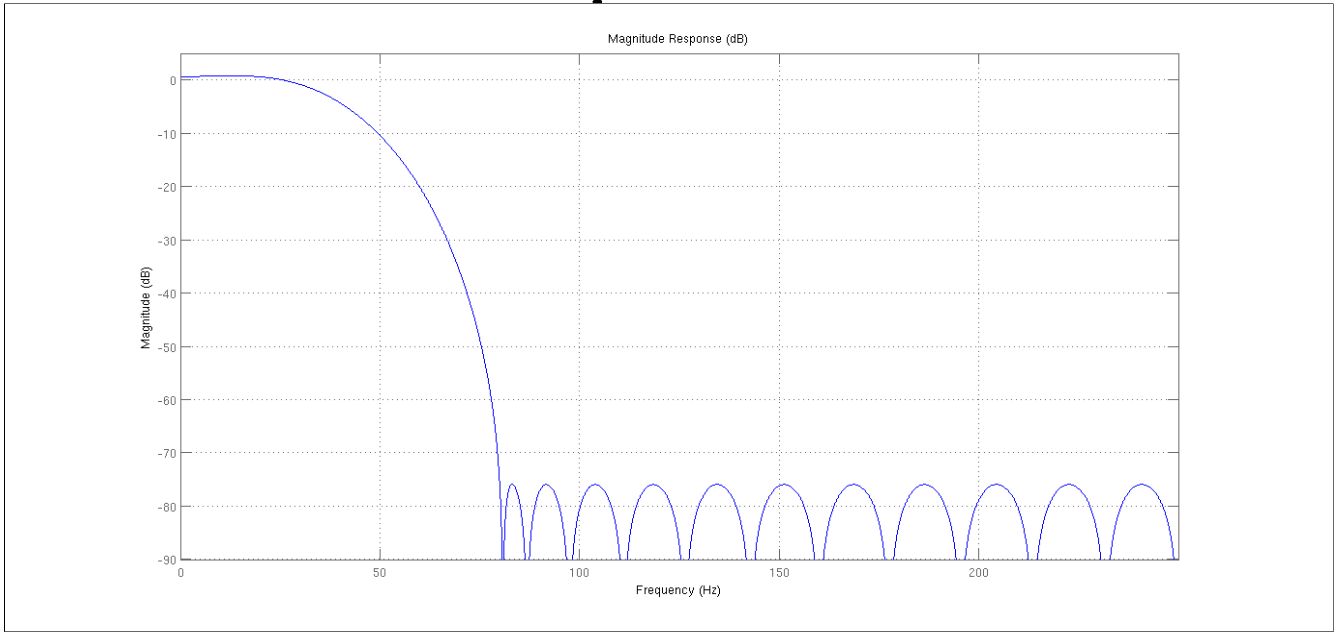
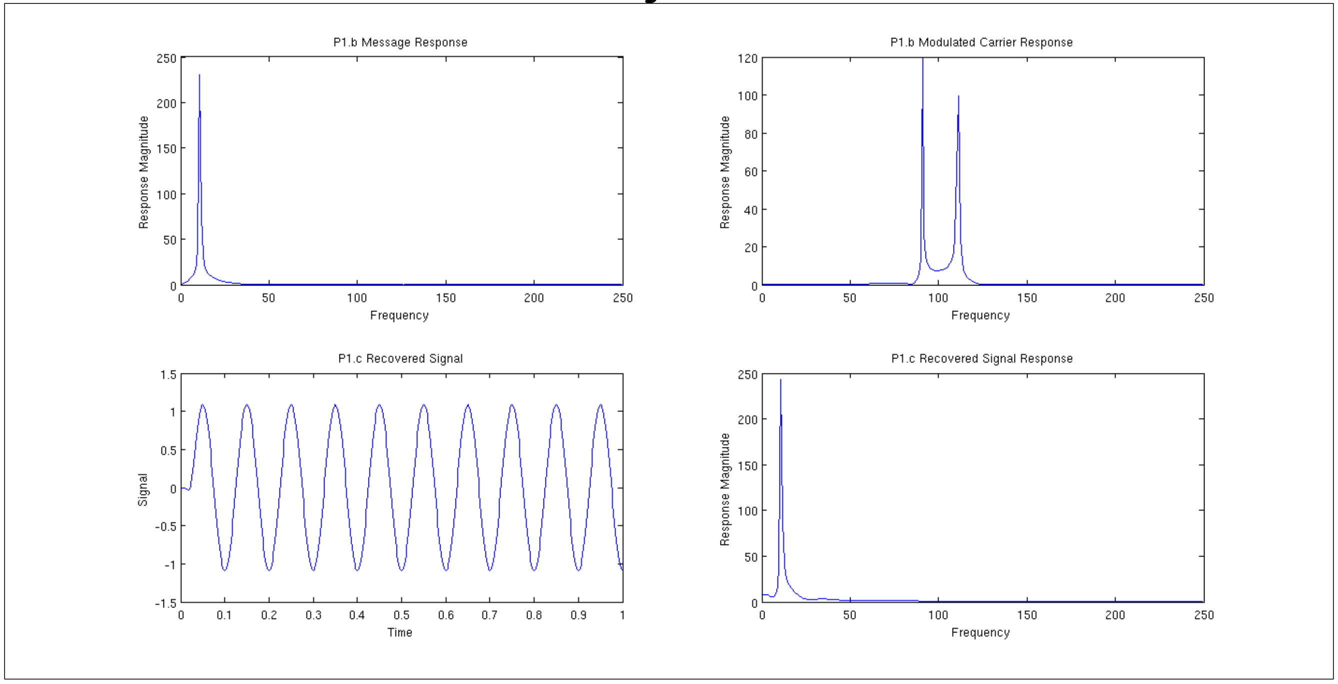
Lab 08
Christopher Bero
EE384

Problem 1

Lowpass Filter



Signals



```
% Lab 08 - Problem 1
% EE384
% Christopher Bero

% P1.a
% Well, AM as we commonly use it operates by taking a low frequency
% signal, such as voice or music, and migrating it to the FM radio
% band, which consumes the 80~110 MHz range. I think that this
% is very similar to superimposed signals like the DTMF.
% To recover the voice signal, our low pass filter will keep voice
% ranged frequencies and attenuate the radio frequencies.


% P1.b
sfs=500; % Sampling frequency
fs=10; % Message frequency
t=0:(1/sfs):1; % Domain
message=sin(2*pi*t*fs); % Message signal
cfs=100; % Carrier frequency
carrier=sin(2*pi*t*cfs); % Carrier signal
carrier_mod=ee384_ammod(message, carrier); % Generate the modulated
signal

[m_bins, message_resp]=freqSpec_1s(message, sfs);
[c_bins, mod_resp]=freqSpec_1s(carrier_mod, sfs);

fig2=figure();
subplot(2,2,1);
plot(m_bins,abs(message_resp));
title('P1.b Message Response');
xlabel('Frequency');
ylabel('Response Magnitude');

subplot(2,2,2);
plot(c_bins,abs(mod_resp));
title('P1.b Modulated Carrier Response');
xlabel('Frequency');
ylabel('Response Magnitude');

% P1.c
recovered=ee384_amdemod(carrier_mod, carrier, sfs, 30, 80);
[r_bins, rec_resp]=freqSpec_1s(recovered, sfs);

subplot(2,2,3);
plot(t,recovered);
title('P1.c Recovered Signal');
xlabel('Time');
ylabel('Signal');

subplot(2,2,4);
plot(r_bins,abs(rec_resp));
title('P1.c Recovered Signal Response');
xlabel('Frequency');
ylabel('Response Magnitude');
% We can see from this frequency response that the original signal
% has been returned with very little distortion
```

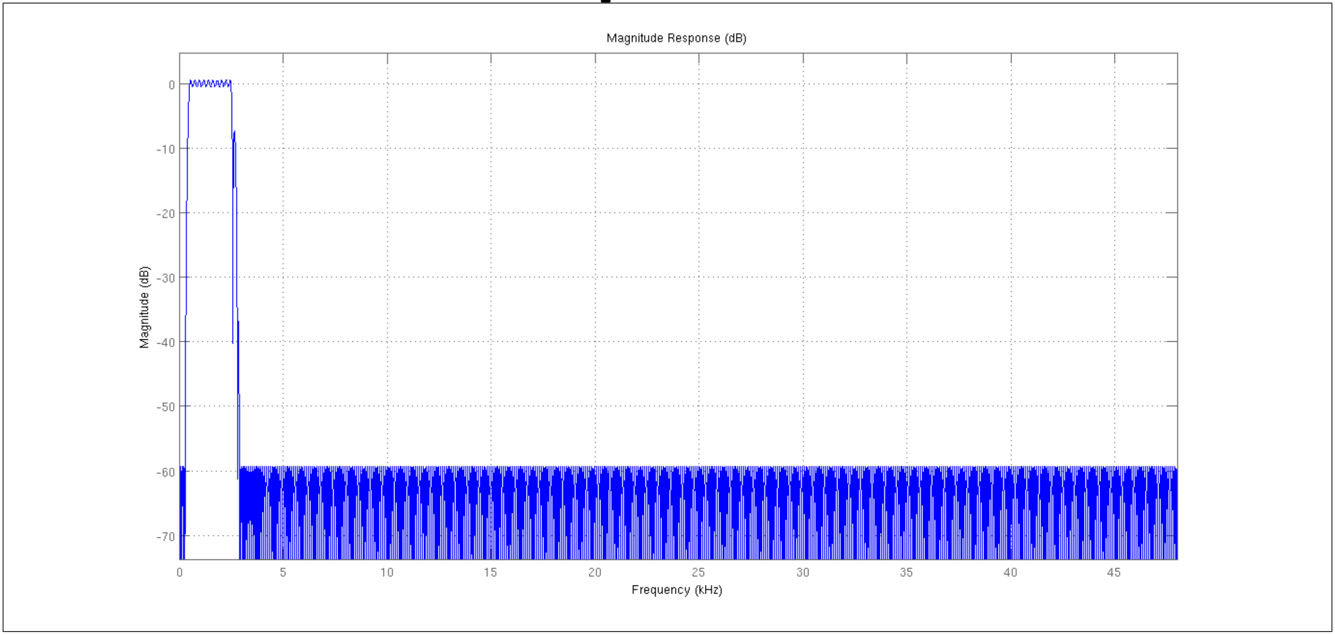
```
function [ mod_signal ] = ee384_ammod( message, carrier )
% ee384_ammod is a function to compute the AM signal
% from a message signal transcribed to a carrier signal.
% Part of lab 08

mod_signal = message .* carrier;

end
```

```
function [ message ] = ee384_amdemod( mod_signal, carrier, sfs,
pass, stop )
% ee384_amdemod to demodulate an AM signal.

% Equiripple Lowpass filter designed using the FIRPM function.
% All frequency values are in Hz.
Fs = sfs;  % Sampling Frequency
Fpass = pass;              % Passband Frequency
Fstop = stop;              % Stopband Frequency
Dpass = 0.057501127785;  % Passband Ripple
Dstop = 0.0001;          % Stopband Attenuation
dens  = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass,
Dstop]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

message = filter(Hd, (2*carrier.*mod_signal));

end
```

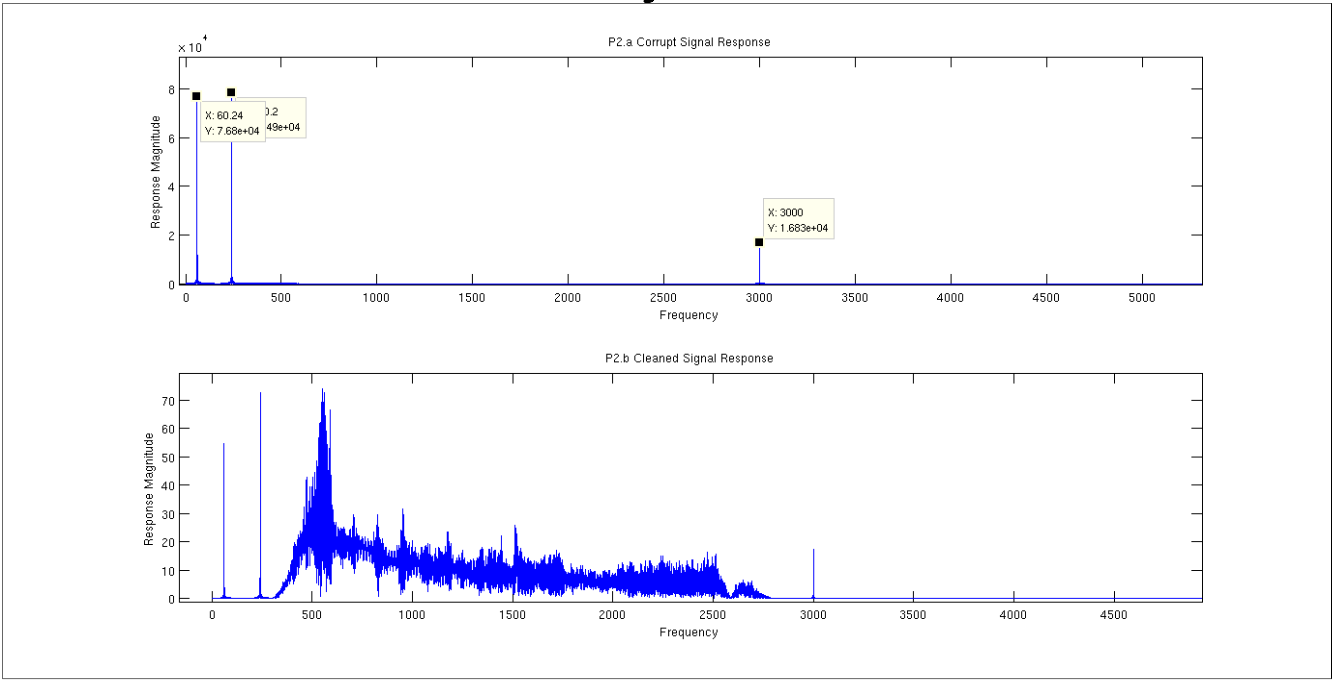
```
function [ f_vals, f_sig ] = freqSpec_1s( signal, sfs )
% freqSpec_1s: Generate one sided frequency spectrum for plotting
% Usage:    [x,y]=freqSpec_1s(time_signal,sample_freq);
%           plot(x,y);

t_len=length(signal); % number of samples
nfft=2^ceil(log2(t_len)); % number of FFT bins
%f_vals=sfs*(-nfft/2:nfft/2-1)/nfft; % Frequency range (w/ neg)
f_vals=sfs*(0:nfft/2-1)/nfft; % Frequency range (positive only)
y1_f=fftshift(fft(signal,nfft)); % Two sided, zero centered FFT
f_sig=y1_f(nfft/2:nfft-1); % Only positive frequencies

end
```

Problem 2

Bandpass Filter



Signals



```matlab
% Lab 08 - Problem 2
% EE384
% Christopher Bero

%P2.a
%uiopen('/home/berocs/Documents/uah/ee384/lab08/corrupted.wav',1);
% data and fs should now be available
[bins, data_resp]=freqSpec_1s(data, fs);

fig2=figure();
subplot(2,1,1);
plot(bins,abs(data_resp));
title('P2.a Corrupt Signal Response');
xlabel('Frequency');
ylabel('Response Magnitude');
% Bad frequencies around 3000Hz

%P2.b
bandpass=p2_filter();
clean_data=filter(bandpass, data);
[c_bins, clean_resp]=freqSpec_1s(clean_data, fs);

subplot(2,1,2);
plot(c_bins,abs(clean_resp));
title('P2.b Cleaned Signal Response');
xlabel('Frequency');
ylabel('Response Magnitude');

% Says: "Testing: one, two, three."
```

```
function Hd = p2_filter
%P2_FILTER Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 7.14 and the Signal Processing Toolbox
6.17.
%
% Generated on: 22-Jul-2015 02:28:46
%

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 96000;  % Sampling Frequency

Fstop1 = 300;              % First Stopband Frequency
Fpass1 = 500;              % First Passband Frequency
Fpass2 = 2500;             % Second Passband Frequency
Fstop2 = 2900;             % Second Stopband Frequency
Dstop1 = 0.001;            % First Stopband Attenuation
Dpass  = 0.057501127785;   % Passband Ripple
Dstop2 = 0.001;            % Second Stopband Attenuation
dens   = 20;               % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1
...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```