Санкт-Петербургский Государственный Политехнический Университет
Институт Компьютерных Наук и Технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №7
на тему
Дискретное преобразование фурье

Работу выполнил
Студент группы 3530901/80203
Тарасенко Н.С.
Преподаватель
Богач Н.В.

Санкт-Петербург, 2021 год

1 Настройка проекта

Перед тем как выполнять задания необходимо настроить проект и сделать все необходимые импорты:

```
from __future__ import print_function, division

import thinkdsp
import thinkplot

import numpy as np

import warnings
warnings.filterwarnings('ignore')

PI2 = 2 * np.pi

np.set_printoptions(precision=3, suppress=True)
%matplotlib inline
```

Рис. 1: 2

2 Ход работы

Возьмем небольшой сигнал и вычислим его DFT:

```
ys = [-0.2, 0.2, 0.8, -0.2]
hs = np.fft.fft(ys)
print(hs)
```

```
[ 0.6+0.j  -1. -0.4j  0.6+0.j  -1. +0.4j]
```

Рис. 2: 2

Реализация DFT из книги:

```
def dft(ys):
    N = len(ys)
    ts = np.arange(N) / N
    freqs = np.arange(N)
    args = np.outer(ts, freqs)
    M = np.exp(1j * PI2 * args)
    amps = M.conj().transpose().dot(ys)
    return amps
```

Рис. 3: 2

Проведем тест, для того, чтобы убедиться что результаты совпадают с использованием np.fft.fft для вычисления DFT

```
: hs2 = dft(ys)
  print(sum(abs(hs - hs2)))
```

```
6.890185696795333e-16
```

Рис. 4: 2

Для начала проектирования рекурсивного DFT я реализую метод, который разбивает входной массив и использует np.fft.fft для вычисления DFT половин.

```
: def fft_norec(ys):
      N = len(ys)
      He = np.fft.fft(ys[::2])
      Ho = np.fft.fft(ys[1::2])

      ns = np.arange(N)
      W = np.exp(-1j * PI2 * ns / N)

      return np.tile(He, 2) + W * np.tile(Ho, 2)
```

Рис. 5: 2

Получаем такие результаты:

```
: hs3 = fft_norec(ys)
  print(sum(abs(hs - hs3)))
```

```
0.0
```

Рис. 6: 2

Наконец, мы можем заменить np.fft.fft рекурсивными вызовами и добавить базовый вариант:

```
:
def fft(ys):
    N = len(ys)
    if N == 1:
        return ys

    He = fft(ys[::2])
    Ho = fft(ys[1::2])

    ns = np.arange(N)
    W = np.exp(-1j * PI2 * ns / N)

    return np.tile(He, 2) + W * np.tile(Ho, 2)
```

```
:
hs4 = fft(ys)
print(sum(abs(hs - hs4)))
```

```
3.5927571778724297e-16
```

Рис. 7: 2

Эта реализация DFT требует времени, пропорционального «log». Это также занимает пространство, пропорциональное "log и тратит время на создание и копирование массивов. Его можно улучшить, чтобы он работал «на месте»; в этом случае он не требует дополнительного места и тратит меньше времени на накладные расходы.