REPORT

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

Lab 01
Date 26.10.2024
Topic: "Windowing"
Variant 11

Szymon Nycz,
Informatyka II stopień,
niestacjonarne,
1 semestr,
Gr.1b

1. Problem statement:

Generate three sine signals of given f1, f2, and f3 and amplitude |x[k]|max for the sampling frequency fs in the range of $0 \leq k < N$.

Plot: 1 1. the "normalized" level of the DFT spectra. 2. the window DTFT spectra normalized to their mainlobe maximum. The intervals for f, Ω, and amplitudes should be chosen by yourself for the best interpretation purposes.

2. Input data:

No	f_1	f_2	f_3	$\|x[k]\|_{\max}$	f_s	N
1	300	300.25	299.75	2	400	2000
2	400	400.25	399.75	2	600	3000
3	500	500.25	499.75	2	800	1800
4	600	600.25	599.75	2	500	2000
5	300	300.25	299.75	2	400	2000
6	600	600.25	599.75	3	800	2000
7	400	400.25	399.75	3	600	3000
8	500	500.25	499.75	3	800	1800
9	600	600.25	599.75	3	500	2000
10	300	300.25	299.75	3	400	2000
11	200	200.25	199.75	4	400	2000
12	400	400.25	399.75	4	600	3000
13	500	500.25	499.75	4	800	1800
14	600	600.25	599.75	4	500	2000
15	500	500.25	499.75	4	800	2000

3. Commands used (or GUI):

a) source code

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import fft, ifft, fftshift
from scipy.signal.windows import hann, flattop    ModuleNotFoundError: No module named 'scipy'
```

```
f1 = 200 # Hz
f2 = 200.25 # Hz
f3 = 199.75 #Hz
fs = 400 # Hz
N = 2000
k = np.arange(N)
x1 = 4 * np.sin(2*np.pi*f1/fs*k)
x2 = 4 * np.sin(2*np.pi*f2/fs*k )
x3 = 4 * np.sin(2*np.pi*f3/fs*k )
```

```
wrect = np.ones(N)
whann = hann(N,sym=False)
wflattop = flattop(N, sym=False)
plt.plot(wrect, 'C0o-', ms=3, label='rect')
plt.plot(whann, 'C1o-', ms=3, label='hann')
plt.plot(wflattop, 'C2o-', ms=3, label='flattop')
plt.xlabel(r"$k$")
plt.ylabel(r"window $w [ k ] $")
plt.xlim(0, N)
plt.legend()
plt.grid(True)
```

```
X1wrect = fft(x1)
X2wrect = fft(x2)
X3wrect = fft(x3)
X1whann = fft(x1*whann)
X2whann = fft(x2*whann)
X3whann = fft(x3*whann)
X1wflattop = fft(x1*wflattop)
X2wflattop = fft(x2*wflattop)
X3wflattop = fft(x3*wflattop)
```

```
def fft2db(X):
    N = X.size
    Xtmp = 2/N*X
    Xtmp[0] *= 1/2
    if N%2 == 0:
        Xtmp[N//2] = Xtmp[N//2]/2
    return 20*np.log10(np.abs(Xtmp))
```

```
df = fs/N
f = np.arange(N)*df
```

```
plt.figure(figsize = (16/1.5, 10/1.5))
plt.subplot(3, 1, 1)
plt.plot(f, fft2db(X1wrect), 'C0o-', ms=3, label='x1')
plt.plot(f, fft2db(X2wrect), 'C3o-', ms=3, label='x2')
plt.plot(f, fft2db(X3wrect), 'C2o-', ms=3, label='x3')
plt.xlim(175, 225)
plt.ylim(-340, -30)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-340, -30, 30))
plt.legend()
plt.ylabel('A / dB')
plt.grid(True)
plt.subplot(3, 1, 2)
plt.plot(f, fft2db(X1whann), 'C0o-', ms=3, label='x1')
plt.plot(f, fft2db(X2whann), 'C3o-', ms=3, label='x2')
plt.plot(f, fft2db(X3whann), 'C2o-', ms=3, label='x3')
plt.xlim(175, 225)
plt.ylim(-320, -50)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-320, -50, 30))
plt.legend()
plt.ylabel('A / dB')
plt.grid(True)
plt.subplot(3, 1, 3)
plt.plot(f, fft2db(X1wflattop), 'C0o-', ms=3, label='x1')
plt.plot(f, fft2db(X2wflattop), 'C3o-', ms=3, label='x2')
plt.plot(f, fft2db(X3wflattop), 'C2o-', ms=3, label='x3')
plt.xlim(175, 225)
plt.ylim(-320, -50)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-320, -50, 30))
plt.legend()
plt.xlabel('f / Hz')
plt.ylabel('A / dB')
plt.grid(True)
```

```
def winDTFTdB(w):
    N = w.size
    Nz = 100*N
    W = np.zeros(Nz)
    W[0:N] = w
    W = np.abs(fftshift(fft(W)))
    W /= np.max(W)
    np.seterr(divide = 'ignore')
    W = 20*np.log10(W)
    Omega = 2*np.pi/Nz*np.arange(Nz)-np.pi
    return Omega, W
```

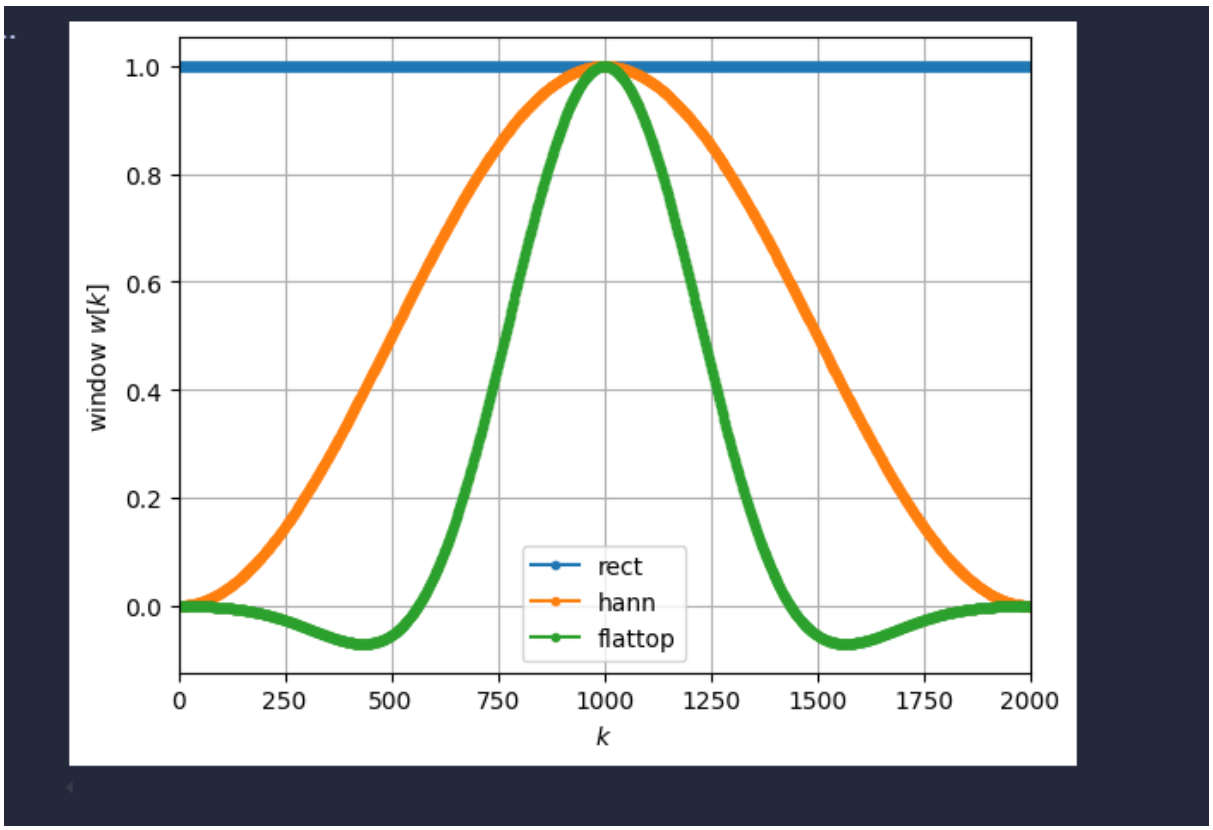
```
plt.plot([-np.pi, +np.pi], [-3.01, -3.01], 'gray')
plt.plot([-np.pi, +np.pi], [-13.3, -13.3], 'gray')
plt.plot([-np.pi, +np.pi], [-31.5, -31.5], 'gray')
plt.plot([-np.pi, +np.pi], [-93.6, -93.6], 'gray')
Omega, W = winDTFTdB(wrect)
plt.plot(Omega, W, label='rect')
Omega, W = winDTFTdB(whann)
plt.plot(Omega, W, label='hann')
Omega, W = winDTFTdB(wflattop)
plt.plot(Omega, W, label='flattop')
plt.xlim(-np.pi, np.pi)
plt.ylim(-120, 10)
plt.xlim(-np.pi/100, np.pi/100)
plt.xlabel(r'$\Omega$')
plt.ylabel(r'|W($\Omega$)| / dB')
plt.legend()
plt.grid(True)
```

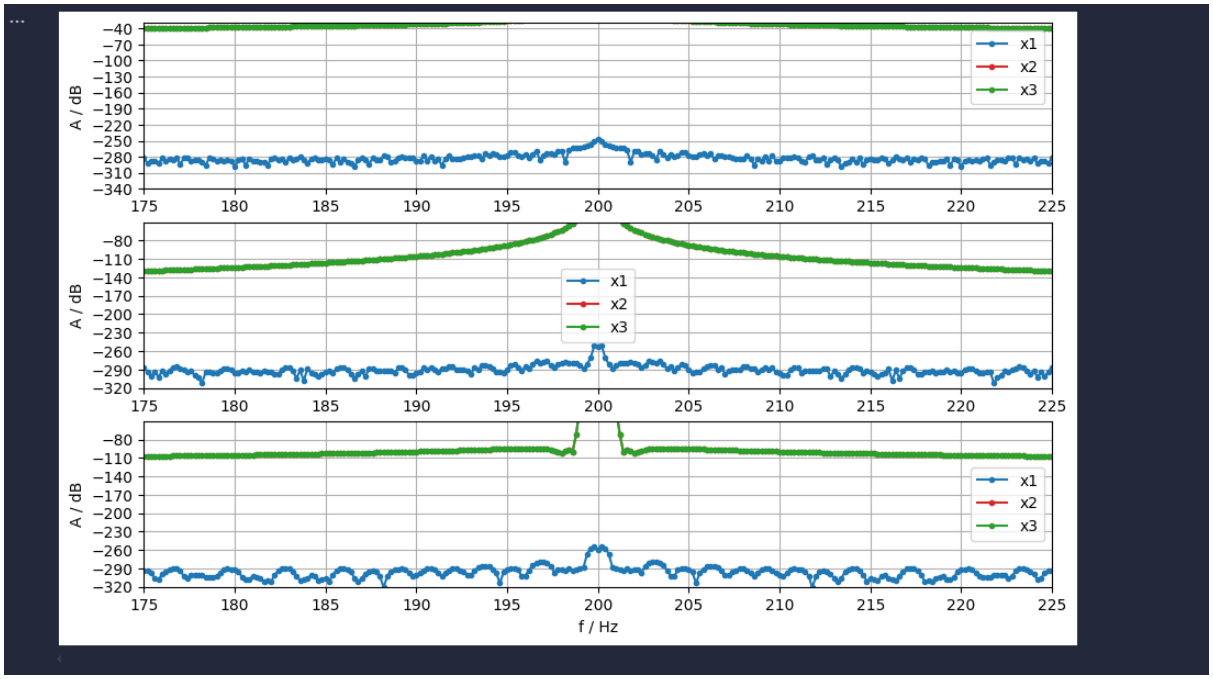
```python
def winDTFTdB(w):
    N = w.size
    Nz = 100*N
    W = np.zeros(Nz)
    W[0:N] = w
    W = np.abs(fftshift(fft(W)))
    W /= np.max(W)
    np.seterr(divide = 'ignore')
    W = 20*np.log10(W)
    Omega = 2*np.pi/Nz*np.arange(Nz)-np.pi
    return Omega, W
```

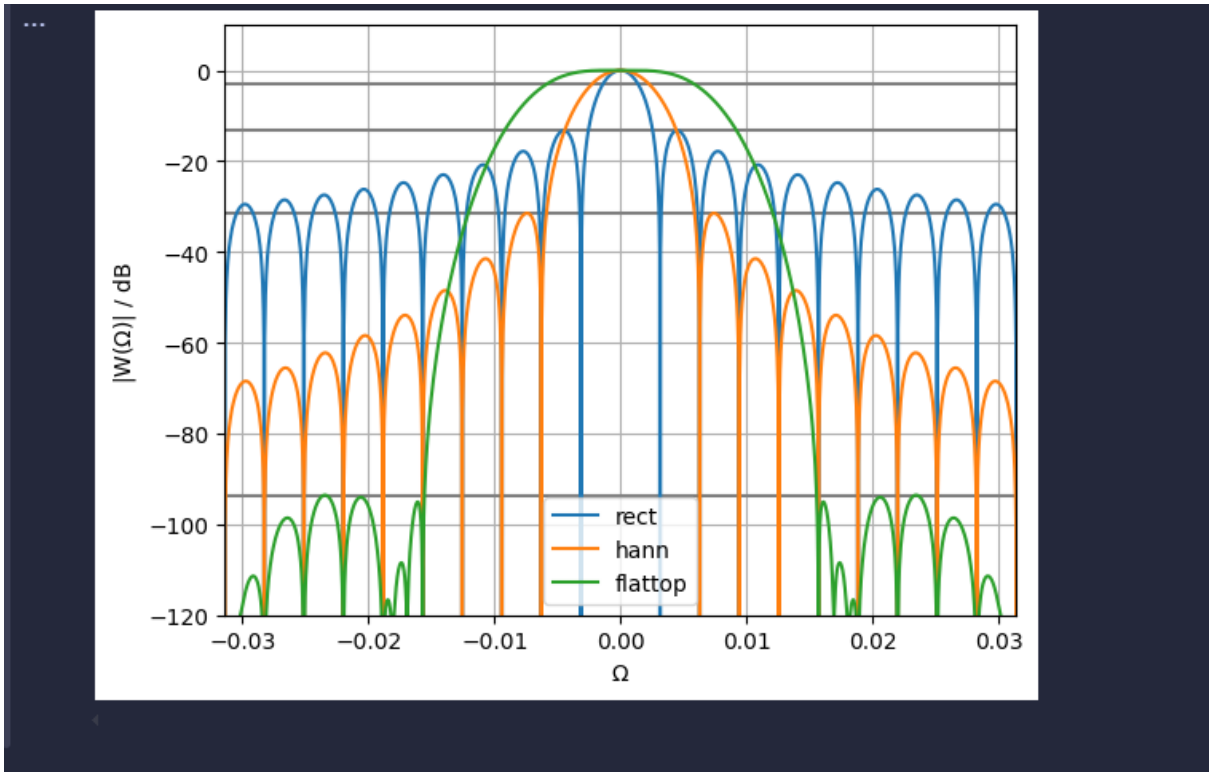
```python
plt.plot([-np.pi, +np.pi], [-3.01, -3.01], 'gray')
plt.plot([-np.pi, +np.pi], [-13.3, -13.3], 'gray')
plt.plot([-np.pi, +np.pi], [-31.5, -31.5], 'gray')
plt.plot([-np.pi, +np.pi], [-93.6, -93.6], 'gray')
Omega, W = winDTFTdB(wrect)
plt.plot(Omega, W, label='rect')
Omega, W = winDTFTdB(whann)
plt.plot(Omega, W, label='hann')
Omega, W = winDTFTdB(wflattop)
plt.plot(Omega, W, label='flattop')
plt.xlim(-np.pi, np.pi)
plt.ylim(-120, 10)
plt.xlim(-np.pi/100, np.pi/100)
plt.xlabel(r'$\Omega$')
plt.ylabel(r'|W($\Omega$)| / dB')
plt.legend()
plt.grid(True)
```

Link to remote repozytorium: https://github.com/Maciek332/Semestr_1_Nycz/tree/master/DSPja/Lab_2

4. Outcomes:







5. **Conclusions:**

With a low sampling frequency (fs), the f2 function aligns almost precisely with f1, resulting in significant overlap.

Decreased sampling frequency often leads to either overlapping functions or highly fluctuating levels, necessitating extensive plot adjustments for proper visualization.