**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Радиофизический факультет

Кафедра радиотехники

Специальность «Информационная безопасность

телекоммуникационных систем»

Отчёт

о практике по получению первичных профессиональных умений, в том числе первичных умений и навыков научно-исследовательской деятельности

**Реализация алгоритма быстрой свертки на процессорной системе**

Руководитель практики от ННГУ   
к.ф.-м.н. \_\_\_\_\_\_\_\_\_\_ Козлов С.А.

Руководитель практики от

профильной организации   
начальник отдела разработки

импульсных приборов \_\_\_\_\_\_\_\_\_\_ Куликов Ю.М.

Студент 3-го курса \_\_\_\_\_\_\_\_\_\_ Галкин М.Е.

Нижний Новгород

2023 год

**Содержание**

Введение………………………………………………………………………3

1. Алгоритм свертки

2. Реализация алгоритма быстрой свертки

3. Оценка вычислительной сложности

Заключение…………………………………………………………………..9

Список литературы………………………………………………………….10

Приложение………………………………………………………………….11

# 

# Введение

Быстрая свертка сигналов является одним из важных и актуальных аспектов современной обработки сигналов и цифровой обработки информации. Вот несколько причин, почему быстрая свертка сигналов остается актуальной:

1. Обработка больших объемов данных: В настоящее время объемы данных, с которыми мы имеем дело, постоянно растут. Быстрая свертка сигналов позволяет эффективно обрабатывать эти объемы данных, обеспечивая быстрое и эффективное вычисление сверток сигналов, что является ключевым фактором во многих областях, таких как обработка изображений, обработка аудио и видео, медицинская диагностика и т.д.
2. Распараллеливание вычислений: Свертка сигналов является вычислительно интенсивной операцией, которая может быть распараллелена для ускорения обработки данных. Быстрые алгоритмы свертки, такие как быстрая свертка Фурье (FFT), позволяют эффективно распараллеливать вычисления и использовать вычислительные ресурсы современных многоядерных процессоров и графических процессоров (GPU).
3. Сжатие и передача данных: Быстрая свертка сигналов также актуальна для сжатия и передачи данных. Например, в сжатии изображений и видео используются алгоритмы свертки, такие как дискретное косинусное преобразование (DCT), которые позволяют эффективно представить сигналы с минимальной потерей информации.
4. Обработка в реальном времени: Многие приложения требуют обработки сигналов в реальном времени, например, обработка звука и видео в потоковых сервисах или обработка сигналов в реальном времени в медицинской диагностике. Быстрая свертка сигналов позволяет обеспечить высокую скорость обработки данных, что особенно важно для таких приложений.

Целью данной работы является реализация алгоритма быстрой свертки на процессорной системе. В качестве языка программирования используется Python. В ходе выполнения практики по получению первичных профессиональных умений, в том числе умений и навыков научно-исследовательской деятельности работы, решались следующие задачи:

− разработка программы на языке программирования Python, реализующую алгоритм свертки;

− оценка вычислительной сложности.

# Алгоритм свертки

Для дальнейших рассуждений нам необходимо понятие импульсной характеристики. Импульсной характеристикой h(k) называется реакция цепи на входное воздействие в виде функции Дирака (δ-функции). Она отражает влияние цепи на сигнал.

Алгоритм выполнения свертки заключается в том, что мы вычисляем значение каждой точки в результирующем сигнале как взвешенную сумму некоторого множества соседних точек исходного сигнала. Коэффициенты этой суммы совпадают с импульсной характеристикой линейной системы, перевернутой относительно точки 0.

Обычно все сигналы, обрабатываемые на компьютере, имеют конечную продолжительность (т.е. отличны от нуля лишь на конечном отрезке). Рассмотрим, что происходит с сигналом конечной продолжительности, когда его сворачивают с конечным ядром свертки.

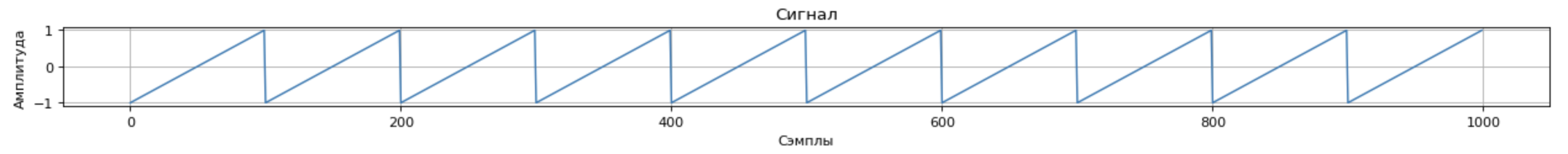
Пусть сигнал x[n] отличен от нуля только на отрезке от 0 до N-1 включительно («имеет длину N»). Пусть ядро свертки h[n] отлично от нуля на отрезке от – m1 до m2 включительно, состоящем из M точек (M = m1 + m2 + 1). Тогда при подстановке этих сигналов в формулу свертки, мы получим сигнал y[n], который отличен от нуля на отрезке от -m1 до N-1+m2 включительно. Таким образом, длина результирующего сигнала равна N+M-1, т.е. сумме длин исходного сигнала и ядра свертки минус один. Итак, операция свертки расширяет сигнал на M-1 точку, где M – длина ядра свертки.

Свойства свертки:

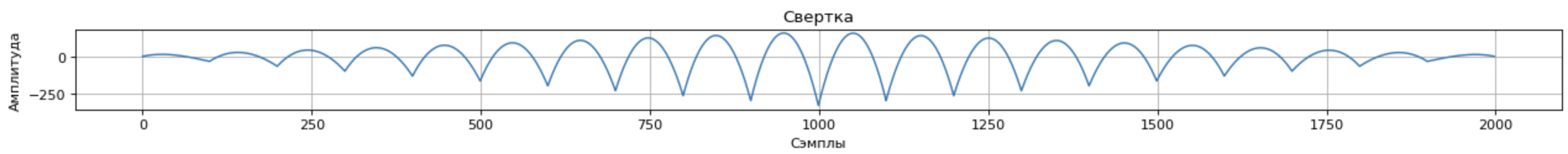
1. x[n]∗ y[n] = y[n]∗ x[n] (т.е. можно переставлять местами исходный сигнал и ядро свертки – это свойство редко используется на практике).
2. (x[n] ∗ y[n]) ∗ z[n] = x[n] ∗ (y[n] ∗ z[n]) (т.е. вместо того, чтобы проводить свертку по очереди в разных системах, можно получить систему с ядром ( y[n] ∗ z[n]), которая является суперпозицией систем y[n] и z[n]).
3. x[n] ∗ y[n] + x[n] ∗ z[n] = x[n] ∗ (y[n] + z[n])

Реализация свертки на Python

1. Имеются два входных сигнала – в данном случае два одинаковых пилообразных сигнала, их и будем сворачивать. Сигнал формируется функцией sawtooth из библиотеки signal.



1. Далее вызывается функция, реализующая свертку по вышеуказанному алгоритму. Она возвращает выходной сигнал.
2. После завершения операции свертки получается выходной сигнал, размерность которого зависит от длины входного сигнала и размера ядра сигнала.



1. **Реализация алгоритма быстрой свертки**

В реализации быстрой свертки участвуют несколько определений.

Во-первых: свертка двух сигналов во временной области равна произведению их спектров в частотной области

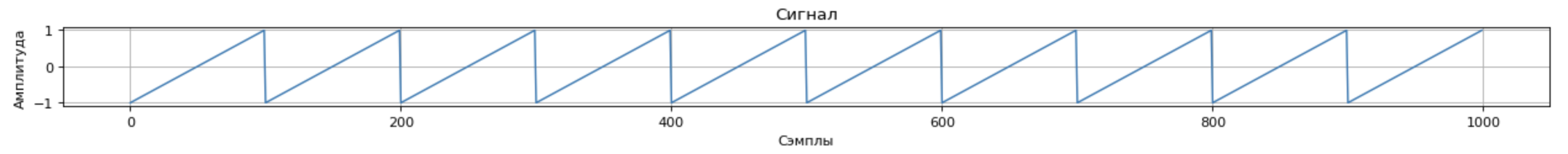
a(n)∗b(n)=A(k)⋅B(k)

Следовательно мы можем представить сигналы в виде ряда Фурье, перемножить их друг с другом и осуществить обратное преобразование во временную область.

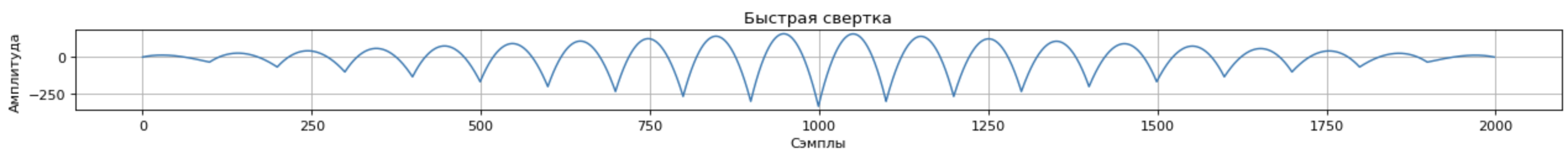
Во-вторых: чтобы добиться выигрыша в скорости и уменьшить количество вычислений, используют быстрое преобразование Фурье (БПФ). В БПФ дискретное преобразование Фурье (ДПФ) разбивается на слагаемые (четные и нечетные) и за счет периодичности нужно посчитать лишь половину итераций, исключая повторные вычисления. ДПФ разбивается на столько слагаемых, сколько имеет смысл (выигрыш в вычислительных операциях).

Реализация быстрой свертки на Python

1. Имеются два входных сигнала – все те же пилообразные сигналы.



1. Вызывается функция, в которой для каждого сигнала вычисляется БПФ функцией fft из пакета numpy, после чего два полученных сигнала перемножаются и берется ОБПФ (обратное БПФ)
2. После завершения операции быстрой свертки, получается выходной сигнал, размерность которого зависит от длины входного сигнала и размера ядра сигнала.



1. **Оценка вычислительной сложности**

В случае обычной свертки, требуется большое количество итераций, а именно N\*N, где N – размер входного сигнала. Однако в быстрой свертке, благодаря разделению ДПФ на несколько частей и отбрасыванию ненужных расчетов, количество итераций снижается до N\*log2(N)

Например, для входного сигнала размером 105 быстрая свертка оказалась быстрее прямой почти в 305 раз

Ниже представлена таблица сравнения эффективности быстрой свертки и свертки, вычисляемой по прямой формуле (по числу вещественных умножений).

|  |  |  |  |
| --- | --- | --- | --- |
| N  (размер входного сигнала) | Свертка  (кол-во итераций) | Быстрая свертка  (кол-во итераций) | Отношение  (свертки к быстрой свертке) |
| 8 | 64 | 432 | 0,15 |
| 16 | 256 | 1152 | 0,22 |
| 32 | 1024 | 2880 | 0,36 |
| 64 | 4096 | 6912 | 0,59 |
| 128 | 16384 | 16128 | 1,02 |
| 256 | 65536 | 36864 | 1,78 |
| 1024 | 1048576 | 184320 | 5,69 |
| 2048 | 4194304 | 405504 | 10,34 |

Как видно, для длин БПФ до 128, быстрая свёртка проигрывает у прямого метода. Однако, при увеличении длины БПФ результаты меняются в обратную сторону — быстрая свертка начинает выигрывать у прямого метода. Очевидно, чем больше длина БПФ, тем лучше выигрыш частотного метода.

Одной из проблем с быстрой дискретной сверткой является ее задержка , которая вызвана ожиданием набора значений (выборок), соответствующих размеру блока N, для вычисления быстрой дискретной свертки: Размер блока должен, по крайней мере, соответствовать длине сигнала во временной области, с которой функция должна быть свернута, иначе результат будет короче, чем импульсная характеристика свертки. Кроме того, при использовании метода добавления с перекрытием или сохранения с перекрытием, если этот метод должен полностью избежать создания артефактов, эта минимальная длина блока также должна быть увеличена на расстояние между отдельными пакетами, в которых выполняется свертывание быть выполненным, вот почему блоки большой длины дают результаты с более высокой эффективностью. Кроме того, в зависимости от конкретной реализации FFT, условие может все еще существуют, что размер блока N должен соответствовать целой степени 2, 4 или 8 - которые в совокупности могут привести к очень большому размеру блока N в конце. Другим недостатком является шум квантования, который труднее вычислить по всей операции свертки. Это обычно выше при быстрой свертке, чем при дискретной свертке. Проблема шума квантования должна быть принята во внимание, особенно с процессорами сигналов с арифметикой с фиксированной точкой.

# Заключение

По результатам выполненной работы:

− разработана программа на языке программирования Python, реализующая алгоритм свертки и быстрой свертки;

− проведена оценка вычислительной сложности алгоритма свертки и быстрой свертки.

Алгоритм быстрой свертки быстрее при больших размерах входного сигнала, и чем больше размер, тем больше выигрыш во времени по сравнению с прямой сверткой.

В случае с размером 105 выигрыш во времени составил 2203%

В целом, быстрая свертка является более эффективным методом для выполнения свертки сигнала по сравнению с прямой сверткой, особенно для длинных сигналов. Однако при выборе между ними необходимо учитывать специфические требования приложения и ограничения алгоритма БПФ.

Код программы можно увидеть по ссылке в ПРИЛОЖЕНИИ 1.

**Список литературы**

1. Введение в цифровую обработку сигналов (математические основы) Алексей Лукин - Учебное пособие Факультета ВМК МГУ, 2007. - 54 с

2.Практикум по компьютерным технологиям на языке программирования python3

[электронный ресурс]. URL:

<https://fadeevlecturer.github.io/python_lectures/docs/index.html>. (25.05.2023)  
3. Аллен Б.Дауни. Цифровая обработка сигналов на языке Python. – М.: ДМК Пресс,O'Reilly Media, 2017. – 160 с.

4. А.И. Солонина, Д.А. Улахович, С.М. Арбузов, Е.Б. Соловьева. Основы цифровой обработки сигналов. Курс лекций. – СПб.: БХВ-Петербург, 2005. – 768 с.

5. С.В. Умняшкин. Основы теории цифровой обработки сигналов. Учебное пособие. – М.: Техносфера, 2016. – 528 с.

**Приложение**

1. Код программы:

<https://colab.research.google.com/drive/16E_tabuDrMd3DOZK_L3_ep5ubbrrIJnG?usp=sharing>

1. Графики

