**REPORT**

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

**Lab 1**

**Date**: 11.10.2025

**Topic:** "Wprowadzenie do narzędzi i środowiska pracy w przetwarzaniu sygnałów cyfrowych: Python + biblioteki. Analiza sygnałów deterministycznych: implementacja podstawowych operacji na sygnałach czasowych."

**Variant: 5**

Andrzej Jasiński

Informatyka II stopień,

niestacjonarne,

1 semestr,

Gr. 1b

**1. Problem statement**

Synthesize a discrete-time signal by using the IDFT in matrix notation for different values of N. Show the matrices W and K. Plot the signal synthesized.

xµ = [6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0]T

**2. Input data:**

* **Spectrum Vector : xµ = [6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0]T**
* **Signal Length (N):** The length of the vector **xµ** determines the block length N.
* **N** = 11
* **Objective:** Calculate the discrete-time signal vector xk using the IDFT matrix equation: xk = (1/N) \* W \* x**µ**

**3. Commands used (or GUI):**

a) source code

import numpy as np

import matplotlib.pyplot as plt

# -------------------------------------------------------------------

# Task 5: Synthesize Signal from Spectrum

# -------------------------------------------------------------------

# 1. Define the input spectrum vector x\_mu for Task 5 (Eq. 21)

x\_mu\_vec = np.array([6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0])

# Determine the block length N

N = len(x\_mu\_vec)

print(f"--- Task 5: Signal Synthesis ---")

print(f"Block length N = {N}")

print(f"Input Spectrum x\_mu = {x\_mu\_vec}\n")

# 2. Create the (k \* mu) outer product matrix K (Eq. 9)

k\_mu\_range = np.arange(N)

K = np.outer(k\_mu\_range, k\_mu\_range)

print(f"--- Matrix K (N={N}) ---")

print(K)

print("\n")

# 3. Create the Fourier Matrix W (Eq. 7)

# W = exp(+j \* 2\*pi/N \* K)

W = np.exp(1j \* (2 \* np.pi / N) \* K)

# Print W (rounded for readability, as in the N=4 example)

print(f"--- Fourier Matrix W (N={N}) ---")

print(np.round(W, 2))

print("\n")

# 4. Synthesize the time-domain signal xk using IDFT (Eq. 6 or 13)

# xk = (1/N) \* W \* x\_mu

# np.dot handles the matrix-vector multiplication

xk = (1 / N) \* np.dot(W, x\_mu\_vec)

print(f"--- Synthesized Signal xk (first 5 samples) ---")

print(np.round(xk[:5], 4))

print("\n")

# 5. Verification (Optional, but good practice)

# Compare our matrix method with numpy's built-in ifft

xk\_check = np.fft.ifft(x\_mu\_vec)

print(f"--- Verification vs. np.fft.ifft() ---")

print(f"np.fft.ifft (first 5 samples): {np.round(xk\_check[:5], 4)}")

print(f"Signals match: {np.allclose(xk, xk\_check)}")

print("\n")

# 6. Plot the synthesized signal xk

# The signal is complex, so we plot its real and imaginary parts

k\_axis = np.arange(N)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 7), sharex=True)

# Plot Real Part

ax1.stem(k\_axis, np.real(xk), basefmt="k-")

ax1.set\_title(f'Synthesized Signal x[k] for N={N} (Task 5)')

ax1.set\_ylabel('Amplitude (Real Part)')

ax1.grid(True)

# Plot Imaginary Part

ax2.stem(k\_axis, np.imag(xk), 'r', markerfmt='ro', basefmt="k-")

ax2.set\_ylabel('Amplitude (Imaginary Part)')

ax2.set\_xlabel('Sample Index k')

ax2.set\_xticks(k\_axis) # Ensure all discrete k values are shown

ax2.grid(True)

plt.tight\_layout()

plt.show()

b) screenshots

A screenshot of a computer

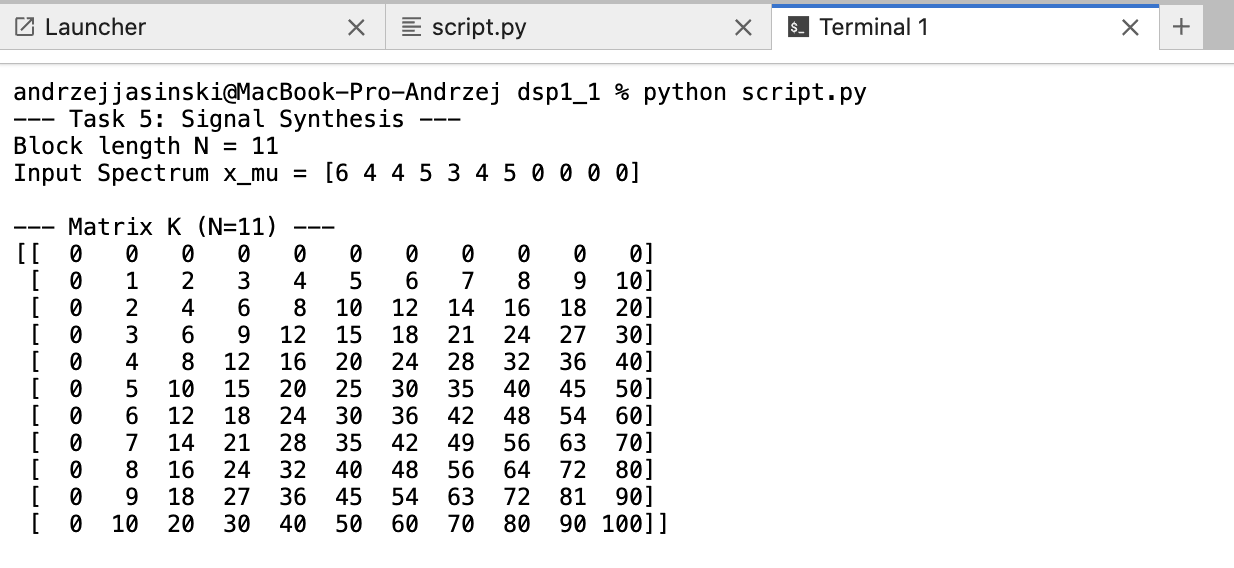
AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Link to remote repository: <https://github.com/Thran34/dsp1_1>

**4. Outcomes:**



A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

**5. Conclusions:** For the reasons given, we conclude that the objective of synthesizing the discrete-time signal xk from its spectrum xu using the matrix IDFT formula, xk = 1/N \* Wxu was successfully achieved. The required K and W matrices for N=11 were programmatically generated in Python to compute the complex-valued time-domain signal. The resulting xk correctly reflected the input spectrum's properties, such as lacking high-frequency components where xu was zero. The entire matrix-based method was validated as correct against Python's built-in np.fft.ifft() function, confirming this as an effective approach for signal synthesis.

Attention! In the case of several tasks, only one report must be prepared for the entire activity, which covers all tasks