POLITECNICO DI TORINO

Electronic and Communications Engineering

Assignment Report 3 - Spectrum Analysis

Applied Signal Processing Laboratory

01TUMLP

Academic Year 2019/20

Gloria DAL SANTO
s245220

Exercise 1

In the first part of this exercise, I plot the absolute value of the DTFT of the rectangular function, defined as follows:

$$X(e^{j2\pi f}) = \frac{sin(\pi f L)}{sin(\pi f)} e^{-j\pi f(L-1)}$$

where the frequency ranges in the interval $[-1,\ 1]Hz$, with sampling period $F_s = 200\mu Hz$.
Figure 1 illustrates the resulting plots for L ranging from 3 to 7. $|X(e^{j2\pi f})|$ is a periodic function with period $F = 1Hz$ and with a number of side lobes, contained in each period, equal to $L - 2$. Moreover, it can be noticed that at $0Hz$ it assumes value L.

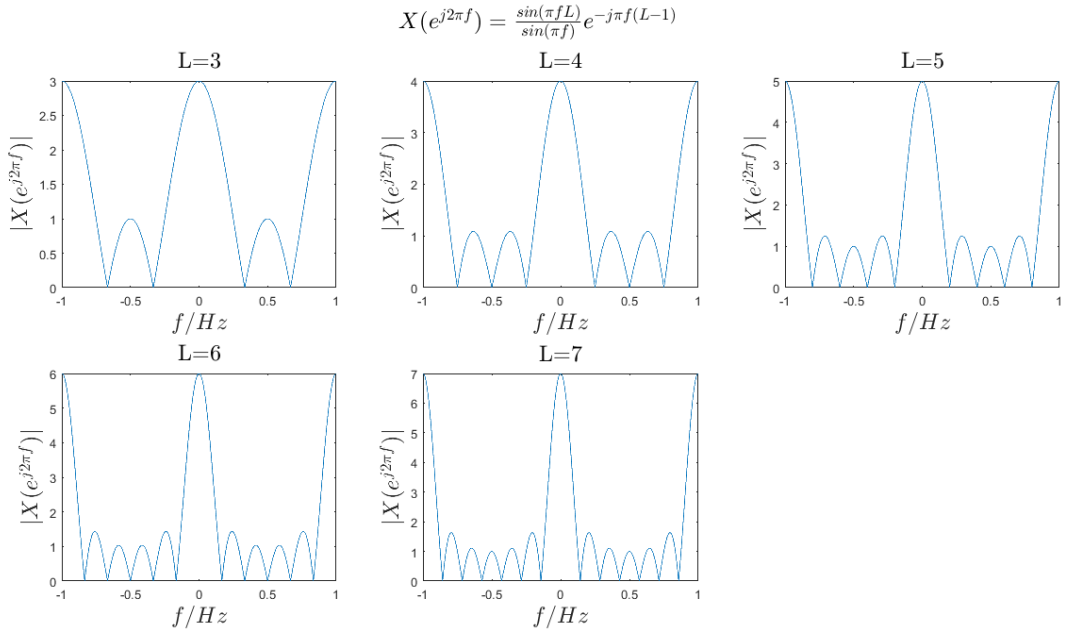

Figure 1: Exercise 1 - *DTFT of the rectangular function*

Then I defined a new function, identical to $X(e^{j2\pi f})$ apart from the phase factor:

$$Z(e^{j2\pi f}) = \frac{sin(\pi f L)}{sin(\pi f)}$$

where frequencies and parameter L are defined as before.
In Figure 2 are shown the plots of $Z(e^{j2\pi f})$, without computing the absolute value. Function $Z(e^{j2\pi f})$ has periodicity dependent on the parity of L, indeed for L odd it has period $F_o = 1Hz$ while for L even it has period $F_e = 2Hz$, twice F_o. To verify this behaviour it can be useful to define the limit of $Z(e^{j2\pi f})$ for L even at integer multiples of F_o, as follows:

$$\lim_{f \to n \cdot F_o} Z(e^{j2\pi f}) = \lim_{f \to n \cdot F_o} \frac{(sin(\pi f L))'}{(sin(\pi f))'} = \frac{L \cdot cos(\pi f L)}{cos(\pi f)} = L \cdot (-1)^n$$

Since each period lasts $2F_o$, it includes one positive and one negative main lobes (at $n = 0$ and $n = 1$ for the first period at positive frequencies). It can be shown that this limit goes to $(+1)^n$ in case of L odd, so that all the main lobes are positive.

Looking at Figure 3 it can be seen that in the interval $[-1/L, 1/L]\ Hz$, $Z(e^{j2\pi f})$ can be approximated to that of a Sinc function defined as $LSinc(\pi f L)$, where in this case $L = 9$. A more accurate evaluation

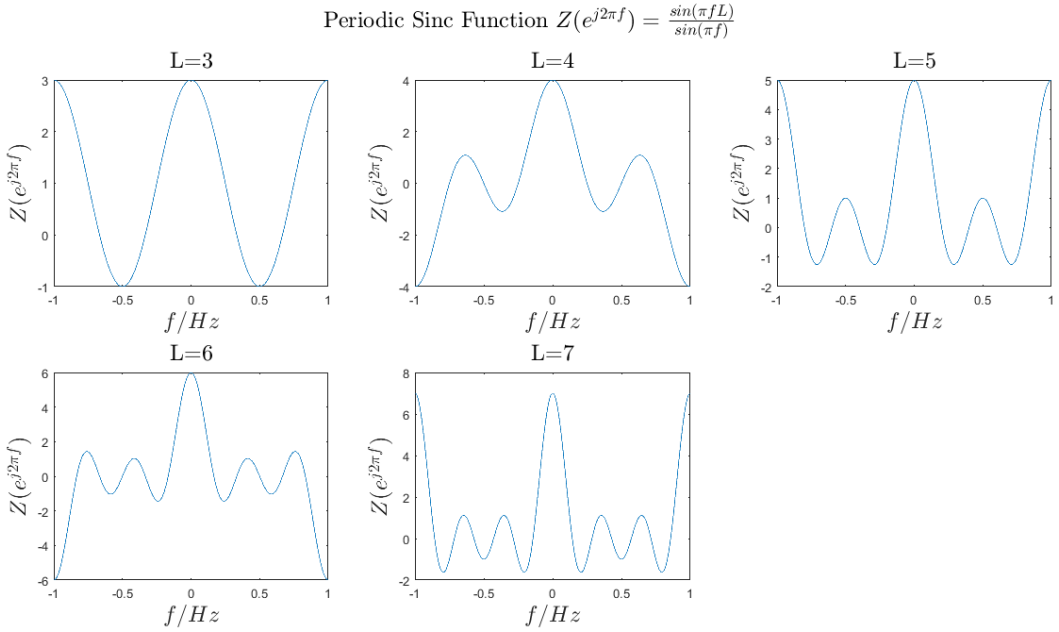

Figure 2: Exercise 1 - $Z(e^{j2\pi f})$

of such approximation can be done computing the energy contained by the two signals. To this purpose I computed the ratio between the energy of the Sinc function $E_s(f)$ and that of $Z(e^{j2\pi f})$ function $E_z(f)$ for three main frequency intervals. The obtained results are reported in Table 1, from which it can be inferred that within the frequency interval corresponding to the main lobe the two signals are almost identical, while for wider frequency intervals the decreasing behaviour of the Sinc function becomes evident, making the two signals differ.

f interval $[Hz]$	$E_s(f)/E_z(f)$ %
$[-1/L, 1/L]$	99.54
$[-3/L, 3/L]]$	98.64
$[-5/L, 5/L]]$	97.37

Table 1: *Energy ratio $E_s(f)/E_z(f)$ in three main frequency intervals*

Exercise 2

To implement the DFT function in Matlab, I used the for loop to compute the sum in eq.(1) at each value k ranging from 0 to N-1 where N is the number of samples in the input sequence $x[n]$.

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi n \frac{k}{N}} \tag{1}$$

To rearrange the vector, so that the frequency components are distributed in the interval $[-1/2, 1/2]$, I use the circshift function to circular shift the elements to the right by $N/2$ samples (rounded to the lowest integer when N is not even). Figure 4 compares the absolute value and the phase of the DFT computed using Matlab function fft with those obtained with the custom function that I made. The signal vector $x[n]$ used for the comparison is a random sequence of 100 samples generated with randn

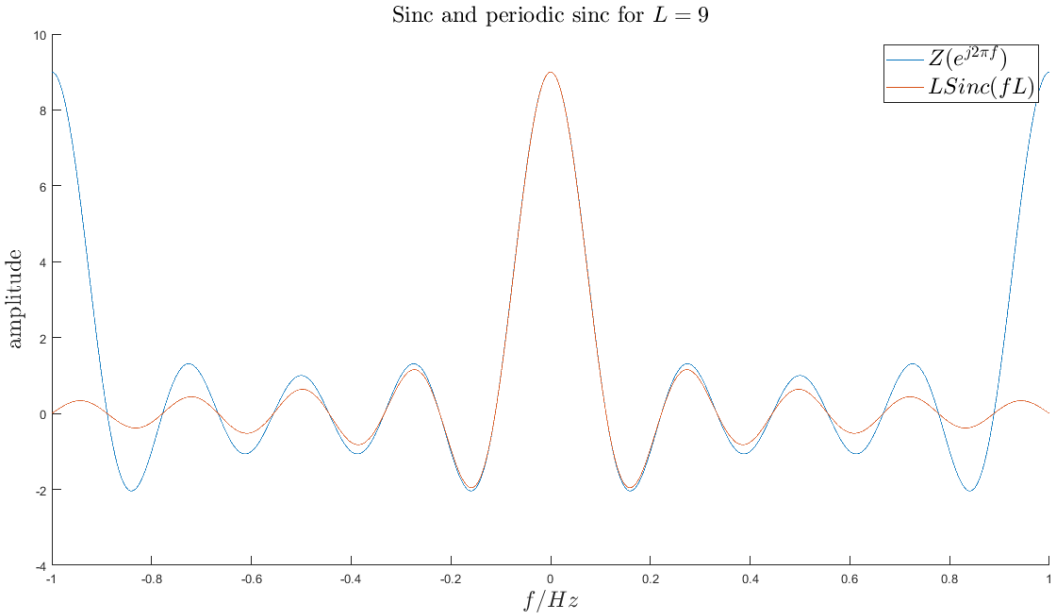

Figure 3: Exercise 1 - *Signals* $Z(e^{j2\pi f})$ *and* $LSinc(\pi fL)$ *for* $L = 9$

function. Even with a sequence of only $N = 100$ samples it can be observed that the fft algorithm is faster in computing the transform. Indeed, to evaluate the Fourier transform depicted in Figure 4 the function that I implemented took $4.095ms$, while the fft took $2.671ms$, nearly 35% less. This speed difference will be higher for larger data sets. As an example, with a sequence of 10000 samples the fft took $33.865ms$, $3.6413s$ less than the custom function.
From Figure 4 it can be seen that the resulting vectors are apparently equal, however small differences can occur due to floating-point errors. In particular, focusing on the custom function, it may happen that in correspondence with $k = N/2$ (before the circular shift) the sum $X(N/2) = \sum_{n=0}^{N-1} x[n]e^{-j\pi n}$ does not yield to a real coefficient only, but it has a "residual" imaginary part. This will cause the phase of the custom DFT represented in the final plot (hence, after the circular shift) to be shifted by $360deg$ with respect to the fft whenever the the residual imaginary part has negative sign. On the contrary, the fft at $k = 0$ and at $k = N$ has imaginary part exactly equal to zero, so I suppose that it implements some kind of rounding algorithm that overcomes the errors related to the floating-point representation.

Exercise 3

To construct the triangular sequence $x[n]$ of $N = 15$ samples I computed the convolution of two rectangular signals each of length $M = (N + 1)/2 = 8$ samples and with height $K/M = 1/2$, where $K = 4$. Figure 5 and 6 show the sequence $x[n]$ and the absolute value of its DFT $X(k)$, respectively. At $n = 0$ signal $x[n]$ has height $K^2/M = 2$, as it was expected from the convolution function. The 15 points DFT $X(k)$ has a rough shape due to the high resolution interval $\Delta f = 1/15$. By means of zero-padding techniques its shape can be made "smoother". This is done by reducing the frequency spacing defined as $\Delta f = 1/N$, hence increasing the observation time by adding zeros to the sequence $x[n]$. Figure 7 and 8 show the set of sequences for $\Delta f = 1/64$, obtained by zero-padding $x[n]$ to reach $N = 64$ samples. This procedure does not add information to the starting sequence, but its DFT, having a better resolution, may highlight some details that otherwise would not have been observed. Increasing the number of samples to $N = 128$ the shape becomes even more smoother, as it is shown in Figure 10.
The correctness of these results can be verified by comparing the obtained $X(k)$ with the one computed

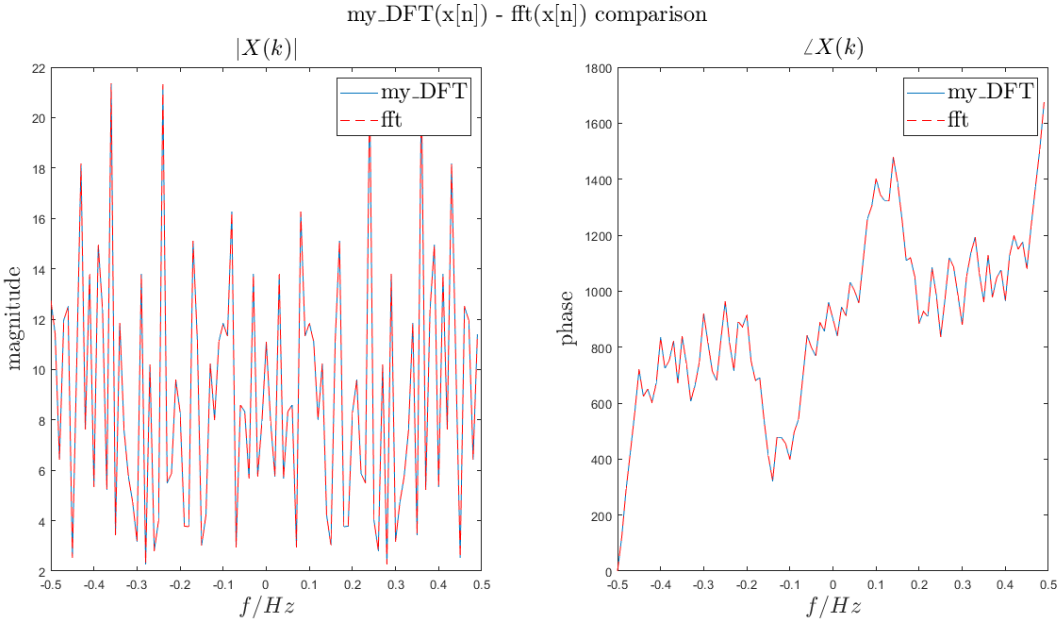

Figure 4: Exercise 2 - $|X(k)|$, $arg(X(k))$ *computed with the custom function and the Matlab function*

using the DTFT general formula for the convolution of the two rectangular sequences:

$$X(e^{j2\pi f}) = X_r^2(e^{j2\pi f}) = \left(e^{-j\pi f(M-1)} \cdot \frac{K}{M}\frac{sin(\pi f M)}{sin(\pi f)}\right)^2$$
$$= e^{-2j\pi f(M-1)} \cdot \frac{K^2}{M^2}\frac{sin^2(\pi f M)}{sin^2(\pi f)} \tag{2}$$

Where $X_r(e^{j2\pi f})$ is the DTFT of the rectangular sequence. From this result it can be verified that the value of the function at $k = 0$ is $X(0) = K^2 = 16$. Figure 11 shows the plot of the DTFT in eq.(2) sampled at intervals equal to $1/128$, compared with the previously depicted 128 points and 64 points DFTs. It can be verified that the samples of the DFTs are aligned with the theoretical function.

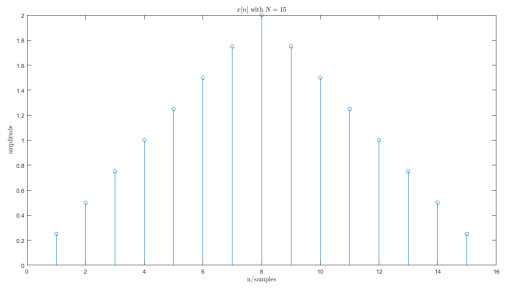

Figure 5: Exercise 3 - *triangular sequence* $x[n]$ *of* $N = 15$ *samples*

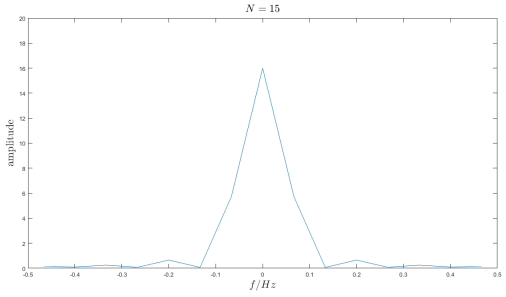

Figure 6: Exercise 3 - $|X(k)|$ *with resolution* $\Delta f = 1/15$

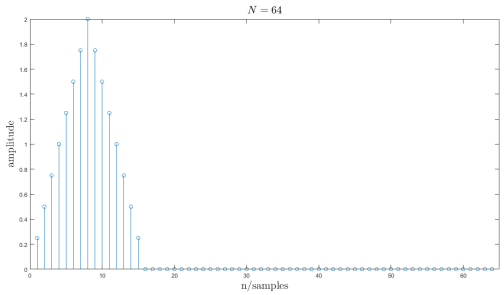

Figure 7: Exercise 3 - *sequence* $x[n]$ *zero-padded to* $N = 64$ *samples*

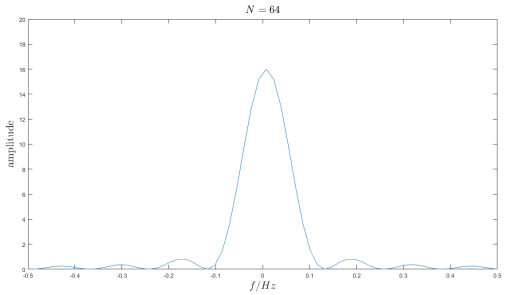

Figure 8: Exercise 3 - $|X(k)|$ *with resolution* $\Delta f = 1/64$

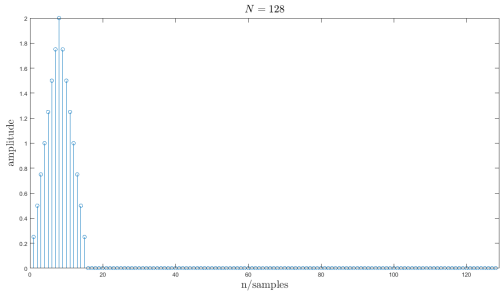

Figure 9: Exercise 3 - *sequence* $x[n]$ *zero-padded to* $N = 128$ *samples*

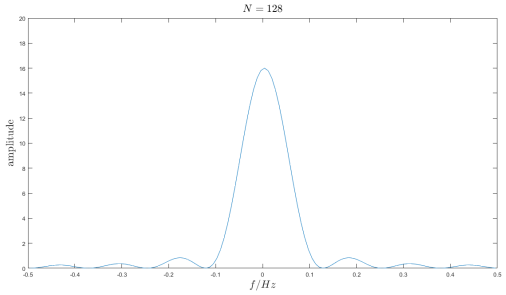

Figure 10: Exercise 3 - $|X(k)|$ *with resolution* $\Delta f = 1/128$

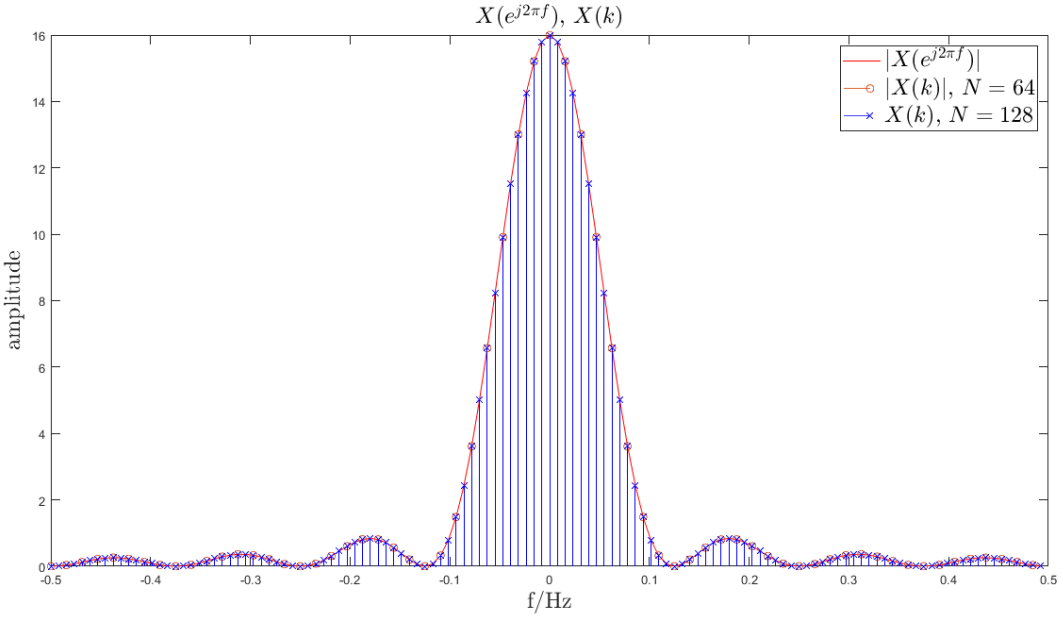

Figure 11: Exercise 3 - *DTFT, DFT comparison*

Exercise 4

The signal under analysis in this exercise is the following:

$$x[n] = 2cos(2\pi f_0 T_c n) + cos(2\pi f_1 T_c n)$$

where the frequencies of the cosines are $f_0 = 8Hz$ and $f_1 = 10Hz$. The signal is sampled at frequency $f_c = 32Hz$ for $N = 128$ samples.
From the initial parameters it can be directly calculated the observation time-interval T_0 and the frequency resolution Δf as follows:

$$T_0 = \frac{N}{f_c} = 4s$$
$$\Delta t = \frac{f_c}{N} = \frac{1}{T_0} = 0.25Hz$$

I evaluated the continuous time Fourier transform performing the following convolution by means of Matlab conv function:

$$X(f) = T_0 Sinc(T_0 f) * \left[(\delta(f - f_0) + \delta(f + f_0)) + \frac{1}{2}(\delta(f - f_1) + \delta(f + f_1))\right] \tag{3}$$

Figure 12 shows the absolute value of the obtained function $X(f)$ together with the DFT evaluated

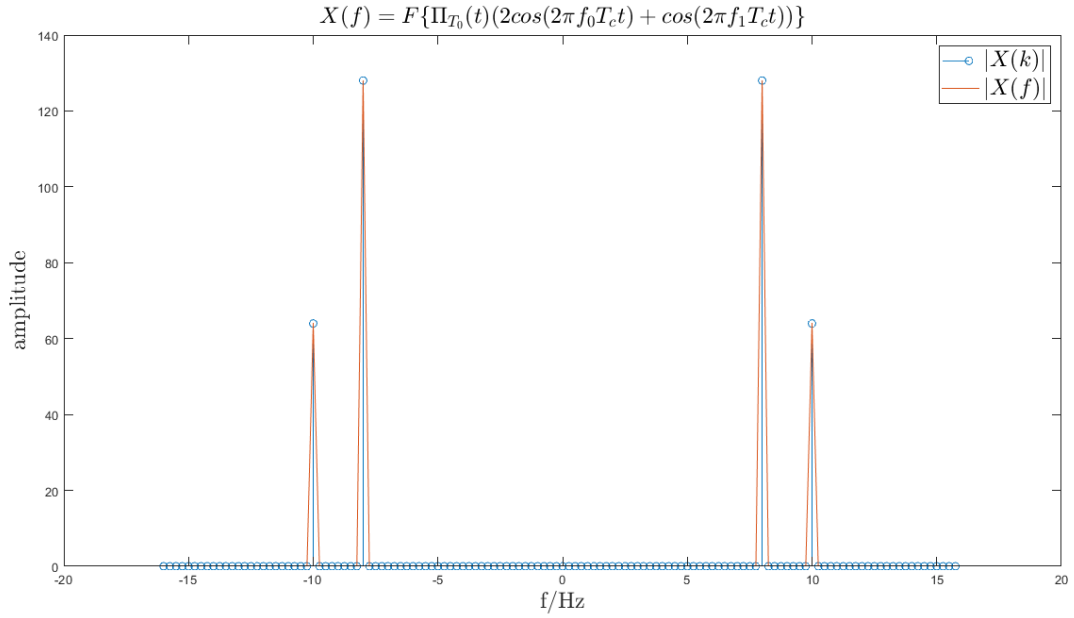

Figure 12: Exercise 4 - *CTFT, DFT comparison*

with the fft function. I plot $|X(f)|/Tc$ to be consistent with the amplitude normalization of the DFT function. The two vectors depict four deltas at $\pm f_0$ and at $\pm f_1$ (the last ones have half the amplitude as a consequence of the multiplication constant in front of the cosine function). Since the sinc function, which comes from the truncation of the signal $x[n]$, goes to zero at integer multiples of the frequency resolution Δf different from $\pm f_0$ and $\pm f_1$, its presence cannot be seen from the values of the vector. However, if the number of samples is increased by zero-padding $x[n]$, the delta functions turn out to be sinc functions centered at $\pm f_0$ and at $\pm f_1$, as expected. Figure 13 shows this result when $x[n]$ is zero-padded with additional 128 samples, leading to a frequency resolution of $\Delta f = 0.125Hz$.
If N is set such that the observation time T_0 does not coincide with a multiple of the least common multiple of the cosine periods T_0 and T_1, which corresponds to $0.5s$, then the Fourier transform will be affected by the spectral-leakage. As an example, Figure 14 shows the case in which $T_0 = NT_c = 3.65625s$. The Fourier transform appears distorted, the amplitude of the peaks around $\pm f_0$ and at $\pm f_1$ is lower with respect to the original spectrum and the vector is non-zero everywhere.

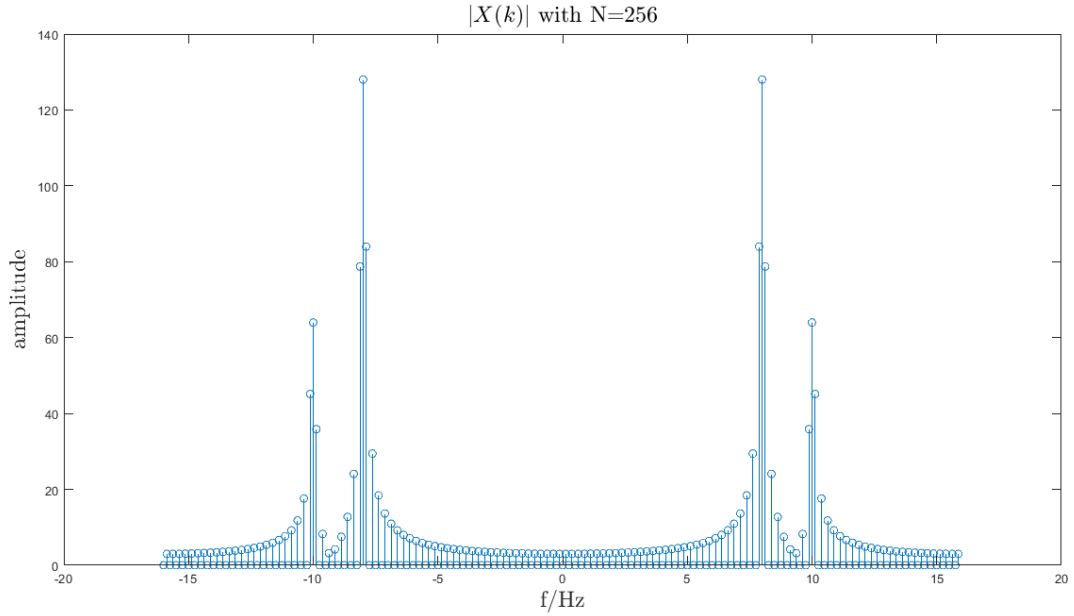

Figure 13: Exercise 4 - *DFT of* $x[n]$ *zero-padded to* $N = 256$ *samples*

Exercise 5

In this exercise I had to estimate the Power Spectral Density (PSD) of the following random process:

$$X(t) = cos(2\pi f_1 t) + cos(2\pi f_2 t) + W(t) \tag{4}$$

where $f_1 = 100Hz$, $f_2 = 110Hz$ and $W(f)$ is a zero-mean White Gaussian Noise (WGN) with variance $\sigma^2 = 25$.
I sampled the signal at sampling frequency $f_s = 1kHz$ and represented it for a time window $T_0 = 20s$. At first step, using the Matlab pwelch function, I estimated its PSD computing the Welch periodogram with zero overlapping and Hamming window function. I set the pwelch to split up the sampled signal $x[n]$, composed of $N = T_0 \cdot f_s = 20000$ samples, into multiple segments of length $D = 128$. Figure 15 shows the obtained estimation of the PSD, where the vertical red lines are located at $\pm f_1$ and $\pm f_2$ to highlight the spectral components of the noiseless signal. Being the spectral resolution $\Delta f = \lfloor f_s/D \rfloor = 7.8Hz$, the two harmonic components that are originally located $10Hz$ apart are not clearly separated, even though two peaks around $\pm 100Hz$ can be noticed on the top of the spectral components related to the WGN, present along the whole frequency axis.
Figure 16 illustrates the PSD estimated by setting the pwelch function for the simple periodogram. In this case the two harmonic components are separated, indeed the frequency resolution is $\Delta f = f_s/N = 0.05Hz$, far smaller than before. However the PSD has a high variance, with the spectrum of the noise ranging within a band of $30dB$.
To lower the variance while keeping a good frequency resolution, I used again the Weltch periodogram with hamming window of the first example, but with the difference of a higher number of samples $D = 500$ and an overlap of 125 samples, 25% of each segment. With these new values I obtained a estimated PSD with a limited variance and a frequency resolution of $2Hz$ that allows to clearly distinguish the frequency components at $\pm 100Hz$ and at $\pm 110Hz$, as it is shown in Figure 17.
Finally, Figure 18 illustrates the results obtained with a Bartlett periodogram and a Welch one that have the same frequency resolution. The Weltch periodogram was obtained with a hamming window and an overlap of 50%. The main difference between the two is in the variance: from the plots it can

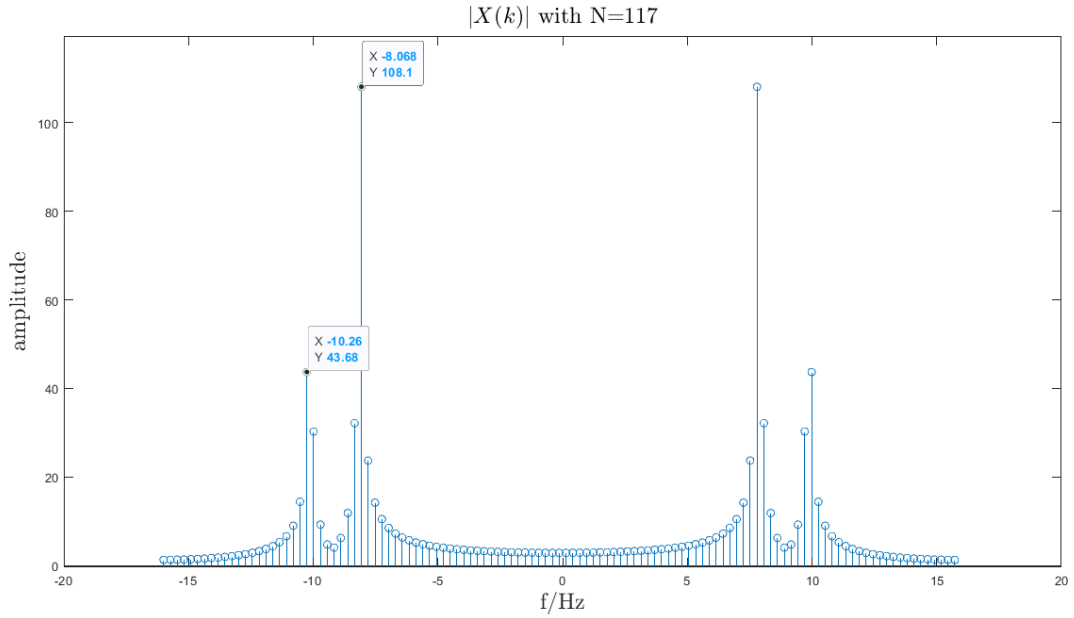

Figure 14: Exercise 4 - *spectral-leakage effect with* $N = 117$

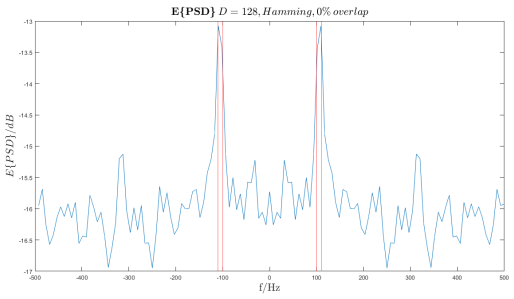

Figure 15: Exercise 5 - *PSD of* $x[n]$ *estimated with the Welch periodogram*

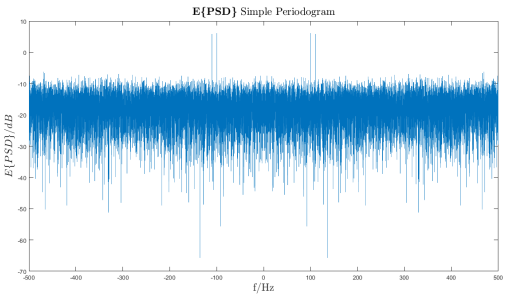

Figure 16: Exercise 5 - *PSD of* $x[n]$ *estimated with the simple periodogram*

be seen that the Bartlett periodogram has an higher variance, due to the fact that the segments are not overlapped and the average is computed on a smaller set of segments. Moreover, since in the Welch periodogram the segments of $x[n]$ are windowed with a smoother function, the result is slightly less biased with respect to the Bartlett.

Exercise 6

I filtered the signal $x[n]$ defined in eq.(4) with a low pass filter with cutoff frequency $250Hz$ to obtain a new sequence $y[n]$. Then I computed the simple periodogram, and both the Bartlett and Welch periodograms of $y[n]$ keeping the parameters defined in the exercise before. In Figure 19 it can be seen that the Bartlett periodogram has the lowest out-of-band attenuation. This result is related to the kind of window involved and to the concept of the bias. The Welch periodogram applies to the segments a Hamming window, which in frequency has lower amplitude of the side lobes with respect to the spectrum of the rectangular window present in the Bartlett periodogram. Even if the simple periodogram has a high out-of-band attenuation it has also a very high variance. In Figure 20 the effects of the rectangular, Hann and Hanning windows are compared. The window whose spectrum has lower side lobes is the Hann one, and indeed it is the one that leads to a lower bias, with the higher out-of-band attenuation. Clearly the one that has the lower out-of-band attenuation is the periodogram

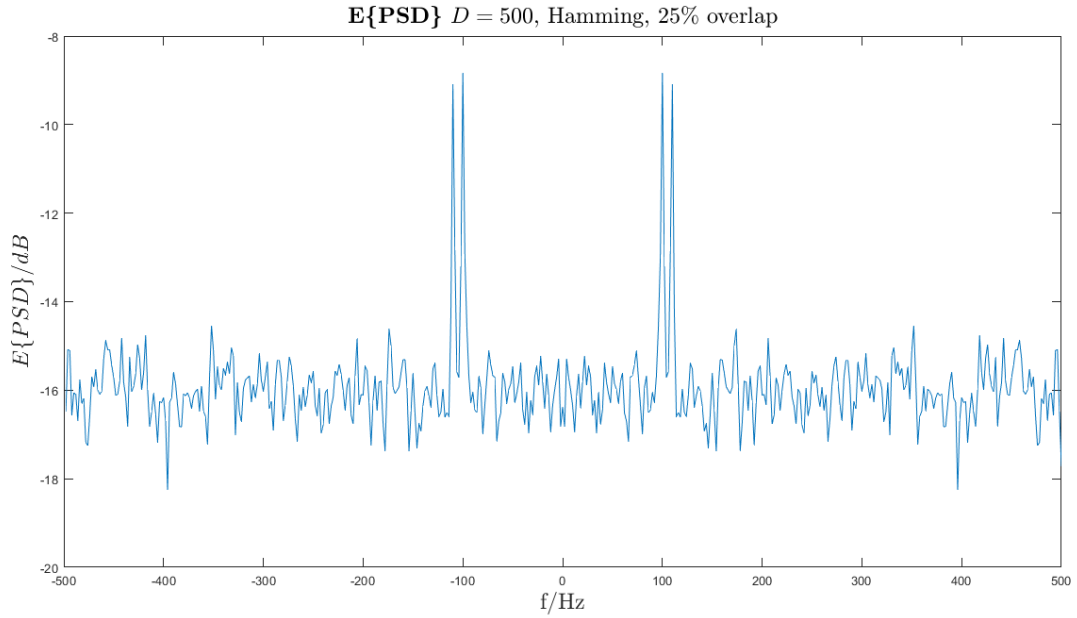

Figure 17: Exercise 5 - *PSD of* $x[n]$ *estimated with the Welch periodogram with 25% overlap*

computed with the rectangular window, which has very high side lobes in the frequency domain.

Exercise 7

In this last exercises I followed the given instruction to create a custom function that computes the Bartlett periodogram by means of $\mathrm{my\_DFT}$ function created in Exercise 2. Figure 21 compares the result obtained through the custom $\mathrm{my\_Bartlett}$ function and Matlab pwelch function for a random sequence $W[n]$ of 20000 samples generated as a WGN with variance $\sigma^2 = 25$ and then filtered with a low pass filter at cutoff frequency $f_c = 0.25Hz$. The two generated periodograms are exactly the same, confirming the correct operation of the custom function.

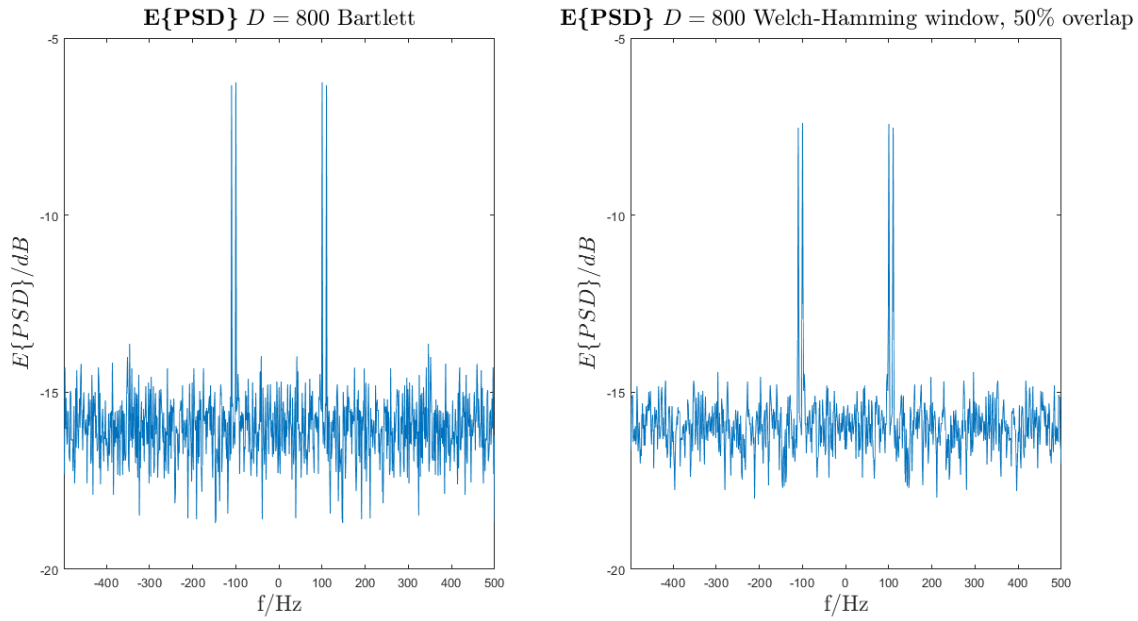

Figure 18: Exercise 5 - *PSD of* $x[n]$ *estimated with the Bartlett periodogram (left) and with the Welch periodogram (right)*

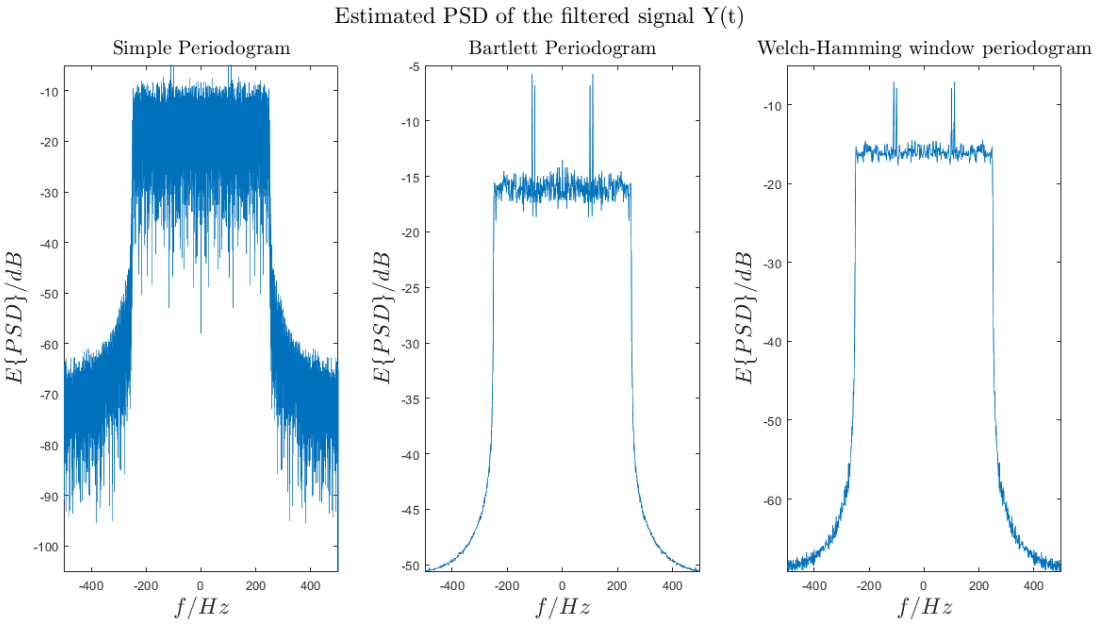

Figure 19: Exercise 6 - *Simple, Bartlett and Welch periodogram of* $Y(t)$

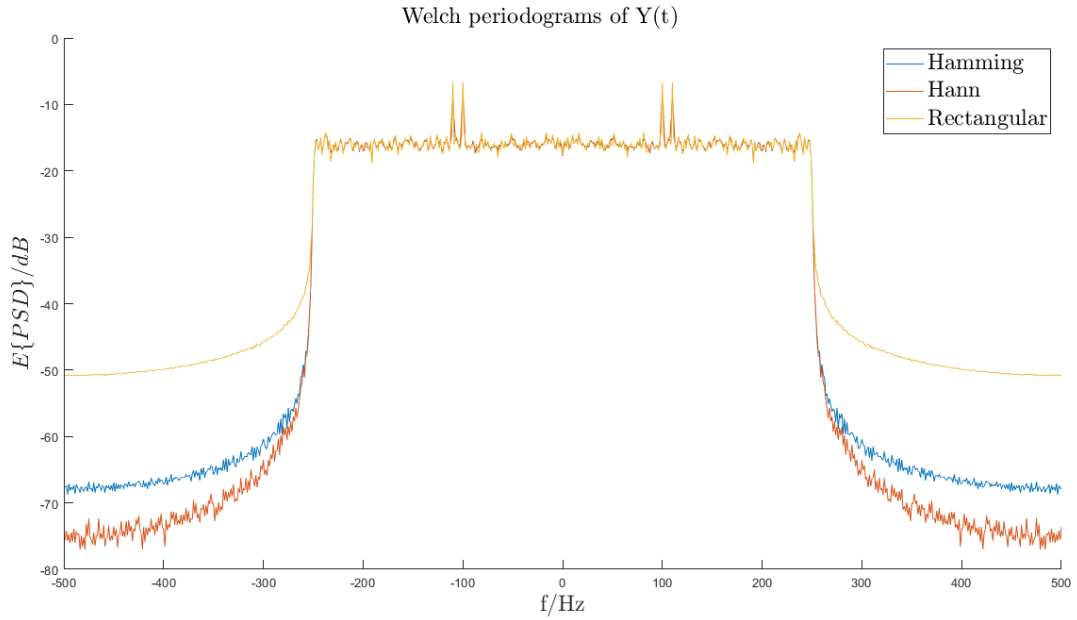

Figure 20: Exercise 6 - *Welch periodograms of* $Y(t)$ *with different windows*

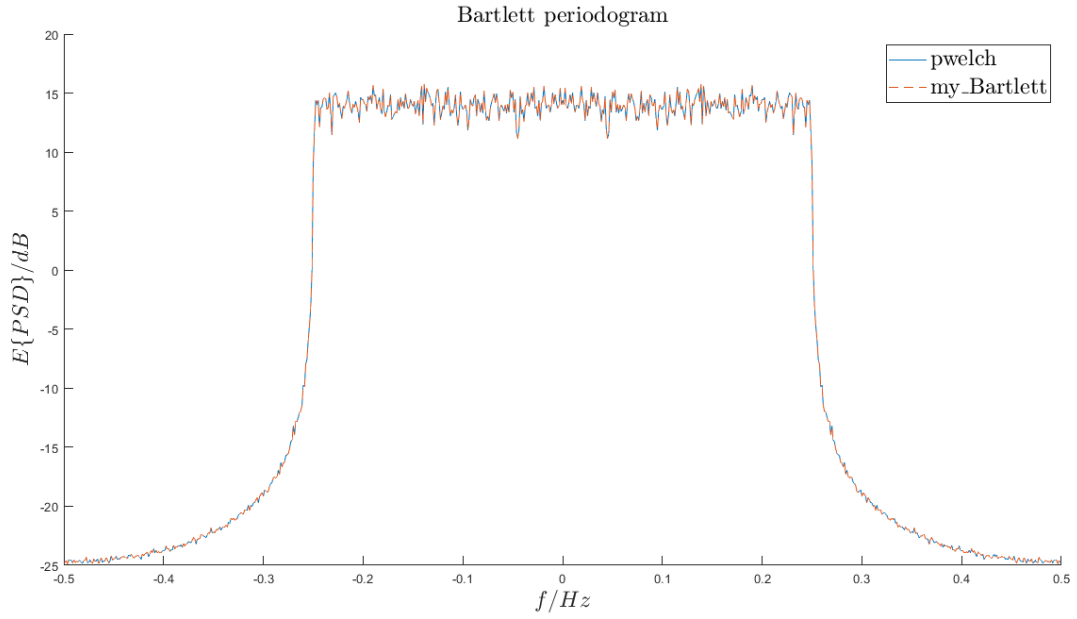

Figure 21: Exercise 7 - *Bartlett periodograms computed with* pwelch *and* my_Bartlett *functions*