Министерство образования Республики Беларусь

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОИЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 3
на тему
«Преобразование Уолша»

Выполнили: Басько А. С.
Какадей С. В.

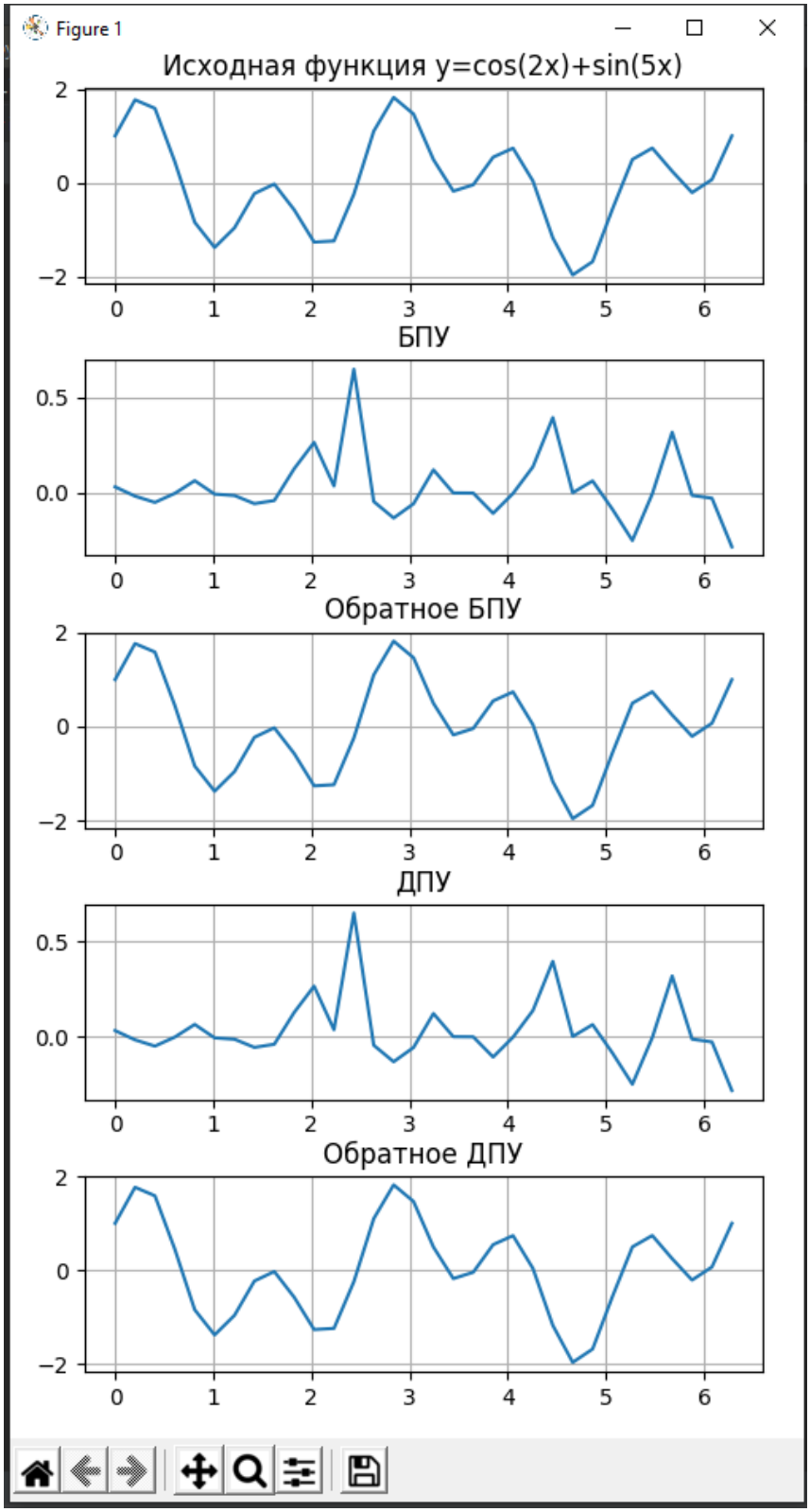
Проверил: Третьяков А. Г.

МИНСК 2021

Задание:
Изучение преобразования Уолша и его основных свойств, а также методики получения быстрого преобразования Уолша (БПУ).

Исходные данные:
1. Номер варианта: 3.
2. Функция сигнала: y=cos(2x) + sin(5x).
3. N: 32.

Результат работы программы:



Листинг кода:

(main.py)

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

from transform import get_wal_funcs, get_matrix, fut, ifut, dut, idut

N = 32


def _fun(x):
```

```
    y = np.cos(2*x) + np.sin(5*x)
    return y


def transformations():
    wal = get_wal_funcs()
    print('Функции Уолша, упорядоченные по частоте')
    for w in wal:

        s = ''
        for i in range(len(w)):
            if w[i] == 1:
                s += '+'
            else:
                s += '-'
        print(s)

    print('Функции Уолша, упорядоченные по Адамару')
    ad_matr = get_matrix(5)
    for m in ad_matr:
        s = ''
        for c in m:
            if c == 1:
                s += '+'
            else:
                s += '-'
        print(s)

    x = np.linspace(0.0, 2 * np.pi, 32)
    function_values = list(map(lambda i: _fun(i), x))

    transform = fut(function_values)
    reverse_transform = ifut(transform)

    first, func = plt.subplots(5, 1, figsize=(5, 8))
    plt.tight_layout()

    func[0].plot(x, function_values)
    func[0].set_title('Исходная функция y=cos(2x)+sin(5x)')
    func[0].grid(True)

    func[1].plot(x, transform)
    func[1].set_title('БПУ')
    func[1].grid(True)

    func[2].plot(x, reverse_transform)
    func[2].set_title('Обратное БПУ')
    func[2].grid(True)

    function_values_dut = list(map(lambda i: _fun(i), x))
    transform_dut = dut(function_values_dut)
    reverse_transform_idut = idut(transform_dut)

    func[3].plot(x, transform_dut)
    func[3].set_title('ДПУ')
    func[3].grid(True)

    func[4].plot(x, reverse_transform_idut)
    func[4].set_title('Обратное ДПУ')
    func[4].grid(True)

    plt.show()


if __name__ == '__main__':
    transformations()
```

```
 (transform.py) import numpy as nm

functions = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1],
             [1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1,
1, 1, -1, -1, -1, -1, -1, -1, -1, -1],
             [1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1,
-1, -1, 1, 1, 1, 1, -1, -1, -1, -1],
             [1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1,
-1, -1, 1, 1, -1, -1, 1, 1, -1, -1],
             [1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -
1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1]]


def int_to_bin(n):
    s = str(bin(n))
    b = [0] * 5
    i = 2
    for c in s[::-1]:
        if c == 'b':
            break
        elif int(c) == 1:
            b[i] = 1
        i -= 1

    return b


def get_wal_funcs():
    wal = [None] * 32
    for i in range(32):
        n = int_to_bin(i)
        r_c = [n[4] ^ n[3], n[3] ^ n[2], n[2] ^ n[1], n[1] ^ n[0], n[0] ^ 0]

        wal[i] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1]

        for j in range(5):
            if r_c[j] != 0:
                for k in range(32):
                    wal[i][k] *= functions[j][k]

    return wal


def get_matrix(coef):

    def create_current_matrix(matrix):
        l = len(matrix)
        result = nm.empty([l * 2, l * 2], dtype=int)

        for r in range(l):
            for c in range(l):
                result[r][c] = result[r + l][c] = result[r][c + l] = matrix[r][c]
                result[r + l][c + l] = - matrix[r][c]

        return result

    if coef == 0:
        return [[1]]
    else:
        prev_matr = get_matrix(coef - 1)
        return create_current_matrix(prev_matr)
```

```
def fut(signal):
    def fun(a):
        l = len(a)
        if l == 1:
            return a
        else:
            up = fun(a[0:int(l / 2)])
            down = fun(a[int(l / 2):l])
            return list(map(lambda x, y: x + y, up, down)) + list(map(lambda x, y: x - y,
up, down))

    return list(map(lambda x: x / 32, fun(signal)))


def ifut(signal):
    def fun(a):
        l = len(a)
        if l == 1:
            return a
        else:
            up = fun(a[0:int(l / 2)])
            down = fun(a[int(l / 2):l])
            return list(map(lambda x, y: x + y, up, down)) + list(map(lambda x, y: x - y,
up, down))

    return fun(signal)


def dut(signal):
    return list(map(lambda x: x / 32,
get_matrix(5).__matmul__(nm.array(signal).transpose())))


def idut(signal):
    return get_matrix(5).transpose().__matmul__(nm.array(signal).transpose())
```