REPORT

Zajęcia: Digital signal processing

Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 3&4

19.10.2024

Topic: "3. Discrete Fourier Transform (DFT): calculating the DFT of signals and analyzing the results.
4. Implementation of FFT algorithms: comparing the performance of different FFT implementations."

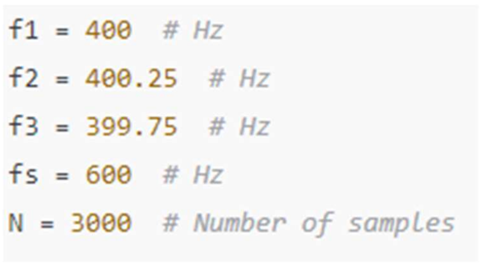
Variant 7

Dawid Klimek
Informatyka II stopień,
niestacjonarne,
1 semestr,
Gr.A

1. **Problem statement:**

Creation of three sine signals with specified f1, f2, and f3 frequencies and a maximum amplitude |x[k]|max, using a sampling frequency fs over the range 0 <= k < N.
Visualization:
 1. the "normalized" DFT spectrum levels.
2. the DTFT spectra for different windows, adjusted to their mainlobe peak. Choosen appropriate intervals for f, Omega, and amplitudes to ensure clear interpretation of the results.

2. Input data:



```
f1 = 400  # Hz
f2 = 400.25  # Hz
f3 = 399.75  # Hz
fs = 600  # Hz
N = 3000  # Number of samples
```

3. Commands used (or GUI):

a) source code
b) from scipy.signal.windows import hann, flattop
c) import numpy as np
d) import matplotlib.pyplot as plt
e) from numpy.fft import fft, ifft, fftshift
f)
g) # Define frequencies and sampling frequency based on the image data
h) f1 = 400 # Hz
i) f2 = 400.25 # Hz
j) f3 = 399.75 # Hz
k) fs = 600 # Hz
l) N = 3000 # Number of samples
m)
n) # Generate signals
o) k = np.arange(N)

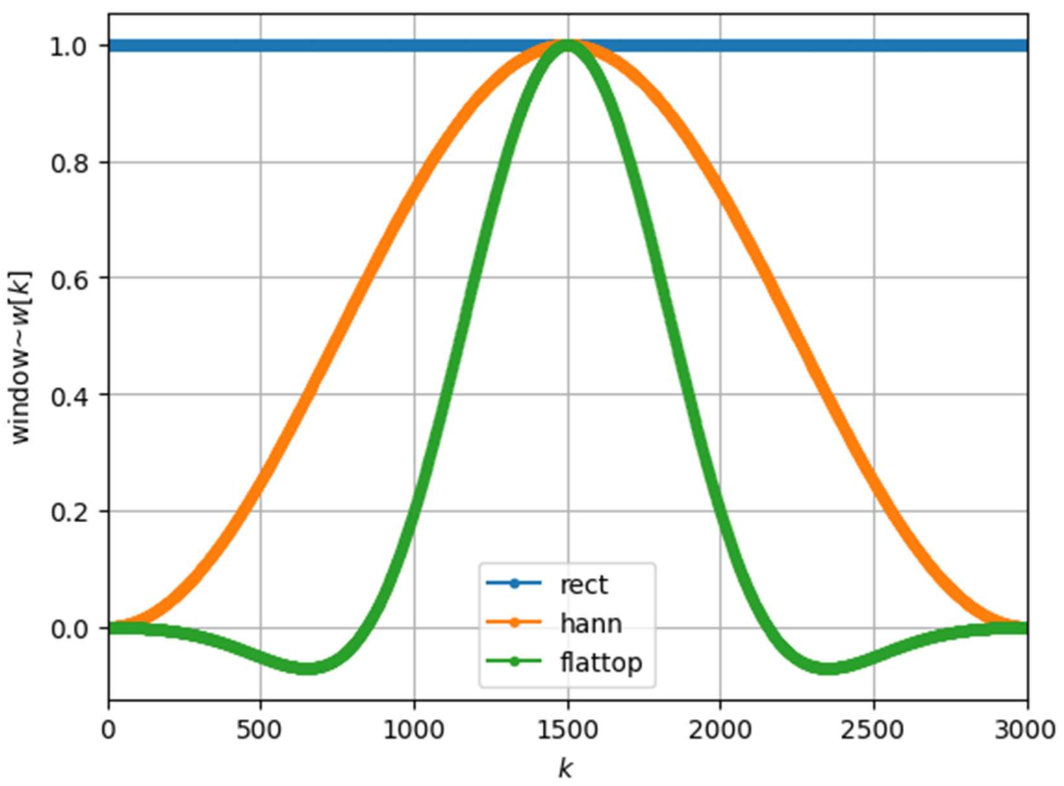
```
    p)  x1 = 3 * np.sin(2 * np.pi * f1 / fs * k)  # Amplitude adjusted to 3 as per
        image
    q)  x2 = 3 * np.sin(2 * np.pi * f2 / fs * k)
    r)  x3 = 3 * np.sin(2 * np.pi * f3 / fs * k)
    s)
    t)  # Define windows
    u)  wrect = np.ones(N)
    v)  whann = hann(N, sym=False)
    w)  wflattop = flattop(N, sym=False)
    x)
    y)  # Plot windows
    z)  plt.plot(wrect, 'C0o-', ms=3, label='rect')
    aa)plt.plot(whann, 'C1o-', ms=3, label='hann')
    bb)         plt.plot(wflattop, 'C2o-', ms=3, label='flattop')
    cc)plt.xlabel(r'$k$')
    dd)         plt.ylabel(r'window~$w[k]$')
    ee)plt.xlim(0, N)
    ff) plt.legend()
    gg)         plt.grid(True)
# Compute FFTs for signals with windows applied
X1wrect = fft(x1 * wrect)
X2wrect = fft(x2 * wrect)
X3wrect = fft(x3 * wrect)

X1whann = fft(x1 * whann)
X2whann = fft(x2 * whann)
X3whann = fft(x3 * whann)

X1wflattop = fft(x1 * wflattop)
X2wflattop = fft(x2 * wflattop)
X3wflattop = fft(x3 * wflattop)

# Function to compute normalized FFT spectrum in dB
def fft2db(X):
   N = X.size
   Xtmp = 2 / N * X
   Xtmp[0] *= 1 / 2
```

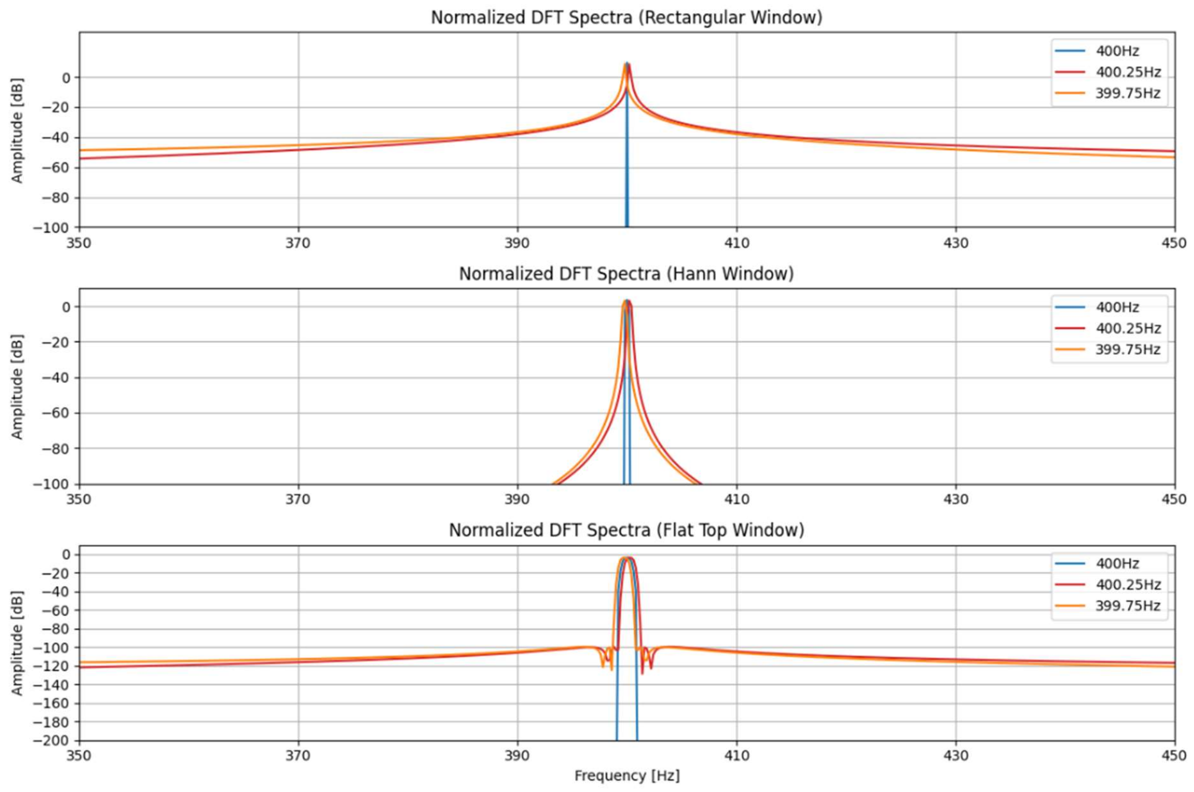
```
    if N % 2 == 0:
        Xtmp[N // 2] = Xtmp[N // 2] / 2
    return 20 * np.log10(np.abs(Xtmp))

# Frequency axis
df = fs / N
f = np.arange(N) * df

# Plot normalized DFT spectra with improved visuals
plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
plt.plot(f, fft2db(X1wrect), "C0-", lw=1.5, label="400Hz")
plt.plot(f, fft2db(X2wrect), "C3-", lw=1.5, label="400.25Hz")
plt.plot(f, fft2db(X3wrect), "C1-", lw=1.5, label="399.75Hz")
plt.xlim(350, 450)
plt.ylim(-100, 30)
plt.xticks(np.arange(350, 451, 20))
plt.yticks(np.arange(-100, 11, 20))
plt.title("Normalized DFT Spectra (Rectangular Window)")
plt.ylabel("Amplitude [dB]")
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(f, fft2db(X1whann), "C0-", lw=1.5, label="400Hz")
plt.plot(f, fft2db(X2whann), "C3-", lw=1.5, label="400.25Hz")
plt.plot(f, fft2db(X3whann), "C1-", lw=1.5, label="399.75Hz")
plt.xlim(350, 450)
plt.ylim(-100, 10)
plt.xticks(np.arange(350, 451, 20))
plt.yticks(np.arange(-100, 11, 20))
plt.title("Normalized DFT Spectra (Hann Window)")
plt.ylabel("Amplitude [dB]")
plt.legend()
plt.grid(True)
plt.subplot(3, 1, 3)
plt.plot(f, fft2db(X1wflattop), "C0-", lw=1.5, label="400Hz")
plt.plot(f, fft2db(X2wflattop), "C3-", lw=1.5, label="400.25Hz")
plt.plot(f, fft2db(X3wflattop), "C1-", lw=1.5, label="399.75Hz")
plt.xlim(350, 450)
plt.ylim(-100, 10)
plt.xticks(np.arange(350, 451, 20))
plt.yticks(np.arange(-200, 11, 20))
plt.title("Normalized DFT Spectra (Flat Top Window)")
plt.xlabel("Frequency [Hz]")
plt.ylabel("Amplitude [dB]")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

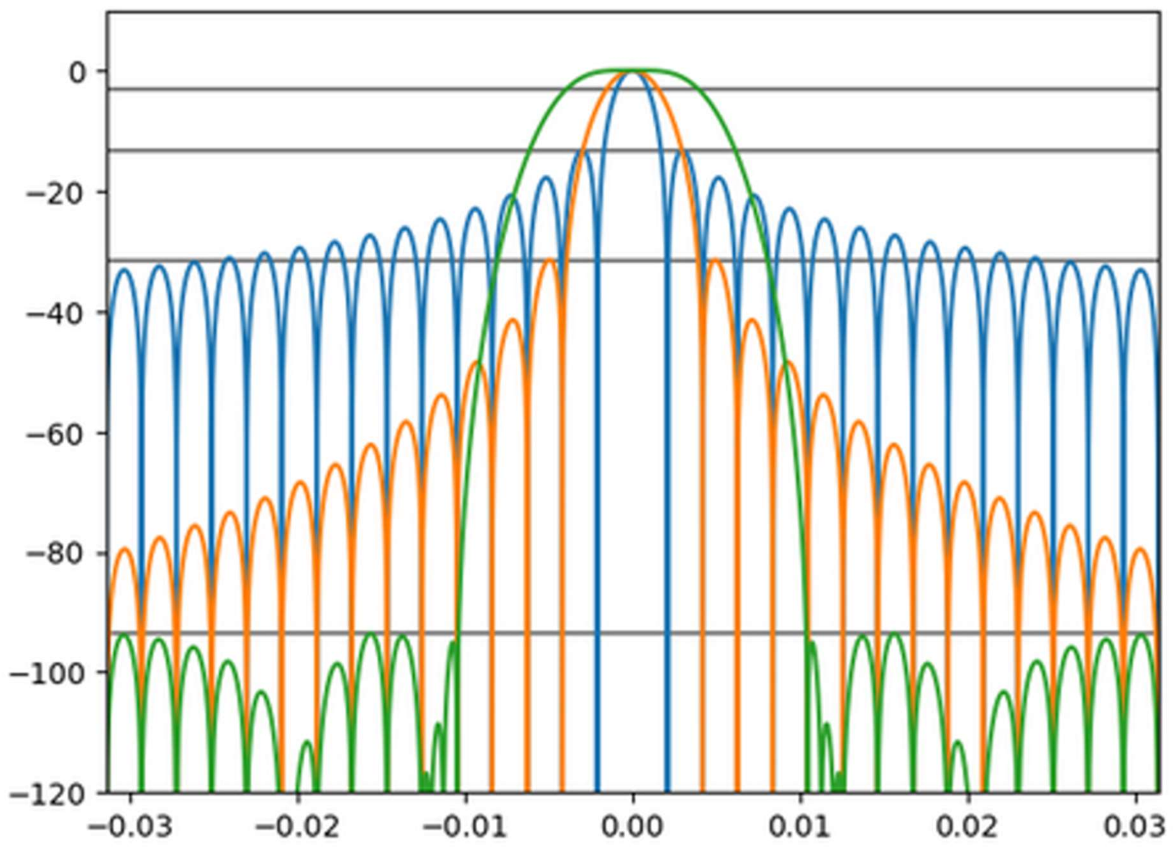
b) screenshots ; as outcomes

https://github.com/Skruty/DSP_UBB

4. Outcomes:


window~w[k]
k
rect
hann
flattop
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000


Normalized DFT Spectra (Rectangular Window)
Normalized DFT Spectra (Hann Window)
Normalized DFT Spectra (Flat Top Window)
Amplitude [dB]
Frequency [Hz]
400Hz
400.25Hz
399.75Hz
350
370
390
410
430
450



5. Conclusions:

This analysis highlights the importance of understanding how different parameters and methods in signal processing influence the results of spectral analysis. By analyzing sine wave signals with varying frequencies and amplitudes, the following observations were made:

Impact of Sampling Frequency and Signal Length:
The choice of sampling frequency fs and the number of samples N significantly affects the resolution and accuracy of the analysis. A higher sampling frequency provides better separation of frequencies in the spectrum, while a larger N improves the resolution of the discrete frequency components.

Normalization in Spectral Analysis:
Normalizing the spectra allows for easier comparison of signals, emphasizing relative differences in amplitude and frequency content. This is particularly useful in understanding the impact of various window functions on the signal's spectrum.

Frequency Components and Overlap:
Signals with closely spaced frequencies (e.g., 400 Hz and 400.25 Hz) demonstrate the importance of precise frequency representation. Misaligned frequencies can cause overlap in the spectrum, highlighting the need for careful design of the analysis framework.

Visualization as an Interpretative Tool:
Plots of normalized spectra and windowed DTFT results provide valuable insights into signal behavior. They help in identifying patterns, such as the spread of energy across the spectrum or the dominance of certain frequency components.

Practical Applications:
These methods and observations are essential in real-world applications such as audio signal processing, vibration analysis, and communications, where understanding frequency behavior and signal characteristics is critical for decision-making.

In summary, this analysis underscores the necessity of tailoring signal processing methods to specific applications, emphasizing flexibility and adaptability in spectral analysis. These findings provide a deeper understanding of how to approach complex signal processing tasks in various domains.