REPORT

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

Lab 3 and 4
Date: 5.11.2024
Topics:
Discrete Fourier Transform (DFT): calculating the DFT of signals and analyzing the results.
Implementation of FFT algorithms: comparing the performance of different FFT implementations.
Variant 8

Hubert Mentel
Informatyka II stopień,
niestacjonarne,
1 semestr,
Gr. A

1. Problem statement:
Creation of tree sine signals with specified f1, f2 and f3 frequencies. Maximum amplitude |x[k]|max, using a sampling frequency fs over the range 0 <= k < N.

Two visualizations:
-DFT (Discrete Fourier Transform)
-DTFT (Discrete-Time Fourier Transform)

2. Input data:
f1 = 500hz
f2 = 500.25hz
f3 = 499.75hz
x[k]max = 3
fs = 800hz
N = 1800

3. Commands used (or GUI):

source code:

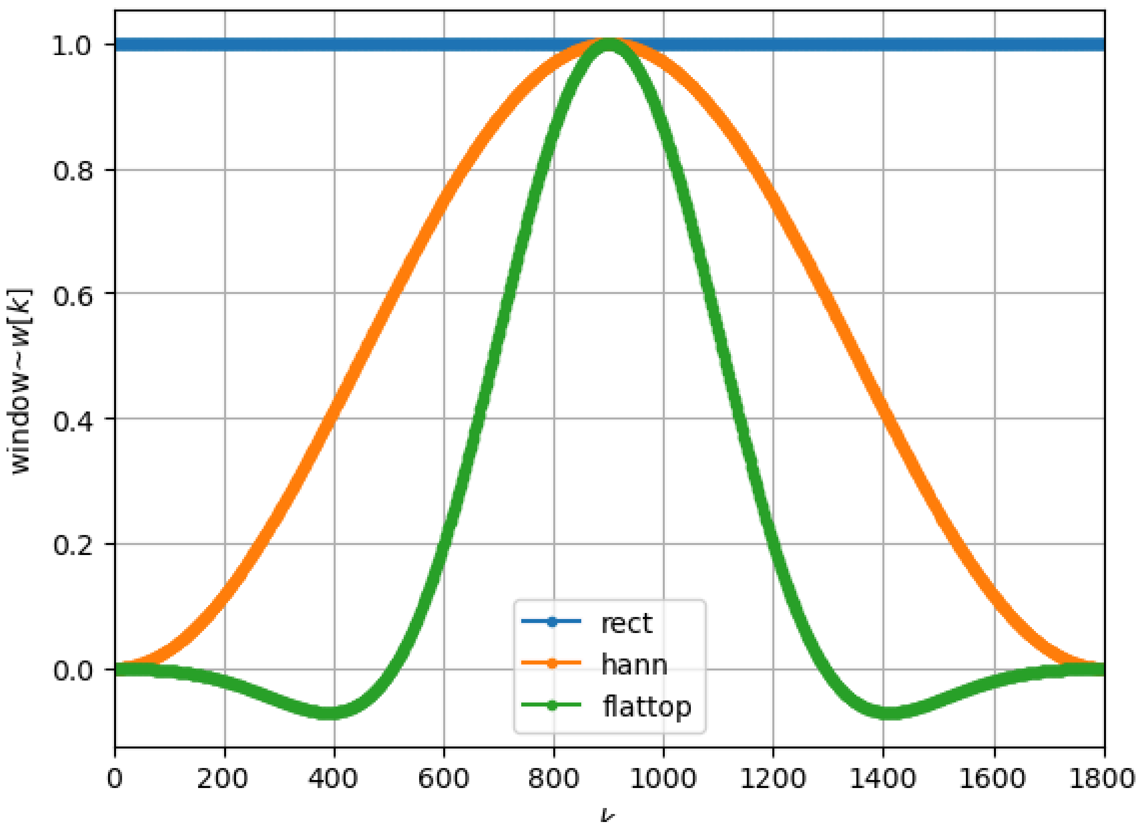
```
from scipy.signal.windows import hann, flattop
import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import fft, ifft, fftshift

# Define frequencies and sampling frequency based on the image data
f1 = 500  # Hz (updated for Variant 8)
f2 = 500.25  # Hz (updated for Variant 8)
f3 = 499.75  # Hz (updated for Variant 8)
fs = 800  # Hz (updated for Variant 8)
N = 1800  # Number of samples remains the same
```

```python
# Generate signals
k = np.arange(N)
x1 = 3 * np.sin(2 * np.pi * f1 / fs * k)  # Amplitude adjusted to 3 as per image
x2 = 3 * np.sin(2 * np.pi * f2 / fs * k)
x3 = 3 * np.sin(2 * np.pi * f3 / fs * k)

# Define windows
wrect = np.ones(N)
whann = hann(N, sym=False)
wflattop = flattop(N, sym=False)

# Plot windows
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)
plt.show()

# 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
```

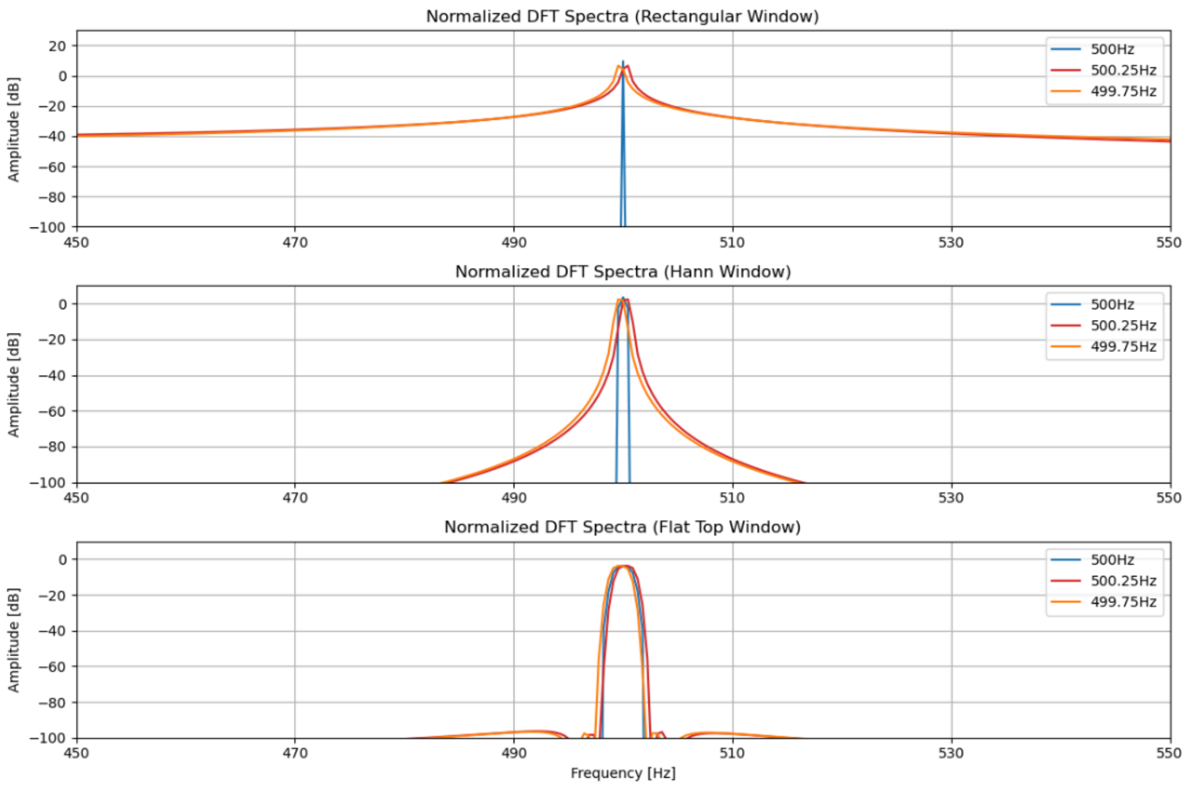
```
def fft2db(X):
    N = X.size
    Xtmp = 2 / N * X
    Xtmp[0] *= 1 / 2
    if N % 2 == 0:
        Xtmp[N // 2] = Xtmp[N // 2] / 2
    return 20 * np.log10(np.abs(Xtmp))

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

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

plt.subplot(3, 1, 1)
plt.plot(f, fft2db(X1wrect), "C0-", lw=1.5, label="500Hz")
plt.plot(f, fft2db(X2wrect), "C3-", lw=1.5, label="500.25Hz")
plt.plot(f, fft2db(X3wrect), "C1-", lw=1.5, label="499.75Hz")
plt.xlim(450, 550)  # Updated range for Variant 8
plt.ylim(-100, 30)
plt.xticks(np.arange(450, 551, 20))
plt.yticks(np.arange(-100, 31, 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="500Hz")
plt.plot(f, fft2db(X2whann), "C3-", lw=1.5, label="500.25Hz")
plt.plot(f, fft2db(X3whann), "C1-", lw=1.5, label="499.75Hz")
plt.xlim(450, 550)  # Updated range for Variant 8
plt.ylim(-100, 10)
plt.xticks(np.arange(450, 551, 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="500Hz")
plt.plot(f, fft2db(X2wflattop), "C3-", lw=1.5, label="500.25Hz")
plt.plot(f, fft2db(X3wflattop), "C1-", lw=1.5, label="499.75Hz")
plt.xlim(450, 550)  # Updated range for Variant 8
plt.ylim(-100, 10)
plt.xticks(np.arange(450, 551, 20))
plt.yticks(np.arange(-100, 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()
```

https://github.com/HubiPX/NOD/tree/master/DSP/DSP%203%204

4. Outcomes:



window~w[k]
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
1600
1800
k
rect
hann
flattop



Normalized DFT Spectra (Rectangular Window)
Amplitude [dB]
20
0
−20
−40
−60
−80
−100
450
470
490
510
530
550
500Hz
500.25Hz
499.75Hz
Normalized DFT Spectra (Hann Window)
Amplitude [dB]
0
−20
−40
−60
−80
−100
450
470
490
510
530
550
500Hz
500.25Hz
499.75Hz
Normalized DFT Spectra (Flat Top Window)
Amplitude [dB]
0
−20
−40
−60
−80
−100
450
470
490
510
530
550
500Hz
500.25Hz
499.75Hz
Frequency [Hz]

5. Conclusions:

The choice of the window depends on what we aim to extract from the signal spectrum analysis. Different windows have their specific advantages and drawbacks, which can affect the quality and accuracy of measurements depending on the application context. If precise frequency localization is important, the rectangular window is the best choice, despite its sidelobe effects. If the goal is to improve overall amplitude analysis quality and reduce spectral leakage, the Hanning window offers a good compromise. For cases requiring exceptional amplitude measurement accuracy, the Flattop window is the optimal choice, though it sacrifices some frequency resolution.

Rectangular Window:
The rectangular window does not modify the signal, allowing for the best frequency resolution. This is ideal when precise frequency localization is crucial in signal analysis. However, this comes with the drawback of strong sidelobes in the spectrum, which can cause interference and complicate the accurate analysis of the signal.

Hanning Window:
The Hanning window strikes a balance between frequency resolution and reducing sidelobe effects. By smoothing the signal edges, this window minimizes spectral leakage and improves amplitude analysis accuracy. While this reduces frequency resolution compared to the rectangular window, the Hanning window is widely used in various applications where improving spectral quality is important.

Flattop Window:
The Flattop window provides very accurate amplitude measurements due to its flat peak in the spectrum. While this comes at the cost of a broader main spectral lobe, it is the best choice when precise amplitude analysis is required, such as in high-precision spectral measurements. It is ideal when amplitude accuracy is paramount, and a slight loss in frequency resolution is acceptable.