REPORT

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

Lab 01
Date 28.09.2024
Topic: "Spectral Analysis of Deterministic Signals"
Variant 11

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

1. Problem statement:

The DFT and IDFT basically solve two sets of linear equations, that are linked as forward and inverse problem.

This is revealed with the important property of the Fourier matrix

$$\mathbf{W}^{-1} = \frac{\mathbf{W}^{\mathrm{H}}}{N} = \frac{\mathbf{W}^{*}}{N},$$

the latter holds since the matrix is symmetric.

Thus, we see that by our convention, the DFT is the inverse problem (signal analysis) and the IDFT is the forward problem (signal synthesis)

$$\text{DFT: } \mathbf{x}_{\mu} = \mathbf{W}^{*}\mathbf{x}_{k} \rightarrow \mathbf{x}_{\mu} = N\mathbf{W}^{-1}\mathbf{x}_{k}$$
$$\text{IDFT: } \mathbf{x}_{k} = \frac{1}{N}\mathbf{W}\mathbf{x}_{\mu}.$$

The occurrence of the N, $1/N$ factor is due to the prevailing convention in signal processing literature.

If the matrix is normalised as $\frac{\mathbf{W}}{\sqrt{N}}$, a so called unitary matrix results, for which the important property

$$(\frac{\mathbf{W}}{\sqrt{N}})^{\mathrm{H}}(\frac{\mathbf{W}}{\sqrt{N}}) = \mathbf{I} = (\frac{\mathbf{W}}{\sqrt{N}})^{-1}(\frac{\mathbf{W}}{\sqrt{N}})$$

holds, i.e. the complex-conjugate, transpose is equal to the inverse $(\frac{\mathbf{W}}{\sqrt{N}})^{\mathrm{H}} = (\frac{\mathbf{W}}{\sqrt{N}})^{-1}$ and due to the matrix symmetry also $(\frac{\mathbf{W}}{\sqrt{N}})^{*} = (\frac{\mathbf{W}}{\sqrt{N}})^{-1}$ is valid.

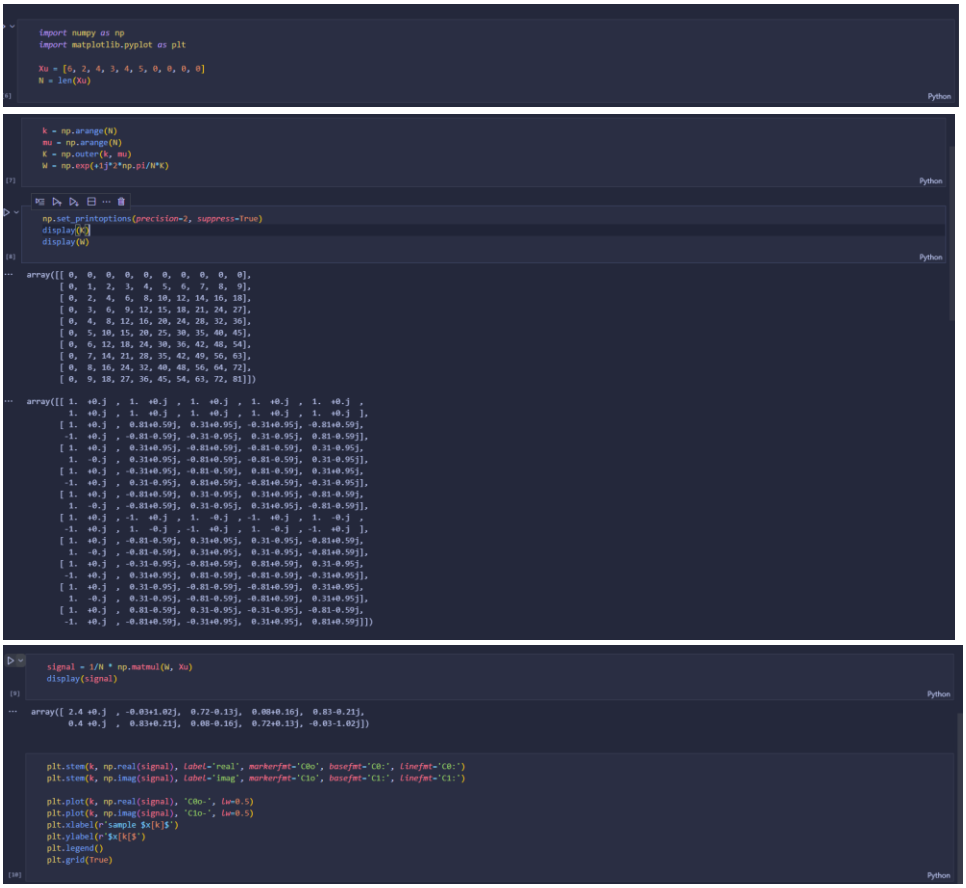
This tells that the matrix $\frac{\mathbf{W}}{\sqrt{N}}$ is **orthonormal**, i.e. the matrix spans a orthonormal vector basis (the best what we can get in linear algebra world to work with) of N normalized DFT eigensignals.

So, DFT and IDFT is transforming vectors into other vectors using the vector basis of the Fourier matrix.

2. **Input data:**

   Variant 11: $\boldsymbol{x_\mu = [6, 2, 4, 3, 4, 5, 0, 0, 0, 0]^T}$

3. **Commands used (or GUI):**
   a) source code



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

Xu = [6, 2, 4, 3, 4, 5, 0, 0, 0, 0]
N = len(Xu)
```

```
k = np.arange(N)
mu = np.arange(N)
K = np.outer(k, mu)
W = np.exp(+1j*2*np.pi/N*K)
```

```
np.set_printoptions(precision=2, suppress=True)
display(K)
display(W)
```

```
array([[ 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]])

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  ,  0.81+0.59j,  0.31+0.95j, -0.31+0.95j, -0.81+0.59j,
        -1.  +0.j  , -0.81-0.59j, -0.31-0.95j,  0.31-0.95j,  0.81-0.59j],
       [ 1.  +0.j  ,  0.31+0.95j, -0.81+0.59j, -0.81-0.59j,  0.31-0.95j,
         1.  -0.j  ,  0.31+0.95j, -0.81+0.59j, -0.81-0.59j,  0.31-0.95j],
       [ 1.  +0.j  , -0.31+0.95j, -0.81-0.59j,  0.81-0.59j,  0.31+0.95j,
        -1.  +0.j  ,  0.31-0.95j,  0.81+0.59j, -0.81+0.59j, -0.31-0.95j],
       [ 1.  +0.j  , -0.81+0.59j,  0.31-0.95j,  0.31+0.95j, -0.81-0.59j,
         1.  -0.j  , -0.81+0.59j,  0.31-0.95j,  0.31+0.95j, -0.81-0.59j],
       [ 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  , -0.81-0.59j,  0.31+0.95j,  0.31-0.95j, -0.81+0.59j,
         1.  -0.j  , -0.81-0.59j,  0.31+0.95j,  0.31-0.95j, -0.81+0.59j],
       [ 1.  +0.j  , -0.31-0.95j, -0.81+0.59j,  0.81+0.59j,  0.31-0.95j,
        -1.  +0.j  ,  0.31+0.95j,  0.81-0.59j, -0.81-0.59j, -0.31+0.95j],
       [ 1.  +0.j  ,  0.31-0.95j, -0.81-0.59j, -0.81+0.59j,  0.31+0.95j,
         1.  -0.j  ,  0.31-0.95j, -0.81-0.59j, -0.81+0.59j,  0.31+0.95j],
       [ 1.  +0.j  ,  0.81-0.59j,  0.31-0.95j, -0.31-0.95j, -0.81-0.59j,
        -1.  +0.j  , -0.81+0.59j, -0.31+0.95j,  0.31+0.95j,  0.81+0.59j]])
```

```
signal = 1/N * np.matmul(W, Xu)
display(signal)
```

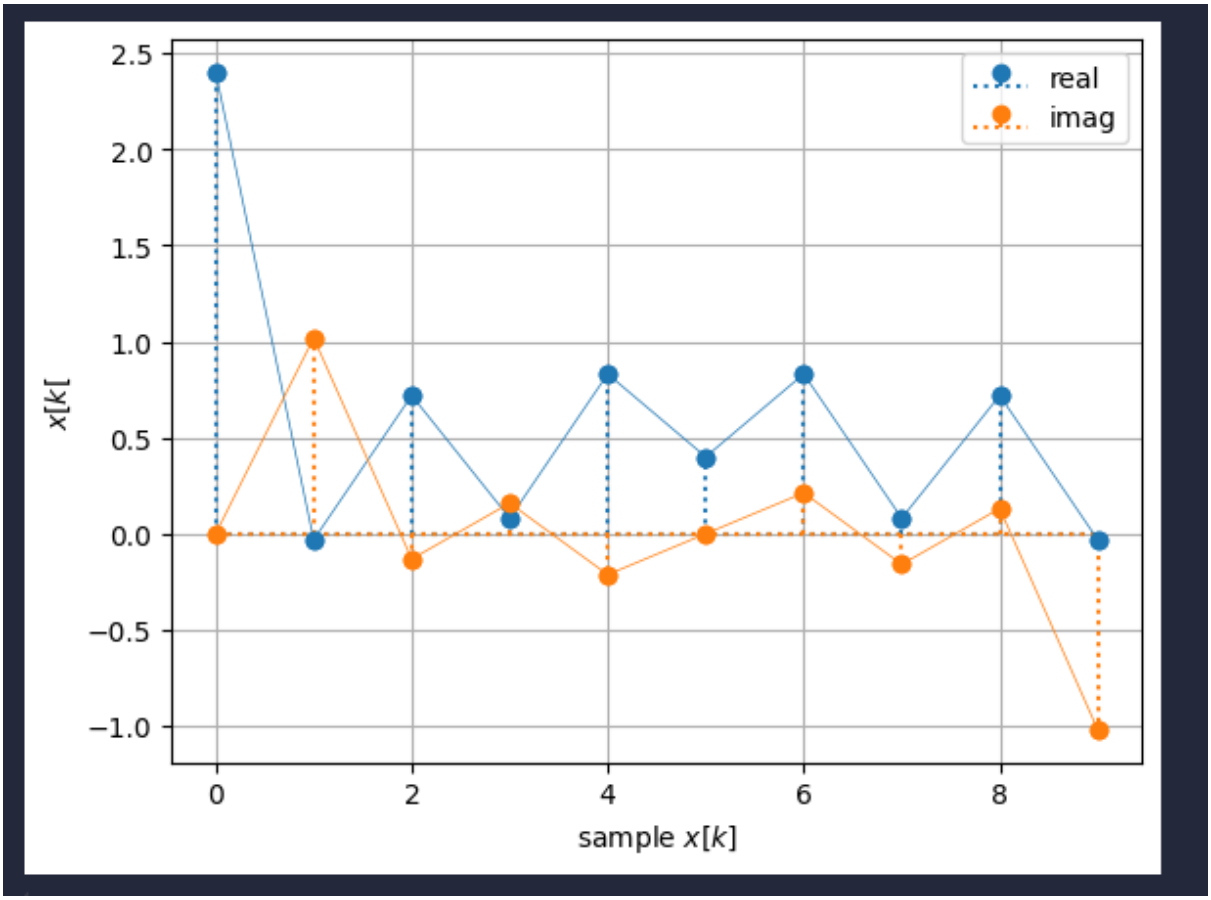
```
array([ 2.4 +0.j  , -0.03+1.02j,  0.72-0.13j,  0.08+0.16j,  0.83-0.21j,
        0.4 +0.j  ,  0.83+0.21j,  0.08-0.16j,  0.72+0.13j, -0.03-1.02j])
```

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

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

Link to remote repozytorium:
https://github.com/Maciek332/Semestr_1_Nycz/tree/master/DSPja/Lab_1

4. **Outcomes:**



5. **Conclusions:**

For the reasons given, we conclude that thanks to convention used in Python's 'numpy.fft.fft()', 'numpy.fft.ifft()' and Matlab's 'fft()', 'ifft()' it's easy for us to note the sign reversal in the exp()-function and the 1/N normalization in the IDFT. Our task was to synthesize a discrete-time signal by using the IDFT in matrix notation for different values of N and to show the matrices W and K. Plot the signal synthesized. Thanks to all knowledge in materials we were able to make this task with not so much effort as we would have to without them.