Assignment 1

Alma Sehic s274208 – Group 6

Applied Signal Processing Laboratory 2023

Introduction

A number of exercises have been performed with the use of MATLAB as an introduction to signal processing. In the following slides a detailed analysis of each of the exercises have been shown with plots, explanations and values

List of exercises:

- Exercise 1.1 : Energy of a rectangular pulse
- Exercise 1.2 : Filtering of 3 sines on the frequency axis
- Exercise 1.3 : Quantization

Exercise 1.1 - Energy of a rectangular pulse

- Choose the integer values of the amplitude A and the duration T of a rectangular pulse
- Fix the total snapshot time: $Tmax = 10$
- Choose a very high sampling frequency, e.g. $fs = \frac{1000}{T}$
- Generate and plot the rectangular pulse on the time axis

$$s(t) = AP_T(t) \qquad 0 \leq t < Tmax$$

- Compute analytically the pulse energy.
- Compute the energy on the time axis.

Exercise 1.1 – Plot the pulse on the time axis

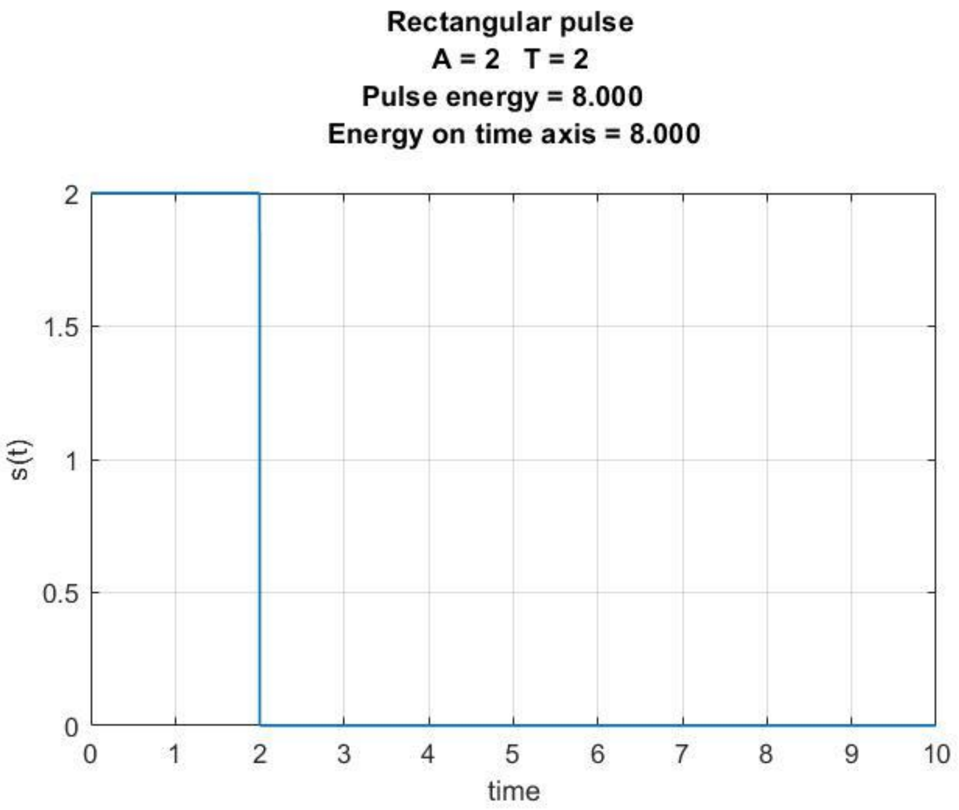

Figure 1.1: Plot of the rectangular pulse with A = 2 and T = 2, including analytical and true energy values calculated in time domain

The method used to generate the rectangular pulse is shown in the following code:

```
s = zeros(1,N_total);
s(1,1:N_pulse) = A;
```

where `N_pulse` is the number of samples of the rectangular pulse where the signal value will be equal to the amplitude `A` of the signal assigned and zero elsewhere

Energy of the signal has been calculated using the following formulae:

$$E_{rectangularPulse} = A^2 * T$$

Formula for calculating the energy of a rectangular pulse

```
e_analitical = A^2*T;
```

$$\sum_{t=-\infty}^{+\infty} |s(t)|^2$$

Formula for calculating the energy of a discrete time signal sampled on the time axis

```
e_time = sum(s.^2)*T_s;
```

where T_s is the sampling period

Exercise 1.1 – Plot the pulse on the frequency axis

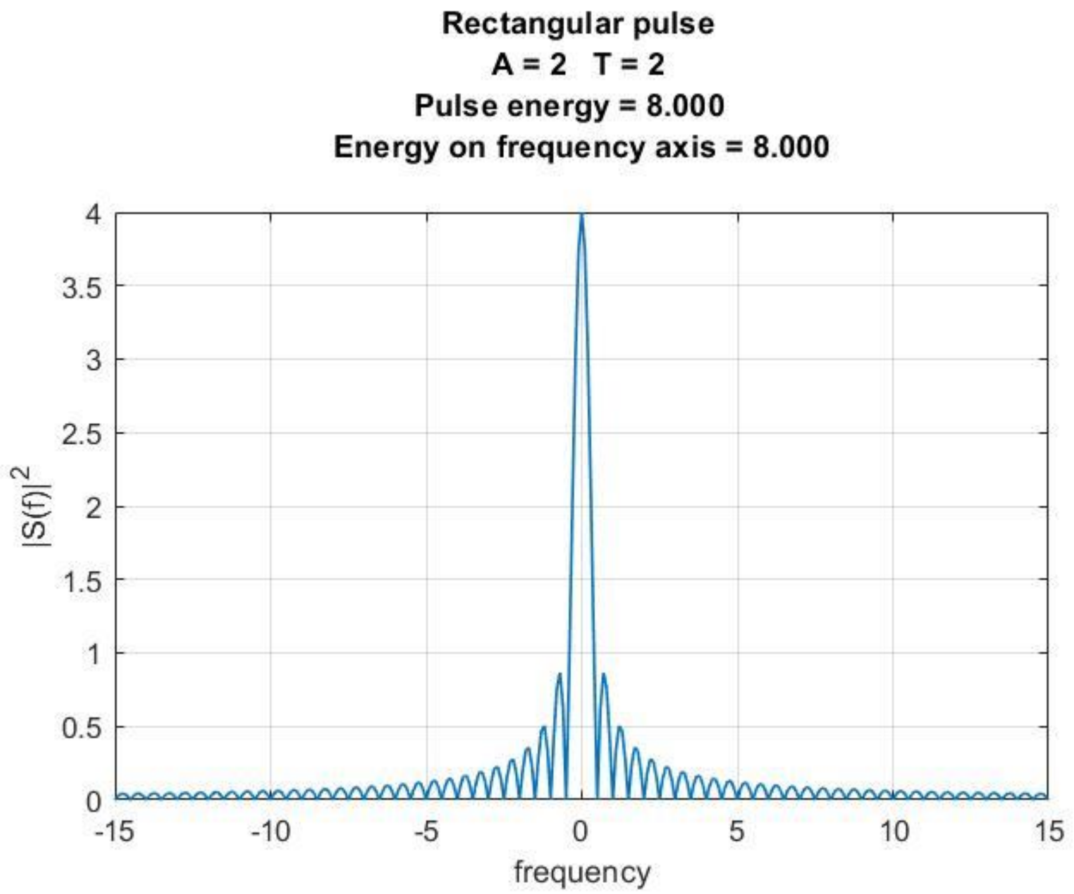

Figure 1.2: Plot of the rectangular pulse in frequency domain with A = 2 and T = 2, including analytical and true energy values calculated in frequency domain

The plot in Figure 1.2 shows the rectangular signal s(t) transformed from time domain into frequency domain using the Fourier Transform, in MATLAB implemented using the fft function as:

```
S = fft(s)*T_s;
```

Due to the Parseval's theorem, the energy of a time domain signal equals the energy of a frequency domain signal which has been shown on the plot

$$\sum_{n=-\infty}^{+\infty} |S(f)|^2$$

Formula for calculating the energy in frequency domain

```
e_frequency = sum(M.^2)*f_res;
```

where f_res is the resolution frequency

Exercise 1.1 – Examples with different values of A and T

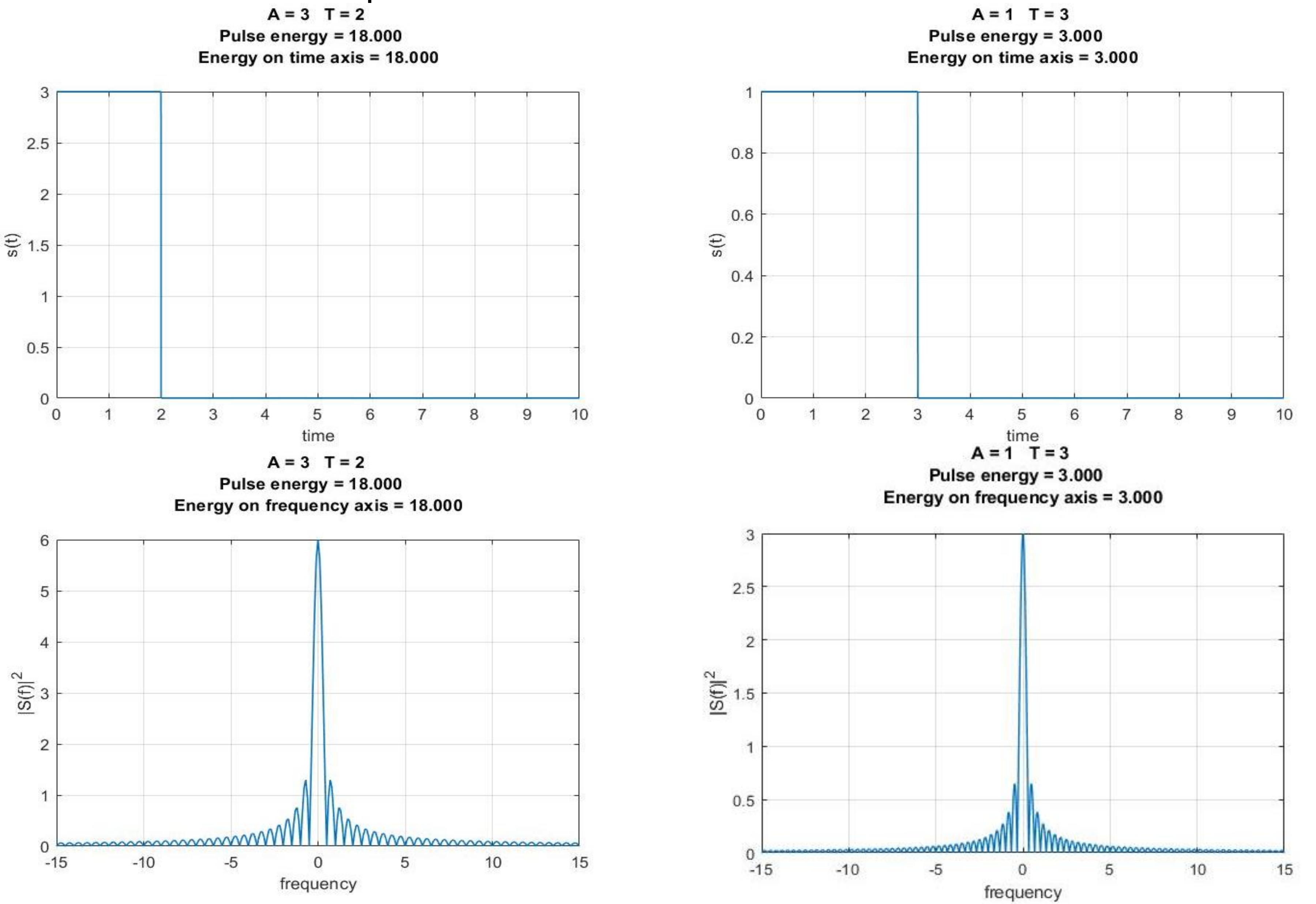

Figure 1.3: plots of the rectangular signal in time and frequency domain with different A and T values

In each of the examples, the energy calculated in the time domain is equal to the energy calculated in frequency domain

Exercise 1.1 – Percentage of energy contained in the first 10 and 100 lobes

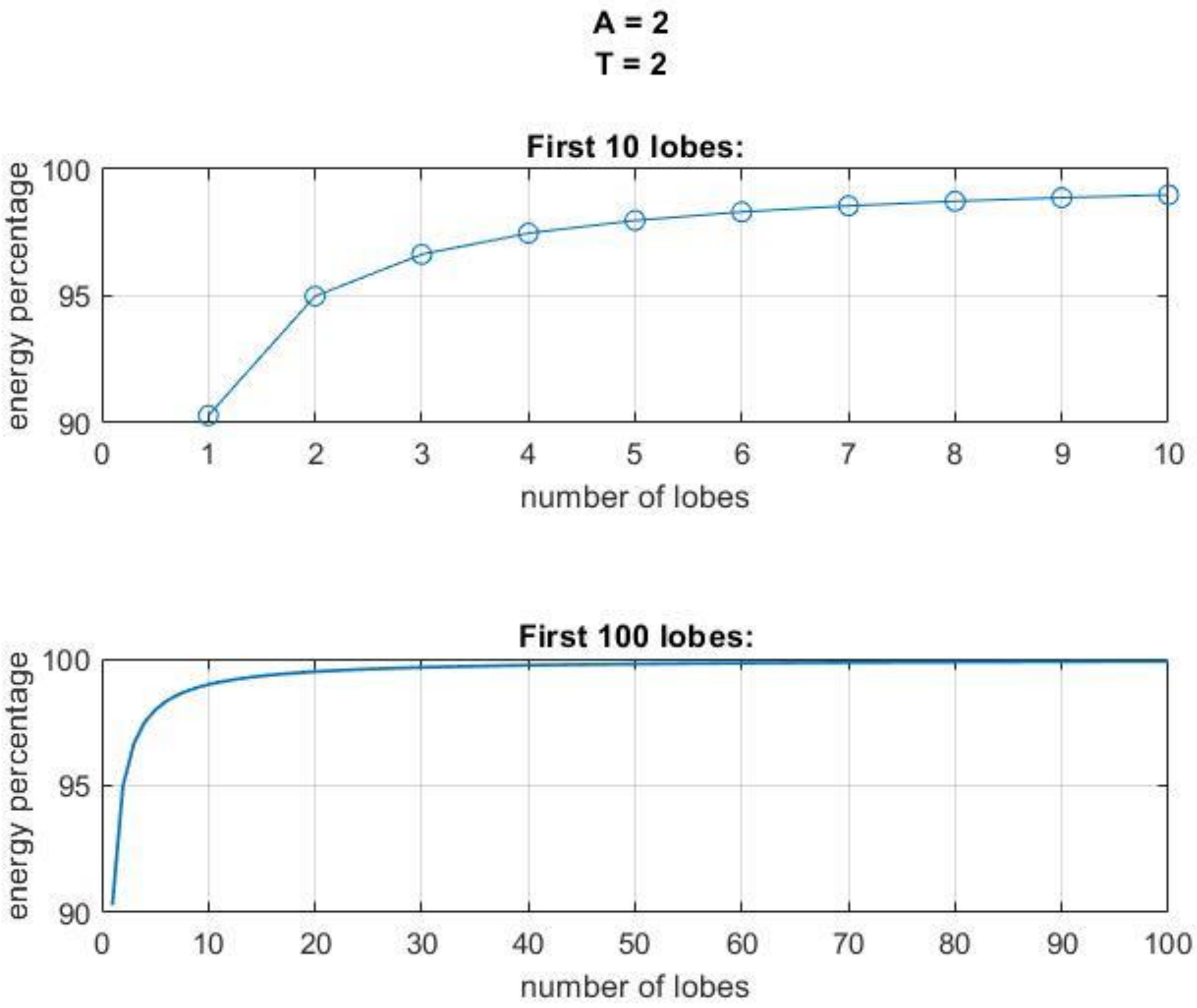

Figure 1.4: energy percentage of the first 10 and the first 100 lobes

The plots in Figure 1.4 show that approximately 90.3% of the total energy of the rectangular signal is contained in the first lobe:

Example of energy contained in first lobe wrt total energy given A=2 and T=2:

Total energy: 8

Energy in first lobe: 7.2226

In the first 10 lobes approximately 98.9% of the total energy is contained, whereas in the first 100 lobes 99.9%

Exercise 1.1 – Percentage of energy contained in the first 10 and 100 lobes – different values of A and T

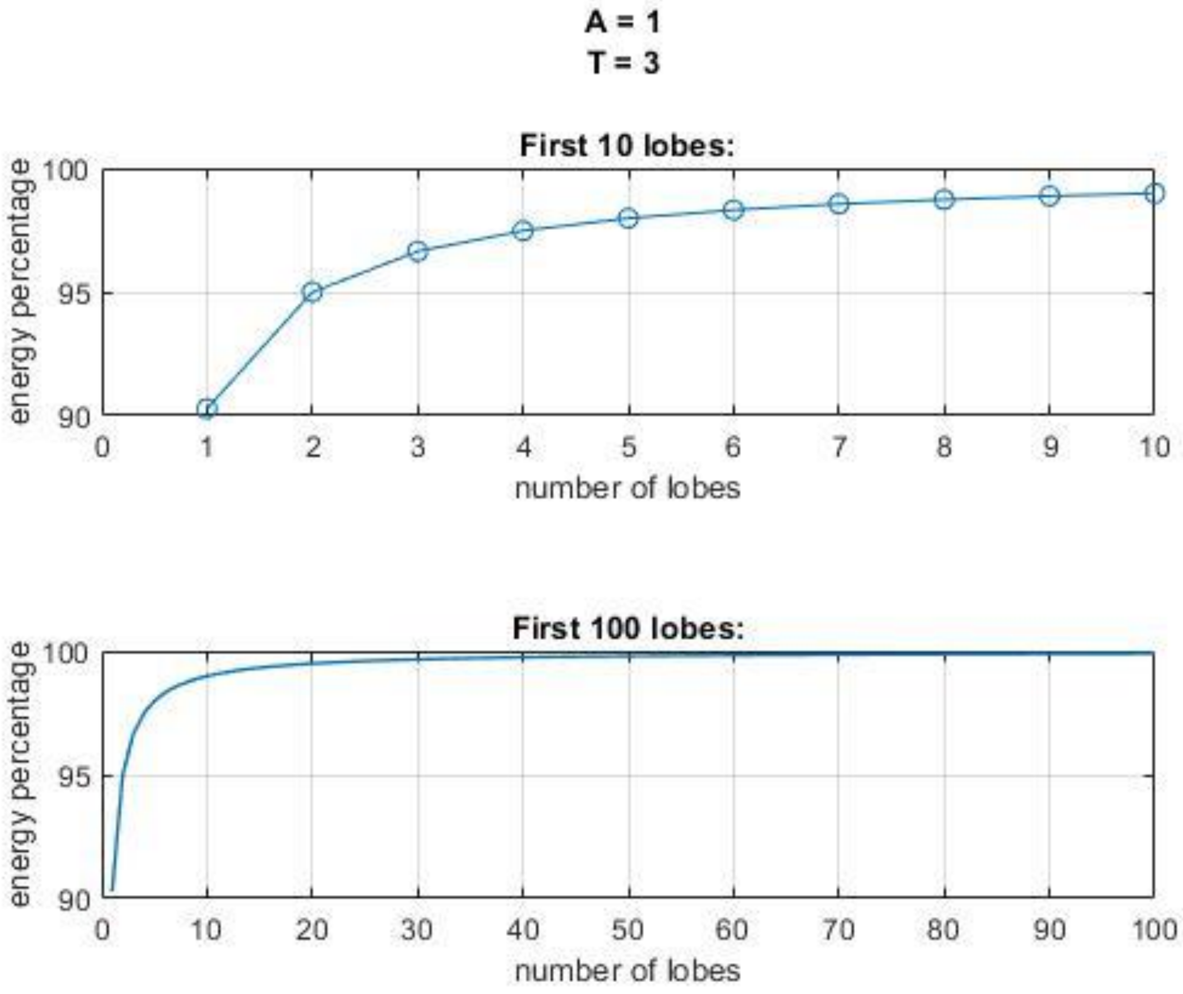

Figure 1.5: energy percentage of the first 10 and the first 100 lobes with A=1 and T=3

In the Figure 1.5 an example of the percentage of energy contained in the first 10 and 100 lobes with different A and T values: in this case A=1 and T=3 is shown.

The same trend as in the previous example is followed: regardless of the A and T values chosen, the percentage of energy in the lobes will not change!

Exercise 1.2 – Filtering of 3 sines on the frequency axis

- Consider a signal x obtained as the sum of three sinusoidal signals:

$$x(t) = s_A + s_B + s_C = \sin(2\pi f_A t + \varphi_A) + \sin(2\pi f_B t + \varphi_B) + \sin(2\pi f_C t + \varphi_C)$$

where: f_A = 1, f_B = 2 and f_C = 3 and φ_A, φ_B and φ_C are three random phases

- Use $Tmax = 10$ and $fsam = 100$

Exercise 1.2 – Plot of x(t) and |X(f)|

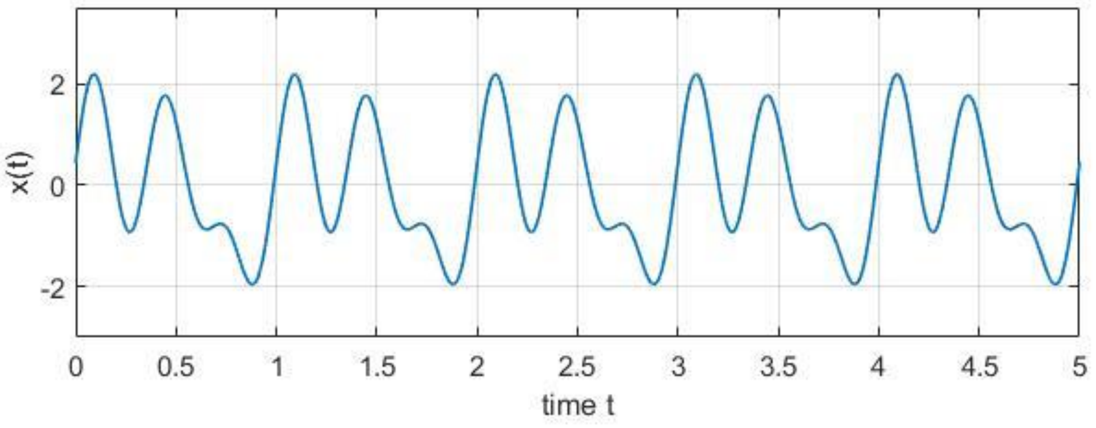

Figure 2.1.1: plot of x(t) wrt. the time axis

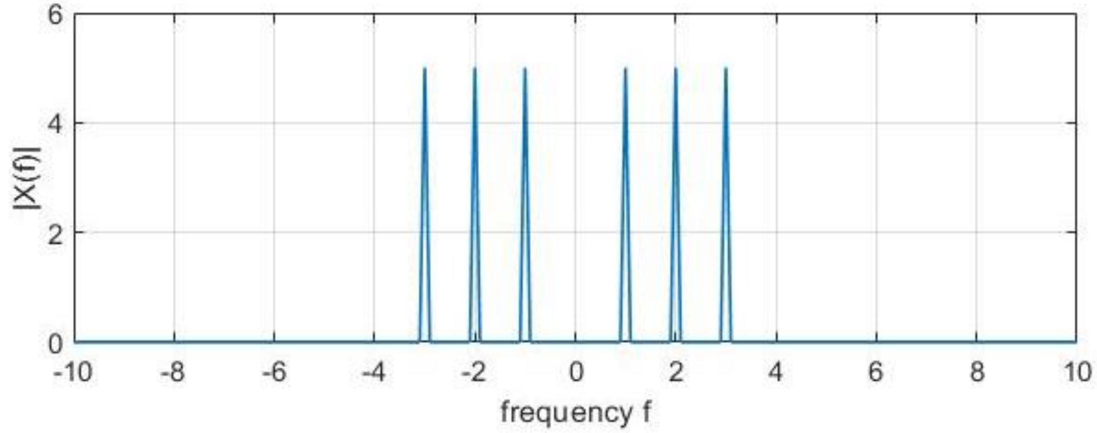

Figure 2.1.2: plot of |X(f)| wrt. the frequency axis

The plot in Figure 2.1.1 shows the signal x(t) obtained by taking the sum of the three sinusoidal signals with three random phase shifts:

Example in Figure 2.1: $\varphi_A = 6.0626$
$\varphi_B = 0.9903$
$\varphi_C = 6.0984$

The plot in Figure 2.1.2 shows the magnitude of the frequency components obtained by performing a Fourier transform on the signal x(t), taking its magnitude and shifting it as to be symmetric wrt. the frequency axis.

Since the frequency components for $s_A,\ s_B,\ s_C$ are equal to 1, 2, 3 respectively, a pulse can be seen at each of the corresponding frequencies.

Exercise 1.2 – Filtering of 3 sines on the frequency axis

- Design a low-pass filter with frequency response $H_A(f)$ to isolate the sinusoidal signal s_A with frequency f_A
- Design a band-pass filter with frequency response $H_B(f)$ to isolate the sinusoidal signal s_B with frequency f_B
- Design a high-pass filter with frequency response $H_C(f)$ to isolate the sinusoidal signal s_C with frequency f_C

- Multiply $X(f)$ by $H_A(f)$ to obtain $Y_A(f)$

- Compute the ifft for each to obtain $y_A(t)$, $y_B(t)$, $y_C(t)$

Exercise 1.2 – Low-pass filter design and plot of $H_A(f), Y_A(f)$, $y_A(t)$, $s_A(t)$

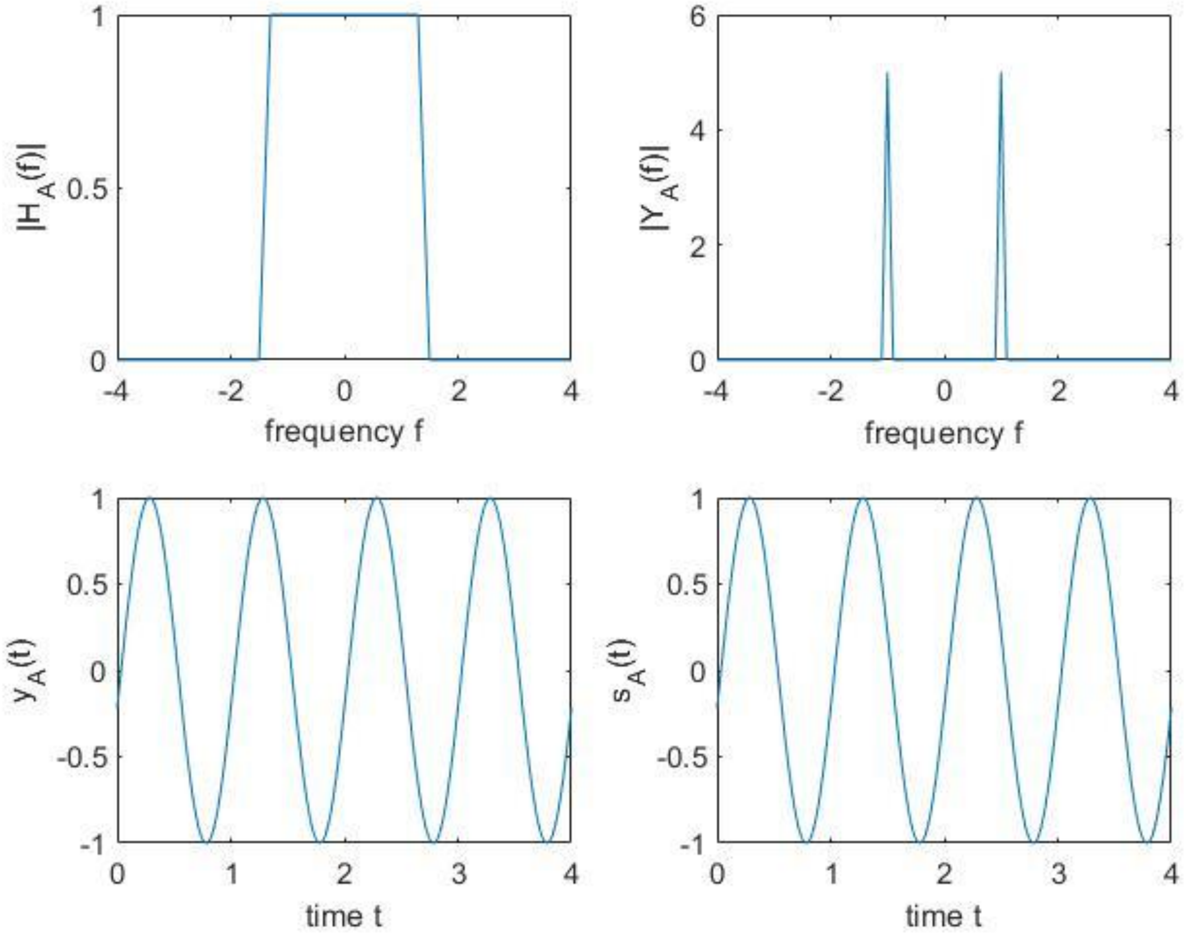

Figure 2.2: low-pass filter with frequency response $H_A(f)$, isolated frequency component of $Y_A(f)$, signal $y_A(t)$ obtained by ifft and the original signal $s_A(t)$

The low-pass filter has been designed by generating a rectangular signal with width 2.8 and centered about zero.

In MATLAB it has been implemented as:

```
H_A = rectangularPulse(-1.4,1.4,f_symm);
```

where f_symm is the symmetric frequency axis (centered at zero).

Multiplying $X(f)$ by $H_A(f)$, we obtained $Y_A(f)$.

By performing an ifft on $Y_A(f)$, we obtain the signal component $y_A(t)$ with frequency equal to 1, which is identical to the original signal $s_A(t)$

Exercise 1.2 – Band-pass filter design and plot of $H_B(f), Y_B(f)$, $y_B(t)$, $s_B(t)$

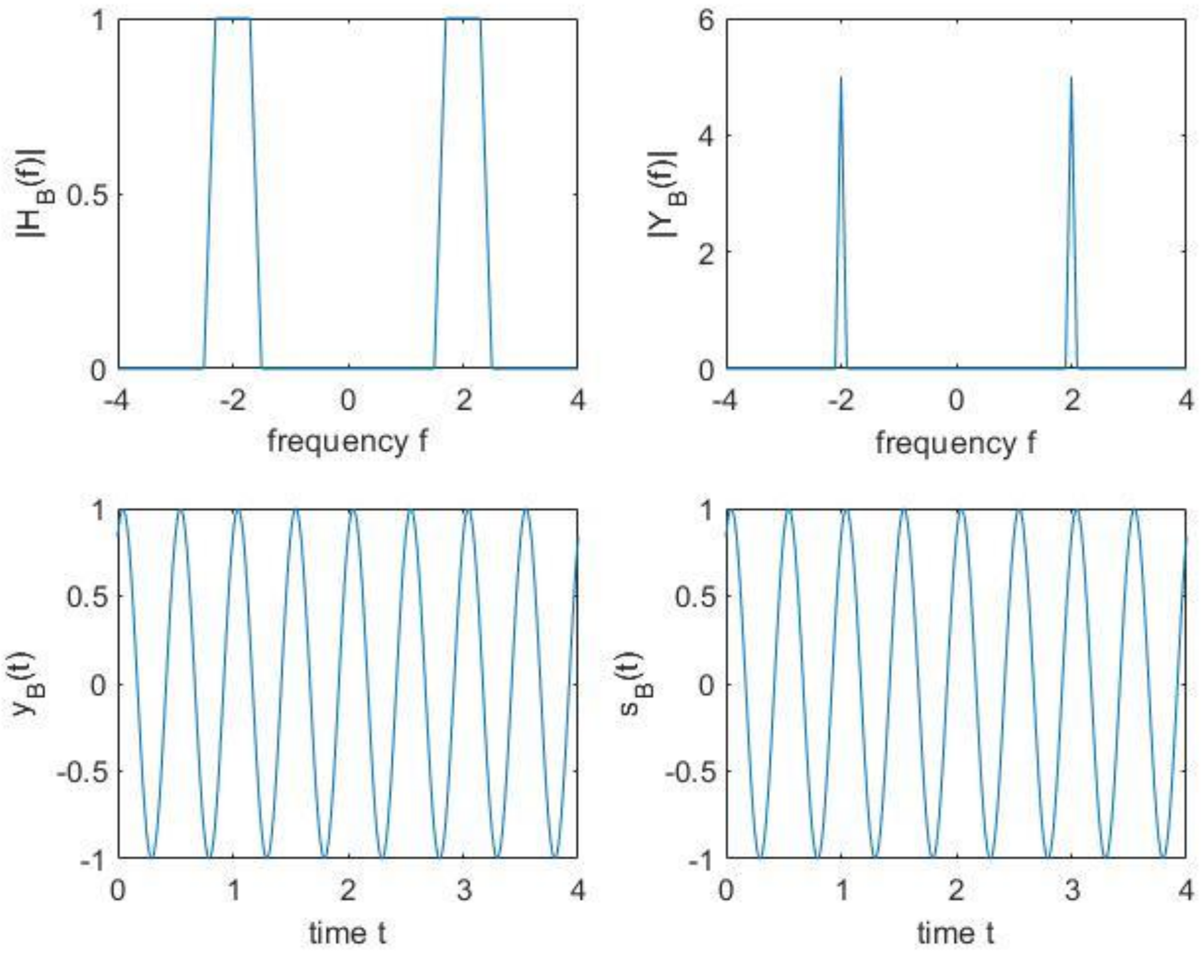

Figure 2.3: low-pass filter with frequency response $H_B(f)$, isolated frequency component of $Y_B(f)$, signal $y_B(t)$ obtained by ifft and the original signal $s_B(t)$

The band-pass filter has been designed by generating and taking a sum of two rectangular signals with width 0.8 each, centered at -2 and 2 respectively.

In MATLAB it has been implemented as:

```
H_B =
rectangularPulse(-2.4,-1.6,f_symm)+
rectangularPulse(1.6,2.4,f_symm);
```

Multiplying $X(f)$ by $H_B(f)$, we obtained $Y_B(f)$.

By performing an ifft on $Y_B(f)$, we obtain the signal component $y_B(t)$ with frequency equal to 2, which is identical to the original signal $s_B(t)$

Exercise 1.2 – High-pass filter design and plot of $H_C(f), Y_C(f)$, $y_C(t)$, $s_C(t)$

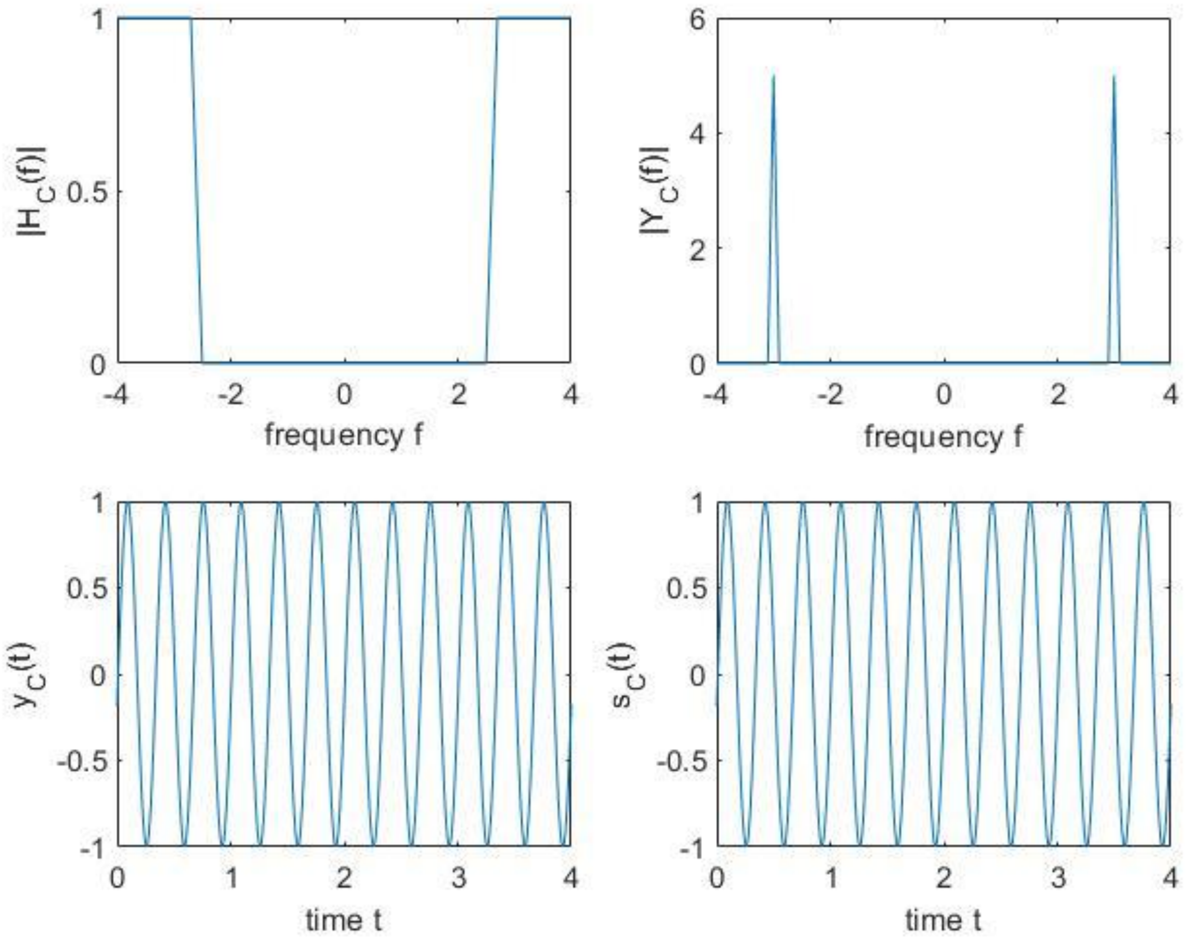

Figure 2.4: low-pass filter with frequency response $H_C(f)$, isolated frequency component of $Y_C(f)$, signal $y_C(t)$ obtained by ifft and the original signal $s_C(t)$

The high-pass filter has been designed by generating a rectangular signal with width 5.2, centered about zero and subtracted from 1.

In MATLAB it has been implemented as:

```
H_C =
1-rectangularPulse(-2.6,2.6,f_symm);
```

Multiplying $X(f)$ by $H_C(f)$, we obtained $Y_C(f)$.

By performing an ifft on $Y_C(f)$, we obtain the signal component $y_C(t)$ with frequency equal to 3, which is identical to the original signal $s_C(t)$

Exercise 1.3 – Quantization

- Consider a time axis $0 \leq t < Tmax = 1$
- Consider a sinusoidal signal with A = 1 and a given frequency value f_0 (e.g., f_0 = 1):

$$s(t) = A\cos(2\pi f_0 t)$$

- Choose a very high sampling frequency above the Nyquist frequency (e.g., f_{sam} = 100).
- Fix the number of quantization bits m.
- Divide the amplitude domain into 2^m segments. Use the value in the middle of the segment to represent it by using the Matlab ***quantiz*** function:

[index,sq] = quantiz(s, partition, codebook)

where:

partition is a vector with $2^m - 1$ values

codebook is a vector with 2^m values

Exercise 1.3 – Plot of the original signal $s(t)$ vs. quantized sinusoidal signal $s_q(t)$

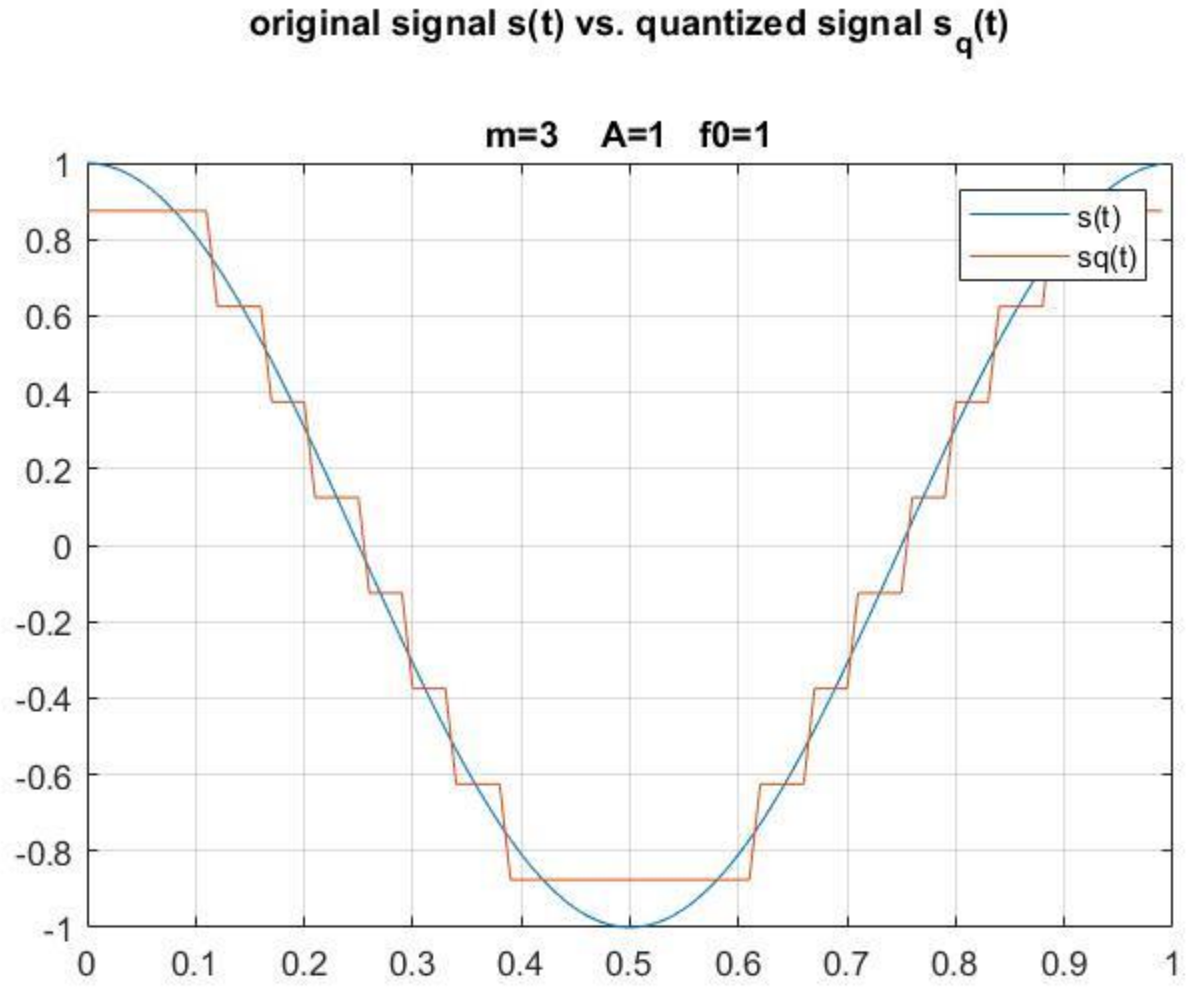

Figure 3.1: the original signal $s(t)$ vs. quantized sinusoidal signal $s_q(t)$ with the quantization bits m equal to 3, A=1 and f_0=1

The following parameters have been chosen in generating $s(t)$ and $s_q(t)$:

Quantization bits $m = 3$

Signal amplitude $A = 1$

Signal frequency $f_0 = 1$

To calculate the length of each segment using MATLAB, we introduce a value *delta* such that:

```
delta = (max(s)-min(s))/(2^m);
```

The partition and codebook are then calculated as:

```
partition=[-((2^m)/2—1)*delta:delta:((2^m)/2-1)*delta];
codebook=[-((2^m)/2-1)*delta-delta/2
                                :delta:
                                    ((2^m)/2-1)*delta+delta/2];

[index,sq] = quantiz(s,partition,codebook);
```

Exercise 1.3 – Comparing the spectrum of the original and the quantized signal

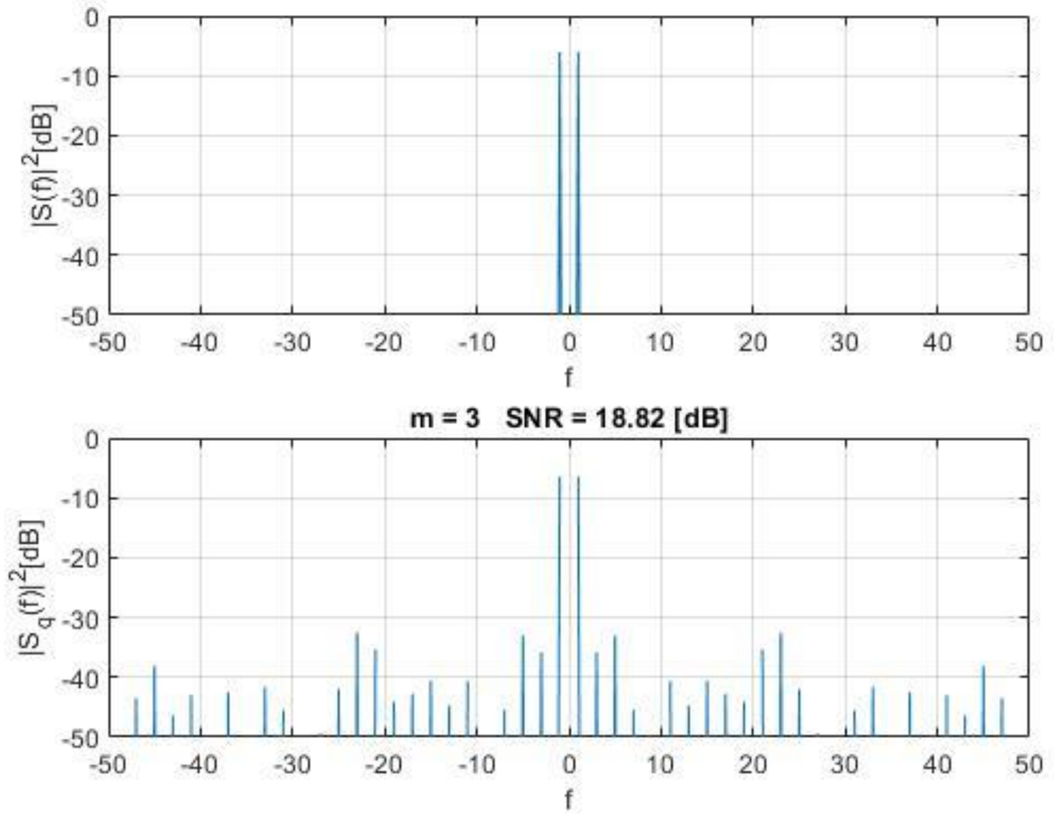

Figure 3.2: the spectrum original signal vs the spectrum of the quantized sinusoidal signal with m = 3 and SNR = 18.82 [dB] ≃ 6m [dB]

In Figure 3.2 it can be seen that in the spectrum of the quantized signal $|S_q(f)|^2$ in dB there are additional frequency components that do not appear in the spectrum of the original signal $|S(f)|^2$ in dB.

Those frequencies are present due to noise produced by the quantization process of the original signal.

An important quantity to introduce is the SNR (signal-to-noise ratio) which compares the levels of the desired signal vs. the level of noise.

The expected value for SNR is ≃ 6m [dB]

Exercise 1.3 – Computing the SNR

- Given the quantized signal, compute the energy E_s of the frequency component f_0, the energy E_n of the noise components at frequencies $f \neq f_0$ and their ratio in dB:

$$SNR = 10log_{10}\frac{E_s}{E_n}\ [dB]$$

The following MATLAB code performs the calculation:

```
E_s = max(SMQ);
New_s = SM - SMQ;
E_n = sum(New_s.^2)*f_res;
SNR1 = 10*log10(E_s/E_n);
```

- The SNR can also be computed using the formula: $SNR = 10log_{10}(1.5 * 4^m)\ [dB]$

Exercise 1.3 – SNR for different values of m

In the following plots examples for m = 4 and m = 5 are shown.
For each, the SNR is indeed ≃ 6m [dB] : m = 4, SNR = 24.92 [dB]
m = 5, SNR = 30.06 [dB]

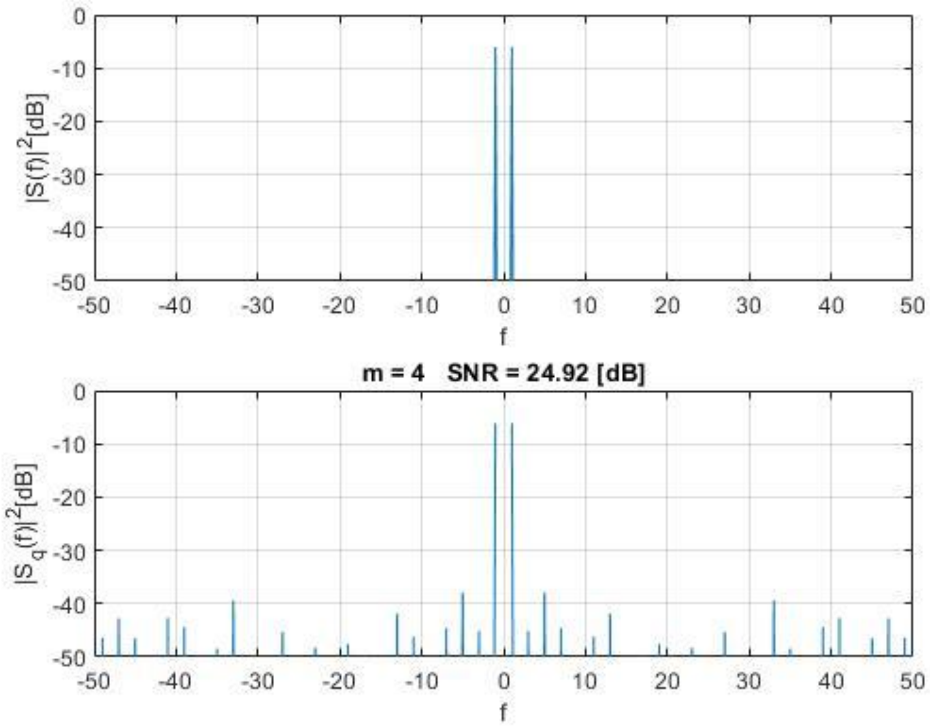

Figure 3.3: the spectrum original signal vs the spectrum of the quantized sinusoidal signal with m = 4

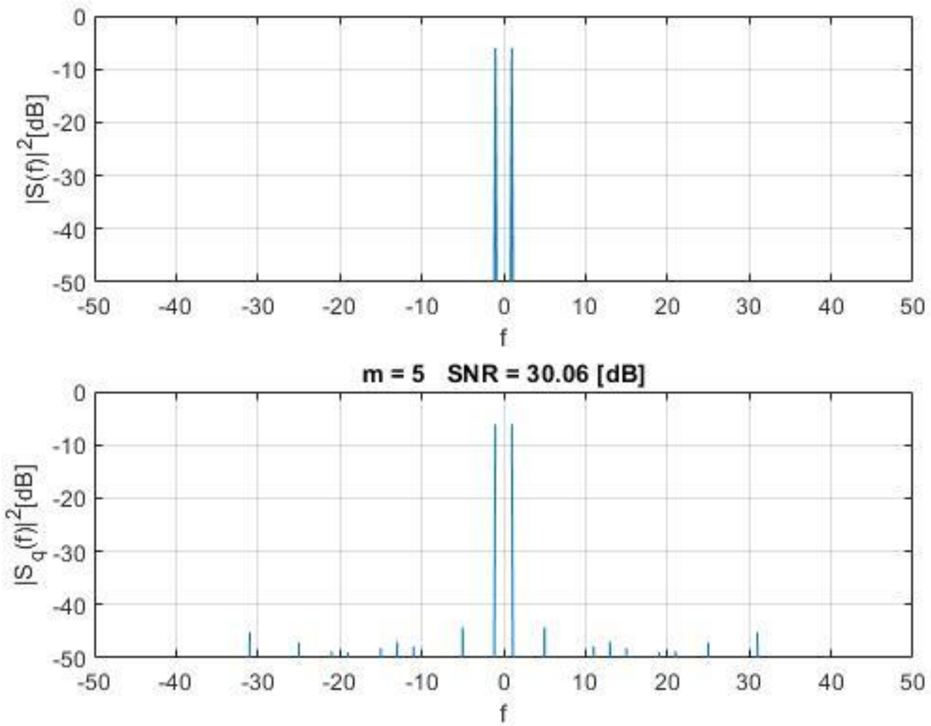

Figure 3.4: the spectrum original signal vs the spectrum of the quantized sinusoidal signal with m = 5