# Laboratory N.1-2

MICHELE TARTARI, LUCIA BERGANTIN, OLENA HRYSHAIENKO

January 4, 2018

## Abstract

In this report, referring to the first and second labs, we concentrate on the signal processing aspect of the course. The first lab's main aim is to master the concept of Fast Fourier Transform and implement it on a Matlab script. The second part focuses instead on the processing of an audio file and its coding and decoding.

## Contents



## 1. How to evaluate Fourier transform?

### 1.1. Theoretical study in the case of a sine wave

In point a) we express the $x$ signal using a cissoid and then calculate the Fourier Transform. After calculating its spectrum, we plot it obtaining two peaks in $+f_0$ and in $-f_0$ of magnitude $a/2$. In point b) we sampled the signal and then calculated the Discrete Fourier Transform. Since the Shannon condition is almost fulfilled because $f_s = 10 \cdot f_0$, for all f$f \in R$ such that $|f| < f_s/2$, its spectrum can be approximated to:

$$F_{cc}\left(\frac{k}{N_f}f_s\right) \approx \frac{1}{f_n}F_{DC} \cdot y\left(\frac{f}{f_s}\right)$$

In point c) we windowed the signal multiplying it for a rect function, expressing it by means of Dirichlet kernel. As in the previous point, its Discrete Fourier Transform spectrum can be approximated. In the plot we obtain cisoid peaks in $\lambda_0$, in $-\lambda_0$, in $1-\lambda_0$ and in $1+\lambda_0$. It has to be noticed that the spectrum is entirely positive since we're considering the absolute value. In point d) we express the sampled waveform $Y_n[k]$ by means of discrete comb $1_{N_f}$ ,0 where $N_f = N_t$ and there exists an integer $k_0$ such that $\lambda_0 = k_0/N_f$.

(a) Fourier transform of x:

$$x(t) = a\cos(2\pi f_0 t + \phi) = \frac{a}{2}\left(e^{i\phi + 2\pi f_0 t} + e^{-i\phi - 2\pi f_0 t}\right) =$$

$$= \frac{a}{2}\left(e^{i\phi} e^{2\pi f_0 t} + e^{-i\phi} e^{-2\pi f_0 t}\right)$$

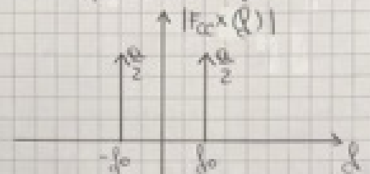
FOURIER TRANSFORM OF x

$$F_{cc}\, x(f) = \frac{a e^{i\phi}}{2}\,\delta(f - f_0) + \frac{a e^{-i\phi}}{2}\,\delta(f + f_0) =$$

$$= \frac{a}{2}\left(e^{i\phi}\delta(f - f_0) + e^{-i\phi}\delta(f + f_0)\right)$$

MAGNITUDE $|F_{cc}\, x(f)| = \left|\frac{a}{2}\right| \left|\delta(f - f_0) + \delta(f + f_0)\right|$

Magnitude spectrum plot:




(b) Discrete Fourier transform:

SAMPLING $x(nT_s) = a\cos(2\pi f_0 n T_s + \phi) = \frac{a}{2}\left(e^{i\phi} e^{2\pi f_0 n T_s} + e^{-i\phi} e^{-2\pi f_0 n T_s}\right)$

$$x_s[n] = \frac{a}{2}\left(e^{i\phi} e^{2\pi f_0 n} + e^{-i\phi} e^{-2\pi f_0 n}\right)$$

DISCRETE FOURIER TRANSFORM

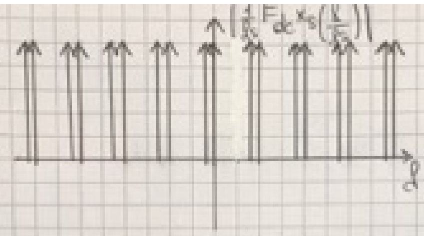
$$F_{dc}\, x_s(\lambda) = \frac{a e^{i\phi}}{2} \sqcup(\lambda - \lambda_0) + \frac{a e^{-i\phi}}{2} \sqcup(\lambda + \lambda_0) =$$

$$= \frac{a}{2}\left(e^{i\phi} \sqcup(\lambda - \lambda_0) + e^{-i\phi} \sqcup(\lambda + \lambda_0)\right)$$

MAGNITUDE $|F_{dc}\, x_s(\lambda)| = \left|\frac{a}{2}\right| \left|\sqcup(\lambda - \lambda_0) + \sqcup(\lambda + \lambda_0)\right|$

Since the Shannon condition is fulfilled (since $f_s = 10 > 2f_0$) then:

$$\left|\frac{1}{f_s} F_{dc}\, x_s\!\left(\frac{f}{f_s}\right)\right| \propto |F_{dc}\, x_s(\lambda)|$$




(c) Fourier transform of the windowed sampled signal $y = x_s \, \text{rect}_{N_t}$:

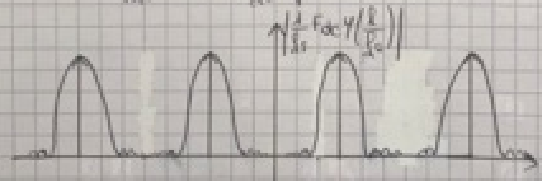
$$F_{dc}\, y(\lambda) = F_{dc}\, x_s(\lambda) \cdot F_{dc}\, \text{rect}_{N_t}(\lambda) =$$

$$= \frac{a}{2}\left(e^{i\phi}\, \text{Ш}(\lambda-\lambda_0) + e^{-i\phi}\, \text{Ш}(\lambda+\lambda_0)\right) \cdot D_{N_t} =$$

$$= \frac{a}{2}\left(e^{i\phi} D_{N_t} \text{Ш}(\lambda-\lambda_0) + e^{-i\phi} D_{N_t} \text{Ш}(\lambda+\lambda_0)\right)$$

Since the Shannon condition if fulfilled:

$$\left|\frac{1}{f_s} F_{dc}\, y\left(\frac{f}{f_s}\right)\right| \approx \left|F_{dc}\, y(\lambda)\right|$$




(d) Since the Shannon condition is fulfilled and the windowing effects are neglected:

$$Y_{N_t}[k] = F_{dc}\, y\left(\frac{k}{N_t}\right) = \frac{a}{2}\left(e^{i\phi} D_{N_t}\left(\frac{k}{N_t} - \frac{k}{N_t}\right) + e^{-i\phi} D_{N_t}\left(\frac{k}{N_t} + \frac{k}{N_t}\right)\right)$$

## 1.2. Numerical implementation

For a $N_t$-points data sequence and $\mathrm{N}_f$ frequencies, we have:

$$Y_{N_f}[k] = \sum_{n=0}^{N_t-1} y[n] \cdot e^{-j2\pi \frac{k \cdot n}{N_f}} \tag{1}$$

where $n \in [0, N_t - 1]$and $k \in [0, N_f - 1$. There are 3 different cases we have to consider:

1. if $N_f = N_t$: In this case, the computation is a Discrete Fourier Transform and the implementation of such transform can be done by means of either the Fast Fourier Transform algorithm (function fft in MATLAB) or by developing a similar algorithm ( function transffourier).

2. if $N_f > N_t$: In this case, "zero padding technique" will be used. We have: $n \in [0, N_t - 1]$, $k \in [0, N_f - 1$ and $N_f > N_t$. When $k \leq N_t - 1$, the algorithm uses the same computation as in the previous cases, on the other hand when $k > N_t - 1$ the values of $Y$ will be padded with 0. Thus, the new sequence has the same number of data points in the time domains as in the frequency domain after zero-padding, meaning $N_t = N_f$ .

3. if $Nf < Nt$: The Fast Fourier Transform algorithm can still be used in the case $N_f < N_t$. However the time signal is cut and only $N_f$ points are considered for computation of the Discrete Fourier Transform by the fft function. The aliasing phenomenom occurs and information might be lost. Therefore, in the computation of the Discrete Fourier Transform, it is important to make sure that $Nf \geq N_t$.

## 1.3. A new Matlab function

In this section we will try to obtain Furier Transform of $x$ where $x$ is a continuous-time signal, defiend between time 0 and $T$, and recorded with frequency $f_s$. We then wrote a MATLAB function transffourier which took as input:

1. a vector $y$ containing the $N_t$ values of the signal to process, where $N_t$ is an rounded value of $T$.

2. the number $N_f$ of frequencies,

3. the sampling frequency $f_s$.

The function output:

- $f$ a vector containing the frequencies for which the transform was calculated,

- $tfx$ the approximation of the Fourier transform of $x$.

```
function [f, tfx] = transffourier(y, Nf, fs)
%% Initialize
Nt = length(y);
f = zeros(Nf,1);
tfx = zeros(Nf,1);
k = 0:1:Nf-1;

%% Transform computation
f = k*fs/Nf;          %frequancy  vector
tfx = fft(y, Nf);
end
```

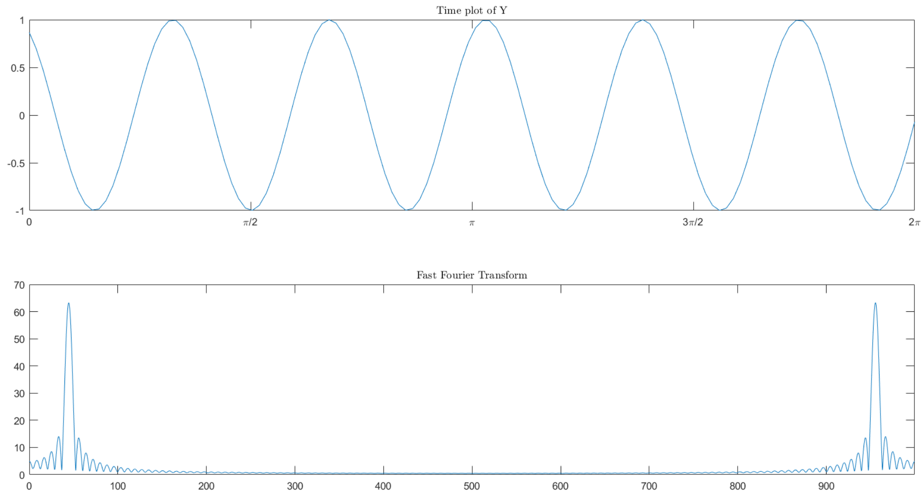
## 1.4. Sine wave spectral analysis

Given the function $x(t) = a * cos(2\pi \cdot f_0 \cdot t + \phi)$ sampled with frequency $f_s$ and windowed, and where $\phi$ is th pahse lag.
We obtain the signal

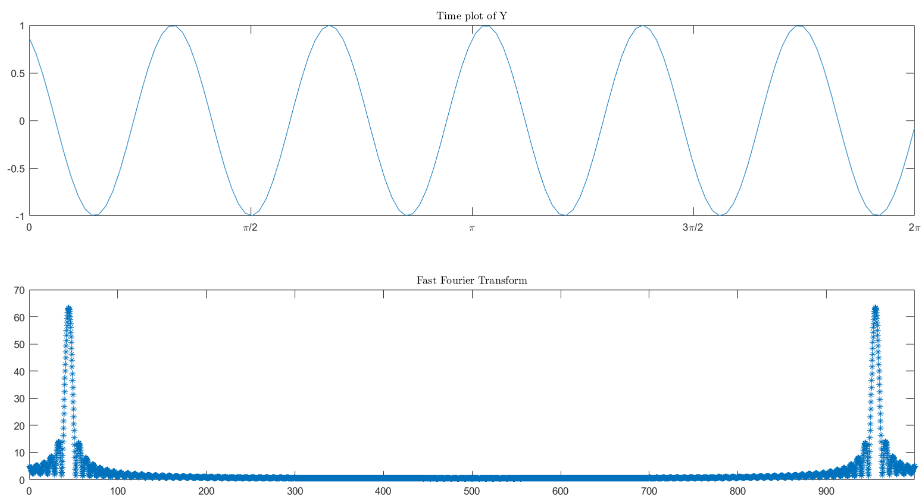
$$y[n] = \begin{cases} x(n/f_s) & \text{if } 0 \leq n \leq N_t - 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

and then use the code:

```
clear all
clc

%% Initialize
Nt = 128;       % timespan for windowing
fs = 1000;      % sampling frequency
Nf = 4096;      % number of frequencies

%% Define time function
a = 1;
phi = pi/6;
n = 0:Nt-1;
f0 = 5.7/128*fs;
y = a*cos (2*pi*f0*n/fs + phi ) ;


%% Compute Fourier transform
[f tfx] = transffourier(y, Nf, fs);

%% Plot
t = linspace(0, 2*pi, Nt);

figure(1)
subplot(2,1,1), plot(t, y);
hold on;
    title('Time plot of Y', 'interpreter','latex')
    xlim([0 max(t)]);
    xticks([0 pi/2 pi 3*pi/2 2*pi])
    xticklabels({'0','\pi/2','\pi', '3\pi/2','2\pi'})
hold off;

subplot(2,1,2), plot(f,abs(tfx));
    hold on;
        xlim([0 max(f)]);
        title('Fast Fourier Transform', 'interpreter','latex')
    hold off;
```

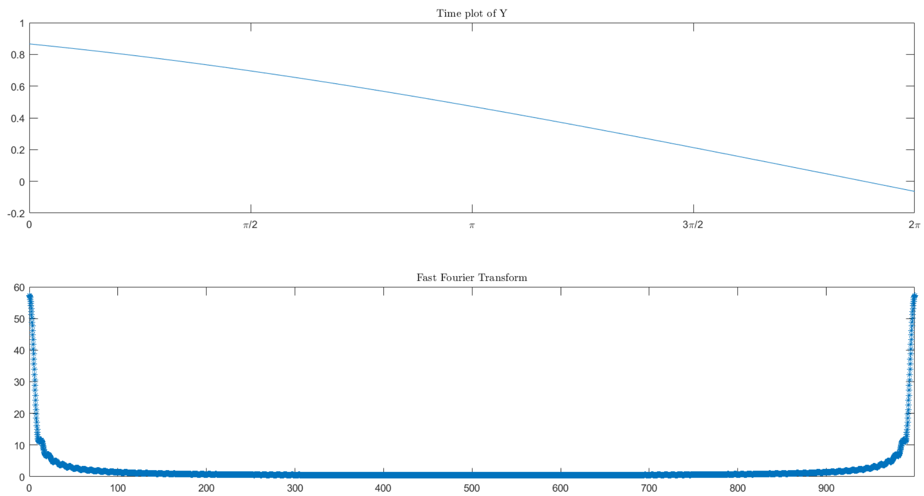



We then run again the code using $\star$ cross-marckers.




```
subplot(2,1,2), plot(f,abs(tfx), ':*');
```

Finally we modified $f_0$ to equal to $5.7/(N_t \cdot f_s)$ and plotted both $y$ as a function of time and its Fourier Transform magnitude.




# 2. A basic codec

## 2.1. Coder-decoder

In this task we write a Matlab script whose aim is to read the given "musique.wav" file using the audioread() function displaying at the same time the sampling frequency fs and the quantization nbits, play the sound using the function sound and encode the file using the given codeur.m script writing what obtained in a second file "musique2.wav".

```
clear all
clc

%% Audio1
[y1 fs] = audioread('musique.wav');
sound(y1, fs)

%% Audio1 File encoding
fmin = 1000; fmax = 8000;    % encoding boundary frequencies
[npt, echelle] = codeur(y1, fs, fmin, fmax, 'musique2.wav');
```

We then have to write a decoder.m function whose aim is to decode the file "musique2.wav" previously coded.

```
function [y, fs] = decodeur(fichier, fmin, fmax, npt, echelle)
%% Create a mask of audio file "fichier"
[masky fs] = audioread(fichier);

%% Compute the Fourier transform
% Scale the signal
masky = masky *echelle;

% Compute Fourier transf. between fmin and fmax
tf = masky(:,1) + j * masky(:,2);
```

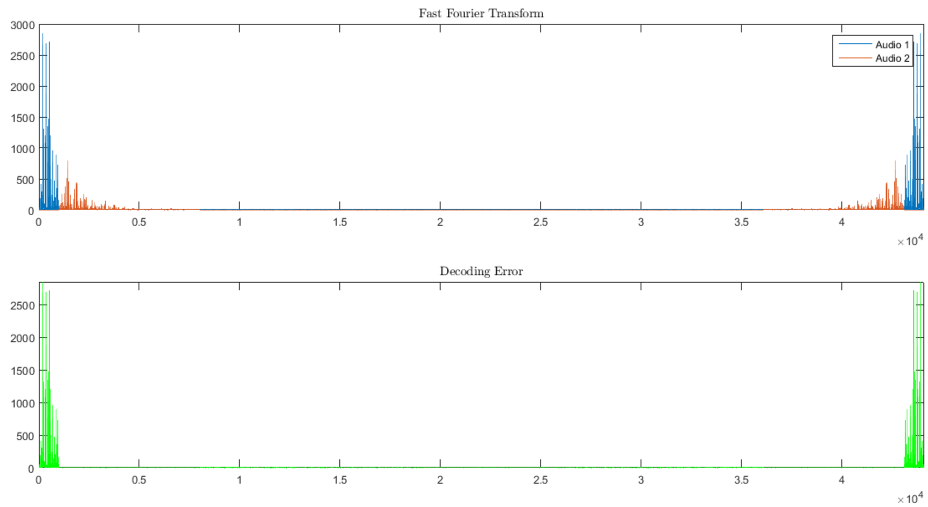
```

% Find kmin and kmax that are the position of fmin and fmax
%   inside the vector npt
kmin = round ( fmin*npt/fs )+1;
kmax = round ( fmax*npt/fs )+1;

% Zero-pad missing frequencies
tfy=[zeros(kmin-1 ,1);  tf;   zeros(npt-kmax,1)];

%% Compute inverse Fourier transform
y = ifft(tfy, 'symmetric');
```

The result is a file audio quite similar to the first one, but not exactly the same when played. This is due to the windowing of the signal that takes place between a set $f_min$ and $f_max$ and that removes the frequencies outsides these boundaries.

```
%% Audio 2 = Decoded file Audio 1
[y2, fs] = decodeur('musique2.wav', fmin, fmax, npt, echelle);
pause(6.5);                 % wait for the Audio 1 to finish playing
sound(y2, fs)
```

We then plot the Fast Fourier Transform of both signals, highlighting the difference ( decoding error) between them. As the cut out of $f > f_max$ and $f < f_min$ this too is an effect of windowing.

```
%% Error Computation (= Audio 1 - Audio 2)
err = abs(fft(y1)) - abs(fft(y2));

%% Plot
f = (0:npt-1)/npt*fs;

figure;
subplot(2,1,1), plot(f,abs(fft(y1)));
    hold on;
        plot(f,abs(fft(y2)));
        xlim([0 max(f)]);
        title('Fast Fourier Transform', 'interpreter','latex')
        legend('Audio 1','Audio 2')
    hold off;

subplot(2,1,2), plot(f,err,'g');
    hold on;
        xlim([0 max(f)]);
        ylim([min(err) max(err)]);
        title('Decoding Error', 'interpreter','latex')
    hold off;
```




## Conclusions

In part 1 we focused on the Fast Fourier Transform computation. In particular in part 1.1 we made the theoretical calculations on a generic cosine signal, while in part 1.2 we concentrated on the $Y_{N_f}[k]$ signal and its theoretical analysis for different values of $N_f$ compared to $N_t$. In part 1.3 and 1.4 we focused on the implementation of such calculations. In part 2 we concentrated on the analysis on a .wav audio file and its coding and decoding. We also analyzed the Fast Fourier Transform of the original file and decoded one, focusing in particular on their differences due to windowing.