Anıl ÖZTÜRK

14067019



Summary

Creating of a real-time beat tracker using IIR and FIR filters and its application on a sample song.

ApplIcatIon of Real-TIme Beat TrackIng

Signal Processing Midterm-2

DESIGN OF AN IIR BUTTERWORTH FILTER

Our Butterworth filter has a 44100 sampling frequency. It has following properties;

* 200 Hz Pass-band edge frequency
* 400 Hz Stop-band edge frequency
* -1 dB ripple at pass-band
* -40 dB minimum attenuation at stop-band

The code of the filter will be as follows;

%% Initialize parameters - DON'T EDIT

Fs = 44100; Ts = 1/Fs;

%% Place your code here

wp = 2\*200\*Ts; % Pass-band edge frequency

ws = 2\*400\*Ts; % Stop-band edge frequency

rp = 1;  % Maximum pass band ripple 1dB

rs = 40;   % Minimum stop band attenuation 40 dB

%  Set bL and aL as coefficients of your filter

[N, wc] = buttord(wp, ws, rp, rs); % Getting the cutoff frequency and order of the filter values

[bL,aL] = butter(N, wc); % Getting the filter coefficients

% 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 (Zoomed version).

% x axes are in terms of Hz.

% y axes are in terms of dBs.

figure(1)

subplot(2,1,1)

[h,w]=freqz(bL,aL); % Getting the frequency and magnitude values from frequency analysis

plot(w/(2\*pi\*Ts),mag2db(abs(h))); % Plotting the magnitude response

xlim([0,22050]); % Plotting for frequencies between 0-22050

subplot(2,1,2)

[h,w]=freqz(bL,aL); % Getting the frequency and magnitude values from frequency analysis

plot(w/(2\*pi\*Ts),mag2db(abs(h))); % Plotting the magnitude response

xlim([0,600]); % Plotting for frequencies between 0-600

% Find the transfer function of the filter.

% Plot the pole-zero map of the filter.

Ts = 1; % Sampling period

Hb = tf(bL, aL, Ts); % Creating the transfer function with coefficient values and the sampling constant

figure (2)

pzmap(Hb); % Getting the pole-zero map of the filter

% End of your code

%% Print the coefficients and save the results - DON'T EDIT

clc

for k = 1:N+1

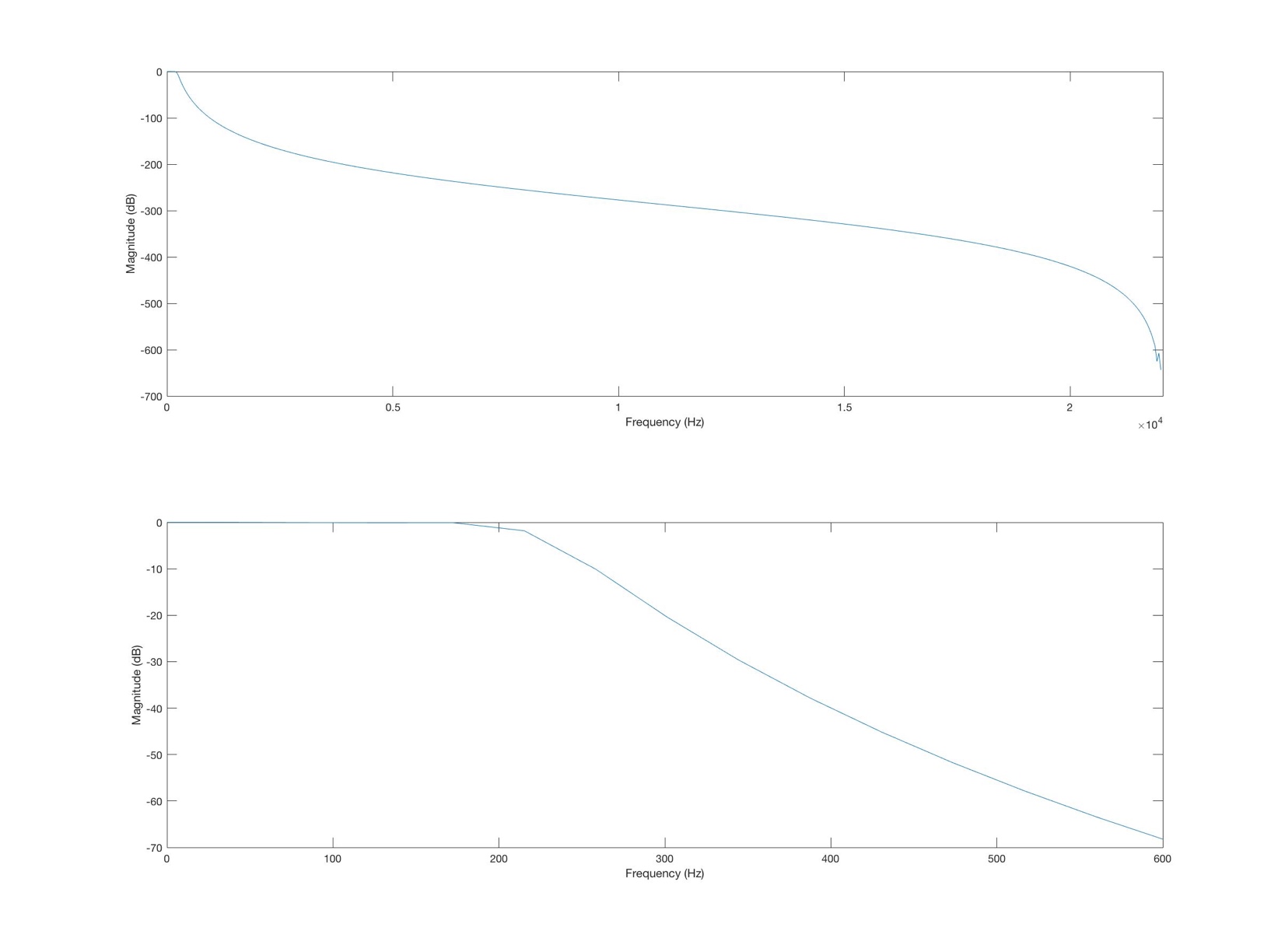
    fprintf('a(%d) = %8.6g, \t b(%d) = %8.6g \n', k-1, aL(k),k-1, bL(k));

end

% Clear needless variables and save the filter

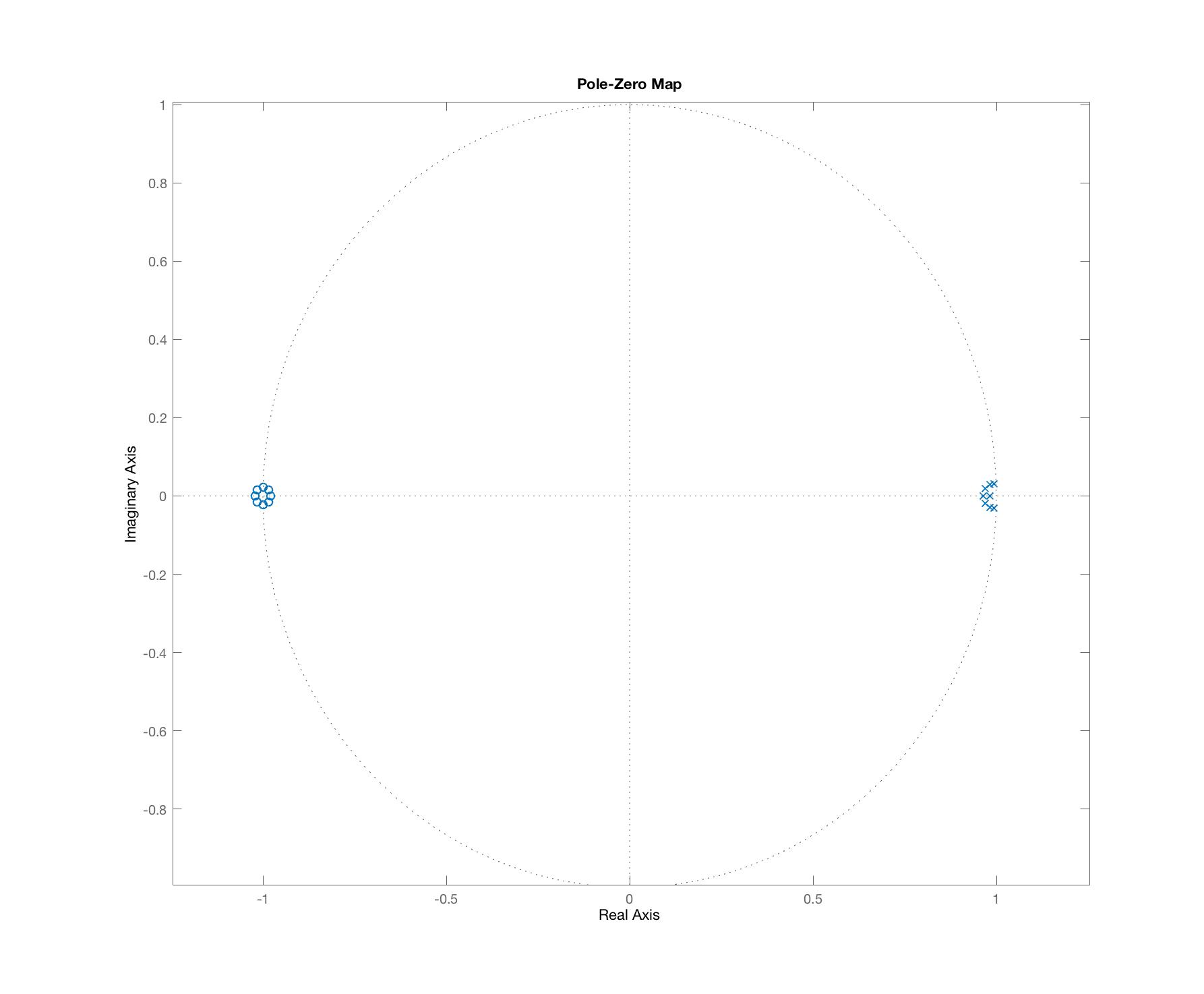
clearvars -except bL aL

save low\_pass\_coefficients;



The magnitude response of the filter is like above.

The filter coefficients are 9 element arrays, their values are;

The filter’s pole-zero map is like below;

We can get the transfer function of the filter with z-domain coefficients like below;

APPLICATION OF THE IIR FILTER

In this task, we are going to apply our IIR Butterworth filter to a noisy signal. The code of the application is below;

%% Initialize parameters - DON'T EDIT

load question2\_signal;

load low\_pass\_coefficients;

Fs = 44100; Ts = 1/Fs;

%

%% Place your code here... - EDIT HERE

wp = 2\*200\*Ts; % Pass-band edge frequency

ws = 2\*400\*Ts; % Stop-band edge frequency

[h,w]=freqz(bL,aL); % The magnitude and phase frequency values from the frequency analysis.

% Plot the input and output in two subplots

figure(1)

subplot(2,1,1)

plot(t, x); % The graph of the value of the input signal.

xlabel('Seconds'); ylabel('Input');

subplot(2,1,2)

y=filtfilt(Bl,aL,x); % Filtering the input signal with Butterworth digital filter.

plot(t,y); % The graph of the value of the filtered signal (output).

xlabel('Seconds'); ylabel('Output');

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

expanded\_w=w(1):(w(512)-w(1))/4410:w(512); % Expanding of frequency array to the same length input signal's matrix has.

maxfrequency=expanded\_w/(2\*pi\*Ts); % Setting the new frequency value of recently expanded frequency matrix's length.

figure(2)

input\_FFT = fft(x); % The DTFT of the input signal.

subplot(2,1,1)

plot(maxfrequency, input\_FFT); % The graph of DTFT of the input signal respect to the frequency

xlabel('Frequency'); ylabel('Input DTFT');

output\_FFT = fft(y); % The DTFT of the output signal.

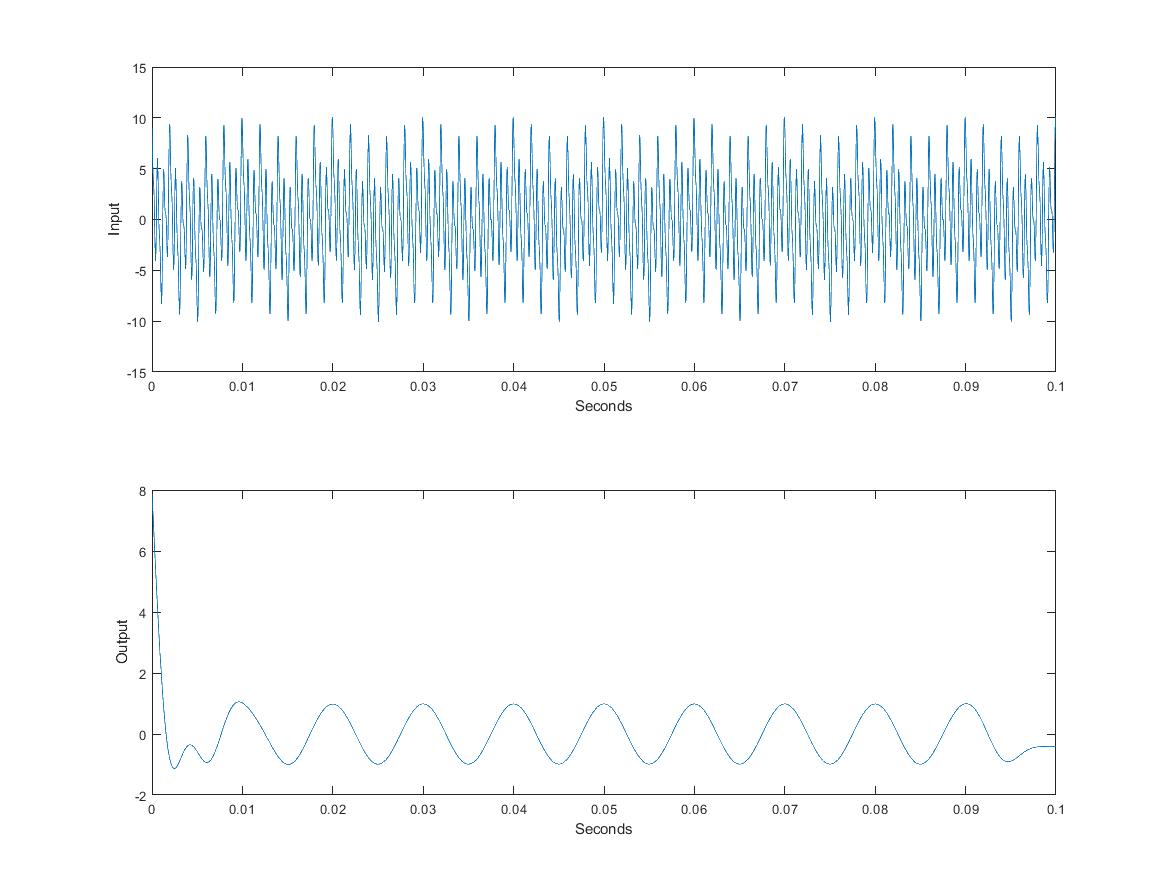
subplot(2,1,2)

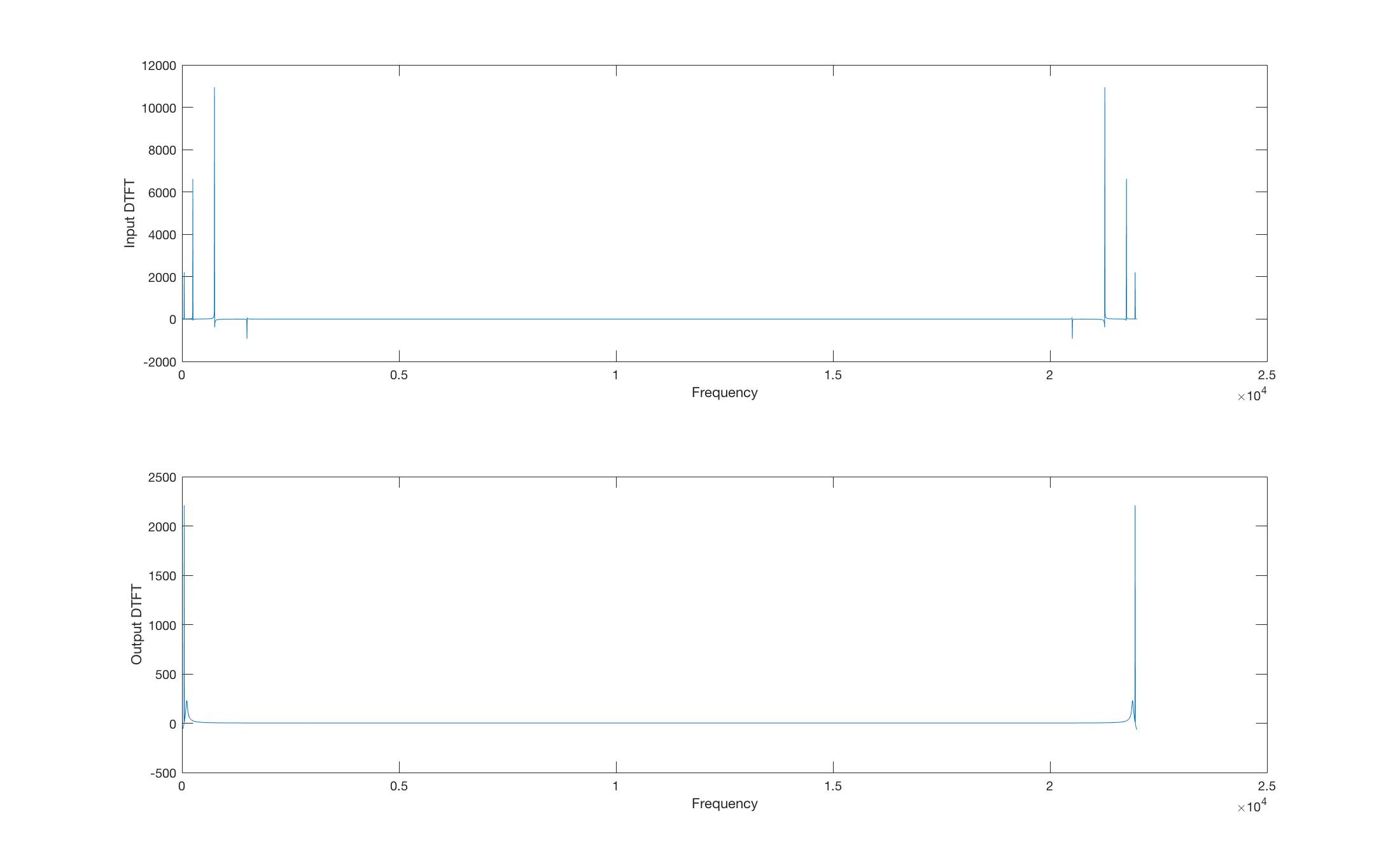
plot(maxfrequency, output\_FFT); % The graph of DTFT of the output signal respect to the frequency

xlabel('Frequency'); ylabel('Output DTFT');

% End of your code - END OF EDIT

The raw (input) and filtered (output) signals are as below;



The DTFT of raw (input) and filtered (output) signals are as below;

DESIGN OF A FIR FILTER

In this task, we are going to create a FIR filter using Parks-McClellan algorithm. The filter will be seventh order. The ideal differentiator is;

We know that . The sampling frequency is . Then, the digital differentiator filter specifications are;

,

The code of this filter is like below;

%% Initialize parameters

W = 40;

Fs = 44100/W;

Ts = 1/Fs;

N = 7; % The order of the filter

f = [0 1];       % Frequency band edges (as pi)

a = [0 pi\*Fs];   % Desired amplitudes

%% Place your code here...

%  Design a FIR differentiator with Parks-Mccellan

%  Set bD as your filter coefficients

bD = firpm(N,f,a,’d’); % The coefficients of the filter, d parameter has been added for optimizing the filter

% 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,

[H,W]=freqz(bD); % Getting the frequency and magnitude values from the frequency analysis

plot(W/(2\*pi\*Ts),abs(H)); % Drawing of the designed filter's response until the Fs/2

hold on;

plot([0 Fs/2], [0 pi\*Fs], '--r'); % Drawing of the ideal filter's response

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));

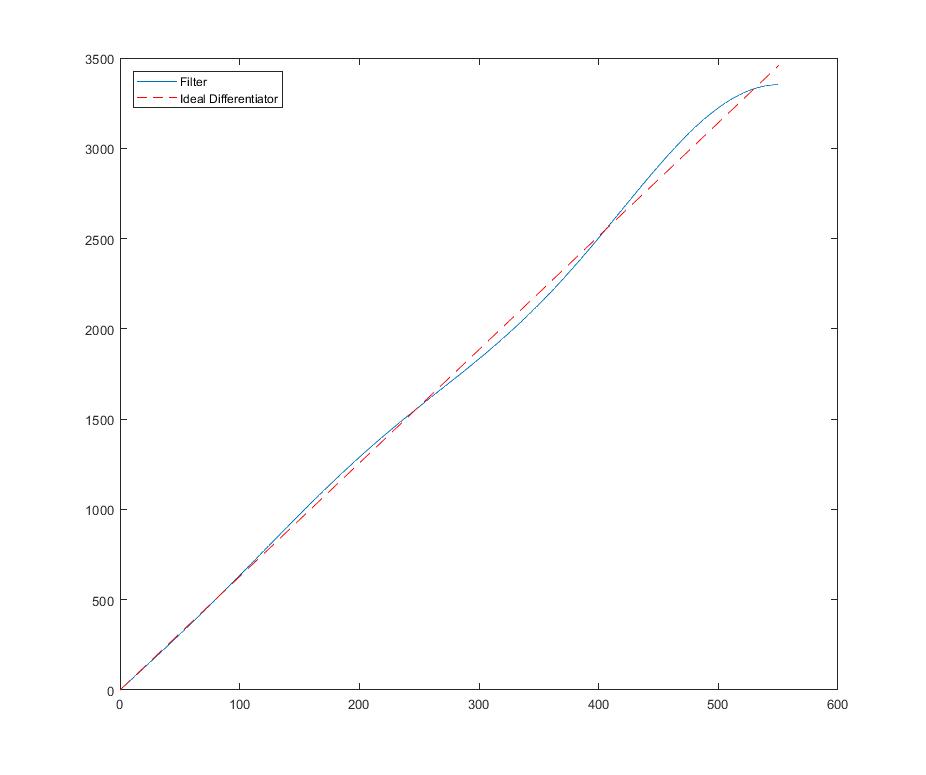
end

% Clear needless variables and save the filter

clearvars -except bD

save differentiator\_coefficients;

The filter coefficient is a 8 element array and its values are;

The magnitude response of designed and ideal filter is below;

APPLICATION OF BEAT-TRACKER

In this task, we are going to implement a tempo estimation algorithm. Following codes are going to do the job;

load barbie\_girl;

load low\_pass\_coefficients;

load differentiator\_coefficients;

Fs=44100;

Ts = 1/Fs;

xL=filtfilt(bL,aL,x); % Application of the IIR filter

L = length(xL);

W = 40;

% Calculating the energy of the signal

for k = 1:floor(L/W)

xp = xL( (k-1)\*W+1 : k\*W );

E(k) = sum(xp.^2);

end

Fs = 44100/W;

Ts = 1/Fs;

Ef=filter(bD,1,E); % Application of the FIR filter

Edh=Ef;

% Rectifying the signal

for k=1:length(Edh)

if Edh(k)<=0

Edh(k)=0;

end

end

expanded\_t=t(1):(t(length(t))-t(1))/(length(Edh)-1):t(length(t)); % Arranging the time matrix for plotting

figure(1)

plot(expanded\_t,Edh); % Plotting the rectified signal

AC=autocorr(Edh,1103,331,3); % Auto correlating the rectified signal

ACwoO=AC;

ACwoO(1)=0; % Neglecting the first value of the matrix, because it is the max value(1)

[value,index] = max(ACwoO); % Getting the maximum value's indice

bpm=round(60/((index-1)\*Ts)); % Calculating the BPM

figure(2)

autocorr(Edh,1103,331); % Drawing of the auto correlation

xlim([331 length(AC)]); % Limiting the x-axis in the drawing

trainmat=zeros(size(Edh));

% Identifying the impulse train matrix

for i=1:length(Edh)

if mod(i,index-1)==0

trainmat(i)=1;

end

end

CC=crosscorr(Edh,trainmat,1500); % Cros correlation of the rectified signal

figure(3)

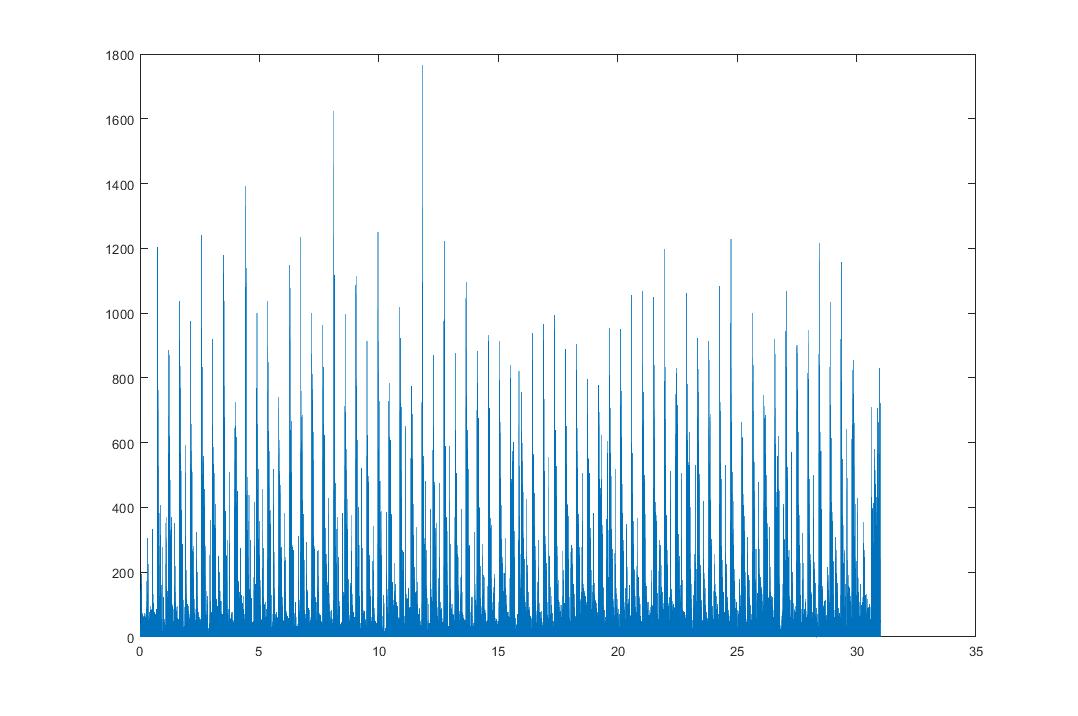
crosscorr(Edh,trainmat,1500); % Drawing of the cross correlation

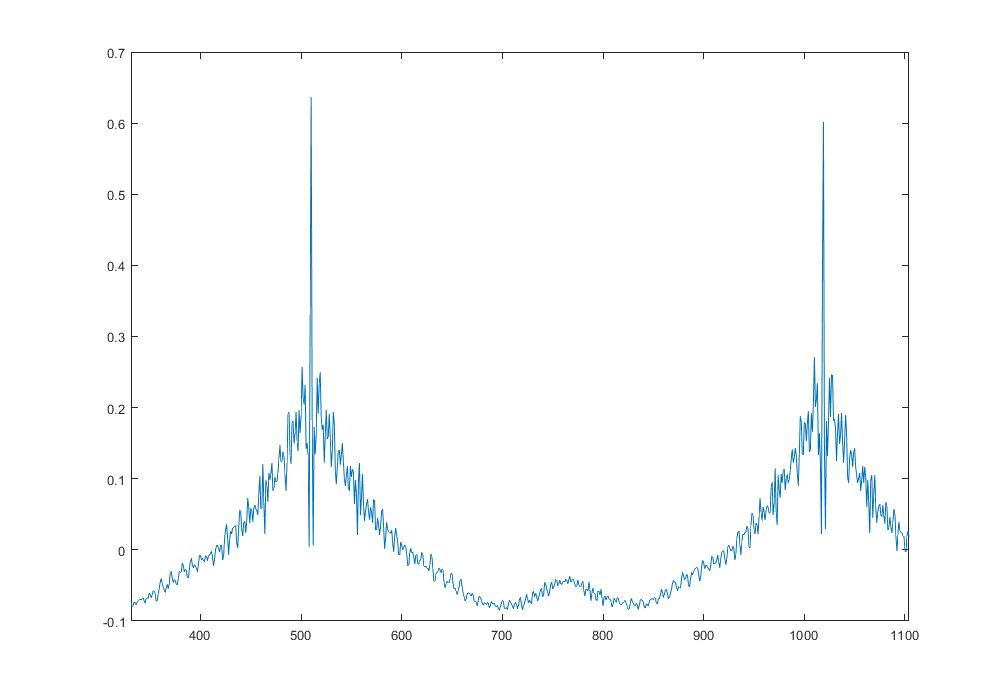
xlim([-1500 0]); % Limiting the x-axis in the drawing

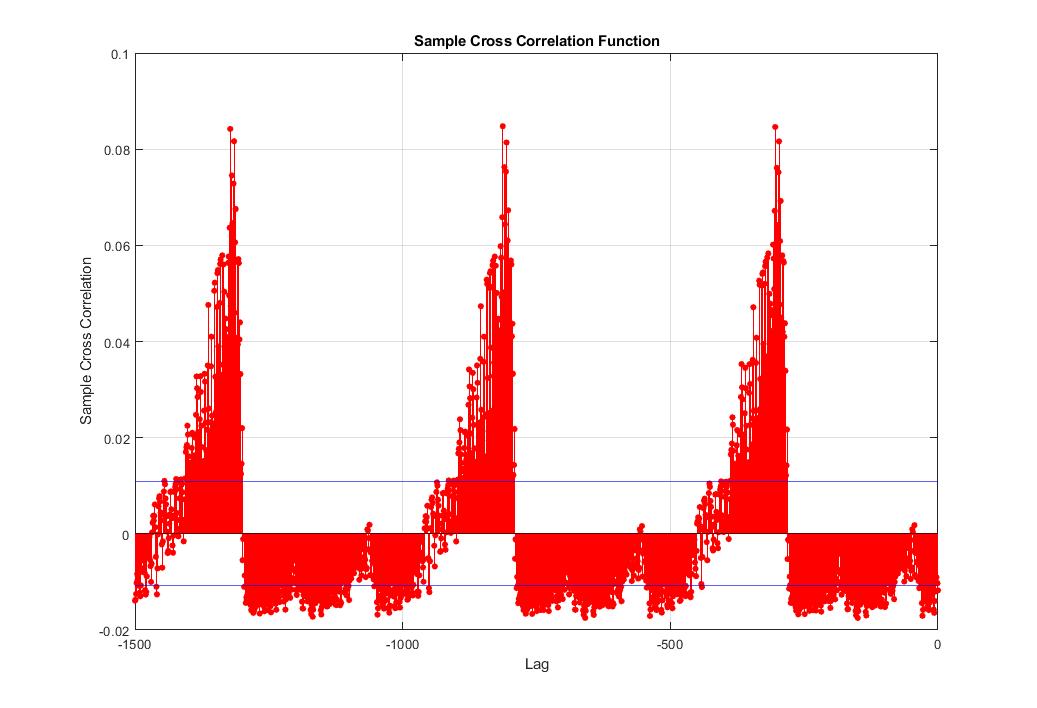
max\_val=299; % The first maximum is at -304 as can be seen from the cross correlation plot

time\_start=max\_val\*Ts; % Calculating the starting time of the song

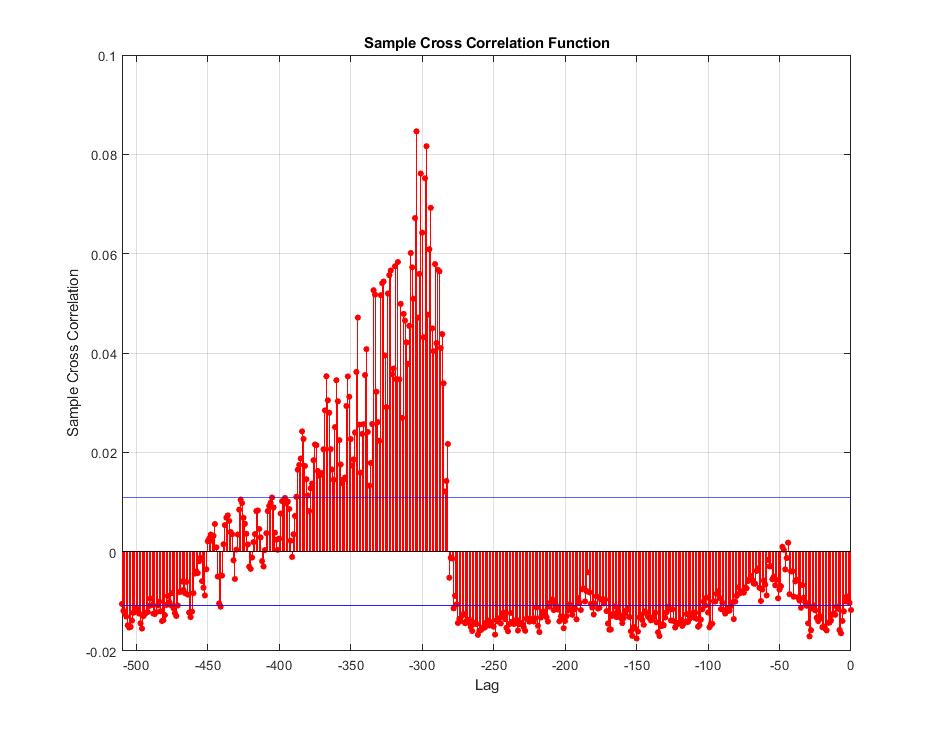
fprintf('BPM = %d, Start = %0.2ds', bpm, time\_start); % Printing the bpm and the starting time values

The rectified half-wave signal is below;

The auto-correlated function plot is below;

The cross-correlated function plot is below;

The interval between two beats is defined as;

Since our value is 509, we will get the BPM value as **130**.

The first beat time is defined as , we can get the value from the intervaled cross-corelation plot. The x-axis value at the maximum is **-304**. So our value is going to be **304**. Our starting time will be **0.276s**.