### Санкт-Петербургский политехнический университет Петра Великого Институт компьютерных наук и кибербезопасности

### Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Телекоммуникационные технологии

### Отчёт по лабораторным работам

**Студент:**

### Д. Д. Брык

### Группа: 5130901/20103

**Преподаватель:**

### Н. В. Богач

### Санкт-Петербург 2024

**Содержание**

1. [Лабораторная работа 1. Сигналы и звуки](#_bookmark0) 4

[1.1. Упражнение 1.2](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

[1.2. Упражнение 1.3](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

[1.3. Упражнение 1.4](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

1. [Лабораторная работа 2. Гармоники](#_bookmark4) 6

[2.1. Упражнение 2.2](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

[2.2. Упражнение 2.3](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

[2.3. Упражнение 2.4](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

[2.4. Упражнение 2.5](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

1. [Лабораторная работа 3. Апериодические сигналы](#_bookmark9) 11
   1. [Упражнение 3.1](#_bookmark10) 11
   2. [Упражнение 3.2](#_bookmark11) 11
   3. [Упражнение 3.3](#_bookmark12) 12
   4. [Упражнение 3.4](#_bookmark13) 12
   5. [Упражнение 3.5](#_bookmark14) 13
   6. [Упражнение 3.6](#_bookmark15) 13
2. [Лабораторная работа 4. Шум](#_bookmark16) 14
   1. [Упражнение 4.1](#_bookmark17) 14
   2. [Упражнение 4.2](#_bookmark18) 15
   3. [Упражнение 4.3](#_bookmark19) 15
   4. [Упражнение 4.4](#_bookmark20) 16
   5. [Упражнение 4.5](#_bookmark21) 16
3. [Лабораторная работа 5. Автокорреляция](#_bookmark22) 17
   1. [Упражнение 5.2](#_bookmark23) 17
   2. [Упражнение 5.3](#_bookmark24) 17
4. [Лабораторная работа 6. Дискретное косинусовое преобразование](#_bookmark25) 18
   1. [Упражнение 6.1](#_bookmark26) 18
   2. [Упражнение 6.2](#_bookmark27) 19
5. [Лабораторная работа 7. Дискретное преобразование Фурье](#_bookmark28) 19
   1. [Упражнение 7.2](#_bookmark29) 20
6. [Лабораторная работа 8. Фильтрация и свертка](#_bookmark30) 20
   1. [Упражнение 8.2](#_bookmark31) 20
   2. [Упражнение 8.3](#_bookmark32) 21
7. [Лабораторная работа 9. Дифференцирование и интегрирование](#_bookmark33) 22
   1. [Упражнение 9.2](#_bookmark34) 22
   2. [Упражнение 9.3](#_bookmark35) 24
   3. [Упражнение 9.4](#_bookmark36) 25
   4. [Упражнение 9.5](#_bookmark37) 26
8. [Лабораторная работа 10. Линейные стационарные системы](#_bookmark38) 28
   1. [Упражнение 10.1](#_bookmark39) 28
   2. [Упражнение 10.2](#_bookmark40) 29
9. [Лабораторная работа 11. Модуляция и выборка](#_bookmark41) 31
   1. [Упражнение 11.3](#_bookmark42) 31

**Весь код и пояснение к нему находится в файле telecomBryk20103.ipynb.**

# Лабораторная работа 1. Сигналы и звуки

## Упражнение 1.2

Скачаем с сайта https://freesound.org/ образец звука и различными способами исследуем его. Для удобной работы с сигналами здесь, и в дальнейших работах будем использовать библиотеку thinkdsp.

Для начала, "распечатаем" нашу аудиозапись с помощью readWave()

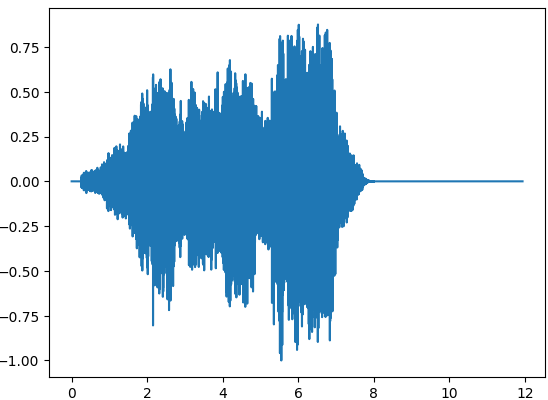


Рисунок 1.1. Wave для полной звукозаписи

Далее выделим промежуток c 6 секунды длительностью 0.7.

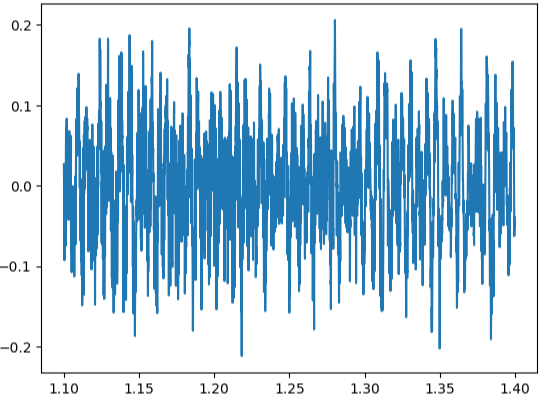


Рисунок 1.2. Wave для части звукозаписи

Теперь разложим в спектр. Другими словами, представим сигнал как сумму синусоид с разными частотами

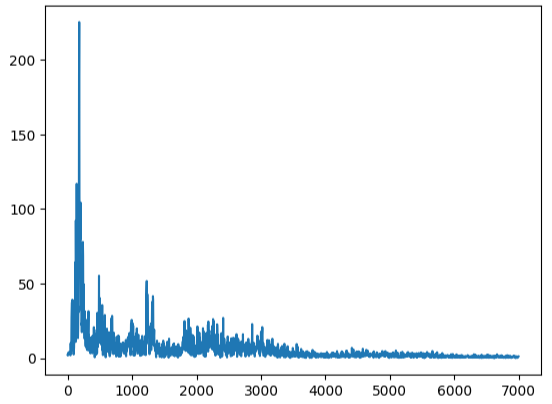


Рисунок 1.3. Разложенный в спектр сигнал

Воспользуемся lowPass и highPass для фильтрации гармоник. После изменений в спектре, звук стал выше и более "отдаленный"

## Упражнение 1.3

Создадим объекты SinSignal и CosSignal, далее просуммируем их. Получили следующий "сложный" сигнал:

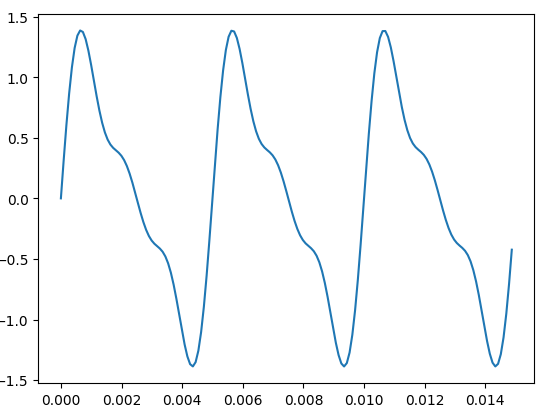


Рисунок 1.4. Сигнал, полученный в результате сложения

Вычислим спектр этого сигнала и распечатаем.

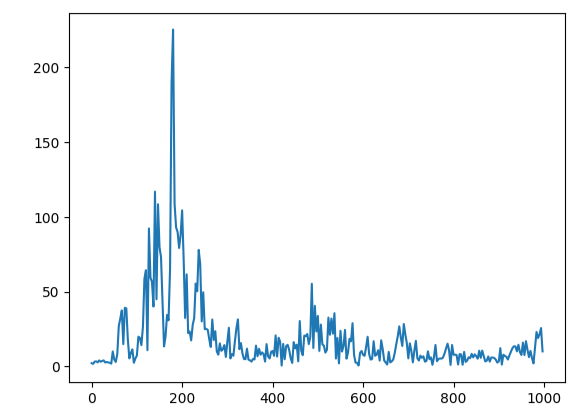


Рисунок 1.5. Спектр полученного сигнала

## Упражнение 1.4

Напишем функцию stretch, которая будет в качестве параметров будет брать wave и коэффициент изменения.

# Лабораторная работа 2. Гармоники

## Упражнение 2.2

Напишем класс, который будет называться SawtoothSignal и будет наследоваться от Signal. Переопределим метод evaluate, который будет строить пилообразный сигнал.



Рисунок 2.1. Полученный сигнал

Можно заметить, что сигнал похож на пилу вверху и внизу. Если включить получившийся звук, то услышим довольно неприятный для человеческого уха звук.

Разложим этот сигнал в спектр и выведем.

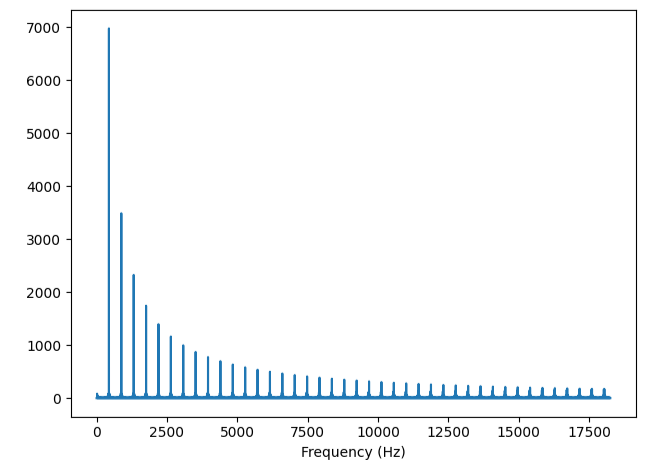


Рисунок 2.2. Спектр полученного пилообразного сигнала

## Упражнение 2.3

Создадим прямоугольный сигнал 1100 Гц.



Рисунок 2.3. Прямоугольный сигнал

Вычислим спектр сигнала

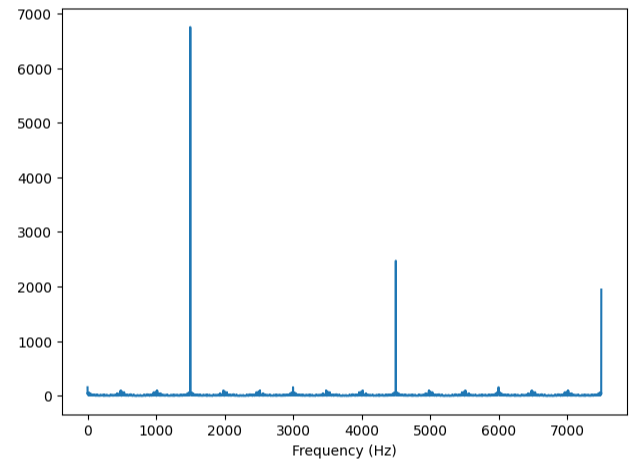


Рисунок 2.4. Спектр прямоугольного сигнала

## Упражнение 2.4

Создадим треугольный сигнал.

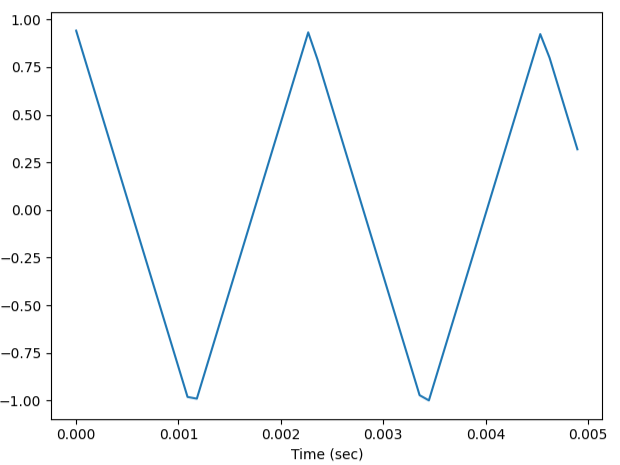


Рисунок 2.5. Треугольный сигнал

Вычислим его спектр. Посмотрим на hs[0], он равен примерно 3.375. Если мы установим hs[0] = 100, то высота графика увеличится.

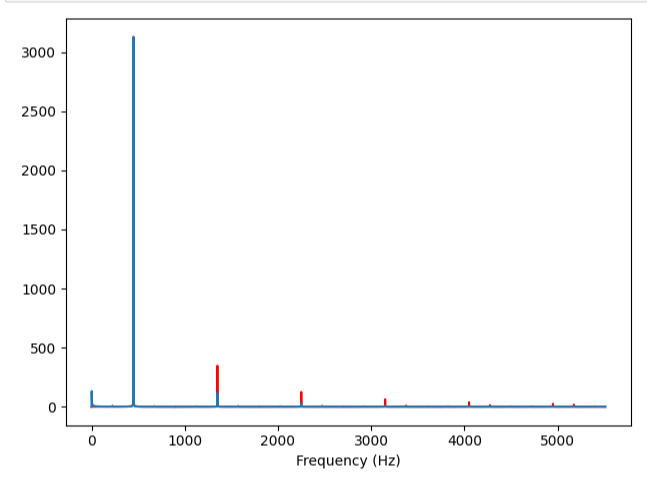


Рисунок 2.6. Треугольный сигнал

## Упражнение 2.5

Напишем функцию, которая devideSpec, которая будет принимать Spectrum как параметр и будем делить hs на fc. Код в приложении.

Протестируем функцию на прямоугольном сигнале.

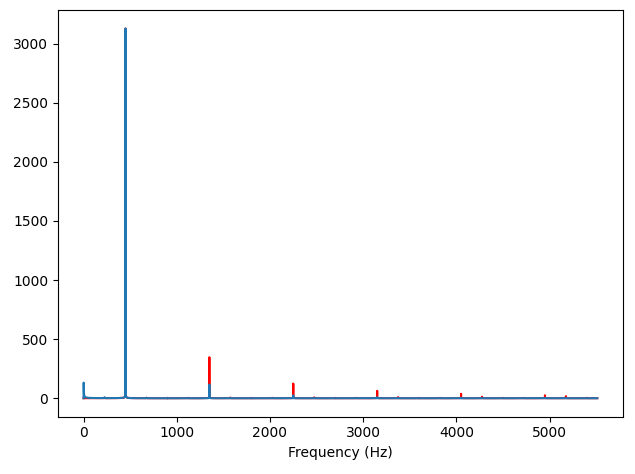


Рисунок 2.7. Спектр прямоугольно сигнала

Результат функции:

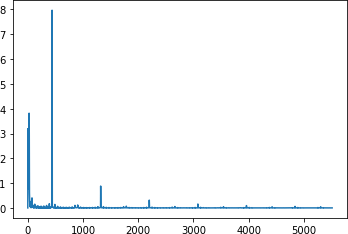


Рисунок 2.8. Результат функции devideSpec

Далее протестируем функцию на пилообразном сигнале.

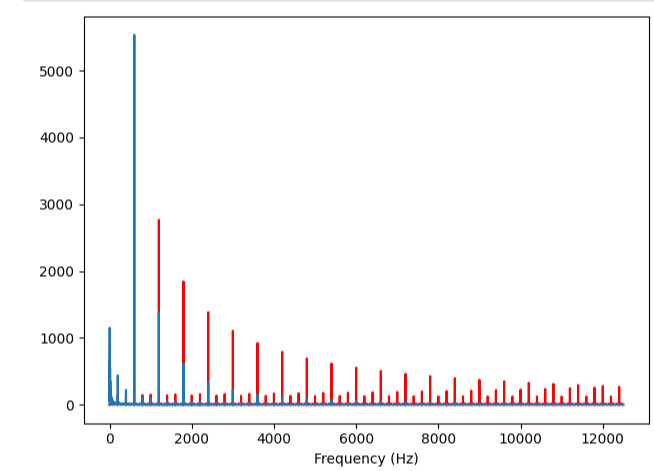


Рисунок 2.9. Результат функции devideSpec на пилообразном сигнале

Посмотрим, что будет, если сделать wave из измененного сигнала

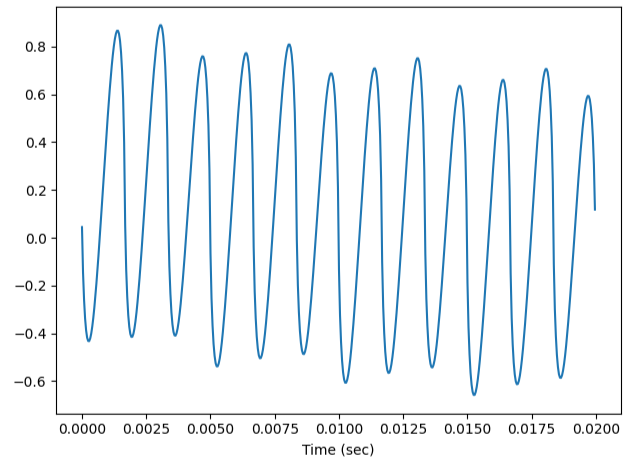


Рисунок 2.10. Wave из измененного сигнала

# Лабораторная работа 3. Апериодические сигналы

## Упражнение 3.1

Утечка — это момент, когда сложно оценить пик, так как некоторые части уходят в другие частоты.

Окна - средства чтобы уменьшить утечку. По сути, окно — это функцию, которая преобразует апериодический сигнал в что-то похожее на периодический. Рассмотрим несколько окон - окно Хэмминга и окно Блэкмена.

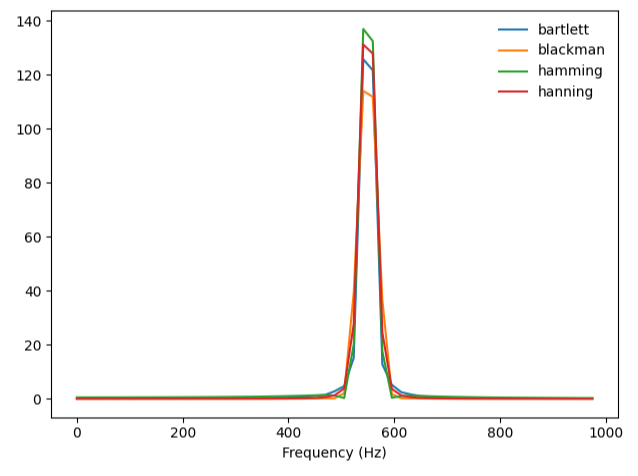


Рисунок 3.1. Окна Хэмминга и Блэкмена

Основная разница в частотах. У окна Хэмминга она выше.

## Упражнение 3.2

Напишем класс SawtoothChirp, который наследуется от Chirp и переопределим метод

evaluate для генерации пилообразного сигнала с линейно увеличивающейся частотой.

Создадим пример такого сигнала и выведем его wave.

Теперь выведем спектрограмму.

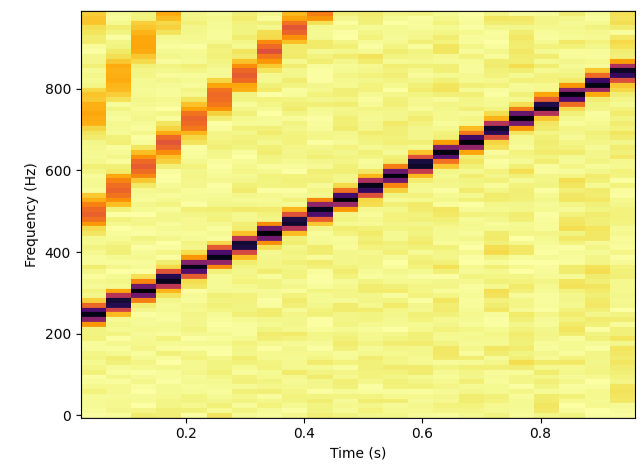


Рисунок 3.2. Спектрограмма полученного Chirp

## Упражнение 3.3

Создадим пилообразный чирп, который меняется от 2500 до 3000 Гц, затем сгенерируем сигнал и посмотрим на спектрум.

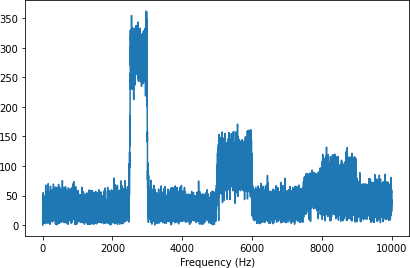


Рисунок 3.3. Спектрограмма полученного сигнала

## Упражнение 3.4

В музыкальной терминологии "глиссандо" — это нота, которая меняется от одной высоты до другой, поэтому она своего рода чирп. Найдем глиссандо и распечатаем ее спектрограмму.

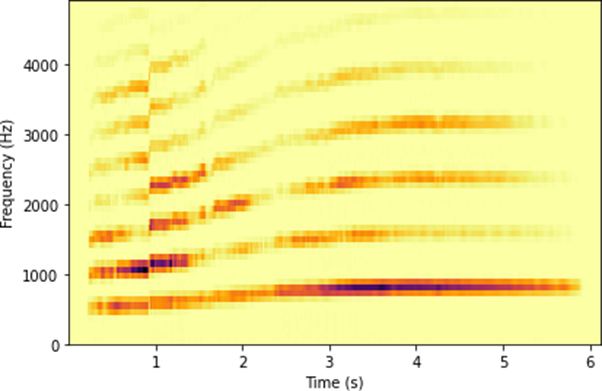


Рисунок 3.4. Спектрограмма глиссандо

## Упражнение 3.5

Создадим класс TromboneGliass, который наследуется от класса Chirp и переопределим evaluate для того, чтобы создать сигнал, имитирующий игру на тромбоне.

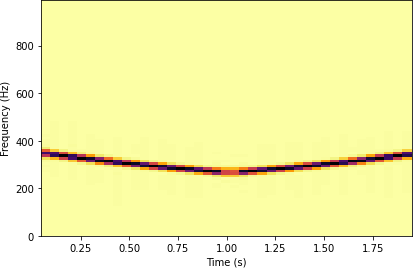


Рисунок 3.5. Спектрограмма сигнала игры на трамбоне

По спектрограмме, мы идем от верху к низу и обратно.

## Упражнение 3.6

Найдем аудиозапись, на которой человек произносит серию гласных звуков и посмотрим на спектрограмму.

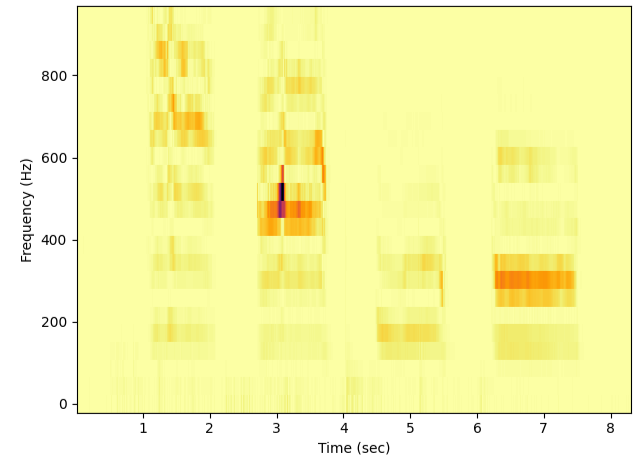


Рисунок 3.6. Спектрограмма сигнала моего голоса

Судя по спектрограмме, буквы очень похожие, но сильно выделяется буква О

# Лабораторная работа 4. Шум

## Упражнение 4.1

Рассмотрим некоторые источники шума из природы (звуки дождя) и вычислим их спектр.

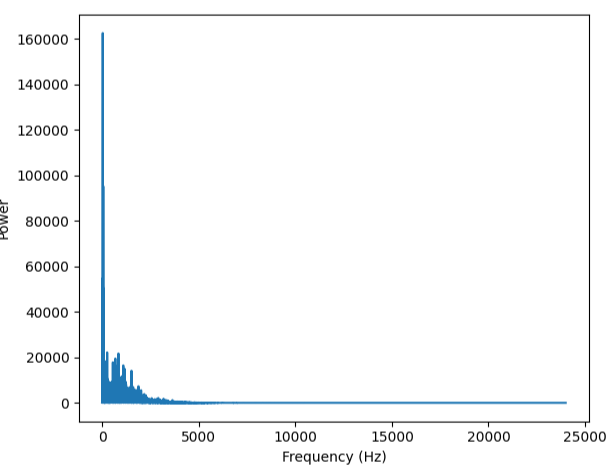


Рисунок 4.1. Спектр звука дождя

Переведем это в двойной логарифмический формат.

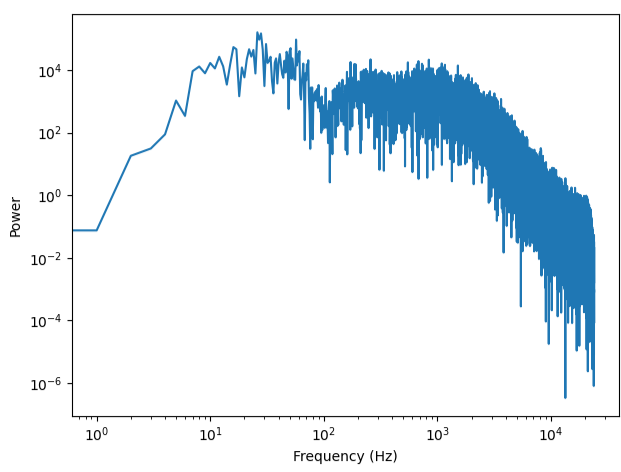


Рисунок 4.2. Спектр звука океана в log формате

Спектр похож на Броуновский шум

## Упражнение 4.2

Реализуем метод Барлета и воспользуемся им для оценки спектра мощности шумового сигнала. Как тестовые данные возьмем 2 фрагмента из записи звуков океана Код в приложении. Результат работы метода:

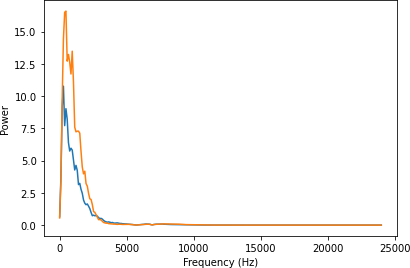


Рисунок 4.3. Метод барлета для сегментов звука

Метод Барлета используется для того, чтобы разорвать сигнал на сегменты, вычис лить спектр мощности для каждого сегмента и затем найти среднее.

## Упражнение 4.3

Скачаем данные об истории стоимости биткоина. Вычислим спектр цен.

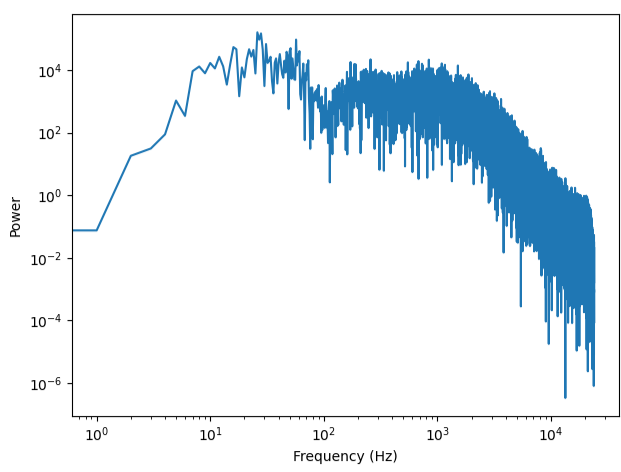


Рисунок 4.4. Спектр графика изменения цен Биткоина

Такой спектр похож на розовый шум

## Упражнение 4.4

Для этого, напишем класс UncorrelatedPoissonNois который наследует Noise. Переопределим метод evaluate. Сгенерируем пару секунд звука, послушаем и вычислим спектры.

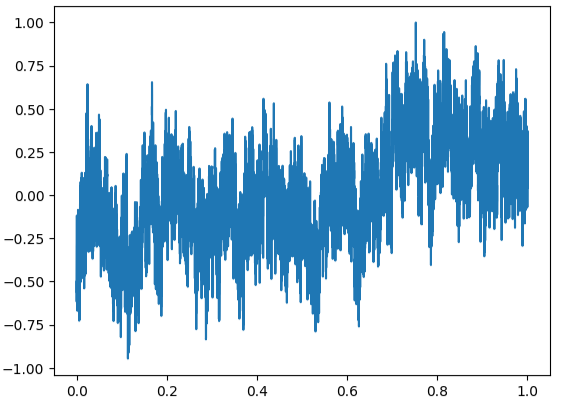


Рисунок 4.5. Спектр сигнала случайного блуждания

## Упражнение 4.5

Реализуем алгоритм Voss-McCartney для генерации розового шума.

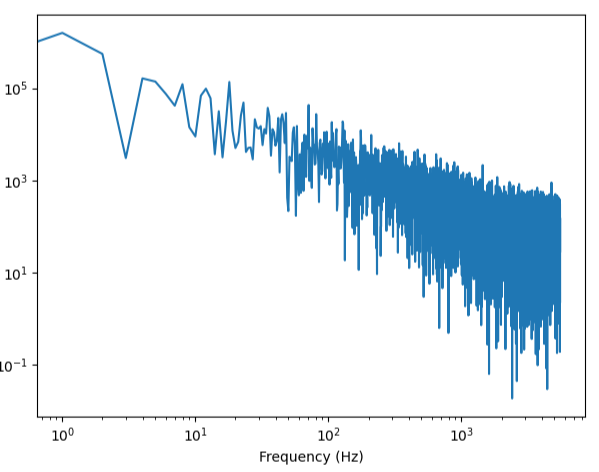


Рисунок 4.6. Результат работы генератора

# Лабораторная работа 5. Автокорреляция

## Упражнение 5.2

Инкапсулируем функцию estimateFundamental и воспользуемся ей для оценки высоты тона записывающего звука

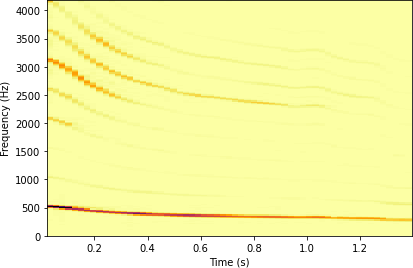


Рисунок 5.1. Спектрограмма исследуемого звука

После выполнения функции получили результат - 364.46

## Упражнение 5.3

Используем данные для биткоина, которые использовали в упражнении 4.2. На этот раз вычислим автокорреляцию. Для этого напишем соответствующие функции.

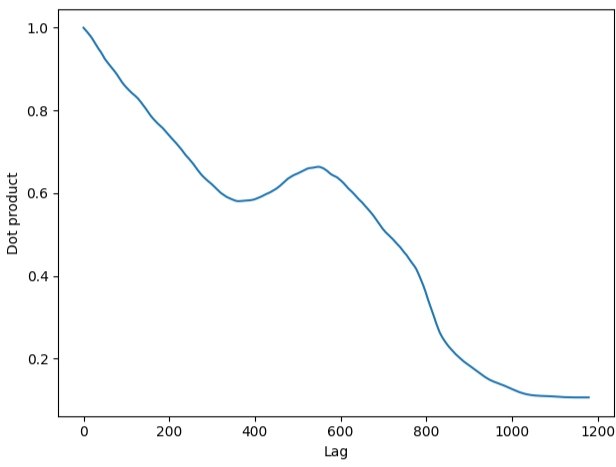


Рисунок 5.2. Результат автокорреляции

# Лабораторная работа 6. Дискретное косинусовое преобразование

Дискретное Косинусовое Преобразование (ДКП) используется в MP3 и прочих форматах сжатия музыки, в JPEG для сжатия изображений.

## Упражнение 6.1

Утверждается, что analyze1 требует времени пропорционально *n*3, а analyze2 *n*2. Проведем небольшое исследование.

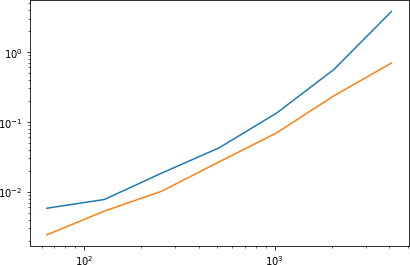


Рисунок 6.1. Результат сравнения

Утверждение похоже на правду

.**6.2. Упражнение 6.2**

Воспользуемся основным свойством ДКП, а именно - сжатие звука. Реализуем алгоритм и посмотрим на результат.

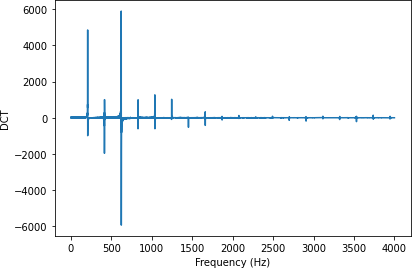


Рисунок 6.2. Wave исходного звука

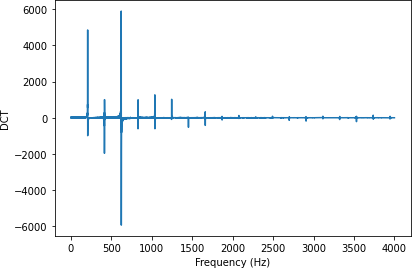


Рисунок 6.3. Wave обработанного звука

Графики очень похожие, звуки тоже идентичны. При этом, мы сжали запись на 8

# Лабораторная работа. Дискретное преобразование Фурье

# Разница между ДКП и ДФП в том, что в ДФП мы используем не косинусы, а комплексные экспоненциальные функции.

*N*

Общая формула:

, где N = 0, 1..N-1

Для начала исследуем библиотечную функцию fft. Подадим ys = [-0.5, 0.1, 0.7, -0.1].

Получили результат:

[ 0.2+0.j -1.2-0.2j 0.2+0.j -1.2+0.2j]

Теперь разработаем алгоритм и подадим те же данные. В результате получили:

array([0.2+0.00000000e+00j,-1.2-2.00000000e-01j, 0.2+1.95943488e-16j, -1.2+2.00000000e])

Можно убедиться, что функция работает верно.

Теперь реализуем рекурсивную функцию fft. Подадим те же данные. В результате получили те же данные. Вывод - функция также работает верно. fft и dft отличаются лишь тем, что fft работает быстрее, а именно за *nlog*(*n*), в то время как dft работает за *n*2

# Лабораторная работа 8. Фильтрация и свертка

## Упражнение 8.2

Рассмотрим, как меняется преобразование Фурье в зависимости от параметра std.

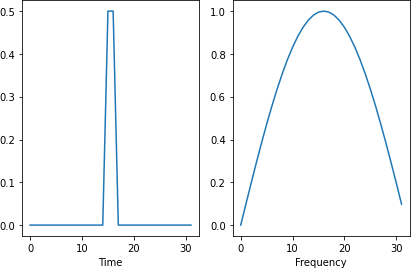


Рисунок 8.1. ДПФ с std = 0.1

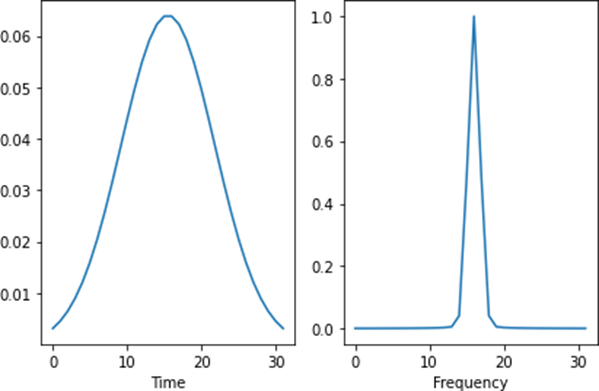


Рисунок 8.2. ДПФ с std = 6.0

## Упражнение 8.3

Рассмотрим различные окна, которые предотвращают утечку.

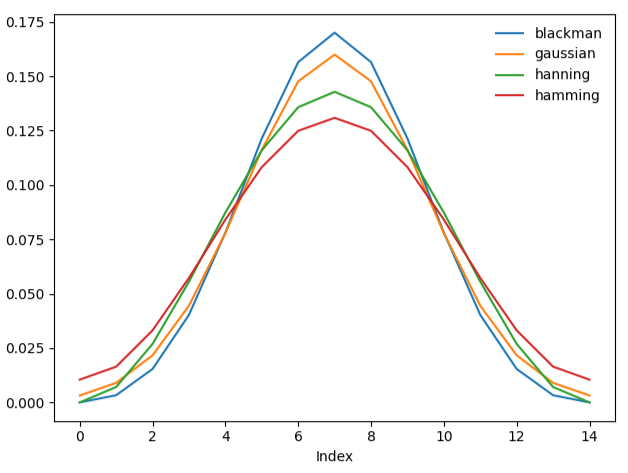


Рисунок 8.3. Различные окна

Также рассмотрим окна с DFT

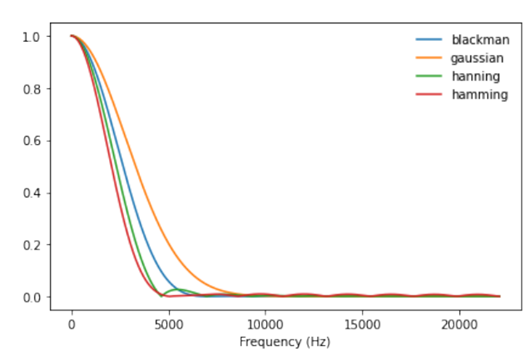


Рисунок 8.4. Различные окна c DFT

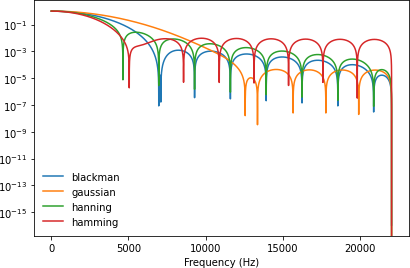


Рисунок 8.5. Различные окна в логарифмическом масштабе

В логарифмическом масштабе мы можем видеть, что Хэмминг и Ханнинг сначала падают быстрее, чем два других.

# Лабораторная работа 9. Дифференцирование и интегрирование

## Упражнение 9.2

Рассмотрим влияние diff и differentiate на сигнал. Для начала, создадим треугольный сигнал.

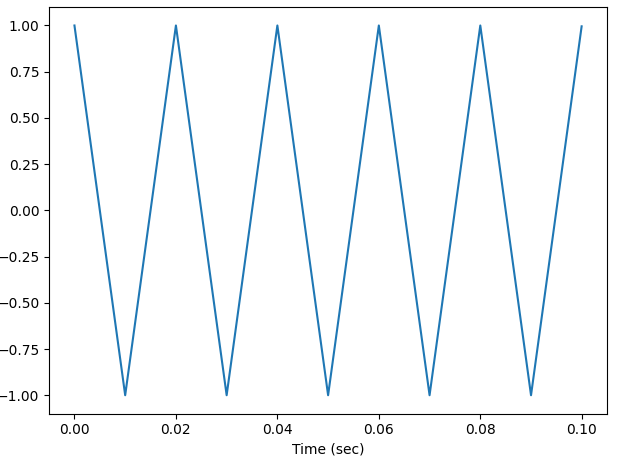


Рисунок 9.1. Треугольный сигнал

Применим diff

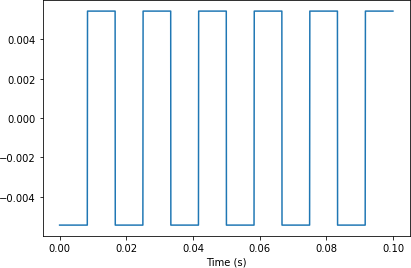


Рисунок 9.2. Результат дифференцирования

Можно заметить, что после дифференцирования мы получили прямоугольный сигнал.

Теперь вычислим спектр, применим differentiate и посмотрим на результат.

Рисунок 9.3. Результат differentiate

## Упражнение 9.3

Создадим прямоугольный сигнал и посмотрим, как работают cumsum и integrate.

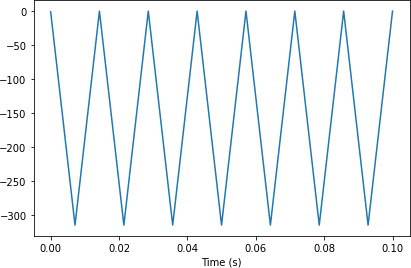


Рисунок 9.4. Результат применения cumsum

Рисунок 9.5. Результат применения integrate

По графикам можно увидеть, что результат применения обеих функций идентичен

## Упражнение 9.4

Рассмотрим двойное интегрирование. Создадим пилообразный сигнал и применим интегрирование.

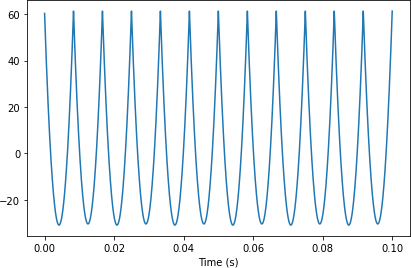


Рисунок 9.6. Результат применения cumsum первый раз

Рисунок 9.7. Результат применения cumsum второй раз

Теперь применим двойное интегрирование к спектру.

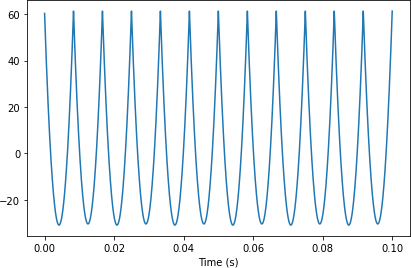


Рисунок 9.8. Результат применения двойного интегрирования

Видно, что результат получился тот же.

## Упражнение 9.5

Исследуем как работает двойная конечная разность. Для этого рассмотрим кубическую функцию.

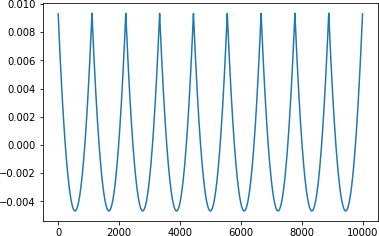


Рисунок 9.9. Результат применения первого применения diff

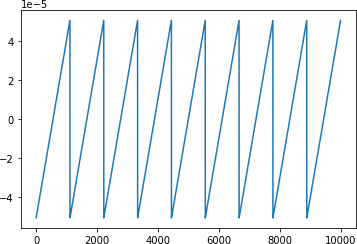


Рисунок 9.10. Результат применения двойного применения diff

Теперь применим двойной diff к спектру

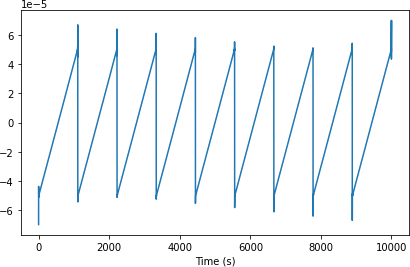


Рисунок 9.11. Результат применения двойного применения diff

# Лабораторная работа 10. Линейные стационарные системы

## Упражнение 10.1

В разделе “Акустическая характеристика” умножение ДПФ сигнала на передаточную функцию соответствует круговой свертке, но в предположении периодичности сигнала. В результате можно заметить, что на выходе, в начале фрагмента, слышна лишняя нота, “затекшая” из конца этого фрагмента. Чтобы устранить эту проблему, нужно перед вычислением ДПФ добавить достаточно нулей в конец сигнала, эффекта “заворота” можно избежать. Я урежу оба сигнала до 216 и добавлю по нулю до 217 Рассмотрим звук виолончели.

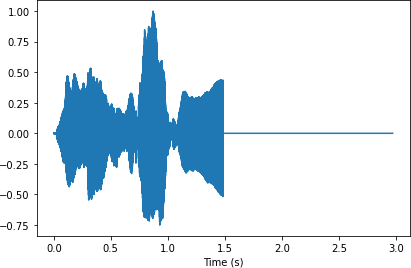


Рисунок 10.1. Наш исходный звук

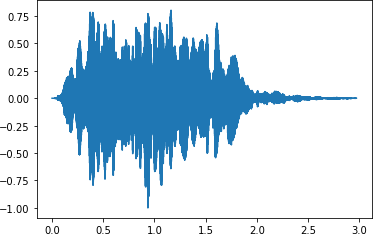


Рисунок 10.2. "Проблемный фрагмент"

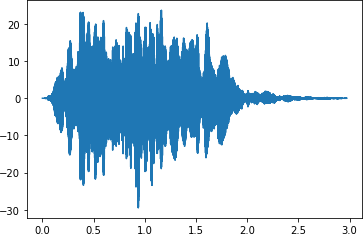


Рисунок 10.3. Результат

Можно заметить, что от проблем с лишней нотой нам удалось избавиться.

## Упражнение 10.2

Возьмем один из звуков с ресурса Open Air с импульсной характеристикой. Дальше, найдем короткие записи с той же частотой дискретизации, что и у скачанной импульсной характеристики. Смоделируем двумя способами звучание записи в том пространстве, где была измерена импульсная характеристика, как сверткой самой записи с импульсной характеристикой, так и умножением ДПФ записи на вычисленный фильтр, соответствующий импульсной характеристике.

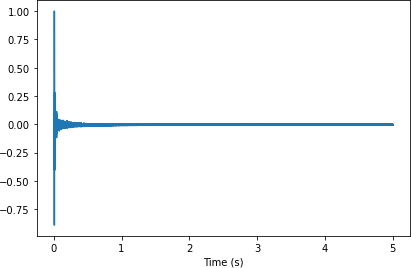


Рисунок 10.4. Wave нашего звука

Рисунок 10.5. ДПФ импульсной характеристики

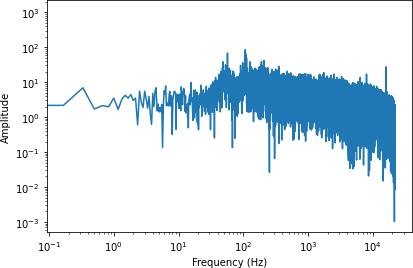


Рисунок 10.6. ДПФ импульсной характеристики в логарифмическом масштабе

Теперь можно смоделировать, как будет звучать запись, если она будет воспроизводиться в комнате, возьмем запись скрипки, которую использовали раньше:

Рисунок 10.7. Wave записи скрипки

Теперь вычислим ДПФ преобразование записи урежем запись до той же длины, что и импульсная характеристика. Результат: (110251, 144001) Умножим в частотной области и преобразуем обратно во временную область.

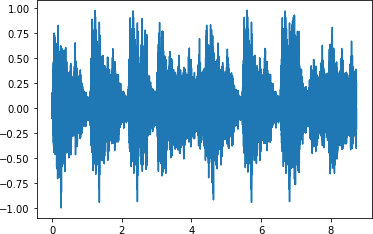


Рисунок 10.8. Wave после преобразования

# Лабораторная работа 11. Модуляция и выборка

## Упражнение 11.3

Рассмотрим пример "Соло на барабане" и применим фильтр НЧ.

Рисунок 11.1. Wave соло на барабане

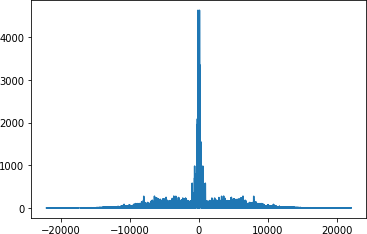


Рисунок 11.2. Спектр сигнала

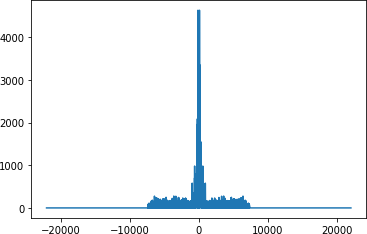


Рисунок 11.3. Спектр отфильтрованного сигнала

Далее напишем функцию, которая имитирует процесс выборки.

Рисунок 11.4. Спектр отфильтрованного сигнала

Однако, тут есть спектральные копии, избавимся от них с помощью НЧ.

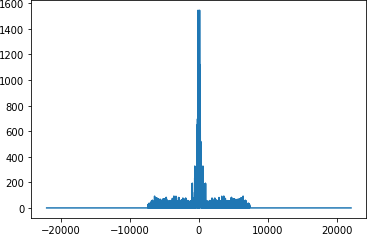


Рисунок 11.5. Спектр отфильтрованного сигнала

Теперь разница между спектром до и после отбора должна быть небольшой: 1.81898940

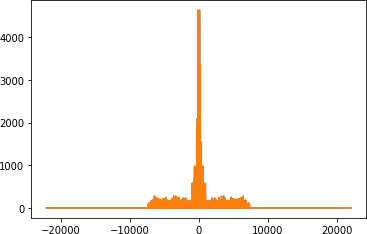


Рисунок 11.6. Разница между спектром до и после отбора

Видно, что разницы почти нет.