Лабораторная работа №7
Дискретное преобразование Фурье

Кобыжев Александр

11 апреля 2021 г.

Оглавление

1 Упражнение 7.1 **3**

2 Упражнение 7.2 **4**

3 Выводы **7**

Листинги

2.1 Вычисление БПФ при помощи `np.fft.fft` 4
2.2 Функция `dft` 4
2.3 Сравнение реализаций 4
2.4 Функция `fft_norec` 4
2.5 Сравнение реализаций 5
2.6 Функция `fft` 5
2.7 Сравнение реализаций 5

Глава 1

Упражнение 7.1

В данном упражнении нас просят открыть `chap07.ipynb`, прочитать пояснения, а также запустить примеры.

Глава 2

Упражнение 7.2

Возьмём небольшой реальный сигнал и вычислим его БПФ:

```
ys = [-0.4, 0.2, 0.8, -0.2]
hs = np.fft.fft(ys)
print(hs)
```

Листинг 2.1: Вычисление БПФ при помощи `np.fft.fft`

Напишем функцию для БПФ:

```
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 2.2: Функция `dft`

Удостоверимся, что эта реализация даёт тот же результат, что и `np.fft.fft`.

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

Листинг 2.3: Сравнение реализаций

Различие в результатах минимально и равняется `7.507415150515407e-16`.

Чтобы сделать рекурсивное БПФ, я начну с версии, которая разбивает входной массив и использует `np.fft.fft` для вычисления БПФ 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 2.4: Функция `fft_norec`

Теперь мы также должны получить такие же результаты.

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

Листинг 2.5: Сравнение реализаций

Разница равняется `0.0`.

Теперь мы можем заменить `np.fft.fft` рекурсивными вызовами и добавить базовый случай:

```
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 2.6: Функция `fft`

Результаты снова совпадают:

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

Листинг 2.7: Сравнение реализаций

Разница между результатами составила `2.220446049250313e-16`.

Эта реализация БПФ требует времени и пространства, пропорционального $nlogn$ и это тратит время на создание и копирование массивов. Его можно улучшить, чтобы он работал «на месте», но в этом случае он

не требует дополнительного места и тратит меньше времени на накладные расходы.

Глава 3

Выводы

Во время выполнения лабораторной работы получены навыки работы с комплексными экспонентами, а также с дискретным преобразованием Фурье.