Пономарев Александр ПМ-31
Вариант 19 mod 4 + 1 = 4
Лабораторная работа 2

Для компиляции и запуска в папке scripts:

$ cmake -B build .
$ make -C build
$ ./build/dft

Весь python код находится в файле dsp.ipynb

Задание 1.

Функции:
TSignal dft(const TSignal& x, bool invert = false);
TSignal idft(const TSignal& x);

Код реализации находится в файле: dsp.cpp
Сигнал находится в файле: ./txt/signal1.txt
Резулультат находится в: ./txt/spectrum_slow.txt

Задание 2.

Функции:
TSignal fft(const TSignal& x, bool invert = false);
TSignal ifft(const TSignal& x);

Код реализации находится в файле: dsp.cpp
Сигнал находится в файле: ./txt/signal1.txt
Резулультат находится в: ./txt/spectrum_fast.txt

Задание 3.
Будем считать среднеквадратичную ошибку для сигнала и восстановленного сигнала, а так же между спектрами.

mse of signal and slow recovered signal: (2.61014e-22,-6.93024e-22)
mse of signal and fast recovered signal: (8.73046e-24,-3.93568e-23)
mse of slow spectrum and fast spectrum: (2.13117e-21,2.78243e-21)

Среднеквадратичная ошибка во всех 3 случаях близка к 0, а значит:
x = одпф(дпф(x))
x = обпф(бпф(x))
дпф(x) = бпф(x)

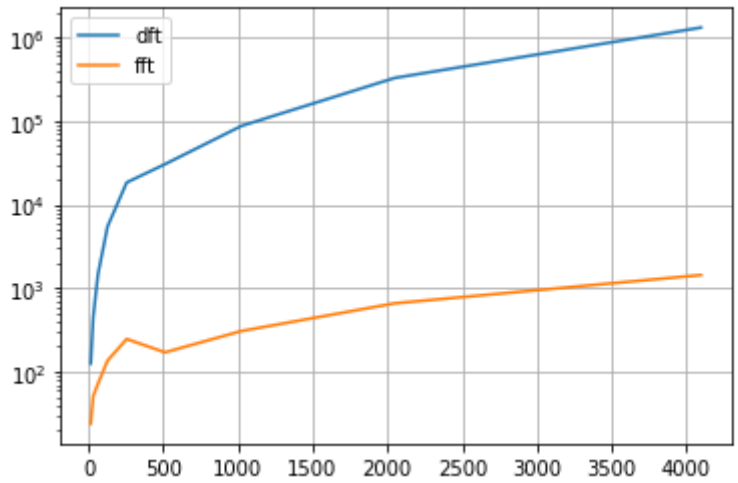
Сравним со значениями полученными в python
$ python3 dsp.py

mse of cpp_spectrum_fast and spectrum: (-3.3074935381067407e-18+1.934576929414519e-18j)
mse of cpp_spectrum_slow and spectrum: (-3.3074935381067407e-18+1.934576929414519e-18j)

Среднеквадратичная ошибка очень мала, значит алгоритм находит спектр правильно.
Задание 4.

```
dft - i: 16 time: 126 microseconds
fft - i: 16 time: 24 microseconds
dft - i: 32 time: 447 microseconds
fft - i: 32 time: 51 microseconds
dft - i: 64 time: 1462 microseconds
fft - i: 64 time: 72 microseconds
dft - i: 128 time: 5466 microseconds
fft - i: 128 time: 136 microseconds
dft - i: 256 time: 18391 microseconds
fft - i: 256 time: 247 microseconds
dft - i: 512 time: 30430 microseconds
fft - i: 512 time: 171 microseconds
dft - i: 1024 time: 87401 microseconds
fft - i: 1024 time: 308 microseconds
dft - i: 2048 time: 327235 microseconds
fft - i: 2048 time: 660 microseconds
dft - i: 4096 time: 1310339 microseconds
fft - i: 4096 time: 1438 microseconds
```

Построим график по этим запускам:



Можно заметить, что dft имеет приблизительно квадратичную сложность, а fft имеет сложность n*log(n)

Задание 5.

Функция:
TSignal convolution_slow(const TSignal& x, const TSignal& y);

Код реализации находится в файле: dsp.cpp
Сигналы для свертки находятся в файлах: ./txt/signal1.txt и ./txt/signal2.txt
Резулультат находится в: ./txt/convolution_slow.txt

Задание 6.

Функция:
TSignal convolution_fast(const TSignal& x, const TSignal& y);

Код реализации находится в файле: dsp.cpp
Сигналы для свертки находятся в файлах: ./txt/signal1.txt и ./txt/signal2.txt
Резулультат находится в: ./txt/convolution_fast.txt

Задание 7.

Посчитаем среднеквадратичное отклонение для 2 наших реализаций:
mse of conv_slow and conv_fast: (2.5566e-10,-6.39682e-11)

Посчитаем среднеквадратичную ошибку для python
mse of cpp_conv_fast and conv: (3.299698489698322e-10-5.979241454052225e-11j)
mse of cpp_conv_slow and conv: (1.4948043173128423e-11+5.181460479686944e-15j)

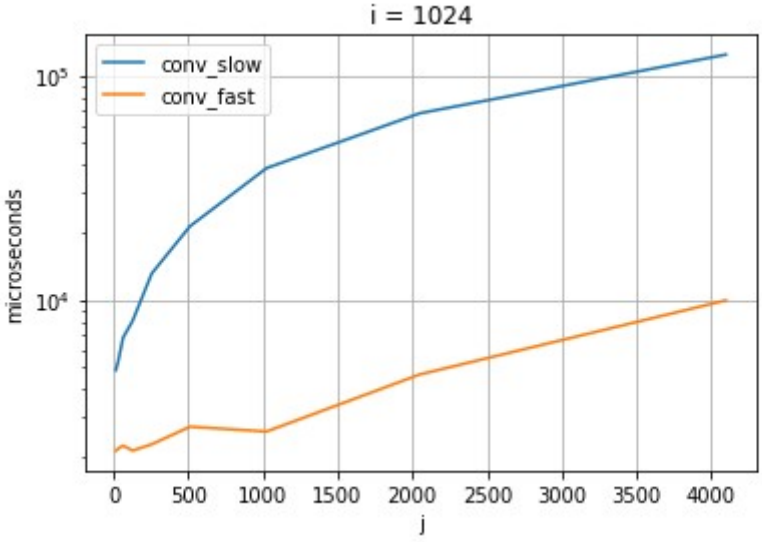
Среднеквадратичная ошибка очень мала, значит алгоритм находит спектр правильно.

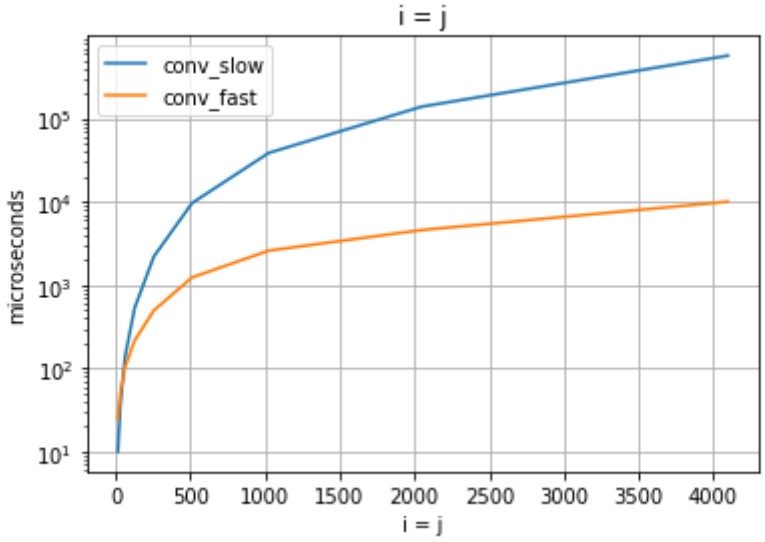
Задание 8.

Проведем серию запусков:

```
convolution_slow - (i, j): (1024, 16) time: 4875 microseconds
convolution_fast - (i, j): (1024, 16) time: 2135 microseconds
convolution_slow - (i, j): (16, 16) time: 10 microseconds
convolution_fast - (i, j): (16, 16) time: 25 microseconds
convolution_slow - (i, j): (1024, 32) time: 5360 microseconds
convolution_fast - (i, j): (1024, 32) time: 2178 microseconds
convolution_slow - (i, j): (32, 32) time: 36 microseconds
convolution_fast - (i, j): (32, 32) time: 50 microseconds
convolution_slow - (i, j): (1024, 64) time: 6795 microseconds
convolution_fast - (i, j): (1024, 64) time: 2246 microseconds
convolution_slow - (i, j): (64, 64) time: 140 microseconds
convolution_fast - (i, j): (64, 64) time: 105 microseconds
convolution_slow - (i, j): (1024, 128) time: 8082 microseconds
convolution_fast - (i, j): (1024, 128) time: 2134 microseconds
convolution_slow - (i, j): (128, 128) time: 535 microseconds
convolution_fast - (i, j): (128, 128) time: 217 microseconds
convolution_slow - (i, j): (1024, 256) time: 13134 microseconds
convolution_fast - (i, j): (1024, 256) time: 2278 microseconds
convolution_slow - (i, j): (256, 256) time: 2205 microseconds
convolution_fast - (i, j): (256, 256) time: 494 microseconds
convolution_slow - (i, j): (1024, 512) time: 21325 microseconds
convolution_fast - (i, j): (1024, 512) time: 2727 microseconds
convolution_slow - (i, j): (512, 512) time: 9683 microseconds
convolution_fast - (i, j): (512, 512) time: 1239 microseconds
convolution_slow - (i, j): (1024, 1024) time: 38692 microseconds
convolution_fast - (i, j): (1024, 1024) time: 2596 microseconds
convolution_slow - (i, j): (1024, 1024) time: 37056 microseconds
convolution_fast - (i, j): (1024, 1024) time: 2415 microseconds
convolution_slow - (i, j): (1024, 2048) time: 67901 microseconds
convolution_fast - (i, j): (1024, 2048) time: 4659 microseconds
convolution_slow - (i, j): (2048, 2048) time: 139445 microseconds
convolution_fast - (i, j): (2048, 2048) time: 4594 microseconds
convolution_slow - (i, j): (1024, 4096) time: 123923 microseconds
convolution_fast - (i, j): (1024, 4096) time: 9967 microseconds
convolution_slow - (i, j): (4096, 4096) time: 570937 microseconds
convolution_fast - (i, j): (4096, 4096) time: 10065 microseconds
```

Построим график по этим значениям:


i = 1024
conv_slow
conv_fast
microseconds
10^5
10^4
0
500
1000
1500
2000
2500
3000
3500
4000
j


i = j
conv_slow
conv_fast
microseconds
10^5
10^4
10^3
10^2
10^1
0
500
1000
1500
2000
2500
3000
3500
4000
i = j