University of Liège

Homework 1

Applied digital signal processing

Quentin Graillet (20164386)
Maxime Meurisse (20161278)
Adrien Schoffeniels (20162843)

3e year of Bachelor Civil Engineer

Academic year 2018-2019

1 Magnitude response of a filter

We consider a filter with the transfer function

$$H(z) = \frac{b_0}{\left[1 - 2r\cos(\omega_0) z^{-1} + r^2 z^{-2}\right]^K}$$

To simplify the study of this function, we consider that it is a cascade of K second-order filters. The function can therefore be decomposed into a product of elementary transfer functions.

$$H(z) = K\left(\frac{b_0^{\frac{1}{K}}}{1 - 2r\cos(\omega_0) z^{-1} + r^2 z^{-2}}\right)$$

We replace the parameters K, r and b_0 by their value, and we store the coefficients of the numerator in a vector b, and those of the denominator in a vector a. We define a linearly spaced vector ranging from $-\pi$ to π and having 500 points. We then use the function `freqz` of Matlab[1] with these data and we display the result.

The magnitude response for $\omega_0 = \frac{\pi}{3}$ is shown in figure 1. To obtain the values of the magnitude in dB, we multiply all the values x by $10\log_{10}(x)$. The normalized angular frequency is scaled by π.

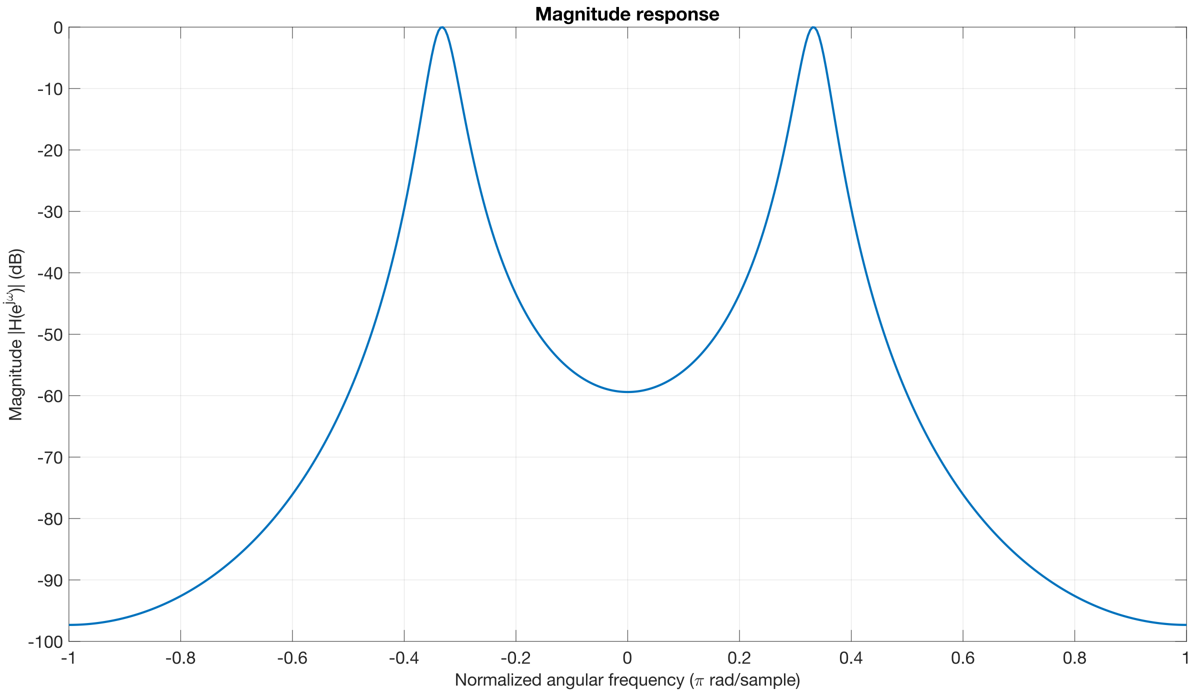

Figure 1 – Magnitude response $|H(e^{j\omega})|$ for $\omega_0 = \frac{\pi}{3}$.

We repeat the same procedure for $\omega = \frac{2\pi}{3}$ (figure 2).

Examining figures 1 and 2, we found that it is a band-pass filter. Indeed, this filter passes only a frequency band around the value ω_0.

[1] All Matlab codes used are attached to this report.

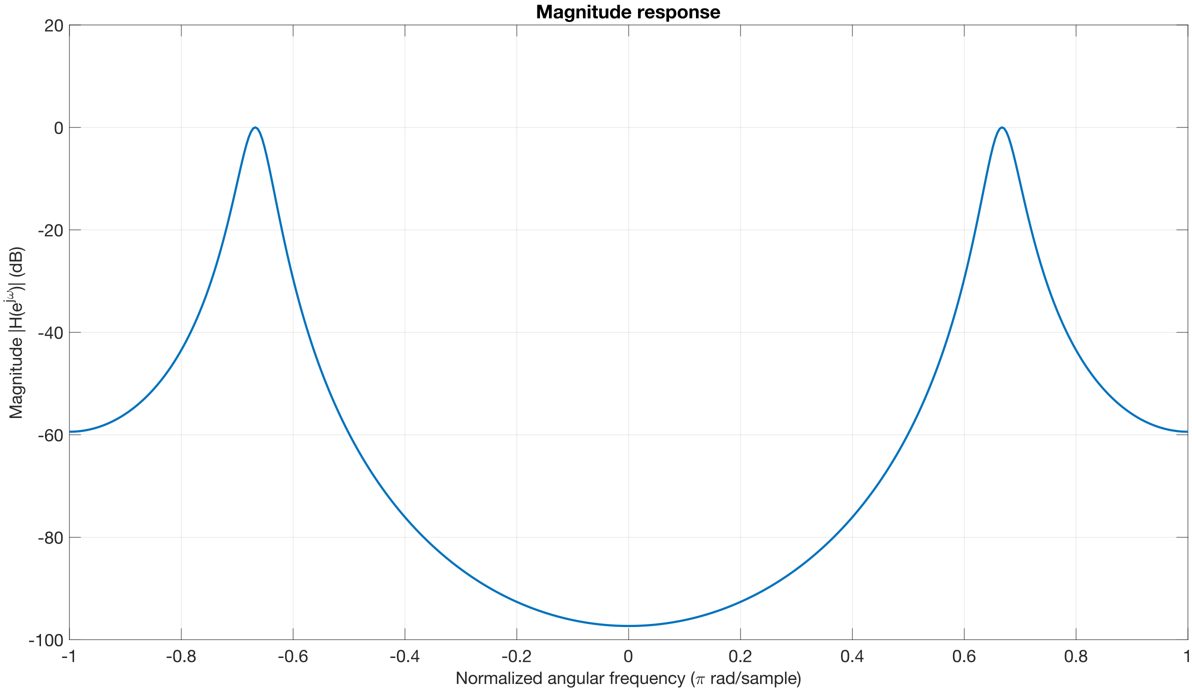

Figure 2 – Magnitude response $|H(e^{j\omega})|$ for $\omega_0 = \frac{2\pi}{3}$.

By varying the value of this parameter ω_0, we observe that the band-pass filter does not let the same frequency band pass.

In conclusion, the impact of the change of the parameter ω_0 is implicit. This defines which frequency band the filter will pass.

2 Autocorrelation of a single echo

Let the expression of a single echo $y[n]$ generated using the FIR filter

$$y[n] = x[n] + ax[n-D], \quad -1 < a < 1 \tag{†}$$

By definition, the expression of the autocorrelation $r_y[l]$ is given by

$$r_y[l] = \sum_{n=-\infty}^{\infty} y[n] \times y[n-l]$$

We can replace $y[n]$ by its expression (†)

$$r_y[l] = \sum_{n=-\infty}^{\infty} (x[n] + ax[n-D]) \times (x[n-l] + ax[n-l-D])$$

By distributing, we get

$$\begin{aligned} r_y[l] &= \sum_{n=-\infty}^{\infty} (a_1 + a_2 + a_3 + a_4) \\ &= \sum_{n=-\infty}^{\infty} a_1 + \sum_{n=-\infty}^{\infty} a_2 + \sum_{n=-\infty}^{\infty} a_3 + \sum_{n=-\infty}^{\infty} a_4 \end{aligned}$$

where

- $a_1 = x[n] \times x[n-l]$
- $a_2 = x[n] \times ax[n-l-D]$
- $a_3 = ax[n-D] \times x[n-l] = ax[n-D] \times x[n-D-(l-D)]$
- $a_4 = ax[n-D] \times ax[n-l-D]$

We note that we fall back on the definition of the autocorrelation of the signal x. By replacing the terms with this definition, we find the expression of the autocorrelation $r_y[l]$ in terms of the autocorrelation $r_x[l]$, D and a

$$\begin{aligned} r_y[l] &= r_x[l] + ar_x[l+D] + ar_x[l-D] + a^2 r_x[l] \\ &= (1+a^2) r_x[l] + a\left(r_x[l-D] + r_x[l+D]\right) \end{aligned}$$

3 Echo cancellation

Using the `audioread` and `xcorr` functions of Matlab, we play the sound and plot its autocorrelation function (figure 3).

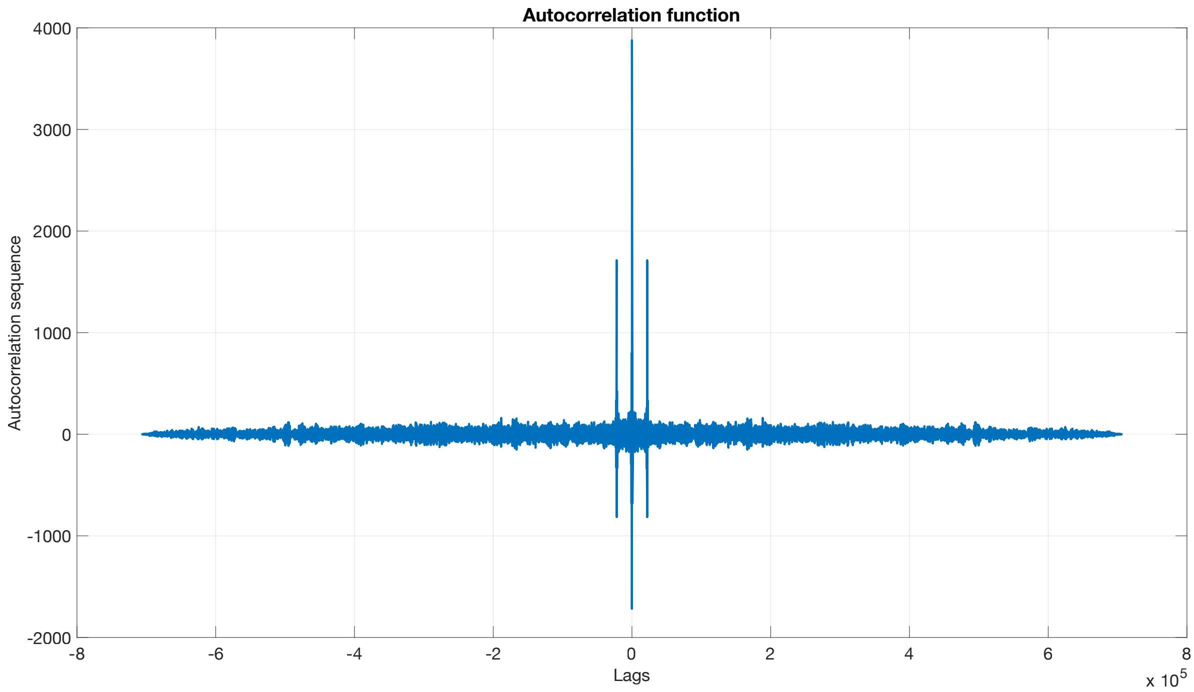

Figure 3 – Autocorrelation function of the sound `hw1_echo.wav`.

The expression of the signal with echo is given by

$$y(n) = x(n) + \alpha x(n-d)$$

with $x(n)$, the original sound without echo. The goal is to recover $x(n)$.

By passing into the frequency domain, we have

$$Y(z) = X(z) + \alpha X(z) z^{-d}$$

We can find the transfer function

$$\begin{aligned} H(z) &= \frac{Y(z)}{X(z)} \\ &= 1 + \alpha z^{-d} \end{aligned}$$

And so, we have

$$X(z) = \frac{Y(z)}{H(z)}$$

This means that we need to filter the observed signal $y(n)$ through the filter $\frac{1}{H(z)}$ in order to recover $x(n)$. To build the transfer function, we have to find the delay d.

In the figure 3, several peaks are observed. The delay can be found by observing the distance between two consecutive peaks.

We thus find the delay (expressed in number of sampling intervals) $d = 2.205 \times 10^4$. We can also find the corresponding delay expressed in seconds by divising this value by `fs`. We obtain (in seconds): $\tau = 0.5$.

We assume that the amplitude of the reflected sound is sixty percent of the emitted one, so $\alpha = 0.6$. We can now build the coefficients of the transfer function. These are stored in the vectors b (for the numerator) and a (for the denominator).

Now just call the function `filter` of Matlab with as arguments the vectors containing the coefficients and the sound to be filtered. The new sound obtained can be played with the function `sound` of Matlab. It can be seen that it no longer contains an echo.

A Matlab codes

A.1 Magnitude response of a filter

```
%% Applied digital signal processing - Homework 1

% Question 1 - Magnitude response of a filter
%
% University of Liege
% Academic year 2018-2019
%
% Authors:
%    - Quentin Graillet
%    - Maxime Meurisse
%    - Adrien Schoffeniels

%% Parameters

K = 8;
r = 0.9;
b_0 = 5.3936*1e-7;
w = linspace(-pi, pi, 500);

%% Magnitude responses

% Subquestion a)
magn_resp(pi/3, K, r, b_0, w);

% Subquestion b)
magn_resp((2*pi)/3, K, r, b_0, w);

%% Function
function H = magn_resp(w_0, K, r, b_0, w)

b = b_0^(1/K);
a = [1, -2*r*cos(w_0), r^2];

H = freqz(b, a, w);

% Plot
figure;
plot(w/pi, K*10*log10(abs(H)));

end
```

A.2 Echo cancellation

```
%% Applied digital signal processing - Homework 1

% Question 3 - Echo cancellation
%
```

```
% University of Liege
% Academic year 2018-2019
%
% Authors:
%    - Quentin Graillet
%    - Maxime Meurisse
%    - Adrien Schoffeniels

%% Parameters

alpha = 0.6;

%% Variables

audiofile = 'hw1_echo.wav';

%% Autocorrelation

[x, fs] = audioread(audiofile);
[acorrX, lagX] = xcorr(x(:, 1), x(:, 1));

figure;
plot(lagX, acorrX);

%% Echo cancellation

d = 2.205e4; % value obtained by looking on the plot
tau = d/fs; % delay in seconds

b = 1;
a = [1, zeros(1, d-1), alpha];

y = filter(b, a, x);
sound(y, fs);
```