Signal Processing Exam #2

2017 - 2018 Fall

Introduction

IEEE Signal Processing Society organizes a competition among universities all around the world. This year, the goal of the competition was to track the beat of a song in real-time.

The winner is a automated drum player, designed and developed by a team of engineering students at University of New South Wales; Jeremy Bell, Max Fisher, Angus Keatinge, James Wagner. Their system is a simple mechanical system which plays the drums while the team members are playing guitars and singing. In order to achieve this, the system has to listen the guitars and estimate the tempo in real-time. You can watch their videos at the links

https://www.youtube.com/watch?v=VkoGZnVEsfw

https://www.youtube.com/watch?v=KBlygm377gw

In midterm 2, you are supposed to implement a simple tempo estimation algorithm. The details are given in the problems.

You will submit your report in pdf format and resulting Matlab codes until 23:59 at Sunday (17.12.2017). What your report should include is presented at the end of each problem.

Figure 1: The winner team of the Signal Processing Cup 2017.

Problem 1 (20 pts)

Sampling frequency is 44100 Hz. Design a digital IIR Butterworth filter

- Pass-band edge frequency is 200 Hz. The ripple at pass-band should be -1 dB.
- Stop-band edge frequency is 400 Hz. The minimum attenuation at stop-band is -40 dB.

Manipulate the Matlab Code at the bottom of the page to design your filter. Plot the resulting magnitude response in two subplots. x axes are frequency in terms of Hertz, y axes are magnitudes in terms of decibels.

- For the first subplot, x axis range is $[0 - 22050]$ Hz.
- For the second subplot, x axis range is $[0 - 600]$ Hz (Hint: *xlim* command).

```
%% Initialize parameters - DON'T EDIT
Fs = 44100; Ts = 1/Fs;

%% Place your code here... - EDIT HERE
%  Design a digital chebyshev IIR filter with
%  pass-band 200 Hz and ripple is 1 dB
%  stop-band 400 Hz and minimum attenuation is 40 dB
%  Set bL and aL as coefficients of your filter

% ...
% ...

% Plot the frequency response in 2 subplots
% Subplot 1 - Plot the frequency between [0 - Fs/2] Hz.
% Subplot 2 - Plot the frequency between [0 - 600] Hz.
% x axes are in terms of Hz.
% y axes are in terms of dBs.

% ...
% ...

% End of your code  - END OF EDIT
%% Print the coefficients and save the results - DON'T EDIT
clc
for k = 1:N+1
    fprintf('a(%d) = %8.6g, \t b(%d) = %8.6g \n', k-1, aL(k),k-1, bL(k));
end

% Clear needless variables and save the filter
clearvars -except bL aL
save low_pass_coefficients;
```

At your report, present

- The final Matlab code,
- The filter coefficients a_i and b_is,
- The transfer function $H(z)$,
- The magnitude response plots (two subplots),
- Pole-zero map of the filter (Hint: *pzmap* command).

Problem 2 (20 pts)

Use the filter designed in Problem 1 to filter the signal in Figure 2. You can find the signal in *question2_signal.mat* file. Use the filtfilt command to avoid delay. Plot the input and output signals.

Fast Fourier Transform (FFT) algorithms are used to approximate discrete-time Fourier transform of the signals. Plot the input and output FFT plots. One can refer to the first example in

https://www.mathworks.com/help/matlab/ref/fft.html

to learn how to plot FFTs in Matlab. Manipulate the following code.

```
%% Initialize parameters - DON'T EDIT
load question2_signal;
load low_pass_coefficients;

%
%% Place your code here - EDIT HERE
% Filter the signal x
% The output signal names should be y

% ....
% ....


% Plot the input and output in two subplots
figure;
subplot(2,1,1)
plot(t, x);
xlabel('Seconds'); ylabel('x');
subplot(2,1,2)
% ....
% ....

% Plot the input and output Fourier plots (FFT plots) in two subplots

% ...
% ...

% End of your code - END OF EDIT
```

At your report, present

- The final Matlab code,
- The input and output signals in two subplots,
- The input and output FFT plots in two subplots.

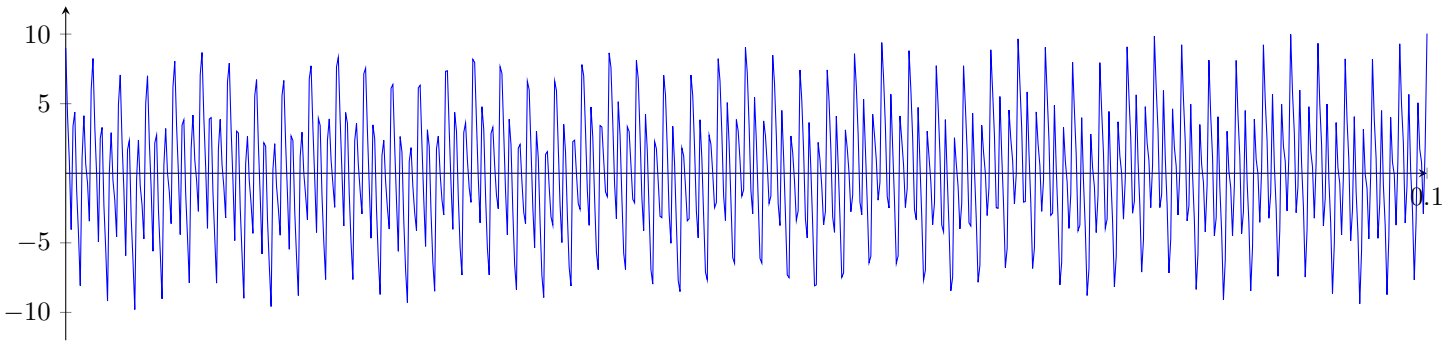

Figure 2: Input signal

Problem 3 (20 pts)

Design a order $N = 7$, FIR differentiator using Parks-McClellan algorithm (*firpm* command). The ideal differentiator is

$$H(j\Omega) = j\Omega$$
$$|H(j\Omega)| = \Omega.$$

Note that $\omega = \Omega T_s$. The sampling frequency is $F_s = 44100/40 = 1102.5$ Hz. Then, the digital differentiator filter specifications are

$$|H(e^{j\omega})| = \omega F_s, \qquad |H(e^{j0})| = 0, \qquad |H(e^{j\pi})| = \pi F_s,$$

Plot the magnitude response.

```
%% Initialize parameters
W = 40;
Fs = 44100/W;
Ts = 1/Fs;
N = 7;

%% Place your code here - EDIT HERE
%  Design a FIR differentiator with Parks-Mccellan
%  Set bD as your filter coefficients

% ...
% ...

% Plot the frequency response of your filter
% y axis is magnitude (not in dB)
% x axis is frequency in Hz. Plot for the range [0 - Fs/2]
% Place your plot code here,

% ...
% ...

% The code below plots the ideal differentiator for comparison purposes
hold on;
plot([0 Fs/2], [0 pi*Fs], '--r');
hold off;
legend('Filter', 'Ideal Differentiator', 'Location', 'NorthWest')

% End of your code - END OF EDIT


%% Plot the results - DON'T EDIT


% Print the coefficients
clc
for k = 1:N+1
    fprintf('b(%d) = %8.6g, \n', k-1, bD(k));
end

% Clear needless variables and save the filter
clearvars -except bD
save differentiator_coefficients;
```

At your report, present

- The final Matlab code,
- The filter coefficients b_is,
- The magnitude response.

Problem 4 (40 pts)

In this problem, we will implement a tempo estimation algorithm. The example song is one of my nephew's and niece's (Elif Hayvacı and Ahmet Kerem Hayvacı) favourite song which is Barbie Girl from Aqua.

In music theory, the tempo is measured by beats per minute (BPM). Hence, the time interval between two beats is

$$\Delta t = \frac{60}{\text{BPM}} \Rightarrow \Delta N = \text{round}\left(\frac{60}{\text{BPM} \cdot T_s}\right).$$

Most widely used BPMs are in the range 60 - 200 in songs. Hence, potential ΔN values are between 1103 - 331 sample interval for the sampling frequency 1102.5 Hz.

We will estimate the tempo of the song from the drum kicks. Drum kicks have lots of energy at low frequencies, e.g., 0-200 Hz frequency band. Hence, we will seperate the drum kicks by using the low pass filter in Problem 1. We will calculate the power of the signal with respect to time. When the drum kicks, the power increases drastically. As a result, there are positive derivative of the energy signal at beats. Use the FIR differentiator in Problem 3 to differentiate the power signal.

Step by step:

- Load the *barbie_girl.mat* file. Sampling frequency is $F_s = 44100$ Hz.
- Listen the song by using *sound(x, Fs)* command of Matlab. You can stop the song *clear sound*. Not add sound command to the script. Just write it in Command Window to listen.
- Filter the song x with LPF in Problem 1. Use *filtfilt* command. Name the output as xL. Not add sound command to your script.
- Listen the filtered song xL. Note that, you can here only the drum kicks.
- Calculate the energy of sub-arrays of xL with length of 40. The energy function is

  $$E[n] = \sum_{k=40n}^{40n+39} x_L^2[k]$$

  Note that the sampling frequency of the E is $F_{s2} = 44100/40 = 1102.5$ Hz.
- Calculate the derivative dE/dt by using the FIR filter in Problem 3. Use the *filter* command, not *filtfilt*.
- Half wave rectify the dE/dt. Let us call the half-wave rectified signal as Edh, then

  $$E_{dh} = \begin{cases} \frac{dE}{dt}, & \frac{dE}{dt} > 0, \\ 0, & \frac{dE}{dt} \leq 0, \end{cases}$$

- Plot the Edh with respect to time.
- Edh contains the beats of the song. In order to estimate the BPM of the song, we use the auto-correlation function of Edh. The auto-correlation function is defined as

  $$R(\Delta N) = \sum_{k=0}^{L_E - \Delta N} E_{dh}[k] E_{dh}[k + \Delta N].$$

  where L_E is the length of the energy signal. You can calculate the auto-correlation function by your own code or using *autocorr* function of Matlab.

  The auto-correlation function takes the highest values for 0, the song's BPM and BPM/2. Plot the auto-correlation function with respect to different ΔN values between 331 - 1103. Determine the song's ΔN for which the auto-correlation function is maximum. Convert it to BPM.
- The BPM is the time interval between two beats. Usually $t = 0$ is not the time of the first beat. We should also determine the time of the first beat, let us denote by t_1 and N_1 where $t_1 = N_1 \cdot T_s$. First, generate a impulse train

  $$s[n] = \sum_{k=-\infty}^{\infty} \delta[n - k \cdot \Delta N], \qquad 0 \leq n \leq L_E$$

Let us define the cross-correlation function as

$$C(N) = \sum_{k=0}^{L_x - N} E_{dh}[k] \cdot s[k+N].$$

You can calculate the cross-correlation function by your own code or using *crosscorr* function of Matlab. Cross-correlation function gets high values when beat times and shifted $s[n+N]$ perfectly matches. Plot the cross-correlation function at the interval $N \in [-\Delta N, 0]$ Take the first negative N as N_1, e.g., $N_1 = -N$ for which the cross-correlation function is maximum. Calculate t_1 from N_1.

- Display the calculated BPM and t_1 at the command window by using *fprintf* command.

At your report, present

- The final Matlab code,
- Edh plot,
- auto-correlation function plot in interval 331 - 1103,
- cross-correlation function plot in interval -1500 - 0,
- calculated BPM and first beat time t_1.