**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №1

по дисциплине «**Вычислительная математика**»

Автор: Баянов Равиль Динарович

Факультет: ПИиКТ

Группа: P3234

Преподаватель: Перл О. В.

Изображение выглядит как черный, темнота

Автоматически созданное описание

Санкт-Петербург, 2024

Оглавление

[Описание метода 3](#_Toc160999556)

[Блок-схема 4](#_Toc160999557)

[Исходный код метода на языке программирования Python 5](#_Toc160999558)

[Примеры работы программы 6](#_Toc160999559)

[Вывод 9](#_Toc160999560)

# Описание метода

**Метод наименьших модулей (МНМ) -** метод, использующийся в регрессионном анализе. Этот метод основан на уменьшении функции ошибки, где функцией ошибки является сумма абсолютных отклонений. С помощью этого метода мы модем построить прямую, максимально приближённую к нашим экспериментальным данным. Также стоит заметить, что этот метод сильно похож на метод наименьших квадратов (МКМ) и отличается от него только той самой функцией ошибки. Вот формула ошибки для этого метода:

Затем отыскав частные производные по x и y, мы сможем найти по методу поиска экстремума функции двух переменных минимум данной функции. И соответственно найдём параметры a и b для нашей аппроксимирующей прямой: y = a \* x + b. Построим прямую и получим аппроксимацию наших экспериментальных точек на графике.

# Блок-схема

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

# Исходный код метода на языке программирования Python

def linear\_regression(x\_axis, y\_axis):  
 a, b = 1, 1  
 e = 1e-6  
 epsilon = 0  
 n = len(x\_axis)  
 modules = [1] \* n  
 for \_ in range(10000):  
 a\_previous = a  
 b\_previous = b  
 sum\_x = sum(x \* m for x, m in zip(x\_axis, modules))  
 sum\_y = sum(y \* m for y, m in zip(y\_axis, modules))  
 sum\_xx = sum(x \* x \* m for x, m in zip(x\_axis, modules))  
 sum\_xy = sum(modules[i] \* x\_axis[i] \* y\_axis[i] for i in range(n))  
 sum\_modules = sum(modules)  
 if n \* sum\_xx - sum\_x \*\* 2 == 0:  
 epsilon += 1e-6  
 a = (sum\_modules \* sum\_xy - sum\_x \* sum\_y) / (sum\_modules \* sum\_xx - sum\_x \*\* 2 + epsilon)  
 b = (sum\_y - a \* sum\_x) / sum\_modules  
 for i in range(n):  
 modules[i] = 1 / (abs(y\_axis[i] - b - a \* x\_axis[i]) + e)  
 if abs(a\_previous - a) < e and abs(b\_previous - b) < e:  
 break  
 return a, b  
  
  
def approximate\_linear\_least\_modules(x\_axis, y\_axis):  
 if len(x\_axis) == 0:  
 return 0  
 params = linear\_regression(x\_axis, y\_axis)  
 deviations = [abs(params[0] \* x\_axis[i] + params[1] - y\_axis[i]) for i in range(len(x\_axis))]  
 max\_deviation = max(deviations)  
 return max\_deviation

# Примеры работы программы

1)

1. X = [1]
2. Y = [2]

Изображение выглядит как снимок экрана, Шрифт, часы, число

Автоматически созданное описание

Изображение выглядит как линия, Параллельный, График, прямоугольный

Автоматически созданное описание

2)

1. X = [2, 3, 4, 5]
2. Y = [5, 6, 7, 9]

Изображение выглядит как текст, часы, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как линия, текст, График, число

Автоматически созданное описание

3)

1. X = [-1, 100, 0]
2. Y = [3.5, 50, -199.89]

Изображение выглядит как текст, Шрифт, часы, снимок экрана

Автоматически созданное описание

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

4)

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Изображение выглядит как текст, линия, График, число

Автоматически созданное описание

5)

Изображение выглядит как текст, Шрифт, снимок экрана, часы

Автоматически созданное описание

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

# Вывод

При выполнении данной лабораторной работы я научился аппроксимировать с помощью метода наименьших модулей и написал алгоритм, реализующий этот метод на Python.

Заметим, что МНМ более устойчив к выбросам в отличие от МНК. Так как мы минимизируем сумму абсолютных отклонений, а не квадратичных отклонений. Также МНМ уместнее при распределении ошибок по Лапласу, в отличие от МНК, который более уместен при нормальном распределении. К сожалению, метод показывает не всегда идеальную аппроксимацию, например, в случаях, когда точки сильно разбросаны друг от друга.

Асимптотика алгоритма: мы видим двойной цикл, именно поэтому сложность алгоритма будет O(10000 \* N), так как количество итераций сильно больше чем кол-во входных значений, то грубо возьмём O(N \* N)

Видим, что ошибка численного метода при отсутствии шумов крайне минимальна. А также заметим, что метод применим почти всегда, особенно когда в наборе данных имеются выбросы, так как этот метод к ним очень даже устойчив.