Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

(ФГАОУ ВО СПбПУ)

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

**КУРСОВАЯ РАБОТА**

**на тему: "****Анализ и обработка модельных радиосигналов и помех (частотная модуляция)"**

по дисциплине «Методы обработки экспериментальных данных»

Выполнил:   
студент гр.5140203/30102 М. С. Антоненко

Преподаватель: И. Н. Белых

Санкт-Петербург 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc156804572)

[1 Теоретические основы 4](#_Toc156804573)

[1.1 Понятие сигнала 4](#_Toc156804574)

[1.2 Двоичное представление сигнала 4](#_Toc156804575)

[1.3 Понятие модуляции 6](#_Toc156804576)

[1.4 ЧМ модуляция 6](#_Toc156804577)

[1.5 ЛЧМ модуляция 9](#_Toc156804578)

[1.6 Цифровая обработка сигнала и преобразование Фурье 10](#_Toc156804579)

[1.7 Алгоритм работы 12](#_Toc156804580)

[2 Практическая часть 13](#_Toc156804581)

[2.1 Генерация ЧМ сигнала 13](#_Toc156804582)

[2.2 Обработка ЧМ сигнала 15](#_Toc156804583)

[ЗАКЛЮЧЕНИЕ 19](#_Toc156804584)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc156804585)

[ПРИЛОЖЕНИЕ А 21](#_Toc156804586)

[Код программы 21](#_Toc156804587)

[ПРИЛОЖЕНИЕ Б 22](#_Toc156804588)

[model.py 22](#_Toc156804589)

[ПРИЛОЖЕНИЕ В 24](#_Toc156804590)

[analysis.py 24](#_Toc156804591)

[ПРИЛОЖЕНИЕ Г 26](#_Toc156804592)

[proccessing.py 26](#_Toc156804593)

[ПРИЛОЖЕНИЕ Д 29](#_Toc156804594)

[in\_out.py 29](#_Toc156804595)

# ВВЕДЕНИЕ

В сфере передачи данных существует множество технологий и подходов к созданию и обработке сигнала, представленного электромагнитным излучением, в зависимости от параметров скорости, надежности и дальности передачи данных, свойств среды, например открытая местность или городская застройка, а также характеристик аппаратной части. Ко всему прочему распространение радиоволн в среде не только вызывает неизбежные потери и шумы, но и накладывает некоторые ограничения на способы формирования сигнала. В частности, системы цифровой связи заменили собой аналоговый способ передачи за счет больших возможностей обработки информации, не доступные при использовании аналоговой передачи. К тому же аппаратная составляющая приемо-передающего тракта, разрабатываемая в рамках аналоговой схемотехники, требует иной подход к разработке и отдельный математический аппарат, что приводит к высокой сложности и дороговизне оборудования. В то же время цифровые устройства обладают простотой разработки, реализации и производства, соответственно и низкую стоимость, имеют гибкость при масштабировании и корректировке (IP-блоки). Также широкому распространению цифровых устройств способствует двоичное представление информации, которая позволяет выполнять ряд обработок информационного потока (пайплайн) с целью минимизации неизбежного влияния шумов, а также потери информации с помощью методов обнаружения и корректировок ошибок.

В рамках данной курсовой работы проводится анализ и обработка модельных радиосигналов и помех на примере частотной модуляции. Целью работы является практическое применение созданных методов классов analysis, model, processing, in\_out. Задачи, решаемые в рамках данной работы, можно разделить на два пункта: синтез сигнала с ЧМ, приближенный к реальному (с искажениями), и его обработка. Предполагается, что, используя реализованные методы цифровой обработки, можно восстановить исходный сигнал.

# 1 Теоретические основы

## 1.1 Понятие сигнала

Основной задачей системы связи является передача информации по каналу связи. Все полезные сигналы сообщений появляются случайным образом, т. е. приемник не знает заранее, какой из возможных символов сообщений будет передан. Кроме того, вследствие различных электрических процессов возникают шумы, которые сопровождают информационные сигналы [1]. По количеству физических переменных, характеризующих состояние источника информации, сигналы делят на одномерные, двумерные и трехмерные. Примерами одномерных сигналов являются ток в цепи микрофона, напряжение на выходе датчика температуры и др. Двумерные сигналы используются в случае, когда состояние источника определяется двумя переменными (координатами) одновременно. Двумерные сигналы, например, используются при обработке изображений для того, чтобы задать координаты точки на плоскости [4]. При помощи трехмерных сигналов можно определить положение пространственных объектов или описать цветные изображения. Передача информации заключается в переносе ее на расстояние при помощи сигналов различной физической природы [1].

## 1.2 Двоичное представление сигнала

В основе современной теории связи лежит понятие цифрового или двоичного сигнала. В данной интерпретации сигнал представляет собой набор нулей и единиц (бит), последовательность бит представляет собой данные, которые при декодировании становятся информацией. Данный подход к представлению информации имеет свои преимущества и недостатки. Так цифровые сигналы менее подвержены искажению и интерференции, чем аналоговые, поскольку в каждый момент времени возможно только одно состояние: включено либо выключено (высокий/низкий уровень). Наличие двух состояний облегчает восстановление сигнала и, следовательно, предотвращает накопление в процессе передачи шумов или других возмущений. В совокупности с применением процедур выявления и коррекции ошибок делают возможным высокую точность сигнала даже в сильно зашумленной среде. Наконец архитектура компьютеров, построенных на бинарной логике, а также компьютерных сетей, связывающих их, требует применения именно цифровых сигналов. Из недостатков можно выделить неизбежные потери при оцифровке сигнала, однако современные методы цифро-аналогового преобразования позволяют восстанавливать исходный сигнал с высокой точностью. Также цифровые системы требуют более интенсивной обработки, чем аналоговые и выделения значительной части ресурсов для синхронизации на различных уровнях [1].

Процесс оцифровки аналогового сигнала называется дискретизацией или иными словами преобразование функции непрерывного времени в функцию дискретного времени, представляемую совокупностью величин, именуемых координатами, по значениям которых начальная непрерывная функция может быть восстановлена с заданной точностью. Роль координат нередко выполняют мгновенные значения функции, отсчитанные в определённые моменты времени (рисунок 1).

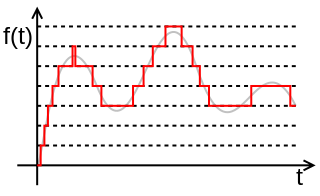


Рисунок 1 ‒ Дискретизация аналогового сигнала

Из рисунка 1 видно, что каждому кванту времени исходному сигналу соответствует значение дискретной функции f(t), разрядность АЦП определяет близость этой функции к исходной, значение кванта времени определяет «гладкость» дискретного сигнала. Варьируя эти параметры, можно изменять соотношение качества цифрового сигнала и затрат памяти и ресурсов на его обработку и хранение [1]].

## 1.3 Понятие модуляции

Модуляция ‒ это процесс, посредством которого символы сообщений или преобразуются в сигналы, совместимые с требованиями, налагаемыми каналом передачи данных. Существует несколько типов модуляции: амплитудная, частотная и фазовая. Передаваемая информация заложена в управляющем (модулирующем) сигнале, а роль переносчика информации выполняет высокочастотное колебание, называемое несущим. Модуляция, таким образом, представляет собой процесс наложения информационного колебания на известную несущую. Для этого изменяются соответственно амплитуду, частоту или фазу сигнала несущей. В результате модуляции спектр низкочастотного управляющего сигнала переносится в область высоких частот. Это позволяет при организации вещания настроить функционирование всех приемопередающих устройств на разных частотах с тем, чтобы они не мешали друг другу. Кроме того, модуляция может использоваться для перемещения сигнала в диапазон частот, в котором легко удовлетворяются специфические конструктивные требования, например, относящиеся к фильтрации и усилению. В частности, перенос рабочей частоты в область промежуточной [2, 3].

Помимо модуляции существует и обратный процесс, называемый демодуляцией, необходимый для получения исходного сигнала. Устройства генерации и приема сигналов содержат в себе модуляторы и демодуляторы соответственно.

## 1.4 ЧМ модуляция

При частотной модуляции (ЧМ) несущий сигнал является более высокочастотным по отношению к информационному сигналу, а амплитуда частотно-модулированного сигнала является неизменной [2]. Графики исходных и модулированного сигналов приведены на рисунке 2.

Изображение выглядит как линия, текст, График, Шрифт

Автоматически созданное описание

Рисунок 2 ‒ а) несущая частота, б) модулирующий сигнал, в) результат модуляции

Для описания ЧМ сигнала, где модулирующий сигнал меняется по гармоническому закону, используется формула (1):

, (1)

Где ‒ значение модулированного сигнала

‒ амплитуда

‒ начальная фаза

‒ индекс модуляции

Для данного спектр будет иметь следующий вид (рисунок 3) [3]:

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рисунок 3 ‒ Спектр ЧМ сигнала

При изменении индекса модуляции будет меняться вид спектра (рисунок 4).

Изображение выглядит как диаграмма, Технический чертеж, линия, схематичный

Автоматически созданное описание

Рисунок 4 ‒ Спектры сигналов с ЧМ при индексе модуляции

Таким образом, при малом индексе частотной модуляции спектр ЧМ-колебания состоит практически из трех гармоник: центральной — на несущей частоте и двух, расположенных симметрично относительно несущей частоты: на нижней боковой частоте , и верхней боковой .

Аппаратная реализация приемника ЧМ сигнала на примере структурной схемы супергетеродинного приемника приведена на рисунке 5. Она содержит следующие устройства (слева направо): антенна, блок приселектора, содержащий входную цепь (колебательный контур) для фильтрации сигнала и усилитель радиочастоты (УРЧ) для повышения амплитуды сигнала, блок промежуточной частоты (ПЧ) со смесителем (СМ) и гетеродином (Г), которые переносят спектр с области высоких частот в область низких для простоты дальнейшей работы с сигналом, усилитель промежуточной частоты (УПЧ), аналого-цифровой преобразователь (АЦП) и последующий тракт цифровой обработки сигнала (ЦОС) [2].

Изображение выглядит как диаграмма, План, Технический чертеж, линия

Автоматически созданное описание

Рисунок 5 ‒ Структурная схема супергетеродинного приемника [2]

## 1.5 ЛЧМ модуляция

Одной из вариаций ЧМ является линейная частотная модуляция (ЛЧМ). В данном случае частота несущей меняется по линейному закону общий вид и график которого изображен на рисунке 6. На рисунке 7 приведен график пакета LoRa, использующего ЛЧМ. Из него видно, что биты кодируются за счет резкого изменения частоты сигнала при его линейном нарастании.

Изображение выглядит как диаграмма, линия, Параллельный

Автоматически созданное описание

Рисунок 6 ‒ а) Общий вид сигнала б) График изменения частоты

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 7 ‒ Информационный пакет ЛЧМ сигнала

К преимуществам ЛЧМ сигнала можно отнести малую восприимчивость к помехам, больший радиус действия и простоту реализации. Получил широкое распространение в радиолокаторах.

## 1.6 Цифровая обработка сигнала и преобразование Фурье

Цифровая обработка сигнала (ЦОС) представляет собой набор способов обработки сигналов на основе численных методов с использованием цифровой вычислительной техники и производится с целью выделения полезного сигнала на фоне шумов и помех, а также его обработки под определенные задачи. Рассмотрим тракт ЦОС на примере технологии LoRa (рисунок 8) [5].

Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 ‒ Тракт цифровой обработки данных LoRa

На рисунке 8 сверху представлена передающая часть тракта, снизу – принимающая. Разберем тракт на примере передающей части. Она состоит из следующих блоков (справа налево) [6]:

1. Прием сигнала антенной

2. Обнаружение начала сигнала и синхронизация (Preamble detection, Synchronization)

3. Демодуляция (Demodulation)

4. Декодирование кода Грея (Gray demapping)

5. Упорядочивание данных (Deinterleaving)

6. Декодирование данных (Dewhitening)

7. Коррекция ошибок (Hamming decoding)

На выходе тракта получается двоичный поток данных, когда на вход поступает аналоговый сигнал.

При анализ линейных систем применяется преобразование функций токов и напряжений из временной области в частотную, то есть преобразования Фурье и Лапласа, позволяющие перевести описание системы с помощью дифференциальных уравнений во временной области к описанию системы с помощью линейных алгебраических уравнений в частотной области. Поскольку решение алгебраических уравнений существенно проще с вычислительной точки зрения, такой подход широко используется на практике. Преобразования Фурье 𝑓(𝑡) определяется соотношением [4]:

*,* (2)

Метод Фурье применяется в задачах предсказания отклика системы, определения передаточной системы, оценки интерпретации результатов тестов. Частотными спектральными характеристиками можно описать как сигналы, так и электрические схемы.

## 1.7 Алгоритм работы

Алгоритм работы программы приведен на рисунке 9. На первом этапе происходит инициализация программы, задаются входные параметры и точка входа основной функции, далее просчитываются значения для функции эталонного сигнала с частотной модуляцией (гармонический модулирующий сигнал), после искусственного искажения сигнала строится его график и спектр для последующего сравнения. Далее алгоритмами устранения шумов и пиков сигнал приводится к первоначальному виду, с последующим построением.



Рисунок 9 ‒ Алгоритм программы

# 2 Практическая часть

## 2.1 Генерация ЧМ сигнала

Для написания программы использовался язык Python и среда разработки PyCharm. Для построения графиков используется сторонняя библиотека matplotlib, также используется некоторые математические конструкции из библиотеки Numpy, например массивы. Из встроенных библиотек применяется time для генерации зерна (seed) в генераторе шума. Классы model, analysis, processing и in\_out содержат методы для построения модели, его анализа и обработки соответственно (приложения Б-Д). Код программы приведен в приложении А. Функция main начинается с создания экземпляров классов и задания значений переменных (листинг 1):

Листинг 1 ‒ Задание переменных

N = 10 \*\* 3 # Число точек сигнала  
A = 30 # Амплитуда сигнала  
f0 = 10 # Частота несущей  
df = 1 # Девиация частоты  
f1 = 2 # Частота модулирующего сигнала  
del\_t = 0.01 # Временной шаг для сигнала  
dt = 0.001 # Постоянная времени для fft

Следующий блок кода генерирует сигнал с заданными характеристиками, а также искажениями в виде шума и всплесков (листинг 2).

Листинг 2 ‒ Формирование сигнала

signal = new\_model.frequencyModulation(N, A, f0, df, f1, del\_t)  
noise = new\_model.noise(N, 5)  
signal = new\_model.addModel(signal, noise, N)  
signal = new\_model.spikes(signal, N, 5, 50, time.time())

На этом месте стоит уточнить, что в реальной схемотехнике мы чаще всего имеем дело именно с этими видами искажений, и реже с интерференцией волн, что решается простыми аппаратными фильтрами, например колебательным контуром. Наличие тренда, либо сдвига части сигнала встречается крайне редко и влияют только на спектр, в то время как преселектор с промежуточным трактом нивелируют данные дефекты сигнала. Наконец большинство детекторов нечувствительны к данным искажениям, а скорее к частотной стабильности гетеродина.

На рисунках 10 и 11 представлены графики и спектры сигнала, видно, что спектр имеет центральную частоту, равную частоте несущей 10 Гц и боковые частоты 8 и 12 Гц. Искажения сигнала не сильно влияют на его спектр, а соответственно и на переданную информацию. Это хорошо видно на рисунке 12.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 10 ‒ Исходный сигнал с шумом и его спектр

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 11 ‒ Исходный сигнал с шумом и всплесками и его спектр

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 12 – Сильно искажённый сигнал и его спектр

## 2.2 Обработка ЧМ сигнала

Как было упомянуто в разделе 1.4, для выделения исходного сигнала используются ЧМ детекторы, для программной реализации достаточно использовать фильтр низких частот и методы для подавления шума/сдвигов/спайков. Код данной части программы представлен в листинге 3. На рисунках приведены сигналы с шумом (рисунок 13) и сильным шумом (рисунок 12)

Листинг 3 ‒ Обработка ЧМ сигнала

signal = Proccessing.antiSpike(signal)  
signal = Proccessing.antiNoise(signal)  
signal = Proccessing.antiShift(signal)  
  
low\_pass\_filter = Proccessing.lpf\_reverse(Proccessing.lpf(3, m, dt/2))  
signal = new\_model.convolModel(signal, low\_pass\_filter, N, M)

Для фильтра низких частот были выбраны частоты 2, 3 и 6 Гц, результаты после прохождения сигнала через данные значения приведены на рисунках 15, 16, 17.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описаниеРисунок 13 ‒ Результат обработки сигнала с шумом

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 14 ‒ Результат обработки сигнала с шумом, спайками и сдвигом

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Рисунок 15 ‒ Результат работы ФНЧ на 2 Гц

Изображение выглядит как текст, диаграмма, График, линия

Автоматически созданное описание

Рисунок 16 ‒ Результат работы ФНЧ на 5 Гц

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Рисунок 17 ‒ Результат работы ФНЧ на 5 Гц

Из рисунков 15, 16, 17 видно, что лучший сигнал с точки зрения спектра дает ФНЧ на 5 Гц, поэтому рекомендуется брать границу среза чуть выше рабочей частоты, чтобы не обрезать частоты ниже и не допустить просачивания высоких частот.

# ЗАКЛЮЧЕНИЕ

В результате работы по анализу и обработке сигналов с частотной модуляцией был освящен широкий спектр проблем, связанных с данной темой. В современной теории связи ключевыми являются такие понятия, как цифровое представление сигнала, модуляция, цифровая обработка сигнала. «Цифра» в радиосвязи позволяет упростить программную и аппаратную части, а также является более помехозащищенной и дает возможность отслеживать и корректировать ошибки. Модуляция также служит для повышения помехоустойчивости канала. В частности, частотная модуляция имеет отличные показатели устойчивости к помехам среды передачи, в сравнении с амплитудной, при сравнительно небольшом усложнении приемо-передатчика. Основными характеристиками ЧМ сигнала является частоты рабочей и несущей частот и девиация частоты.

Из полученных в ходе работы моделей можно сделать вывод, что изменение амплитуды сигнала не сказывается на его информационных свойствах, что является ключевым параметром. Именно поэтому ЧМ почти полностью вытеснила АМ. Ко всему прочему ЧМ приемо-передатчики обладают большим КПД, в сравнении с АМ. Для получения рабочей частоты достаточно использовать ФНЧ с частотой среза на несколько Гц большей. При работе с частотной модуляцией рекомендуется, чтобы девиация частоты была больше 2, а также должны отсутствовать фазовые сдвиги в сигнале, хотя они и возникают крайне редко в природе, однако очень трудно исправимы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Скляр Б. «Цифровая связь. Теоретические основы и практическое применение» ‒ Вильямс, 2017 г.1100 стр.
2. А.С.Коротков. Устройства приема и обработки сигналов: микроэлектронные высокочастотные устройства радиоприемников систем связи, СПб: Изд-во Политехн. Унта, 2010, 223 с.
3. О.В.Головин. Устройства генерирования, формирования, приема и обработки сигналов. Учебное пособие для вузов, М.: Горячая линия – Телеком, 2014, 783 с
4. Лекции Белых И.Н. по дисциплине «Методы обработки экспериментальных данных».
5. Wireless communication method ‒ US20160094269A1 United States, 2014
6. A Low-Complexity LoRa Synchronization – URL: <https://arxiv.org/pdf/1912.11344.pdf> (дата обращения: 20.01.24)

# ПРИЛОЖЕНИЕ А

# Код программы

import matplotlib.pyplot as plt  
import numpy as np  
import time  
from classes.model import Model  
from classes.analysis import Analysis  
from classes.proccessing import Proccessing  
  
plt.rcParams["figure.figsize"] = [20, 7.5]  
plt.rcParams["figure.autolayout"] = True  
  
  
def main():  
 new\_model = Model()  
 new\_analysis = Analysis()  
  
 N = 10 \*\* 3  
 A = 30  
 f0 = 10  
 df = 1  
 f1 = 2  
 del\_t = 0.01  
 dt = 0.001  
  
 harm = new\_model.harm(N, A, f1, del\_t)  
 signal = new\_model.frequencyModulation(N, A, f0, df, f1, del\_t)  
 noise = new\_model.noise(N, 5)  
 signal = new\_model.addModel(signal, noise, N)  
 signal = new\_model.spikes(signal, N, 5, 50, time.time())  
 signal\_source = Proccessing.antiSpike(signal)  
 signal\_source = Proccessing.antiSpike(signal)  
 signal\_fourier = new\_analysis.Fourier(signal, N)  
 new\_X\_n = new\_analysis.spectrFourier([i for i in range(N)], N, dt)  
 k = np.arange(0, N, 1)  
  
 fig, ax = plt.subplots(nrows=2, ncols=1)  
 fig.suptitle("Курсовая работа", fontsize=15)  
 # ax[0].plot(k, signal, 'r', k, harm, 'g')  
 ax[0].plot(k, signal)  
 ax[1].plot(new\_X\_n, signal\_fourier)  
 ax[1].set\_xlim([0, 1 / (dt \* 2)])  
 ax[0].set\_title("ЧМ сигнал")  
 ax[1].set\_title("Спектр ЧМ сигнала")  
 plt.show()  
  
main()

# ПРИЛОЖЕНИЕ Б

# model.py

import math  
import numpy as np  
import random  
  
class Model:  
  
 def trend\_linear(self, N, a, b):  
 t = np.arange(0, N, 1)  
 x = a \* t + b  
 return x  
  
 def exp(self, N, a, b):  
 t = np.arange(0, N, 1)  
 x = b \* np.exp(a \* t)  
 return x  
  
 def noise(self, N, R):  
 noise = np.random.randint(R+1, size=(N)) - 10  
 return noise  
  
 @staticmethod  
 def lin\_rand\_arr\_flxd(seed, size):  
 m = 32768  
 a = 23  
 b = 12345  
 if size == 1:  
 return math.ceil(math.fmod(a \* math.ceil(seed) + b, m))  
 r = [0 for i in range(size+1)]  
 r[0] = math.ceil(seed)  
 for i in range(1, size+1):  
 r[i] = math.ceil(math.fmod((a \* r[i - 1] + b), m))  
 return r[1:size+1]  
  
 def my\_noise(self, seed, N, R):  
 noise = self.lin\_rand\_arr\_flxd(seed, N)  
 my\_noise = [i for i in range(N)]  
 for i in my\_noise:  
 my\_noise[i] = (((noise[i] - min(noise))/(max(noise)-min(noise)))-0.5)\*2\*R  
 return my\_noise  
  
 def shift(self, inData, C, N1, N2):  
 k = np.arange(N1, N2, 1)  
 for i in range(N2-N1):  
 inData[k[i]] += C  
 return inData  
  
   
 def spikes(self, inData, N, M, R, seed):  
 Rs = 0.1 \* R  
 sampl = self.my\_noise(seed, M, Rs)  
 signs = np.random.randint(2, size=M)  
 for i in range(M):  
 if signs[i] == 0:  
 signs[i] = -1  
 sampl[i] += Rs \* R \* signs[i]  
 t\_1 = random.sample(list(range(1, N + 1)), M)  
 for i in range(M):  
 inData[t\_1[i]] += sampl[i]  
 return inData  
  
 def harm(self, N, A, f, dt):  
 k = np.arange(0, N, 1)  
 x = A \* np.sin(2 \* math.pi \* f \* dt \* k)  
 return x  
  
 def polyHarm(self, N, A0, f0, A1, f1, A2, f2, dt):  
 k = np.arange(0, N, 1)  
 x0 = A0 \* np.sin(2 \* math.pi \* f0 \* dt \* k)  
 x1 = A1 \* np.sin(2 \* math.pi \* f1 \* dt \* k)  
 x2 = A2 \* np.sin(2 \* math.pi \* f2 \* dt \* k)  
 x = x0 + x1 + x2  
 return x  
  
 def frequencyModulation(self, N, A, f0, df, f1, dt):  
 k = np.arange(0, N, 1)  
 return A \* np.sin(f0 \* dt \* k + df \* np.cos(f1 \* dt \* k))  
  
 def windowing(self, data, N, L):  
 for i in range(N-L, N):  
 data[i] = 0  
 return data  
  
 def addModel(self, data1, data2, N):  
 addResult = []  
 for i in range(N):  
 addResult.append(data2[i] + data1[i])  
 return addResult  
  
 def multModel(self, data1, data2, N):  
 multResult = []  
 for i in range(N):  
 multResult.append(data2[i] \* data1[i])  
 return multResult  
  
 def convolModel(self, x, h, N, M):  
 out\_data = []  
 for i in range(N):  
 y = 0  
 for j in range(M):  
 y += x[i - j] \* h[j]  
 out\_data.append(y)  
 return out\_data

# ПРИЛОЖЕНИЕ В

# analysis.py

import numpy as np  
import math  
  
class Analysis:  
  
 def min(self, data):  
 return min(data)  
  
 def max(self, data):  
 return max(data)  
  
 def avg(self, data):  
 return sum(data) / len(data)  
  
 def covariance(self, data, N):  
 avg\_x = self.avg(data)  
 out\_data = []  
 for l in range(N):  
 R\_xx = 0  
 for k in range(0, N - l):  
 R\_xx += (data[k] - avg\_x) \* (data[k + l] - avg\_x)  
 R\_xx = R\_xx / N  
 out\_data.append(R\_xx)  
 return out\_data  
  
 def hist(self, data, N, M):  
 hist = dict()  
 x\_min = min(data)  
 x\_max = max(data)  
 step = (x\_max - x\_min) / M  
 for i in range(M):  
 left\_border = x\_min + i \* step  
 right\_border = left\_border + step  
 count = 0  
 for j in range(N):  
 if left\_border <= data[j] <= right\_border:  
 count += 1  
 hist[left\_border] = count  
 return hist  
  
 def acf(self, data, N):  
 covariance = self.covariance(data, N)  
 max\_R\_xx = self.max(covariance)  
 out\_data = []  
 for l in range(N):  
 out\_data.append(covariance[l] / max\_R\_xx)  
 return out\_data  
  
 def ccf(self, dataX, dataY, N):  
 avg\_x = self.avg(dataX)  
 avg\_y = self.avg(dataY)  
 out\_data = []  
 for l in range(N):  
 R\_xy = 0  
 for k in range(0, N - l):  
 R\_xy += (dataX[k] - avg\_x) \* (dataY[k + l] - avg\_y)  
 R\_xy = R\_xy / N  
 out\_data.append(R\_xy)  
 return out\_data  
  
 # Расчет прямого преобразования Фурье  
 def Fourier(self, data, N):  
 out\_data = []  
 for i in range(N):  
 Re\_Xn = 0  
 Im\_Xn = 0  
 for k in range(N):  
 Re\_Xn += data[k] \* np.cos(2 \* math.pi \* i \* k / N)  
 Im\_Xn += data[k] \* np.sin(2 \* math.pi \* i \* k / N)  
 Re\_Xn = Re\_Xn / N  
 Im\_Xn = Im\_Xn / N  
 Xn = np.sqrt((Re\_Xn \*\* 2) + (Im\_Xn \*\* 2))  
 out\_data.append(Xn)  
 return out\_data  
  
 # Амплитудный спектр Фурье  
 def spectrFourier(self, Xn, N, dt):  
 out\_data = []  
 f\_border = 1 / (2 \* dt)  
 df = 2 \* f\_border / N  
 for i in range(N):  
 out\_data.append(Xn[i] \* df)  
 return out\_data

# ПРИЛОЖЕНИЕ Г

# proccessing.py

import numpy as np  
import math  
  
class Proccessing:  
  
 @staticmethod  
 def antiShift(data):  
 out\_data = []  
 avg = sum(data) / len(data)  
 for i in range(len(data)):  
 out\_data.append(data[i] - avg)  
 return out\_data  
  
 @staticmethod  
 def antiSpike(data):  
 out\_data = data  
 for i in range(1, len(out\_data)-1):  
 if math.fabs(out\_data[i]\*5) > math.fabs(out\_data[i+1]):  
 out\_data[i] = (out\_data[i-1]+out\_data[i+1])/2  
 return out\_data  
  
 @staticmethod  
 def antiTrendLinear(data, N):  
 out\_data = []  
 for i in range(N - 1):  
 out\_data.append(data[i + 1] - data[i])  
 out\_data.append(out\_data[N - 2])  
 return out\_data  
  
 @staticmethod  
 def antiTrendNonLinear(data, N, W):  
 out\_data = []  
 for i in range(N - W):  
 x\_n = 0  
 for k in range(W):  
 x\_n += data[i + k]  
 x\_n = x\_n / W  
 out\_data.append(x\_n)  
 # out\_data.append(out\_data[N - W - 2])  
 return out\_data  
  
 @staticmethod  
 def antiNoise(data, N, M):  
 out\_data = []  
 for i in range(N):  
 avg = 0  
 for j in range(M):  
 avg += data[j]  
 avg = avg / M  
 out\_data.append(avg)  
 return out\_data  
  
  
 @staticmethod  
 def lpf(fc, m, dt):  
 d = [0.35577019, 0.2436983, 0.07211497, 0.00630165]  
 fact = fc \* dt  
 lpw = []  
 lpw.append(fact)  
 arg = fact \* math.pi  
 for i in range(1, m + 1):  
 lpw.append(np.sin(arg \* i) / (math.pi \* i))  
 lpw[m] = lpw[m] / 2  
 sumg = lpw[0]  
 for i in range(1, m + 1):  
 sum = d[0]  
 arg = math.pi \* i / m  
 for k in range(1, 4):  
 sum += 2 \* d[k] \* np.cos(arg \* k)  
 lpw[i] = lpw[i] \* sum  
 sumg += 2 \* lpw[i]  
 for i in range(m + 1):  
 lpw[i] = lpw[i] / sumg  
 return lpw  
  
 @staticmethod  
 def lpf\_reverse(lpw):  
 return lpw[:0:-1] + lpw  
  
 @staticmethod  
 def hpf(fc, m, dt):  
 lpw = Proccessing.lpf\_reverse(Proccessing.lpf(fc, m, dt))  
 hpw = []  
 Loper = 2 \* m + 1  
 for k in range(Loper):  
 if k == m:  
 hpw.append(1 - lpw[k])  
 else:  
 hpw.append(- lpw[k])  
 return hpw  
  
 @staticmethod  
 def bpf(fc1, fc2, m, dt):  
 lpw1 = Proccessing.lpf\_reverse(Proccessing.lpf(fc1, m, dt))  
 lpw2 = Proccessing.lpf\_reverse(Proccessing.lpf(fc2, m, dt))  
 bpw = []  
 Loper = 2 \* m + 1  
 for k in range(Loper):  
 bpw.append(lpw2[k] - lpw1[k])  
 return bpw  
  
 @staticmethod  
 def bsf(fc1, fc2, m, dt):  
 bsw = []  
 lpw1 = Proccessing.lpf\_reverse(Proccessing.lpf(fc1, m, dt))  
 lpw2 = Proccessing.lpf\_reverse(Proccessing.lpf(fc2, m, dt))  
 Loper = 2 \* m + 1  
 for k in range(0, Loper):  
 if k == m:  
 bsw.append(1. + lpw1[k] - lpw2[k])  
 else:  
 bsw.append(lpw1[k] - lpw2[k])  
 return bsw  
  
 @staticmethod  
 def frequencyResponse(data, N):  
 out\_data = []  
 for i in range(N):  
 out\_data.append(data[i] \* N)  
 return out\_data

# ПРИЛОЖЕНИЕ Д

# in\_out.py

import wave  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.ticker as ticker  
import math  
  
types = {  
 1: np.int8,  
 2: np.int16,  
 4: np.int32  
}  
  
class IN\_OUT:  
  
 @staticmethod  
 def readWAV(path):  
 wav = wave.open(path, mode="r")  
 (nchannels, sampwidth, framerate, nframes, comptype, compname) = wav.getparams()  
 print(wav.getparams())  
 content = wav.readframes(nframes)  
 samples = np.fromstring(content, dtype=types[sampwidth])  
 return samples, nchannels, sampwidth, framerate, nframes  
  
 @staticmethod  
 def writeWAV(path, data, nchannels, sampwidth, framerate, nframes):  
 wav = wave.open(path, mode="w")  
 wav.setparams((nchannels, sampwidth, framerate, 0, 'NONE', 'not compressed'))  
 for i in range(nframes \* nchannels):  
 wav.writeframes(data[i])  
 wav.close()