REPORT

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

Lab 1
08.03.2024
Topic: "Spectral Analisys of Deterministic Signals"
Variant 5

Michał Herczyński
Informatyka II stopień,
stacjonarne,
2 semestr,
Gr.1b

1. Problem statement:

The primary objective of this exercise is to synthesize a discrete-time signal using the Inverse Discrete Fourier Transform (IDFT) in matrix notation. The anticipated outcome involves recreating and plotting the signal based on frequency indices. It is essential to present the matrices W and K.

2. Input data:

Input data is DTF frequency indices, given as

Xmu = [6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0]

3. Commands used (or GUI):

a) source code

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import ifft

def main():
input_array = []
print("Defining Xmu table ('f' for finish)")
while True:
print(input_array)
ipt = input(">> ")
if ipt == "f": break
try:
input_array.append(int(ipt))
except:
print("It is not a number.")

mu = np.array(input_array)
N = len(mu) # Długość tablicy mu
k = np.arange(N)
A = 10

print("getting all possible entries k")
K = np.outer(k, np.arange(N))
print(K)

print("Fourier matrix setting up")
W = np.exp(+1j * 2*np.pi/N * K)
print(W)
```

```
fig, ax = plt.subplots(1, N)
fig.set_size_inches(6, 6)
fig.suptitle(
r'Fourier Matrix for $N=$%d, blue: $\Re(\mathrm{e}^{+\mathrm{j} \frac{2\pi}{N} \mu k})$,
orange: $\Im(\mathrm{e}^{+\mathrm{j} \frac{2\pi}{N} \mu k})$' % N)

for tmp in range(N):
ax[tmp].set_facecolor('lavender')
ax[tmp].plot(W[:, tmp].real, k, 'C0o-', ms=7, lw=0.5)
ax[tmp].plot(W[:, tmp].imag, k, 'C1o-.', ms=7, lw=0.5)
ax[tmp].set_ylim(N-1, 0)
ax[tmp].set_xlim(-5/4, +5/4)
if tmp == 0:
ax[tmp].set_yticks(np.arange(0, N))
ax[tmp].set_xticks(np.arange(-1, 1+1, 1))
ax[tmp].set_ylabel(r'$\longleftarrow k$')
else:
ax[tmp].set_yticks([], minor=False)
ax[tmp].set_xticks([], minor=False)
ax[tmp].set_title(r'$\mu=$%d' % tmp)
fig.tight_layout()
fig.subplots_adjust(top=0.91)
fig.savefig('fourier_matrix.png', dpi=300)
print("Figure: fourrier_matrix is generated.")

X_test = mu
print("Processing matrix multiplication...")
x_test = 1/N * np.matmul(W, X_test)
print("testing... (test 1)")
if not np.allclose(ifft(X_test), x_test):
raise ValueError("Something went wrong... (test 1)")
print("test 1 ok")


print("applying linear combination of the Fourrier matrix...")
x_test2 = np.sum([X_test[i] * W[:, i] for i in range(N)], axis=0)
x_test2 *= 1/N
print("testing... (test 2)")

if not np.allclose(x_test, x_test2):
raise ValueError("Something went wrong... (test 2)")
print("test 2 ok")

plt.figure(figsize=(10,5), dpi=300)
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:')
```

```
# note that connecting the samples by lines is actually wrong, we
# use it anyway for more visual convenience

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)
plt.savefig("IDFT_result.png")
print("Figure: IDFT_result is generated.")

if __name__ == "__main__":
main()
```

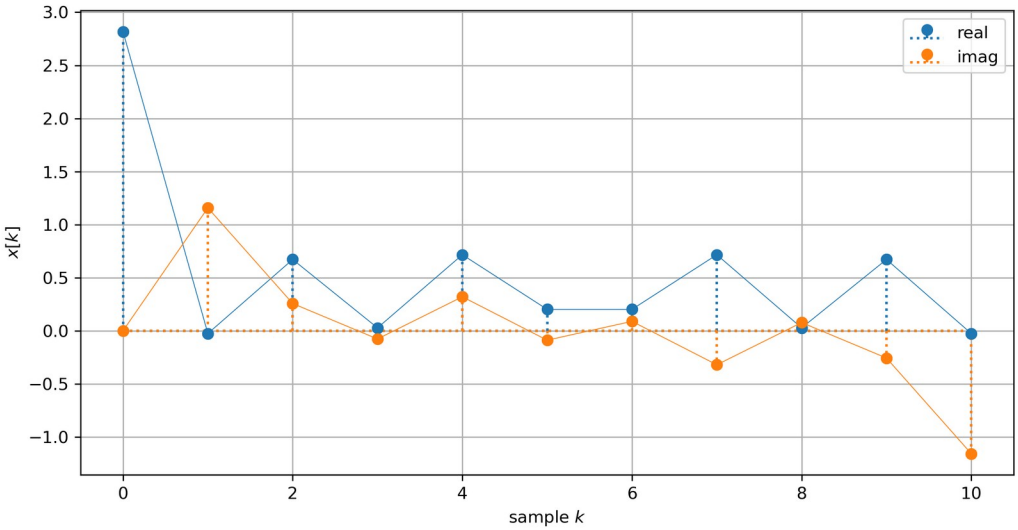
4. Outcomes:

Defining Xmu table ('f' for finish)
[]
>> 6
[6]
>> 4
[6, 4]
>> 4
[6, 4, 4]
>> 5
[6, 4, 4, 5]
>> 3
[6, 4, 4, 5, 3]
>> 4
[6, 4, 4, 5, 3, 4]
>> 5
[6, 4, 4, 5, 3, 4, 5]
>> 0
[6, 4, 4, 5, 3, 4, 5, 0]
>> 0
[6, 4, 4, 5, 3, 4, 5, 0, 0]
>> 0
[6, 4, 4, 5, 3, 4, 5, 0, 0, 0]
>> 0
[6, 4, 4, 5, 3, 4, 5, 0, 0, 0, 0]

```
>> f
getting all possible entries k
[[  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]]
Fourier matrix setting up
[[ 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]
 [ 1.        +0.j         -0.65486073+0.75574957j -0.14231484-0.98982144j
   0.84125353+0.54064082j -0.95949297+0.28173256j  0.41541501-0.909632j
   0.41541501+0.909632j   -0.95949297-0.28173256j  0.84125353-0.54064082j
```

-0.14231484+0.98982144j -0.65486073-0.75574957j]
[1. +0.j -0.95949297+0.28173256j 0.84125353-0.54064082j
-0.65486073+0.75574957j 0.41541501-0.909632j -0.14231484+0.98982144j
-0.14231484-0.98982144j 0.41541501+0.909632j -0.65486073-0.75574957j
0.84125353+0.54064082j -0.95949297-0.28173256j]
[1. +0.j -0.95949297-0.28173256j 0.84125353+0.54064082j
-0.65486073-0.75574957j 0.41541501+0.909632j -0.14231484-0.98982144j
-0.14231484+0.98982144j 0.41541501-0.909632j -0.65486073+0.75574957j
0.84125353-0.54064082j -0.95949297+0.28173256j]
[1. +0.j -0.65486073-0.75574957j -0.14231484+0.98982144j
0.84125353-0.54064082j -0.95949297-0.28173256j 0.41541501+0.909632j
0.41541501-0.909632j -0.95949297+0.28173256j 0.84125353+0.54064082j
-0.14231484-0.98982144j -0.65486073+0.75574957j]
[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]
[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 0.84125353-0.54064082j 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]]
Figure: fourrier_matrix is generated.
Processing matrix multiplication...
testing... (test 1)
test 1 ok
applying linear combination of the Fourrier matrix...
testing... (test 2)
test 2 ok
Figure: IDFT_result is generated.

This is how recreated signal looks like:



5. Conclusions: Based on the reasons provided, we assert that the discrete Fourier transform is a fully reversible process. The input of the created program consists of Fourier frequency indices, which are utilized to recreate the original signal.

6. Repository:
https://github.com/MichalHer/aadec_mgr/tree/main/ex1_spectral_analysis_of_deterministic_signals