Лабораторная работа № 2:

“Быстрое преобразование Фурье. Дискретная свёртка”

Выполнил:

студент группы МП-30,

Алимагадов Курбан Алимагадович

МИЭТ 2019 г.

Задание

1. Реализовать на C или C++ алгоритмы непосредственного вычисления ДПФ и ОДПФ по формулам (1) и (2) для комплексного входного сигнала с двойной точностью (double). Входные данные загружать из текстового файла (разделитель – пробел), сгенерированного, например, в MATLAB.

Алгоритм реализован в функции – `vec_compl ft(vec_compl vect_in, int sign)`, код на языке C++ в файле code.txt. Входные данные генерируются в m-файле lab2_task_1.m. При значении параметра sign = 1, функция осуществляет ДПФ, при значении параметра sign = -1, функция осуществляет ОДПФ.

2. Реализовать на C или C++ алгоритмы прямого и обратного БПФ для комплексного входного сигнала длиной 2^n, n – любое натуральное число:

б) с прореживанием по времени без двоично-инверсных перестановок (вариант 2);

Алгоритм реализован в функции – `vec_compl fft(vec_compl vect_in, int sign)`, код на языке C++ в файле code.txt. Входные данные генерируются в m-файле lab2_task_1.m. При значении параметра sign = 1, функция осуществляет ДПФ, при значении параметра sign = -1, функция осуществляет ОДПФ.

3. Убедиться в корректности работы алгоритмов:

а) проверить выполнение равенства X = ОДПФ (ДПФ (X)), а также равенства X = ОБПФ (БПФ (X));

б) сравнить результаты ДПФ(**X**) и БПФ(**X**);

в) сравнить результаты работы реализованного алгоритма, например, с результатами процедуры fft, встроенной в MATLAB.

Проверка на равенство осуществляется в функции `void task3_eq()` с помощью функции `void equal(string x1s, string x2s, string result)`, входные данные, для которых вычисляются преобразования, генерируются в m-файле lab2_task_1.m. ДПФ, ОДПФ и БПФ, ОБПФ вычисляются с помощью

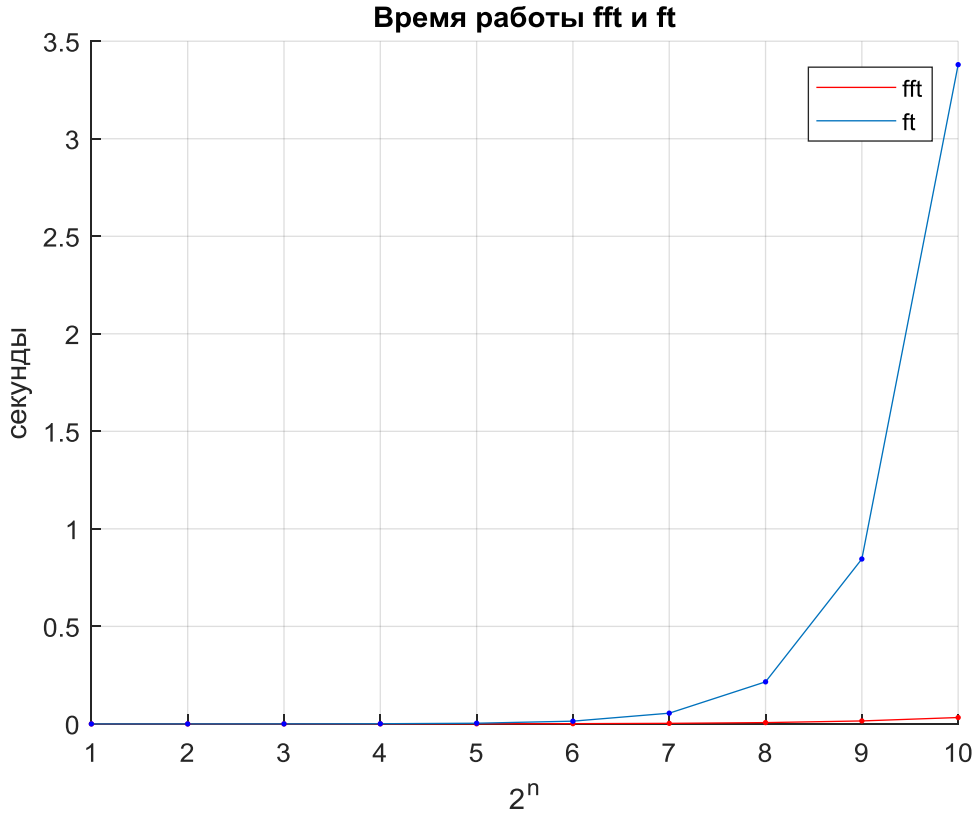
функций `void task3ft()` и `void task3fft()` соответственно. Код на языке C++ в файле code.txt.

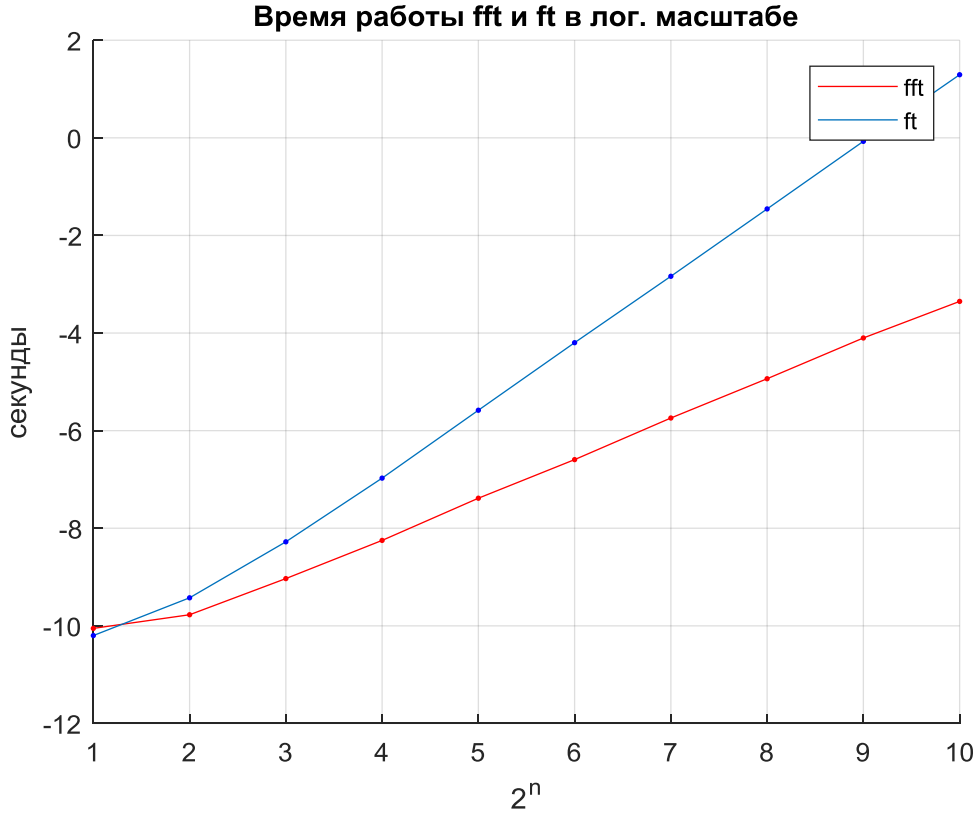
Векторы значений, получившиеся в результате выполнения функций ft (ДПФ) и fft (БПФ), равны. Также равны векторы обратных преобразований. В результате сравнения функции fft с функцией fft среды MATLAB (код с БПФ на языке MATLAB в файле lab2_task_1.m) получилось, что элементы векторов БПФ отличаются не более чем на 1.02956e-05.

4. Проанализировать зависимость времени выполнения БПФ и непосредственного вычисления ДПФ от длины $N = 2^n$ преобразования. Отобразить результаты в виде графика зависимости времени T выполнения преобразования от размерности: T = T(n).

Чтобы проанализировать зависимость времени выполнения БПФ и непосредственного вычисления ДПФ от длины входного вектора, используется функция `void task4()`, которая создаёт 2 txt-файла, с наборами значений времени (в секундах) соответствующим ДПФ (БПФ) входных векторов длины 2^n, где $n = 1,\ldots,10$. Входные данные генерируются в файле lab2_task_2.m, код, строящий графики по результатам вычисленным функцией task4(), там же.

Графики построенные по полученным результатам:





Так как сложность алгоритма ДПФ порядка $O(N^2)$ операций, а сложность алгоритма БПФ порядка $O(NlogN)$ операций, то, при увеличении длины вектора входных значений, время преобразования для ДПФ возрастает

быстрее чем время преобразования для БПФ, что можно наблюдать на графике.

5. Реализовать на C или C++ процедуру прямого вычисления свертки двух последовательностей по формуле (3). Входные данные загружать из текстового файла (разделитель – пробел), сгенерированного, например, в MATLAB.

Алгоритм реализован в функции – `vec_compl conv(vec_compl vect_in1, vec_compl vect_in2)`, код на языке C++ в файле code.txt. Входные данные генерируются в m-файле lab2_task_3.m.

6. Реализовать процедуру нахождения дискретной свертки, основанную на БПФ. При вычислении БПФ использовать результаты п. 2 задания.

Алгоритм реализован в функции – `vec_compl conv_fft(vec_compl vect_in1, vec_compl vect_in2)`, код на языке C++ в файле code.txt. Входные данные генерируются в m-файле lab2_task_3.m.

7. Убедится в корректности работы процедуры из п. 5 и п. 6 задания, сравнив полученные результаты с результатами работы встроенной функций MATLAB conv.

В результате сравнения функций conv и conv_fft с функцией conv среды MATLAB (код на языке MATLAB с вычислением свёртки в файле lab2_task_3.m) получилось, что элементы векторов свёртки отличаются не более чем на 3.77359e-05.

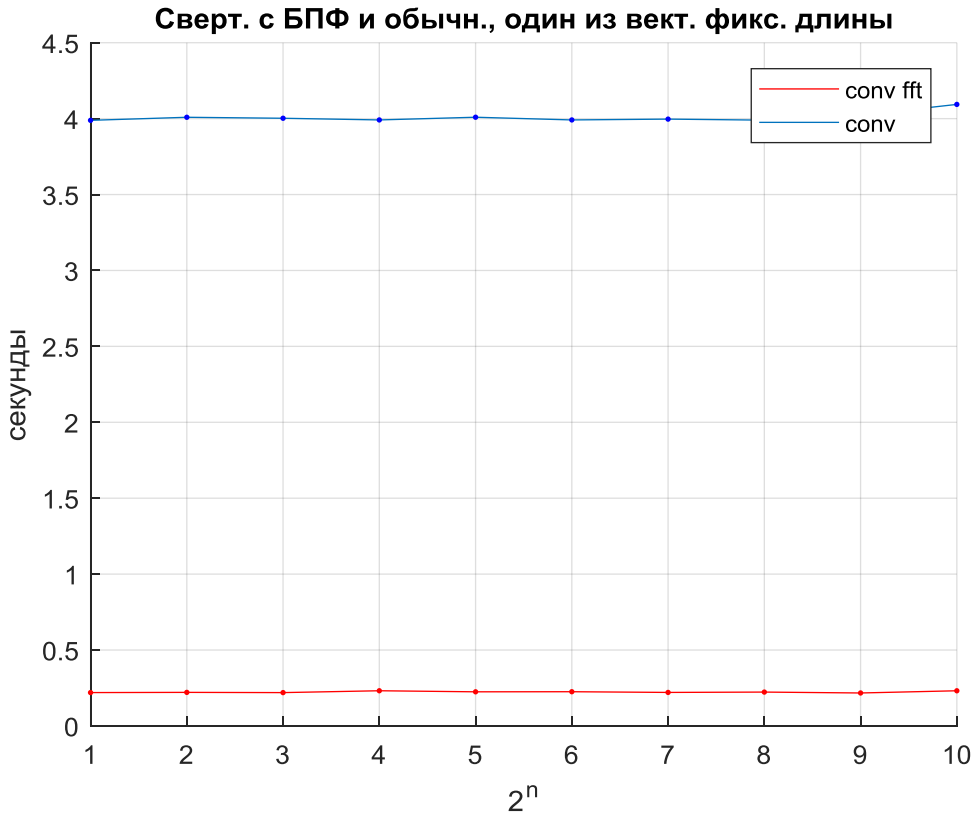
8. Сравнить производительность алгоритмов вычисления свертки по определению (3) и с помощью БПФ в двух случаях: когда размер одной из последовательностей фиксирован, и когда меняются длины обеих последовательностей.

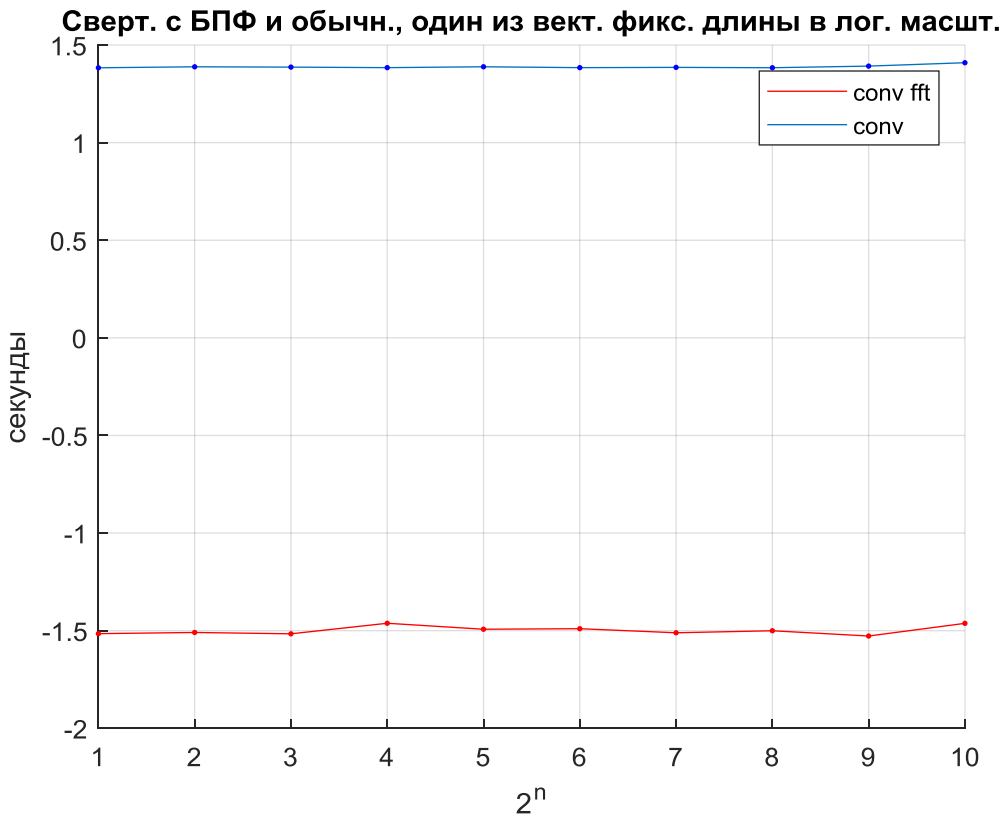
Чтобы сравнить производительность алгоритмов вычисления свертки, были рассмотрены: свёртка входных векторов одинаковой длины 2^n, где $n = 1,\ldots,10$; свёртка входных векторов разной длины, где один из векторов длины 2^n ($n = 1,\ldots,10$), а второй – фиксированной длины $N = 1024$; свёртка

входных векторов разной длины, где один из векторов длины 2^n (n = 1,…,5), а второй – длины 2^{2n} (n = 1,…,5).

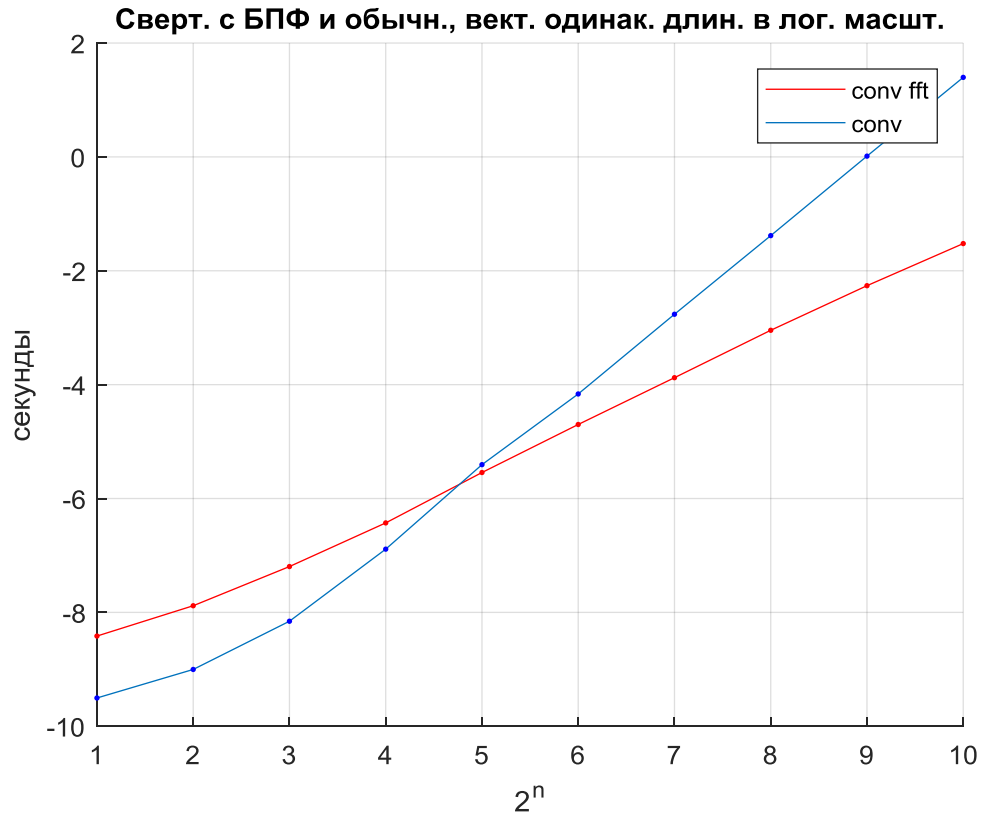
Чтобы проанализировать зависимость времени непосредственного вычисления свёртки и вычисления свёртки с помощью БПФ от длины входных векторов, используется функция `void task8()`, которая создаёт 2 txt-файла, с наборами значений времени (в секундах) соответствующим свёрткам векторов различной длины. Входные данные генерируются в файле lab2_task_4.m, код, строящий графики по полученным результатам, там же.

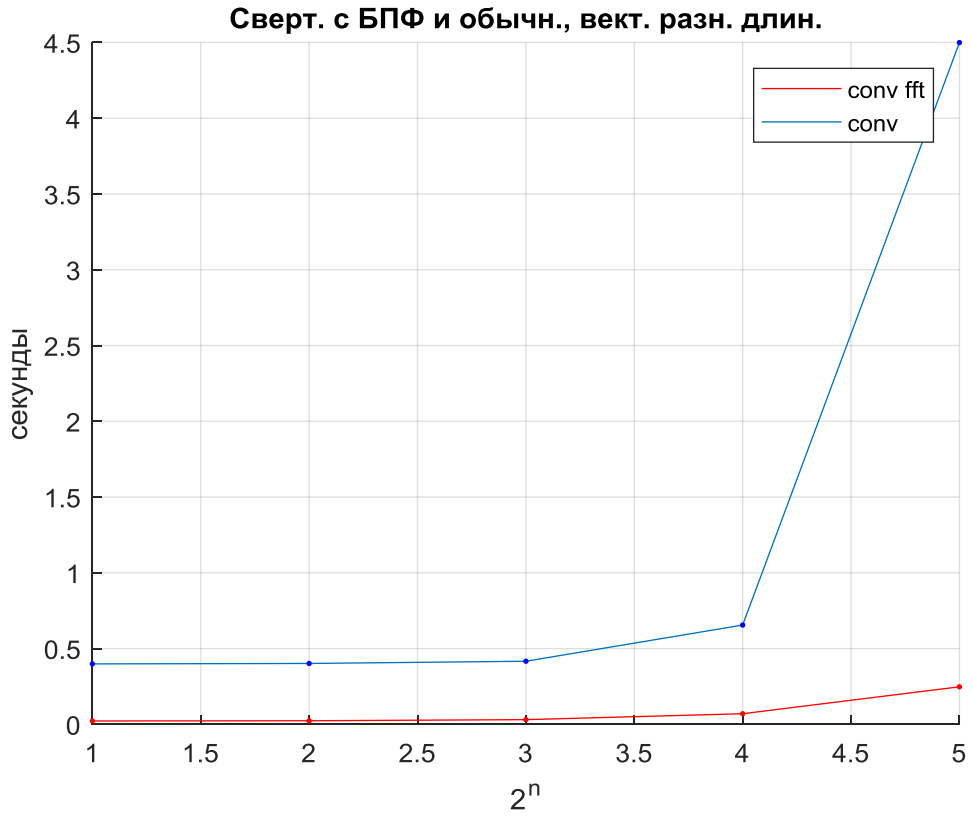
Графики построенные по полученным результатам:

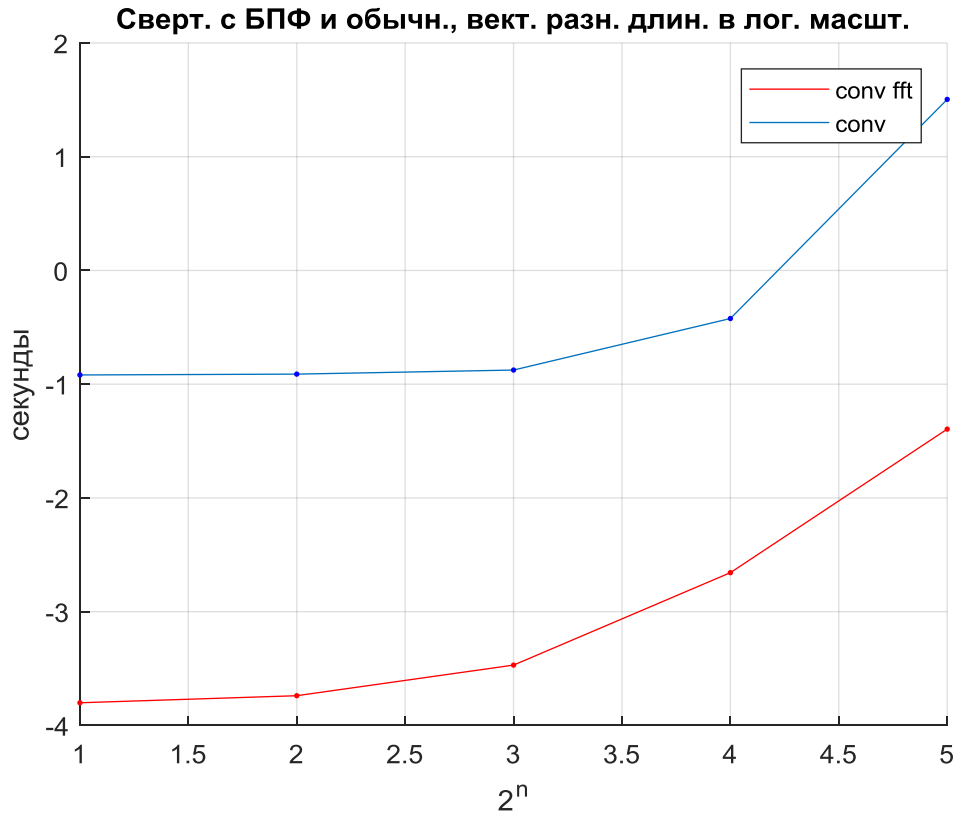



Сверт. с БПФ и обычн., один из вект. фикс. длины в лог. масшт.
conv fft
conv
секунды
1.5
1
0.5
0
-0.5
-1
-1.5
-2
1
2
3
4
5
6
7
8
9
10
2^n


Сверт. с БПФ и обычн., вект. одинак. длин.
conv fft
conv
секунды
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9
10
2^n


Сверт. с БПФ и обычн., вект. одинак. длин. в лог. масшт.
conv fft
conv
секунды
2^n
2
0
-2
-4
-6
-8
-10
1
2
3
4
5
6
7
8
9
10


Сверт. с БПФ и обычн., вект. разн. длин.
conv fft
conv
секунды
2^n
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
1.5
2
2.5
3
3.5
4
4.5
5



В случае, когда длина одного из входных векторов фиксирована и равна $N_1 = 1024$, а длина другого равна $N_2 = 2^n$, ($n = 1,\ldots 10$), количество комплексных умножений при непосредственном вычислении свёртки всегда равно: $\max(N_1, N_2) \wedge 2 = N_1^2 = 2^{20}$, а для свёртки с использованием БПФ количество комплексных умножений всегда равно: $\max(N_1, N_2)*log(\max(N_1, N_2)) = = N_1 log(N_1) = 2^{10} log(2^{10}) = 5 * 2^{11} log(2)$, поэтому время вычисления в обоих случаях практически не зависит от значения n, однако в первом случае требуется значительно больше времени чем во втором, что можно наблюдать на графиках.

В случае когда вычисляется свёртка входных векторов одинаковой длины, где оба вектора длины $N = 2^n$ ($n = 1,\ldots,10$), непосредственное вычисление свёртки требует $N^2 = 2^{2n}$ комплексных умножений, а вычисление свёртки с использованием БПФ требует порядка $NlogN = 2^n log(2^n) = n2^n log(2)$ комплексных умножений, и, при увеличении длин векторов входных значений, время для непосредственного вычисления свёртки возрастает быстрее чем время вычисления свёртки с использованием БПФ, что можно наблюдать на графиках.

В случае когда вычисляется свёртка входных векторов разной длины, где один из векторов длины $N_1 = 2^n$ ($n = 1,\ldots,5$), а второй – длины $N_2 = 2^{2n}$ ($n = 1,\ldots,5$) количество комплексных умножений при непосредственном вычислении свёртки равно: $\max(N_1, N_2)\text{^}2 = N_2^2 = 2^{4n}$, для свёртки с использованием БПФ количество комплексных умножений равно: $\max(N_1, N_2)*log(\max(N_1, N_2)) = N_2 log(N_2) = 2^{2n} log(2^{2n}) = n2^{2n+1} log(2)$, время вычисления во втором случае растёт значительно быстрее чем в первом, что видно на графиках.