

Лабораторная работа №7
Дискретное преобразование Фурье

Смирнов Никита

26 апреля 2021 г.

Оглавление

1	Упражнение 7.1	3
2	Упражнение 7.2	4
3	Выводы	7

Листинги

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

Глава 1

Упражнение 7.1

В данном упражнении нас просят открыть `chap07.ipynb`, прочитать пояснения и запустить примеры. Поэтому я просто изучил все примеры и комментарии к ним.

Глава 2

Упражнение 7.2

В данном упражнении предлагается исследовать и реализовать быстрое преобразование Фурье. Реализация заключается в делении массива значений некоторого сигнала на подмассивы с четными и не четными элементами, вычислении ДПФ для них.

Возьмём небольшой реальный сигнал и вычислим его БПФ:

```
1 ys = [-0.1, 0.3, 0.6, -0.2]
2 hs = np.fft.fft(ys)
3 print(hs)
```

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

Результаты применения к нему быстрого преобразования: `[0.6+0.j -0.7-0.5j 0.4+0.j -0.7+0.5j]`.

Напишем функцию для БПФ:

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

Листинг 2.2: Функция `dft`

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

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

Листинг 2.3: Сравнение реализаций

Все верно и результаты сходятся $6.169420266625131e-16$.

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

```
1 def fft_norec(ys):
2     N = len(ys)
3     He = np.fft.fft(ys[:2])
4     Ho = np.fft.fft(ys[1:2])
5
6     ns = np.arange(N)
7     W = np.exp(-1j * PI2 * ns / N)
8
9     return np.tile(He, 2) + W * np.tile(Ho, 2)
```

Листинг 2.4: Функция `fft_norec`

Проверяем результат.

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

Листинг 2.5: Сравнение реализаций

И получим ошибку равную $1.2326877045398918e-16$

Теперь заменим библиотечную функцию `np.fft.fft` на рекурсивными вызовами и добавим базовый случай:

```
1 def fft(ys):
2     N = len(ys)
3     if N == 1:
4         return ys
5
6     He = fft(ys[:2])
7     Ho = fft(ys[1:2])
8
9     ns = np.arange(N)
10    W = np.exp(-1j * PI2 * ns / N)
11
12    return np.tile(He, 2) + W * np.tile(Ho, 2)
```

Листинг 2.6: Функция `fft`

Результаты снова совпадают:

```
1 hs4 = fft(ys)
```

```
2 print(sum(abs(hs - hs4)))
```

Листинг 2.7: Сравнение реализаций

Разница между результатами $2.4739547811635286\text{e-}16$. Эта реализация БПФ требует времени и пространства, пропорционального $n \log n$ 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.