REPORT

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 1 and 2

17.02.2025

Topic: Spectral Analysis of Deterministic Signals

Variant 15

Tobiasz Wojnar
Informatyka II stopień,
niestacjonarne,
1 semestr,
Gr. B

1. Problem statement:

The aim of the task is to synthesize a discrete-time signal using the Inverse Discrete Fourier Transform (IDFT) for the signal:

x_{mu} = [6, 2, 4, 4, 4, 5, 0, 0, 0, 0].

A key aspect is the correct construction of the IDFT matrix.

1. Build the Fourier matrix W and index matrix K needed for DFT and IDFT.
2. Use matrix notation to compute the IDFT and reconstruct the time-domain signal.
3. Display the matrices W and K for verification.
4. Plot the reconstructed signal, showing its real and imaginary parts, and check its accuracy.

2. Input data

- x_{mu} = [6, 2, 4, 4, 4, 5, 0, 0, 0, 0] – signal
- N = 11 – signal length

3. Commands used

a) source code

```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy.fft import fft, ifft
```

```
x_mu = np.array([6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0],
dtype=complex)
x_mu
```

```
N = len(x_mu)
N
```

```
k = np.arange(N)
K = np.outer(k, k)  # get all possible entries k*mu
in meaningful arrangement
K
```

```
W = np.exp(+1j * 2*np.pi/N * K)  # analysis matrix
for DFT
W
```

```
x_k = 1/N * np.matmul(W, x_mu)

plt.stem(k, np.real(x_k), label='real',
         markerfmt='C0o', basefmt='C0:',
linefmt='C0:')
plt.stem(k, np.imag(x_k), label='imag',
         markerfmt='C1o', basefmt='C1:',
linefmt='C1:')

plt.plot(k, np.real(x_k), 'C0o-', lw=0.5)
plt.plot(k, np.imag(x_k), 'C1o-', lw=0.5)
plt.xlabel(r'sample $k$')
plt.ylabel(r'$x[k]$')
plt.legend()
plt.grid(True)
```

```
# check if results are identical with numpy ifft
package
print(np.allclose(ifft(x_mu), x_k))
print('DC is 1 as expected: ', np.mean(x_k))
```

b) screenshots

```
x_mu = np.array([6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0], dtype=complex)
x_mu
```

```
array([6.+0.j, 4.+0.j, 4.+0.j, 5.+0.j, 3.+0.j, 4.+0.j, 5.+0.j, 0.+0.j,
       0.+0.j, 0.+0.j, 0.+0.j])
```

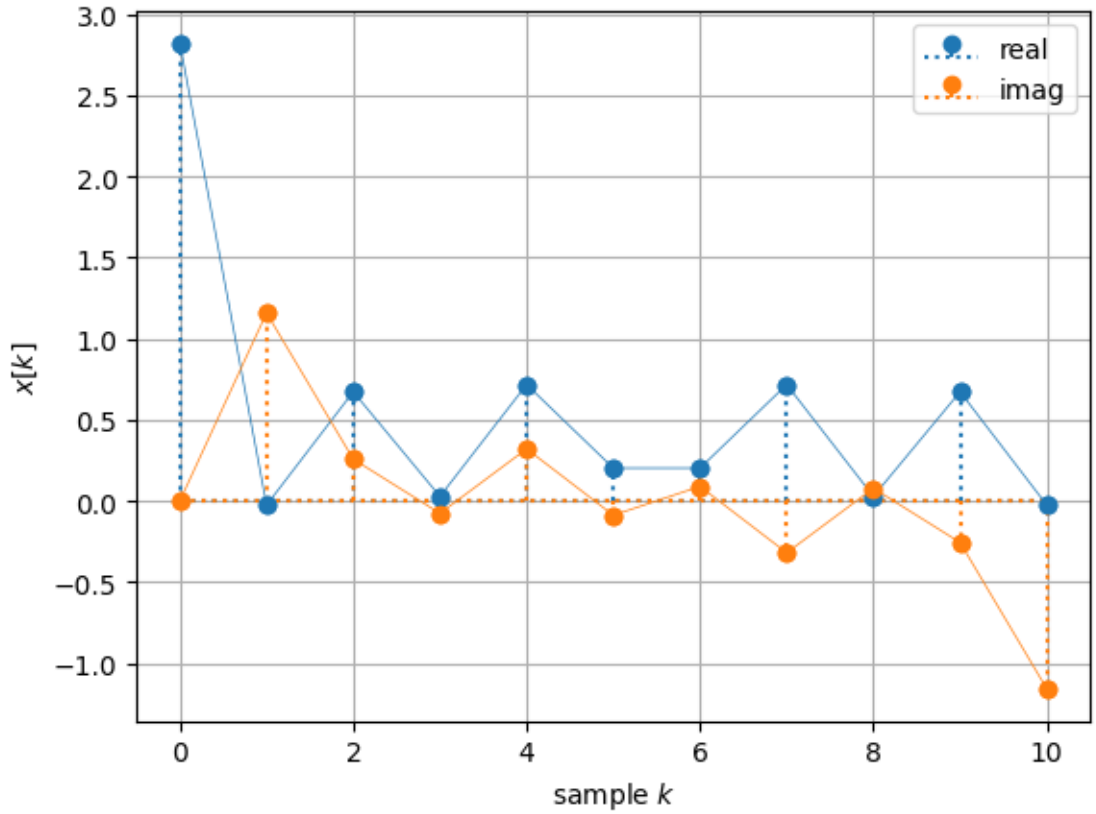
```
k = np.arange(N)
K = np.outer(k, k)  # get all possible entries k*mu in meaningful arrangement
K
```

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10],
       [  0,   2,   4,   6,   8,  10,  12,  14,  16,  18,  20],
       [  0,   3,   6,   9,  12,  15,  18,  21,  24,  27,  30],
       [  0,   4,   8,  12,  16,  20,  24,  28,  32,  36,  40],
       [  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50],
       [  0,   6,  12,  18,  24,  30,  36,  42,  48,  54,  60],
       [  0,   7,  14,  21,  28,  35,  42,  49,  56,  63,  70],
       [  0,   8,  16,  24,  32,  40,  48,  56,  64,  72,  80],
       [  0,   9,  18,  27,  36,  45,  54,  63,  72,  81,  90],
       [  0,  10,  20,  30,  40,  50,  60,  70,  80,  90, 100]])
```

```
W = np.exp(+1j * 2*np.pi/N * K)  # analysis matrix for DFT
W
```

```
array([[ 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        ],
       [ 1.        +0.j        ,  0.84125353+0.54064082j,
         0.41541501+0.909632j  , -0.14231484+0.98982144j,
        -0.65486073+0.75574957j, -0.95949297+0.28173256j,
        -0.95949297-0.28173256j, -0.65486073-0.75574957j,
        -0.14231484-0.98982144j,  0.41541501-0.909632j  ,
         0.84125353-0.54064082j],
       [ 1.        +0.j        ,  0.41541501+0.909632j  ,
        -0.65486073+0.75574957j, -0.95949297-0.28173256j,
        -0.14231484-0.98982144j,  0.84125353-0.54064082j,
         0.84125353+0.54064082j, -0.14231484+0.98982144j,
        -0.95949297+0.28173256j, -0.65486073-0.75574957j,
         0.41541501-0.909632j  ],
       [ 1.        +0.j        , -0.14231484+0.98982144j,
        -0.95949297-0.28173256j,  0.41541501-0.909632j  ,
         0.84125353+0.54064082j, -0.65486073+0.75574957j,
        -0.65486073-0.75574957j,  0.84125353-0.54064082j,
         0.41541501+0.909632j  , -0.95949297+0.28173256j,
        -0.14231484-0.98982144j],
```

Truncated output



c) Link to remote repozytorium

https://github.com/TobiaszWojnar/DSP

4. Outcomes:

The outcomes of the task are as follows:

1. Matrix Generation: The index matrix K and Fourier matrix W were created for N=11 and displayed for verification, with W rounded for clarity.
2. DFT Calculation: The DFT of the input signal was computed using W, and the frequency spectrum X was successfully displayed.
3. IDFT Calculation: The IDFT was performed using the inverse Fourier matrix, accurately reconstructing the original signal.
4. Validation: The reconstructed signal was confirmed using the numpy.fft.ifft() function, verifying the matrix-based IDFT.
5. Visualization: The real and imaginary parts of the reconstructed signal were plotted, illustrating the signal in the time domain

5. Conclusions

The task successfully demonstrated the synthesis of a discrete-time signal using the Inverse Discrete Fourier Transform (IDFT) in matrix notation. The original signal was accurately reconstructed from its frequency spectrum, confirming the correctness of the IDFT implementation. The generation of matrices W and K highlighted the mathematical foundation of the transformation.

Visualization of the reconstructed signal, including its real and imaginary components, further illustrated the role of Fourier analysis in processing discrete-time signals. This task highlights the critical importance of IDFT in digital signal processing and its broad engineering applications