Lab 8 – Filtering Using the DFT

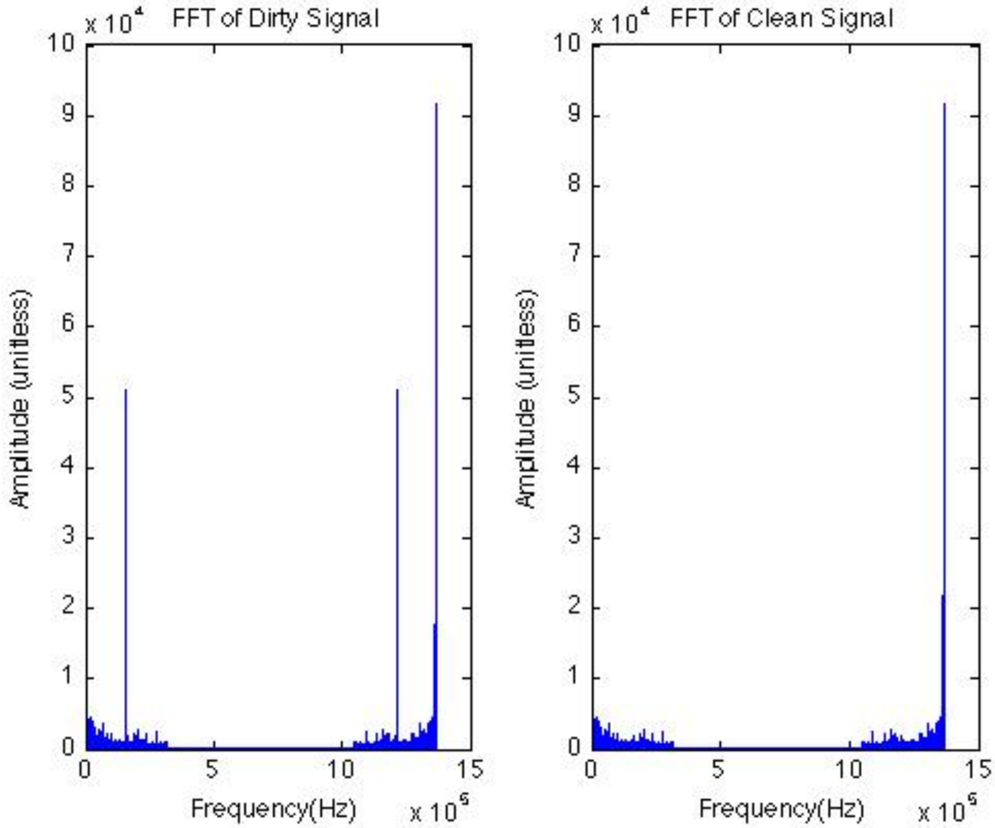
Connor McCullough

MMI 502 Fall 2012

Part 1: Eliminating Noise From A Signal

```
[y fs]=
wavread('/Users/muestudent/Desktop/Connor502Lab8/drivehard.wav');
mix=lab8(y,fs);                    % Add noise to original signal
wavwrite(mix,fs,'connorlab8dirtysignal.wav');
%soundsc(mix,fs);

fwav = fft(mix);                   % FFT of noisy signal
fwavmag = abs(fwav);
fwavphase = angle(fwav);           % Split signal into magnitude and phase
components.;
fy = fft(y);                       % FFT of original file
fdiff = abs(fwav) - abs(fy);       % Isolate undesired frequencies by
comparing dirty magnitude with clean magnitude
ffixed = abs(fwav) - fdiff;        % Subtract undesired frequency from
dirty signal

subplot(1,2,1);
plot(fwavmag);
title('FFT of Dirty Signal');
xlabel('Frequency(Hz)');
ylabel('Amplitude (unitless)');

subplot(1,2,2);
plot(ffixed);
title('FFT of Clean Signal');
xlabel('Frequency(Hz)');
ylabel('Amplitude (unitless)');

R = ffixed;
theta = fwavphase;                 % put into form Y = R*e^(j)(theta)
e = 2.71828182846;                 % Define e


Y = R.*e.^(1i*theta);               % Reconstruct complex spectrum

ynew = ifft(Y);                    % Transform signal back into time time
domain.
ynewreal = real(ynew);             % Only use the real part

soundsc(ynewreal,fs);
wavwrite(ynewreal,fs,'connorlab8cleansignal.wav');
```

This code first adds noise to a sound file using the given function. The noise is eliminated by finding the FFT of both the pre-noise and post-noise signals, measuring the difference, and subtracting the difference from the post-noise signal. The signal is than reconstructed using the ifft function.



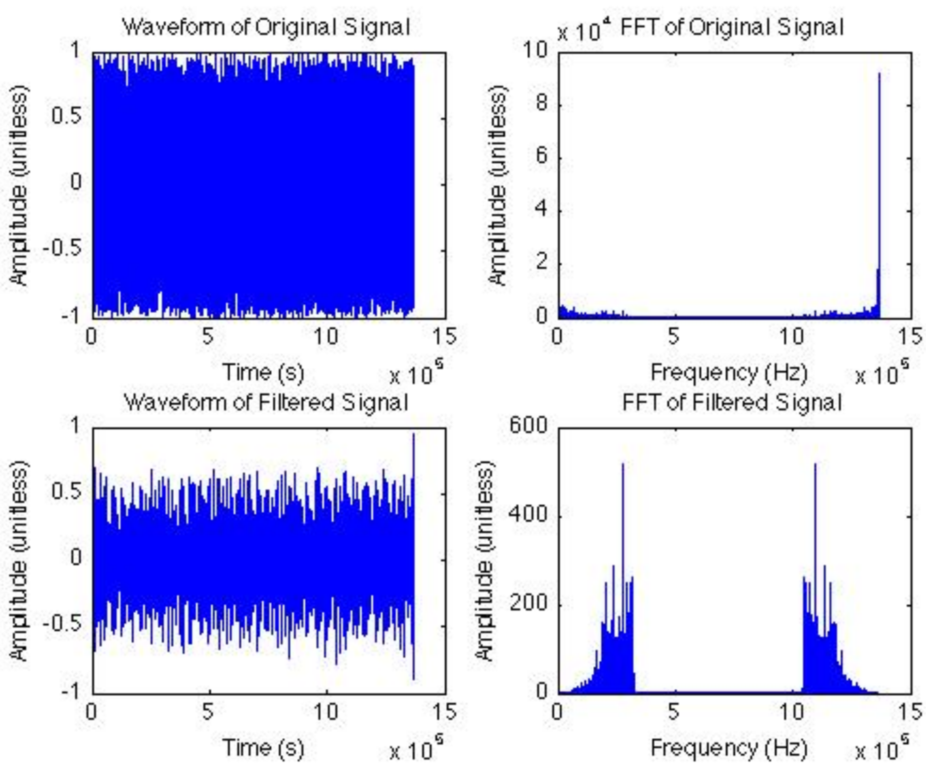
Comparison of the Magnitude Response of the signal with and without the undesired frequency.

Part 2: Filtering Using FFT

```
[y fs]=
wavread('/Users/muestudent/Desktop/Connor502Lab8/drivehard.wav');

subplot(2,2,1);
plot(y);
title('Waveform of Original Signal');
xlabel('Time (s)');
ylabel('Amplitude (unitless)');

fy = fft(y);
ymag = abs(fy);          %  Magnitude of FFT
yphase = angle(fy);      %  Phase of FFT

subplot(2,2,2);
plot(ymag);
title('FFT of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude (unitless)');

magl = length(ymag);         %Length of magnitude vector
w = (hanning(magl)).^1.5;    %Band pass filter, the exponent is the Q of
the filter
yfilt = ymag.*w;
```

```
subplot(2,2,4);
plot(yfilt);
title('FFT of Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude (unitless)');

R = yfilt;
theta = yphase;                 % put into form Y = R*e^(j)(theta)
e = 2.71828182846;              % Define e

Y = R.*e.^(1i*theta);              % Reconstruct complex spectrum

ynew = ifft(Y);                    % Transform signal back into time time
domain.
ynewreal = real(ynew)*10;          % 10 is the make up gain.

subplot(2,2,3);
plot(ynewreal);
title('Waveform of Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude (unitless)');

%soundsc(ynewreal,fs);
wavwrite(ynewreal,fs,'connorlab8filtersignal.wav');
```

This code implements a high pass filter using a Hanning Window, which cuts the lowest frequencies below Nyquist and the highest frequencies above Nyquist. The signal is converted into the frequency domain using the fft function, where the windowing takes place, and then converted back to the time domain using the ifft function.



Waveform and Magnitude plots of the unfiltered and filtered signals.