Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №7
Дисциплина: Телекоммуникационные технологии
Тема: Дискретное преобразование Фурье

Работу выполнил:
Ляшенко В.В.
Группа: 3530901/80201
Преподаватель:
Богач Н.В.

Санкт-Петербург
2021

Оглавление

1 Упражнение 7.1 **4**

2 Упражнение 7.2 **5**

2.1 Алгоритм 5

2.2 Реализация 5

3 Выводы **8**

Список иллюстраций

Листинги

2.1 Вычисление БПФ с помощью np.fft.fft 5
2.2 Полученные результаты 5
2.3 Функция dft 5
2.4 Применение функции ДПФ 6
2.5 Полученные результаты 6
2.6 Функция fft_norec 6
2.7 Применение функции fft_norec 6
2.8 Полученные результаты 6
2.9 Функция fft 6
2.10 Применение функции fft 7
2.11 Полученные результаты 7

Глава 1

Упражнение 7.1

В начале мы должны для Jupyter загрузить `chap07.ipynb`, прочитать пояснения и запустить примеры.

Все примеры были успешно запущены.

Глава 2

Упражнение 7.2

2.1 Алгоритм

Реализуем алгоритм быстрого преобразования Фурье (БПФ), время работы которго $N \log N$. Для этого воспользуемся леммой Дэниелсона-Ланцоша.

$$DFT(y)[n] = DFT(e)[n] + exp(-2nin/N)DFT(o)[n] \tag{2.1}$$

В этой формуле DFT(y)[n] - это n-й элемент ДПФ от y, e и o - массивы сигнала, содержащие соответственно четные и нечетные элементы y.

Эта лемма предлагает рекурсивный алгоритм для ДПФ:

1. Дан массив сигнала y. Разделим его на чётные элементы e и нечётные элементы o.
2. Вычислим DFT e и o, делая рекурсивные вызовы.
3. Вычислим DFT(y) для каждого значения n, используя лемму Дэниелсона-Ланцоша.

В простейшем случае эту рекурсию надо продолжать, пока длина y не дойдет до 1. Тогда DFT(y) = y. А если длина y достаточно мала, можно вычислить его ДПФ перемножением матриц, используя заранее вычисленные матрицы.

2.2 Реализация

Возьмём небольшой сигнал и вычислим его БПФ с помощью имеющейся функции `np.fft.fft`.

```
ys = [-0.3, 0.8, 0.6, -0.4]
hs = np.fft.fft(ys)
print(hs)
```

Листинг 2.1: Вычисление БПФ с помощью np.fft.fft

```
[ 0.7+0.j -0.9-1.2j -0.1+0.j -0.9+1.2j]
```

Листинг 2.2: Полученные результаты

Теперь реализуем функцию ДПФ.

```
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
```

Листинг 2.3: Функция dft

Воспользуемся ей, чтобы убедиться, что результаты одинаковые.

```
hs2 = dft(ys)
np.sum(np.abs(hs - hs2))
```

Листинг 2.4: Применение функции ДПФ

```
1.0537365376317067e-15
```

Листинг 2.5: Полученные результаты

Как мы можем видеть, различия минимальны.

Для того, чтобы создать рекурсивное БПФ, напишем функцию, которая разбивает входной массив и использует np.fft.fft для вычисления БПФ полученных 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)
```

Листинг 2.6: Функция fft_norec

Применим эту функцию и убедимся, что результат тот же.

```
hs3 = fft_norec(ys)
np.sum(np.abs(hs - hs3))
```

Листинг 2.7: Применение функции fft_norec

```
3.820527793534411e-16
```

Листинг 2.8: Полученные результаты

Разница также мала.

Теперь реализуем функцию `fft`, где заменим np.fft.fft на рекурсивные вызовы.

```
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)
```

Листинг 2.9: Функция fft

Проверим её работу.

```
hs4 = fft(ys)
np.sum(np.abs(hs - hs4))
```

Листинг 2.10: Применение функции fft

```
3.820527793534411e-16
```

Листинг 2.11: Полученные результаты

Всё работает верно.

Полученная реализация работает за $N \log N$, однако имеют недостатки - занимаемое пространство так же составляет $N \log N$, к тому же тратится время на создание и копирование массивов.

Функцию можно улучшить, реализовав работу «на месте».

Глава 3

Выводы

В результате выполнения данной работы мы изучили дискретное преобразование Фурье и быстрое преобразование Фурье. Во многих случаях достаточно ДПФ, которое работает за N^2, но иногда требуется БПФ, которое работает за $N \log N$.