Lab session 4

On signal windowing and filtering

report by Grebennikova Anna & Kseniia Ovchinnikova

Université Grenoble Alpes, UGA

Exercise 1. *Signal windowing*

We consider the Hamming window of length N:

$$w_h(n) = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{N}), & \textit{if } n \in \{0,...,N-1\} \\ 0, & \textit{otherwise}, \end{cases}$$

and rectangular window of length N:

$$w_r(n) = \begin{cases} 1, & \textit{if } n \in \{0,...,N-1\} \\ 0, & \textit{otherwise}. \end{cases}$$

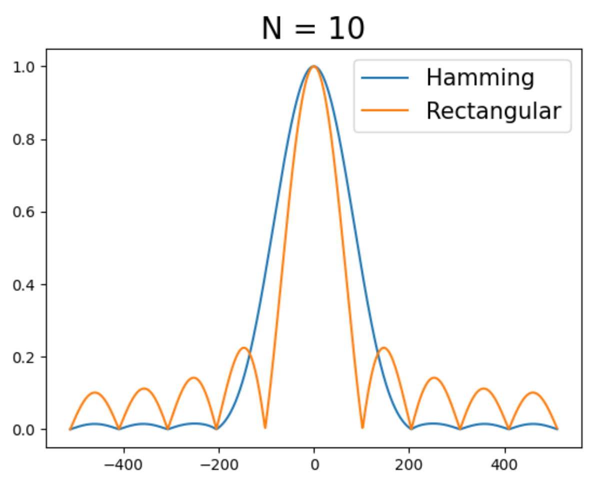
Question 1.1

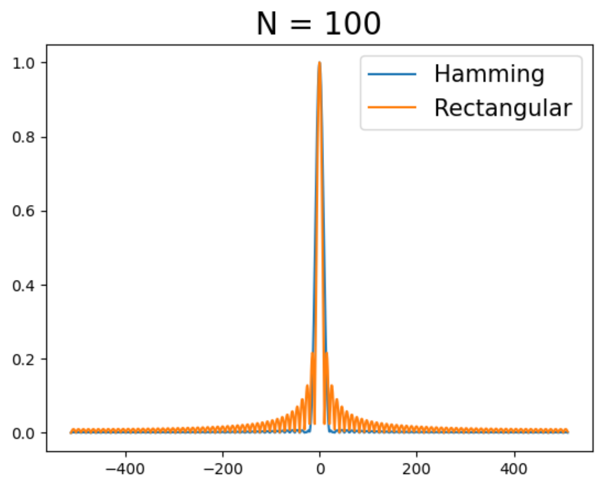
We write a program to calculate and plot the modulus of the Fourier transforms of windows:

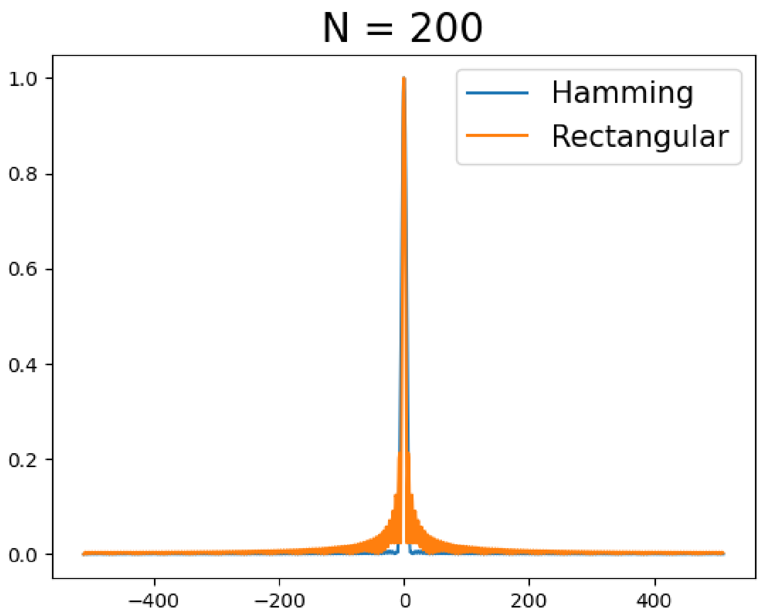
```
import numpy as np
import matplotlib.pyplot as plt

#we define w_h
def w_h(n, N):
    if 0<=n<N:
      return 0.54-0.46*np.cos(2*np.pi*n/N)
    return 0

#we define w_r
def w_r(n,N):
    if 0<=n<N:
      return 1
    return 0

#we define l1-normalisation of vector
def norm_l_1(x):
    s = sum(x)
    for i in range(len(x)):
        x[i] /= s
    return x

#We define N and L
N = [10, 100, 200]
L = 1024

#We create x-axis for the plots
ax = np.linspace(-L//2, L//2-1, L, endpoint = 'true')

plt.figure()

#for each N we have we do the following
for i in range(len(N)):
    #we initialise w_h and w_r
    W_h = []
    W_r = []

    #We compute w_h and w_r
    for n in range(N[i]):
        W_h.append(w_h(n,N[i]))
        W_r.append(w_r(n,N[i]))

    #We normalise w_h and w_r
    W_h = norm_l_1(W_h)
    W_r = norm_l_1(W_r)
```

```
    #We shift w_h and w_r
    W_h_T = np.fft.fftshift(np.fft.fft(W_h, L))
    W_r_T = np.fft.fftshift(np.fft.fft(W_r, L))

    #We plot w_h and w_r in regards to previously created axis
    plt.figure()
    plt.plot(ax, np.abs(W_h_T), label = 'Hamming')
    plt.plot(ax, np.abs(W_r_T), label = 'Rectangular')
    plt.title('N = '+str(N[i]), fontsize = 20)
    plt.legend(fontsize = 15, loc = 'upper right')

plt.show()
```

After the execution of the program we get:







Question 1.2

As we can see from graphs the bigger N gets the closer to desired results both Hamming and Rectangular window transform gets. This happens because a bigger window includes more data-points and this provides a more precise frequency representation.

We also can note that Hamming window in this case is more precise than Rectangular one.

Exercise 2. *Low-pass signal filtering*

We are building a finite impulse response filter h whose Fourier transform approximates $\mathbf{1}_{[-f_0,f_0]}$, the indicator function of the interval $[-f_0, f_0]$ for $f_0 < \frac{1}{2}$.

Question 2.1 and Question 2.2

Assume $H(\lambda)$ to be the transfer function of the filter h, i.e.,

$$H(\lambda) = \sum_{n\in\mathbb{Z}} h_n e^{-2i\pi n\lambda}.$$

Let $H(\lambda) = \mathbf{1}_{[-f_0,f_0]}(\lambda)$ for $\lambda \in \left[-\frac{1}{2}, \frac{1}{2}\right[$, assumed to be periodic with period 1, and $f_0 < \frac{1}{2}$ (low-pass filter). One would like to keep only N coefficients h_n. We have two cases:

If N is odd, one keeps the indices $-\frac{(N-1)}{2} \leq n \leq \frac{(N-1)}{2}$ and can compute the Fourier coefficients of this function, denoted as h_n, in the following way:

$$h_n = \int_{-\frac{1}{2}}^{\frac{1}{2}} H(\lambda) e^{2\pi n i\lambda}\, d\lambda = \int_{-f_0}^{f_0} e^{2\pi n i\lambda}\, d\lambda = \frac{e^{2\pi i n f_0} - e^{-2\pi i n f_0}}{2\pi i n} = \frac{2i\sin(2\pi n f_0)}{2\pi i n} = \frac{\sin(2\pi n f_0)}{\pi n}.$$

We should also note that for $n = 0$:

$$h_0 = \lim_{n\to 0} \frac{\sin(2\pi n f_0)}{\pi n} = \frac{2\pi n f_0}{\pi n} = 2f_0.$$

If N is even, to obtain a realizable filter with linear phase, one modifies h_n to $\tilde{h}_n$, which satisfies:

$$\tilde{h}_n = \int_{-\frac{1}{2}}^{\frac{1}{2}} H(\lambda) e^{i\pi(2n-1)\lambda}\, d\lambda = \int_{-f_0}^{f_0} e^{i\pi(2n-1)\lambda}\, d\lambda = \frac{e^{i\pi(2n-1)f_0} - e^{-i\pi(2n-1)f_0}}{i\pi(2n-1)}$$

$$= \frac{2i\sin(\pi(2n-1)f_0)}{2\pi i(2n-1)} = \frac{\sin(\pi(2n-1)f_0)}{\pi(2n-1)}.$$

Question 2.3

We write a function $FIR(f0, N)$ to compute g:

```python
import numpy as np

#We define Hamming window
def w_h(n, N):
    if 0<=n<N:
      return 0.54-0.46*np.cos(2*np.pi*n/N)
    return 0

#We define filter h(n)
def h_n(f0, N):
    h = []

    #We compute h by the following formula if N is even
    if N\%2 == 0:
        for n in range(-N//2+1, N//2+1):
            h.append(np.sin(np.pi*(2*n-1)*f0)/(np.pi*(2*n-1)))

    #We compute h by the folloring formula if N is odd
    else:
        for n in range((-N+1)//2, (N+1)//2):
          if n == 0:
            h.append(2 * f0)
          else:
```

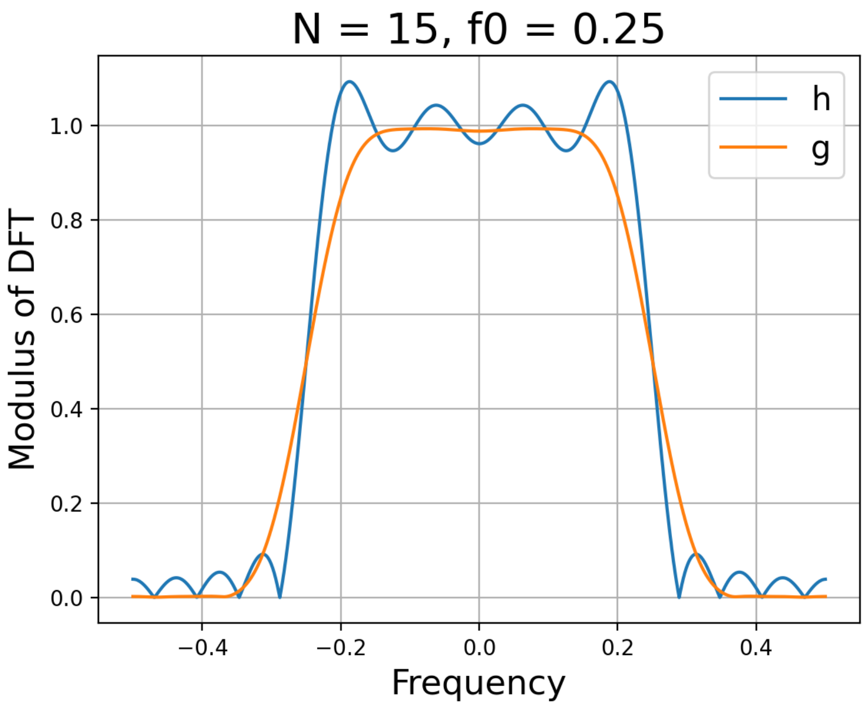
```python
            h.append(np.sin(2 * np.pi * n * f0) / (np.pi * n))
    return h

#We define FIR filter g
def FIR(f0, N):
    W_h = []
    #We compute Hamming window
    for n in range(N):
      W_h.append(w_h(n,N))

    #We compute h
    H_n = h_n(f0, N)

    #We apply Hamming window
    g = np.multiply(H_n, W_h)
    return g
```

Question 2.4

We write a program to compute the modulus of the DFT of h and g and plot it:

```python
import matplotlib.pyplot as plt
#We define N and f0
N = 15
f0 = 0.25

#We define number of frequency bins
L = 1024

#We define axis for plots
ax = np.linspace(-1/2, 1/2, L, endpoint = 'true')

#We calculate h
H_n = h_n(f0, N)
H_n_T = np.fft.fftshift(np.fft.fft(H_n, L))

#We calculate g
g = FIR(f0,N)
g_T = np.fft.fftshift(np.fft.fft(g, L))

#We plot h and g
plt.figure()
plt.plot(ax, np.abs(H_n_T), label = 'h')
plt.plot(ax, np.abs(g_T), label = 'g')
plt.title('N = '+str(N)+', f0 = '+str(f0), fontsize = 20)
plt.legend(fontsize = 15, loc = 'upper right')
plt.xlabel("Frequency")
plt.ylabel("Modulus of DFT")
plt.grid()
plt.show()
```

The above-stated program prints the following graph:



We observe that g which is acquired by application of Hamming window is better than h in cutting off undesired frequencies and providing a smoother response.

Exercise 3. *Band-pass signal filtering*

We are building a finite impulse response filter h whose Fourier transform approximates $\mathbf{1}_{[-f_0+f_1, f_0+f_1]}$ for $f_0 < \frac{1}{2}$ and $f_1 > f_0$ and $f_0 + f_1 < \frac{1}{2}$.

Question 3.1

Consider the expression for the Fourier transform of the sequence $2h_n \cos(2\pi f_1 n)$:

$$F(\xi) = \sum_{n\in\mathbb{Z}} 2h_n \cos(2\pi f_1 n) e^{-2i\pi n\xi} = \sum_{n\in\mathbb{Z}} h_n \left(e^{2i\pi n f_1} + e^{-2i\pi n f_1}\right) e^{-2i\pi n\xi} = \sum_{n\in\mathbb{Z}} h_n \left(e^{-2i\pi n(\xi - f_1)} + e^{-2i\pi n(\xi+f_1)}\right) =$$

$$= \sum_{n\in\mathbb{Z}} h_n e^{-2i\pi n(\xi-f_1)} + \sum_{n\in\mathbb{Z}} h_n e^{-2i\pi n(\xi+f_1)} = H(\xi - f_1) + H(\xi + f_1)$$

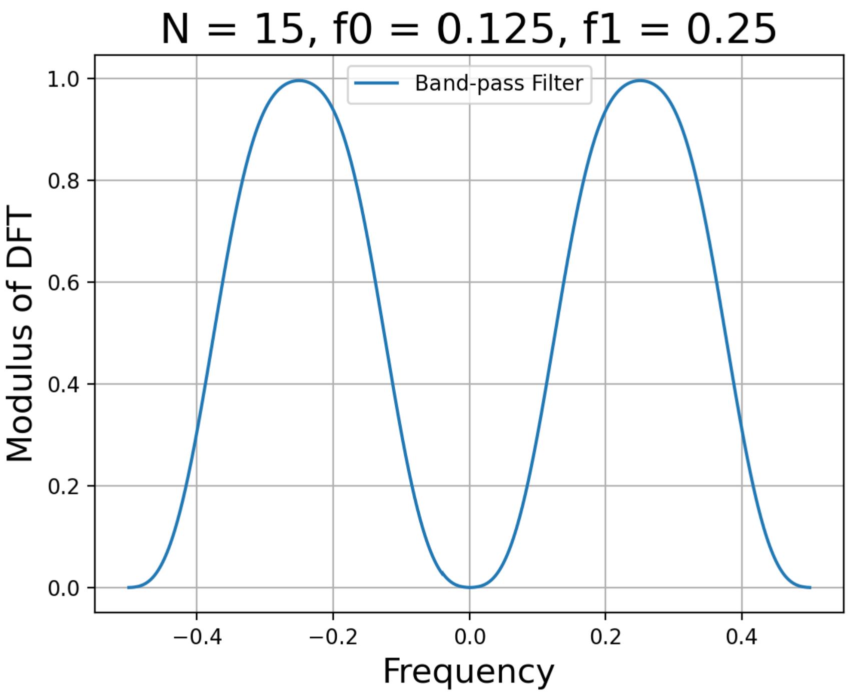
Question 3.2

We modify a program from **Exercise 2** to obtain filters of size N with Fourier transform approximating $\mathbf{1}_{[-f_0-f_1, f_0-f_1]} + \mathbf{1}_{[-f_0+f_1, f_0+f_1]}$:

```python
#We define f0 and f1
f0 = 0.125
f1 = 0.25

# We define filter coefficients for band-pass filter
def combined_filter(f0, f1, N):
    H_n = h_n(f0,N)
    filter_coefficients = []
    for n in range(-N//2+1, N//2+1):
      cos_component = 2 * H_n[n + N//2 - 1] * np.cos(2 * np.pi * f1 * n)

      # We apply Hamming window
      filter_coefficients.append(cos_component * w_h(n + N//2, N))
```

```
    return filter_coefficients

# We calculate the filter
filter_coeffs = combined_filter(f0, f1, N)

#We compute DFT of the filter
dft_filter = np.fft.fft(filter_coeffs, L)

# We plot the modulus of the DFT
plt.figure()
plt.plot(ax, np.abs(dft_filter), label = 'Band-pass Filter')
plt.title('N = '+str(N)+', f0 = '+str(f0)+', f1 = '+str(f1), fontsize = 20)
plt.xlabel("Frequency", fontsize = 16)
plt.ylabel("Modulus of DFT", fontsize = 16)
plt.legend()
plt.grid()
plt.show()
```

As a result of the program we get the following graph:



As we can see, band-pass filter passes frequencies within a fixed range (in our case $[-f_0 - f_1, f_0 - f_1]$ and $[-f_0 + f_1, f_0 + f_1]$).