EE-556
Project -2
Cleaning up an Audio File

Due: 12-17-2015 by 11:55PM

Reading and writing audio files in MATLAB

One can use MATLAB as a very good environment to work on audio signals. MATLAB has powerful capabilities to read and write audio wave files (.wav) and play them back. An audio signal in MATLAB is a column or row vector containing real numbers between –1 and +1.

The following command can be use to read a "wav" file:

```
[x,fs]=audioread('file name')
```

In the above command '`x`' is a column vector, which will contain the audio signal in the MATLAB environment and '`fs`' is the sampling frequency of the signal. The sampling frequency of each signal is stored in the wav file and it should be remained unchanged during the processing of the audio file.

The following command can be used to write an audio signal to a "wav" file:

```
audiowrite('file name',x,fs)
```

The vector '`x`' is a row or column vector containing the audio signal and '`fs`' is its sampling frequency. In playback, you should use the same sampling frequency, which used to sample the signal. Changing the sampling frequency from its correct value will cause a faster or slower sound (I recommend to try it, it is funny!).

To playback an audio signal in MATLAB you should use the following commands:

```
P=audioplayer(x,fs)
Play(P)
```

Note: Since the valid range of the numbers in a voice file is between –1 and +1, the "wavwrite" command will clip the signal exceeding this range, from both negative and positive sides, to keep the signal between –1 and +1. This will cause a significant distortion in the audio signal and has to be prevented. In order to prevent this from happening you should create (manipulate) the audio signal in such a way that it will not exceed –1 and +1.

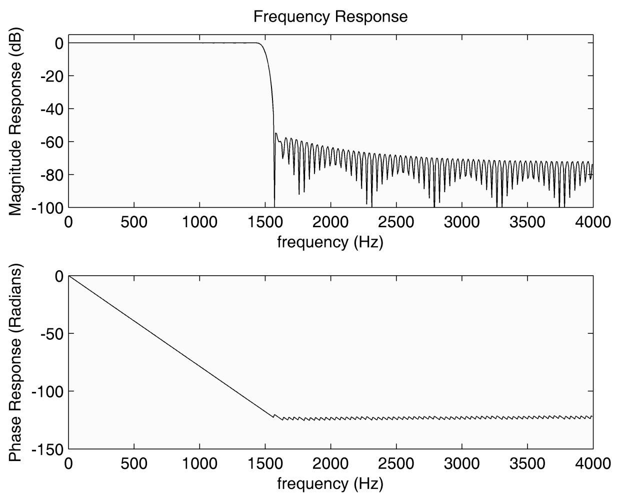
Designing linear-phase FIR filter in MATLAB

To design a linear-phase FIR filter we should find its impulse response, which is a finite discrete-time sequence. The "`fir1`" command in MATLAB can design an FIR filter. The format of the "`fir1`" command is as follows:

1. Low-Pass linear-phase FIR filter:
   `h=fir1(N,wn,'low')`
   Where "`h`" is the impulse response of the filter and "`wn`" is the normalized cut-off frequency of the low-pass filter and "`N`" is the order of the filter. If the actual cut-off frequency of the filter is "`w0`", then "`wn=2*wo/fs`", where "`fs`" is the sampling frequency. For example, if the sampling frequency is 10000Hz and the cut-off frequency is 1000Hz, then `wn=2000/10000=0.2`.
2. High-Pass linear-phase FIR filter:
   `h=fir1(N,wn,'high')`
   Where "`h`" is the impulse response of the filter and "`wn`" is the normalized cut-off frequency of the high-pass filter and "`N`" is the order of the filter. The normalized cutt-off frequency "`w0`" is found the same as of the low-pass filter.
3. Band-Pass linear-phase FIR filter:
   `h=fir1(N,wn,'bandpass')`
   Where "`h`" is the impulse response of the filter and "`wn`" is the normalized cut-off frequency vector containing the lower and higher cutt-off frequencies "`wn=[w1 w2]`" of the band-pass filter and "`N`" is the order of the filter. The normalized cut-off frequencies "`w1`" and "`w2`" are found the same as of the low-pass filter.
4. Band-Stop linear-phase FIR filter:
   `h=fir1(N,wn,'stop')`
   Where "`h`" is the impulse response of the filter and "`wn`" is the normalized cut-off frequency vector containing the lower and higher cut-off frequencies "`wn=[w1 w2]`" of the stop-band filter and "`N`" is the order of the filter. The normalized cut-off frequencies "`w1`" and "`w2`" are found the same as of the low-pass filter.

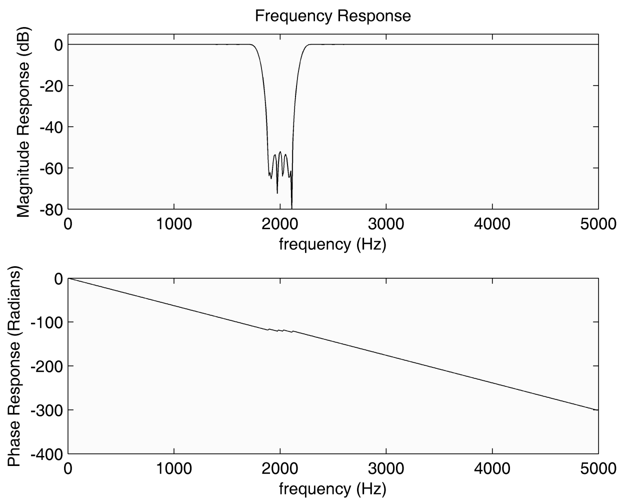
Example 1: Find the impulse response of a 20th order linear-phase low-pass FIR filter with cut-off frequency of 1500Hz. The sampling frequency is 8000Hz. Plot the frequency response of the filter (with respect to the actual frequency).

```
clear all;
close all;
clc;
fs=8000;
wc=1500;
wn=wc/(fs/2);
N=200;
h=fir1(N,wn,'low');
[H,w]=freqz(h,1);
subplot(2,1,1);plot(w*fs/(2*pi),20*log10(abs(H)));
title('Frequency Response');xlabel('frequency (Hz)');ylabel('Magnitude Response
(dB)');axis([0,fs/2,-80,5])
subplot(2,1,2);plot(w*fs/(2*pi),unwrap(angle(H)));
xlabel('frequency (Hz)');ylabel('Phase Response (Radians)')
```



Example 1: Find the impulse response of a 200th order linear-phase band-stop FIR filter with cut-off frequencies of 1800Hz and 2200Hz. The sampling frequency is 10000Hz. Plot the frequency response of the filter (with respect to the actual frequency).

```
clear all;
close all;
clc;
fs=10000;
wc1=1800;
wc2=2200;
wn=[wc1 wc2]/(fs/2);
N=200;
h=fir1(N,wn,'stop');
[H,w]=freqz(h,1);
subplot(2,1,1);plot(w*fs/(2*pi),20*log10(abs(H)));
title('Frequency Response');xlabel('frequency (Hz)');ylabel('Magnitude Response
(dB)')
axis([0,fs/2,-80,5])
subplot(2,1,2);plot(w*fs/(2*pi),unwrap(angle(H)));
xlabel('frequency (Hz)');ylabel('Phase Response (Radians)')
```



PROJECT

The provided audio file is corrupted by three signals, a colored noise and two annoying sinusoidal signals. The colored noise is a band-limited random signal, which can be generated by passing a white noise through a band pass filter. The frequency of one of the annoying sinusoidal signals is less than 200Hz and the frequency of the other one is above 1000Hz. The frequency range of the colored noise is less than 5500Hz. At some point in the audio file, the frequency of one of the annoying sinusoids changes. You need to find the time stamp of this event and consider it in the clean-up process.

The goal of this project is to design as many FIR filters as needed to clean up the entire audio file. Obviously, it is impossible to completely clean up the audio file because of the existence of random signal but the file can be cleaned to obtain an acceptable quality.

In this project, you should write a MATLAB program to clean up the audio file. First, you should analyze the spectrum of the signal by using "`freqz`" or "`fft`" function to find out the annoying signals' frequencies and bandwidths. Then you should design different filters to clean up the audio signal. You have to explain the result of your analysis; also, you need to explain how you determined the cut-off frequencies of the filter. You should plot the spectrums of the signal before and after filtering and compare them. Moreover, you need to plot the transfer functions of the filters.

You may work in a group of **maximum two students**. You should submit **ONE** report per group. You are not allowed to seek help from others except your classmates or me.

Deliverables:

1. A report explaining your analysis of the noisy signal and the method you chose to clean it up. You should exactly state the parameters of the filters that you used. The report should not be more than 5 pages (including graphs but excluding appendixes). The report must be in **MS Word** format and submitted to the Turnitin assignment (Project-2 (Report) created on the course website in the Blackboard system in the Assignment folder. The MATLAB code should be included in the report **as an appendix**.
2. The **cleaned-up audio** file in the **"wav" format** that should be submitted to my email as an attachment.

Please don't ZIP the files together. **You should submit two different files to different locations**. Please make the title of the files according to these formats:

The wave file name:	**<your name1_your name2>.wav**
The report file name:	**< your name1_your name2>.doc (or .docx)**

The reports that do not comply with the above guidelines **will not be graded**.