МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



Лабораторная работа №1

**по дисциплине: «Методы обработки данных (классические байевские фильтрв)»**

**«Метод наименьших квадратов»**

Выполнил: Проверил:

Студент гр. «*АПИМ-224*», «*АВТФ*» Доцент Разуваев Владислав ВалерьевичГошко Е. Ю.

« » 20 г. « » 20 г.

(подпись) (подпись)

Новосибирск 2025

**Содержание**

[Цель работы 3](#_Toc208340598)

[Задание 3](#_Toc208340599)

[Ход работы 3](#_Toc208340600)

[Вывод 10](#_Toc208340601)

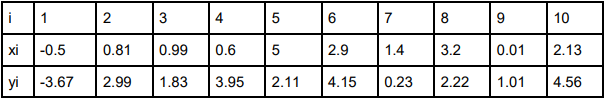
# Цель работы

Изучить метод наименьших квадратов и реализовать его.

# Задание

Дан набор точек, который получен зашумлением некоторой прямой с исходными коэффициентами. Используя метод наименьших квадратов, необходимо реализовать функцию solution, которая будет возвращать исходные коэффициенты прямой (При написании программы нельзя использовать встроенную в numpy функцию numpy.linalg.lstsq).

Методом наименьших квадратов посчитайте расстояние между функцией y = x2 + 1 и набором точек:



Назовите основное отличие интерполяции точек от их аппроксимации.

Для чего нужен resampling?

# Ход работы

Задание 1.

Пусть даны (xi, yi) - координаты точек из заданного набора.   
y = ax + b – прямая, которую мы хотим найти нам нужно минимизировать сумму квадратов расстояний, n - количество точек.

*Листинг 1 – Функция для генерации набора точек прямой*

import numpy as np

import random

import matplotlib.pyplot as plt

import math

def generate\_Noisy\_Linear\_Data(a, b, data\_size, noise\_sigma, random\_state):

x = np.arange(0, data\_size, 1.)

mu = 0.0

random.seed(random\_state)

noise = np.empty((data\_size, 1))

y = np.empty((data\_size, 1))

for i in range(data\_size):

noise[i] = random.gauss(mu, noise\_sigma)

y[i] = b + a\*x[i] + noise[i]

return x, y

В идеале у нас должна быть верна система уравнений c постоянными коэффициентами. Запишем систему в матричном виде:

Где:

ошибка(погрешность)

Нужно минимизировать погрешности, а точнее их квадраты т.к. e – вектор, то нужно минимизировать скалярное произведение , а именно частная производная скалярного произведения по параметрам должна равнятся нулю: .

Для нахождения экстремума приравняем выражение к нулю и найдем х:

Реализуем функцию для представления матриц и проведения соответствующих математических операций над ними:

*Листинг 2 – Функция нахождения решения по МНК для случая прямой*

def solution(x, y):

# Создаём матрицу A: первый столбец — x, второй — единицы

A = np.column\_stack((x, np.ones(len(x))))

B = np.array(y)

# Вычисляем A^T \* A

ATA = np.dot(A.T, A)

# Вычисляем обратную матрицу (ATA)^{-1}

ATA\_inv = np.linalg.inv(ATA)

# Вычисляем (ATA)^{-1} \* A^T

ATA\_inv\_AT = np.dot(ATA\_inv, A.T)

# Вычисляем коэффициенты: (ATA)^{-1} \* A^T \* B

res = np.dot(ATA\_inv\_AT, B)

return res

Зададим уранение прямой с параметрами a = 4, b = -10 и сгенерируем 50 точек (случайная величина, распределённая по закону Гаусса с фиксированным значением среднеквадратического отклонения (СКО) равной 10, нулевым математическим ожиданием (МО) и с автокорреляционной функцией в виде дельта-функции).

*Листинг 3 – Код первого задания*

import numpy as np

import random

import matplotlib.pyplot as plt

import math

def generate\_Noisy\_Linear\_Data(a, b, data\_size, noise\_sigma, random\_state):

x = np.arange(0, data\_size, 1.)

mu = 0.0

random.seed(random\_state)

noise = np.empty((data\_size, 1))

y = np.empty((data\_size, 1))

for i in range(data\_size):

noise[i] = random.gauss(mu, noise\_sigma)

y[i] = b + a\*x[i] + noise[i]

return x, y

def predict(a, b, xi):

return [a \* val + b for val in xi]

def solution(x, y):

# Создаём матрицу A: первый столбец — x, второй — единицы

A = np.column\_stack((x, np.ones(len(x))))

B = np.array(y)

# Вычисляем A^T \* A

ATA = np.dot(A.T, A)

# Вычисляем обратную матрицу (ATA)^{-1}

ATA\_inv = np.linalg.inv(ATA)

# Вычисляем (ATA)^{-1} \* A^T

ATA\_inv\_AT = np.dot(ATA\_inv, A.T)

# Вычисляем коэффициенты: (ATA)^{-1} \* A^T \* B

res = np.dot(ATA\_inv\_AT, B)

return res

def main\_Task1():

x\_linear, y\_linear = generate\_Noisy\_Linear\_Data(4, -10, 50, 10, 42)

a, b = solution(x\_linear, y\_linear)

print(a, b)

y\_linear\_pred = predict(a, b, x\_linear)

plt.plot(x\_linear, y\_linear, 'o', label = "Истинные значения")

plt.legend(loc = "best", fontsize=12)

plt.xlabel('x (порядковый номер измерения)', fontsize=14)

plt.ylabel('y (Значение измерения)', fontsize=14)

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

main\_Task1()

pass

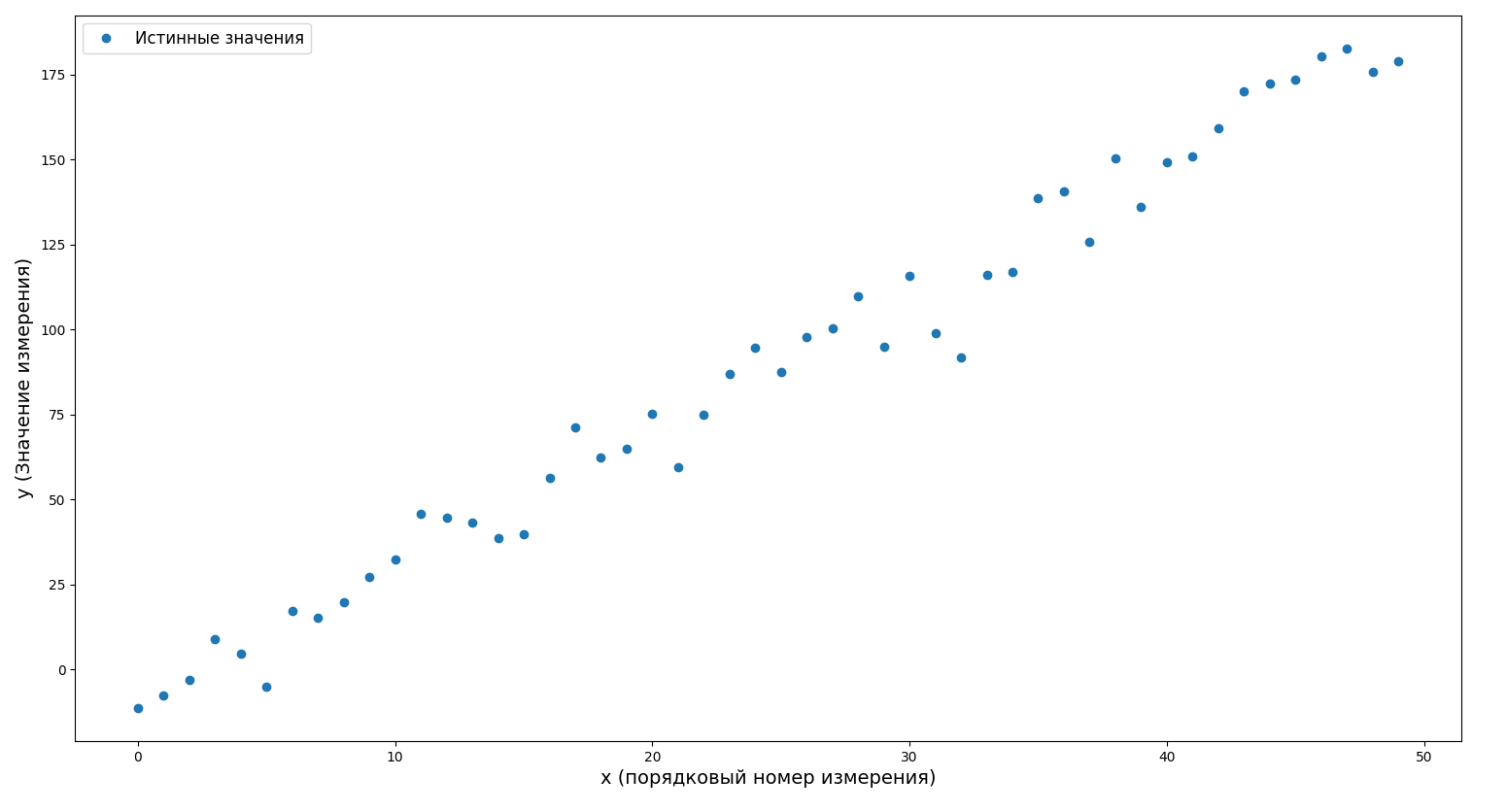


Рисунок 1.1 – Сгенерированные точки (50 измерений)

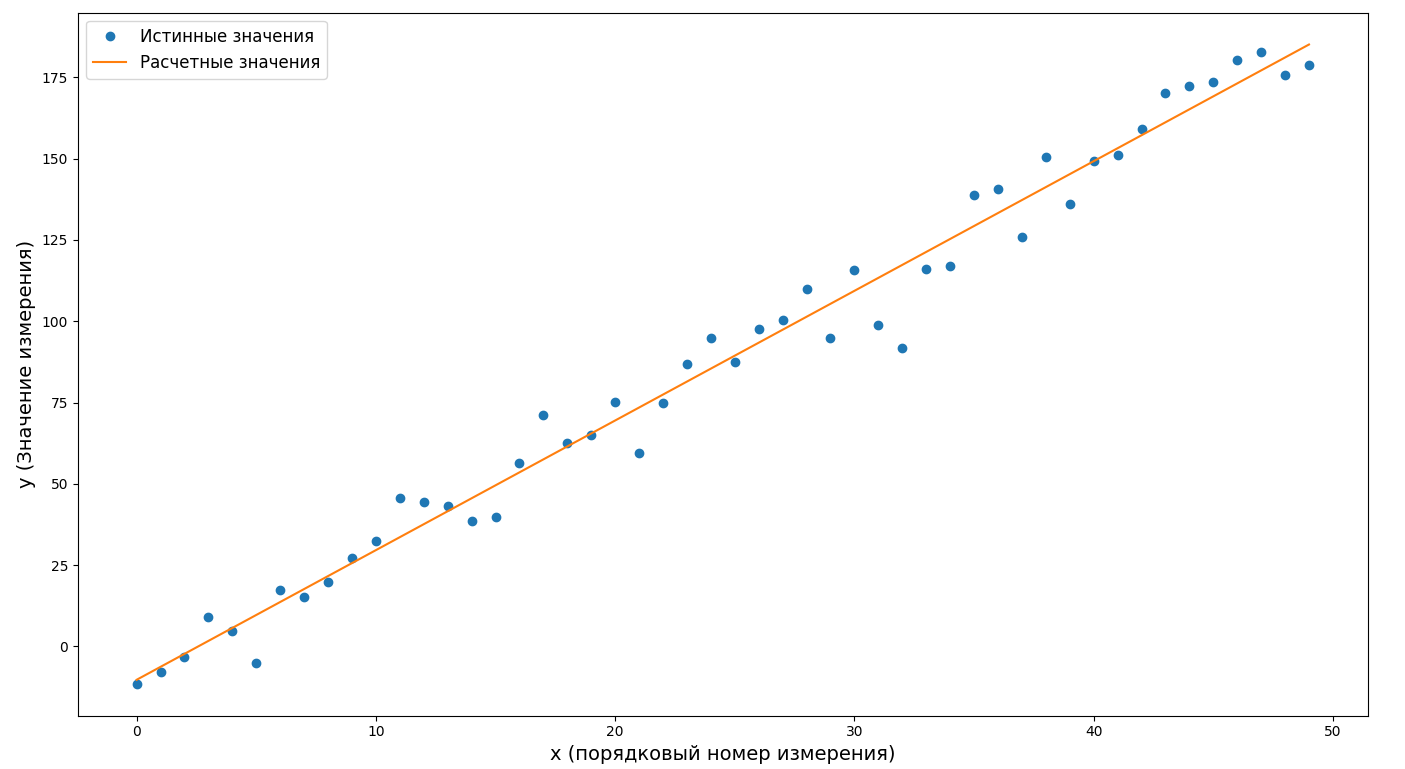


Рисунок 1.2 – Апроксиммированная прямая

Расчетные значения параметров a = 3.987, b = -10.268. Разброс с заданными параметрами не превышает 3% для параметра b и 0,4% для параметра a.

Задание 2.

Для нахождения расстояния между функцией f = x2 + 1 и набором точек необходимо найти величину R:

Для начала необходимо произвести сортировку точек по порядку возрастанию момента измерения, то есть оси абсцисс. После чего найти сумму квадратов разницы между измеренными точками и расчетными и, взять корень этой суммы.

*Листинг 4 – Код второго задания*

import numpy as np

import random

import matplotlib.pyplot as plt

import math

def main\_Task2():

x = [-0.5, 0.81, 0.99, 0.6, 5., 2.9, 1.4, 3.2, 0.01, 2.13]

y = [-3.67, 2.99, 1.83, 3.95, 2.11, 4.15, 0.23, 2.22, 1.01, 4.56]

# Объединяем x и y в список кортежей

points = list(zip(x, y))

# Сортируем по первому элементу кортежа (координате x)

points\_sorted = sorted(points, key=lambda point: point[0])

# Разделяем обратно на x\_sorted и y\_sorted

x, y = zip(\*points\_sorted)

y\_porabol = [i\*i + 1 for i in x]

res = 0

for i in range(len(x)):

res += (y\_porabol[i] - y[i])\*\*2

res = math.sqrt(res)

print(res)

plt.plot(x, y, 'o', label = "Набор точек")

plt.plot(x, y\_porabol, label = "Порабола")

plt.legend(loc = "best", fontsize=12)

plt.xlabel('x (порядковый номер измерения)', fontsize=14)

plt.ylabel('y (Значение измерения)', fontsize=14)

plt.show()

pass

if \_\_name\_\_ == "\_\_main\_\_":

main\_Task2()

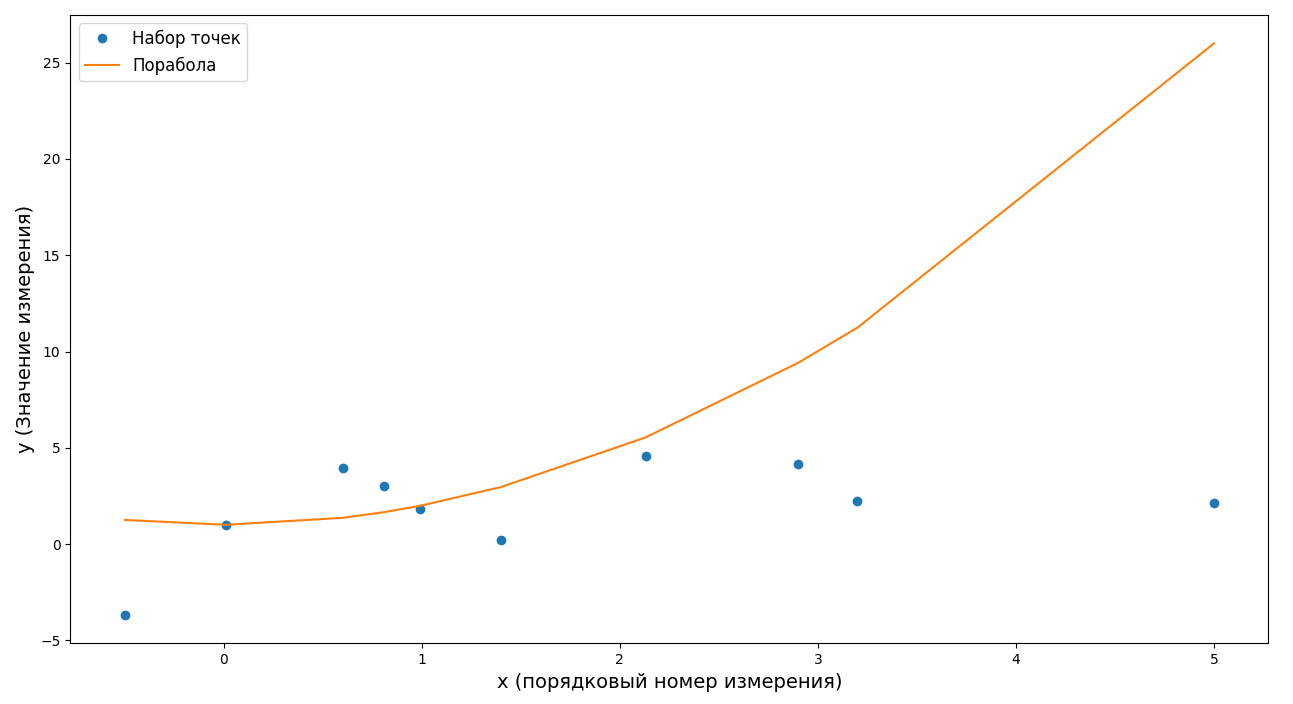


Рисунок 2.1 – График пораболы и измеренные точки

Расчитанное расстоняние составило 26,85.

# Вывод

Расчетные значения параметров для первого задания: a = 3.987, b = -10.268. Разброс с заданными параметрами не превышает 3% для параметра b и 0,4% для параметра a. Из чего можно сделать вывод, что МНК справляется с задачей нахождения параметров линейных моделей с высокой точностью. При уменьшении количества точек до 10 расхождение составило 10,5% и 1,85% соответственно. Точность нахождение углового коэффициента a намного выше чем точность нахождение коэффициента смещения b.

Расчитанное расстояние для второго задания составило 26.85, что достаточно много. Из графика (рисунок 2.1) видно, что происходит серьезное расхождение при величине х больше 3. Это рассхождение вносит основной вклад в расстояние, так как этот параметр определяется суммой квадрата разности значений измерения и сравниваемой функции, то большая величина разности будет вносить больший вклад в итоговую сумму (чем больше расстояние между точками, тем больший вклад в итоговое значение расстояния они будет вносить).

Resampling (ресемплирование) - это метод многократного отбора выборок из имеющихся данных с целью получения дополнительной информации о модели, оценки точности и неопределенности. Эта методика часто используется при малом объеме данных или для оценки вариабельности статистических оценок. Ресемплирование позволяет повысить надежность модели, исследовать ее стабильность и производительность, а также справляться с проблемами несбалансированных данных с помощью техник, таких как бутстрап и кросс-валидация.

Основное отличие интерполяции от аппроксимации заключается в том, что интерполяция предполагает построение функции, которая проходит точно через все заданные точки. Аппроксимация же допускает некоторую погрешность, и функция "прилизывает" или приближает набор точек, но не обязательно проходит через все из них. Интерполяцию обычно применяют при малом объеме точных данных, а аппроксимацию - когда данные зашумлены или содержат ошибки, и нужно найти функцию, максимально близкую к этим данным с точки зрения некоторого критерия, например, метода наименьших квадратов.