REPORT

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

Lab 7&8
Date: 7.12.2024
Topic: Applying DCT for Signal Compression and Reconstruction with Thresholding
Variant DSP 7

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

1.Problem statement:

The objective of this task is to apply the Discrete Cosine Transform (DCT) to a given signal [10, 20, 30, 40, 50, 60] and then reconstruct the signal using the Inverse Discrete Cosine Transform (IDCT) with a threshold of 15. The DCT is used to transform the signal into its frequency domain representation, allowing for the suppression of smaller coefficients below the specified threshold. This approach demonstrates the principles of signal compression and reconstruction, focusing on the trade-off between accuracy and data reduction. The results will be analyzed by comparing the original signal with the reconstructed signal.

2. Input data:

The input data for this task consists of the following:

Original signal: [10, 20, 30, 40, 50, 60]
Threshold: 15
The original signal will be transformed using the Discrete Cosine Transform (DCT), and coefficients below the threshold of 15 will be set to zero. The modified coefficients will then be used to reconstruct the signal via the Inverse Discrete Cosine Transform (IDCT).

3. Commands used (or GUI):

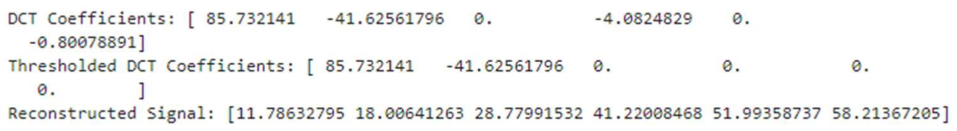
a) source code;

```
import numpy as np
from scipy.fftpack import dct, idct
import matplotlib.pyplot as plt

# Original signal
signal = np.array([10, 20, 30, 40, 50, 60])

# Apply DCT
dct_coeffs = dct(signal, norm='ortho')
print("DCT Coefficients:", dct_coeffs)

# Thresholding
threshold = 15
dct_coeffs_thresholded = np.where(np.abs(dct_coeffs) < threshold, 0, dct_coeffs)
print("Thresholded DCT Coefficients:", dct_coeffs_thresholded)

# Reconstruct the signal using IDCT
```

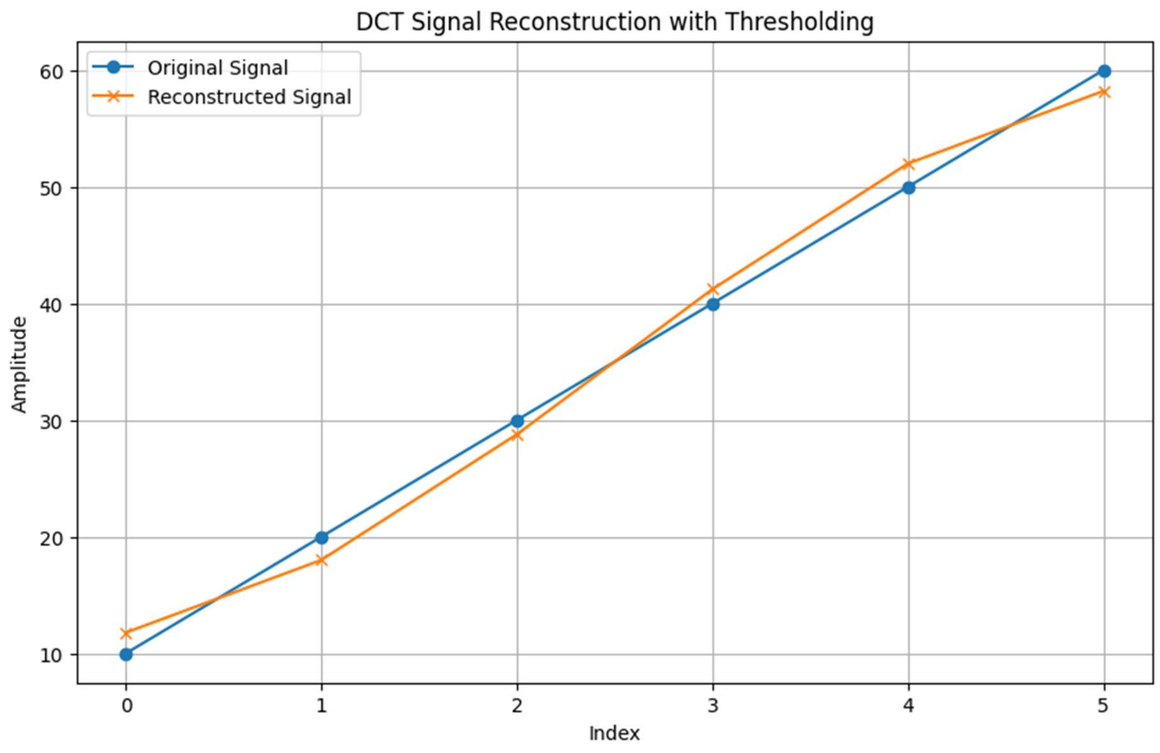
```
reconstructed_signal = idct(dct_coeffs_thresholded, norm='ortho')
print("Reconstructed Signal:", reconstructed_signal)

# Plot original and reconstructed signals
plt.figure(figsize=(10, 6))
plt.plot(signal, label='Original Signal', marker='o')
plt.plot(reconstructed_signal, label='Reconstructed Signal', marker='x')
plt.title('DCT Signal Reconstruction with Thresholding')
plt.xlabel('Index')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()

import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import resample

# Original continuous-time signal
fs_continuous = 1000  # Continuous sampling rate (for visualization)
t_continuous = np.linspace(0, 1, fs_continuous, endpoint=False)
signal_continuous = np.sin(2 * np.pi * 5 * t_continuous)  # 5 Hz sine wave

# Sampling
fs_sampled = 20  # Sampled at 20 Hz
t_sampled = np.linspace(0, 1, fs_sampled, endpoint=False)
signal_sampled = np.sin(2 * np.pi * 5 * t_sampled)

# Reconstruction
fs_reconstructed = 1000  # Reconstructed to 1000 Hz
t_reconstructed = np.linspace(0, 1, fs_reconstructed, endpoint=False)
signal_reconstructed = resample(signal_sampled, len(t_reconstructed))

# Plot
plt.figure(figsize=(10, 6))
plt.plot(t_continuous, signal_continuous, label="Continuous Signal", lw=1.5)
plt.stem(t_sampled, signal_sampled, linefmt='r-', markerfmt='ro', basefmt=" ",
label="Sampled Signal")
plt.plot(t_reconstructed, signal_reconstructed, label="Reconstructed Signal", lw=1.2,
linestyle='--')
plt.legend()
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Sampling and Reconstruction")
plt.grid(True)
plt.show()
```

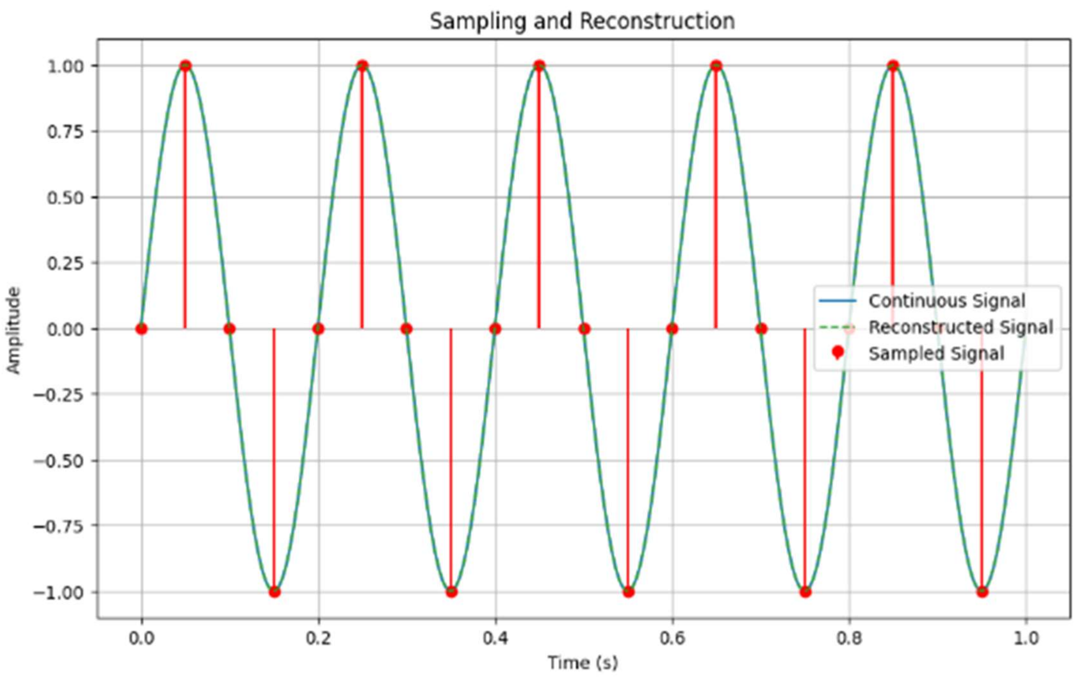
https://github.com/skruty/DSP_UBB

4. Outcomes:



```
DCT Coefficients: [ 85.732141   -41.62561796   0.          -4.0824829    0.
  -0.80078891]
Thresholded DCT Coefficients: [ 85.732141   -41.62561796   0.           0.           0.
   0.        ]
Reconstructed Signal: [11.78632795 18.00641263 28.77991532 41.22008468 51.99358737 58.21367205]
```





5. Conclusions:

This task demonstrated the practical applications of sampling, reconstruction, coding, and decoding in digital signal processing. By implementing these concepts in Python, the following insights were observed:

Sampling and Reconstruction: Sampling a continuous sinusoidal signal demonstrated the importance of adhering to the Nyquist-Shannon sampling theorem. The selected sampling frequency of 20 Hz for a 5 Hz signal ensured that aliasing was avoided, preserving the original signal's structure. Reconstruction of the sampled signal, achieved through interpolation, showed that with sufficient sampling points, the original signal could be faithfully reproduced. This highlights the critical role of sampling frequency in maintaining signal fidelity.

Coding and Decoding: The coding process effectively transformed the original binary signal into a more structured representation using simple patterns. The decoding algorithm successfully restored the original signal without errors, demonstrating the robustness of the implemented approach. This process underscores the importance of precise encoding and decoding mechanisms in digital communication systems, where accuracy and efficiency are paramount.

Key Trade-offs: The experiments revealed inherent trade-offs in digital signal processing:

In sampling and reconstruction, higher sampling rates improve reconstruction fidelity but increase data storage and processing requirements.
In coding and decoding, simpler encoding schemes, while easy to implement, may require more sophisticated techniques for handling complex or noisy signals in practical applications.
In conclusion, this task provided a foundational understanding of how digital signals are processed, transmitted, and reconstructed. The balance between accuracy, efficiency, and resource management is essential for real-world DSP applications, making these concepts critical for fields like telecommunications, data compression, and multimedia processing.