MultimediaSignal Processing 1st Module

MATLAB part [12 pts]

August 30th, 2021

Text:

1. [3 pt] You are given the file "y.mat", which contains an audio signal y and its sampling rate Fs.
    - Load the file into the MATLAB workspace by running: "load('y.mat')"
    - The audio signal can be listened (it is not required for the exam, but it can help you in understanding the exam questions) by running "sound(y, Fs)".
    - The signal y contains the sound of a cuckoo, mixed with some environmental noise and some warbling birds. In particular, the cuckoo spectrum is centered around normalized frequency = 0.06, and has a total normalized bandwidth = 0.03.
    - Design a FIR filter h(n) using the windowing method in order to maintain only the cuckoo and filter out the other signals. Use a filter order = 64.
    - Apply the filter h(n) to the signal y, obtaining the signal y_f, and verify (optional) that only the cuckoo sound is maintained by running "sound(y_f, Fs)".
    - Plot the absolute values of the DFT of y and the DFT of y_f as a function of normalized frequencies, starting from 0. In particular, plot the DFT behaviour in dB. Looking at the plot, which is (approximately) the maximum gap in dB between the DFT(y) and the DFT(y_f) around the Nyquist frequency? Which is the gap in the band of the cuckoo?
2. [1.5 pt] The function "overlap_save.m" (you can find the code below) implements the signal filtering by means of the overlap and save method.
    - The function receives as inputs:
        - The signal to be filtered; the FIR filter to be used; the block length to be used for the overlap and save; the amount of overlapping samples to be used for the overlap and save.
    - The function returns as output the filtered signal.
    - Call the function on the input signal y using the filter h, in order to maintain only the cuckoo. Use a block length which is twice the length of h. Define the output signal as y_os.
    - Verify, in the DFT domain, that the signal y_f and the signal y_os are the same.
3. [3.5 pt] A signal x, with the same length of y, is summed to y. In particular, x is the contribution of three cosinusoidal signals and has a period of 300 samples. The three signals have normalized frequencies f0, f1=f0/25, f2=f0/3, and have all the same amplitude = 0.25.
    - Define the signal x, and the signal z = x + y.
    - Filter the signal z with the filter h, defining z_f.
    - Plot the absolute value of the DFT(z) and the DFT(z_f) in dB in the same figure, as a function of positive normalized frequencies. Is the filter able to maintain again only the sound related to the cuckoo? Which is the sinusoid, among the three, that we can still hear? (you can verify it (optional) with the function "sound").

4. [4 pt] We want to project a zeros-poles filter G(z) in order to remove the sinusoidal component (choosing among f0, f1, and f2) with the strongest contribution on z_f.
    - Choose the poles and the zeroes of the filter to remove the selected sinusoid and to avoid altering too much the other frequency components (hint: you should select only two zeroes and two poles).
    - Set the filter gain such that the filter has no impact on the frequency component = 0.
    - Plot the amplitude of the frequency response of the filter as a function of the frequency axis starting from 0. Use the function "freqz", using 2048 samples and using the flag "whole" to see the entire frequency spectrum. Which is the name typically used to define a filter with this behaviour?
    - Filter the signal z_f, defining the signal z_f_f.
    - Verify, in the DFT domain, that the signal z_f_f contains only the cuckoo sound as main sound. You can also listen (optional) to the signal z_f_f to verify it.

```
%%%%%%%%%%%%%%%%%% FUNCTION CODE TO BE COPIED INTO MATLAB %%%%%%%%%%%%%%%%%%%%%%%%

function [filtered_signal] = overlap_save(signal, filter, L, overlap)
%overlap_save filters the input signal with the overlap and save method.
%   INPUTS:
%   signal: input signal to be filtered
%   filter: FIR filter to use
%   L: block size to perform fft-domain multiplications
%   overlap: amount of samples for the overlap
%   OUTPUT:
%   filtered_signal: output filtered signal

signal_pad = padarray(signal, [0, overlap], 'pre');
num_blocks = ceil((length(signal_pad) - L)/(L-overlap)) + 1;
if length(signal_pad) < (num_blocks - 1)*(L-overlap) + L
    signal_pad = padarray(signal_pad, [0, (num_blocks - 1)*(L-overlap) + L - length(signal_pad)], 'post');
end
filtered_signal = [];
filter_f = fft(filter, L);
for b = 1: num_blocks
    block_f = fft(signal_pad(1 + (b-1)*(L - overlap):(b-1)*(L - overlap) + L));
    y_block = ifft(block_f .* filter_f);
    filtered_signal = [filtered_signal, y_block(length(filter):end)];
end
filtered_signal = filtered_signal(1:length(signal));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Solution:

```
close all
clearvars
clc

%% 1.

% [3 pt] You are given the file y.mat, which contains an audio signal y
and its sampling rate Fs.
% Load the file into the MATLAB workspace by running: load('y.mat')
% The audio signal can be listened (it is not required for the exam,
% but it can help you in understanding the exam questions) by running
sound(y, Fs).
% The signal y contains the sound of a cuckoo, mixed with some
environmental
% noise and some warbling birds. In particular, the cuckoo spectrum is
% centered around normalized frequency = 0.06, and has a total normalized
bandwidth = 0.03.
% Design a FIR filter h(n) using the windowing method in order to maintain
% only the cuckoo and filter out the other signals. Use a filter order =
64.
% Apply the filter h(n) to the signal y, obtaining the signal y_f, and
% verify (optional) that only the cuckoo sound is maintained by running
sound(y_f, Fs).
% Plot the absolute values of the DFT of y and the DFT of y_f as a
function
% of normalized frequencies, starting from 0. In particular, plot the DFT
behaviour in dB.
% Looking at the plot, which is (approximately) the maximum gap in dB
between the DFT(y)
% and the DFT(y_f) around the Nyquist frequency? Which is the gap in the
band of the cuckoo?

% load the file
load('y.mat');

% uncomment for listening to the input signal y
% sound(y, Fs);

% cuckoo frequency specifications
f_cuckoo = 0.06;
Bw_cuckoo = 0.03;

% Design a FIR filter h(n) using the windowing method in order to maintain
% only the cuckoo and filter out the other signals. Use a filter order =
64.
F1 = f_cuckoo - Bw_cuckoo/2;
F2 = f_cuckoo + Bw_cuckoo/2;
```

```
h = fir1(64, [2*F1, 2*F2]);

% Apply the filter h(n) to the signal y, obtaining the signal y_f, and
% verify (optional) that only the cuckoo sound is maintained by running
sound(y_f, Fs).

y_f = filter(h, 1, y);
% uncomment for listening to the signal y_f
% sound(y_f, Fs);

% Plot the absolute values of the DFT of y and the DFT of y_f as a
function
% of normalized frequencies, starting from 0. In particular, plot the DFT
behaviour in dB.

Y = fft(y);
Y_f = fft(y_f);
norm_freq_axis = 0: 1/length(y): 1 - 1/length(y);
figure; plot(norm_freq_axis, 20*log10(abs(Y)));
hold on, plot(norm_freq_axis, 20*log10(abs(Y_f)));
legend('|Y(f)| [dB]', '|Y_{f}(f)| [dB]');
grid;

% Looking at the plot, which is (approximately) the maximum gap in dB
between the DFT(y)
% and the DFT(y_f) around the Nyquist frequency? Which is the gap in the
band of the cuckoo?

% The gap around the Nyquist frequency is more or less 60 dB.
% The gap in the band of the cuckoo is more or less 0 dB, as the filter
has
% to maintain these frequencies almost unaltered.

%% 2.

% [1.5 pt] The function overlap_save.m (you can find the code below)
implements
% the signal filtering by means of the overlap and save method.
% The function receives as inputs:
% The signal to be filtered; the FIR filter to be used; the block length
to be
% used for the overlap and save; the amount of overlapping samples to be
used for the overlap and save.
% The function returns as output the filtered signal.
% Call the function on the input signal y using the filter h, in order to
% maintain only the cuckoo. Use a block length which is twice the length
of h.
% Define the output signal as y_os.
```

```
% Verify, in the DFT domain, that the signal y_f and the signal y_os are
the same.

% Call the function on the input signal y using the filter h, in order to
% maintain only the cuckoo. Use a block length which is twice the length
of h.
% Define the output signal as y_os.
L = 2*length(h);
% By definition of the overlap and save method:
overlap = length(h) - 1;
y_os = overlap_save(y, h, L, overlap);

% Verify, in the DFT domain, that the signal y_f and the signal y_os are
the same.
Y_os = fft(y_os);
figure;
plot(norm_freq_axis, 20*log10(abs(Y_f)));
hold on, plot(norm_freq_axis, 20*log10(abs(Y_os)), '--');
legend('|Y_f(f)| [dB]', '|Y_{os}(f)| [dB]');
grid;

%% 3.

% [3.5 pt] A signal x, with the same length of y, is summed to y.
% In particular, x is the contribution of three cosinusoidal signals and
% has a period of 300 samples. The three signals have normalized
frequencies
% f0, f1=f0/25, f2=f0/3, and have all the same amplitude = 0.25.
% Define the signal x, and the signal z = x + y.
% Filter the signal z with the filter h, defining z_f.
% Plot the absolute value of the DFT(z) and the DFT(z_f) in dB in the same
figure,
% as a function of positive normalized frequencies. Is the filter able to
% maintain again only the sound related to the cuckoo? Which is the
sinusoid,
% among the three, that we can still hear? (you can verify it (optional)
with the function sound.

P = 300;

% Define the signal x, and the signal z = x + y.

% The period of a sum of sinusoidal signal is the least common multiple
% of their periods.
% The periods of the three sinusoids (in number of samples) are:
% p0 = 1/f0; p1 = 1/f1; p2 = 1/f2;
% we have to find the least common multiple of 1/f0, 25/f0, 3/f0.
% --> lcm = 1/f0 * 25 * 3 = 75/f0.
% 75/f0 = P --> f0 = 75/P
```

```
f0 = 75/P;
f1 = f0/25;
f2 = f0/3;
n = 0:length(y) - 1;
x = 0.25*cos(2*pi*f0*n) + 0.25*cos(2*pi*f1*n) + 0.25*cos(2*pi*f2*n);

z = x + y;

% Filter the signal z with the filter h, defining z_f.
z_f = filter(h, 1, z);

% Plot the absolute value of the DFT(z) and the DFT(z_f) in dB in the same
figure,
% as a function of positive normalized frequencies.

Z = fft(z);
Z_f = fft(z_f);
figure; plot(norm_freq_axis, 20*log10(abs(Z)));
hold on; plot(norm_freq_axis, 20*log10(abs(Z_f)), '--');
legend('|Z(f)| [dB]', '|Z_{f}(f)| [dB]');

% Is the filter able to maintain again only the sound related to the
cuckoo? Which is the sinusoid,
% among the three, that we can still hear?

% We still hear the sinusoid
% centered aroung f2, which is the nearest one to the band of the cuckoo.
% We can notice it from the peak related to f2 in the spectrum of z_f,
% which maintains relatively high.

% uncomment for listening to the signal z_f
% sound(z_f, Fs);

%% 4.

% [4 pt] We want to project a zeros-poles filter G(z) in order to remove
the
% sinusoidal component (choosing among f0, f1, and f2) with the strongest
contribution on z_f.
% Choose the poles and the zeroes of the filter to remove the selected
sinusoid
% and to avoid altering too much the other frequency components
% (hint: you should select only two zeroes and two poles).
% Set the filter gain such that the filter has no impact on the frequency
% component = 0.
% Plot the amplitude of the frequency response of the filter as a function
of
```

```
% the frequency axis starting from 0. Use the function "freqz", using 2048
samples
% and using the flag "whole" to see the entire frequency spectrum.
% Which is the name typically used to define a filter with this behaviour?
% Filter the signal z_f, defining the signal z_f_f.
% Verify, in the DFT domain, that the signal z_f_f contains only the
% cuckoo sound as main sound. You can also listen (optional) to the signal
z_f_f to verify it.

% Choose the poles and the zeroes of the filter to remove the selected
sinusoid
% and to avoid altering the other frequency components
% (hint: you should select only two zeroes and two poles).

% We have to select zeroes and poles centered around the frequency f2.
% Zeroes can have amplitude = 1, poles should be inside the unit circle
for
% stability. For instance, we can select poles with amplitude = 0.95.

zeroes = [exp(2*1i*pi*f2); exp(-2*1i*pi*f2)];
poles = [0.95*exp(2*1i*pi*f2); 0.95*exp(-2*1i*pi*f2)];

% Set the filter gain such that the filter has no impact on the frequency
% component = 0.

A = poly(poles);
B = poly(zeroes);
% The filter gain should be = 1.
k = sum(A)/sum(B);
B = B*k;

% Plot the amplitude of the frequency response of the filter as a function
of
% the frequency axis starting from 0. Use the function "freqz", using 2048
samples
% and using the flag "whole" to see the entire frequency spectrum.

[G, omega] = freqz(B, A, 2048, 'whole');
figure; plot(omega/(2*pi), abs(G));
grid;

% Which is the name typically used to define a filter with this behaviour?
% This is a Notch filter.

% Filter the signal z_f, defining the signal z_f_f.
z_f_f = filter(B, A, z_f);

% Verify, in the DFT domain, that the signal z_f_f contains only the
% cuckoo sound as main sound.
```

```
Z_f_f = fft(z_f_f);

figure;
plot(norm_freq_axis, 20*log10(abs(Z)));
hold on;
plot(norm_freq_axis, 20*log10(abs(Z_f)), '*-');
plot(norm_freq_axis, 20*log10(abs(Z_f_f)), '--');
legend('|Z(f)| [dB]', '|Z_f(f)| [dB]', '|Z_{f_f}(f)| [dB]');

% uncomment for listening to the signal z_f
%sound(z_f_f, Fs);

%%%%%%%%%%%%%%%%%%%%%%%% function code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [filtered_signal] = overlap_save(signal, filter, L, overlap)
%overlap_save filters the input signal with the overlap and save method.
%   INPUTS:
%   signal: input signal to be filtered
%   filter: FIR filter to use
%   L: block size to perform fft-domain multiplications
%   overlap: amount of samples for the overlap
%   OUTPUT:
%   filtered_signal: output filtered signal

signal_pad = padarray(signal, [0, overlap], 'pre');
num_blocks = ceil((length(signal_pad) - L)/(L-overlap)) + 1;
if length(signal_pad) < (num_blocks - 1)*(L-overlap) + L
    signal_pad = padarray(signal_pad, [0, (num_blocks - 1)*(L-overlap) + L
- length(signal_pad)], 'post');
end
filtered_signal = [];
filter_f = fft(filter, L);
for b = 1: num_blocks
    block_f = fft(signal_pad(1 + (b-1)*(L - overlap):(b-1)*(L - overlap) +
L));
    y_block = ifft(block_f .* filter_f);
    filtered_signal = [filtered_signal, y_block(length(filter):end)];
end
filtered_signal = filtered_signal(1:length(signal));
end
```