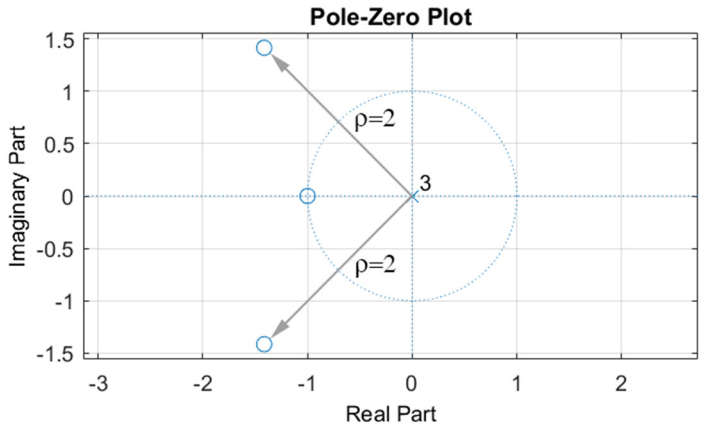
Multimedia Signal Processing 1st Module and Fundamentals of Multimedia Signal Processing

date: November 7th, 2022

Ex.1 (Pt.12)

A FIR filter with 3 zeros has the following pole-zero plot



The two conjugate zeros are placed with an angle of $\pm\frac{3}{4}\pi$ with respect to the positive real axis direction.

- [3 pts] Define the z-transform $H_1(z)$ and the impulse response $h_1[n]$.
- [3 pts] Define a minimum phase FIR filter $h_2[n]$ with the same amplitude response.
- [4 pts] A signal $x(t)=2\cdot\cos(2\pi 500t)+3\cdot\cos(2\pi 1000t)$ is sampled at 2kHz obtaining the signal $x[n]$ that is then filtered with the filter $h_2(n)$. Calculate just the amplitude of the different frequencies of the output signal $y(n)$.
- [2 pts] What would be the differences between the signal $x[n]$ filtered with the filter $h_1[n]$ and $x[n]$ filtered with the filter $h_2[n]$?

Ex.2 (Pt.11)

A signal $x[n]=\{2,1,-2,3,0,1,0,-2,1,-2\}$ is filtered with a filter $H(z)=1-z^{-1}$.

- [2 pts] Evaluate the output $y[n]$ in the time domain.
- [3 pts] Evaluate the output $y[n]$ using the Overlap and Add approach on blocks of 4 samples describing all the processing steps.
- [3 pts] Evaluate the output $y[n]$ using the Overlap and Save approach on blocks of 4 samples describing all the processing steps.
- [3 pts] Just describe (without calculations) how the output $y[n]$ can be obtained using the Overlap and Save approach on blocks of 4 samples working in the frequency domain.

CONTINUES ON THE BACK

Ex.3 (Pt.12) To be solved writing the MATLAB code on the sheet.

1) [3 pt] The signal x(n) contains two sinusoidal contributions (with the same amplitude = 1) at the normalized frequencies 0.1 and 0.25. The period of x(n) is 1.25 [ms] and the duration is 62.5 [ms]. Define the signal x(n).
2) [3 pt] Define the filter H(z) as $H(z) = (0.81 + z^{-2})\cdot(1 - 1.6\cos(\pi/5)z^{-1} + 0.64z^{-2}) / (1 + 0.81z^{-2})$.
   - Filter the signal x(n) with H(z), defining y(n).
   - Which is the value of y(n = 0)? Define it in MATLAB, but specify also the numerical value that you expect.
3) [3.5 pt] Compute the all-pass minimum-phase decomposition of the filter H(z), defining H_ap(z) and H_min(z) as the two components. (Hint: no computations are needed!)
   - Filter the signal x(n) with H_ap(z) and H_min(z), defining y_ap(n) and y_min(z).
   - Define the signal $w(n) = 0.5 \cdot x(n) + 0.5 \cdot y\_ap(n)$.
   - Find the filter H_w(z) such that $W(z) = H\_w(z) \cdot X(z)$.
4) [2.5 pt] Compute the DFTs of the signals x(n), y(n), y_min(n), y_ap(n), w(n) and plot their absolute values as a function of the normalized frequency axis, starting from frequency -0.5. Comment on the position/amplitude of the peaks you expect to see for every signal.

Solutions

Ex.1

$$H_1(z) = (1+z^{-1}) \cdot (1+2\sqrt{2}z^{-1}+4z^{-2}) = 1+(2\sqrt{2}+1)z^{-1}+(2\sqrt{2}+4)z^{-2}+4z^{-3}$$

$$h_1[n] = \{1, 2\sqrt{2}+1, 2\sqrt{2}+4, 4\}$$

$$H_2(z) = A(1+z^{-1}) \cdot \left(1+\frac{\sqrt{2}}{2}z^{-1}+\frac{1}{4}z^{-2}\right)$$

In order to obtain the same amplitude response for H_1 and H_2 we can set $H_1(z=1) = H_2(z=1)$, i.e.

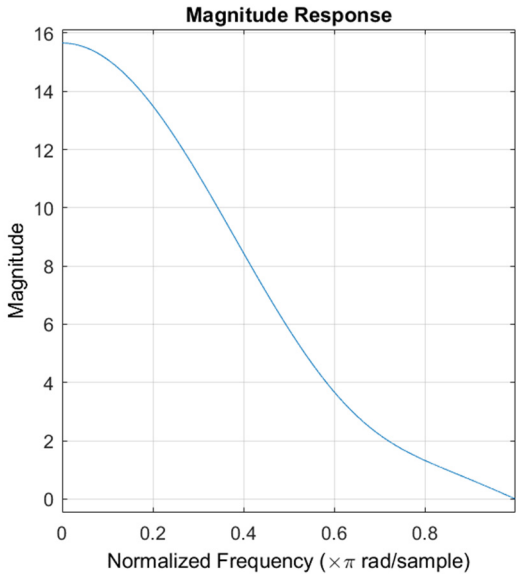
$$10+4\sqrt{2} = A\left(\frac{5}{2}+\sqrt{2}\right) \rightarrow A = 4$$

$$h_2[n] = \{4, 4+2\sqrt{2}, 1+2\sqrt{2}, 1\}$$

The sampled signal will be:

$$x[n] = \left(e^{-j\frac{\pi}{2}n}+e^{j\frac{\pi}{2}n}\right)+3\left(e^{-j\pi n}+e^{j\pi n}\right)$$

According to the filter h_2 amplitude response:



The component at the Nyquist frequency will be completely removed while the component at the $\frac{\pi}{2}$ will be amplified by a factor of: $|H_2(z=j)| = \left|4+(4+2\sqrt{2})(-j)+(1+2\sqrt{2})(-1)+j\right| = \sqrt{34} \simeq 5.83$

Filtering $x[n]$ with $h_1[n]$ or with $h_2[n]$ will give two outputs with the same amplitude but with a different phase.

Ex.2

The filter is an high-pass filter and the output, obtained using the convolution is

$$y[n]=\{2,-1,-3,5,-3,1,-1,-2,3,-3,2\}$$

The Overlap and Add approach, working on blocks of 4 samples, gives the following processing steps:

$y_1[n]=\{2,1,-2,3\}*\{1,-1\}=\{2,-1,-3,5,-3\}$ where the last sample in red is removed and stored to be added to the next output block:

$y_2[n]=\{0,1,0,-2\}*\{1,-1\}$ with remainder $=\{0\text{-}3,1,-1,-2,\ 2\}$ where the "-3" is the remainder from the previous convolution while "2" is stored as the remainder for the next output block.

For the last input block, since the length is shorter than 4 elements, we need to zero-pad the signal and then, I have to apply again the same procedure.

For the Overlap and Save, since the length of the filter is 2, I have to add 2-1=1 zeros in front of my signal (just to the first block) to account for the circular convolution:

$y_1[n]=\{0,2,1,-2\}\circledast\{1,-1\}=\{\underline{2},2,-1,-3\}$: the first output sample (the underlined one) must be removed since it is due to the spurious effect of circular convolution.

$y_2[n]=\{-2,3,0,1\}\circledast\{1,-1\}=\{-\underline{3},5,-3,1\}$ again, the first output sample must be removed.

Etc.

Working in the frequency domain implies that we will use the DFT with 4 samples and the filter has to be zero-padded to 4 elements: $h(n)=[1\quad -1\quad 0\quad 0]$.

Then, e.g., $y_1(n)=\mathfrak{F}^{-1}\{\mathfrak{F}\{x(n)\}\cdot\mathfrak{F}\{h(n)\}\}$ where $\mathfrak{F}$ and $\mathfrak{F}^{-1}$ are the Discrete Fourier Transform and the Inverse Discrete Fourier Transform respectively.

Ex.3 (MATLAB CODE)

```
close all
clearvars
clc

%% 1. [3 pt]

% The signal x(n) contains two sinusoidal contributions (with the same
% amplitude = 1) at the normalized frequencies 0.1 and 0.25.
% The period of x(n) is 1.25 [ms] and the duration is 62.5 [ms].
```

```
% Define the signal x(n).

f0 = 1/10;
f1 = 1/4;
P_seconds = 1.25e-3;
duration = 62.5e-3;

% We need to find the sampling rate to define the time-axis.
% To find Fs, we know that P_samples and P_seconds are related one
with the
% other by P_samples = Fs * P_seconds.
% First, find P_samples, then compute Fs = P_samples/P_seconds.

P_samples = 20; % lcm(10, 4) = lcm(1/f0, 1/f1)
Fs = P_samples/P_seconds;

time = 0:1/Fs:duration - 1/Fs;

x = cos(2*pi*f0*Fs*time) + cos(2*pi*f1*Fs*time);

%% 2. [3 pt]

% Define the filter H(z) as H(z) = (0.81 + z^(-2))(1 -
1.6cos(pi/5)z^(-1)+
% 0.64z^(-2)) / (1 + 0.81z^(-2)).
% Filter the signal x(n) with H(z), defining y(n).
% Which is the value of y(n = 0)? Define it in Matlab, but specify
also
% the numerical value that you expect.

% To define the filter, we can exploit the convolution property
B = conv([0.81, 0, 1], [1, -1.6*cos(pi/5), 0.64]);
A = [1, 0, 0.81];

% % to better analyze the filter (not required)
% figure;
% zplane(B, A);
% title('Z-plane of filter H(z)');
% grid;
% [H, omega] = freqz(B, A, 1024, 'whole');
% figure,
% plot(omega./(2*pi), abs(H));
% title('|DTFT| of the filter H(z)');
% grid;
% xlabel('f [norm]');

% filter the signal
y = filter(B, A, x);

y_0 = x(1)*B(1);
% we expect that y_0 = x(n=0) * h(n=0)
% x(n=0) = 2, as we have the sum of two cosinusoidal signals being 1
in
% n=0.
```

```
% h(n=0) is only due to the numerator coefficient in n = 0, therefore
it
% will be 0.81.
% y_0 = 2 * 0.81 = 1.62

%% 3. [3.5 pt]

% Compute the all-pass minimum-phase decomposition of the filter H(z),
% defining H_ap(z) and H_min(z) as the two components.
% (Hint: no computations are needed!)
% Filter the signal x(n) with H_ap(z) and H_min(z), defining y_ap(n)
and y_min(z).
% Define the signal w(n) = 0.5 x(n) + 0.5 y_ap(n).
% Find the filter H_w(z) such that W(z) = H_w(z)X(z).

% The filter is already decomposed in an all-pass component and a
% minimum-phase component.
% The all-pass component H_ap(z) = (0.81 + z^(-2))/ (1 + 0.81z^(-2)).
% The minimum-phase component H_min(z) = (1 - 1.6cos(pi/5)z^(-1)+
% 0.64z^(-2)).
% Even if you want to follow the standard methodology, you will find
that,
% after all the steps, you end up with these two exact components.

B_ap = [0.81, 0, 1];
A_ap = [1, 0, 0.81];

B_min = [1, -1.6*cos(pi/5), 0.64];
A_min = 1;

% % to better analyze the filters (not required)
% figure;
% zplane(B_ap, A_ap);
% title('Z-plane of filter H_{ap}(z)');
% grid;
% [H, omega] = freqz(B_ap, A_ap, 1024, 'whole');
% figure,
% plot(omega./(2*pi), abs(H));
% title('|DTFT| of the filter H_{ap}(z)');
% grid;
% xlabel('f [norm]');
%
% figure;
% zplane(B_min, A_min);
% title('Z-plane of filter H_{min}(z)');
% grid;
% [H, omega] = freqz(B_min, A_min, 1024, 'whole');
% figure,
% plot(omega./(2*pi), abs(H));
% title('|DTFT| of the filter H_{min}(z)');
% grid;
% xlabel('f [norm]');

% Filter the signal x
y_ap = filter(B_ap, A_ap, x);
```

```
y_min = filter(B_min, A_min, x);

% Define the signal w
w = 0.5*x + 0.5*y_ap;

% Find the filter H_w such that W(z) = X(z) * H_w(z)
% W(z) = (X(z) + X(z)*H_ap(z)) / 2 = X(z) (H_ap(z) + 1)/2
% --> H(z) = (H_ap(z) + 1)/2.
% By making easy hand-written computations, we find the numerator and
% denominator coefficients.
B_w = 1.81/2*[1, 0, 1];
A_w = [1, 0, 0.81];
% --> This is a notch filter in f1. We can understand it by looking at
the
% position of the zeros and poles: they have the same phase, but the
zeros
% are on the unit circle.

% % to better analyze the filter (not required)
% figure;
% zplane(B_w, A_w);
% title('Z-plane of filter H_w(z)');
% grid;
% [H, omega] = freqz(B_w, A_w, 1024, 'whole');
% figure,
% plot(omega./(2*pi), abs(H));
% title('|DTFT| of the filter H_{w}(z)');
% grid;
% xlabel('f [norm]');

%% 4. [2.5pt]

% Compute the DFTs of the signals x(n), y(n), y_min(n), y_ap(n), w(n)
% and plot their absolute values as a function of the normalized
frequency
% axis, starting from frequency -0.5.
% Comment on the position/amplitude of the peaks you expect to see
% for every signal.

X = fft(x);
Y = fft(y);
Y_min = fft(y_min);
Y_ap = fft(y_ap);
W = fft(w);

N = length(y);
freq_axis = (0:1/N:1 - 1/N) - 0.5;

figure;
stem(freq_axis, abs(fftshift(X)));
title('Absolute value of the DFT of the signal x(n)');
grid;
xlabel('f [norm]');
% We find four peaks in -f0, -f1, f0, f1. They have the same
amplitude.
```

```
figure;
stem(freq_axis, abs(fftshift(Y)));
title('Absolute value of the DFT of the signal y(n)');
grid;
xlabel('f [norm]');
% We find four peaks in -f0, -f1, f0, f1.
% The peaks in frequency f0 and -f0 are slightly attenuated
% with respect to those of x(n),
% because the filter has zeros in f0 with absolute value = 0.8
% The peaks in frequency f1 and -f1 are not attenuated by the filter.
% Their amplitude differs from that of x(n) because the filter
introduces
% a gain in f = f1 which is = abs(1 -1.6*cos(pi/5)*1i-0.64) = 1.34

figure;
stem(freq_axis, abs(fftshift(Y_min)));
title('Absolute value of the DFT of the signal y_{min}(n)');
grid;
xlabel('f [norm]');
% We find basically no differences with respect to y(n). H(z) and
H_min(z)
% differ only for the all-pass component, which has no effect on the
% amplitude.

figure;
stem(freq_axis, abs(fftshift(Y_ap)));
title('Absolute value of the DFT of the signal y_{ap}(n)');
grid;
xlabel('f [norm]');
% The peaks are basically the same as x(n), because the filter is an
all-pass

figure;
stem(freq_axis, abs(fftshift(W)));
title('Absolute value of the DFT of the signal w(n)');
grid;
xlabel('f [norm]');
% The filter is a notch in f = f1. Therefore, we find only the
contribution
% at f0 and -f0, with an amplitude which is similar to that of x(n).
```