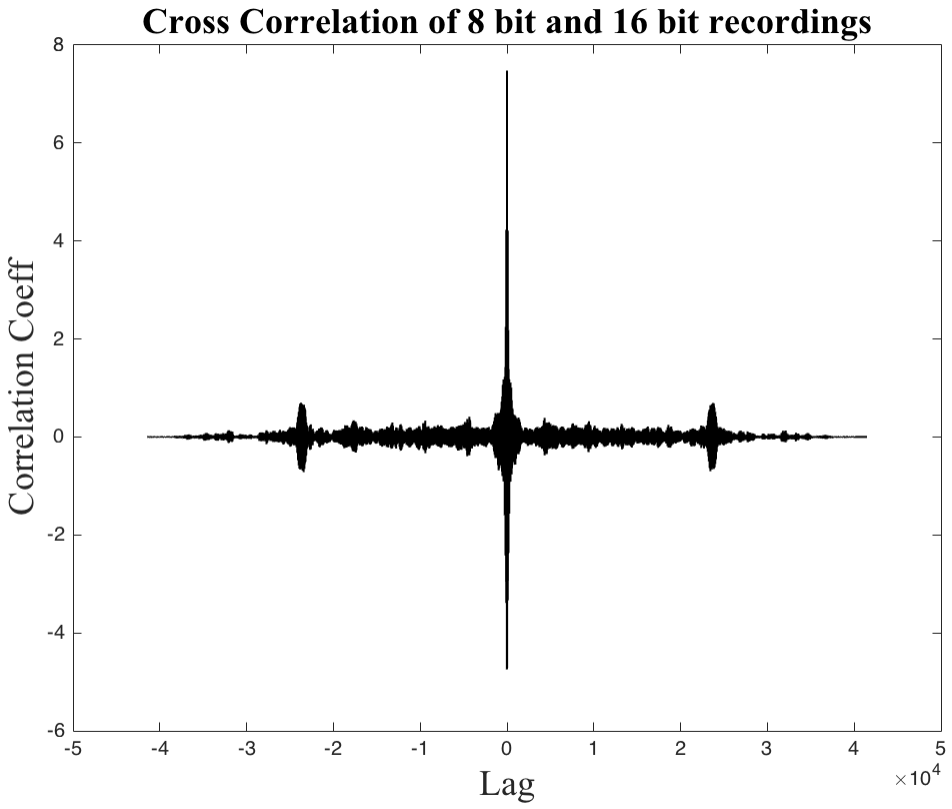
```
Built-in Microph (Core Audio)
Start speaking
```

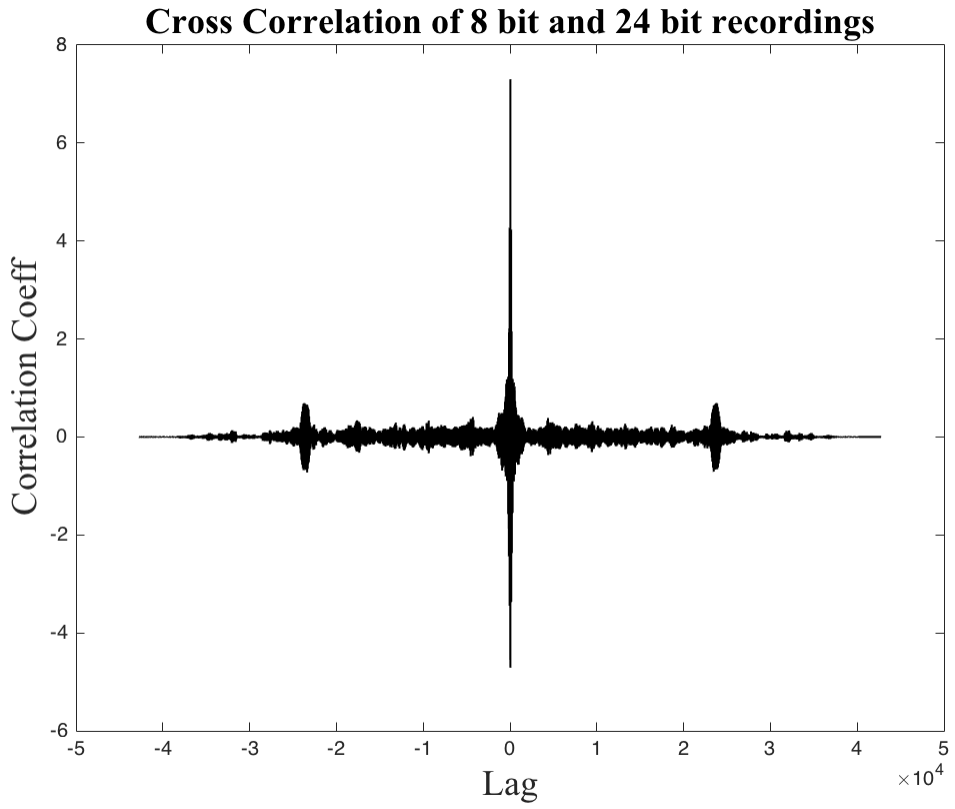
Recording Complete

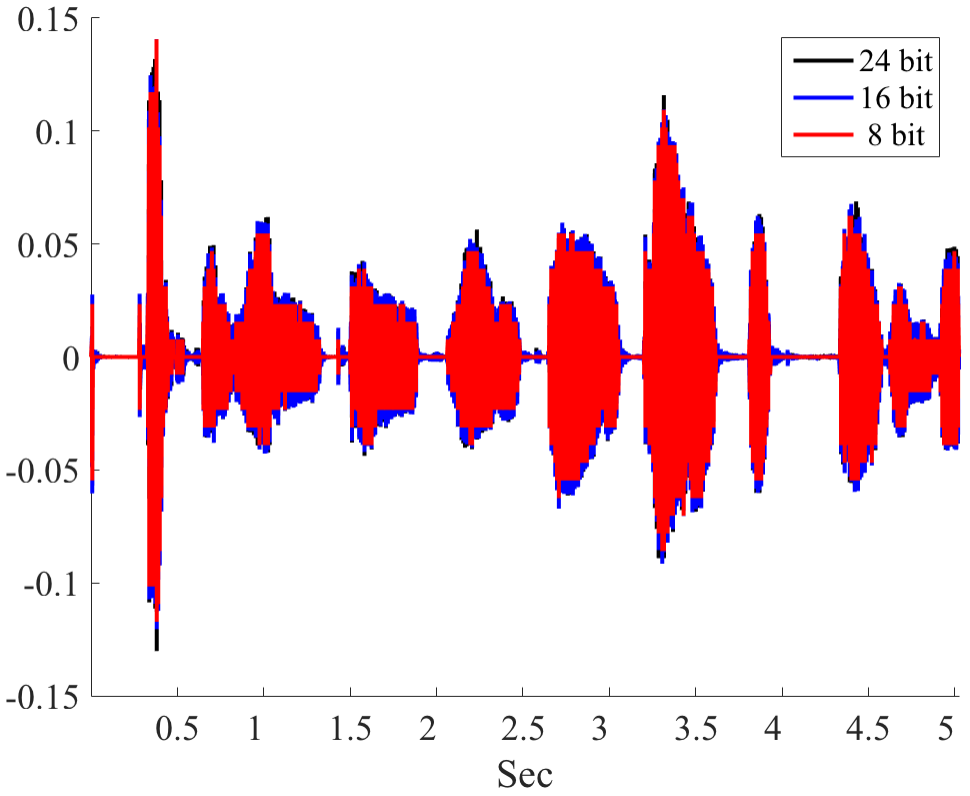
Look at the data

```
% Pull the audio data out of the recording object
    sig08 = getaudiodata(recObj08);
    sig16 = getaudiodata(recObj16);
    sig24 = getaudiodata(recObj24);

% Align The Recordings all to the 8 bit recording

% Calculate the Cross Correlation 8 bit bit & 16 recordings
    [C16,Lags16] = xcorr(sig08,sig16); % Cross Correlation
    [~, i] = max(C16); % Find the index (i) of the max correlation
    Shift16 = numel(sig16)-i; % adjust the index by the # of points
 (numel)
                              % in the second signal.  The numel in
 the
                              % correlation is equal to
                              % numel(sig1) + numel(sig1) -1

% Display the Cross Correlation 8 bit & 16 bit recordings
    figure
    plot(Lags16,C16, 'k', 'linewidth', 1)
    xlabel('Lag', 'FontSize', 18, 'FontName', 'Times New Roman')
    ylabel('Correlation Coeff', 'FontSize', 18, ...
           'FontName', 'Times New Roman')
    title('Cross Correlation of 8 bit and 16 bit recordings', ...
          'FontSize', 18, 'FontName', 'Times New Roman')

% Calculate the Cross Correlation 8 bit bit & 16 recordings
    [C24,Lags24] = xcorr(sig08,sig24);
    [~, i] = max(C24);
    Shift24 = numel(sig24)-i;

% Display the Cross Correlation 8 bit & 24 bit recordings
    figure
    plot(Lags24,C24, 'k', 'linewidth', 1)
    xlabel('Lag', 'FontSize', 18, 'FontName', 'Times New Roman')
    ylabel('Correlation Coeff', 'FontSize', 18, ...
           'FontName', 'Times New Roman')
    title('Cross Correlation of 8 bit and 24 bit recordings', ...
          'FontSize', 18, 'FontName', 'Times New Roman')

% Plot the 3 aligned signals
    figure
    hold on
        plot(-Shift24+1:(numel(sig24)-Shift24),
 sig24, 'k', 'linewidth', 2)
        plot(-Shift16+1:(numel(sig16)-Shift16),
 sig16, 'b', 'linewidth', 2)
        plot(sig08, 'r', 'linewidth', 2)
    hold off
```

```
xlabel('Sec', 'FontSize', 18, 'FontName', 'Times New Roman')
legend('24 bit', '16 bit', ' 8 bit')
set(gca, 'FontSize', 18, 'FontName', 'Times New Roman')
set(gca,'xtick', Fsample/2:Fsample/2:numel(sig08))
set(gca,'xticklabel', 0.5:0.5:round(numel(sig08)/Fsample))
xlim([1,numel(sig08)])
```

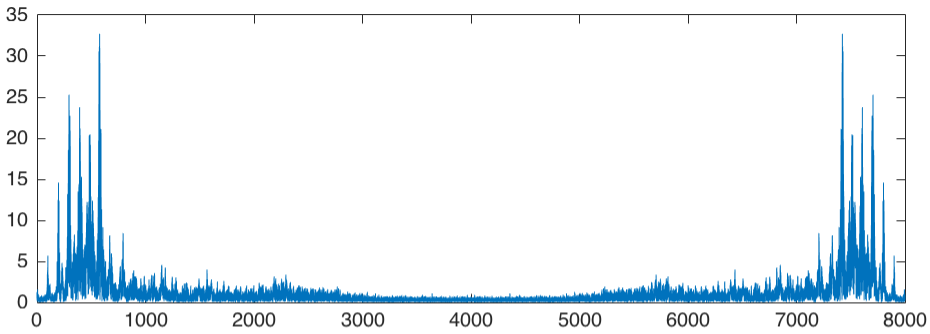


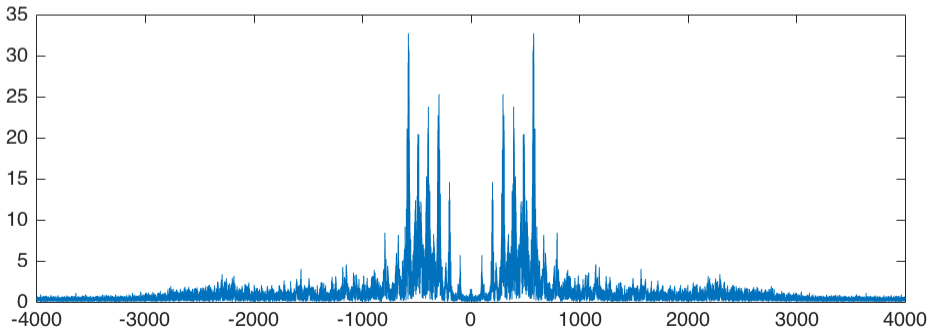

Cross Correlation of 8 bit and 24 bit recordings
Correlation Coeff
Lag
×10^4


24 bit
16 bit
8 bit
Sec

Let's look at the spectrum of the 8 bit Recording

```
% Take the FFT of the 8 bit signal
    SIG08 = fft(sig08);

% Create Corresponding Frequencies to plot against
    f1 = Fsample*linspace(0,1,round(numel(abs(SIG08)))); % 0-Fs
    f2 = Fsample*linspace(-0.5,0.5,round(numel(abs(SIG08)))); %-
Fs/2:Fs/2

% Display the magnitude Spectrum of the 8 bit recording
    figure
        subplot(2,1,1) % 0-Fs
            plot(f1,abs(SIG08))
        subplot(2,1,2) % -Fs/2-Fs/2
            plot(f2,fftshift(abs(SIG08)))
```

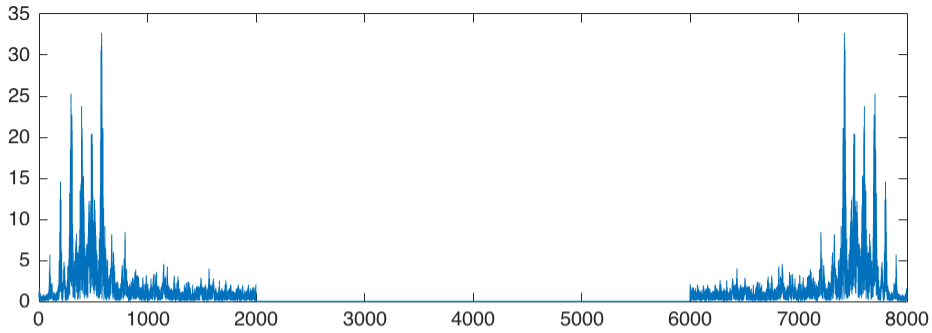


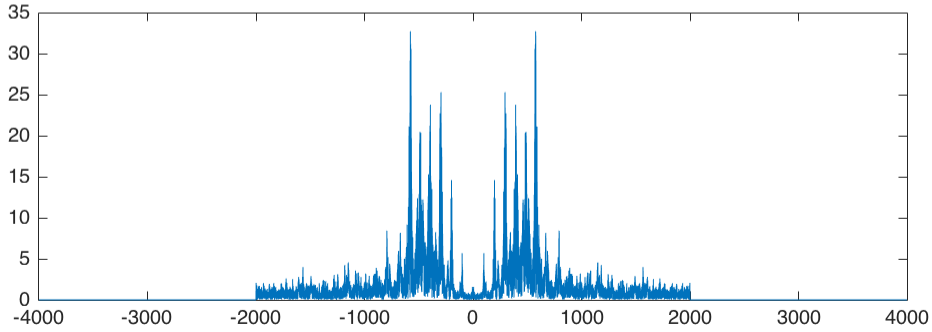


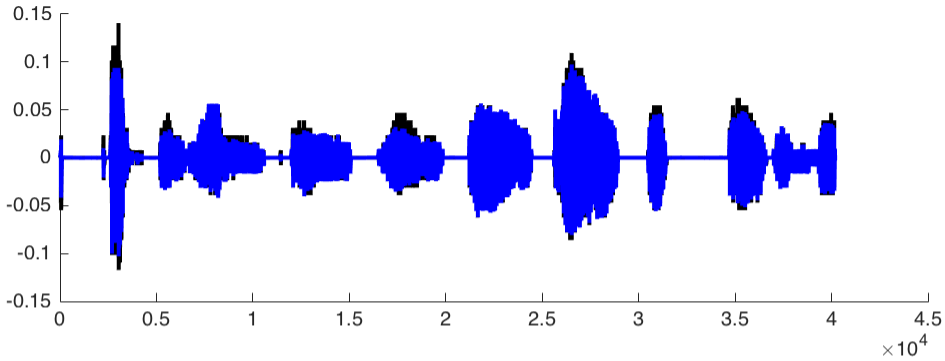
Let's LPF the 8 bit Recording

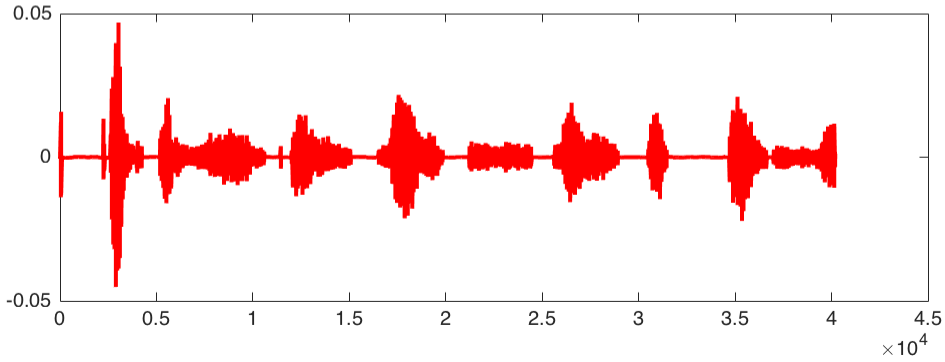
```
% Apply an "ideal" low pass filter to the data
    fcut = 2000;
    SIG08LPF =SIG08;
    SIG08LPF((abs(f1) > fcut) & (abs(f1) < (Fsample - fcut)) ) = 0;
```

```
% Display the magnitude Spectrum of the LPF 8 bit recording
    figure
        subplot(2,1,1)
            plot(f1,abs(SIG08LPF))
        subplot(2,1,2)
            plot(f2,fftshift(abs(SIG08LPF)))

% Take the inverse FFT of the LPF data
    sig08LPF = real(ifft(SIG08LPF));

% Display the time series of the raw and lpf 8 bit recordings
    figure
        subplot(2,1,1)
        hold on
            plot(sig08, 'k', 'linewidth', 2)
            plot(sig08LPF, 'b', 'linewidth', 2)
        hold off
        subplot(2,1,2)
            plot(sig08 - sig08LPF, 'r', 'linewidth', 2)

% Covert the LPF filtered data back into and audio object
    recObj08LPF = audioplayer(sig08LPF, Fsample);


% Play the 8 bit audio object again
    play(recObj08);  pause(recLenght);

% Wait 1 second
    pause(1);

% Play the 8 bit LPF audio object
    play(recObj08LPF);  pause(recLenght);
```


35
30
25
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
7000
8000

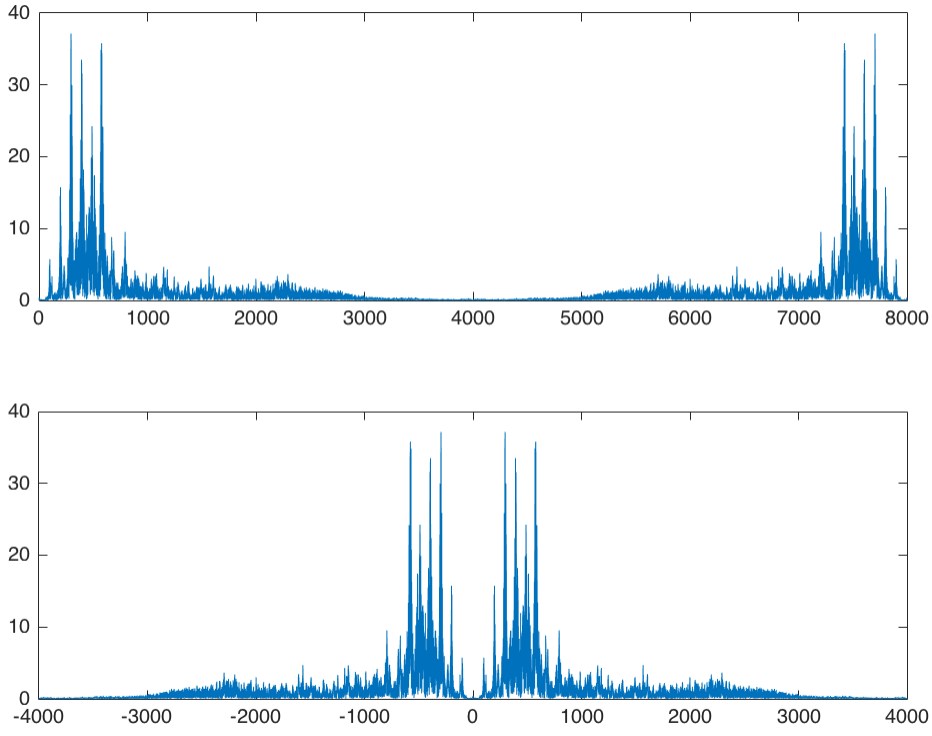

35
30
25
20
15
10
5
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000


0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^4


0.05
0
-0.05
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^4

Let's look at the spectrum of the 24 bit Recording

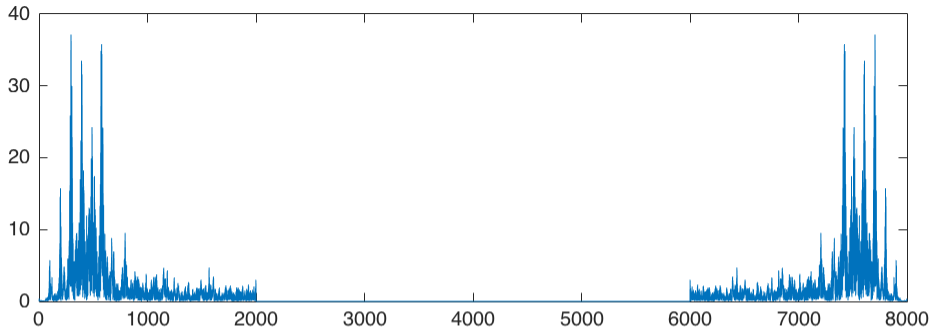
```
% Take the FFT of the 24 bit signal
    SIG24 = fft(sig24);

% Create Corresponding Frequencies to plot against
    f3 = Fsample*linspace(0,1,round(numel(abs(SIG24))));
    f4 = Fsample*linspace(-0.5,0.5,round(numel(abs(SIG24))));

% Display the magnitude Spectrum of the 8 bit recording
    figure
        subplot(2,1,1) % 0-Fs
            plot(f3,abs(SIG24))
        subplot(2,1,2) % -Fs/2:Fs/2
            plot(f4,fftshift(abs(SIG24)))
```

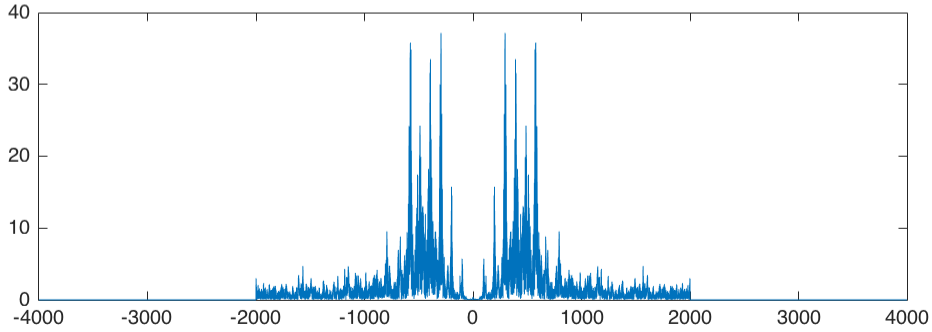


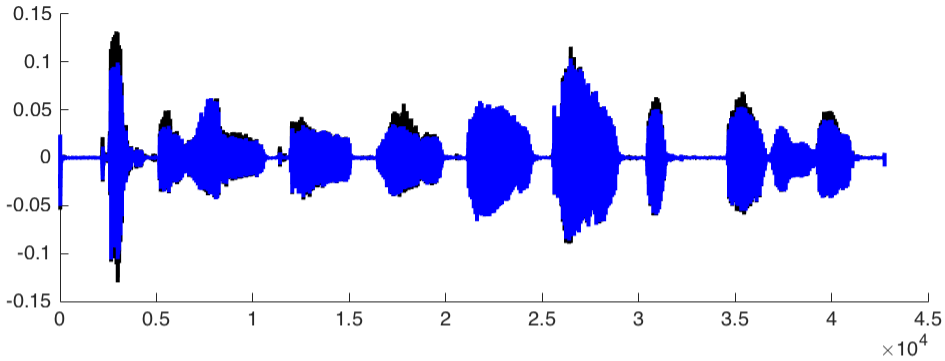
Let's LPF the 24 bit Recording

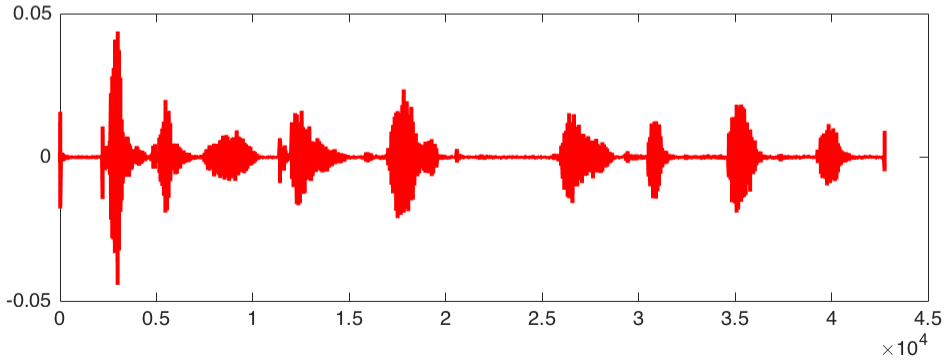
```
% Apply an "ideal" low pass filter to the data
    fcut = 2000;
    SIG24LPF =SIG24;
    SIG24LPF((abs(f3) > fcut) & (abs(f3) < (Fsample - fcut)) ) = 0;
```

```
% Display the magnitude Spectrum of the LPF 24 bit recording
    figure
        subplot(2,1,1)
            plot(f3,abs(SIG24LPF))
        subplot(2,1,2)
            plot(f4,fftshift(abs(SIG24LPF)))

% Take the inverse FFT of the LPF data
    sig24LPF = real(ifft(SIG24LPF));

% Display the time series of the raw and lpf 24 bit recordings
    figure
        subplot(2,1,1)
            hold on
                plot(sig24, 'k', 'linewidth', 2)
                plot(sig24LPF, 'b', 'linewidth', 2)
            hold off
        subplot(2,1,2)
            plot(sig24 - sig24LPF, 'r', 'linewidth', 2)

% Covert the LPF filtered data back into and audio object
    recObj24LPF = audioplayer(sig24LPF, Fsample);

% Play the 24 bit audio object again
    play(recObj24);  pause(recLenght);

% Wait 1 second
    pause(1);

% Play the 24 bit LPF audio object
    play(recObj24LPF);  pause(recLenght);
```


40
30
20
10
0
0
1000
2000
3000
4000
5000
6000
7000
8000

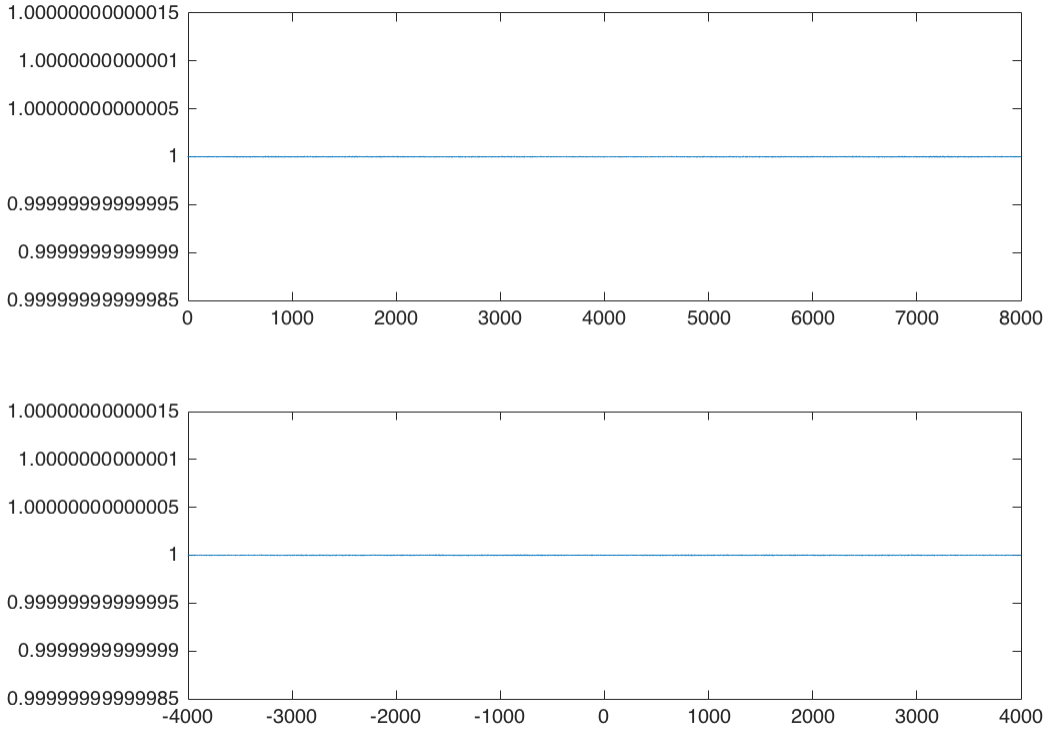

40
30
20
10
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000


0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^4


0.05
0
-0.05
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^4

Let's Look At A Spike

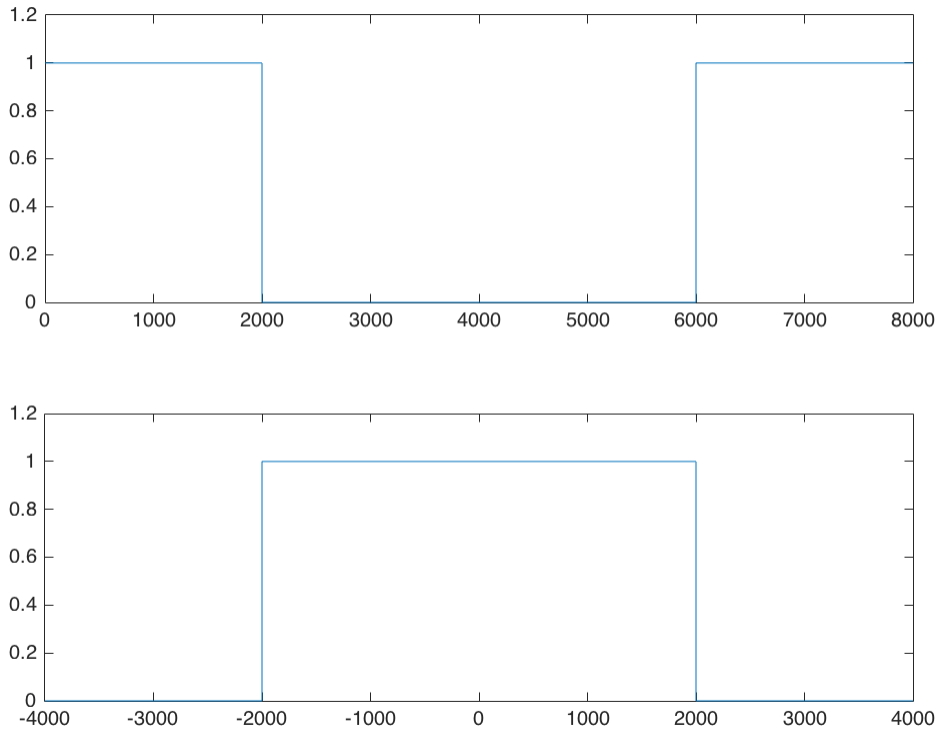
```
% Create a time series with a spike at t=0
    spike = zeros(size(sig08));
    spike(1) = 1;

% Take the FFT of the spike (impluse) time series
    SPIKE = fft(spike);

% Display the magnitude Spectrum of the impluse data
    figure
        subplot(2,1,1) % 0-Fs
            plot(f1,abs(SPIKE))
        subplot(2,1,2) % -Fs/2:Fs/2
            plot(f2,fftshift(abs(SPIKE)))
```

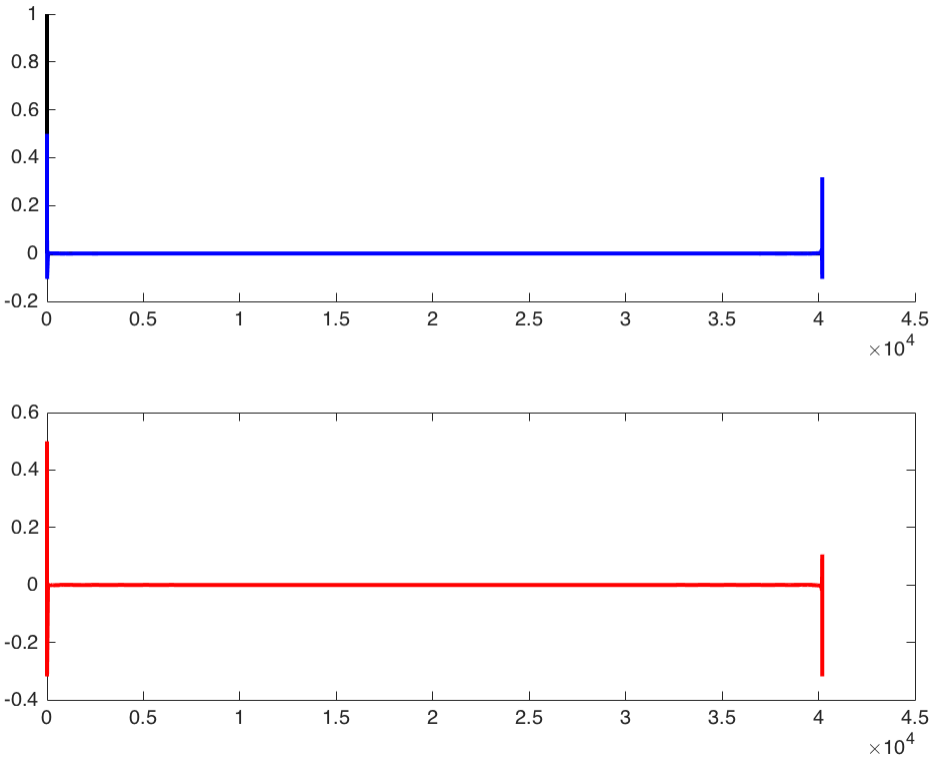


Let's LPF the Spike / Impulse Recording

```
% Apply an "ideal" low pass filter to the data
    fcut = 2000;
    SPIKELPF = SPIKE;
    SPIKELPF((abs(f1) > fcut) & (abs(f1) < (Fsample - fcut)) ) = 0;

% Display the magnitude Spectrum of the LPF impuse recording
    figure
```

```
        subplot(2,1,1)
            plot(f1,abs(SPIKELPF))
        subplot(2,1,2)
            plot(f2,fftshift(abs(SPIKELPF)))

% Take the inverse FFT of the LPF data
    spikeLPF = real(ifft(SPIKELPF));

% Display the time series of the raw and lpf spike recordings
    figure
        subplot(2,1,1)
            hold on
                plot(spike, 'k', 'linewidth', 2)
                plot(spikeLPF, 'b', 'linewidth', 2)
            hold off
        subplot(2,1,2) % Difference
            plot(spike-spikeLPF, 'r', 'linewidth', 2)
```





Published with MATLAB® R2015b