Lab session 2 report

Basile Dubois Bonnaire & Yann Vincent

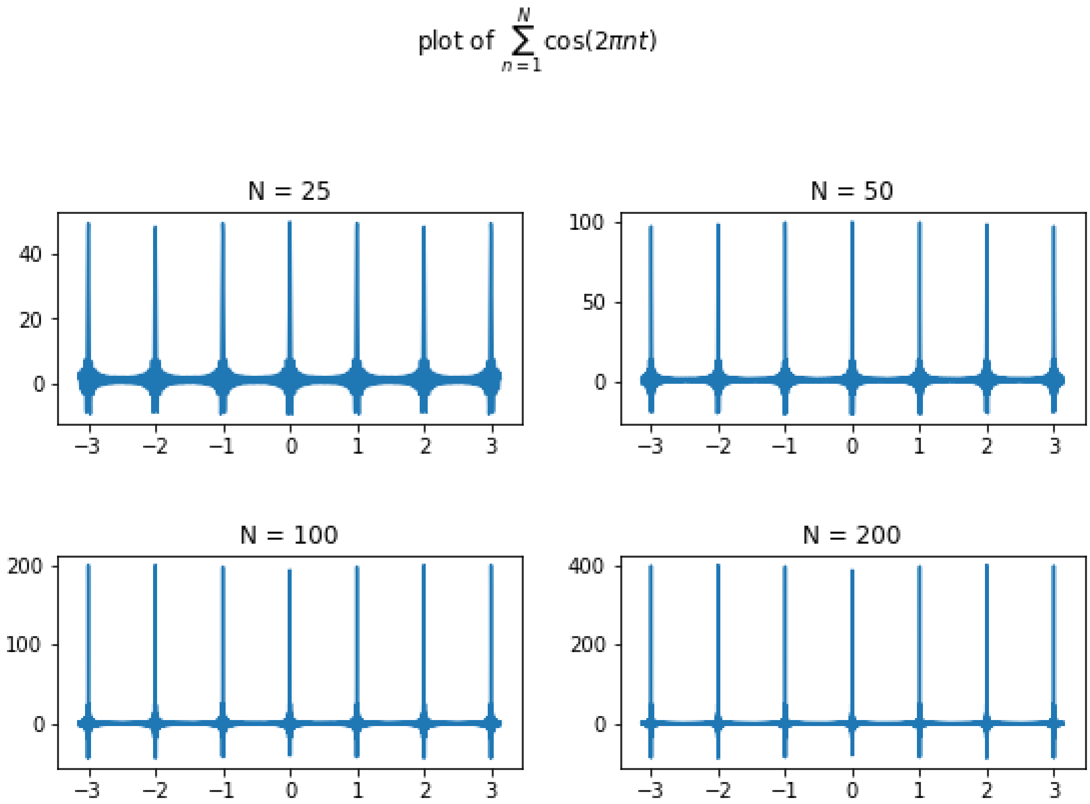
Exercice 1

Construction of the Dirac Comb

1.1

In [1]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

N=[25, 50, 100, 200]
fig, ax = plt.subplots(2,2)
ax.resize(1,4)

function = lambda n,t : 2*np.cos(2*np.pi*n*t)

def serie_calc(n,t):
    serie=0
    for i in range(n):
        serie += function(i,t)
    return serie



for n in N:
    x = np.linspace(-np.pi,np.pi, int(2*np.pi*n*7))
    ax[0,N.index(n)].plot(x,serie_calc(n,x), label="N = " + str(n))
    ax[0,N.index(n)].set_title("N = " + str(n))
fig.suptitle("plot of $\sum_{n=1}^N \cos(2 \pi n t)$")
fig.tight_layout()
fig.set_size_inches(8,6)

fig.show()
```



1.2

for each integer the sum tends towards infinity while it tends to 0 for other values.

1.3

Taking $t \in \mathbb{Z}$ we obtain $cos(2\pi nt) = 1$ so the serie diverges on $\mathbb{Z}$.
Thus it cannot converge point-wise on $\mathbb{R}$.

1.4

Considering $S_N(t) = \sum_{n=-N}^{N} e^{i2\pi nt}$, we have

$$
\begin{aligned}
S_N(t) &= 1 + \sum_{n=1}^{N} e^{i2\pi nt} + e^{-i2\pi nt} \\
&= 1 + \sum_{n=1}^{N} \cos(2\pi nt)
\end{aligned}
$$

Which is our previous sum.
Thus if the sum were to be absolutely convergeant we could write it that way.

1.5

Considering the dirac comb at point 1, We have the relation $\hat{\Delta}_1 = \Delta_1$.
However $\hat{\Delta}_1 = \sum_{n\in\mathbb{Z}} e^{i2\pi nt}$ which is equal to our serie.

Thus the serie is in fact the Dirac comb centered on 1.

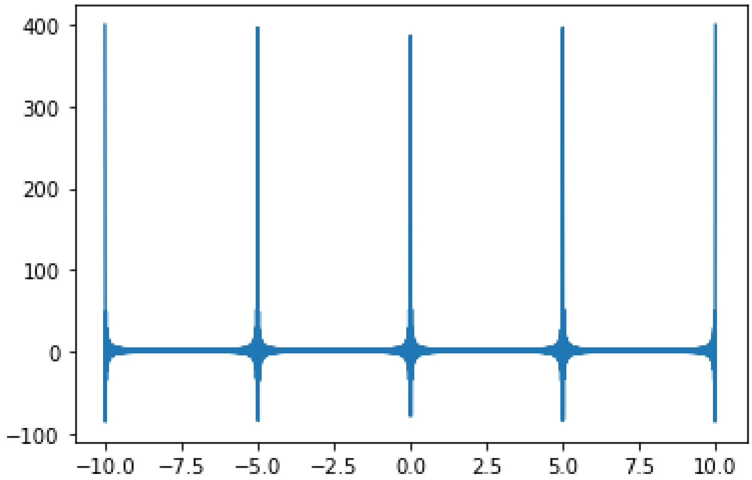
1.6

The last term in the serie has a period of $1/N$. To represent it and its periodicity well we need at least 5 points per period. There are N revolutions of the last term between 0 and 1 so we will need at least $5N$ points.

1.7

In [2]:

```
N=200
Q=5

function = lambda n,t : 2*np.cos(2*np.pi*n*t/Q)

def serie_calc(n,t):
    serie=0
    for i in range(n):
        serie += function(i,t)
    return serie

x = np.linspace(-10,10, int(20*n*7/Q))

plt.plot(x,serie_calc(n,x), label="N = " + str(N))
plt.show()
```



The points were the serie diverges to $+\infty$ are now on $Q\mathbb{Z}$ we thus have the dirac comb on Q Δ_Q.

This is not surprising given the result of **1.5** : We know that our serie S is in fact the dirac comb on 1, Δ_1.
Δ_1 being 1-periodic $t \mapsto \Delta_1(t/Q)$ is then Q-periodic and due to its value is equal to the Dirac comb on Q, Δ_Q.

Exercice 2

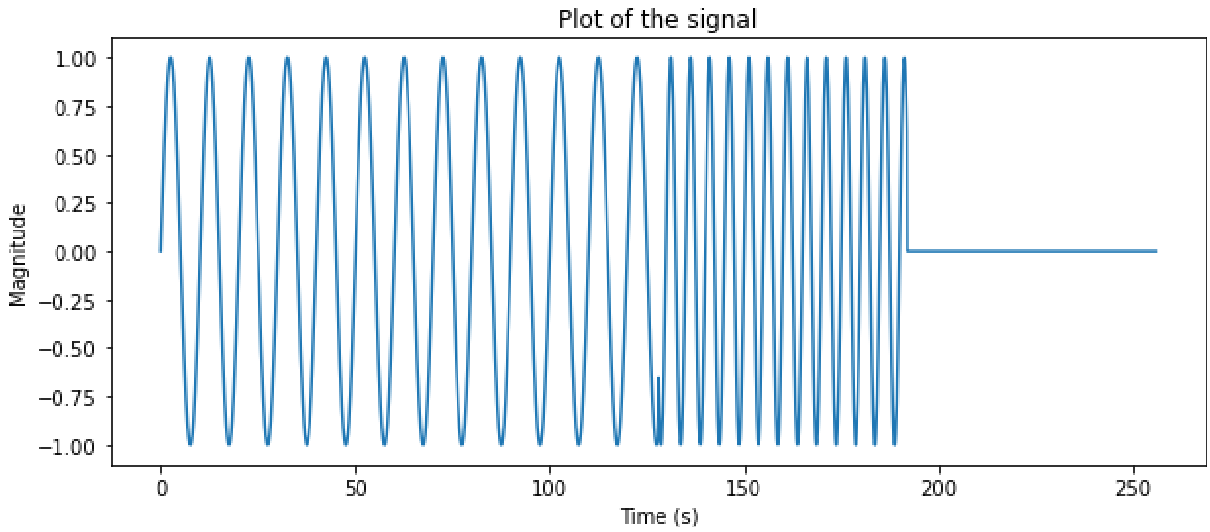
Short term Fourier Transform

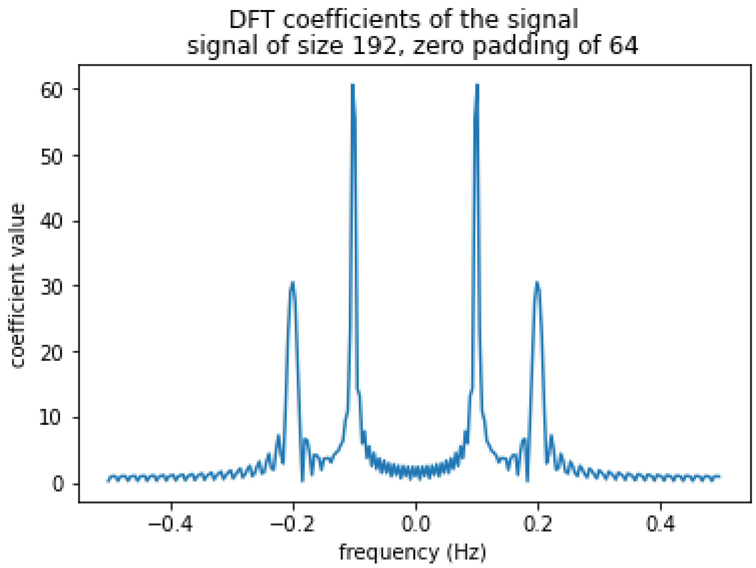
2.1

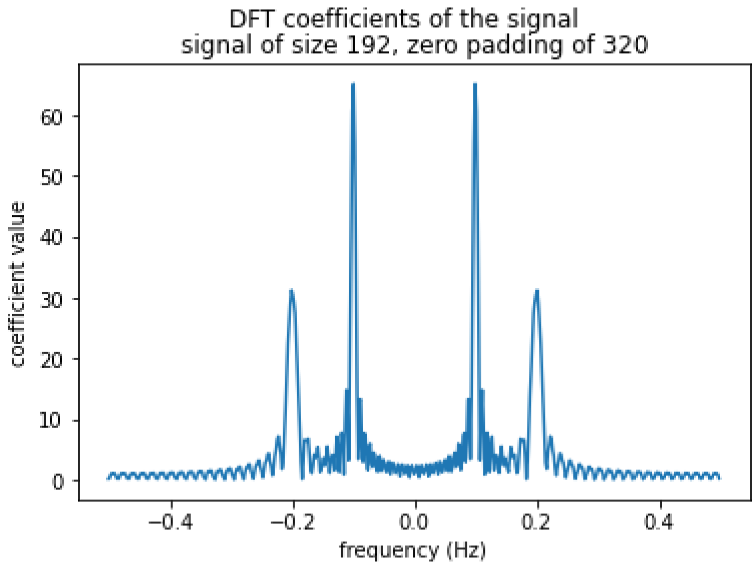
In [3]:

```
from numpy.fft import *
```

```
fig0, ax0 = plt.subplots()
fig1, ax1 = plt.subplots()
fig2, ax2 = plt.subplots()

sin1 = lambda t : np.sin(2*np.pi*0.1*t)
sin2 = lambda t : np.sin(2*np.pi*0.2*t)

def signal(t):
    if t>=192:
        return 0
    elif(0 <= t < 128):
        return sin1(t)
    elif(128<= t < 192):
        return sin2(t)

############################## Plotting of the signal #######################

ax0.set_title("Plot of the signal")
ax0.set_xlabel("Time (s)")
ax0.set_ylabel("Magnitude")
x = np.linspace(0, 256, 2000)
ax0.plot(x, np.array([signal(i) for i in x ]))

fig0.set_size_inches(10,4)

############################ DFT, signal of size 256 ########################

signal_val = np.zeros(256)
for i in range(256):
    signal_val[i] = signal(i)

signal_fft = fft(signal_val)
signal_fft = fftshift(signal_fft)

ax1.set_xlabel("frequency (Hz)")
ax1.set_ylabel("coefficient value")
fig1.suptitle("DFT coefficients of the signal")
ax1.set_title("signal of size 192, zero padding of 64")
ax1.plot(np.arange(-128,128)/256,abs(signal_fft))

fig1.show()

########################### DFT, signal of size 512 #########################

signal_val = np.zeros(512)
for i in range(512):
    signal_val[i] = signal(i)

signal_fft = fft(signal_val)
signal_fft = fftshift(signal_fft)

ax2.set_xlabel("frequency (Hz)")
ax2.set_ylabel("coefficient value")
fig2.suptitle("DFT coefficients of the signal")
ax2.set_title("signal of size 192, zero padding of 320")
ax2.plot(np.arange(-256,256)/512,abs(signal_fft))

fig2.show()
```







The symmetry is normal behaviour for the DFT of a real signal. It comes from the hermitian symmetry of the coefficients which implies the symmetry of the modulus.

2.2

We can observe that on $[-0.2, 0.2]$ more oscillations have appeared. This comes from the increased number of frequencies taken into account in the DFT.

The DFT of a sequence s of N values is the calculation of the N following values :

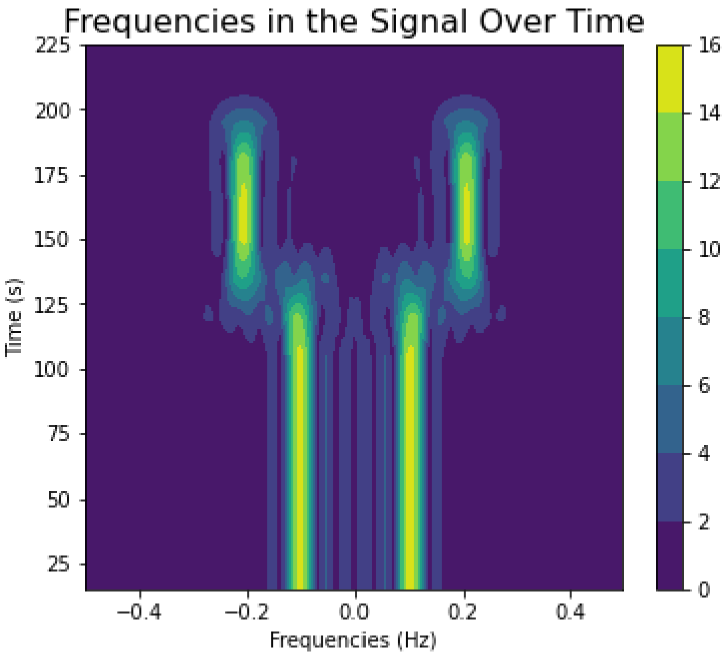
$$S(k) = \sum_{n=0}^{N-1} s(n)e^{-i\frac{2\pi k}{N}n} \qquad \text{with } k \text{ in } \{0, \ldots, N-1\}$$

Each $S(k)$ represents the calculation at frequency $\frac{k}{N}$. Making N bigger thus augments the frequency resolution of our transform.
For example doubling N will consider N more frequencies than before.
This can be achieved without augmenting the sampling of our signal by simply padding the rest of the sequence with zeros.

2.3

In [9]:

```
#size of the window, should not be bigger than 60
N=30

fig, ax = plt.subplots()

# We do the same thing as 2.1 here but for each windows of size N :
#
#    For each window, N values are taken and the rest is 0 padded forming
#    a sequence of size 256.
#
#    We then take the modulus of each value of the DFT with the appropriate
#    shift and store it in the k_th row.
#
#    The result is a real valued matrix of size Nx256 where each row
#    corresponds to the DFT of the signal on the respective window.

mat = np.zeros((256//(N//2)-2, 256))
for k in range(N//2):
    # creation of the current window, of size N
    window = np.linspace(k*(N//2), (k+2)*(N//2), N)
    for i in range(N):
        mat[k,i] = signal(window[i])
    mat[k,:] = abs(fftshift(fft(mat[k,:])))

#xx, yy = np.meshgrid(np.arange(-128,128)/256, np.arange(15,235,15), sparse=T
# Plotting of the matrix :
#
#       the x values are the frequencies. Given our sequence size of 256 the
#       256
#       The y values represent time. We associate to each window its center v
#       we begin with N//2 (center value of the [0,N] window) and end with
#       N//2*(256//(N//2)-1).
cf = ax.contourf(np.arange(-128,128)/256, np.arange(N//2, (N//2)*(256//(N//2)
ax.set_xlabel("Frequencies (Hz)")
ax.set_ylabel("Time (s)")
fig.set_size_inches(6,5)
ax.set_title("Frequencies in the Signal Over Time", fontdict={'fontsize': 16,
fig.colorbar(cf)
fig.show()
```



We can clearly see the shift in frequency around 128 from 0.1Hz to 0.2Hz in accordance with our signal.