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

Телекоммуникационные технологии

Отчёт по лабораторной работе №7

Работу выполнил:
Смирнов Л. Д.
Группа:
3530901/80202
Преподаватель:
Богач Н. В.

Санкт-Петербург
2021

Содержание

1. Выполнение работы

1.1. Упражнение 1

В данном упражнении предлагается изучить и запустить примеры, приведенные в файле chap07, что я собственно и сделал

1.2. Упражнение 2

В данном упражнении предлагается исследовать и реализовтать быстрое преобразование Фурье, в основе которого лежат дискретное преобразование Фруье и лемма Дэниэлсона-Ланцоша, котороя звучит так:

$$DFT(y)[n] = DFT(e)[n] + e^{-\frac{2\pi in}{N}} \cdot DFT(o)[n]$$

Реализация зключается в делении массива значений некторого сигнала на подмассивы с четными и нечетными элементами, вычислении ДПФ для них и применении к каждому значению приведенной выш леммы. Завершается алгоритм когда деление у дойдет до массива в 1 элемент. Для начала реализую нерекурсивный вариант алгоритма на основе следующего сигнала:

```
ys = [-0.3, 0.7, 0.2, -0.4]
hs = np.fft.fft(ys)
print(hs)
```

Результаты применения к нему быстрого преобразования "Фурье из коробки": [0.2+0.j -0.5-1.1j -0.4+0.j -0.5+1.1j]. Они мне понадобятся для сравнения с тем, что даст на выходе реализованный алгоритм.

```
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
```

Как результат имеем разницу между двумя алгоритмами, равную 6.677606079408769e-16. Следующим шагом идет реализация алгоритма, разбивающего массив на половины, и применяющего к каждой из них fft из коробки:

```
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)
```

Применим его:

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

И получим на выходе ошибку, равную 3.142951601307097e-16, то есть в 2 раза меньше предыдущей. Финальным шагом реализации алгоритма идет замена библиотечной функции fft на рекурсивные вызовы, которые в конце приведут нас к рассмотрению базы, то есть последнего оставшегося элемента y.

```
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)
```

И получаем как итог массив [0.2+0.j , -0.5-1.1j, -0.4-0.j , -0.5+1.1j], который, как можно посмотреть в исходных данных, совпадает с заданным, что меня вполне устраивает.

2. Выводы

В данной лабораторной работе я познакомился с понятиями комплексных экспонент и комплексных сигналов, а так же изучил Дискретное преобразование Фурье. Кроме того, упражнения позволили реализовать рекурсивный алгортим с его использованием.