MultimediaSignal Processing 1st Module

MATLAB part [11 pts]

June 26-th, 2020

Text:

1. [0.5 pt] Close the opened figures, clear the workspace and clear the command window.
2. [2 pt] Define the signal z = x + y, where:
    - x is a sinusoidal signal with period = 10 milliseconds, amplitude 3.
    - y is a sinusoidal signal with period = 3 milliseconds, amplitude 1.5.
    - both signals have duration = 1.25 seconds and are sampled every 1 millisecond.
3. [2 pt] Define a FIR filter h(n) using the window-based method, with 32 samples, normalized cut-off frequency = 0.25:
    - Is the filter stable? How to check it?
    - Filter the signal z using the function “filter.m”, defining the signal z_filtered.
4. [1 pt] Given the function "overlap_add.m" (copy the function code into your matlab):
    - This function requires as input parameters the signal to be filtered, the FIR filter to use, the amount of signal samples to filter at a time (block-size);
    - This function returns the filtered signal with the same number of samples of the input signal;
    - Filter the signal z using the function "overlap_add.m", considering signal blocks of length 64 samples. Define this signal as z_filtered_overlap_add.
5. [0.5 pt] Plot the first 300 samples of z_filtered and z_filtered_overlap_add in the same figure. Is there any relationship between the two signals?
6. [2 pt] Compute the DFT of the original signal z and of the filtered signal z_filtered:
    - plot (in the same figure) the magnitudes of the DFT of the signals as a function of frequency [Hz].
    - Is there any difference between the two magnitudes? If yes, why?
7. [3 pt + 1 extra point] Downsample the signals z and z_filtered by a factor M = 2.
    - Compute their DFTs and plot (in the same figure) the DFT magnitudes as a function of NORMALIZED frequency.
    - Is there any difference between the two magnitudes? If yes, why?

```
%%%%%%%%%%%%%%%%% FUNCTION CODE TO BE COPIED INTO MATLAB %%%%%%%%%%%%%%
function [filtered_signal] = overlap_add(signal, filter, B)
% INPUT PARAMETERS:
% signal = input signal
% filter = FIR filter used to filter the signal
% B = block-size for the overlap and add method.
% OUTPUT:
% filtered_signal = output of the overlap and add method

% support of the convolution
Lconv = B + length(filter) - 1;
% how many non-overlapping blocks of length B are there in signal?
num_blocks = ceil(length(signal) / B);
if length(signal) < num_blocks * B
   % pad z with zeros
   signal_pad = padarray(signal, [0, num_blocks*B - length(signal)], 'post');
else
   signal_pad = signal;
end
% define the filtered signal
filtered_signal = zeros(1, num_blocks*Lconv);
% FFT of the filter over Lconv samples.
filter_f = fft(filter, Lconv);
for b = 1:num_blocks
   block_f = fft([signal_pad(1 + (b-1)*B:(b-1)*B + B), zeros(1, Lconv-B)]);
   output_block = ifft(block_f .* filter_f);
   filtered_signal(1 + (b-1)*B:(b-1)*B + Lconv) = ...
      filtered_signal(1 + (b-1)*B:(b-1)*B + Lconv) + output_block;
end
filtered_signal = filtered_signal(1:length(signal));
end

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

Solution:

```
%% 1. [0.5 pt]

% close the opened figures, clear the workspace and clear the command
window

close all
clearvars
clc

%% 2. [2 pt]

% Given the signal z = x + y, where:
% x is a sinusoidal signal with period = 10 milliseconds, amplitude 3.
% y is a sinusoidal signal with period = 3 milliseconds, amplitude 1.5.
% both signals have a duration = 1.25 seconds and are sampled every 1
millisecond.

ampl_x = 3;
period_x = 0.01;
f0_x = 1 / period_x;

ampl_y = 1.5;
period_y = 0.003;
f0_y = 1 / period_y;

ts = 0.001;
duration = 1.25;
time = 0:ts:duration;
Fs = 1/ts;

x = ampl_x * cos(2*pi*f0_x*time);
y = ampl_y * cos(2*pi*f0_y*time);
z = x + y;

%% 3. [2 pt]

% Define a FIR filter h(n) using the window-based method, with 32 samples,
normalized cut-off frequency = 0.25.

filter_order = 31;
cutoff_filter = 0.25 * 2;
h = fir1(filter_order, cutoff_filter);

% Is the filter stable? How to check it?
% Since the filter is a FIR filter, it is stable by definition.
```

```matlab
% filter the signal z using the function "filter", defining the signal
% z_filtered.

z_filtered = filter(h, 1, z);

%% 4. [1 pt]

% given the function "overlap_add.m", which requires as input parameters:
% the signal to be filtered
% the FIR filter
% the amount of signal samples to filter at a time (block-size)

% the function "overlap_add.m" returns the filtered signal with the same
amount of samples
% as the input signal.

% Filter the signal z using the function "overlap_add.m", considering
signal blocks
% of length 64 samples. Define this signal as z_filtered_overlap_add.

z_filtered_overlap_add = overlap_add(z, h, 64);

%% 5. [0.5 pt]

% plot the first 300 samples of z_filtered and z_filtered_overlap_add in
the same figure.
% Is there any relationship between the two signals?

figure;
plot(z_filtered(1:300));
hold on;
plot(z_filtered_overlap_add(1:300), '--');

% The two signals coincide.

%% 6. [2 pt]

% compute the DFT of the original signal z and of the filtered signal
% z_filtered.
% plot (in the same figure) the magnitudes of the DFT of the signals as a
function of frequency
% [Hz].
% Is there any difference between the two magnitudes? If yes, why?

z_fft = fft(z);
z_filtered_fft = fft(z_filtered);
N_samples_fft = length(z);
```

```matlab
freq_axis = 0:(1/N_samples_fft) * Fs:Fs - (1/N_samples_fft) * Fs;

figure;
plot(freq_axis, abs(z_fft));
hold on;
plot(freq_axis, abs(z_filtered_fft));

% The two magnitudes are different because the filtered signal does not
% contain the frequency peaks to due the sinusoid y, as they have been
% filtered out by the low-pass filter.

%% 7. [3 pt + 1 extra point]

% downsample the signals z and z_filtered by a factor M = 2. Compute their
% DFTs and plot (in the same figure) their magnitudes as a function of
% NORMALIZED frequency.
% Is there any difference between the two magnitudes? If yes, why?

M = 2;

z_down = z(1:M:end);
z_filtered_down = z_filtered(1:M:end);

z_down_fft = fft(z_down);
z_filtered_down_fft = fft(z_filtered_down);

N_samples_fft_down = length(z_down);
norm_freq_axis = 0:1/N_samples_fft_down:1 - 1/N_samples_fft_down;

figure;
plot(norm_freq_axis, abs(z_down_fft));
hold on;
plot(norm_freq_axis, abs(z_filtered_down_fft));

% The two magnitudes are different.
% The spectrum of the downsampled signal z_down contains frequency ALIAS:
% the peaks in normalized frequency = 0.2 and 0.8 are due to the sinusoid
% x, but the peak in normalized frequency = 0.66 (and its symmetric in
% 0.33) are aliasing components which need to be filtered out.
% By filtering the signal with a LPF before to downsample it, we avoid
% introducing alias. This is why the spectrum of z_filtered_down does not
% contain aliasing components.

%% function code

function [filtered_signal] = overlap_add(signal, filter, B)
% INPUT PARAMETERS:
```

```
% signal = input signal
% filter = FIR filter used to filter the signal
% B = block-size for the overlap and add method.
% OUTPUT:
% filtered_signal = output of the overlap and add method

% support of the convolution
Lconv = B + length(filter) - 1;

% how many non-overlapping blocks of length B are there in signal?
num_blocks = ceil(length(signal) / B);

if length(signal) < num_blocks * B
    % pad z with zeros
    signal_pad = padarray(signal, [0, num_blocks*B - length(signal)],
'post');
else
    signal_pad = signal;
end

% define the filtered signal
filtered_signal = zeros(1, num_blocks*Lconv);

% FFT of the filter over Lconv samples.
filter_f = fft(filter, Lconv);

for b = 1:num_blocks

    block_f = fft([signal_pad(1 + (b-1)*B:(b-1)*B + B), zeros(1, Lconv-
B)]);

    output_block = ifft(block_f .* filter_f);

    filtered_signal(1 + (b-1)*B:(b-1)*B + Lconv) = ...
        filtered_signal(1 + (b-1)*B:(b-1)*B + Lconv) + output_block;

end

filtered_signal = filtered_signal(1:length(signal));

end
```