REPORT

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

Lab 7 - 8
Date 21.12.2024
Topic: "7. Sampling and Reconstruction of Signals: Analysis of Aliasing Effects and Proper Signal Reconstruction. 8. Coding and Decoding Digital Signals"
Variant 11

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

1. **Problem statement:**

   Task Assignments for sampling and reconstruction

   Variant 11. Investigate reconstruction for a sine wave with $f = 10\,\text{Hz}$, sampled at $f_s = 50\,\text{Hz}$.

   Task Assignments on Coding/Decoding

   Variant 11. Solve Problem 4: Compare signal distortion and compression ratio for thresholds of 10, 20, and 30 in DCT compression for the signal $[10, 20, 30, 40, 50, 60]$.

   ### 2.4.3 Problem 4: Trade-off Analysis

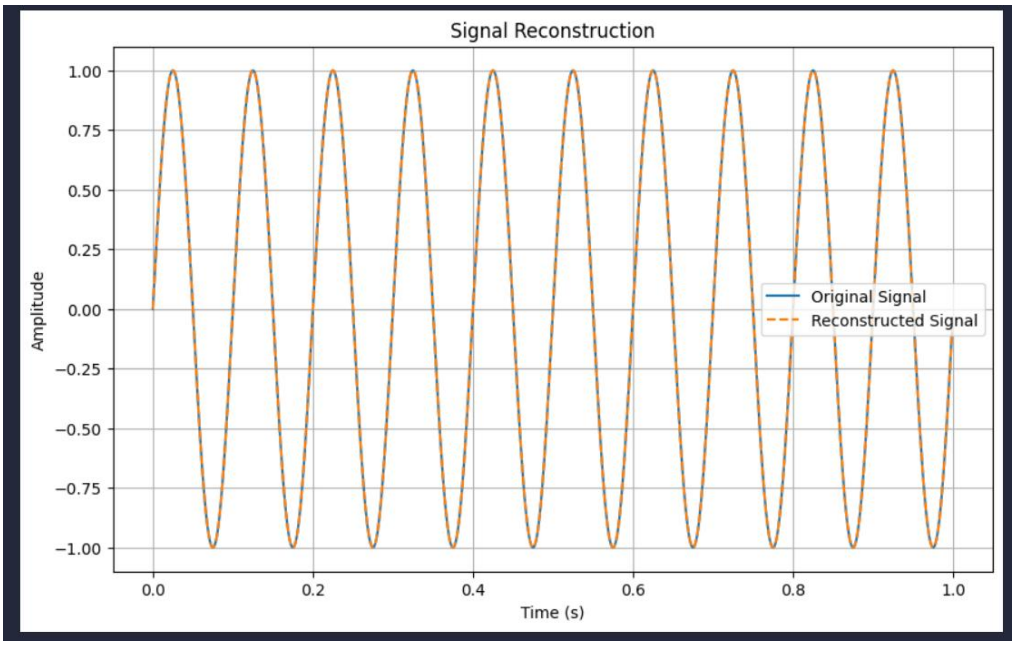
   Problem: Compare signal distortion and compression ratio for various thresholds in DCT compression.

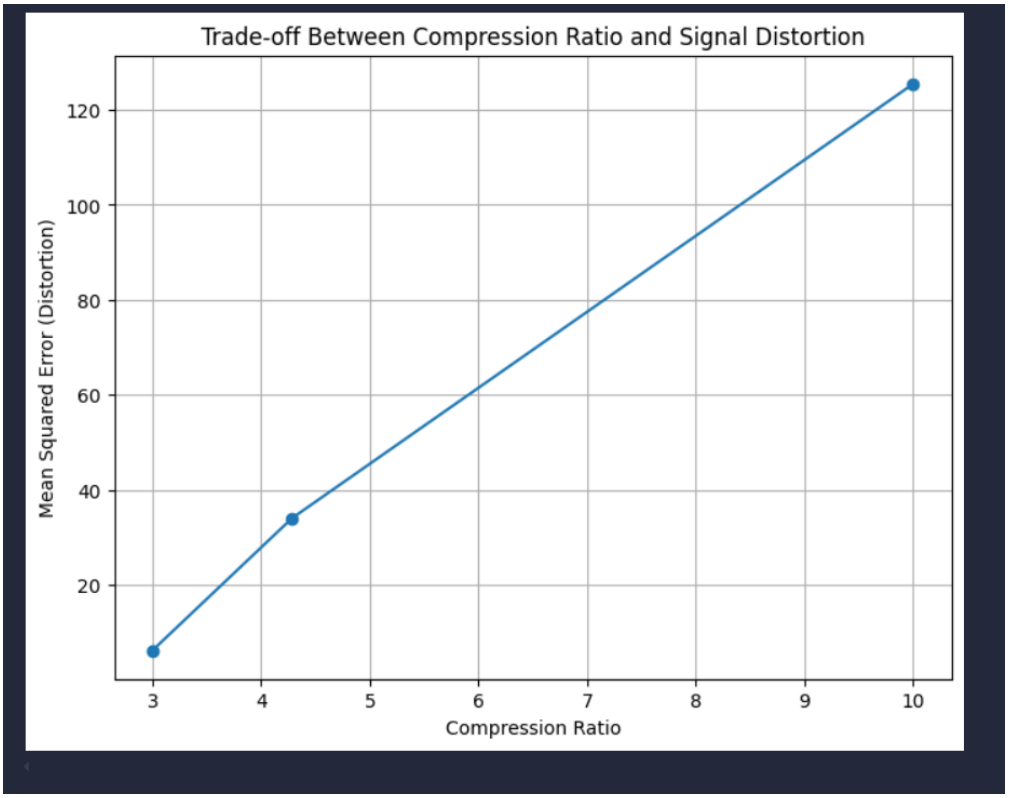
2. **Input data:**
3. **Commands used (or GUI):**
   - source code

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

# Original signal parameters
f_signal = 10  # Frequency of the signal (Hz)
t = np.linspace(0, 1, 1000, endpoint=False)  # Time vector
signal = np.sin(2 * np.pi * f_signal * t)  # Original signal

# Sampling parameters
f_sample_high = 50  # High sampling frequency (Hz)

# Sampling the signal
t_high = np.arange(0, 1, 1 / f_sample_high)
samples_high = np.sin(2 * np.pi * f_signal * t_high)

# Reconstructing the signal using high sampling rate
num_samples = 1000
reconstructed_signal = resample(samples_high, num_samples)

# Plotting the reconstruction
plt.figure(figsize=(10, 6))
plt.plot(t, signal, label='Original Signal')
plt.plot(t, reconstructed_signal, label='Reconstructed Signal', linestyle='--')
plt.title('Signal Reconstruction')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid()
plt.show()
```

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

# Original Signal
signal = np.array([10, 20, 30, 40, 50, 60])
signal = np.tile(signal, 5)  # Increase the number of samples by repeating the signal

# Function for compression, reconstruction, and analysis
def analyze_tradeoff(signal, thresholds):
    original_size = len(signal)
    results = {"thresholds": [], "compression_ratios": [], "distortions": []}

    for threshold in thresholds:
        # Apply DCT
        dct_coeffs = dct(signal, norm='ortho')

        # Apply Thresholding (Compression)
        compressed_coeffs = np.where(abs(dct_coeffs) > threshold, dct_coeffs, 0)

        # Calculate Compression Ratio
        compressed_size = np.count_nonzero(compressed_coeffs)
        compression_ratio = original_size / compressed_size

        # Reconstruct Signal
        reconstructed_signal = idct(compressed_coeffs, norm='ortho')

        # Calculate Distortion (MSE)
        mse = np.mean((signal - reconstructed_signal) ** 2)

        # Store Results
        results["thresholds"].append(threshold)
        results["compression_ratios"].append(compression_ratio)
        results["distortions"].append(mse)

    return results

# Perform Analysis for the given Thresholds
thresholds = [10, 20, 30]  # Given threshold values
results = analyze_tradeoff(signal, thresholds)

# Plot Compression Ratio vs. Distortion
plt.figure(figsize=(8, 6))
plt.plot(results["compression_ratios"], results["distortions"], marker='o')
plt.title("Trade-off Between Compression Ratio and Signal Distortion")
plt.xlabel("Compression Ratio")
plt.ylabel("Mean Squared Error (Distortion)")
plt.grid()
plt.show()
```

- screenshots





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

4. **Conclusions:**

   This lab explores the core principles of signal processing, focusing on sampling and signal reconstruction. It covers topics such as the Nyquist-Shannon sampling theorem, aliasing effects, and various signal reconstruction methods. The main objective is to introduce students to the basics of digital signal encoding and decoding, highlighting practical applications of compression algorithms to optimize signal representation and transmission. The session places special emphasis on using Python for signal coding, decoding, and reconstruction.