Лабораторная работа 7

Упражнение 7.1

> Реализовать рекурсивный алгоритм ДПФ

Создадим тестовый сигнал и проверим на нем работу функции Быстрого Преобразования Фурье из библиотеки `numpy`

In [ ]:
```
import numpy as np

ys = [-0.5, 0.1, 0.7, -0.1]
hs = np.fft.fft(ys)
hs
```

Out[ ]:
```
array([ 0.2+0.j , -1.2-0.2j,  0.2+0.j , -1.2+0.2j])
```

Начнем с реализации нерекурсивного алгоритма, в котором вычислим БПФ по отдельности для четных и нечетных элементов. Для этого воспользуемся леммой Дэниелсона-Ланцоша:

$$DFT(y)[n] = DFT(e)[n] + exp(-2\pi in/N)DFT(o)[n]$$

In [ ]:
```
def fft_no_recursion(ys):
    N = len(ys)
    even_dft = np.fft.fft(ys[::2])
    odd_dft = np.fft.fft(ys[1::2])

    ns = np.arange(N)
    return np.tile(even_dft, 2) + np.exp(-2 * np.pi* 1j * ns / N) * np.tile(odd_dft, 2)

hs1 = fft_no_recursion(ys)
hs1
```

Out[ ]:
```
array([ 0.2+0.j , -1.2-0.2j,  0.2+0.j , -1.2+0.2j])
```

In [ ]:
```
np.abs(hs - hs1)
```

Out[ ]:
```
array([0., 0., 0., 0.])
```

Результат совпал

In [ ]:
```
def recursion_fft(ys):
    N = len(ys)
    if N == 1:
        return ys

    even_dft = recursion_fft(ys[::2])
    odd_dft = recursion_fft(ys[1::2])

    ns = np.arange(N)

    return np.tile(even_dft, 2) + np.exp(-1j * 2 * np.pi * ns / N) * np.tile(odd_dft, 2)
```

In [ ]:
```

```