REPORT

Classes: Analog and Digital Electronic Circuits

Presenter: prof. dr hab. inż. Vasyl Martsenyuk

Laboratorium No. 2 Date: 28.10.2023 Topic: „Windowing" Version 6	Adam Kubliński Informatyka II stopień, niestacjonarne, zaoczne, I semestr, gr. 1A

GitHub Repository:
https://github.com/Adamadacho/Analog_and_Digital_Electronic_Circuits.git

1. Topic of the laboratory

The objective is to be able the results of different type of windowing the signals.

2. Task

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 ≤ k < N accordingto variant 6. Complete inputs are highlighted in the table below.

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

Table 1: Variants

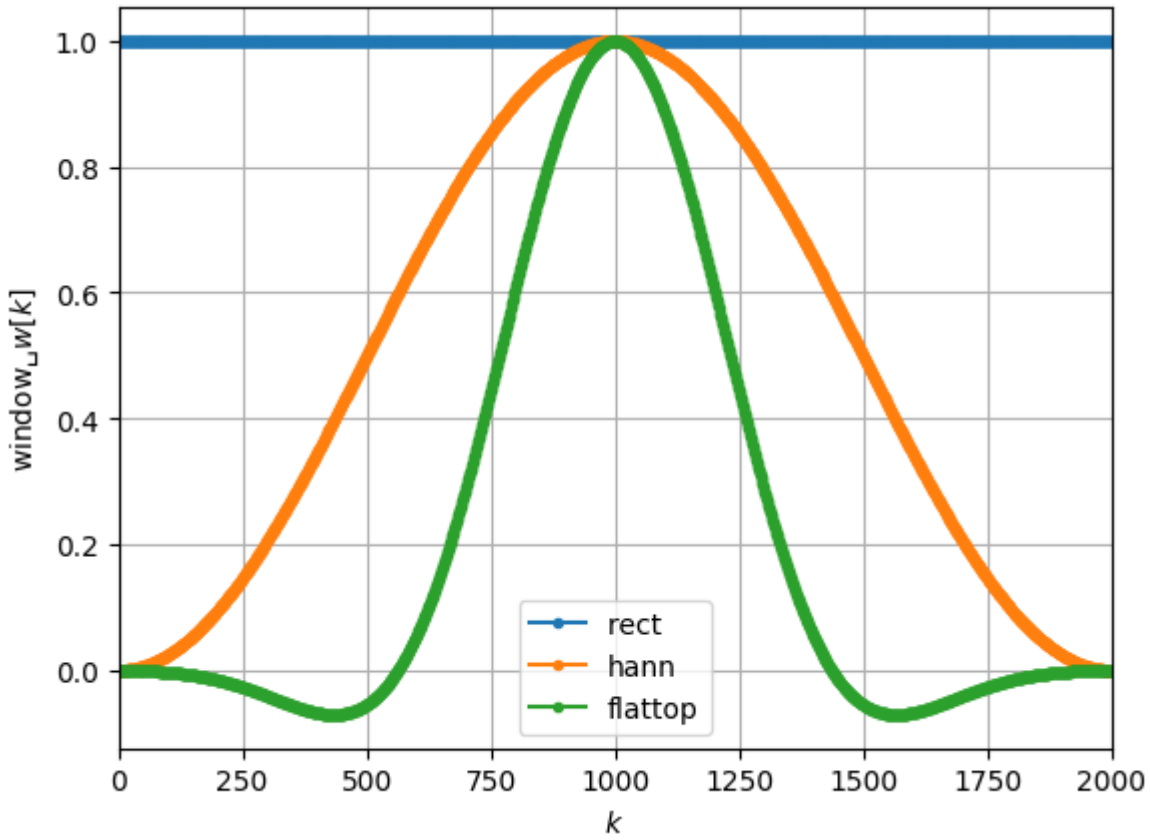
The code has been done according to instruction and has been adjusted to the variant 6. It was made in Jupyther Notebook.

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

# Generating signals
#Variant 6
f1 = 600  # Hz
f2 = 600.25  # Hz
f3 = 599.75  # Hz
#|x[k]|max = 3
fs = 800 #Hz
N = 2000


amplitude = 3
k = np.arange(N)
x1 = amplitude * np.sin(2*np.pi*f1/fs*k)
x2 = amplitude * np.sin(2*np.pi*f2/fs*k)
x3 = amplitude * np.sin(2*np.pi*f3/fs*k)

#Generating Windows
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)

#DFT spectra uding FFT algorithm
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)

# this handling is working for N even and odd:
def fft2db(X):
    N=X.size
    Xtmp=2/N * X # independent of N, norm for sine amplitudes
    Xtmp[0] *= 1/2 #bin for f=0Hz 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

#Solution
plt.figure(figsize=(16/1.5, 10/1.5))
plt.subplot(3, 1, 1)
plt.plot(f , fft2db(X1wrect), "C0o-" , ms=3, label="best case rect ")
plt.plot(f , fft2db(X2wrect), "C3o-" , ms=3, label="worst case upper rect ")
plt.plot(f , fft2db(X3wrect), "C4o-" , ms=3, label="worst case lower rect ")
plt.xlim(175, 225)
plt.ylim(-60, 20)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-60, 20, 10))
plt.legend()
#plt.xlabel("f / Hz")
plt.ylabel("A / dB")
plt.grid(True)
```

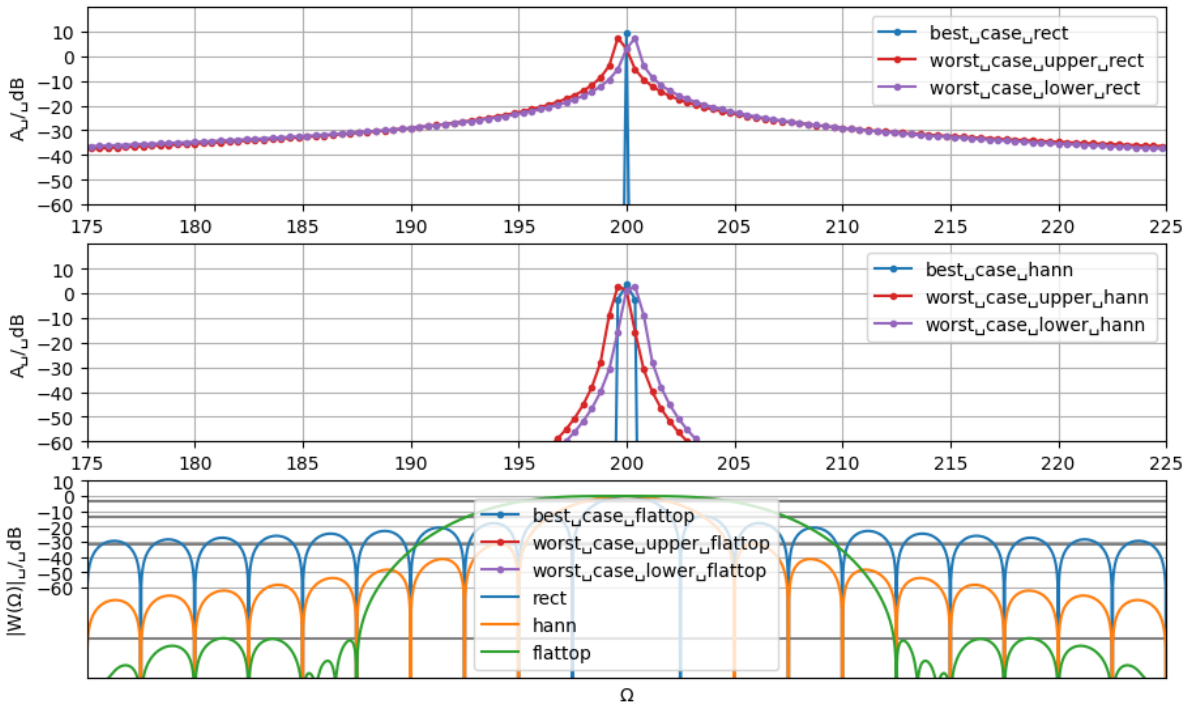
```
plt.subplot(3, 1, 2)
plt.plot(f , fft2db(X1whann), "C0o-" , ms=3, label="best case hann")
plt.plot(f , fft2db(X2whann), "C3o-" , ms=3, label="worst case upper hann")
plt.plot(f , fft2db(X3whann), "C4o-" , ms=3, label="worst case lower hann")
plt.xlim(175, 225)
plt.ylim(-60, 20)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-60, 20, 10))
plt.legend()
#plt.xlabel("f / Hz")
plt.ylabel("A / dB")
plt.grid(True)

plt.subplot(3, 1, 3)
plt.plot(f , fft2db(X1wflattop), "C0o-" , ms=3, label="best case flattop")
plt.plot(f , fft2db(X2wflattop), "C3o-" , ms=3, label="worst case upper flattop")
plt.plot(f , fft2db(X3wflattop), "C4o-" , ms=3, label="worst case lower flattop")
plt.xlim(175, 225)
plt.ylim(-60, 20)
plt.xticks(np.arange(175, 230, 5))
plt.yticks(np.arange(-60, 20, 10))
plt.legend()
plt.xlabel("f / Hz")
plt.ylabel("A / dB")
plt.grid(True)

#Preparations for solutions
def winDTFTdB(w):
    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

plt.plot([-np.pi , +np.pi ] , [-3.01, -3.01], "gray")#mainlobe bandwidth
plt.plot([-np.pi , +np.pi ] , [-13.3, -13.3], "gray")# rect max sidelobe
plt.plot([-np.pi , +np.pi ] , [-31.5, -31.5], "gray")#hannmax sidelobe
plt.plot([-np.pi , +np.pi ] , [-93.6, -93.6], "gray")# flattop max #sidelobe


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) #zoom into mainlobe
plt.xlabel(r"$\Omega$")
plt.ylabel(r"|W($\Omega$)| / dB")
plt.legend()
plt.grid(True)
```

Results of the laboratory are presented below

- The "normalized" level of the DFT spectra.



- The window DTFT spectra normalized to their mainlobe maximum. The intervals for f, Ω, and amplitudes



3. Conclusions

Based on the uploaded images showing graphs of time windows and their amplitude spectrum on a decibel scale, we can interpret the results in terms of the best case and worst case for different windows. The first chart shows three different time windows:

- Rectangular
- Hanning (hann)
- Flat (flattop)

The second and third graphs show the spectrum of signals multiplied by windows on the decibel scale. They show the frequency domain response for the best and worst case using each window. "Best case" usually means a signal at a frequency that exactly matches one of the FFT bins (called the frequency bin), while "worst case" refers to a frequency located between the bins, which causes the signal energy to be diluted between adjacent bins.

The second graph shows that the rectangular window has a very sharp peak, but also high sidelobes, which leads to a high level of spectral leakage. Hanning and planar windows, on the other hand, have lower side lobes, which means that spectral leakage is significantly reduced.

Reasons for differences in results for frequency signals f1 and f2 may be related to their location relative to the FFT bins. A signal with a frequency exactly matching the FFT bin (best case) will not blur and will have a clear, sharp peak in the spectrum. On the other hand, a signal with a frequency located between FFT bins (worst case) will spread the energy between adjacent bins, resulting in a less pronounced peak and a higher level of spectral leakage.

These spectra are used to analyze how windows affect resolution and spectral leakage in the Fourier transform, which is critical in applications such as signal analysis, image processing, and communications systems. The choice of window depends on the specific application and the desired compromise between resolution and spectral leakage.