LABORATORY 1
DISCRETE-TIME SIGNALS

1.1. Introduction

A discrete-time signal is represented as a sequence of numbers, called samples. A sample value of a typical discrete-time signal or sequence is denoted as:

$$x[n] \quad , \quad N_1 \le n \le N_2 \tag{2.1}$$

In MATLAB these sequences can be defined as row or column vectors, with real or complex elements. A first limitation arises from the fact that these vectors are of finite length, while the digital signal processing problems can work with infinite length sequences.

1.2. Generation of Sequences

Two basic discrete-time sequences are the unit impulse sequence and the unit step sequence.

- ***The unit impulse sequence***

$$\delta[n] = \begin{cases} 1 & , \quad n = 0 \\ 0 & , \quad n \neq 0 \end{cases} \tag{2.2}$$

Delayed unit impulse

$$\delta[n - n_0] = \begin{cases} 1 & , \quad n = n_0 \\ 0 & , \quad n \neq n_0 \end{cases} \tag{2.3}$$

Since MATLAB cannot define sequences of infinite length, the range of values for n must be stated. To facilitate the definition of sequences of this kind, a MATLAB function will be created

```
function [y,n]=impuls(li,ls,k);
```

1. Discrete-Time Signals

```
%IMPULS – the unit (Dirac) impulse in discrete time,
%defined on a finite temporal support
%-syntax:
%y=impuls(li,ls,k)
%[y,n]=impuls(li,ls,k)
%-output parameters:
%y=line vector which represents delta(n-k) on the support
%[li;ls]
%n=line vector which represents the support [li;ls]
%- input parameters:
%li=the inferior limit of the temporal support;
%ls= the superior limit of the temporal support;
%k=the index from delta(n-k)
%-for display: stem(n,y)

if nargin<3
  error('Too few input arguments')
elseif nargin>3
  error('Too many input arguments')
end

if nargout>2
  error('Too many output arguments')
end

if li>=ls
  error('The temporal support is not valid')
end

if (k<li)|(k>ls)
  error('The temporal support is not valid')

end

L=ls-li+1;
y=zeros(1,L);
y(k-li+1)=1;
n=li:ls;
```

1. Discrete-Time Signals

Remarks:

- The rules for a correct definition of Matlab functions can be found here: https://www.mathworks.com/help/matlab/ref/function.html
- The rules for a correct definition of `Octave` functions can be found here: https://octave.org/doc/v4.0.0/Defining-Functions.html
- The name of the functions is the same as the name of the file holding that function. You will not use function names already defined/used in Matlab!
- Functions are called from other Matlab files saved in the same current folder! Do NOT run a function as a usual script!

***Examples*:**

Define and plot the following discrete time sequences:

1. $x_1[n] = \delta[n]$
2. $x_2[n] = 0,5\delta[n-3]$, for $-10 \le n \le 10$.

```
[y1,n]=impuls(-10,10,0)
stem(n,y1),grid
[y2,n]=impuls(-10,10,3)
stem(n,0.5*y2),grid
```

- ***The unit step sequence***

$$u[n] = \begin{cases} 1 & , \quad n \ge 0 \\ 0 & , \quad n < 0 \end{cases} \tag{2.4}$$

Delayed unit step sequence:

$$u[n-n_0] = \begin{cases} 1 & , \quad n \ge n_0 \\ 0 & , \quad n < n_0 \end{cases} \tag{2.5}$$

A MATLAB function can also be created to define the unit step sequences:

```
function [y,n]=treapta(li,ls,k);

%TREAPTA - unit step sequence in discrete time, %defined
on a finite temporal support
%-syntax:
%y=treapta(li,ls,k)
%[y,n]=treapta(li,ls,k)
%-output parameters:
%y=line vector which represents u(n-k) on the support
%[li;ls]
%n=line vector which represents the support [li;ls]
```

1. Discrete-Time Signals

```
%- input parameters:
%li=the inferior limit of the temporal support;
%ls= the superior limit of the temporal support;
%k=the index from u(n-k)
%-for display: stem(n,y)


if nargin<3
  error('Too few input arguments')
elseif nargin>3
  error('Too many input arguments')
end

if nargout>2
  error('Too many output arguments')
end
if li>=ls
  error('The temporal support is not valid')
end
if (k<li)|(k>ls)
  error('The temporal support is not valid')


L=ls-li+1;
y=zeros(1,L);
y(k-li+1:L)=1;
n=li:ls;
```

***Examples*:**
Define and plot the following discrete time sequences:
1. $x_1[n] = u[n]$
2. $x_2[n] = 0{,}7(u[n+3] - u[n-3])$, for $-5 \le n \le 10$.

```
[y1,n]=treapta(-5,10,0)
stem(n,y1),grid
[y21,n]=treapta(-5,10,-3)
[y22,n]=treapta(-5,10,3)
y2=0.7*(y21-y22)
stem(n,y2),grid
```

1. Discrete-Time Signals

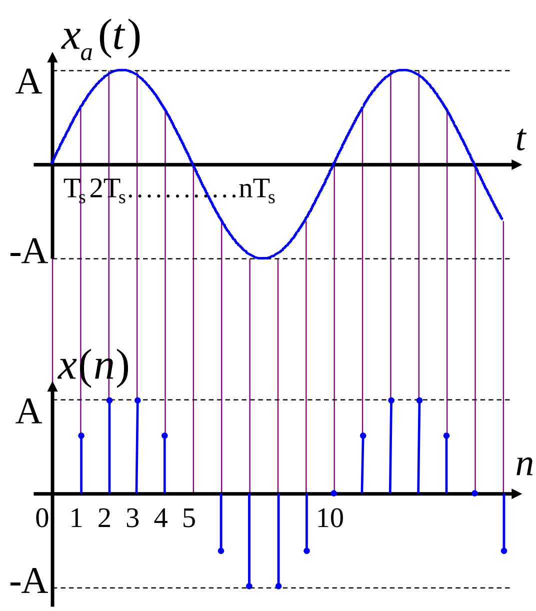
E1. ***Exercises*:**

Define and plot the following discrete time sequences:

1. $x_1[n] = \delta[n] - \delta[n-1]$ for $-10 \le n \le 10$
2. $x_2[n] = \delta[n] - 0,5\delta[n-1] + 0,3\delta[n-2] - 2\delta[n+1]$ for $-10 \le n \le 10$
3. $x_3[n] = u[n] + 0,5u[n-4] - 0,5u[n+4]$ for $-10 \le n \le 10$
4. $x_4[n] = \delta[n-1] + u[n-5] + \delta[n+2] - 2\delta[n-9]$ for $-10 \le n \le 20$
5. $x_1[n] = \left(\frac{1}{2}\right)^n - \left(-\frac{1}{2}\right)^n$ for $0 \le n \le 10$
6. $x_2[n] = \ln\left|\cos\left(\frac{n\pi}{15}\right) - \sin\left(\frac{n\pi}{15}\right)\right|$ for $-20 \le n \le 20$
7. $x_1[n] = (-1)^n \cos\left(\frac{n\pi}{15}\right)$ for $0 \le n \le 10$

A discrete signal usually appears from the sampling of a continuous signal with a sampling time T_s. The values stored in the vector representing the discrete signal correspond to the amplitude of the continuous signal taken at integer moments in time, multiples of T_s: $n \cdot T_s,\ n \in \mathbb{N}$.

$$s(n) = s_c(nT_s)$$



For example, a sine wave on the frequency F_0

$$s_c(t) = A_0 \cdot \sin(2\pi \cdot F_0 \cdot t)$$

The sampled signal is:

$$s(n) = A_0 \cdot \sin(2\pi \cdot F_0 \cdot n \cdot T_s) = A_0 \sin\left(2\pi \frac{F_0}{F_s} n\right)$$

Denoting the normalized frequency $f_0 = \frac{F_0}{F_s}$, and the angular normalized frequency $\omega_0 = 2\pi f_0 = 2\pi \frac{F_0}{F_s}$, the result is:

$$s(n) = A_0 \sin(2\pi \cdot f_0 \cdot n) = A_0 \sin(\omega_0 n)$$

Examples:

1. Represent graphically, using the **stem** function, a discrete signal obtained by sampling a sine wave of 300 Hz frequency, time span of 10 milliseconds and amplitude 2. The sampling frequency is F_s = 4kHz. What is the number of samples in the discrete signal?

```
F0 = 300; Fs = 4000;
Tmax = 10^(-2);    % signal duration of 10 ms
N = Tmax*Fs;       % number of samples N=10ms*4kHz
w0 = 2*pi*F0/Fs;   % normalized angular frequency
n = 0:N-1;
x = 2*sin(w0*n);
figure(1), stem(n,x), grid
```

2. Represent graphically, using the **plot** function, the analog signal obtained from the previous discrete signal through digital to analog conversion (F_s = 4kHz). Represent the analog signal in absolute time (ms) and compute the period and the normalized frequency of the analog signal.

```
F0 = 300; Fs = 4000;
Tmax = 10^(-2);      % signal duration 10 ms
Ts = 1/Fs;
t = 0:Ts:Tmax-Ts;
xa = 2*sin(2*pi*F0*t);
figure(2), plot(t,xa,'-o'),xlabel('t[ms]'), grid
```

Remark: Having the discrete signal's samples, the analog signal can be represented using `plot`, taking as parameters the absolute time in [ms] and the vector with the discrete samples. You do not need to compute the analog signal, you can rebuild it directly:

```
%xa = 2*sin(2*pi*F0*t);
figure(3),
subplot(121),plot(n,x,'-or','markersize',10)
subplot(122),plot(t,xa,'-*b','markersize', 15)
```

1.3. Linear convolution of discrete-time signals

Linear convolution of two discrete-time sequences $x_1[n]$ and $x_2[n]$ (assumed as infinite length sequences) is defined as:

$$x[n] = x_1[n] * x_2[n] = \sum_{k=-\infty}^{\infty} x_1[k] \cdot x_2[n-k] \tag{2.6}$$

In MATLAB, the sequences are arrays of finite length *N* starting at index 1. Therefore the above formula is changed to:

$$x[n+1] = \sum_{k=0}^{N-1} x_1[k+1] \cdot x_2[n-k] \tag{2.7}$$

The MATLAB function to implement convolution is:

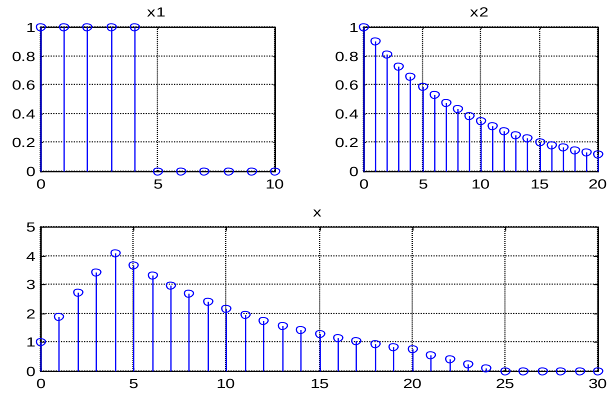
`y=conv(x1,x2)`

- returns in the vector `y` the result of linear convolution of the vectors `x1` and `x2`. The resulting vector length is equal to length of vector `x1` plus length of vector `x2` minus 1.

***Example*:**

Compute and plot the linear convolution of the sequences $x_1[n] = u[n] - u[n-5]$ ($0 \le n \le 10$) and $x_2[n] = (0{,}9)^n$ ($0 \le n \le 20$).

```
x1=treapta(0,10,0)-treapta(0,10,5);
n=0:20;
x2=0.9.^n;
x=conv(x1,x2);
subplot(2,2,1),stem(0:10,x1),title('x1'),grid
subplot(2,2,2),stem(n,x2),title('x2'),grid
subplot(2,1,2),stem(0:length(x)-1,x),title('x'),grid
```



E2. ***Exercises*:**

Compute and plot the linear convolution of the following sequences:

1. $x_1[n] = u[n-5] - u[n-15]$ $\quad x_2[n] = |10-n|$ $\quad$ for $0 \le n \le 20$
2. $x_1[n] = \sin(0.1 \cdot \pi \cdot n)$ $\quad x_2[n] = \delta[n-5]$ $\quad$ for $0 \le n \le 20$

1.4. Discrete Fourier Transform

The *Discrete Time Fourier Transform* (DTFT) of a sequence $x[n]$ is defined by:

$$X(e^{j\omega}) = \sum_n x[n]e^{-j\omega n} \tag{2.8}$$

where ω is the normalized angular frequency: $\boxed{\omega = 2\pi \frac{F}{F_S}}$.

F is the un-normalized frequency (measured in Hz) and F_S is the *sampling frequency*.

Also, the normalized frequency is: $\boxed{f = \frac{F}{F_S}}$.

The function $X(e^{j\omega})$ is periodical of period 2π, so it is sufficient to know the behavior in the interval $[-\pi, \pi)$ (*base interval*). Because this function is defined over ω, a continuous variable which can take an infinite number of values, the implementation on a computing machine is not possible.

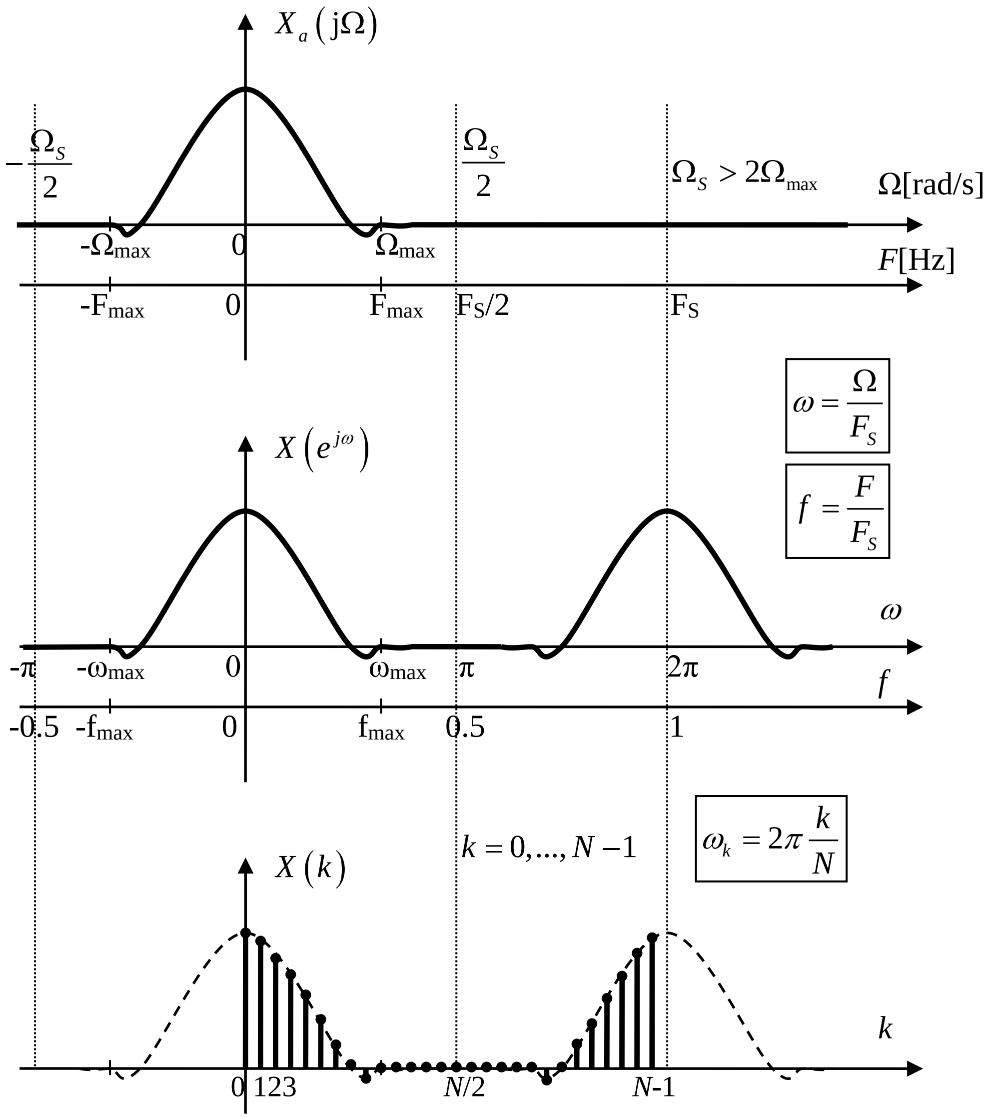
However, in order to achieve a frequency analysis, the *Discrete Fourier Transform* (DFT) is used, computed by replacing ω over the interval $[0, 2\pi)$ with N uniformly distributed points:

$$\boxed{\omega_k = 2\pi \frac{k}{N}}, \quad \text{with} \quad k = 0,1,\ldots,N-1.$$

Therefore, the *Discrete Fourier Transform* of a sequence $x[n]$ is defined by the relation:

$$X[k] = \sum_{n} x[n] e^{-j\frac{2\pi}{N}kn} \qquad \text{with} \quad k = 0,1,\ldots,N-1 \qquad (2.9)$$

The figure below shows the spectrum of a discrete-time signal representation, based on the normalized angular frequency or normalized frequency and the correspondence with the analog frequency. We also notice the correspondence between the DFT spectral components of index k and the spectrum represented in normalized angular frequency.



1. Discrete-Time Signals

In MATLAB, the function used to compute the discrete Fourier transform is the FFT function (Fast Fourier Transform) and it uses a fast algorithm for calculating the DFT.

***Syntax*:**

`y = fft(x)`

- <u>for a vector `x`</u> the result is the discrete Fourier transform (DFT) of vector `x`, returned in a vector `y` having the same length as `x`.
- <u>for a matrix `x`</u> the result is a matrix `y` of the same size to `x`, and the fft operation is applied to each column of the matrix `x`.

`y = fft(x,N)`

- is the N-point fft, padded with zeros if `x` has less than N points and truncated if it has more.

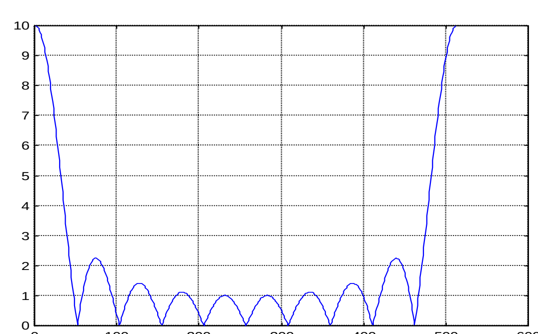
***Examples*:**

Compute the discrete Fourier transform of the following sequence:

$$x_1[n] = u[n] - u[n-10],\ \text{for}\ \ 0 \le n \le 20.$$

Plot the magnitude, phase, real and imaginary parts for the computed Fourier transform.

```
x1=treapta(0,20,0)-treapta(0,20,10);
X=fft(x1);
figure(1),plot(X)
```
// yields a wrong graphical representation, since X has complex values
// the magnitude of the Fourier transform can be represented using absolute values
```
figure(2),plot(abs(X))
```
// the length of the computed Fourier transform is the same as the length of the vector x1
// for a better representation, we compute the Fourier transform in a large number of points (N = 512):
```
X1=fft(x1,512);
figure(3),plot(abs(X1)),grid
```

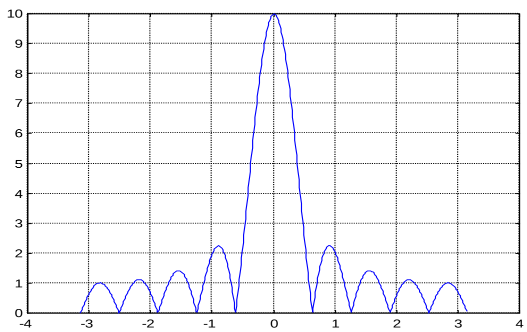


This representation corresponds to the frequency interval $[0,2\pi)$. Usually we want a symmetrical representation of the spectrum in the interval $[-\pi,\pi)$. Since the discrete Fourier transform is periodic of period 2π, then the representation in the interval $[-\pi,0)$ corresponds to the representation in interval $[\pi,2\pi)$.

A reversal of the two halves of the vector X1 must therefore be done, using the MATLAB function `fftshift`, which swaps the left and right halves of the vector X1. This is useful for visualizing the Fourier transform with the zero-frequency component in the middle of the spectrum.

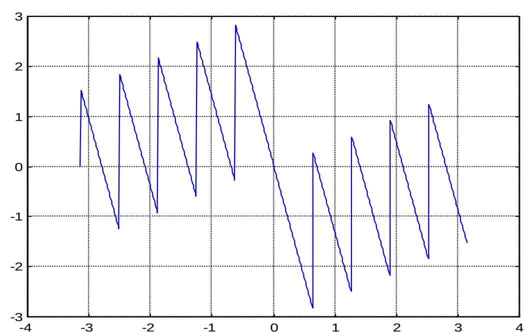
In addition, to have the representation over the abscissa within the interval $[-\pi,\pi)$, a linear step vector must be generated, having a total number of elements equal to the length of the represented spectrum vector:

```
w=-pi:2*pi/512:pi-2*pi/512;
figure(4),plot(w,fftshift(abs(X1))),grid
```

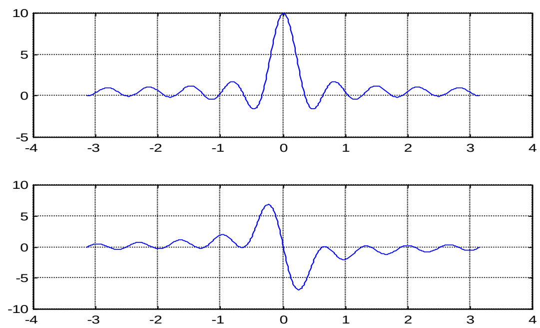


// Figure 4 plots the amplitude characteristics and figure 5 plots the phase characteristics using the function `angle()`.

```
figure(5),plot(w,fftshift(angle(X1))),grid
```



// Finally, we represent the real and imaginary parts of the DFT:
```
figure(6)
subplot(2,1,1),plot(w,fftshift(real(X1))),grid
subplot(2,1,2),plot(w,fftshift(imag(X1))),grid
```



Comments:

- The spectrum representation was performed according to the normalized angular frequency $\omega \in [-\pi, \pi)$.
- The values computed by `X1=fft(x1,512)` represent the spectrum computed in $\omega_k = 2\pi \dfrac{k}{N}$, with $k = 0,1,\ldots,N-1$. The number of points N used to calculate the FFT determines the spectral resolution. If N is bigger, the spectrum approximation of $X(e^{j\omega})$ is more accurate.
- If the $x[n]$ sequence length is shorter than N, the sequence is filled with zeros, the calculated spectrum representing the convolution between the spectrum of infinite length sequence function and the spectrum of the square gate sequence.
- The spectrum computed by `fft` can also be represented as a function of:
    - DFT index: $k = 0,1,\ldots,N-1$.
    - Normalized frequency: $f = \dfrac{F}{F_S} = \dfrac{\omega}{2\pi}$, $f \in [-0.5, 0.5)$.

      `f = -0.5:1/N:0.5-1/N;`
    - Un-normalized frequency $F = f \cdot F_S$ [Hz], $F \in [-F_S/2, F_S/2)$.

***Example*:**

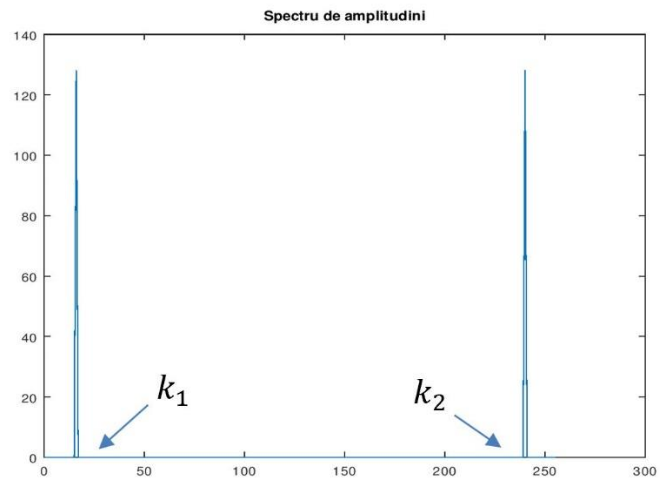
4. a) Generate the discrete sequence s[n] obtained by sampling with the sampling frequency $F_S = 8\text{kHz}$ of the signal $s(t) = \sin(2\pi F_0 t)$, with frequency $F_0 = 500\text{Hz}$ and duration $t_{MAX} = 40\text{ms}$. How many samples does the discrete sequence have?

```
Fs = 8000; F0 = 500; t_max = 0.04;
t = 0:1/Fs:t_max-1/Fs;
s = sin(2*pi*F0*t);
No_samples = length(s);  % or: No_samples = t_max*Fs
```

// the sequence $s[n]$ is obtained by 'walking' with a step size of $T_s = {}^1/_{F_s}$ over a t_{MAX} time span (seconds).

b) Compute the DFT of the sequence in **N=256** points using `fft` and plot the amplitude spectrum (modulus of the DFT) as a function of **k=0:N-1**.

```
N_fft = 256;
S = fft(s,N_fft);
k = 0:N_fft-1;
figure(1),plot(k,abs(S))
```



// in Figure 1, two Dirac impulses can be noticed, which corresponds theoretically to a Fourier transform of a sine:

$$\mathcal{F}\{\sin(\omega_0 t)\}(\omega) = \frac{i}{2}[\delta(\omega+\omega_0) - \delta(\omega-\omega_0)]$$

Let us therefore compute the normalized angular frequencies corresponding to the 2 Dirac impulses, knowing that the representation is done in a $[0, 2\pi)$ period. We get the sampling indexes for the Fourier transform, k_1 and k_2, matching the Dirac impulses. It can be easily verified from the graphical representation that the computed values are correct:

$$\omega_{k_1} = 2\pi\frac{k_1}{N_{fft}} = \omega_0 = 2\pi f_0 = 2\pi\frac{F_0}{F_s} \quad \Longrightarrow \quad k_1 = \frac{N_{fft}F_0}{F_s} = 16$$

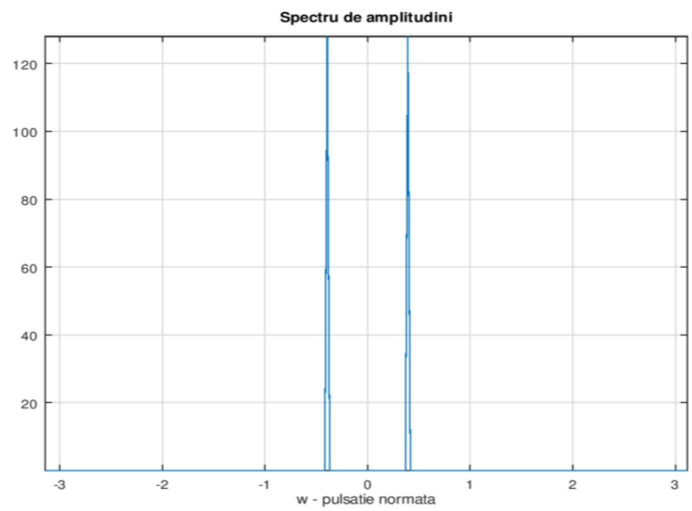
$$\omega_{k_2} = 2\pi \frac{k_2}{N_{fft}} = 2\pi - \omega_0 = 2\pi(1 - f_0) = 2\pi \frac{F_s - F_0}{F_s} \qquad \Longrightarrow$$

$$k_2 = \frac{N_{fft}(F_s - F_0)}{F_s} = 240$$

c) Plot the amplitude spectrum as a function of the normalized angular frequency $\omega \in [-\pi, \pi)$.

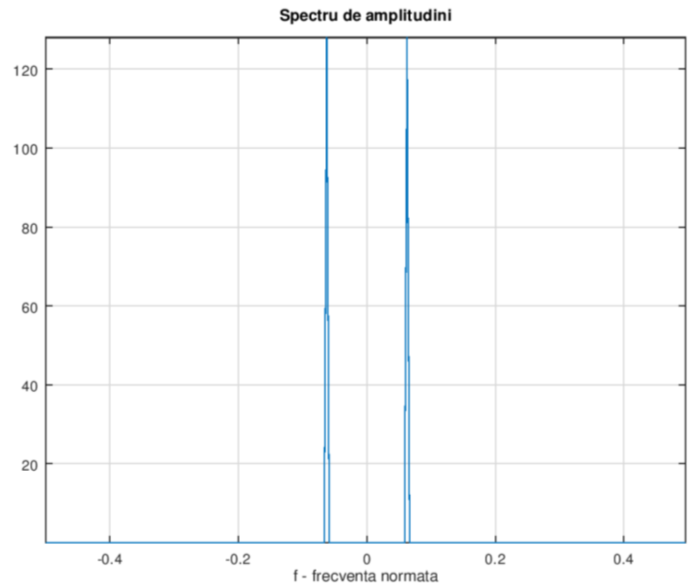
```
w = -pi: 2*pi/N_fft :pi-2*pi/N_fft;
figure(2),plot(w,fftshift(abs(S))),grid
```

// The amplitude spectrum is represented in Figure 2 in the base period $[-\pi, \pi)$. Using the *Zoom in* tool from the figure menu, it can be checked that the normalized angular frequency $\omega_0 = 2\pi F_0/F_s = 0.39$ is the correct value for the Dirac impulses.



d) Plot the amplitude spectrum as a function of the normalized frequency $f \in [-0.5, 0.5)$.

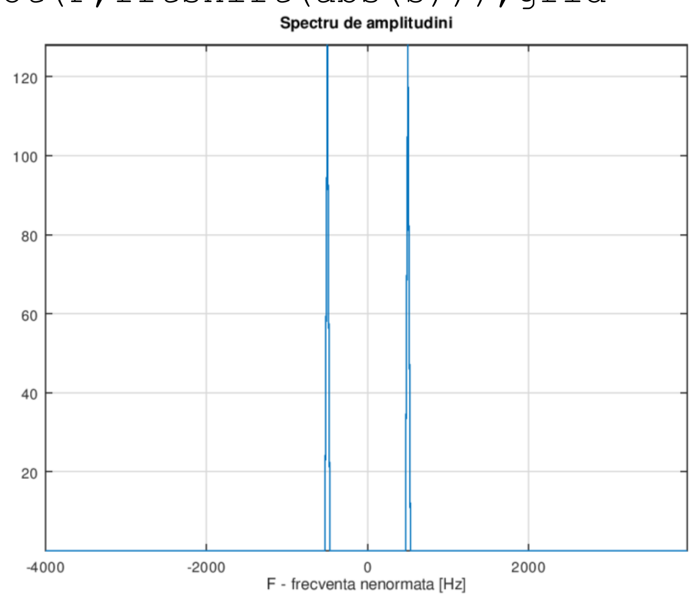
```
f = -0.5: 1/N_fft :0.5-1/N_fft;  % sau: f = w/(2*pi)
figure(3),plot(f,fftshift(abs(S))),grid
```



// The amplitude spectrum is represented in Figure 3 as a function of normalized frequency $f = \frac{F[Hz]}{F_s}$, so that the unity impulses appear on the horizontal axes at $f_0 = 0.0625$.

e) Plot the amplitude spectrum as a function of the un-normalized frequency [frequency in Hz].

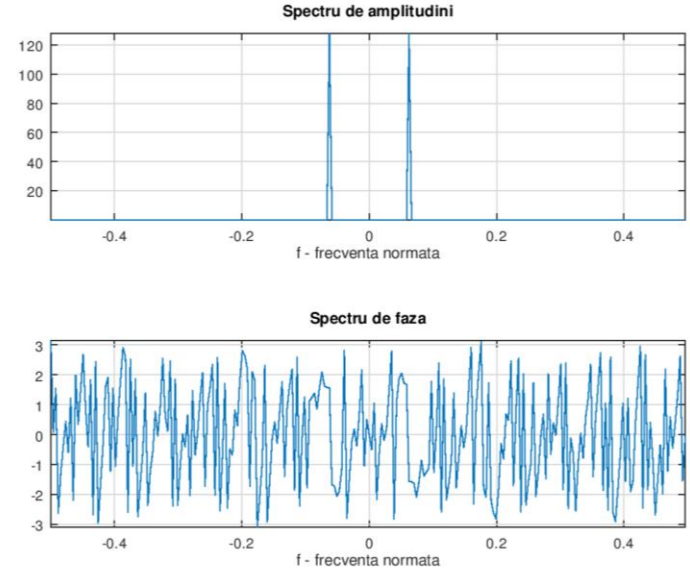
```
F = -Fs/2: Fs/N_fft :Fs/2-Fs/N_fft; %sau:F=w/(2*pi)*Fs
figure(4),plot(F,fftshift(abs(S))),grid
```



// The amplitude spectrum of the original sine wave is represented in Figure 4 as a function of not-normalized frequency, in Hz, so that the Dirac impulses appear on the horizontal axes at F_0.

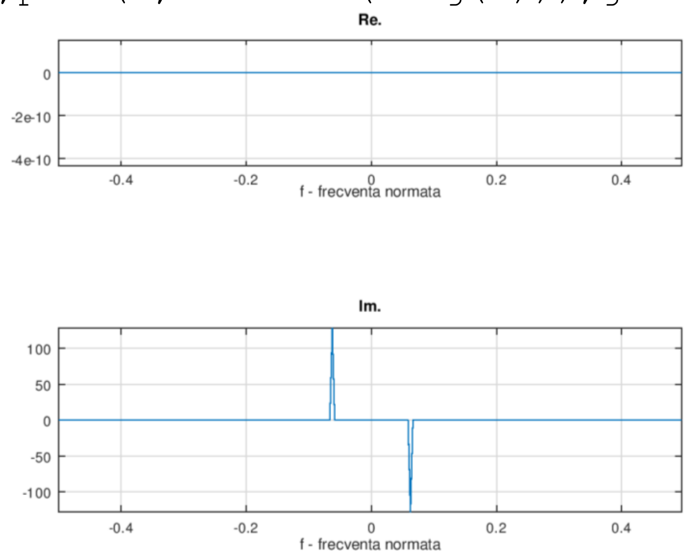
f) Plot the amplitude and the phase spectrum as a function of the normalized frequency on the same figure, using subplot.

```
figure(5),subplot(211),plot(f,fftshift(abs(S))),grid
subplot(212),plot(f,fftshift(angle(S))),grid
```



g) Plot the real and the imaginary part of the DFT as a function of the normalized frequency on the same figure, using subplot.

```
figure(6),subplot(211),plot(f,fftshift(real(S))),grid
subplot(212),plot(f,fftshift(imag(S))),grid
```



Each spectrum should be represented in a separate figure using the function `figure()`. Also, the plots should be accompanied by `grid`, `title` and `xlabel`.

E3. ***Exercise:***

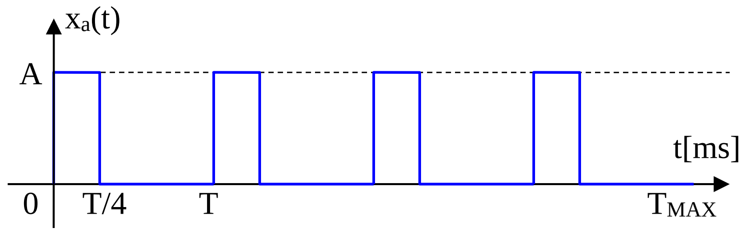
For the program from the previous exercise, modify the following parameters and explain the changes that occur in the spectrum representation:

a) The signal's frequency $F_0 = 200\text{Hz}$.

b) The number of points of the DFT **N=1024** (for the initial signal with $F_0 = 500\text{Hz}$).

E4. ***Homework*:**

Consider the analog signal $x_a(t)$ from the figure with the following parameters: frequency $F_0 = 400Hz$, amplitude $A = 2$, acquisition duration $T_{MAX} = 50ms$.



a) Generate the discrete signal $x(n)$ resulted from the sampling of $x_a(t)$ using the sampling frequency of $F_s = 8kHz$. *Indication:* use the **square** function (`help square` in *Command Window*).

Represent graphically using `stem` (n on the horizontal axes) the discrete signal $x(n)$.

Compute: the total number of samples L for $x(n)$, the number of samples N in the T period of time, the number of k periods in the total acquisition time T_{MAX}.

Represent graphically using `plot` (time in milliseconds on the horizontal axes) the analog signal $x(t)$ reconstructed from the discrete signal through digital to analog conversion.

b) Compute the DFT of $x(n)$ in a total number of L points equal to the signal length. Represent the amplitude and phase spectrum in normalized frequencies.

Compute the normalized frequency f_0 and check that value in the graphical representation, then check also the values matching the harmonic components in the signal; F_0 is the fundamental frequency.

Represent the amplitude spectrum $|X(k)|$ as a function of index k in the DFT.

Compute the index k_0 matching the fundamental frequency F_0. What is the relation between the normalized frequency f_0 and k_0? What about the relation between k_0 and the number of *k* periods obtained previously at point a) ? Explain.
Represent the amplitude spectrum as a function of frequency in Hz. Determine from the graphical representation the amplitudes of the spectral components matching the continuous component, the fundamental F_0 and the harmonics. On what frequencies do the harmonics appear? What is the relation between the amplitude A of the signal and the amplitudes measured in the figure?
Each figure will have title and labels.

c) Repeat point b) for the number of points in the DFT: $N_{fft} = 256$ and for $N_{fft} = 512$. Explain the differences between the spectra obtain at b) and c) !

d) Write a function that computes the duty cycle of the square signal (the percent of the period in which the signal is positive) and the signal energy, using the discrete samples of $x(n)$. Call that function from the main script and display the results.