Fourier Transform Using MATLAB

1.1. Fourier Transform for Continuous-Time Signals

In the first paper, techniques for analyzing periodic signals over time were discussed using cosine or sine functions to express periodic signals as a linear combination of these functions. However, the majority of signals we work with are aperiodic. Therefore, to analyze the spectrum of these signals in the frequency domain, the Fourier Transform is utilized.

Given a non-periodic signal $x(t)$, we ask ourselves in which frequency domain this signal exists, or, in other words, what combination of frequencies is necessary to reconstruct the signal using only standard cosine functions. Knowing already how a periodic signal is represented as the sum of standard cosine functions, we will construct a periodic signal from the non-periodic one by repeating the non-periodic signal with a period T. Using this procedure will be ultimately proven that the Fourier transform of a non-periodic signal $x(t)$ is:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt \tag{1}$$

The Fourier Transform is a continuous function of frequency. If we know the expression of the function $X(\omega)$, to reconstruct the original signal $x(t)$, we will use the inverse Fourier Transform using the formula:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\,d\omega \tag{2}$$

The conditions for the existence of the Fourier transform are:

a. The signal $x(t)$ must be absolutely integrable;
b. If $x(t)$ is a discontinuous signal, it must have a finite number of discontinuities within any time interval;
c. The signal $x(t)$ must have a finite number of minima and maxima within any time interval.
d. For various common signals, a list of Fourier transforms can be found here http://www.thefouriertransform.com/pairs/fourier.php.

1.2. Discrete Fourier Transform

Given that the Fourier transform is the result of a mathematical operation involving integration that cannot be analytically solved by a computing system, we will need to use various numerical methods to calculate it. Thus, to compute the Fourier transform using MATLAB, we will employ what is called the Discrete Fourier Transform, which approximates the Fourier transform.

The discrete-time Fourier transform of a sequence x[n] = x(nT_s) is given by the relation:

$$X\left(\frac{kF_s}{N}\right) \cong T_s \sum_{n=0}^{N} x(nT_s)e^{-j2\pi kn/N} \qquad (3)$$

where:

- $T_s = \frac{1}{F_s}$ is the sampling period of the signal $x(t)$, chosen so that the signal does not exhibit significant variations within the time interval T_s;
- k is the summation index;
- N represents the number of sampling points, so that the duration of the signal is $N \cdot T_s$. The larger N is, the better the approximation of the continuous frequency spectrum.

In MATLAB, to compute the Discrete Fourier Transform, the function **`fft()`** is used. Its name is an abbreviation for *Fast Fourier Transform*, indicating that it uses a fast algorithm. The **`fft()`** function is described in MATLAB here

https://www.mathworks.com/help/MATLAB/ref/fft.html

```
X = fft(x, N)
```

- **`X`** contains the values of the discrete Fourier transform applied to the elements of the vector **`x`**;
- length of the Fourier transform of length **`N`**.

The reference range for frequency representation is [0, Fs). However, it is usually desired to represent within the base interval [−Fs/2, Fs/2). Given that the discrete Fourier transform is periodic with a period of Fs, the representation in the interval [−Fs/2, 0) corresponds to the representation in the interval [Fs/2, Fs). Therefore, an inversion of the two halves of the vector X must be performed using the function **`fftshift()`**.

The function **`fftshift()`** is described in MATLAB here:

https://www.mathworks.com/help/matlab/ref/fftshift.html

```
Xshift = fftshift(X)
```

- **`Xshift`** is the vector obtained by swapping the two halves of the vector **`X`** among them;

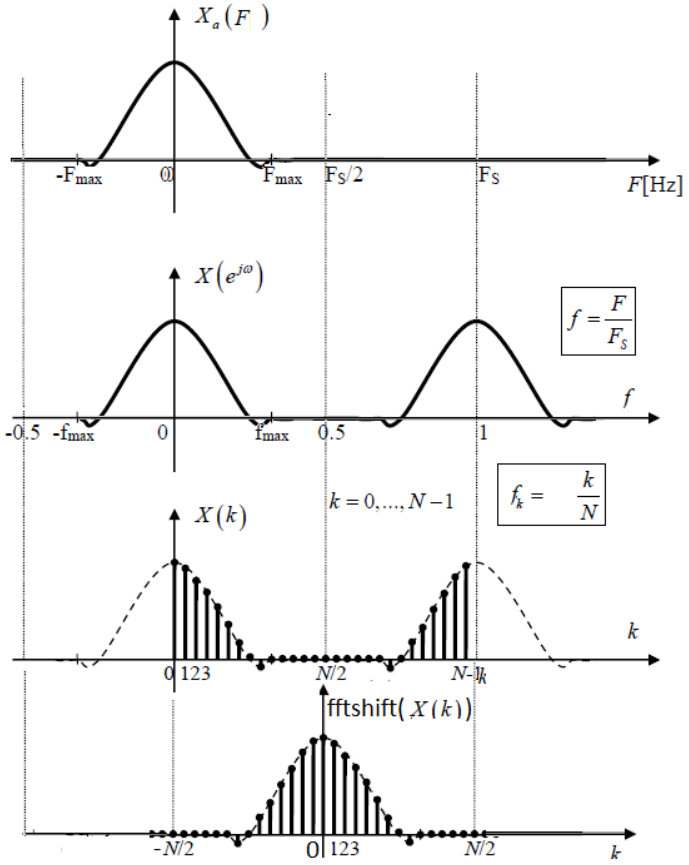

Fig. 1. Representations of the spectrum of a discrete signal as a function of normalized frequency and its correspondence to analog frequency.
The correspondence between the spectral components of index *k* calculated with DFT and the spectrum represented in normalized frequencies.

Example 1

Using MATLAB, calculate and plot the magnitude and phase of the Discrete Fourier Transform of the sequence:

$$x(t) = \begin{cases} t(1-t) & ,0 \le t < 1 \\ 0 & ,1 \le t < 2 \end{cases}$$

```
%% Calculating the Fourier transform using 32 samples in
the interval %% 0-2 seconds

close all;
clear
clc
```

```
Fs = 16; % sampling frequency
Ts = 1/Fs; % sampling period
N = 512; % Length of the Fourier transform
df = Fs/N; % frequency domain resolution
Tmax = 2; % signal duration
% vector of time values at the sampling instants
t = 0:Ts:Tmax-Ts;
% vector of function values at the sampling instants
x = t.*(1-t).*(sign(t)-sign(t-1))/2;

X = Ts*fft(x,N); % discrete Fourier transform
F = 0:df:Fs/2-df; % values of the frequency index

% Xshift = Ts*fftshift(fft(x,N));
% Fshift = -Fs/2:df:Fs/2-df;

%The plots
subplot(3,1,1);
p = plot(t,x,'k');
set(p,'LineWidth',2); grid on;
xlabel('Time, t(s)');
ylabel('x(t)');
subplot(3,1,2);
p = plot(F,abs(X(1:N/2)),'k');
% p = plot(Fshift,abs(Xshift),'k');
set(p, 'LineWidth',2); grid on;

xlabel('Frequency, F(Hz)');
ylabel('IX(f)I')
subplot(3,1,3);
p = plot(F,angle(X(1:N/2)),'k');
% p = plot(Fshift,angle(Xshift),'k');
set(p,'LineWidth',2); grid on;

xlabel( 'Frequency, F(Hz)');
ylabel('Phase X(f)');
```

After running the code, Fig. 2 is obtained, which includes the time representation of the signal x(t) and the magnitude and phase representation of the Discrete Fourier Transform X(f), calculated using the function **`fft()`**.

By commenting out the lines 15, 16, 28, 35:

- `X = Ts*fft(x,N);`

- `F = 0:df:Fs/2-df;`
- `p = plot(F,abs(X(1:N/2)),'k');`
- `p = plot(F,abs(X(1:N/2)),'k');`

and uncommenting the lines 18, 19, 29, 36:

- `Xshift = Ts*fftshift(fft(x,N));`
- `Fshift = -Fs/2:df:Fs/2-df;`
- `p = plot(Fshift,abs(Xshift),'k');`
- `p = plot(Fshift,angle(Xshift),'k');`

the magnitude and phase of the Discrete Fourier Transform X(f) are obtained after swapping the two halves of the vector X using the function **`fftshift()`**. The obtained plots can be observed in Fig. 3.

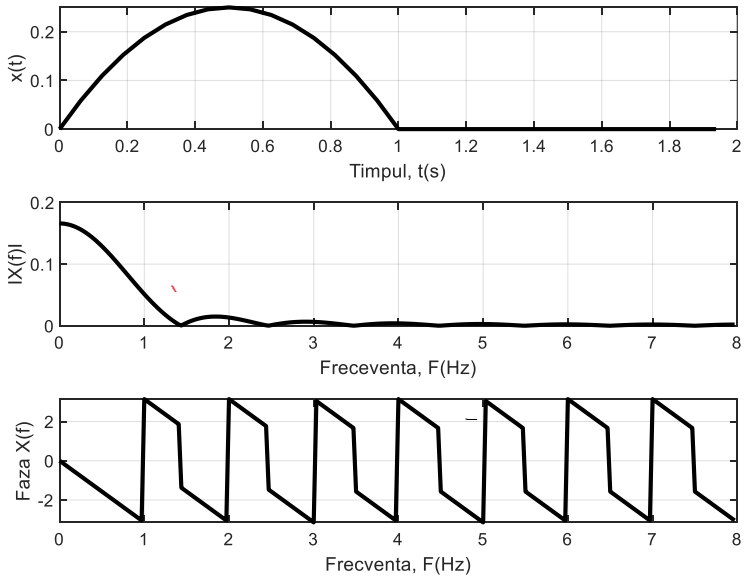

Fig. 2. The plots of Example 1 using **`fft()`**

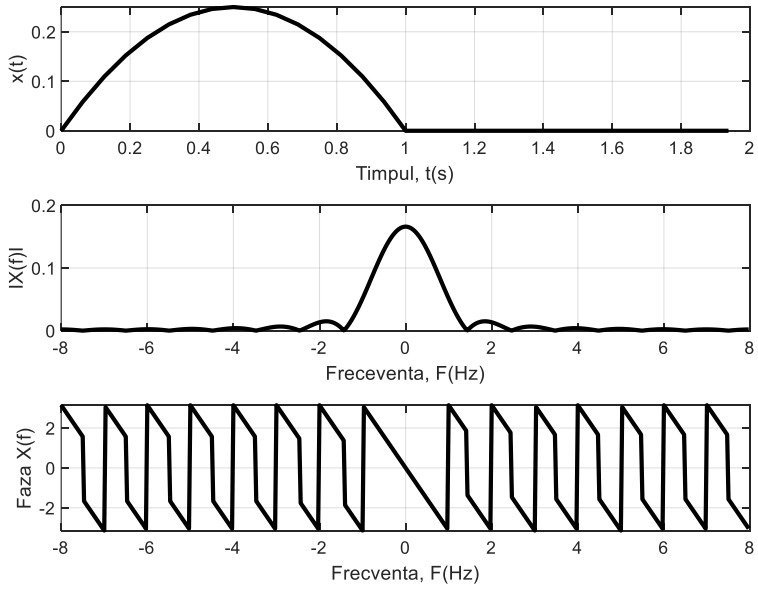

Fig. 3. The plots of Example 1 using **`fftshift()`**

Proposed exercise 1

Write a MATLAB program that displays the magnitude and phase of the Discrete Fourier Transform of the signal:

$$x(t) = \begin{cases} 9 - (t-3)^2 & , 0 < t < 3 \\ 0 & , \text{for the rest} \end{cases}$$

1.3. Spectrum of Periodic Signals Using MATLAB

As mentioned in the first laboratory, common signals encountered in practice have a moment of appearance and a moment of disappearance, which means rigorously periodic signals do not exist in practice.

However, in certain situations, it is useful to model a signal of finite duration, having a periodic-like variation throughout its existence, through a periodic time function. This modeling does not lead to errors if the signal's duration is much larger than the repetition period and the duration of transient regimes occurring in the circuit when the signal is applied or suppressed. Additionally, if these transient regimes are not of particular interest.

Example 2

Using Matlab, plot the time-domain signal below and its amplitude spectrum, in the presence and absence of noise:

$$x(t) = \begin{cases} 0.6 \cdot \sin(2\pi \cdot f_1 \cdot t) + \sin(2\pi \cdot f_2 \cdot t), 0 < t < 50\text{ms} \\ 0, \text{for the rest} \end{cases}$$

```
clear
clc
close all

prompt = ' Case =1 analysis of the signal without noise.
Case 2: signal with noise. What is the value of the case?
:';
caz = input(prompt);
%%case =1 signal without noise
%%case =2 signal with noise

F1 = 50;
F2 = 120;
```

```
Fs = 1000;  % sampling frequency
Ts = 1/Fs;   % sampling period
N = 1024;  %length of the Fourier transform
df = Fs/N;  % Fourier transform resolution
Tmax = 0.05; %signal duration
t = 0: Ts :Tmax-Ts;        % time vector
F = 0:df:Fs/2-df; % frequency vector
%% Signal composed of two sinusoidal signals with f1=50Hz
and f2=120Hz.
x = 0.6*sin(2*pi*F1*t) + sin(2*pi*F2*t);
if caz == 1
plot(1000*t,x),grid
title('The noise-free signal')
xlabel('t (ms)')
ylabel('x(t)')

%% pause is used if a pause in the program execution is
desired, and any key press resumes the program execution
% pause

%%FFT applied to S
X = Ts*fft(x,N);
df = Fs/N;
% frequency vector:
figure (2)
plot(F,abs(X(1:N/2))),grid
title('Amplitude spectrum for x(t)')
xlabel('F (Hz)')
ylabel('|X(F)|')
else
%% Disturbing the signal with a noise signal.
xzg = x+0.5*randn(1,length(t));
plot(1000*t,xzg),grid
title('The noise-disturbed signal')
xlabel('t (ms)')
ylabel('x_zg(t)')

% pause

%%FFT applied to S
Xzg = Ts*fft(xzg,N);
figure (2)
```

```
plot(F,abs(Xzg(1:N/2))),grid
title('Amplitude spectrum for xnoisy(t)')
xlabel('F (Hz)')
ylabel('|X_zg(F)|')
end
```

After running the code by selecting case 1, the graphs in Fig. 4 are obtained, which include the time representation of the noise-free signal x(t) and its amplitude spectrum calculated using the function **`fft()`**.

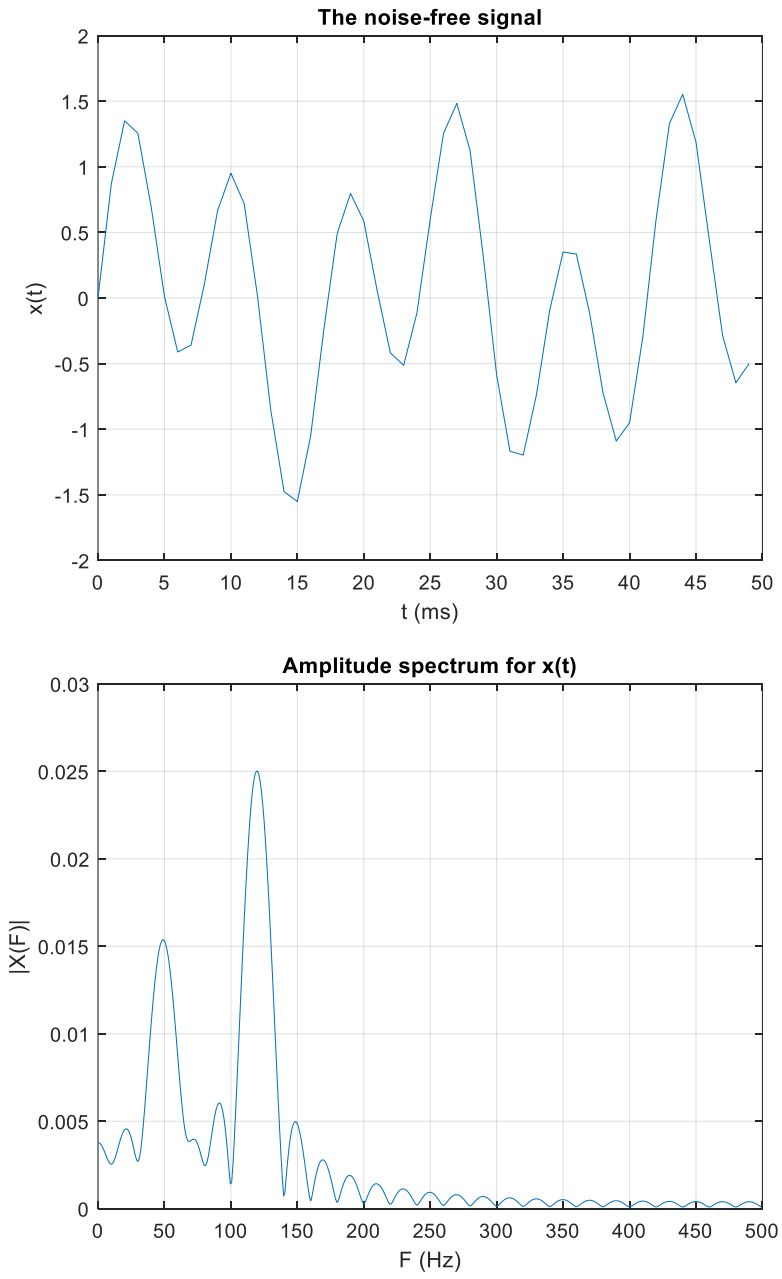

Fig. 4. The plots of Example 2 using case 1 (signal without noise)

After running the code by selecting case 2, the graphs in Fig. 5 are obtained, which include the time representation of the noise-disturbed signal x(t) and its amplitude spectrum calculated using the function **`fft().`**

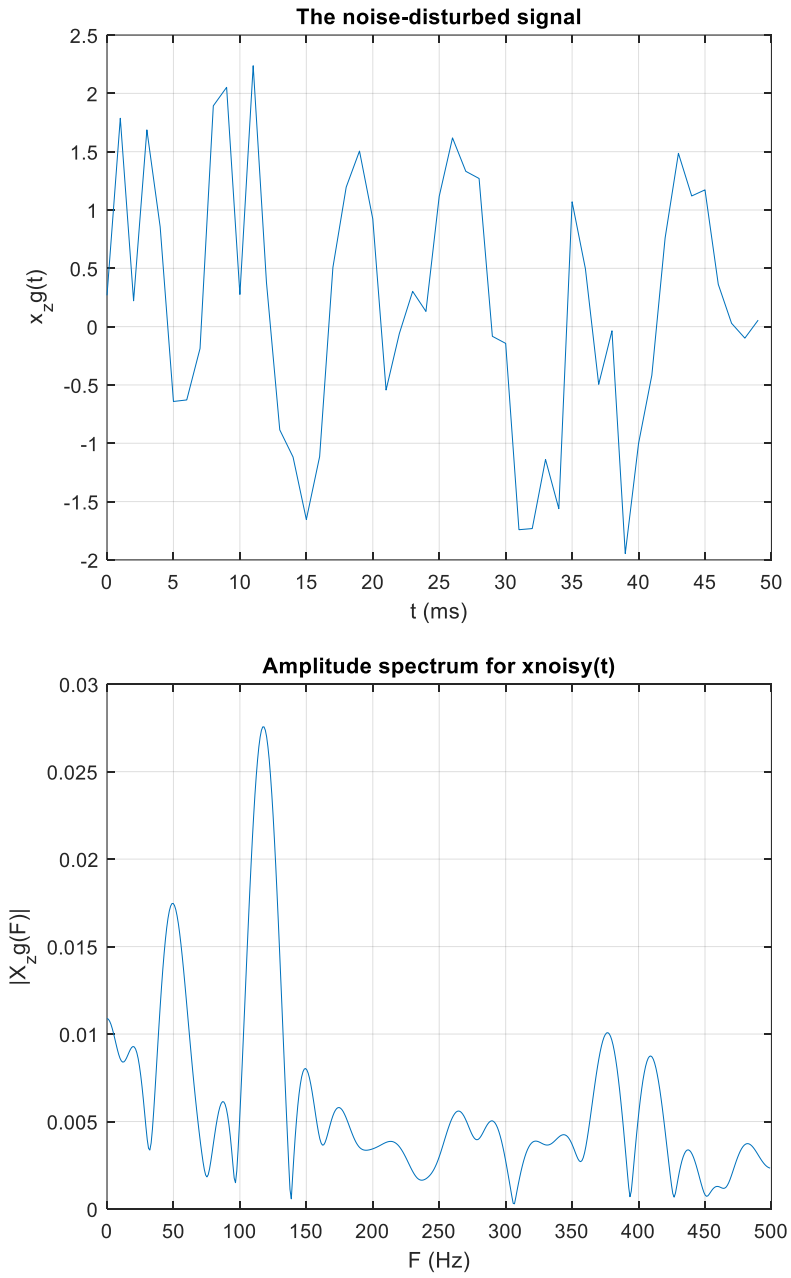

Fig. 5. The plots of Example 2 using case 2 (signal affected by noise)

Proposed exercise 2

Simulate the above program for a sampling frequency set to 500. What do you observe? Return to Fs=1000 and modify Tmax=0.5. What do you observe? How about if Fs=500 and Tmax=0.5, what is observed?

Proposed exercise 3

Using MATLAB, plot a periodic rectangular signal with a 50% duty cycle and its amplitude spectrum, in the presence and absence of noise (for the time representation, use the functions from the previous work). Change the duty cycle to 25% and 10%, respectively.

Proposed exercise 4

Using MATLAB, plot a triangular signal in the time domain and its amplitude spectrum, both in the presence and absence of noise (for the time representation, use the functions from the previous work).

Bibliography

1. Mateescu, Adelaida, Dumitriu, N., Stanciu, L., **Signals, Circuits, and Systems**, Teora, Bucharest, 2001.
2. Introduction to Signal Processing using Python / Valentin-Adrian Niţă, Radu Alexandru Badea, Răzvan-Eusebiu Crăciunescu. – Timişoara : Politehnica Publishing House, 2022