REPORT

Zajęcia: Analog and digital electronic circuits
Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 2
Date: 8.03.2024
Topic: "Windowing"
Variant 5

Michał Herczyński
Informatyka II stopień,
stacjonarne,
1 semestr,
Gr.1b

1. Problem statement:

The exercise objective is to explore and analyze different types of signal windowing.

2. Input data:

The input data are parameters:

```
F1=300
F2=300.25
F3=299.75
FS=400
N=2000
TMPMU = 2
```

3. Commands used (or GUI):

a) source code

importng necessary libraries

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal.windows import hann, flattop
from numpy.fft import fft, ifft, fftshift
```

defining signal parameters

```
F1=300
F2=300.25
F3=299.75
FS=400
N=2000
TMPMU = 2
```

generating signals with fx period and n steps

```
k= np.arange(N)
x1=np.sin(TMPMU*np.pi * F1/ FS *k)
x2=np.sin(TMPMU*np.pi * F2/ FS *k)
x3=np.sin(TMPMU*np.pi * F3/ FS *k)
```

Generating windows.

Windows are define how signal samples are fetch.
Windowing function defines differences between observed signal spectrum and observation result.

The idea of windowing a signal is to force continuity at the boundaries, prior to performing the DFT or FFT. The simplest way to achieve this is by multiplying the signal x[n] by another signal w[n] of the same duration, such that w[0] = w[N-1] = 0, resulting in the windowed DFT or FFT X.

used windows:
rectangural window - where B = 1
Hann window with B = 1.5
Flat top window with B = 3.7702

```
wrect = np.ones(N)
whann = hann(N, sym=False)
wflattop = flattop(N, sym=False)
```

signals windowing

```
# 0-rect, 1-hann, 3-flattop
x1windows = (fft(x1), fft(x1 * whann), fft(x1 * wflattop))
x2windows = (fft(x2), fft(x2 * whann), fft(x2 * wflattop))
x3windows = (fft(x3), fft(x3 * whann), fft(x3 * wflattop))
```

The function written below, converts FFT in term of sine normalization. Implementation can handle odd and even value of N.

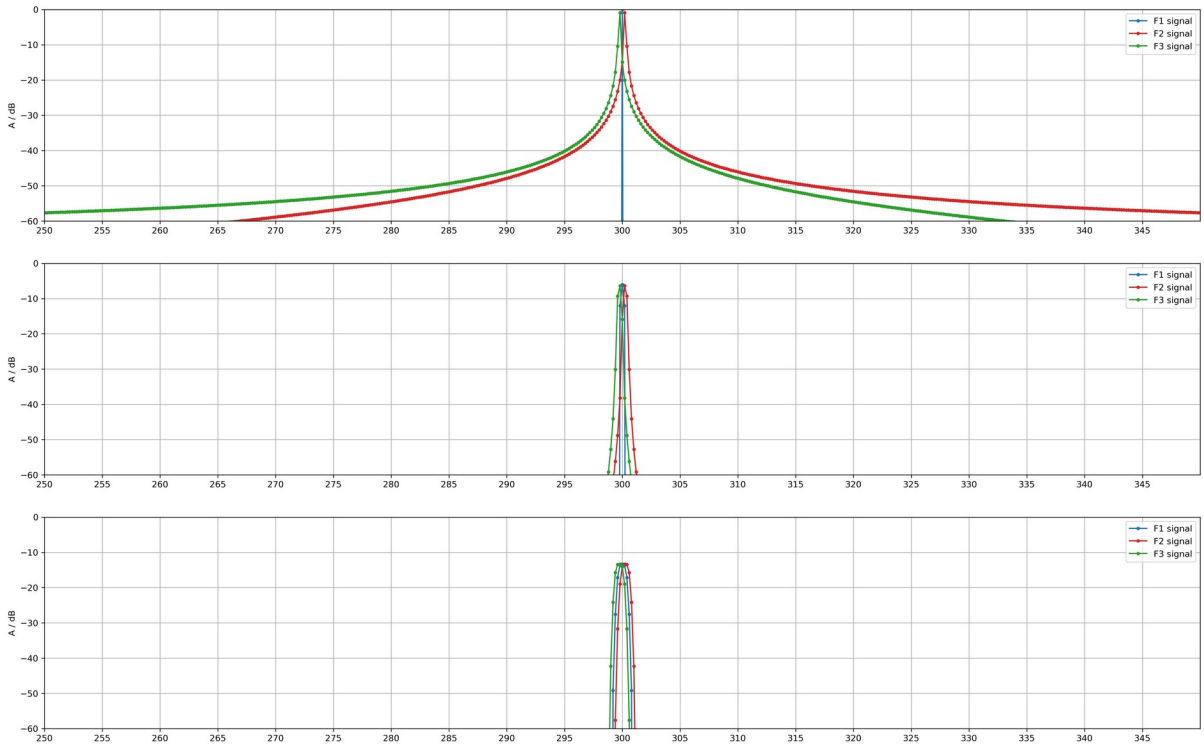
```
def fft2db(X:np.ndarray):
N = X.size
Xtmp = 2 / N * X # independent of N, norm for sine amplitudes
Xtmp[0] *= 1 / 2 # bin for f=0 Hz is existing only once, so cancel *2 from above
if N % 2 == 0: # fs/2 is included as a bin
# fs/2 bin is existing only once, so cancel *2 from above
Xtmp[N // 2] = Xtmp[N // 2] / 2
return 20 * np.log10(np.abs(Xtmp)) # in dB


# setup of frequency vector this way is independent of N even/odd:
df = FS / N
f = np.arange(N) * df
```

the window DTFT spectra normalized to their mainlobe maximum.

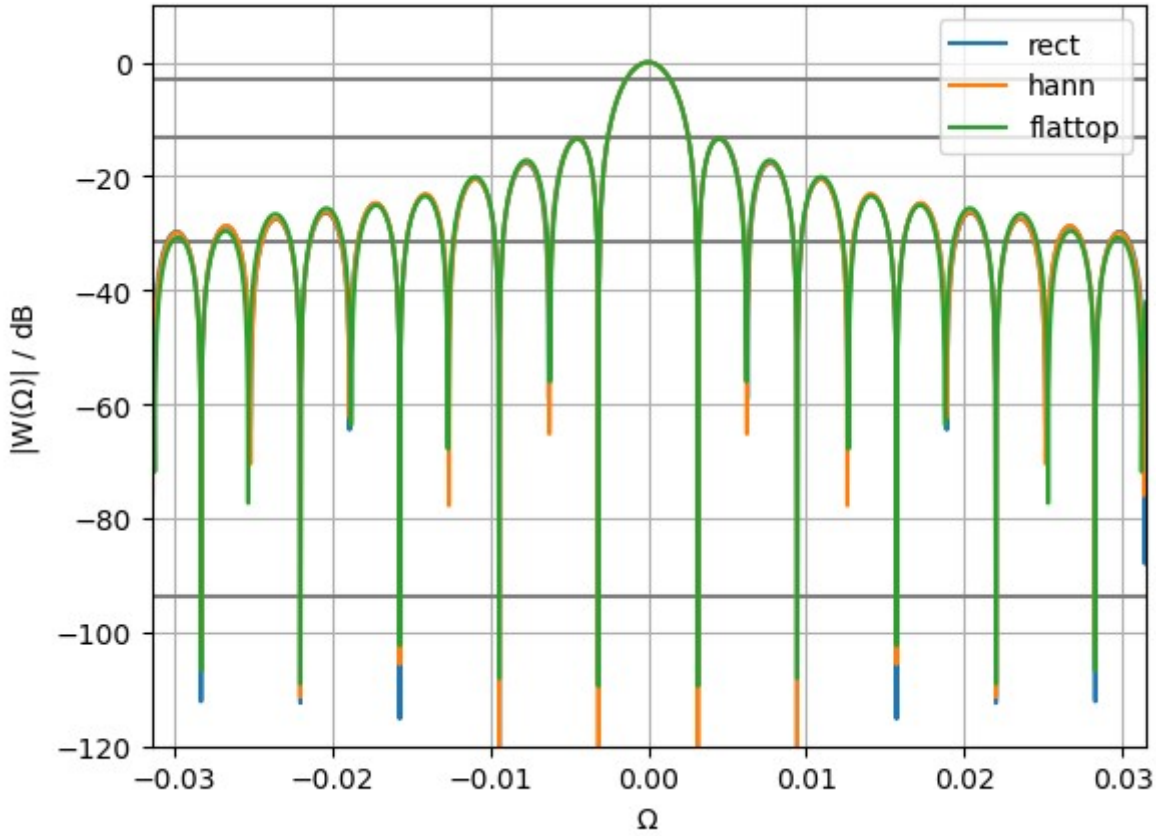
```
def winDTFTdB(w:np.ndarray):
N = w.size # get window length
Nz = 100 * N # zeropadding length
W = np.zeros(Nz) # allocate RAM
W[0:N] = w # insert window
W = np.abs(fftshift(fft(W))) # fft, fftshift and magnitude
W /= np.max(W) # normalize to maximum, i.e. the mainlobe maximum here
W = 20 * np.log10(W) # get level in dB
# get appropriate digital frequencies
Omega = 2 * np.pi / Nz * np.arange(Nz) - np.pi # also shifted
return Omega, W
```

b) screenshots

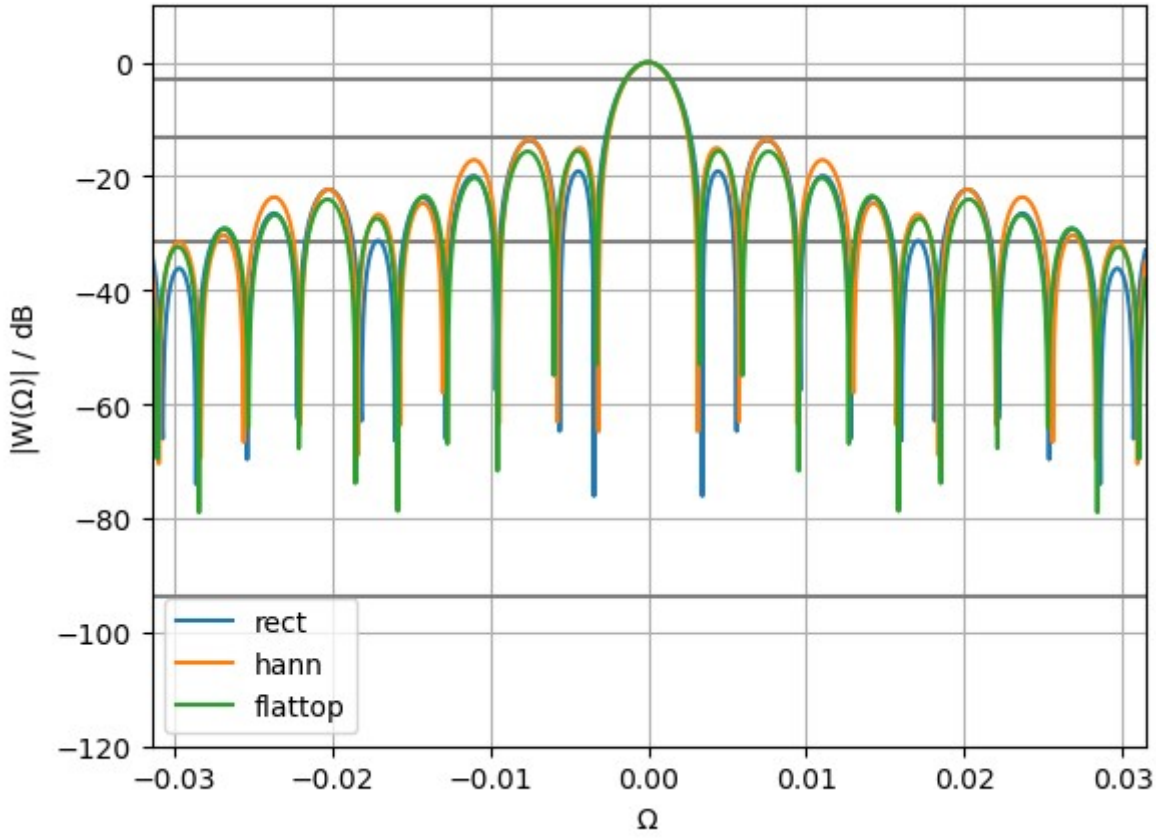
DFT spectra. The frequency of each signal can be easilly read.



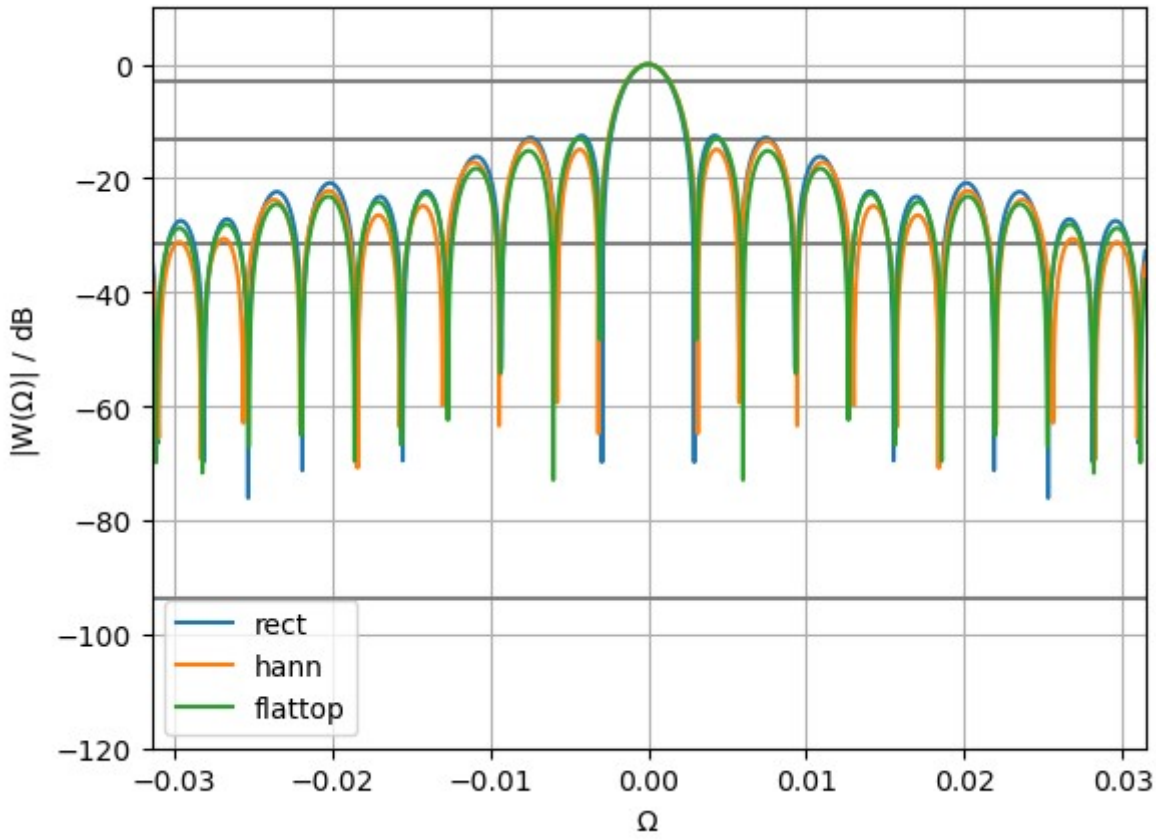
Signal X1, freq=300Hz
rect window mainlobe bandwidth is 0.1760000000000275 Hz,
0.00276460153515945 rad
hann window mainlobe bandwidth is 0.1760000000000275 Hz,
0.00276460153515945 rad
flattop window mainlobe bandwidth is 0.1760000000000275 Hz,
0.00276460153515945 rad



Signal X2, freq=300.25Hz
rect window mainlobe bandwidth is 0.18000000000001143 Hz,
0.002827433388230993 rad
hann window mainlobe bandwidth is 0.1720000000000436 Hz,
0.0027017696820879067 rad
flattop window mainlobe bandwidth is 0.18000000000001143 Hz,
0.002827433388230993 rad



Signal X3, freq=299.75Hz
rect window mainlobe bandwidth is 0.1640000000000192 Hz,
0.002576105975943932 rad
hann window mainlobe bandwidth is 0.1720000000000436 Hz,
0.0027017696820879067 rad
flattop window mainlobe bandwidth is 0.16800000000005966 Hz,
0.0026389378290163634 rad



3. Conclusions: For the reasons given, we conclude that the idea of windowing a signal is to force continuity at the boundaries, prior to performing the DFT or FFT. Windowing and normalizing Fourier transforms allow us to analyze the frequency spectrum of a signal. Normalizing the DTFT spectra to their mainlobe max can provide a consistent reference point for comparing and analyzing different signals or systems. This normalization helps in focusing on the relative distribution of energy within the mainlobe of the spectrum, making it easier to identify and compare key features.

4. Repository:
https://github.com/MichalHer/aadec_mgr/tree/main/ex2_windowing