МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка ПО для получения функциональной зависимости по данным эксперимента

Студент Е. В. Азанов
(И.О. Фамилия) (личная подпись)

Руководитель С. В. Митин
(ученая степень, звание, И.О. Фамилия)

Консультант И. Ю. Усатова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Аннотация

Название бакалаврской работы: «Разработка ПО для получения функциональной зависимости по данным эксперимента».

Выпускная работа состоит из введения, пяти разделов, заключения и списка литературы, включая зарубежные источники.

Ключевым вопросом бакалаврской работы является создание программного обеспечения для построения графика зависимости. Мы затрагиваем проблему структуризации исходных данных, а также проблему интерполяции этих данных.

Целью работы является разработка программы для создания таблицы итоговых значений после интерполяции функции, а также создания самого графика интерполированной функции.

Бакалаврская работа может быть разделена на следующие логически взаимосвязанные части: анализ существующих решений; постановку задачи; описание калибровки и сравнение алгоритмов, благодаря которым происходит фильтрация исходных данных; анализ методов интерполяции и построения графиков для наглядности сравнения; сравнение всех выбранных методов интерполяции.

В конце исследования предоставлен результат работы программы, вывод графика интерполяции, а также таблицы значений.

Подводя итоги, мы бы хотели подчеркнуть, что данная работа актуальна во всех сферах, где необходимо получение функциональной зависимости.

Abstract

The title of the graduation work is: « Development of software for obtaining functional dependence according to experiment data».

The graduation work consists of an introduction, four chapters, a conclusion and list of 25 references including 16 foreign sources.

The key issue of the thesis is the creation of software for plotting dependency graphs. We touch upon the problem of structuring the initial data, as well as the problem of interpolation of this data.

The aim of the work is developing a program to create a table of final values after function interpolation, as well as to create the graph of the interpolated function itself.

The graduation work may be divided into several logically connected parts, which are analysis of existing solutions; task setting; a description of the calibration and comparison of algorithms that filter the initial data; analysis of interpolation methods and plotting for clarity of comparison; comparison of all selected interpolation methods.

Finally, we present the result of the program performance, the output of the interpolation graph, as well as tables of values are provided.

In conclusion we'd like to stress this work is relevant in all areas where it is necessary to obtain a functional dependence.

Содержание

Введение

Методы обработки экспериментальных данных начали разрабатываться более двух веков тому назад в связи с необходимостью решения практических задач по агробиологии, медицине, экономике, социологии. Полученные при этом результаты составили фундамент такой научной дисциплины, как математическая статистика.

При проведении эксперимента с целью калибровки устройства нужно знать при каких входных параметрах какое будет выходное значение. Для того, чтобы это выяснить, нужно большое количество времени подавать данные, выписывать результат и вручную строить таблицу значений. На это тратится очень много времени, а также не исключена погрешность человеческого фактора.

Целью работы является получение функциональной зависимости перевода входящих значений в физическую величину. Данный вопрос возникает при калибровке электронных устройств, которые используются для измерения, измеряемые значения поступают с выхода аналого-цифрового преобразователя. Он характеризуется разрядностью 2^n, где n от 10 до 16 разрядов.

Практическое значение состоит в необходимости автоматизации этого процесса, что позволит снизить затраты человеко-часов на процесс калибровки, а также позволит уменьшить влияние человеческого фактора.

В процессе эксперимента происходит запись показаний аналого-цифрового преобразователя в файл. Также во второй файл записывается информация о введенных данных, что является вторым параметром функции. После этого происходит анализ данных и выделение узловых точек, которые характеризуют те физические значения, которые подавались на вход аналого-цифрового преобразователя, что позволяет, применяя методы интерполяции, получить функциональную зависимость перевода показания аналого-цифрового преобразователя в физическую величину в виде таблицы значений.

1 Постановка задачи

1.1 Анализ условных данных. Условия эксперимента.

Для корректного выполнения поставленной задачи, нужно задать начальные условия, на которые будет опираться алгоритм. Устройство измеряет физическую величину, например, вольтметр измеряет напряжение. Устройство измеряет данные 100 раз в секунду и на выходе дает файл со значениями в виде временной последовательности.

Задаваемые значения выдерживаются в течении определенного времени, так, чтобы в этой временной последовательности было не менее заданного количества точек (например, 100-200 значений). Таким образом формируем ступенчатую последовательность, которую записываем в файл со ступенчатыми значениям.

Так как при проведении эксперимента не исключена погрешность устройств, также нужно задать условие разности выходных значений, чтобы алгоритм мог отличить погрешность от отдельного параметра (например, шаг в 5 значений).

1.2 Анализ известных решений

Дальнейшее развитие программного обеспечения привело к созданию большого количества прикладных пакетов по статистике. Удобной универсальной вычислительной средой для решения задач обработки экспериментальных данных является табличный процессор Microsoft Excel [3].

1.3 Постановка задачи

Для выполнения поставленной задачи алгоритм получает на вход непрерывный ряд необработанных значений. В качестве первого шага программа

должна вывести график исходных значений, который наглядно показывает какие значения показывал АЦП при заданных условиях.

Теперь перейдем ко второму шагу, в нем программа должна обработать полученные данные. На выходе после этого шага должен выйти массив значений, который должен соответствовать второму набору исходных данных, который был определен заранее.

На третьем шаге алгоритм должен построить функциональную зависимость в виде графика, который наглядно показывает какое значение будет выведено при определенных заданных параметрах.

Отсюда вытекают 3 необходимые задачи:

– построение графика исходных значений;

– анализ и фильтрация данных, для выделения последовательности узловых точек;

– получение зависимости перевода показаний измерителя в физическую величину.

Для разработки программного модуля будет использован язык программирования Python. Данный язык больше всего подходит для работы с большим объемом данных. Он позволяет наглядно выводить их, а также имеет множество инструментов для удобной работы. Python также имеет низкий уровень вхождения, так как является высокоуровневым языком программирования и не имеет проблем с установкой.

Также стоит отметить, что по данному языку программирования существует большое количество руководств, которые помогут в решении поставленной задачи

Вывод по разделу 1

В результате данного раздела, были заданы условия для исходных данных, проведена постановка задачи, выделены пункты, необходимые для ее решения, а также выбрана среда, в которой будет реализована программа.

2 Фильтрация и структуризация данных

2.1. Визуализация исходных значений.

Для решения поставленной задачи, нужно наглядно понять, как выглядят данные, и с какими проблемами мы можем столкнуться, решая ее. Для этого нужно визуализировать входные данные путем построения графика. С такой задачей отлично справляется библиотека Matplotlib в python. Matplotlib предоставляет огромное количество различных настроек, которые можно использовать для того, чтобы придать графику вид, который вам нужен: цвет, толщина и тип линии, стиль линии и многое другое.

Рассмотрим реализацию первого алгоритма. Он необходим для визуализации работы эксперимента по выводу полученных данных. Это позволит отследить правильно ли прошел эксперимент, результаты которого будут обработаны в программе.

Реализация кода на python представлена на рисунке 1, на выходе имеем наглядную демонстрацию всех значений, которая показана на рисунке 2

```
import matplotlib.pyplot as plt

file = []
with open('data.txt') as f:
    file = f.read().splitlines()
num = len(file)

for i in range(num):
    index = file[i].find('\t')
    file[i] = file[i][index+1:]
    index = file[i].find('\t')
    file[i] = file[i][:index]
    file[i] = int(file[i])

x_list = list(range(num))
plt.plot(x_list, file)
plt.show()
```

Рисунок 1 – Реализация вывода исходных данных

Полученный график позволяет наглядно увидеть проблемы, которые нужно решить при написании алгоритма фильтрации исходных значений. Во-первых, программа должна выделить ступеньки, который четко видны на графике, и выделить в них средние значения, так как исходные данные имеют вариабельность из-за погрешности АЦП. Также алгоритм должен игнорировать случайные выбросы, чтобы не разделять участки на 2 части и не вносить изменения в среднее значение текущей ступеньки.

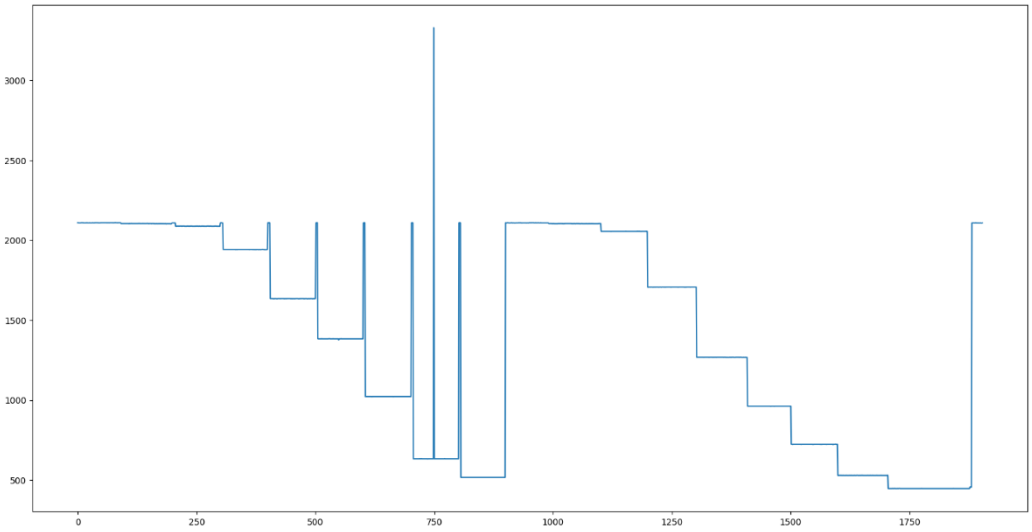

Рисунок 2 – Визуализация исходных данных.

2.2. Анализ и фильтрация данных

Для построения графика программе на вход нужны конкретные точки. Исходным набором данных является выборка с большим количеством данных, что не подходит для построения графика. Весь набор данных нужно отфильтровать, чтобы получились готовые для следующей части программы значения. На рисунке 2 четко видны “ступеньки”, их и нужно выделить. Проблемой решаемой задачи является то, что данные в этих “ступеньках” варьируются в определенных диапазонах.

2.2.1 Частотный анализ

Для решения поставленной задачи можно воспользоваться частотным анализом. Он позволяет оценить количество слов в тексте, в нашем же случае он поможет нам определить конкретное количество каждого из чисел в исходном наборе данных. Частотный анализ заключается в подсчете появления каждой буквы в тексте. Он основан на том факте, что в любом фрагменте текста определенные буквы и сочетания букв встречаются с разной частотой [9].

Частотность – термин лексикостатистики, который употребляется для обозначения наиболее часто употребляемых слов. Также определяется частотность для букв. Частотный анализ текста используется для выявления наиболее часто используемых слов того или иного языка. В словарях частотность слов может отражаться пометками - употребительное, малоупотребительное [23].

Частотный анализ является одним из сравнительно простых методов обработки текста на естественном языке (NLP). Его результатом является список слов, наиболее часто встречающихся в тексте. Частотный анализ также позволяет получить представление о тематике и основных понятиях текста. Визуализировать его результаты удобно в виде «облака слов». Эта диаграмма содержит слова, размер шрифта которых отражает их популярность в тексте [8].

Сама программа на питоне реализуется следующим образом, фрагмент кода предоставлен на рисунке 3.

```
import re
from collections import Counter
with open('text.txt') as infile,\
        open('result.txt', 'w') as out:
    txt = infile.read()
    s = re.sub(r'[^\w\s]',' ',txt).split()
    words = [(v,k) for v, k in Counter(s).items()]
    words.sort(key = lambda x: (-x[1], x[0]))
    for w in words:
        print(*w, file = out)
```

Рисунок 3 – Реализация частотного анализа.

По итогу работы программы мы имеем на выходе файл, в котором нам показано количество каждого числа, его работа продемонстрирована на рисунке 4.

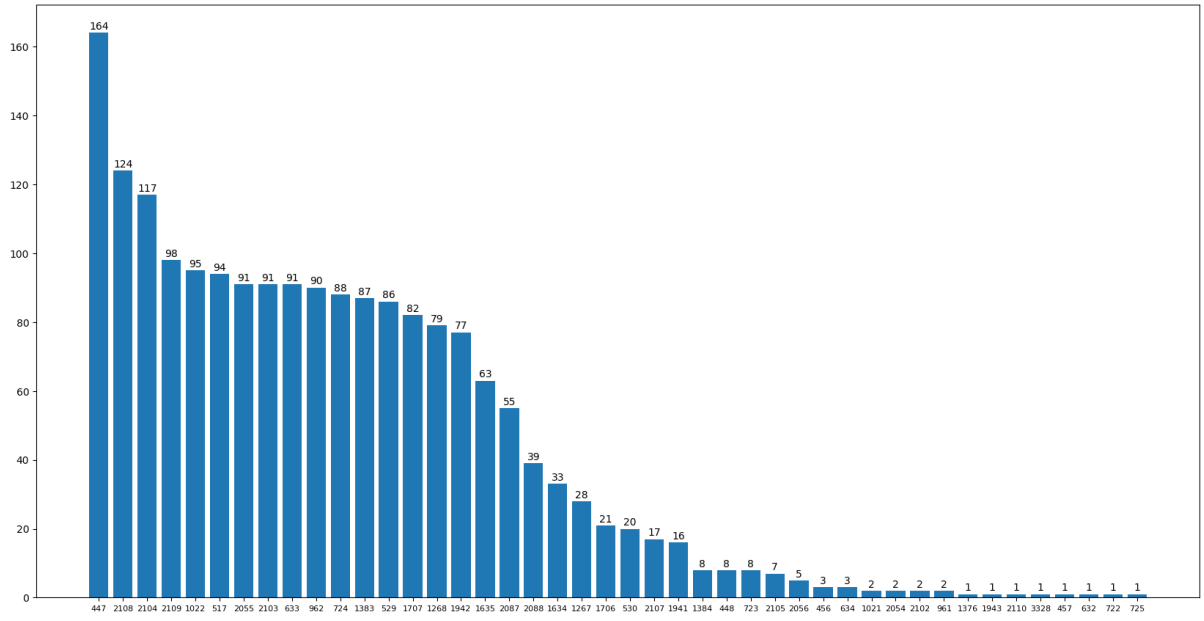

Рисунок 4 – Частотный анализ.

Проанализировав результат выполнения программы, можно сделать вывод что данный метод не подходит для обработки исходных значений. Наглядно видны следующие проблемы. Во-первых, пропадает возможность отследить порядок вхождения данных, это может стать проблемой при связке со вторым параметром функции. Во-вторых, из-за погрешности датчиков, не всегда можно правильно отследить “выбросы”, так как нельзя точно сказать к какому отрезку принадлежит конкретное значение. В-третьих, сложно определить какие данные объединять, так как непонятно является это погрешностью, или другой “ступенькой”.

2.2.2. Фильтрация данных

Для решения поставленной задачи нужно написать программу, которая будет проходить через все данные, выделяя “ступеньки” постепенно.

Для корректной работы программы нужно задать начальные условия, на которые она будет опираться. Во-первых, как видно из графика, средней длиной

ступеньки является 100 значений. Минимальное значение здесь можно смело брать 50, так как сами промежутки сильно больше этого значения.

Дальше нам нужно выделить переменную, предназначенную для вариабельности промежутков. Для определения этого значения нужно приблизить график в самом начале, так как там видна самая минимальная разница среди всех промежутков, которая видна на рисунке 5. На нем хорошо видно, что погрешность исходных данных равна ±1. Также мы видим, что минимальная разница между ступеньками в данном эксперименте равна 5 единицам. Отталкиваясь от этого, оптимальным вариантом будет принять значение переменной погрешности за 3 единицы.

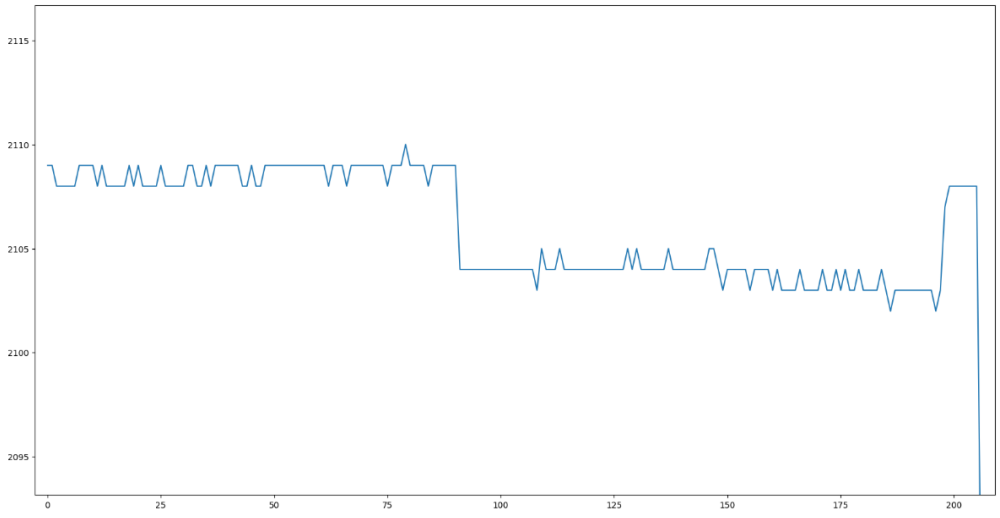

Рисунок 5 – Разница промежутков.

Последней условной переменной надо задать значение для игнорирования случайных выбросов. Случайные выбросы представлены на рисунке 6 на 6 и 8 ступеньке.

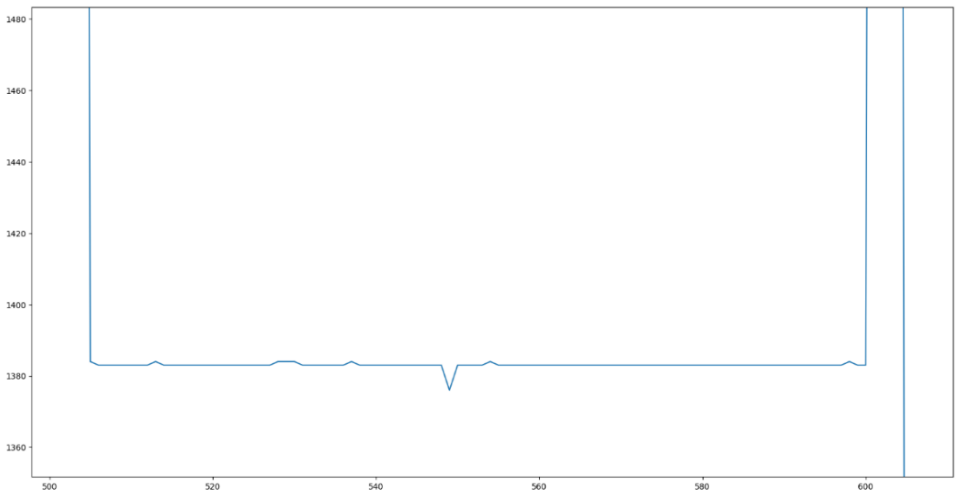

Рисунок 6 – Случайный выброс на 6 ступеньке.

Данные погрешности помешают получить среднее значение искомого промежутка, так как разделят его на 2 части, и могут внести небольшую погрешность. Так как эти выбросы занимают не больше 3 значений, возьмем это число для данной переменной. В результате мы имеем следующие заданные переменные:

- k=3 – переменная для разброса;
- n=50 – переменная для минимального количества;
- x=3 - переменная для выбросов;
- x1=0 – переменная счетчик для выбросов.

Имея заданные условные переменные, составить программный код не составляет больших трудностей, суть его работы заключается в прохождение по исходным данным и проверке на соблюдение заданных условий, код показан на рисунке 7.

```
for i in range(num):
    if ((abs(file[i]-result) > k) and x1>x):
        if (len(b) < n):
            b=[];
            first=last;
            i-=2;
            result=file[i];
            x1=0;
        else:
            a.append(round(sum(b) / len(b)));
            first=last;
            b=[];
            i-=2;
            result=file[i];
            x1=0;
    elif (abs(file[i]-result) > k):
        x1+=1;
    else:
        b.append(file[i]);
        result = (sum(b) / len(b));
        last=i;
```

Рисунок 7 – Реализация структуризации данных.

Переменная a является списком, который содержит итоговый массив значений, список b содержит промежуточный массив, среднее значение которого добавляется в список a. В итоге мы получаем пригодный для построения графика массив, с отфильтрованными значениями.

Вывод по разделу 2

Анализ исходных данных позволил определится со способом их структуризации. После проведенных экспериментов выяснилось, что частотный анализ не подходит для решения поставленной задачи. Было принято решение написать свой алгоритм, который бы полностью удовлетворял всем условиям задачи. Итогом второго раздела стало написание алгоритма структуризации данных.

3 Методы аппроксимации

3.1. Общее определение аппроксимации

Аппроксимация (от латинского "approximate" - "приближаться") - приближенное выражение каких-либо математических объектов (например, чисел или функций). Аппроксимацией является ближайшая оценка, которую вы можете получить, не имея точного размера или измерения чего-либо. Это приблизительная цифра. Для нахождения используются похожие, но более простые, более удобные в пользовании объекты. В научных исследованиях аппроксимация применяется для описания, анализа, обобщения и дальнейшего использования эмпирических результатов [4].

Между величинами, которые принадлежат одним исходным данным, может существовать функциональная зависимость, когда одному значению X соответствует определенное значение Y, и менее точная (корреляционная) связь, когда одному конкретному значению аргумента соответствует приближенное значение или некоторое множество значений функции, в той или иной степени близких друг к другу. В нашем случае, для обработки результатов наблюдений эксперимента, мы столкнулись с первым вариантом. При проведении опыта для получения количественной зависимости различных показателей, в основном имеется вариабельность этих показателей. Поэтому при выполнении подобного рода научно-исследовательской работы возникает необходимость выявления подлинного характера зависимости изучаемых показателей, этой или иной степени замаскированных неучтенностью вариабельности значений. С этой проблемой помогает справится метод аппроксимации, который является приближенным описанием корреляционной зависимости переменных подходящим уравнением функциональной зависимости, передающим основную тенденцию зависимости (или ее "тренд") [1].

При выборе метода аппроксимации следует исходить из самой задачи исследования. Если нет необходимости в точно полученной зависимости, а

нужна лишь приблизительная, можно использовать простое уравнение, в обратном случае – наоборот. При описании зависимости эмпирически определенных значений можно добиться и гораздо большей точности, используя какое-либо более сложное, много параметрическое уравнение.

Существует два принципиально различных типа аппроксимации функций:

– Интерполяцией называется аппроксимирующая функция, значения которой точно совпадают с исходными значениями;

– Метод наименьших квадратов − аппроксимирующая функция F(x)может не совпадать ни с одним табличным значением $y_0, y_1, \dots, y_n$ функции $f(x)$, максимально приближаясь к ним в среднем [7].

Так как в решении данной задачи необходимо точное совпадение с исходными значениями, нужно воспользоваться интерполяцией.

3.2. Исходные значения

Для начала нужно импортировать библиотеки, которые будут использоваться в программе, они показаны на рисунке 8, а также загрузить данные, полученные из прошлой части программы.

```
Import numpy as np
Import scipy as sp
Import pandas as pd
Import matplotlib.pyplot as plt
From matplotlib.axes import Axes
Import seaborn as sns
```

Рисунок 8 – Импортированные библиотеки.

Разберемся для чего нужна каждая из импортированных библиотек.

NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python». Также данная библиотека предоставляет базовые методы

для манипуляции с большими массивами и матрицами, чем мы и будем пользоваться для хранения данных и работы с ними [17].

SciPy — это библиотека Python с открытым исходным кодом, предназначенная для решения научных и математических проблем. Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд. С помощью данной библиотеки мы будем проводить сравнение методов интерполяций [21].

Pandas — это библиотека Python для обработки и анализа структурированных данных, её название происходит от «panel data» («панельные данные»). Панельными данными называют информацию, полученную в результате исследований и структурированную в виде таблиц. Для работы с такими массивами данных и создан Pandas [19].

Библиотеку matplotlib мы рассматривали в прошлого раздела. Здесь она нам также понадобится для визуализации полученных функций.

Seaborn — это библиотека для создания статистических графиков на Python. Она основывается на matplotlib и тесно взаимодействует со структурами данных pandas. Она нам понадобится, так как в matplotlib нельзя использовать данные, созданные с помощью библиотеки pandas [22].

Для работы с данными библиотеками, для начала их нужно установить в среду Python, иначе он просто их не увидит и выдаст ошибку. Процесс установки библиотек в python довольно прост, все что необходимо сделать, это открыть командную строку и прописать следующую команду: pip install (название библиотеки), после чего начнется процесс установки, демонстрация процесса показана на рисунке 9.

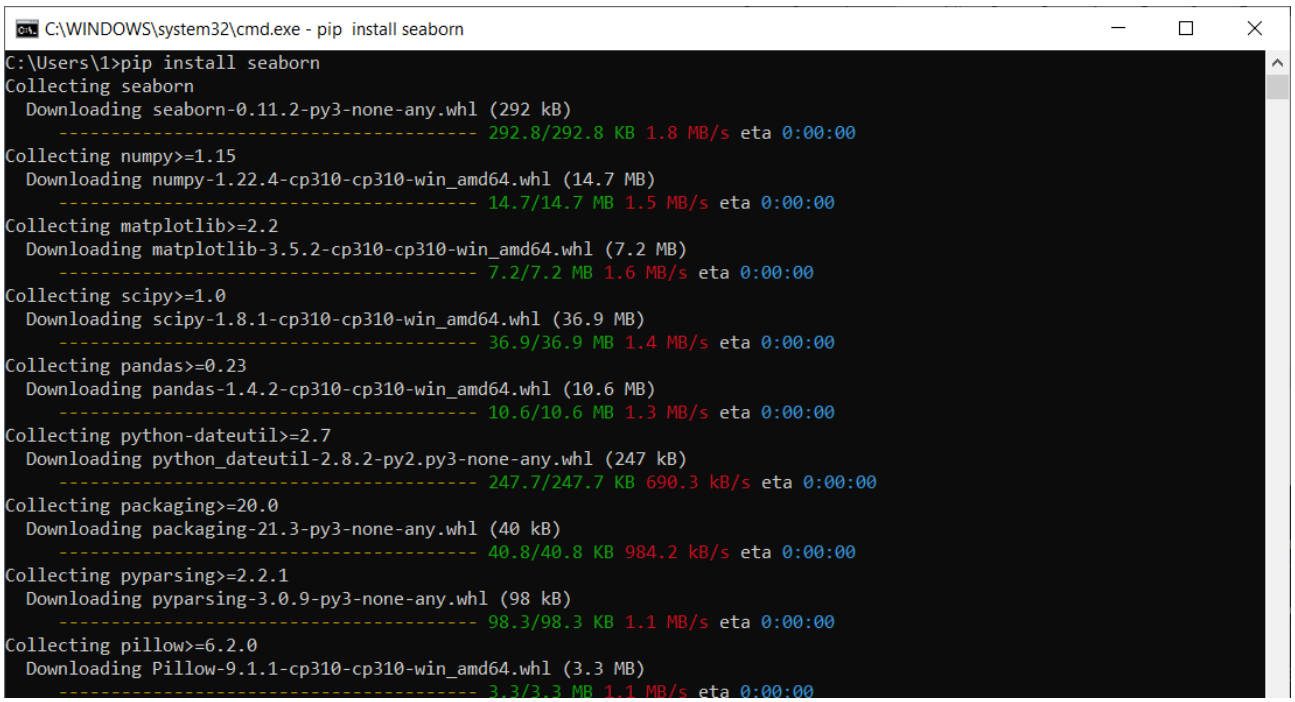

Рисунок 9 – Процесс установки библиотеки.

Перед началом интерполяции нужно занести в программу данные, полученные из прошлых этапов, и вывести их.

Программа принимает на вход 2 массива данных из входных файлов. Первым набором данных является итог работы программы по фильтрации данных, вторым набором данных является исходно введенные значения. В итоге программа имеет 2 массива, x и y которых совпадают друг с другом по ключевому элементу в массиве. Далее для корректного отображения графика, программа должна отсортировать значения, для этого нужно применить синхронную сортировку ее реализация показана на рисунке 10.

```
xy = zip(x,y)
xys = sorted(xy, key=lambda tup: tup[0])
x = [xy[0] for xy in xys]
y = [xy[1] for xy in xys]
```

Рисунок 10 – Сортировка значений.

После этого данные готовы чтобы передать их в переменную библиотеки pandas, для дальнейшей работы с ней при построении графика. Запишем их в переменную, рисунок 11. X и Y данной переменной являются заголовками.

```
df = pd.DataFrame({
    'X': x,
    'Y': y
    })
```

Рисунок 11 – Переменная, задающая функцию.

Далее необходимо написать код, который выводит данные переменные в удобной для понимая форме, рисунок 12.

```
ax = sns.scatterplot(data=df, x='X', y='Y')
ax.set_title('Исходные данные', fontsize=25)
sns.set(rc={'figure.figsize':(8,8)})
plt.show()
```

Рисунок 12 – Код для вывода переменной.

Теперь, подготовив исходные данные к работе, мы можем вывести их в виде графика, вывод исходных данных показан на рисунке 13.

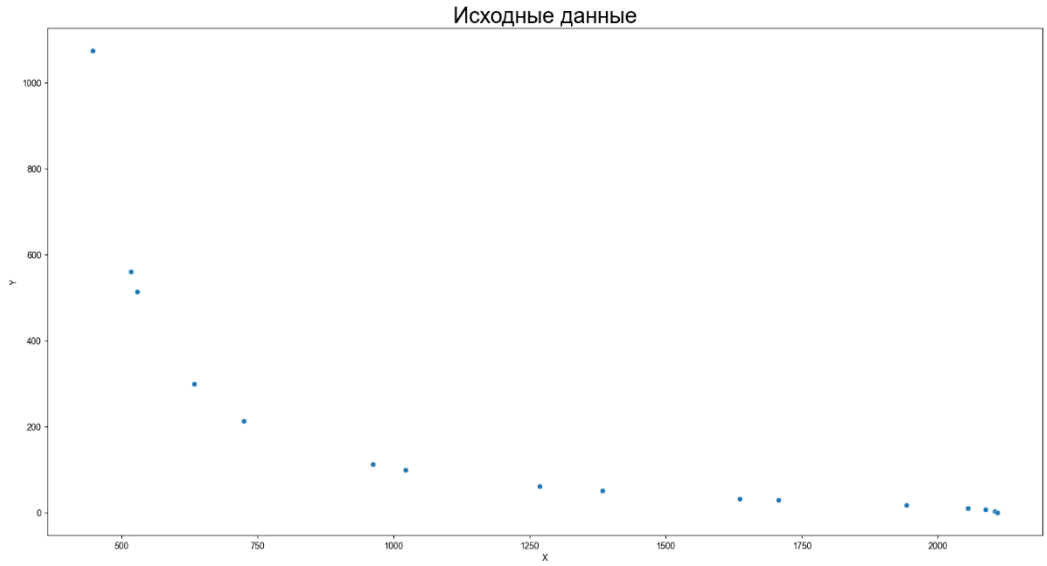

Рисунок 13 – Исходные данные.

Чтобы определить лучший метод интерполяции, нам необходимо иметь полный набор данных, выделить из него небольшое количество точек, и на их основе произвести интерполяцию функции. На данном графике видно, что исходные точки напоминают график функции $y = \frac{1}{x}$, так что для сравнения методов возьмем эту функцию за основу. Выберим из этой функции 5% точек и выделим их другим цветом для наглядности. Итого мы имеем график для сравнения, который показан на рисунке 14.

Теперь у нас все готово для сравнения методов интерполяции.

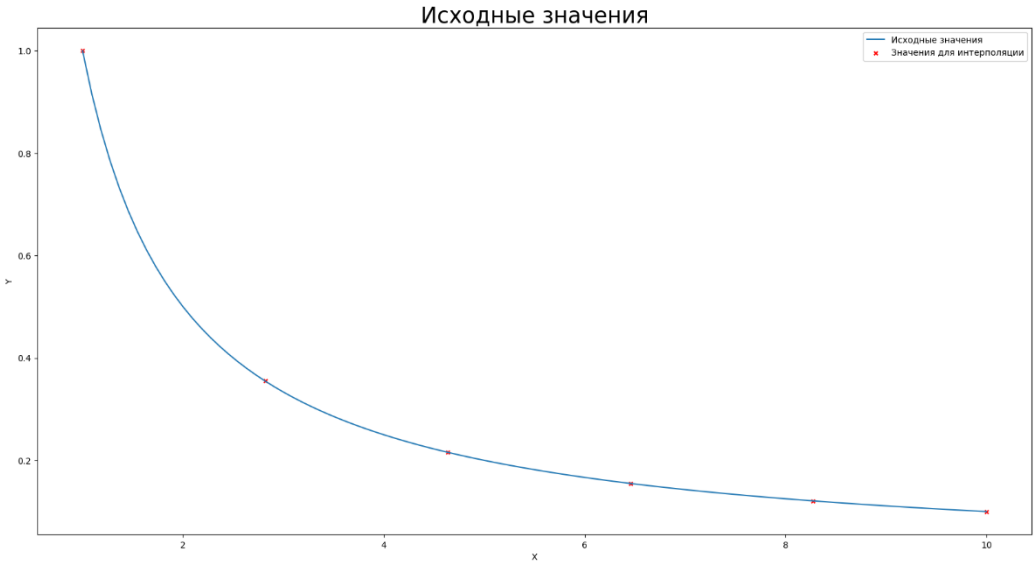

Рисунок 14 – Сравнение установок

3.3. Линейная интерполяция

Самым простым методом интерполяции является линейная интерполяция.

Линейная интерполяция — интерполяция алгебраическим двучленом $P_1(x) = ax + b$ функции $f(x)$, заданной в двух точках x_0 и x_1 отрезка [a, b]. В случае, если заданы значения в нескольких точках, функция заменяется кусочно-линейной функцией [18].

Для реализации ПО она не подходит, но она хорошо подходит для наглядного сравнения работы других методов интерполяции. Этот метод

представляет из себя объединение точек между собой. Библиотека scipy позволяет провести линейную интерполяцию. Построим график линейной интерполяции с выбранными точками и со всеми точками, код показан на рисунке 15.

```
f_interp_select_lin = interp1d(df_selected_points.X,
                               df_selected_points.Y,
                               kind='linear',
                               bounds_error=False,
                               fill_value="extrapolate")
f_interp_full_lin = interp1d(df.X,
                             df.Y,
                             kind='linear',
                             bounds_error=False,
                             fill_value="extrapolate")
```

Рисунок 15 – Реализация линейной интерполяции

На графике, который показан на рисунке 16 хорошо видно, что метод линейной интерполяции не годится для решения поставленной задачи, так как имеет слишком большие отклонения от изначального графика.

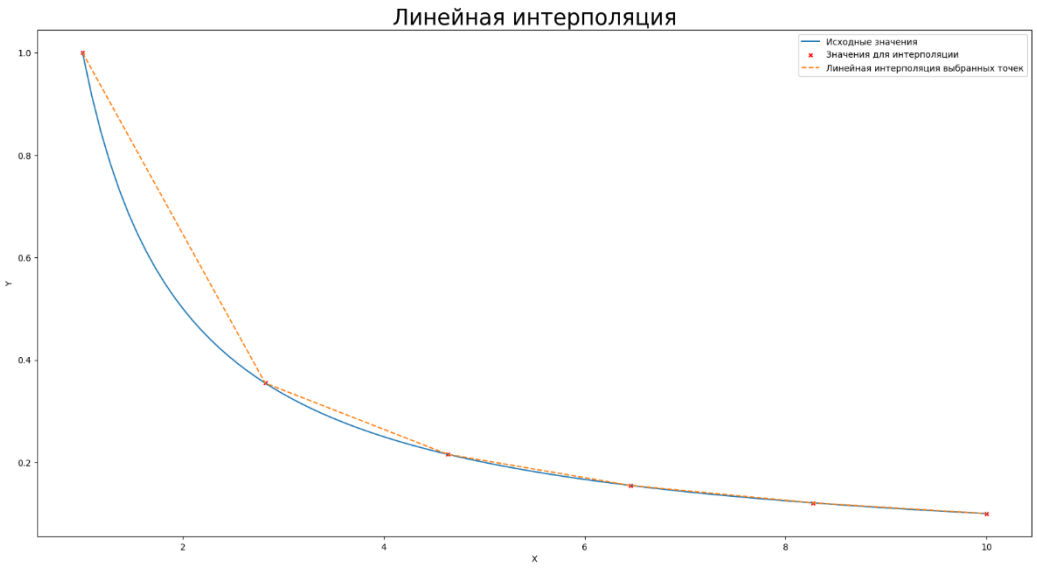

Рисунок 16 – Линейная интерполяция.

3.4. Сплайн аппроксимация

Математические сплайны берут свое начало от тонких гибких стержней, которыми пользовались чертежники для проведения плавных кривых через

заданные точки. Стержень закреплялся в точках (x_i, y_i) и принимал форму кривой y(x) с минимальной "энергией натяжения", пропорциональной $\int (y")^2 dx$ [25].

Если перейти к математическому описанию сплайна, то сплайн-функцией степени k с точками соединения $x_0 < x_1 < \ldots < x_n$ будет функция $y(x)$, которая на отрезке $[x_0, x_n]$ имеет непрерывные производные до $(k-1)$ включительно и на каждом из отрезков $[x_{i-1}, x_i]$ равна многочлену степени k. Далее нас будут интересовать лишь кубические сплайны $(k = 3)$. Такой сплайн обеспечивает совпадение в узлах с исходной функцией и непрерывность первой и второй производных в точках соединения [2].

Прежде всего определяются первые производные во всех точках соединения. Решается трехдиагональная система $(n-1)$ уравнений с доминирующей главной диагональю. Она может быть решена, если задать два краевых условия [15]. Программы, включенные в Графор, позволяют задавать четыре типа краевых условий:

- известны значения первых производных на концах сетки (y'_0, y'_n);

- известны значения вторых производных на концах сетки (y'_0, y'_n);

- при построении периодического сплайна значения функции в первой и последней точках совпадают, а также $y'_0 = y'_n$ и $y''_0 = y''_n$;

- если краевые условия не заданы, то считается, что $y''_0 = y''_n = 0$.

После того как построена трехдиагональная матрица (или дополненная трехдиагональная матрица в случае в) и вектор свободных членов, система уравнений может быть решена. Для решения такой системы существует эффективный алгоритм, который в отечественной литературе называется методом прогонки. В результате решения системы уравнений получается вектор первых производных y'_i в точках соединения [6].

Теперь значение $y(x)$ для $x_{i-1} \le x \le x_i$ определяется из многочлена

$$y(x) = a_i x^3 + b_i x^2 + c_i x + d_i \qquad (1)$$

Коэффициенты которого легко найти, поскольку известны значения функции и первые производные в точках соединения [5].

3.5. Квадратичная интерполяция

Квадратичные сплайны, это тоже относительно простой метод интерполяции.

Если интерполирующая функция - многочлен второго порядка $y = ax^2 + bx + c$, то интерполяция называется квадратичной. Иногда ее называют параболической на отрезке $[x_{i-1}, x_{i+1}]$ так как квадратный трехчлен - это парабола $y = a_i x^2 + b_i x + c_i$, где a_i, b_i, c_i - неизвестные. Для их определения необходимо условие прохождения параболы через три точки: $(x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1})$ [16].

Эти условия записываются в виде:

$$\begin{cases} a_i x_{i-1}^2 + b_i x_{i-1} + c_i = y_{i-1} \\ a_i x_i^2 + b_i x_i + c_i = y_i \\ a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \end{cases} \quad (2)$$

Решив систему, получим значения a_i, b_i, c_i , а, следовательно, и уравнение параболы на участке $[x_{i-1}, x_{i+1}]$. Уравнения парабол на разных отрезках $[x_{i-1}, x_{i+1}]$ разные, i=1 ,…, n. Квадратичная интерполяция является локальной интерполяцией [13].

Для построения графика также воспользуемся библиотекой scipy, код выглядит следующим образом, рисунок 17.

```
f_interp_select_quad = interp1d(df_selected_points.X,
                                df_selected_points.Y,
                                kind='quadratic',
                                bounds_error=False,
                                fill_value="extrapolate")
f_interp_full_quad = interp1d(df.X,
                              df.Y,
                              kind='quadratic',
                              bounds_error=False,
                              fill_value="extrapolate")
```

Рисунок 17 – Реализация квадратичной интерполяции.

После выполнения кода, мы получаем график, который показан на рисунке 18, интерполяции значений методом квадратичных сплайнов. На этом графике мы также видим, что для интерполяции экспериментальных данных, этот метод также является недостаточно точным, хотя он и лучше, чем линейная интерполяция.

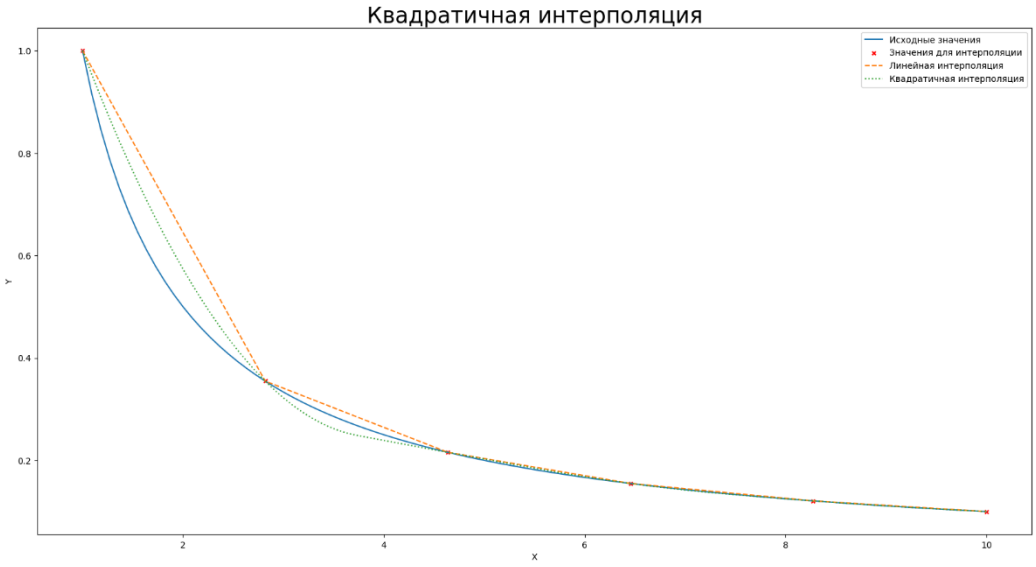

Рисунок 18 - Квадратичная интерполяция.

3.6. Кубическая интерполяция

Если значения функции f(x) и ее производной известны в точках x=0 и x=1, тогда функция может быть интерполирована на интервале [0, 1] используя полином третьего порядка. Формула для вычисления может быть легко получена [14].

Полином третьего порядка и его производная:

$$f(x) = ax^3 + bx^2 + cx + d \quad (3)$$

$$f'(x) = 3ax^2 + 2bx + c \quad (4)$$

Значения полинома и его производной в точках x=0 и x=1

$$f(0) = d \tag{5}$$

$$f(1) = a + b + c + d \tag{6}$$

$$f'(0) = c \tag{7}$$

$$f'(1) = 3a + 2b + c \tag{8}$$

Эти четыре тождества могут быть записаны как:

$$a = 2f(0) - 2f(1) + f'(0) + f'(1) \tag{9}$$

$$b = -3f(0) + 3f(1) - 2f'(0) = f'(1) \tag{10}$$

$$c = f'(0) \tag{11}$$

$$d = f(0) \tag{12}$$

Итак, мы получили нашу интерполяционную формулу

На практике алгоритм используют для интерполяции функции, имея некие известные значения в заданных точках. В этом случае мы не можем знать производную функции. Мы могли бы принять производную в заданных точках, как 0, однако для получения более гладких и правдоподобных графиков функций мы примем за производную уклон линии между предыдущей и следующей точкой. Таким образом для расчетов нам понадобится 4 точки. Предположим, мы имеем 4 значения функции в точках p_0, p_1, p_2, p_3, расположенных соответственно на x=-1, x=0, x=1 и x=2. Подставим полученные значения $f(0), f(1), f(2), f(3)$:

$$f(0) = p_1 \tag{13}$$

$$f(1) = p_2 \tag{14}$$

$$f'(0) = \frac{p_2 - p_0}{2} \tag{15}$$

$$f'(1) = \frac{p_3 - p_1}{2} \tag{16}$$

Сопоставив эти данные с полученными ранее формулами, мы имеем:

$$a = -\frac{1}{2}p_0 + \frac{3}{2}p_1 - \frac{3}{2}p_2 + \frac{1}{2}p_3 \tag{17}$$

$$b = p_0 - \frac{5}{2}p_1 + 2p_2 - \frac{1}{2}p_3 \tag{18}$$

$$c = -\frac{1}{2}p_0 + \frac{1}{2}p_2 \tag{19}$$

$$d = p_1 \tag{20}$$

Для построения графика также воспользуемся библиотекой scipy, код предоставлен на рисунке 19.

```
f_interp_select_cubic = interp1d(df_selected_points.X,
                                 df_selected_points.Y,
                                 kind='cubic',
                                 bounds_error=False,
                                 fill_value="extrapolate")
f_interp_full_cubic = interp1d(df.X,
                               df.Y,
                               kind='cubic',
                               bounds_error=False,
                               fill_value="extrapolate")
```

Рисунок 19 – Реализация кубической интерполяции.

После выполнения кода, мы получаем график, рисунок 20, интерполяции значений методом кубических сплайнов. На этом графике мы видим, что этот метод также недостаточно точен, по графику сложно сказать какой из этих двух методов является более точным. Перейдем к последнему методу перед сравнением всех методов интерполяции.

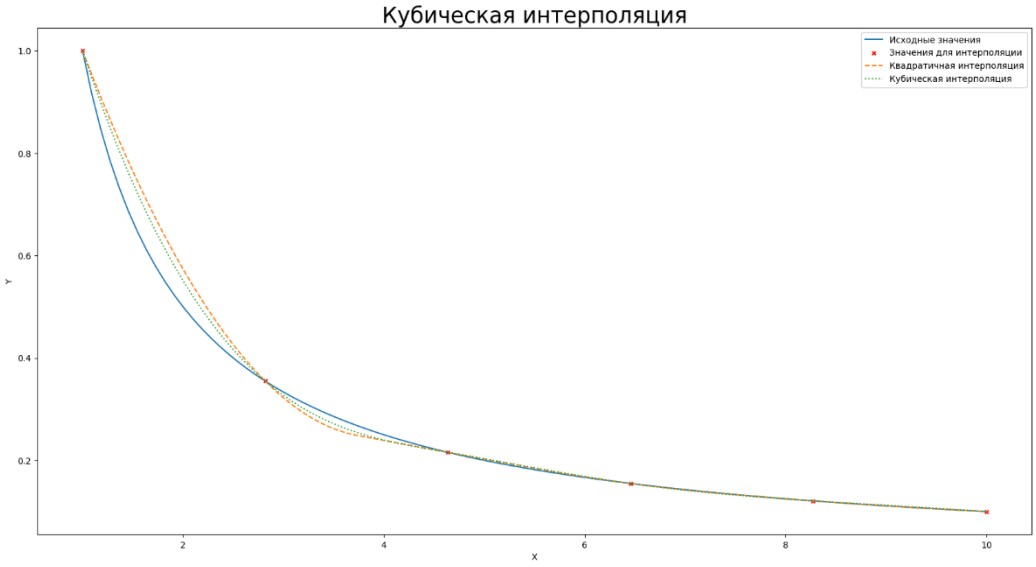

Рисунок 20 – Кубическая интерполяция.

3.7. Интерполяция методом сплайна Акимы

Наиболее часто обсуждаемым методом сплайна является кубический сплайн. Хорошо известная проблема с кубическим сплайном заключается в том, что он не является локальным. Это означает, что если в наборе данных есть выброс, то кривая вокруг окружающих точек будет «шатающейся». Сплайн Akima намного более стабилен и не подвержен влиянию выбросов в той же степени. Ниже, на рисунке 21, приведен наглядный пример этого пункта, приведенный в «Руководстве по компьютерной графике» Дэвида Саломона [20].

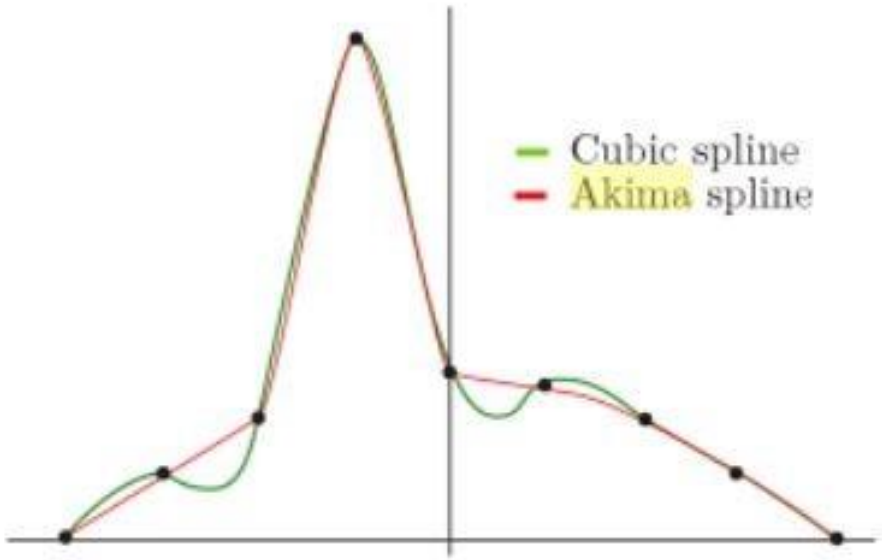

Рисунок 21 – Сравнение сплайнов.

Акима предложил в своей статье 1970 года подобрать кубический полином между двумя точками x(i) и x(i+1).

$$y = p_0 + p_1(x - x_i) + p_2(x - x_i)^2 + p_3(x - x_i)^3 \quad (21)$$

Два ограничения на полином заключаются в том, что функция слева от x(i) дает то же значение y, что и функция справа от x_i. Аналогичное ограничение непрерывности накладывается на x_{i+1} [10]. Кроме того, Акима также наложил в каждой точке x(i) ограничение, согласно которому наклон t в точке x(i) равен:

$$t_1 = \frac{(|m_{i+1} - m_i|m_{i-1} + |m_{i-1} - m_{i-2}|m_i)}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} \quad (22)$$

где каждый m представляет собой прямую линию между окружающими сегментами кривой (секанс) и определяется как:

$$m_i = \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)} \tag{23}$$

В частном случае, когда знаменатель в формуле для t равен нулю, вместо этого мы будем использовать:

$$t_i = .5(m_i + m_{i-1}) \tag{24}$$

Как мы можем подогнать кубический многочлен, который начинается с крайней левой точки и заканчивается в самой правой точке, если у нас недостаточно информации для вычисления секущих за этими точками данных? Акима предложил экстраполировать данные, предположив, что 2 точки слева и две точки справа от наших данных лежат на квадратичной функции, определяемой как:

$$y = g_0 + g_1(x - x_i) + g_2(x - x_i)^2 \tag{25}$$

где все g постоянны. Это дает (при условии, что конечная точка равна x_3):

$$\frac{y_5 - y_4}{x_5 - x_4} - \frac{y_4 - y_3}{x_4 - x_3} = \frac{y_4 - y_3}{x_4 - x_3} - \frac{y_3 - y_2}{x_3 - x_2} = \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1} \tag{26}$$

Это, конечно, означает, что мы можем вычислить экстраполированные секущие, предшествующие нашему первому значению x, как:

$$m_{-2} = 2m_{-2} - m_0 \; and \; m_{-1} = 2m_0 - m_1 \tag{27}$$

и мы можем вычислить экстраполированные секущие за пределами нашей последней точки данных как:

$$m_{n+2} = 2m_{n+1} - m_n \; and \; m_{n+3} = 2m_{n+2} - m_{n+1} \tag{28}$$

Теперь у нас достаточно информации, чтобы вывести коэффициенты интерполяционного кубического многочлена между любой парой наших заданных данных [11][12]. Окончательная формула дается:

$$y = p_0 + p_1(x - x_i) + p_2(x - x_i)^2 + p_3(x - x_i)^3 \tag{29}$$

Где:

$$p_0 = y_i \tag{30}$$

$$p_1 = t_i \tag{31}$$

$$p_2 = \frac{[3\frac{(y_{i+1}-y_i)}{(x_{i+1}-x_i)}-2t_i-t_{i+1}]}{(x_{i+1}-x_i)} \quad (32)$$

$$p_3 = \frac{[t_i+t_{i+1}-2\frac{(y_{i+1}-y_i)}{(x_{i+1}-x_i)}]}{(x_{i+1}-x_i)^2} \quad (33)$$

Для построения графика нужно импортировать библиотеку Akima1DInterpolator:

from scipy.interpolate import Akima1DInterpolator

Код построения графика методом сплайна Акимы показан на рисунке 22.

```
f_interp_select_akima_spl = Akima1DInterpolator(df_selected_points.X,
                                                df_selected_points.Y)
f_interp_all_akima_spl = Akima1DInterpolator(df.X,
                                             df.Y)
```

Рисунок 22 – Реализация Сплайна Акимы.

После выполнения кода, мы получаем график, рисунок 23, интерполяции значений методом сплайнов Акимы. На этом графике мы четко видим, насколько метод сплайнов Акимы превосходит по точности остальные методы, так как на втором промежутке практически полностью совпадает с исходным графиком функции. Но для точного анализа нужно провести сравнение точности методов.

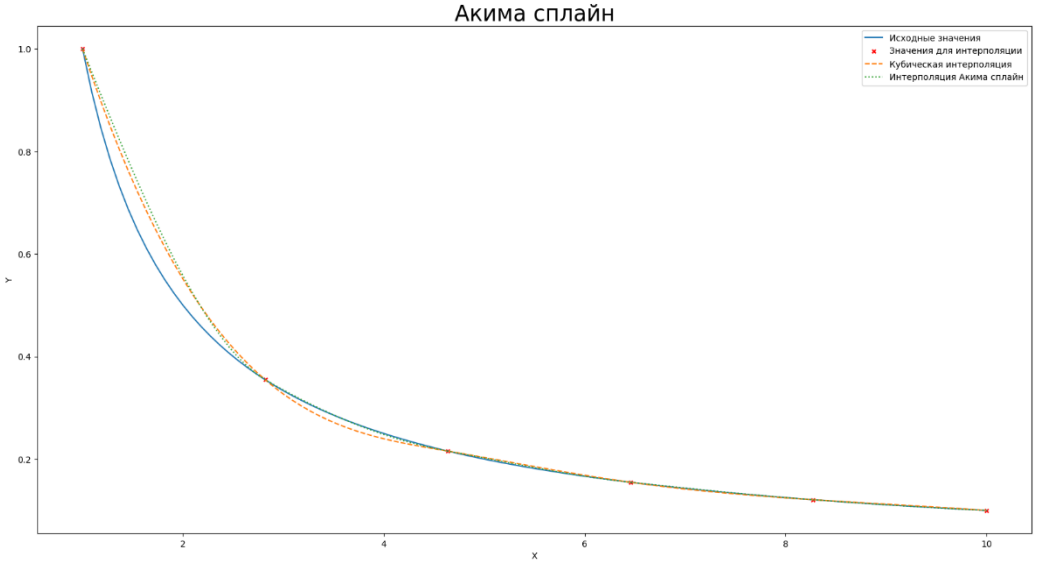

Рисунок 23 – Сплайн Акимы.

3.8. Сравнение методов интерполяции

После того, как был выведен график каждого метода интерполяции, необходимо провести оценку точности каждого метода. Это позволит определить какой из методов будет использован для конечной реализации программы. Чтобы это реализовать, необходимо сравнить их всех вместе. Для этого нужно построить график, на котором представлены все вышеупомянутые методы, рисунок 24.

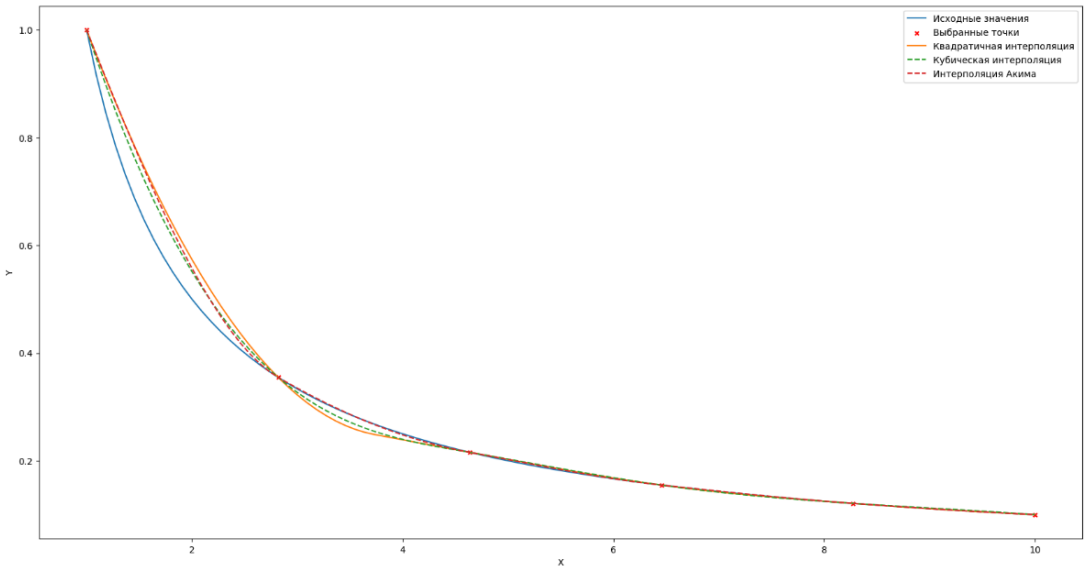

Рисунок 24 – График сравнения методов интерполяции на выбранной функции.

По данному графику можно предположить, что метод сплайнов Акимы лучше всего справился с задачей, но, чтобы точно определить это, нужно вычислить среднюю ошибку всех точек.

Для каждого представленного метода мы будем интерполировать данные для исходного значения X и вычислять ошибку. Как только интерполяция рассчитана, мы можем, наконец, определить ошибку. Для этого вычисления ошибки в данном исследовании, мы будем использовать среднеквадратичную ошибку.

Среднеквадратичная ошибка — это функция риска, соответствующая ожидаемому значению квадрата потерь из-за ошибки. Тот факт, что Среднеквадратичная ошибка почти всегда является строго положительной (а не нулевой), это объясняется случайностью или тем, что оценщик не учитывает информацию, которая могла бы дать более точную оценку. Среднеквадратичная ошибка — это мера качества оценки — она всегда неотрицательна, а значения ближе к нулю — лучше [24].

Мы вычислим корень квадратной разницы между исходными и интерполированными данными, деленный на исходные данные, умноженный на 100. Он покажет процент ошибки.

$$error = \frac{\sqrt{(y_{original} - y_{interpolated})^2}}{y_{original}} \quad (34)$$

Программный код выглядит следующим образом, он показан на рисунке 25.

```
df_error = pd.DataFrame()
for col in new_df_full.columns[2:]:
    df_error[f'{col}_err'] = np.sqrt((new_df_full.Y - new_df_full[col])**2
                                     )/new_df_full.Y*100
print(df_error)
new_df_full.index = new_df_full.X
df_error.index = new_df_full.X
print('Interpolation Method Error:\n')
for i,j in zip(df_error.mean().sort_values().index,
               df_error.mean().sort_values()):
    print(f'{i}:\t{j:2.2f}%')
```

Рисунок 25 – Реализация расчета среднеквадратичной ошибки.

Выводом программы является график, который показан на рисунке 24, а также средняя ошибка методов интерполяции, таблица 1.

Таблица 1 - Средняя ошибка интерполяции.

Метод интерполяции	Среднеквадратичная ошибка
Akima_err	1.96%
Cubic_err	2.66%

Quad_err	3.23%
Linear_err	5.43%

В полученной таблице видно превосходство метода сплайнов Акимы над другими методами интерполяции. Это также было видно и на полученном графике. Основываясь на данных результатах, для реализации конечного результата работы, мы будем использовать самый эффективный метод.

Вывод по разделу 3

В данном разделе был описан и рассмотрен метод интерполяции. Были представлены различные его способы реализации, а именно:

- линейный;
- квадратичный;
- кубический;
- сплайн Акимы.

С помощью построения графиков было наглядно представлена работа каждого из методов, а также сравнение их эффективности.

В конце раздела было проведено сравнение всех методов с помощью среднеквадратичной ошибки. Было выяснено, что метод сплайнов Акимы является самым эффективным для решения поставленной задачи.

4 Вывод исходных данных в таблицу функциональной зависимости

4.1. Получение функциональной зависимости

Конечным результатом проделанной работы является получение таблицы зависимости по исходным данным. Был определен метод интерполяции, на основе которого будет реализована программа, им является метод сплайнов Акимы. Так как все предыдущие пункты уже выполнены, и мы имеем набор отфильтрованных данных, а также график интерполяции, осталось вывести эти данные в виде таблицы.

Вывод данных будет произведен с помощью функции библиотеки pandas to_excel. Эта функция позволяет вывести таблицу данных в удобном для чтения виде в формате xlsx.

Для вывода нужно перенести все полученные в процессе интерполяции значения и занести их в новую переменную. Данные этой переменной будут перенесены в excel.

Так как переменная со значениями X и Y была создана выше, осталось только провести интерполяцию её значений. Вывод графика интерполяции исходных данных предоставлен на рисунке 26.

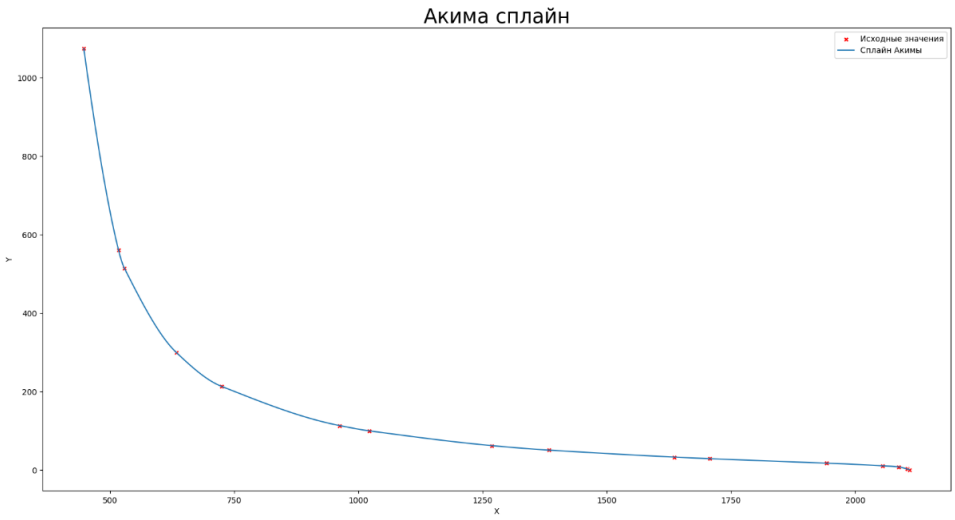

Рисунок 26 – Интерполяция исходных данных

После получения функциональной зависимости в виде графика, исходные данные после процесса интерполяции переносятся в excel. Количество всех полученных точек задается перед выполнением интерполяции в переменную, точность вычислений можно изменить. Пример вывода данных после интерполяции предоставлен на таблице 2

Таблица 2 - Пример вывода данных

	X	Y
0	447	1073,5
1	448,0006	1064,446
2	449,0012	1055,426
3	450,0018	1046,442
4	451,0024	1037,494
5	452,003	1028,583
6	453,0036	1019,709
7	454,0042	1010,873
8	455,0048	1002,077
9	456,0054	993,3191
0	457,006	984,6017

С помощью данной таблицы, можно получить примерные значения Y, которые соответствуют конкретным значениям X.

4.2. Проверка работоспособности программы

После разбора задачи на первом примере, чтобы убедится в его универсальности и работоспособности, нужно получить функциональную зависимость на втором массиве данных. Сложность следующего набора данных заключается в малом количестве замеров данных по сравнению с первым экспериментом. Из-за этого точность интерполяции может снизиться, но программа все равно должна произвести интерполяцию. Для этого нужно пройти все уже выполненные выше шаги.

Для начала необходимо визуализировать исходный набор данных, полученный после экспериментального замера. Исходный набор данных

продемонстрирован на рисунке 27, на нем четко видно 5 точек, которые программа должна определить.

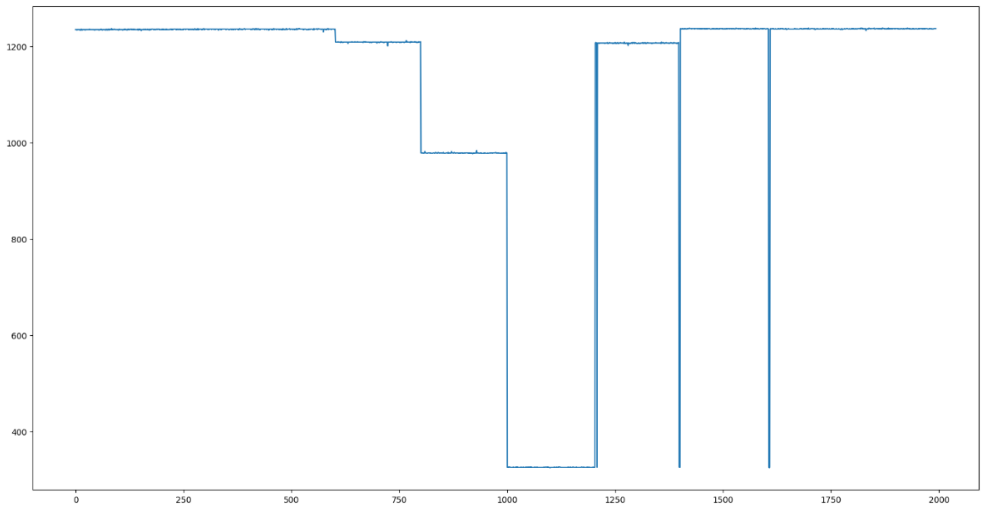

Рисунок 27 – Визуализация второго набора данных.

Затем необходимо структурировать их для дальнейшей интерполяции. После обработки исходного массива данных, мы получили искомые 5 точек, они показаны на рисунке 28.

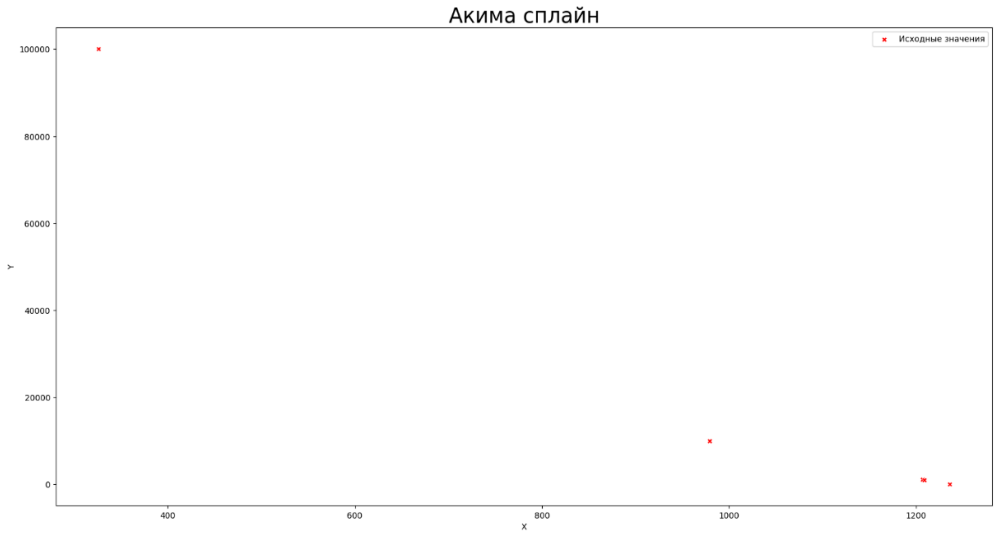

Рисунок 28 – Структуризация данных.

Дальше необходимо провести интерполяцию с использованием полученных значений и получить функциональную зависимость. На рисунке 29 видно, что программа справилась со своей задачей, а также вывела таблицу значений, которая показана на таблице 3.

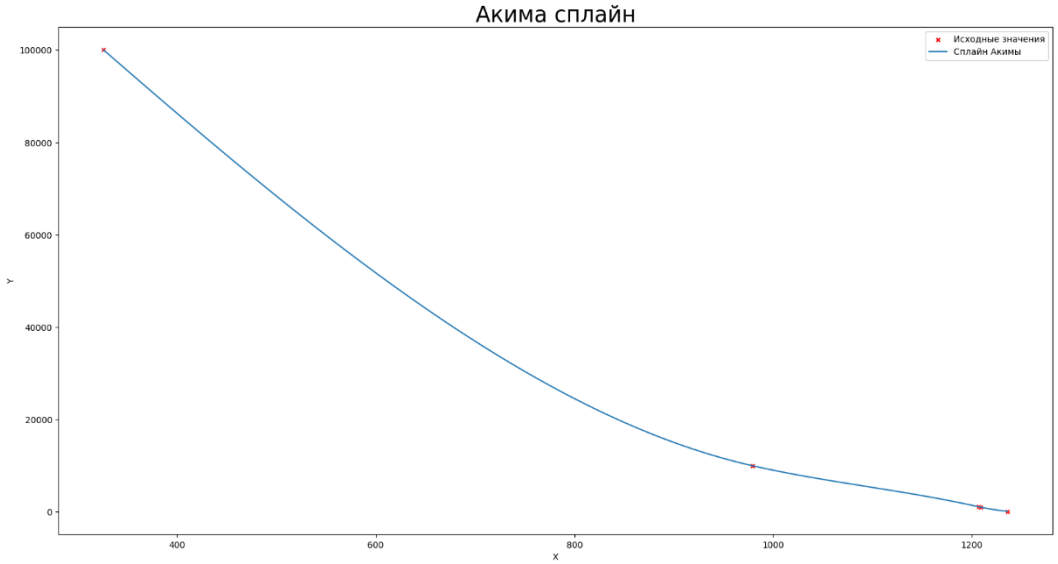

Рисунок 29 – Интерполяция второго набора данных.

Таблица 3 - Пример вывода данных.

	X	Y
0	326	100000
1	327,0011	99812,59
2	328,0022	99625,21
3	329,0033	99437,86
4	330,0044	99250,54
5	331,0055	99063,26
6	332,0066	98876,01
7	333,0077	98688,79
8	334,0088	98501,6
9	335,0099	98314,46
10	336,011	98127,34

С помощью данной таблицы, можно получить примерные значения Y, которые соответствуют конкретным значениям X. Точность выходных данных можно увеличить. Это делается путем увеличения значения переменной,

которая отвечает за количество вводных в функцию для интерполирования значений.

Вывод по разделу 4

В четвертом разделе был реализован перенос работы программы в удобно читаемый вид, в вид excel файла. Это позволит сопоставлять значения X со значениями Y.

Четвертый раздел стала итогом всей работы, через алгоритм был прогнан второй набор исходных данных, который отличался от первого малым количеством значений. Данная проверка показала пригодность работы программы с малым количеством исходных данных.

Заключение

В ходе проведенной работы, были рассмотрены различные методы структуризации и фильтрации данных для дальнейшей работы с ними. Были выбраны способы, с помощью которых можно было провести структуризацию данных для дальнейшего удобства работы с ними. В результате была создана программа, которая принимает набор данных дает на выходе структурированные данные. Ее создание было необходимо для ввода обработанных исходных значений в функцию интерполяции.

Был рассмотрен метод аппроксимации, с помощью которого должна решаться задача нахождения функционально зависимости. Были исследованы различные алгоритмы интерполяции, среди которых были:

- линейный;
- квадратичный;
- кубический;
- сплайн Акимы.

Был подробно рассмотрен метод сплайнов Акимы. По каждому из приведенных выше методов была реализована функция, которая выводила график интерполяции исходных данных.

Был проведено сравнение эффективности всех методов интерполяции с помощью среднеквадратичной ошибки, в результате чего был выделен наиболее эффективный метод для решения данной задачи.

В результате работы программы, был осуществлен вывод интерполированных значений в удобно читаемом виде.

Вся работа программы была изначально проверена на тестовой функции. Были выбраны исходные точки, на которых был проведена проверка работоспособности и сравнение эффективности функций различных методов интерполяции. Данная функция была выбрана максимально схожей с исходным набором данных.

После проверки эффективности алгоритма, была проведена проверка работоспособности программы на маленьком количестве исходных данных.

Список используемой литературы

1. Ахиезер Н. И. ЛЕКЦИИ ПО ТЕОРИИ АППРОКСИМАЦИИ // НАУКА. 1965. (409).
2. Бикубическая интерполяция, теория и практическая реализация [Электронный ресурс] // URL: https://habr.com/ru/post/111402/
3. Гребенникова И. В. МЕТОДЫ МАТЕМАТИЧЕСКОЙ ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ // Издательство Уральского университета. 2015. (126)
4. Грищенко Н.В. Семериков С.А. Хараджян А.А. Чернов Е.В. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ АППРОКСИМАЦИИ // 1998 (26)
5. К. Де Бор. Практическое руководство по сплайнам // РиС. 1985. (305)
6. Оптимальная аппроксимация сплайнами [Электронный ресурс] // URL: https://habr.com/ru/post/314218/
7. Степанов М. М., Потапова Н. Н., Ерещенко Т. В. АППРОКСИМАЦИЯ ФУНКЦИЙ. ВолгГАСУ. 2012. (34)
8. Частотность [Электронный ресурс] // URL: https://wiki.cologne/частотность
9. Частотный анализ русского текста и облако слов на Python [Электронный ресурс] // URL: https://habr.com/ru/post/517410/
10. Akima H. A new method of interpolation and smooth curve fitting based on local procedure // Journal of the ACM, Vol. 17, Issue 4. 1970. (589–602). URL: http://www.leg.ufpr.br/lib/exe/fetch.php/wiki:internas:biblioteca:akima.pdf.
11. Akima H. Algorithm 761. Scattered data surface fitting // TOMS. 1996. (10)
12. Akima H., Petzold T., Maechler M. Interpolation of Irregularly and Regularly Spaced Data // ACM. 2022. (28). URL: https://cran.r-project.org/web/packages/akima/akima.pdf.
13. Dierckx P. Curve and Surface Fitting with Splines // Claredon press. 1995. (157)

14. Gurruchaga J. R. Python Recipes for Engineers and Scientists // Independently published. 2018 (104)
15. Interpolation. [Электронный ресурс] // URL: https://scipy-cookbook.readthedocs.io/items/Interpolation.html
16. Miscellaneous - ScottPlot 4.1 Cookbook [Электронный ресурс] // URL: https://scottplot.net/cookbook/4.1/category/misc/
17. NumPy Documentation [Электронный ресурс] // URL: https://numpy.org/doc/
18. Ozdemir H., Comparison of linear, cubic spline and akima interpolation methods. (3) URL: https://www.jive.eu/jivewiki/lib/exe/fetch.php?media=expres:fabric:interpolation.pdf.
19. pandas documentation [Электронный ресурс] // URL: https://pandas.pydata.org/docs/
20. Parametric Models of Function [Электронный ресурс] // URL: http://www.brnt.eu/phd/node11
21. SciPy documentation [Электронный ресурс] // URL: https://docs.scipy.org/doc/scipy/
22. seaborn: statistical data visualization [Электронный ресурс] // URL: https://seaborn.pydata.org/
23. Simple Frequency Analysis Rules [Электронный ресурс] // URL: https://docs.oracle.com/cd/E19182-01/821-0859/dsgn_mi-freq-smpl_r/index.html
24. Understanding Regression Error Metrics in Python [Электронный ресурс] // URL https://www.kaggle.com/code/caesarlupum/understanding-regression-error-metrics-in-python/notebook
25. Using B-splines in scipy.signal [Электронный ресурс] // URL: https://scipy.github.io/old-wiki/pages/Cookbook/Interpolation.html

Приложение А

Вывод исходных данных

```
import matplotlib.pyplot as plt

file = []
with open('Data.txt') as f:
    file = f.read().splitlines()

num = len(file)
for i in range(num):
    file[i] = int(file[i])

x_list = list(range(num))
print(x_list)
plt.plot(x_list, file)
plt.show()
```

Приложение Б

Структуризация исходных данных

```
import matplotlib.pyplot as plt

file = []
with open('Data.txt') as f:
     file = f.read().splitlines()
num = len(file)
print(num);
for i in range(num):
     file[i] = int(file[i])

k=3;    #разброс
n=50;   #минимальное количество
x=3;    #для выбросов
x1=0;  #счетчик для выбросов
a=[];   #итоговый массив
b=[];   #промежуточный массив
first=0;#вывод прохода
last=0;
result=file[0];#среднее временное значение

for i in range(num):
     if  ((abs(file[i]-result)  >  k)  and  x1>x):#проверка  выхода
текущего значения из диапазона
     if (len(b) < n):
        b=[];
        first=last;
        i-=2;
        result=file[i];
        x1=0;
     else:
        a.append(round(sum(b) / len(b)));
        first=last;
        b=[];
```

```
        i-=2;
        result=file[i];
        x1=0;
     elif (abs(file[i]-result) > k):
        x1+=1;
     else:
        b.append(file[i]);
        result = (sum(b) / len(b));
        last=i;

a2=[];
for item in a:
     if item not in a2:
        a2.append(item)
x=0;
a3 = a2;
n=len(a3);
delM = [];#массив на удаление

for i in range(n):
     for j in range(i+1, n):
        if (abs(a3[j] - (a3[i])) <= 1):
           delM.append(a3[j]);

for item in delM:
     a3.remove(item);

with open('calib_data2.txt', 'w') as f:
     for item in a3:
          f.write(str(item) + '\n');
```

Приложение В

Вывод графика и таблицы результатов

```
import numpy as np
import scipy as sp
import pandas as pd
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
from matplotlib.axes import Axes
import seaborn as sns

with open('calib_data3.txt') as f:
    file = f.read().splitlines() #записываем откалиброванные данные
в переменную для дальнейше работы с ними
x=file;
result = [float(item) for item in x]#переводим данные в числовые
значения
x=result

with open('second_data3.txt') as f:
    file = f.read().splitlines()
y1=file;
result = [float(item) for item in y1]
y1=result
y=[]
for item in y1: #убираем повторения
     if item not in y:
         y.append(item)

"""Синхронная сортировка"""
#соединим два списка специальной функцией zip
xy = zip(x,y)
#отсортируем, взяв первый элемент каждого списка как ключ
xys = sorted(xy, key=lambda tup: tup[0])
#и последний шаг - извлечем
```

```
x = [xy[0] for xy in xys]
y = [xy[1] for xy in xys]
print(x)
print(y)
df = pd.DataFrame({#структурируем x и y для дальнейшей работы
    'X': x, #загаловки
    'Y': y
    })
#задаем количество точек
first=x[0];
last=x[-1];
num_points=int((last-first))
x_to_interp = np.linspace(first, last, num_points)

#Akima spline
from scipy.interpolate import Akima1DInterpolator
f_interp_full_akima_spl = Akima1DInterpolator(df.X,
                                              df.Y)



y1=f_interp_full_akima_spl(x_to_interp)
df_akima=pd.DataFrame()
df_akima.insert(0,"Y", y1, False)
df_akima.insert(0, "X", x_to_interp, False)
df_akima.to_excel('./interpolation.xlsx')

#Показ исходных значений
ax = sns.scatterplot(data=df,
                     x='X', y='Y',
                     color='r', marker='X')
#akima spline
sns.lineplot(x=x_to_interp,
             y=f_interp_full_akima_spl(x_to_interp),
             linestyle='-')
```

```
df_akima = pd.DataFrame({
    'X': df.X,
    'Y': f_interp_select_akima_spl(new_df_full.X)
    })
ax.legend(['Исходные значения',
           'Сплайн Акимы',
           ])
ax.set_title('Акима сплайн', fontsize=25)
plt.show()
```