POLITECNICO DI TORINO

Final Project report

Andrea Senacheribbe
s224178

Bachelor degree in
Electronic and Communications Engineering
AY 2017/18

Task 1: analysis of filtered periodic signals

1.1 Description of the task

The scope of this task is to analyze the response of the IIR filter with transfer function

$$H(z) = \frac{\alpha z}{z - a} \tag{1.1}$$

where a is a parameter that can assume the values $a = [-0.9, -0.8, -0.4, 0, 0.4, 0.8, 0.9]$ and $\alpha = 1 - a$ to satisfy the condition $H(z = 1) = 1$.
To test the filter, two periodic signals must be used as input: a square wave (period 64, duty cycle 50%, mean value 0, avg power 1) and a sinusoidal wave (period 64, mean value 0, average power 1).
It is required to evaluate the output of the filter in MATLAB for both input signals (consider 8 periods) and for all values of a, using the finite difference equation corresponding to eq. (1.1).
Then the absolute value of the DFT of the input and output signals should be calculated and plotted, again for all values of a of the filter.
The output of the filter with $a = -0.8$ and $a = 0.8$ with the one-sided sinusoidal signal has to be compared with the response to a theoretical sinusoid with infinite extension.
Finally the average power of all output signals should be plotted with respect to a.

1.2 MATLAB code

```
% Final Project - Task 1
% Andrea Senacheribbe s224178

%% signals and filter definitions
clear variables
clc
close all

N=64; % period
N_p=8; % number of periods
N_tot=N*N_p; % total number of samples

n=[0:(N_tot-1)]';

% generating the square wave
DC=0.5; % duty cycle
x_sw=repmat([ones(DC*N,1); -ones((1-DC)*N,1)], N_p, 1);
    % one period of the signal is repeated N_p times

% generating the sine wave
x_sin=sqrt(2)*sin(2*pi*n/N);

% filter parameters
a=[-0.9,-0.8,-0.4,0,0.4,0.8,0.9];
alpha=1-a;

% outputs of the filter, one column for each val of a
y_sw=zeros(N_tot,length(a));
y_sin=zeros(N_tot,length(a));
```

```
30
31 for i=1:length(a) % iterating for different values of a
32     y_sw(1, i)=alpha(i)*x_sw(1);
33     y_sin(1, i)=alpha(i)*x_sin(1);
34         for j=2:N_tot
35         % computing the output using finite difference equations
36         y_sw(j, i)=alpha(i)*x_sw(j)+a(i)*y_sw(j-1, i);
37         y_sin(j, i)=alpha(i)*x_sin(j)+a(i)*y_sin(j-1, i);
38         end
39 end
40
41 %% plot output square wave
42 close all
43 for i=1:length(a)
44    figure('PaperOrientation','landscape')
45    stem(n,x_sw), hold on, grid on % plotting input
46    stem(n, y_sw(:,i)) % vs output of the filter
47    title(strcat('Input and output signals for the filter with a=', num2str(a(i))
          )), legend('input x[n]', 'output y[n]')
48    xlabel('n'), ylabel('x[n], y[n]')
49    axis([-1 N_tot -2.8 2.8]), pbaspect([2.5 1 1])
50    print('-fillpage', strcat('latex/graphics/task1/io_sw_',int2str(i)),'-dpdf')
51 end
52
53 %% plot output sin wave
54 close all
55 for i=1:length(a)
56    figure('PaperOrientation','landscape')
57    stem(n,x_sin), hold on, grid on % plotting input
58    stem(n, y_sin(:,i)) % vs output of the filter
59    title(strcat('Input and output signals for the filter with a=', num2str(a(i))
          )), legend('input x[n]', 'output y[n]')
60    xlabel('n'), ylabel('x[n], y[n]')
61    axis([-1 N_tot -1.5 1.5]), pbaspect([2.5 1 1])
62    print('-fillpage', strcat('latex/graphics/task1/io_sin_',int2str(i)),'-dpdf')
63 end
64
65 %% DFT for square wave
66 close all
67 X_sw=fft(x_sw(1:N));
68 for i=1:length(a)
69    figure('PaperOrientation','landscape')
70    stem(0:N-1,abs(X_sw)), hold on, grid on % dft of input
71    stem(0:N-1, abs(fft(y_sw(N*(N_p-1)+1:end,i))))
72         % dtf of the output (from last period)
73    title(strcat('DFT of input and output signals (a=', num2str(a(i)), ')')),
          legend('DFT of input x[n]', 'DFT of output y[n]')
74    xlabel('k'), ylabel('DFT(x[n]), DFT(y[n])')
75    axis([0 N-1 0 45]), pbaspect([2.5 1 1])
76    print('-fillpage', strcat('latex/graphics/task1/dft_sw_',int2str(i)),'-dpdf')
77 end
78
79 %% DFT for sin wave
80 close all
81 X_sin=fft(x_sin(1:N));
82
83 for i=1:length(a)
84    figure('PaperOrientation','landscape')
85    stem(0:N-1,abs(X_sin)), hold on, grid on % dft of input
86    stem(0:N-1, abs(fft(y_sin(N*(N_p-1)+1:end,i))))
```

```
        % dtf of the output (from last period)
    title(strcat('DFT of input and output signals (a=', num2str(a(i)), ')')),
        legend('DFT of input x[n]', 'DFT of output y[n]')
    xlabel('k'), ylabel('DFT(x[n]), DFT(y[n])')
    axis([0 N-1 0 50]), pbaspect([2.5 1 1])
    print('-fillpage', strcat('latex/graphics/task1/dft_sin_',int2str(i)),'-dpdf'
        )
end

%% theoretical sinusoid
close all
H_resp=((1-[-0.8; 0.8])*exp(1j*2*pi/N))./(exp(1j*2*pi/N)-[-0.8; 0.8]);
    % evaluating H(z) (for two val of a) at exp(j 2 pi / N)

% for a = -0.8
figure('PaperOrientation','landscape')
stem(n,y_sin(:,2)), hold on, grid on % plotting filter output
stem(n, abs(H_resp(1))*sqrt(2)*sin(2*pi*n/N+angle(H_resp(1))))
    % vs theoretical output from theoretical sinusoid
title('Comparison of theoretical and simulated output of the filter (a=-0.8)'),
    legend('simulated output y[n]', 'theoretical output y_{the}[n]')
xlabel('n'), ylabel('y[n], y_{the}[n]')
axis([-1 N*2 -1.5 1.5]), pbaspect([2.5 1 1])
print('-fillpage', 'latex/graphics/task1/theor_sin_2','-dpdf')

% for a = 0.8
figure('PaperOrientation','landscape')
stem(n,y_sin(:,6)), hold on, grid on % plotting filter output
stem(n, abs(H_resp(2))*sqrt(2)*sin(2*pi*n/N+angle(H_resp(2))))
    % vs theoretical output from theoretical sinusoid
title('Comparison of theoretical and simulated output of the filter (a=0.8)'),
    legend('simulated output y[n]', 'theoretical output y_{the}[n]')
xlabel('n'), ylabel('y[n], y_{the}[n]')
axis([-1 N*2 -1.5 1.5]), pbaspect([2.5 1 1])
print('-fillpage', 'latex/graphics/task1/theor_sin_6','-dpdf')

%% power
close all
avg_power_sw=zeros(1,length(a));
avg_power_sin=zeros(1,length(a));
for i=1:length(a) % iterating for different values of a
    % evaluating avg power on last period of output signals
    avg_power_sw(i)=y_sw(N*(N_p-1)+1:end,i)'*y_sw(N*(N_p-1)+1:end,i)/N;
    avg_power_sin(i)=y_sin(N*(N_p-1)+1:end,i)'*y_sin(N*(N_p-1)+1:end,i)/N;
end

figure('PaperOrientation','landscape')
stem(a, avg_power_sw), hold on, grid on
stem(a, avg_power_sin)
title('Comparison of average power for different filter outputs'), legend('
    square wave', 'sinusoid')
xticks(a), xlabel('a'), ylabel('avg power of y[n]')
pbaspect([2.5 1 1])
print('-fillpage', 'latex/graphics/task1/power','-dpdf')
```

1.3 Results and comments

1.3.1 Filter definition

To satisfy the requirements on the input signals, we use the following expressions for the square wave

$$x_{sw}[n] = \sum_{s=0}^{N_p-1} q[n - sN] \ , \qquad q[n] = \begin{cases} 1 & \text{if } 0 \leq n < N/2 \\ -1 & \text{if } N/2 \leq n < N \end{cases} \tag{1.2}$$

and for the sinusoidal wave

$$x_{sin}[n] = \sqrt{2} sin\left(\frac{2\pi}{N} n\right) \quad \text{for } 0 \leq n < NN_p \tag{1.3}$$

where $N = 64$ the period of both signal, and $N_p = 8$ is the number of periods considered.
The filter in eq. (1.1) can be written using the corresponding finite difference equation

$$y[n] = \alpha x[n] + ay[n-1] = (1-a)x[n] + ay[n-1] \tag{1.4}$$

1.3.2 Evaluation of the outputs

The expression in eq. (1.4) was coded into the script, and by iterating for all possible values of a, the filter outputs were computed for both type of inputs.
The DFT of one period of the input and output signals were also computed using the MATLAB command *fft()*. For the outputs, the last simulated period was considered, so to reduce the effect of transients.

From fig. 1.4 to fig. 1.10 (7 figures) the responses to the square wave input are shown with their DFT, while from fig. 1.11 to fig. 1.17 (7 figures) the one corresponding to the sinusoidal input are shown.

For a negative, the square wave signal is highly distorted, producing alternating peaks. In the DFT, this corresponds to the amplification of high frequencies. For $a = 0$, the filter becomes $y[n] = x[n]$, and the output is the same as the input. For a positive, the low-pass behavior is clearly visible from the plot of y[n]. Indeed the DFT presents the attenuation of high frequencies.

For the sinusoidal input with negative values of the parameter a, we notice very little changes on the output, while for $a = 0$ the same argument done for the square wave holds: the filter tf is 1 and the output is equal to the input. For $a > 0$, we notice instead an attenuation in magnitude and a phase shift of the sinusoid.

1.3.3 Theoretical sinusoid

For evaluating analytically the response of the filter to a sinusoidal signal with infinite extension we use

$$y[n] = \sqrt{2}|H(e^{j2\pi/N})|sin(2\pi n/N + \angle H(e^{j2\pi/N})) \tag{1.5}$$

since we can recall from theory that the sinusoid can be written as sum of two complex exponential, eigenfunctions of the filter. Using MATLAB to compute eq. (1.5) for $a = -0.8$ and $a = 0.8$, the results are plotted in figs. 1.1 and 1.2 (only first 2 periods), together with the filter response to a one-sided sinusoid. The simulated outputs differ from the theoretical ones just for the first half period, when the transients are not extinguished yet.

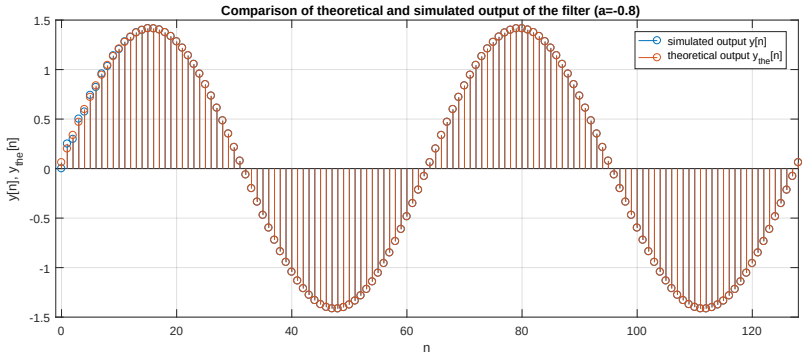

Figure 1.1: Comparison of theoretical and simulated output, sinusoidal wave input, $a = -0.8$

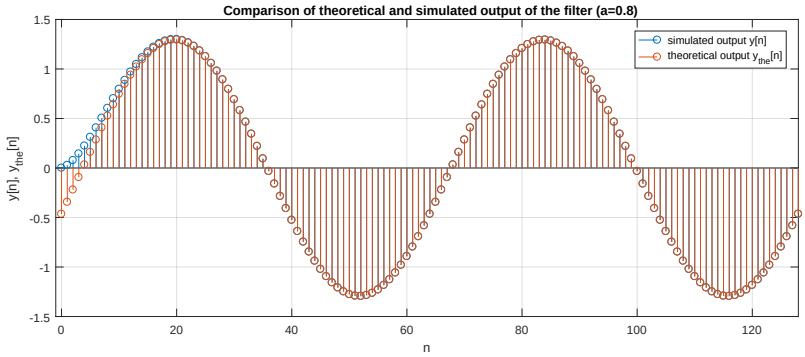

Figure 1.2: Comparison of theoretical and simulated output, sinusoidal wave input, $a = 0.8$

1.3.4 Average power

To measure the average power, only the last simulated period of the outputs was considered, as done previously with the DFT. Since the signals are periodic, it is sufficient to measure the energy of one period and divide by the period itself

$$\mathcal{P} = \frac{\mathcal{E}(y_t)}{N} = \frac{y_t^\dagger y_t}{N} \tag{1.6}$$

where y_t is the signal truncated on the last period considered in the simulation. The results for the energies are shown in fig. 1.3. It can be noticed that for the square wave input with negative a, we have higher average power (amplification), due to presence of the previously described peaks. For both types of signal the average power of the output is attenuated for a positive, and it is equal to 1 for $a = 0$ (same as the average power of the input).

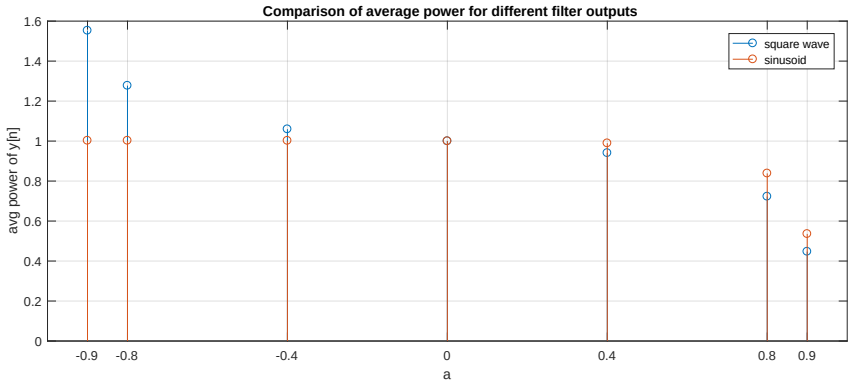

Figure 1.3: Average power of the outputs with different values of a

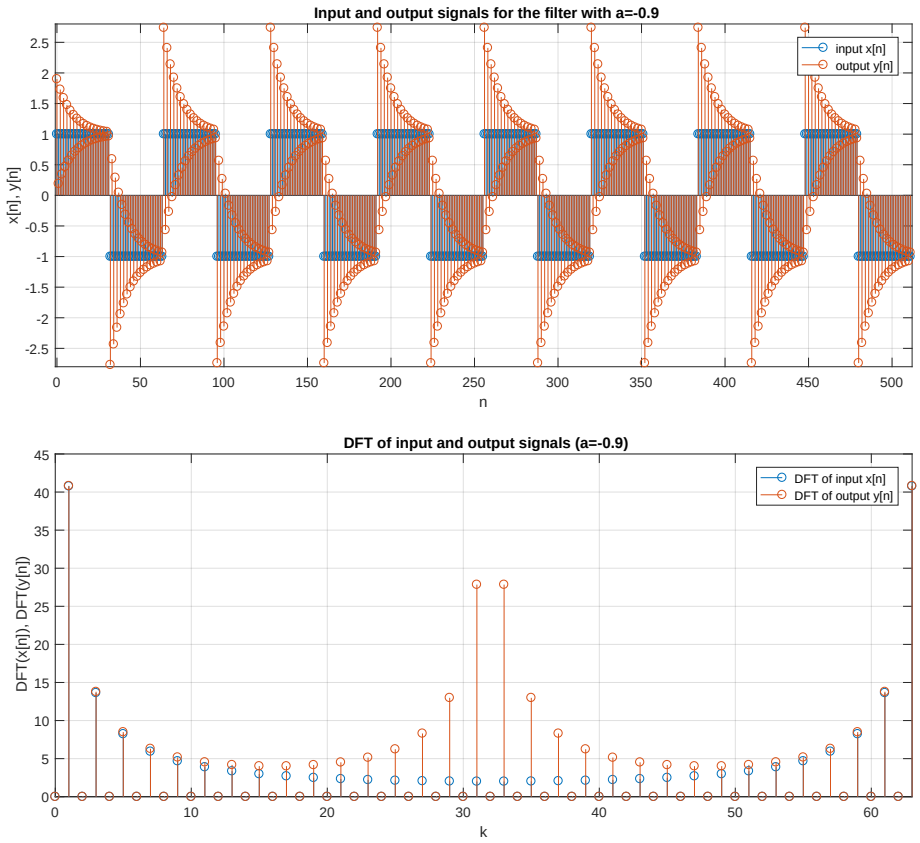

Figure 1.4: Input and output of the filter and their DFT, square wave input, $a = -0.9$ - alternating peaks on the rising edge of the square wave can be noticed, HF are amplified

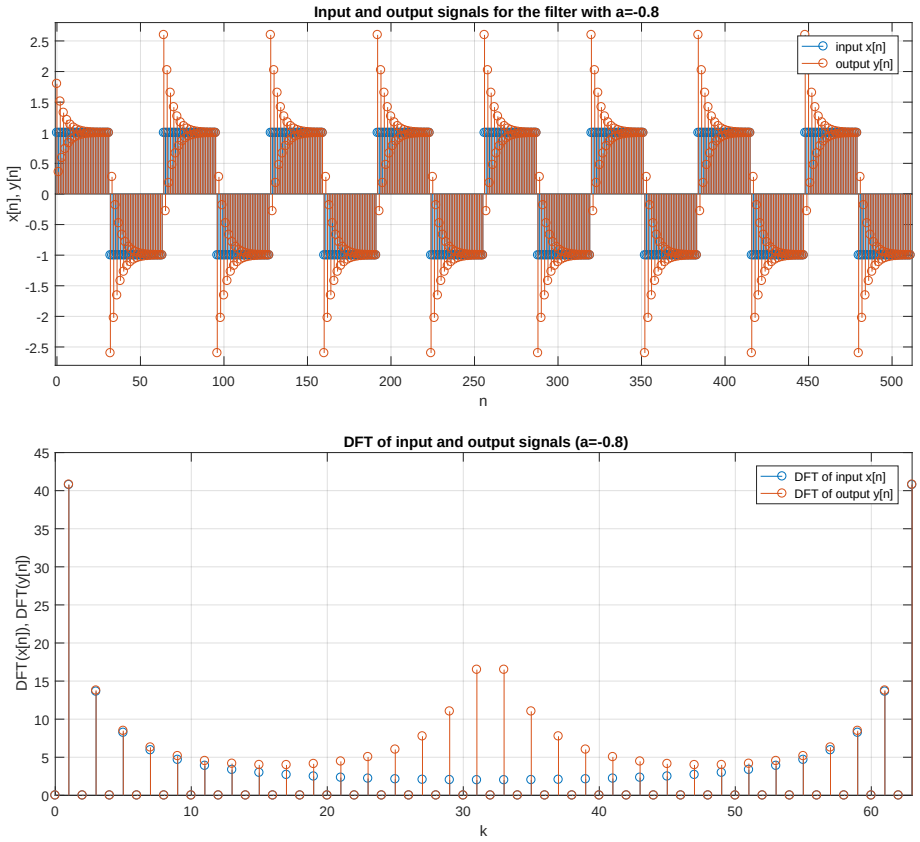

Figure 1.5: Input and output of the filter and their DFT, square wave input, $a = -0.8$

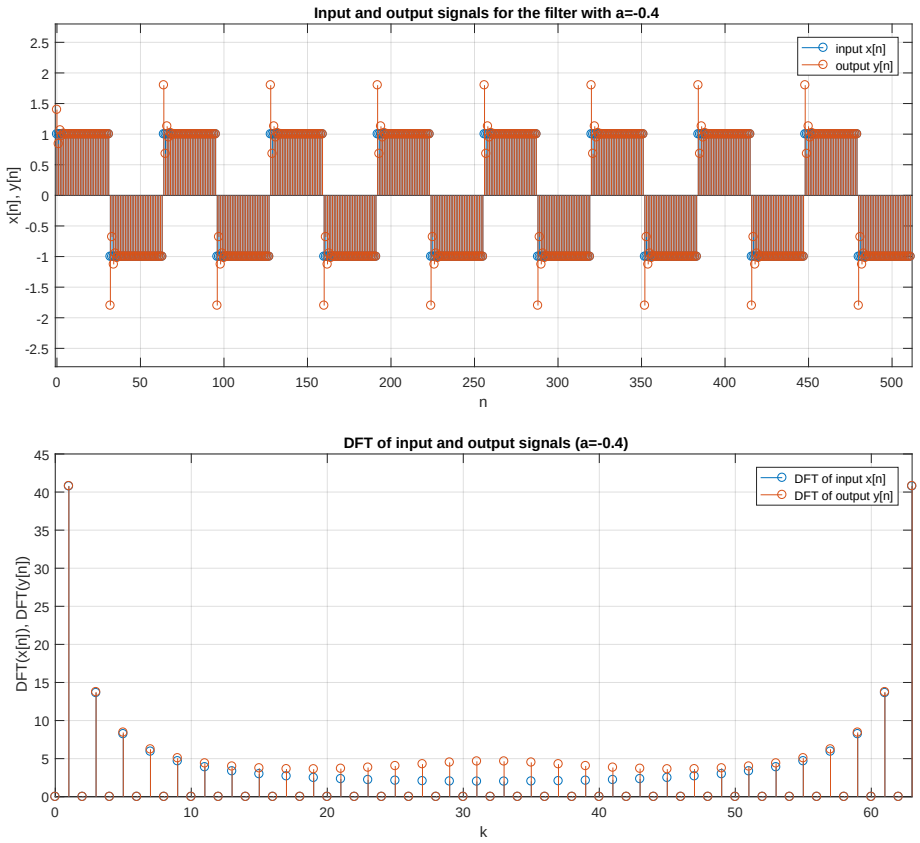

Figure 1.6: Input and output of the filter and their DFT, square wave input, $a = -0.4$

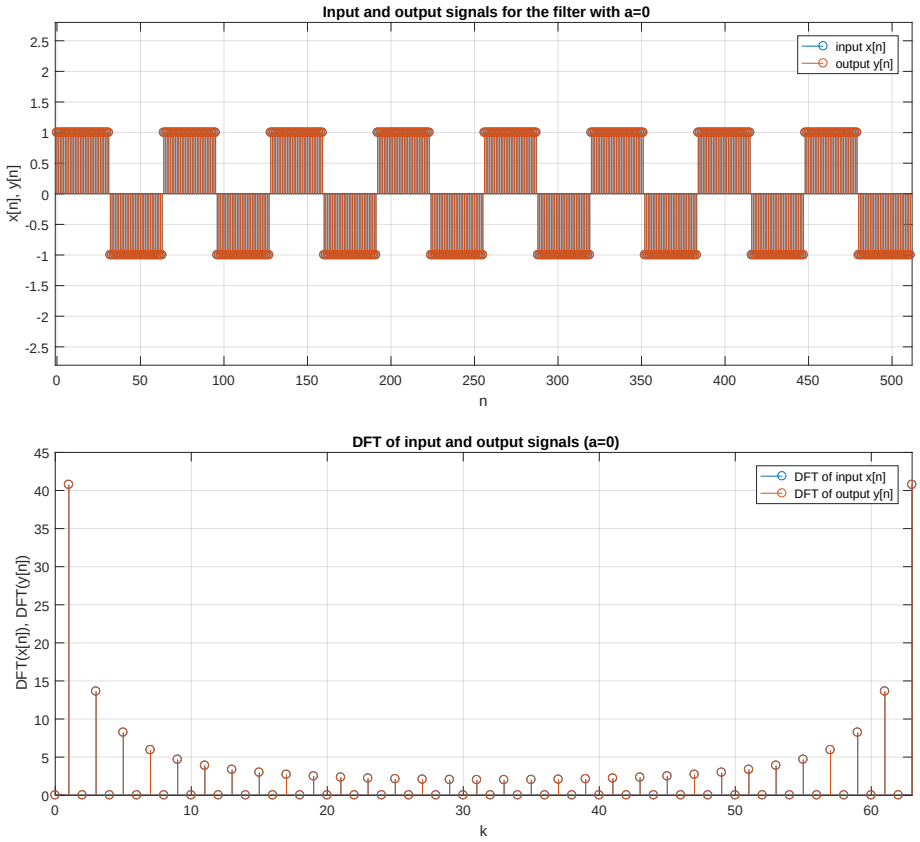

Figure 1.7: Input and output of the filter and their DFT, square wave input, $a = 0$ - the output is equal to the input, unitary tf

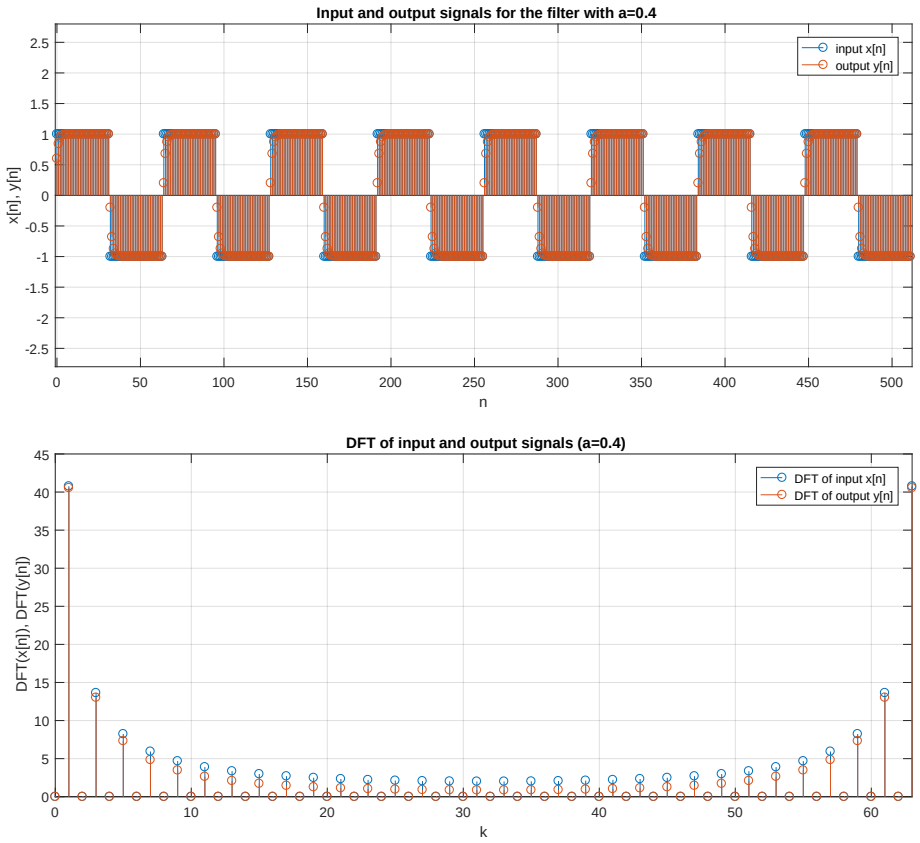

Figure 1.8: Input and output of the filter and their DFT, square wave input, $a = 0.4$

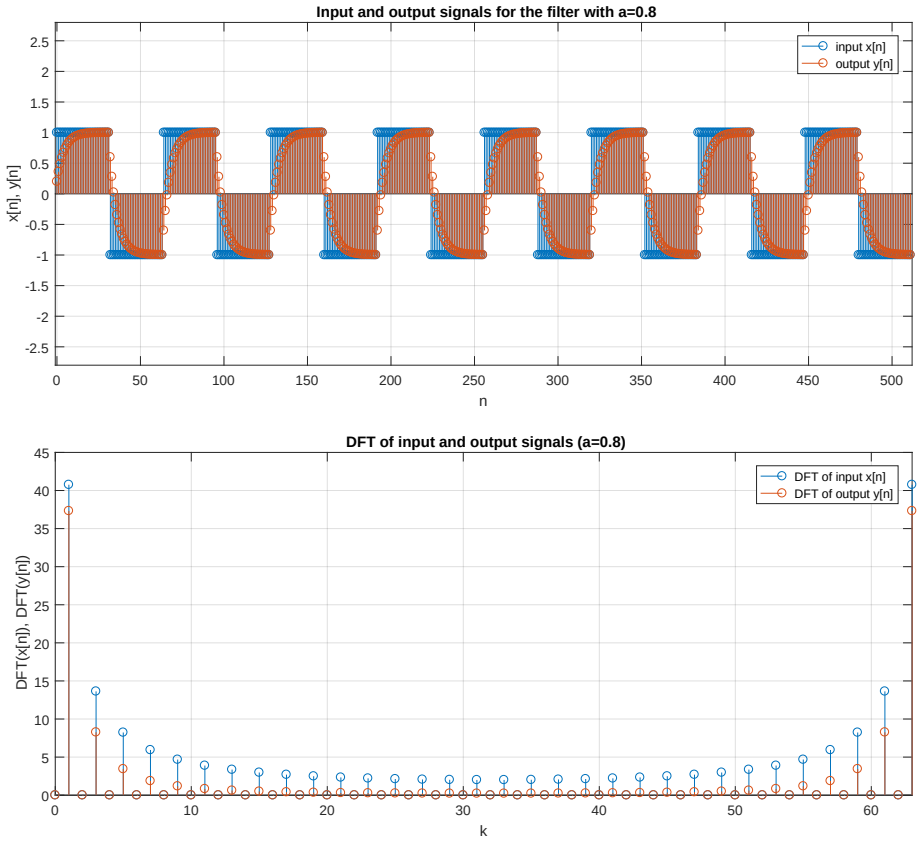

Figure 1.9: Input and output of the filter and their DFT, square wave input, $a = 0.8$ - lowpass behavior, HF are attenuated

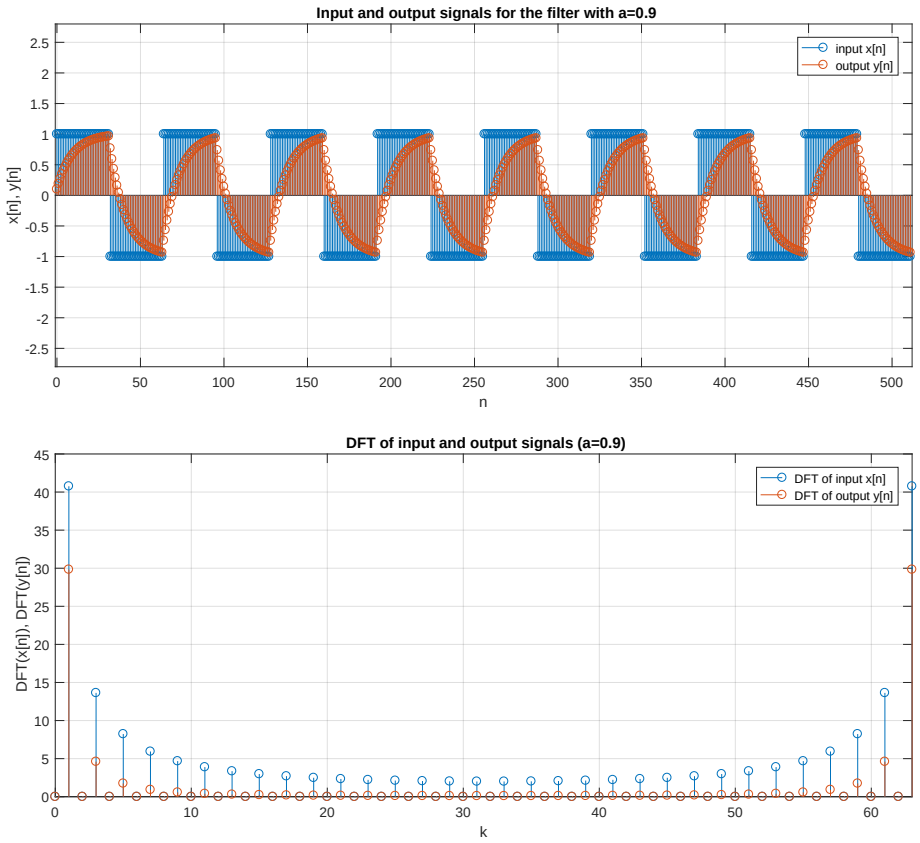

Figure 1.10: Input and output of the filter and their DFT, square wave input, $a = 0.9$

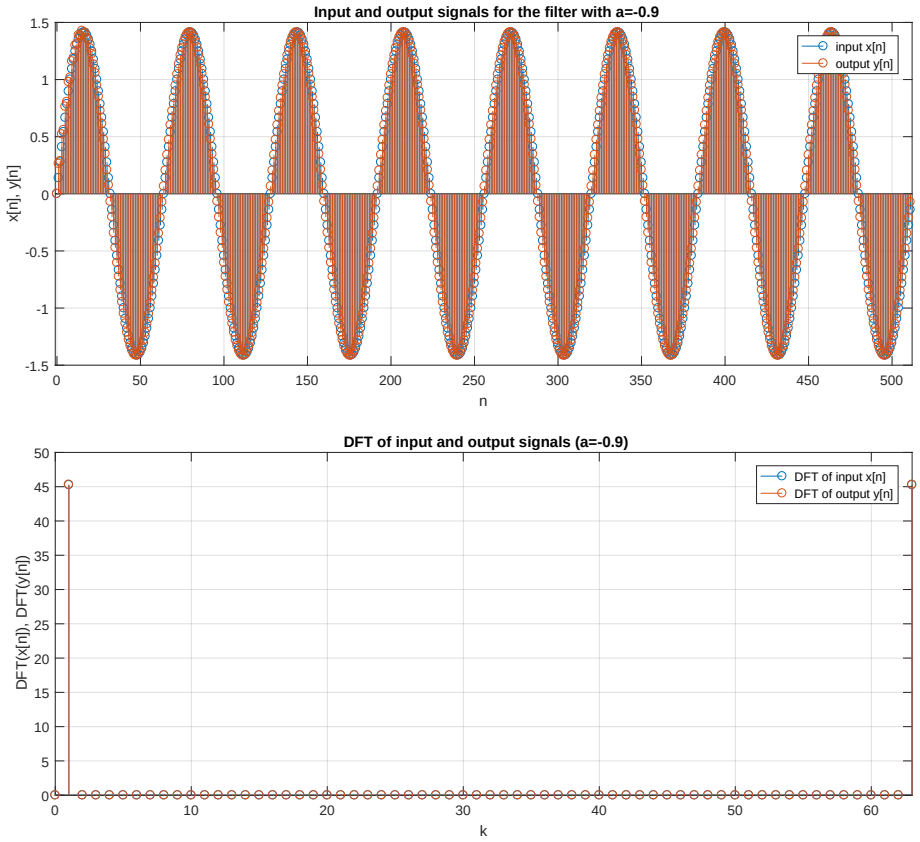

Figure 1.11: Input and output of the filter and their DFT, sinusoidal wave input, $a = -0.9$ - little differences between input and output

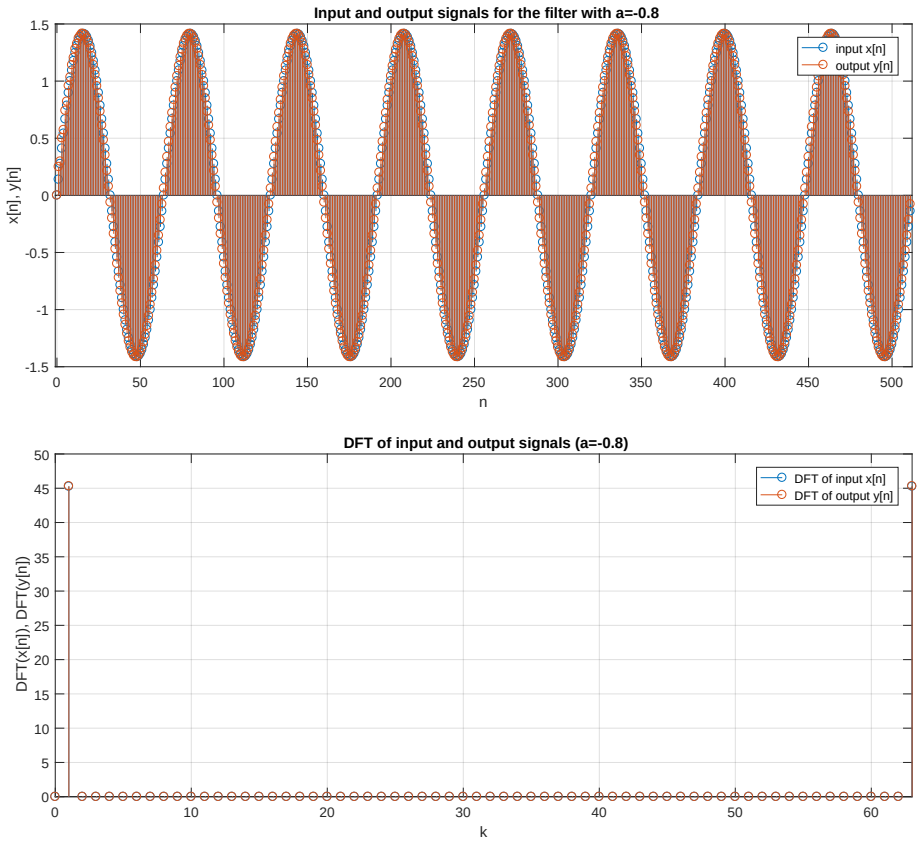

Figure 1.12: Input and output of the filter and their DFT, sinusoidal wave input, $a = -0.8$

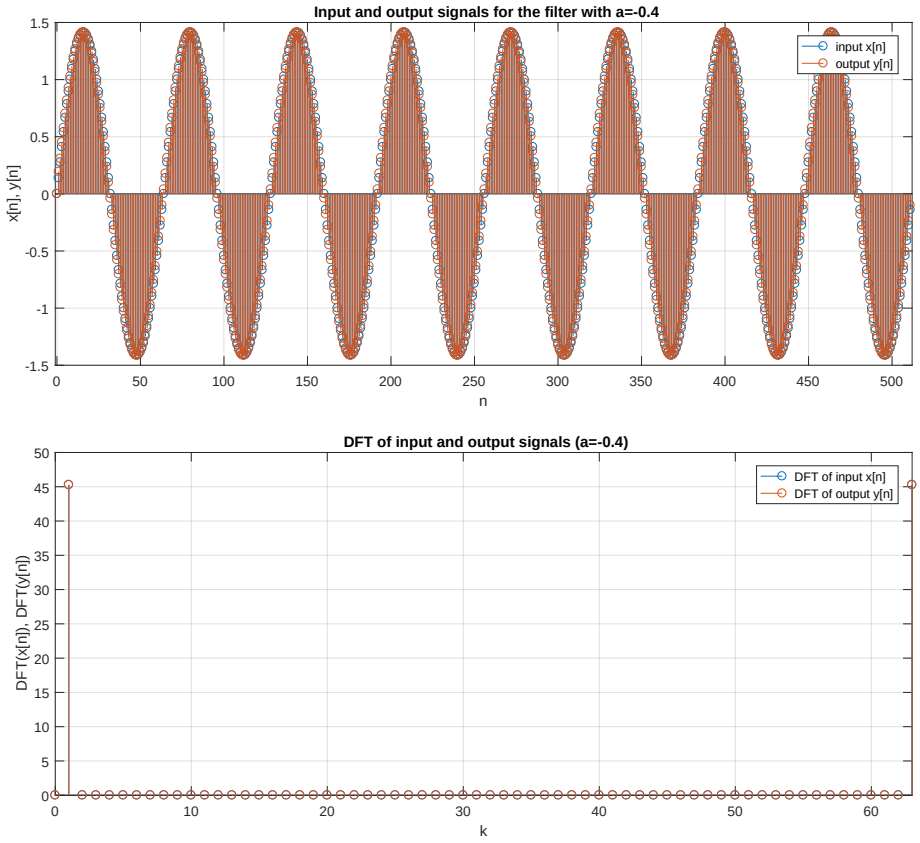

Figure 1.13: Input and output of the filter and their DFT, sinusoidal wave input, $a = -0.4$

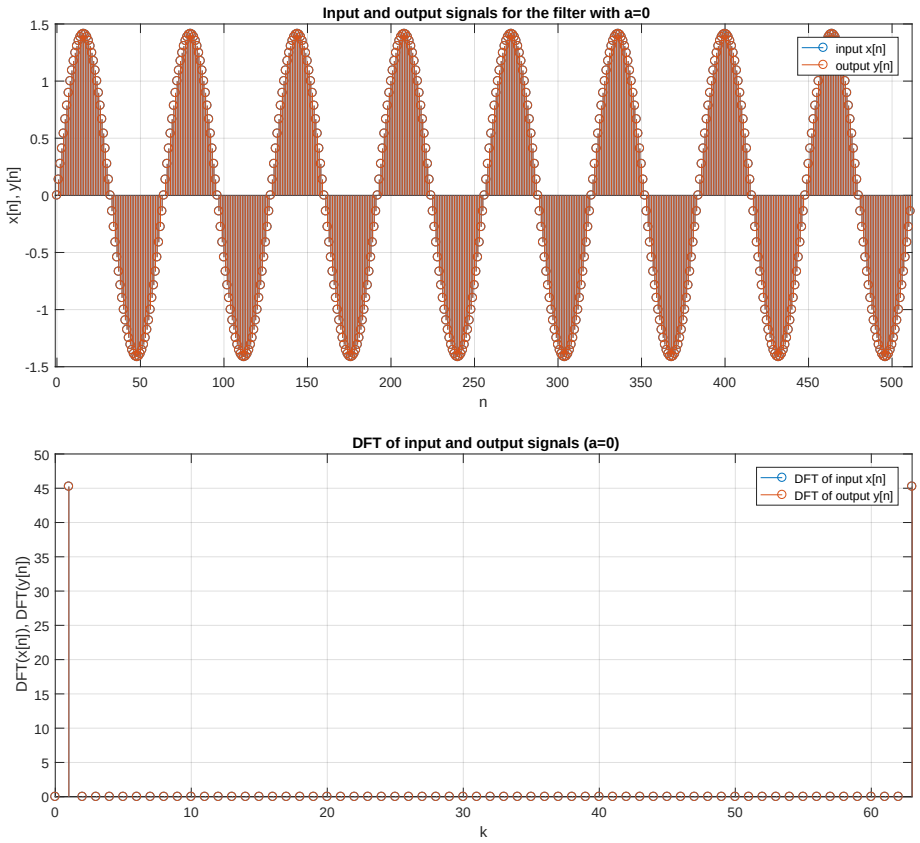

Figure 1.14: Input and output of the filter and their DFT, sinusoidal wave input, $a = 0$ - the output is equal to the input, unitary tf

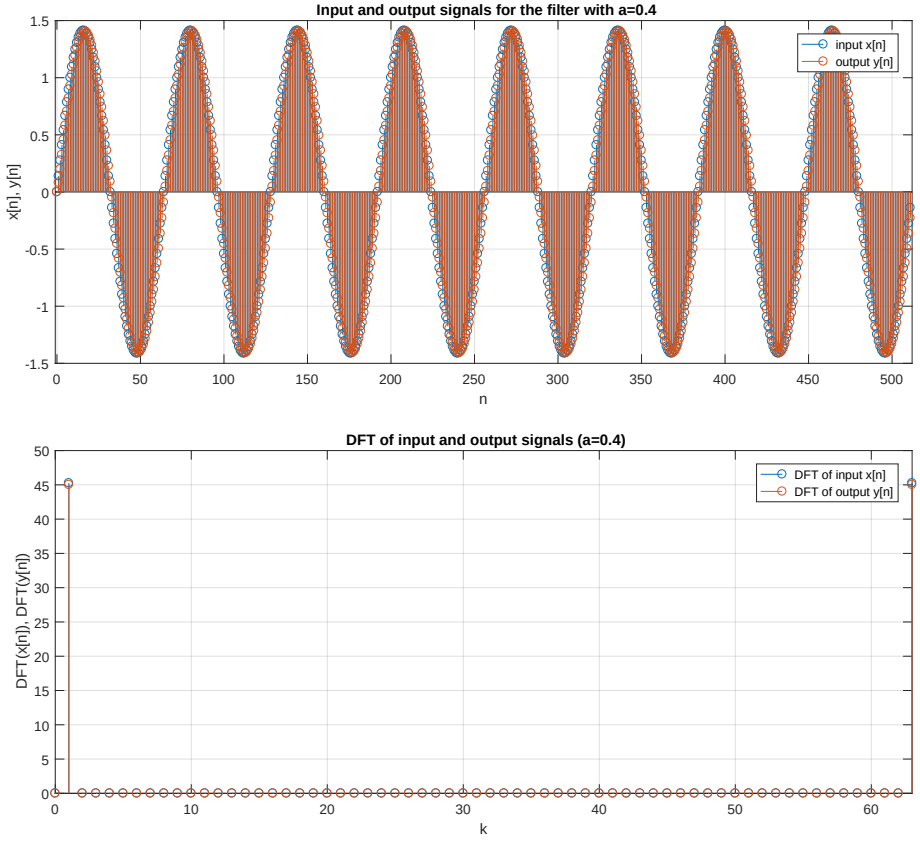

Figure 1.15: Input and output of the filter and their DFT, sinusoidal wave input, $a = 0.4$

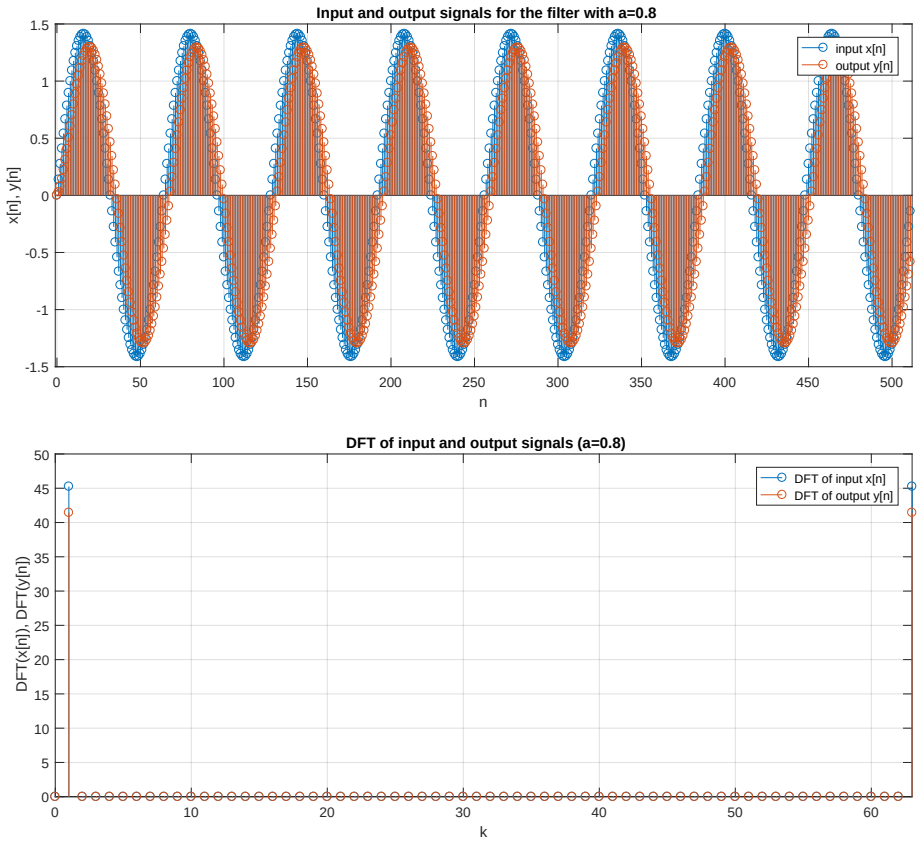

Figure 1.16: Input and output of the filter and their DFT, sinusoidal wave input, $a = 0.8$

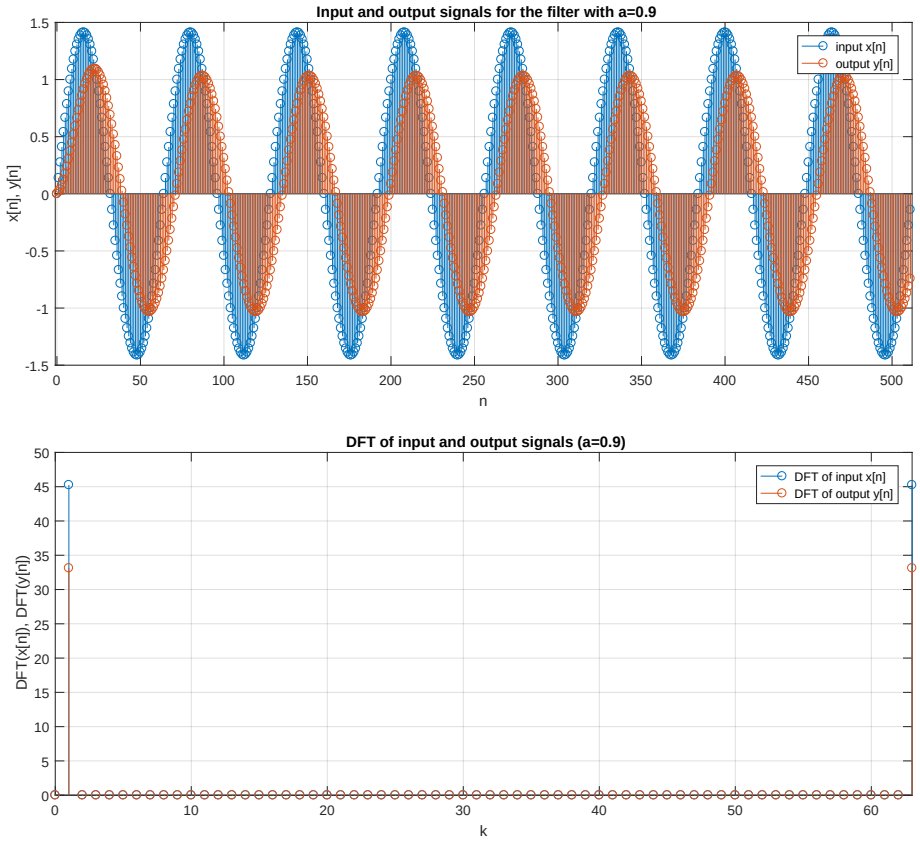

Figure 1.17: Input and output of the filter and their DFT, sinusoidal wave input, $a = 0.9$ - the sinusoid on the output is phase shifted and attenuated

Task 2: implementation of discrete-time low-pass Butterworth filter

2.1 Description of the task

The aim of this task is to implement a third order low-pass Butterworth filter in discrete time using the bilateral transformation. The analog transfer function is given by

$$H_a(s) = \frac{\omega_c^3}{(s-p_1)(s-p_2)(s-p_3)}, \quad p_1 = -\omega_c, p_2 = -\omega_c e^{j\pi/3}, p_2 = -\omega_c e^{-j\pi/3} \tag{2.1}$$

The derivation of the digital transfer function must be performed analytically and then the filter has to be coded into MATLAB for an angular frequency $\omega_c = 20$ (ω_c must be left as parameter).
The impulse response must be evaluated analytically for the analog filter and compared with the one of the discrete-time filter. The frequency behavior of the digital implementation should then be compared using the DTFT and FFT against the frequency response of the analog equivalent. Finally the energy of the impulse response of the filter must be computed both analytically from the analog tf and using the discrete-time impulse response of the filter (in time and frequency domain).

2.2 MATLAB code

```
% Final Project - Task 2
% Andrea Senacheribbe s224178

clear variables
clc
close all

% signals and filter definitions

% params of the filter
wc=20;
dt=0.005;

t_max=1;
N_tot=t_max/dt; %number of samples
t=(0:(N_tot-1))*dt;

ha= wc*exp(-wc*t)+2*wc*(sqrt(3)/3)*exp(-wc*t/2).*cos(-wc*(sqrt(3)/2)*t+5*pi/6);

%defining the input signal
x=zeros(N_tot,1);
x(1)=1; %discrete delta

y=zeros(N_tot,1);
k=zeros(N_tot,1);

%constants of finite difference equation
A=(wc*dt)/(2+wc*dt);
B=(2-wc*dt)/(2+wc*dt);
C=((wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);
```

```
D=(8-2*(wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);
E=(-4+2*wc*dt-(wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);

% computing the output of H1
k(1)=A*x(1);
for i=2:N_tot
    k(i)=A*(x(i)+x(i-1))+B*k(i-1);
end

% computing the output of H2
y(1)=C*k(1);
y(2)=C*(k(2)+2*k(1))+D*y(1);
for i=3:N_tot
     y(i)=C*(k(i)+2*k(i-1)+k(i-2))+D*y(i-1)+E*y(i-2);
end

figure('PaperOrientation','landscape')
stem(t, y/dt), hold on, grid on %plotting discrete impulse response
plot(t,ha) % vs continuos
title('Impulse response of the filter'), legend('discrete', 'continuous')
xlabel('t (sec)'), ylabel('h[t/\Deltat], h_a(t)')
print('latex/graphics/task2/impulse_resp','-dpdf')


% frequency response

%defining discrete transfer function
H1 = @(z) (wc*dt*(1+z.^-1))./((z.^-1)*(-2+wc*dt)+2+wc*dt);
H2 = @(z) (wc*dt)^2*(1+z.^-2+2*z.^-1)./((z.^-2)*(4+(wc*dt)^2-2*wc*dt)+(z.^-1)
    *(-8+2*(wc*dt)^2)+4+(wc*dt)^2+2*wc*dt);

Haf= @(f) (wc^6)./(wc^6+(2*pi*f).^6); %square modulus of frequency response

df=0.01;
f=-(1/(dt)):df:(1/(dt));
dtft=abs(H1(exp(1j*2*pi*f*dt)).*H2(exp(1j*2*pi*f*dt)));

figure('PaperOrientation','landscape')
plot(f, dtft), hold on, grid on
plot(f, sqrt(Haf(f)))
title('Modulus of frequency response'), legend('DTFT', 'analog')
xlabel('f (Hz)'), ylabel('|DTFT[f]|, |H_a(f)|')
axis([-Inf, Inf, -Inf, 1.05])
print('latex/graphics/task2/frequency_resp_lin','-dpdf')


figure('PaperOrientation','landscape')
semilogx(f, 20*log10(dtft)), hold on, grid on
semilogx(f, 10*log10(Haf(f)))
title('Modulus of frequency response (logaritmic scale)'), legend('DTFT', '
    analog')
xlabel('log_{10}(f/f0)'), ylabel('20log_{10}|DTFT[f]|, 20log_{10}|H_a(f)|')
axis([-Inf, 1/(2*dt), 10*log10(Haf(1/(2*dt))),3])
print('latex/graphics/task2/frequency_resp_log','-dpdf')

% FFT

f=(-N_tot/2:N_tot/2-1)/(N_tot*dt);
fft_y=abs(fftshift(fft(y)));

```

```
figure('PaperOrientation','landscape')
plot(f, fft_y), grid on, hold on
plot(f, sqrt(Haf(f)))
title('Modulus of FFT and analog frequency response'), legend('FFT', 'analog')
xlabel('f (Hz)'), ylabel('|FFT[f]|, |H_a(f)|')
axis([-Inf, Inf, -Inf, 1.05])
print('latex/graphics/task2/fft','-dpdf')

figure('PaperOrientation','landscape')
plot(f, 20*log10(fft_y)), grid on, hold on
plot(f, 10*log10(Haf(f)))
title('Modulus of FFT and analog frequency response (log scale on y)'), legend('FFT', 'analog')
xlabel('f (Hz)'), ylabel('20log_{10}|FFT[f]|, 20log_{10}|Ha(f)|')
axis([-Inf, Inf, 10*log10(Haf(1/(2*dt))),3])
print('latex/graphics/task2/fft_log','-dpdf')

%energy

ha2= @(t) (wc*exp(-wc*t)+2*wc*(sqrt(3)/3)*exp(-wc*t/2).*cos(-wc*(sqrt(3)/2)*t+5*pi/6)).^2;
energy_the=integral(ha2, 0, Inf)
energy_hn=y'*y/dt
energy_hk=fft_y'*fft_y
```

2.3 Results and comments

2.3.1 Derivation of the discrete-time transfer function

We can rewrite eq. (2.1) by multiplying the two complex conjugate poles and getting a real-coefficients second order term

$$H_a(s) = \frac{\omega_c^3}{(s+\omega_c)(s^2+\omega_c^2+s\omega_c)} = \frac{\omega_c}{(s+\omega_c)}\,\frac{\omega_c^2}{(s^2+\omega_c^2+s\omega_c)} = H_{a1}(s)\,H_{a2}(s) \tag{2.2}$$

From eq. (2.2), the substitution for the bilinear transformation is applied considering separately $H_{a1}(s)$ and H_{a2}.

$$\begin{aligned} H_{a1}(s) =& \frac{\omega_c}{(s+\omega_c)}, \quad s \leftarrow \frac{2}{\Delta_t}\frac{1-z^{-1}}{1+z^{-1}} \quad \Longrightarrow \\ H_1(z) =& \frac{Y(z)}{X(z)} = \frac{\omega_c\Delta_t(1+z^{-1})}{z^{-1}(-2+\omega_c\Delta_t)+2+\omega_c\Delta_t} \qquad (2.3) \\ Y(z) =& \underbrace{\frac{\omega_c\Delta_t}{2+\omega_c\Delta_t}}_{A}(X(z)+z^{-1}X(z)) + \underbrace{\frac{2-\omega_c\Delta_t}{2+\omega_c\Delta_t}}_{B} z^{-1}Y(z) \qquad (2.4) \end{aligned}$$

$$\begin{aligned} H_{a2}(s) =& \frac{\omega_c^2}{(s^2+\omega_c^2+s\omega_c)}, \quad s \leftarrow \frac{2}{\Delta_t}\frac{1-z^{-1}}{1+z^{-1}} \quad \Longrightarrow \\ H_2(z) =& \frac{Y(z)}{X(z)} = \frac{\omega_c^2\Delta_t^2(1+2z^{-1}+z^{-2})}{z^{-2}(4+\omega_c^2\Delta_t^2-2\omega_c\Delta_t)+z^{-1}(-8+2\omega_c^2\Delta_t^2)+4+\omega_c^2\Delta_t^2+2\omega_c\Delta_t} \qquad (2.5) \\ Y(z) =& \underbrace{\frac{\omega_c^2\Delta_t^2}{4+\omega_c^2\Delta_t^2+2\omega_c\Delta_t}}_{C}(1+2z^{-1}+z^{-2})X(z) + \underbrace{\frac{8-2\omega_c^2\Delta_t^2}{4+\omega_c^2\Delta_t^2+2\omega_c\Delta_t}}_{D} z^{-1}Y(z) \\ &+ \underbrace{\frac{-4-\omega_c^2\Delta_t^2+2\omega_c\Delta_t}{4+\omega_c^2\Delta_t^2+2\omega_c\Delta_t}}_{E} z^{-2}Y(z) \qquad (2.6) \end{aligned}$$

The finite difference equations corresponding to eqs. (2.4) and (2.6) are coded into MATLAB in cascade. The angular frequency ω_c is kept as parameter that can be varied to change the cutoff frequency of the filter. The block diagram of the digital filter is presented in fig. 2.1.

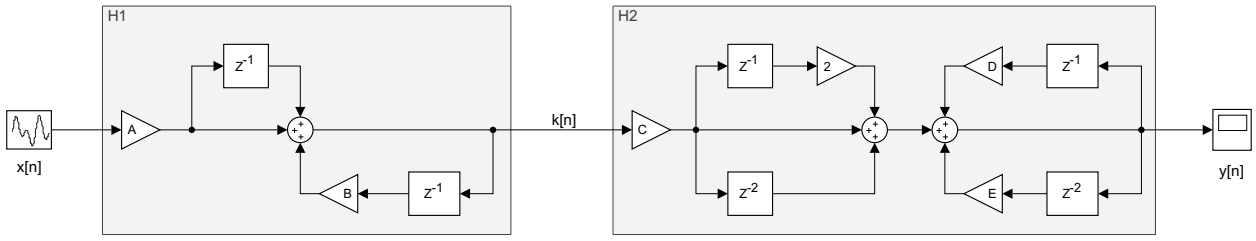

Figure 2.1: Block diagram of the filter - the constants *A, B, C, D, E* are reported in eq. (2.4) and eq. (2.6)

2.3.2 Impulse response

In order to evaluate the impulse response of the analog filter, we compute the residues of eq. (2.2), getting the following result

$$H_a(s) = \frac{\omega_c}{s+\omega_c} + \frac{\omega_c \frac{\sqrt{3}}{3} e^{j\frac{5}{6}\pi}}{s+\omega_c e^{j\frac{\pi}{3}}} + cc \implies$$
$$h_a(t) = \left(\omega_c e^{-\omega_c t} + 2\omega_c \frac{\sqrt{3}}{3} e^{-\omega_c \frac{1}{2} t} cos\left(-\frac{\sqrt{3}}{2}\omega_c t + \frac{5}{6}\pi\right)\right) u(t) \tag{2.7}$$

The impulse response of the digital filter is instead obtained by evaluating the output of the filter with a discrete delta function as input. Both the continuous (eq. (2.7)) and discrete responses are shown in fig. 2.2. By setting the filter parameter $\Delta_t = 0.005\,\mathrm{s}$, the two plots are coincident.

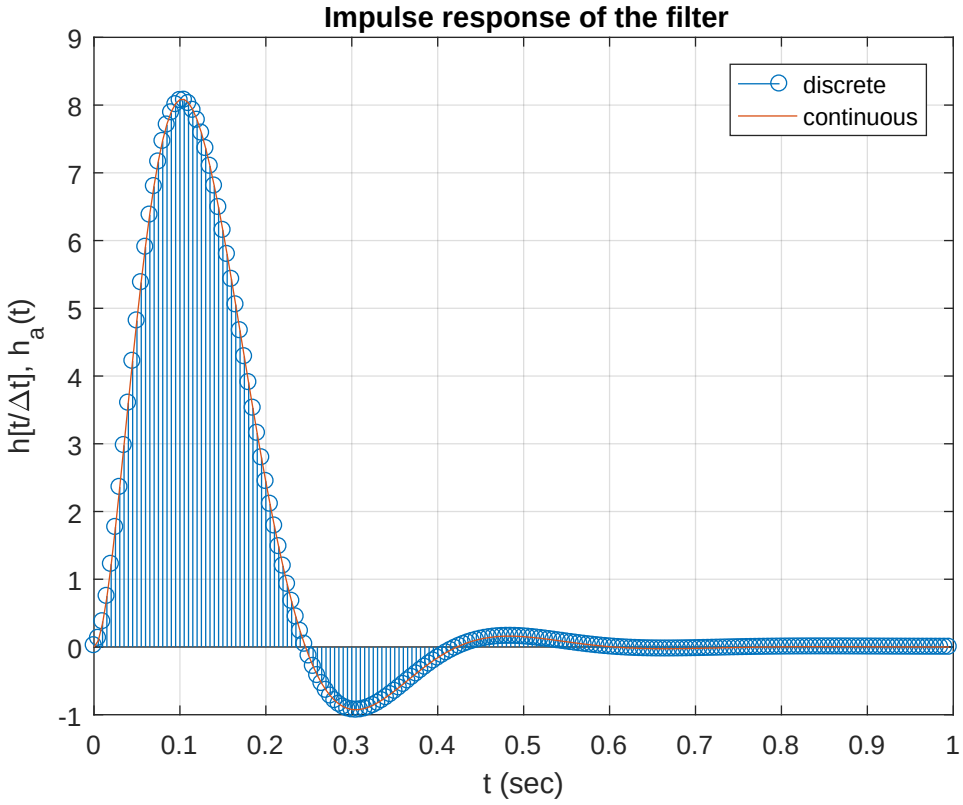

Figure 2.2: Impulse response of the filter - the analog and digital responses are coincident

2.3.3 Frequency response

The frequency response of the analog filter is given by

$$|H_a(f)|^2 = \frac{\omega_c^6}{\omega_c^6 + (2\pi f)^2} \tag{2.8}$$

To prove this result we start from eq. (2.2) and using $H_a^*(j2\pi f) = H_a(-j2\pi f)$ we get

$$\begin{aligned}
|H_a(f)|^2 &= H_a(j2\pi f)H_a^*(j2\pi f) = H_a(j2\pi f)H_a(-j2\pi f) \\
&= \frac{\omega_c^6}{(\omega_c + 2j\pi f)(\omega_c - 2j\pi f)(-4\pi^2 f^2 + \omega_c^2 - j2\pi f\omega_c)(-4\pi^2 f^2 + \omega_c^2 + j2\pi f\omega_c)} \\
&= \frac{\omega_c^6}{(\omega_c^2 + 4\pi^2 f^2)(\omega_c^4 + 16\pi^4 f^4 - 4\pi^2 f^2\omega_c^2)} \\
&= \frac{\omega_c^6}{(\omega_c^2)^3 + (4\pi^2 f^2)^3}
\end{aligned}$$

For the digital filter, the DTFT is computed numerically using MATLAB, starting from the transfer function of the filter in the z-domain (eqs. (2.3) and (2.5)) as following

$$DTFT[f] = H(e^{j2\pi f\Delta_t}) = H_1(e^{j2\pi f\Delta_t})H_2(e^{j2\pi f\Delta_t})$$

The modulus of DTFT is plotted in figs. 2.3 and 2.4 and compared with $|H_a(f)|$ (from eq. (2.8)). The graphs are coincident around frequency 0, but they differ at around $1/(2\Delta_t) = 100\,\text{Hz}$ (clearly visible on the semilog plot). This is due to frequency warping, since the continuous frequency range $[-\infty, +\infty]$ is compressed on the range $\left[-\frac{1}{2\Delta_t}, \frac{1}{2\Delta_t}\right]$ for the discrete filter. To reduce this effect, a smaller value for Δ_t can be chosen or some pre-warping techniques may be applied.

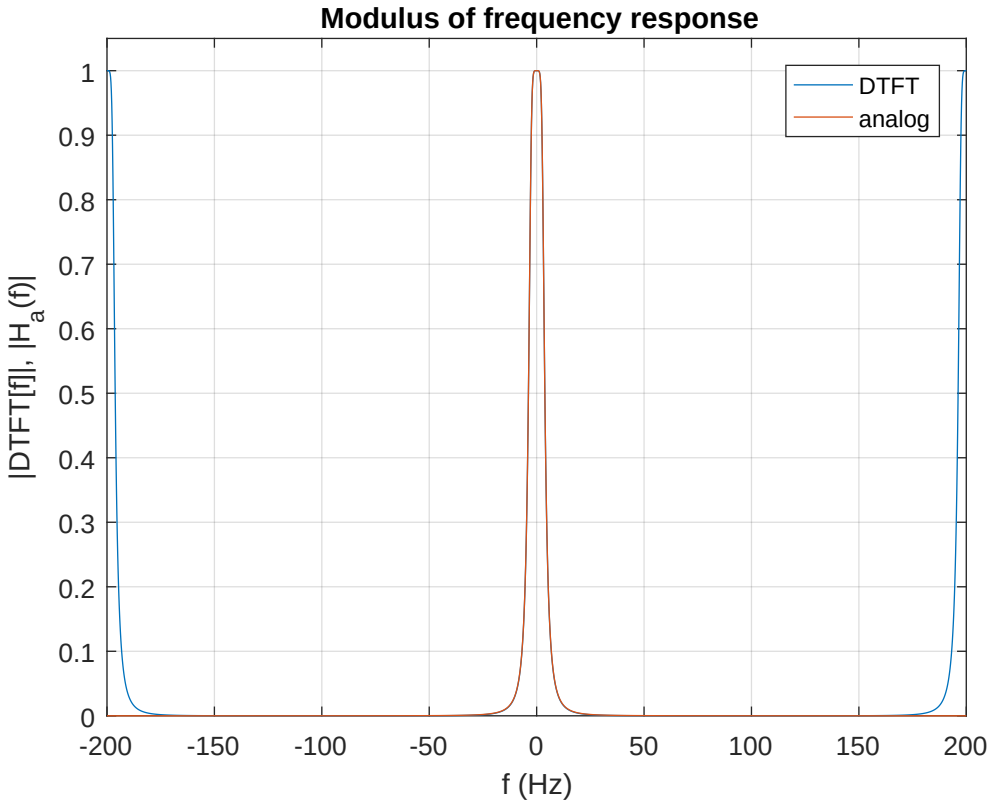

Figure 2.3: Modulus of frequency response - the periodicity of the DTFT is clearly visible

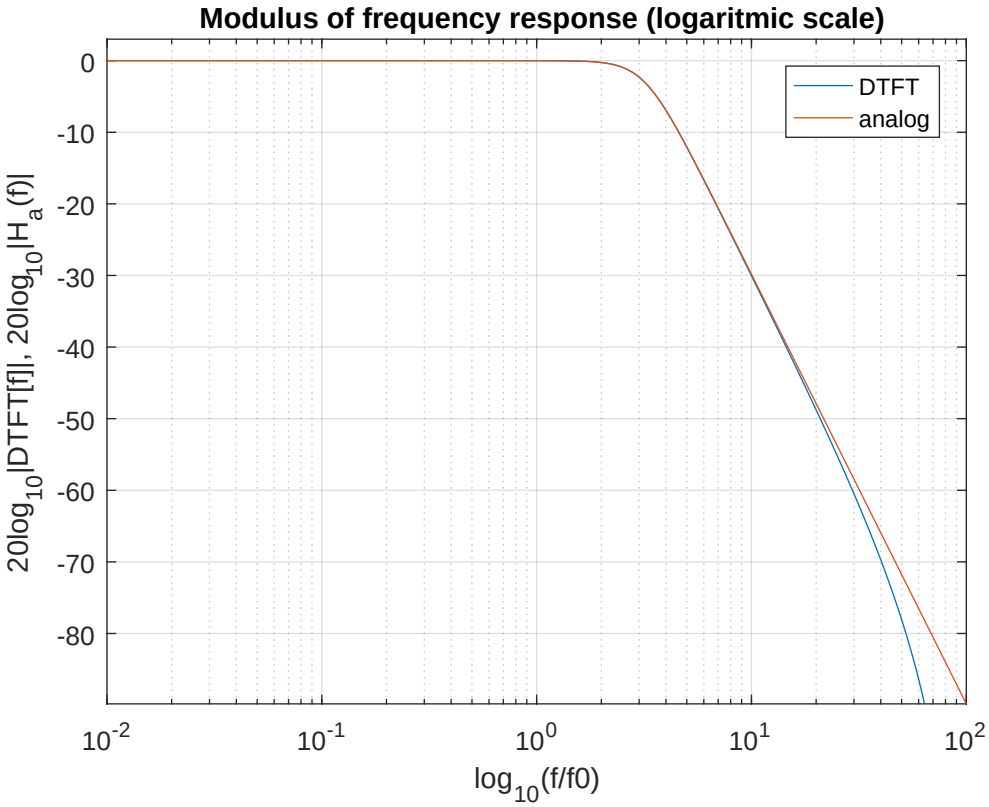

Figure 2.4: Modulus of frequency response (semi log) - the frequency warping effect can be noticed at $\approx 100\,\mathrm{Hz}$

The frequency behavior of the digital filter can also be evaluated from the FFT of the impulse response of the discrete filter, computing it using MATLAB function *fft()* and then shifting it around frequency zero. The plots in figs. 2.5 and 2.6 show again the frequency warping effect around $1/(2\Delta_t)$.

2.3.4 Energy of impulse response

The energy of the analog impulse response $h_a(t)$ was computed analytically using Wolfram|Alpha and the result confirmed using numerical integration in MATLAB:

$$\mathcal{E}(h_a(t)) = \int_0^{+\infty} |h_a(t)|^2 = \frac{20}{3} \approx 6.6667 \tag{2.9}$$

On the other hand, the energy computed from the time discrete impulse response and from its FFT is

$$\mathcal{E}(h[n]) = \mathcal{E}(FFT[k]) = 6.6584 \tag{2.10}$$

which is compatible with the result of the analog response in eq. (2.9).

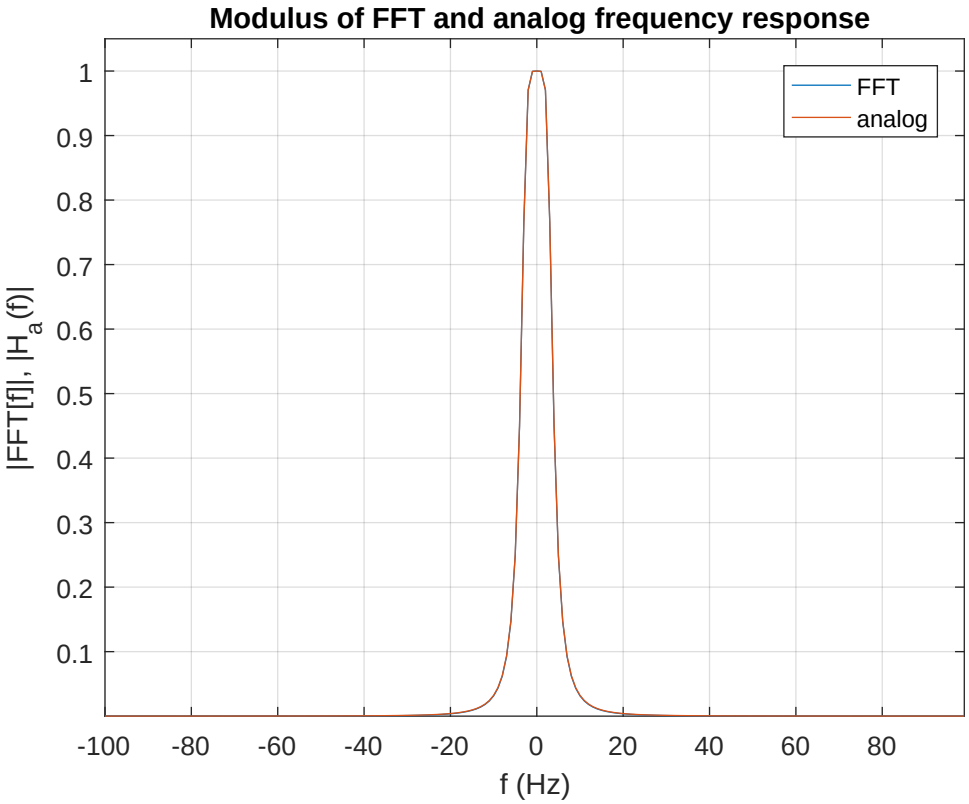

Figure 2.5: FFT of the impulse response (linear scale)

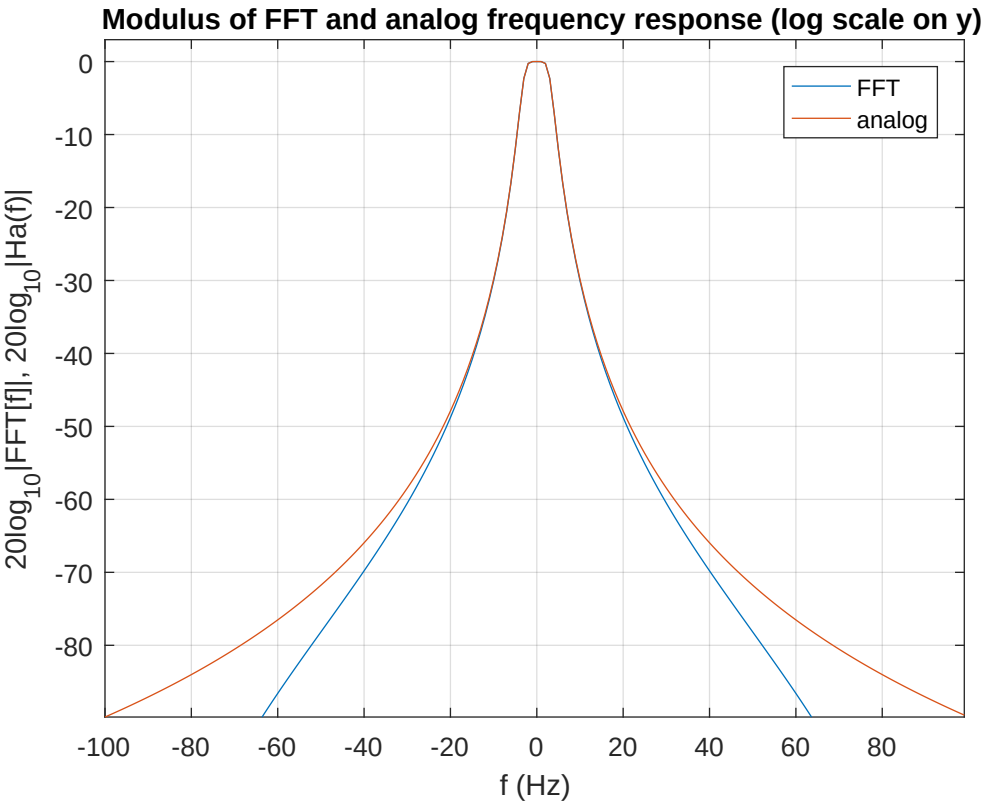

Figure 2.6: FFT of the impulse response (log scale on y) - the warping effect is again visible around 100 Hz

Task 3: filtering white Gaussian noise with discrete-time LTI filter

3.1 Description of the task

The scope of this task is to generate a white Gaussian noise signal with $G_x = \frac{N_0}{2}$ and use it as input for the low-pass Butterworth filter of Task 2.
The variance of the output signal has to be measured for 5 different realizations of the input signal (10000 samples each), using the values for $N_0 = [1, 10^{-5}, 10^{-10}, 10^{-15}, 10^{-20}]$. In the same plot, the measured variances should be compared with the theoretical variances of the response to an ideal WGN.
The histogram of the distribution for the values of one output with $N_0 = 1$ has to be presented.
Finally two realizations of the input with $N_0 = 1$ and 1000 samples should be plotted with the corresponding outputs.

3.2 MATLAB code

```
% Final Project - Task 3
% Andrea Senacheribbe s224178

clear variables
clc
close all

% params of the filter
wc=20;
dt=0.005;

N_tot=10000; %number of samples

N0=[1 1e-5 1e-10 1e-15 1e-20];
N_real=5; %number of realisation for each N0

y=zeros(N_tot,1);
k=zeros(N_tot,1);
variances=zeros(N_real,length(N0)); %variances matrix, each column corresponds
    to a value of N0, each row is a different realisation

%constants of finite difference equation
A=(wc*dt)/(2+wc*dt);
B=(2-wc*dt)/(2+wc*dt);
C=((wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);
D=(8-2*(wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);
E=(-4+2*wc*dt-(wc*dt)^2)/(4+2*wc*dt+(wc*dt)^2);

for i_n0=1:length(N0) %iterating for all values of N0
    for i_real=1:N_real %N_real times for each N0

        x=randn(N_tot, 1)*sqrt(N0(i_n0)/(2*dt)); %definition of the WGN

        %Butterworth filter
```

```
34          k(1)=A*x(1);
35          for i=2:N_tot % computing the output of H1
36              k(i)=A*(x(i)+x(i-1))+B*k(i-1);
37          end
38
39          y(1)=C*k(1);
40          y(2)=C*(k(2)+2*k(1))+D*y(1);
41          for i=3:N_tot % computing the output of H2
42               y(i)=C*(k(i)+2*k(i-1)+k(i-2))+D*y(i-1)+E*y(i-2);
43          end
44
45          variances(i_real, i_n0)=var(y); %computing variances
46
47          if (i_n0==1)&&(i_real==1)
48              %plotting the histogram for one realisation, N0=1
49              figure('PaperOrientation','landscape')
50              histogram(y, 'Normalization','probability')
51              title('Histogram of distribution of the output signal')
52              xlabel('y[n]'), ylabel('probability')
53              axis([-max(abs(y)), max(abs(y)), -Inf, Inf])
54              print('-fillpage','latex/graphics/task3/histogram','-dpdf')
55          end
56
57      end
58  end
59
60  figure('PaperOrientation','landscape')
61  loglog(kron(N0,ones(1,N_real)), variances(:), '*'), hold on %plotting the
        measured variances
62  loglog(N0, N0*10/3, 'or') %vs the theoretical ones
63  title('Variances of the output of the filter'), legend('measured', 'theoretical'
        )
64  xlabel('N_0'), ylabel('\sigma^2(y[n])')
65  print('latex/graphics/task3/variances','-dpdf')
66  axis([0.9, 1.1, 2.8 , 3.8])
67  print('latex/graphics/task3/variances_zoom','-dpdf')
68
69
70  N_tot=1000; %reducing the number of samples
71  n=0:(N_tot-1);
72
73  y=zeros(N_tot,1);
74  k=zeros(N_tot,1);
75
76  for i_real=1:2
77          x=randn(N_tot, 1)*sqrt(1/(2*dt)); %definition of the WGN
78
79          %Butterworth filter
80          k(1)=A*x(1);
81          for i=2:N_tot % computing the output of H1
82              k(i)=A*(x(i)+x(i-1))+B*k(i-1);
83          end
84
85          y(1)=C*k(1);
86          y(2)=C*(k(2)+2*k(1))+D*y(1);
87          for i=3:N_tot % computing the output of H2
88               y(i)=C*(k(i)+2*k(i-1)+k(i-2))+D*y(i-1)+E*y(i-2);
89          end
90
91          figure('PaperOrientation','landscape')
```

```
92          plot(n, x), hold on, grid on %plotting the input
93          plot(n, y) % vs output
94          title(strcat('Input and output of the filter (realisation #', int2str(
                i_real), ')')), legend('input', 'output')
95          xlabel('n'), ylabel('x[n], y[n]')
96          axis([-Inf, Inf, -35, 35]), pbaspect([2.5 1 1])
97          print('-fillpage', strcat('latex/graphics/task3/io_',int2str(i_real)),'-
                dpdf')
98  end
```

3.3 Results and comments

3.3.1 Generation of white noise

To generate a WGN we use the MATLAB command *randn()*, which outputs a vector of normally distributed samples with $\mu = 0$ and $\sigma_x^2 = 1$. To obtain the requirement of $G_x = \frac{N_0}{2}$ in the frequency range $[-\frac{1}{2\Delta_t} \frac{1}{2\Delta_t}]$ we recall that

$$P_x = \int_{-\frac{1}{2\Delta_t}}^{\frac{1}{2\Delta_t}} G_x(f)df = R_x(\tau = 0) = \sigma_x^2$$

and by solving the integral we get the desired value for σ_x^2

$$\sigma_x^2 = \frac{N_0}{2\Delta_t} \tag{3.1}$$

We can obtain the required signal by multiplying the vector generated with *randn()* by the standard deviation σ_x found in eq. (3.1).

3.3.2 Evaluation of the output

The output of the filter has been evaluated using the implementation of Task 2. Two realizations for the input with $N_0 = 1$ (1000 sample) and the corresponding outputs are plotted in figs. 3.1 and 3.2. It can be noticed that the input has higher standard deviation ($\sigma_x \approx 10$) than the output ($\sigma_x \approx 1.8$), as we will compute analytically after.
The distribution of the values of the output for a realization with $N_0 = 1$ and 10000 sample is presented in the histogram in fig. 3.3. The samples are Gaussian distributed as we can expect by filtering a Gaussian input with a LTI system.

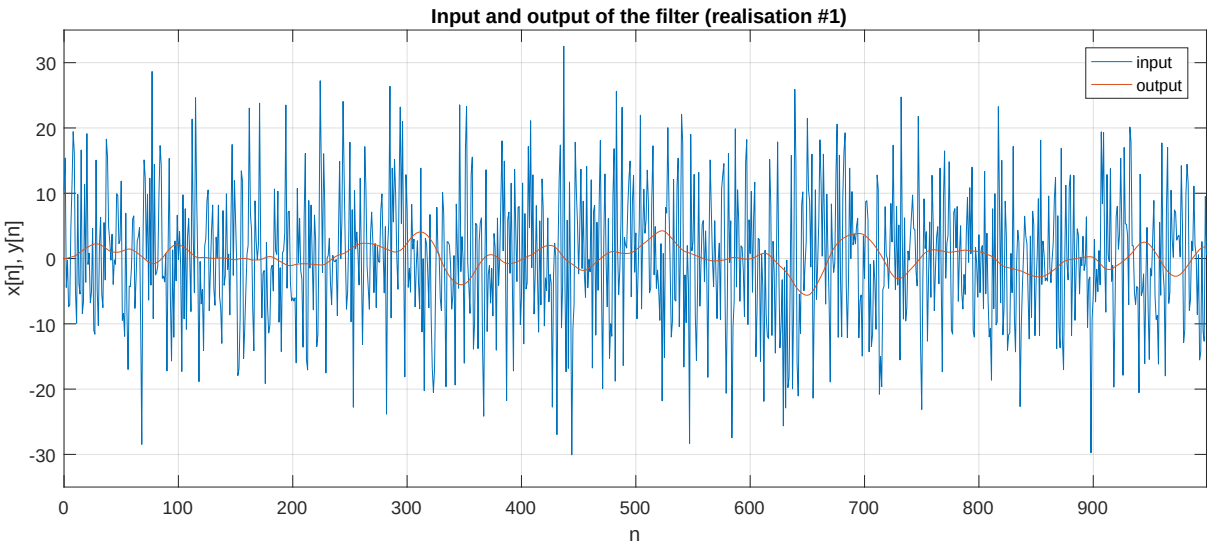

Figure 3.1: Input and output of the filter (realization #1)

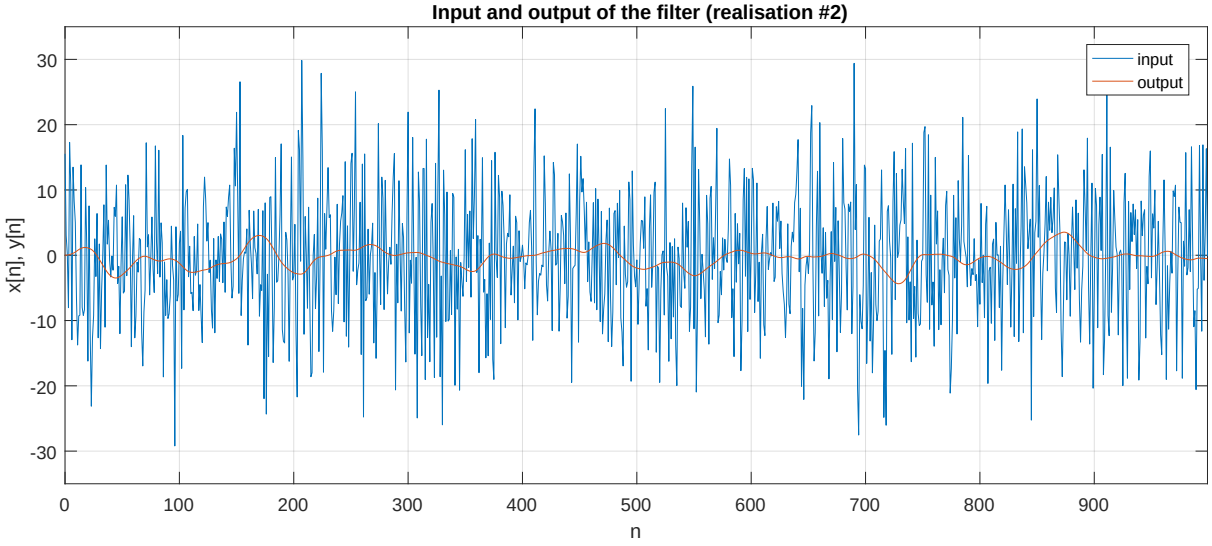

Figure 3.2: Input and output of the filter (realization #2)

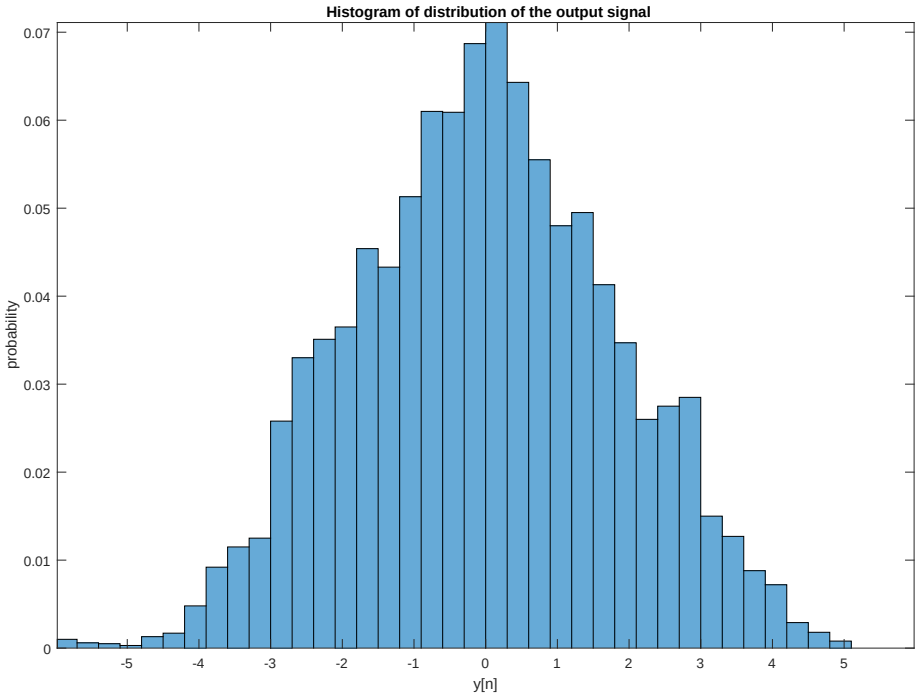

Figure 3.3: Histogram of distribution of the output signal (10000 samples, $N_0 = 1$) - the output is Gaussian distributed with $\sigma \approx 1.8$

3.3.3 Variance of the output

The mean value and the variance of the output of the filter can be derived analytically as

$$\mu_y = \cancelto{0}{\mu_x} H(0) = 0$$
$$\sigma_y^2 = E(Y^2(t)) = \frac{N_0}{2}\mathcal{E}(h(t)) = \frac{N_0}{2}\frac{20}{3} = N_0\frac{10}{3} \tag{3.2}$$

In fig. 3.4 the variances measured by evaluating the response of the filter (5 realizations for each value of N_0, 10000 samples each) are compared with the theoretical result in eq. (3.2). The result for $N_0 = 1$ is presented in detail in fig. 3.5. As we can see from both figures, the results are consistent with theory.

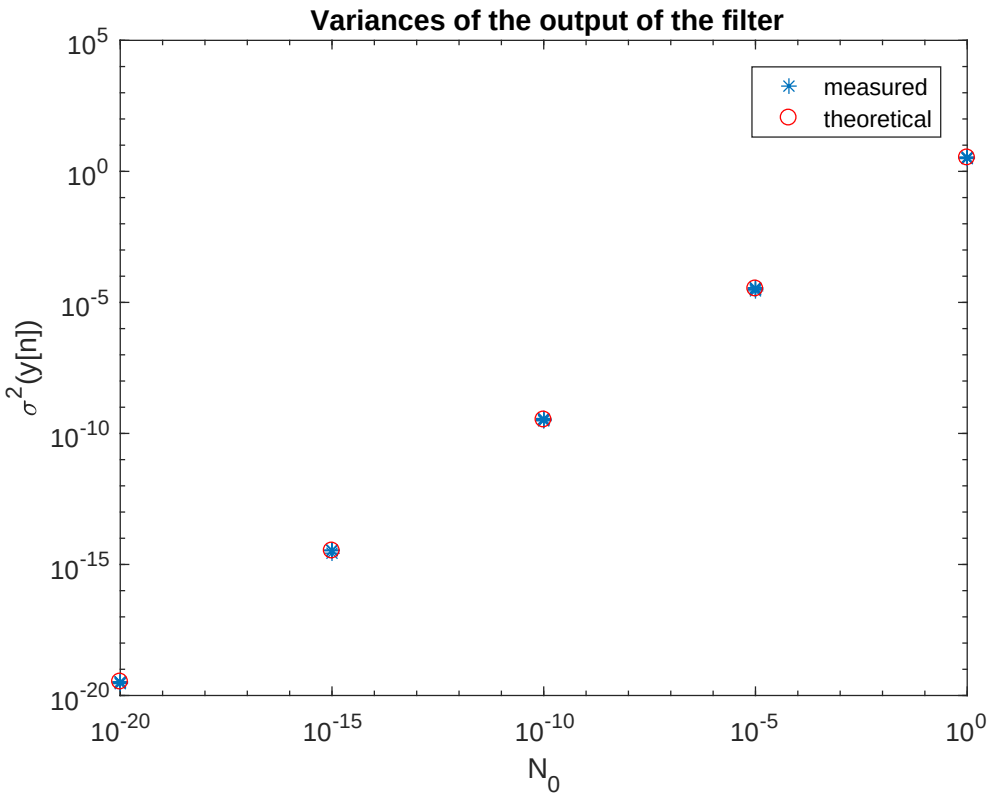

Figure 3.4: Variances of the output of the filter - the results are consistent with theory

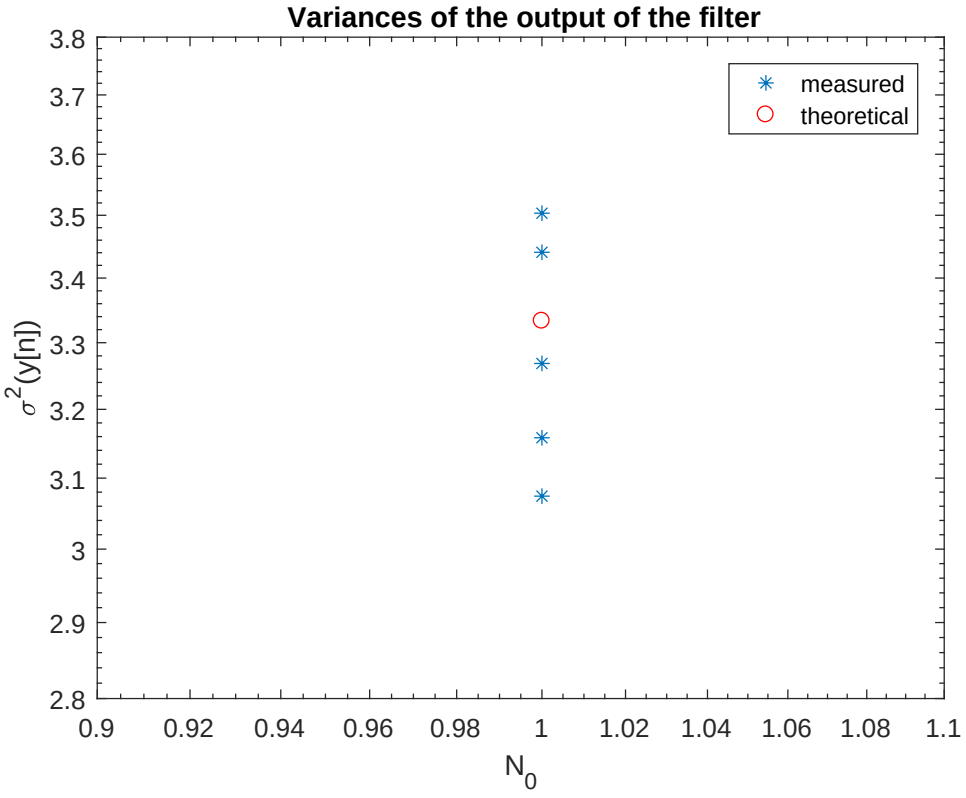

Figure 3.5: Variances of the output of the filter, zoom for $N_0 = 1$