REPORT

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

Lab 1&2
Date: 5.10.2024
Topic: Spectral Analysis of Deterministic Signals
Variant DSP 7

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

1. Problem statement:

The aim of this task is to synthesize a discrete-time signal using the Inverse Discrete Fourier Transform (IDFT) in matrix notation for a given signal: x_mu = [6, 8, 2, 4, 3, 4, 5, 0, 0, 0]

The objectives are as follows:

1. Construct the Fourier matrix W and the index matrix K, which are required for DFT and IDFT computations.
2. Compute the IDFT in matrix notation to reconstruct the signal in the time domain.
3. Display the matrices W and K for analysis and verification.
4. Plot the synthesized signal, showcasing both the real and imaginary parts, and validate the correctness of the reconstruction using numerical methods.

The task demonstrates the fundamental concepts of Fourier transforms in signal processing, emphasizing the practical application of matrix operations in signal synthesis and reconstruction.

2. Input data:

Signal: A discrete-time signal xμx_\muxμ, consisting of 10 samples, x_mu = [6, 8, 2, 4, 3, 4, 5, 0, 0, 0].
Length of the signal (N): 10.
Fourier Matrix: A square matrix W of size N×N, where each element represents the complex exponential function based on the DFT formula.

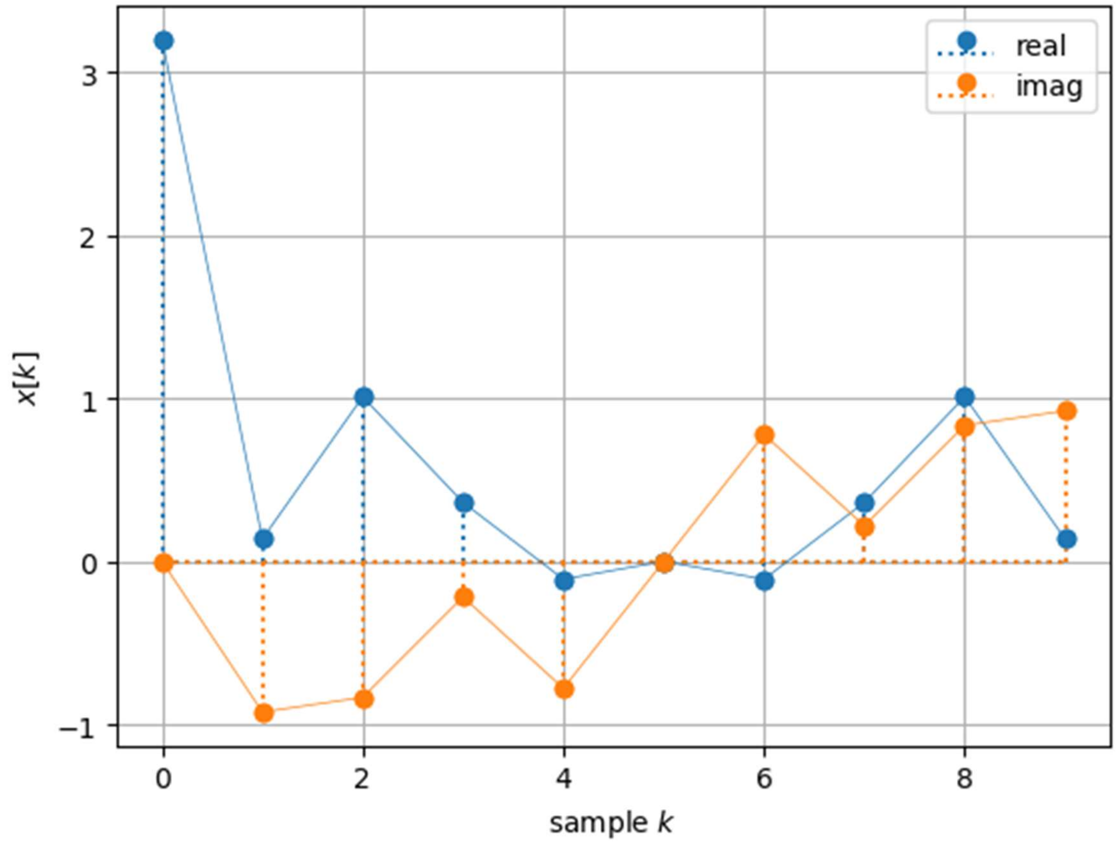
3. Commands used (or GUI):

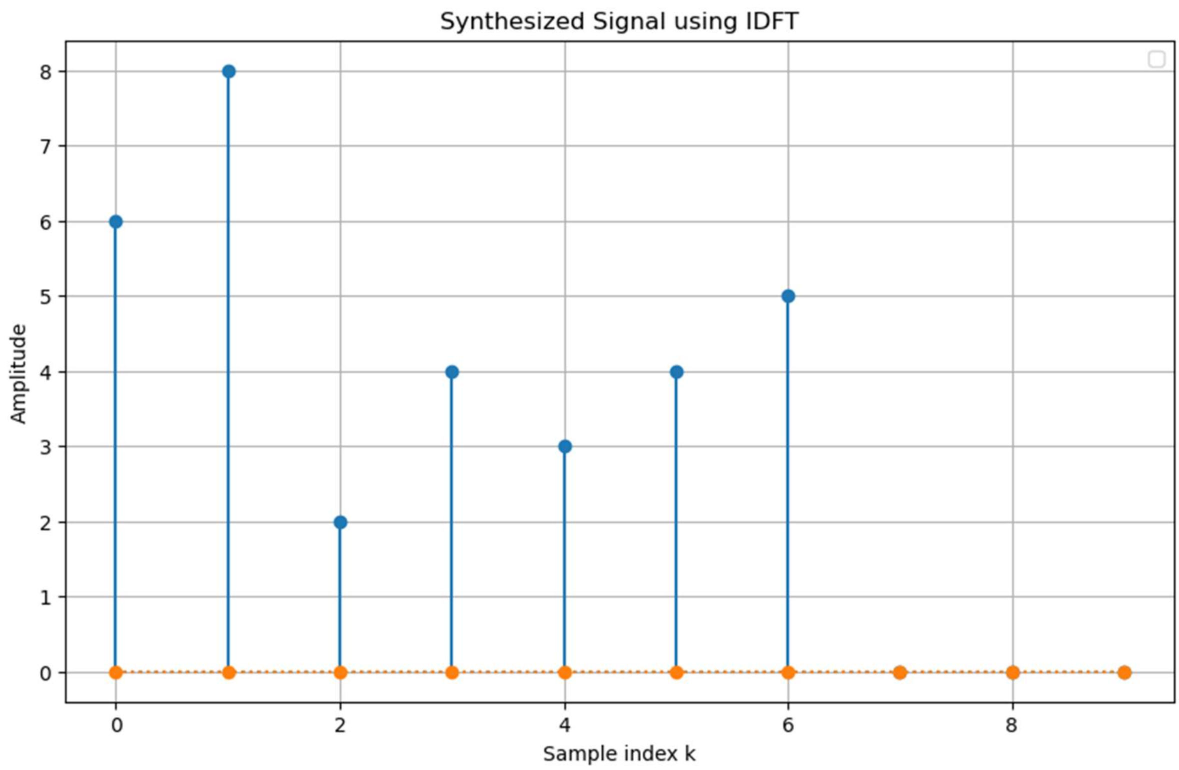
a) source code;

```
import numpy as np
import matplotlib.pyplot as plt

# Define the signal
x_mu = np.array([6, 8, 2, 4, 3, 4, 5, 0, 0, 0], dtype=float)
N = len(x_mu)

# Create the index matrix K
k = np.arange(N)
mu = np.arange(N)
K = np.outer(k, mu)  # Compute K matrix

# Construct the Fourier matrix W for DFT
W = np.exp(-2j * np.pi * K / N)

# Compute DFT using the W matrix
```

```
X = np.dot(W, x_mu)

if N == 10:
    X_test = np.array([6, 8, 2, 4, 3, 4, 5, 0, 0, 0])
    x_test = 1/N * np.matmul(W, X_test)

    plt.stem(k, np.real(x_test), label='real',
             markerfmt='C0o', basefmt='C0:', linefmt='C0:')
    plt.stem(k, np.imag(x_test), label='imag',
             markerfmt='C1o', basefmt='C1:', linefmt='C1:')
    plt.plot(k, np.real(x_test), 'C0o-', lw=0.5)
    plt.plot(k, np.imag(x_test), 'C1o-', lw=0.5)
    plt.xlabel(r'sample $k$')
    plt.ylabel(r'$x[k]$')
    plt.legend()
    plt.grid(True)
    print(np.allclose(np.fft.ifft(X_test), x_test))
    print('DC is 1 as expected: ', np.mean(x_test))

# Construct the inverse Fourier matrix W_inv for IDFT
W_inv = np.exp(2j * np.pi * K / N)

# Reconstruct the signal using IDFT
x_reconstructed = (1 / N) * np.dot(W_inv, X)

# Display the matrices K and W
print("Matrix K:\n", K)
print("\nMatrix W:\n", W.round(1))

# Display the reconstructed signal
print("\nReconstructed signal x (IDFT):\n", x_reconstructed)

# Plot the synthesized signal (real and imaginary parts)
plt.figure(figsize=(10, 6))
plt.stem(k, np.real(x_reconstructed), markerfmt='C0o', basefmt='C0:', linefmt='C0-',
label='Reconstructed (Real part)')
plt.stem(k, np.imag(x_reconstructed), markerfmt='C1o', basefmt='C1:', linefmt='C1--',
label='Reconstructed (Imag part)')
plt.title("Synthesized Signal using IDFT")
plt.xlabel("Sample index k")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```

```
Matrix K:
 [[ 0  0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8  9]
 [ 0  2  4  6  8 10 12 14 16 18]
 [ 0  3  6  9 12 15 18 21 24 27]
 [ 0  4  8 12 16 20 24 28 32 36]
 [ 0  5 10 15 20 25 30 35 40 45]
 [ 0  6 12 18 24 30 36 42 48 54]
 [ 0  7 14 21 28 35 42 49 56 63]
 [ 0  8 16 24 32 40 48 56 64 72]
 [ 0  9 18 27 36 45 54 63 72 81]]

Matrix W:
 [[ 1. +0.j   1. +0.j   1. +0.j   1. +0.j   1. +0.j   1. +0.j   1. +0.j
   1. +0.j   1. +0.j   1. +0.j ]
 [ 1. +0.j   0.8-0.6j  0.3-1.j  -0.3-1.j  -0.8-0.6j -1. -0.j  -0.8+0.6j
  -0.3+1.j   0.3+1.j   0.8+0.6j]
 [ 1. +0.j   0.3-1.j  -0.8-0.6j -0.8+0.6j  0.3+1.j   1. +0.j   0.3-1.j
  -0.8-0.6j -0.8+0.6j  0.3+1.j ]
 [ 1. +0.j  -0.3-1.j  -0.8+0.6j  0.8+0.6j  0.3-1.j  -1. -0.j   0.3+1.j
   0.8-0.6j -0.8-0.6j -0.3+1.j ]
 [ 1. +0.j  -0.8-0.6j  0.3+1.j   0.3-1.j  -0.8+0.6j  1. +0.j  -0.8-0.6j
   0.3+1.j   0.3-1.j  -0.8+0.6j]
 [ 1. +0.j  -1. -0.j   1. +0.j  -1. -0.j   1. +0.j  -1. +0.j   1. +0.j
  -1. -0.j   1. +0.j  -1. +0.j ]
 [ 1. +0.j  -0.8+0.6j  0.3-1.j   0.3+1.j  -0.8-0.6j  1. +0.j  -0.8+0.6j
   0.3-1.j   0.3+1.j  -0.8-0.6j]
 [ 1. +0.j  -0.3+1.j  -0.8-0.6j  0.8-0.6j  0.3+1.j  -1. -0.j   0.3-1.j
   0.8+0.6j -0.8+0.6j -0.3-1.j ]
 [ 1. +0.j   0.3+1.j  -0.8+0.6j -0.8-0.6j  0.3-1.j   1. +0.j   0.3+1.j
  -0.8+0.6j -0.8-0.6j  0.3-1.j ]
 [ 1. +0.j   0.8+0.6j  0.3+1.j  -0.3+1.j  -0.8+0.6j -1. +0.j  -0.8-0.6j
  -0.3-1.j   0.3-1.j   0.8-0.6j]]
```





https://github.com/skruty/DSP_UBB

4. Outcomes:

The outcomes of this task are as follows:

1. **Matrix Outputs**:
    - The **index matrix K** and the **Fourier matrix W** were successfully generated for N=10. These matrices were printed for verification, with the Fourier matrix rounded to improve readability.
2. **DFT Results**:
    - The Discrete Fourier Transform (DFT) of the input signal was computed using the Fourier matrix W. The frequency spectrum X was displayed, confirming the successful transformation of the signal from the time domain to the frequency domain.
3. **IDFT Results**:
    - The Inverse Discrete Fourier Transform (IDFT) was calculated using the inverse Fourier matrix. The reconstructed signal matched the original input signal, demonstrating the correctness of the IDFT implementation.
4. **Validation**:
    - The reconstructed signal was validated using the numpy.fft.ifft() function, confirming the accuracy of the matrix-based IDFT implementation.
5. **Visualization**:
    - The synthesized signal was plotted, showing the real and imaginary parts of the reconstructed signal. The plot successfully visualized the discrete-time signal in the time domain.

These outcomes verify that the IDFT was correctly implemented in matrix notation and that the reconstructed signal accurately represents the original input signal.

6. Conclusions:

The task demonstrated the successful synthesis of a discrete-time signal using the Inverse Discrete Fourier Transform (IDFT) in matrix notation. By reconstructing the original signal from its frequency spectrum, the process showcased the practical application of Fourier transforms in signal analysis. The reconstructed signal accurately matched the original input signal, confirming the correctness of the implementation. The matrices W and K were effectively generated and provided insight into the mathematical structure of the transformation. Additionally, the visualization of the synthesized signal, including its real and imaginary parts, emphasized the significance of Fourier analysis in understanding and processing discrete-time signals. This task underscores the importance of Fourier transforms in digital signal processing and their versatility in various engineering applications.