Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №4**

**По вычислительной математике**

**Вариант: 3**

Выполнил:

студент 2 курса

Батманов Даниил Евгеньевич

Группа: Р3207

Принял:

Рыбаков Степан Дмитриевич

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

**Оглавление**

[Задание 3](#_Toc162833322)

[Рабочие формулы 3](#_Toc162833323)

[Часть I: вычислительная 4](#_Toc162833324)

[Часть II: программная 5](#_Toc162833325)

[Вывод 13](#_Toc162833326)

# 

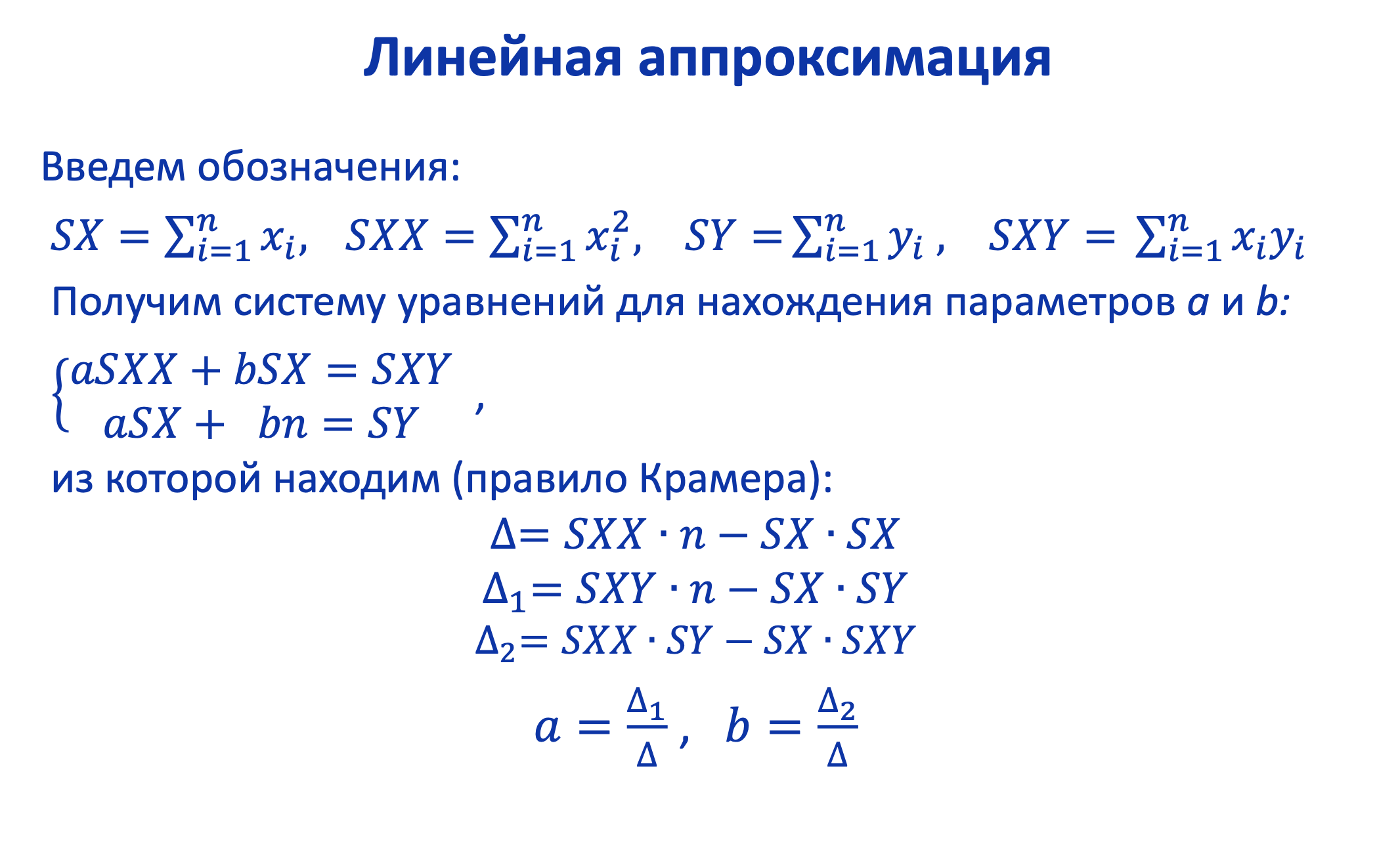
# 

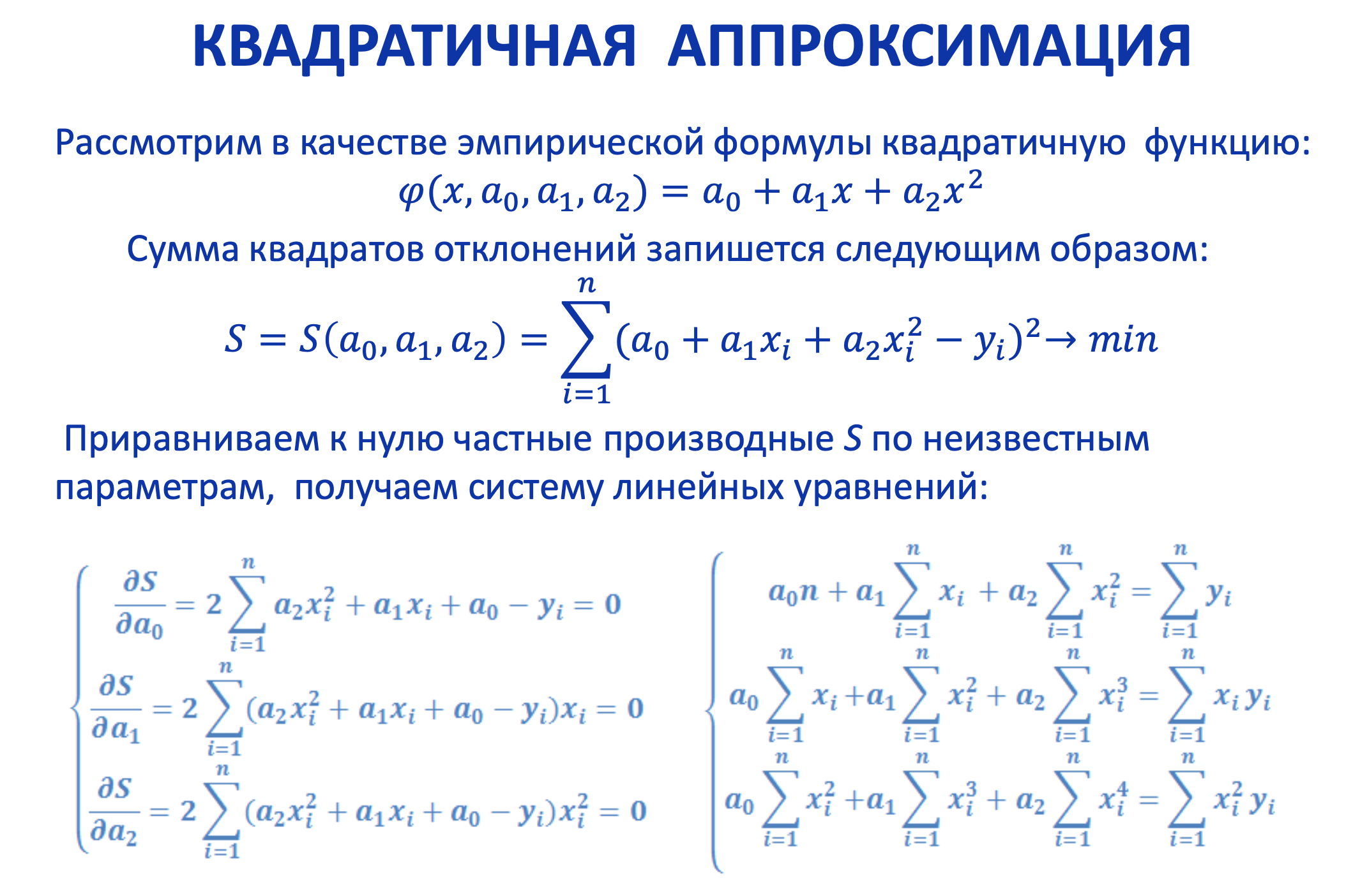
# 

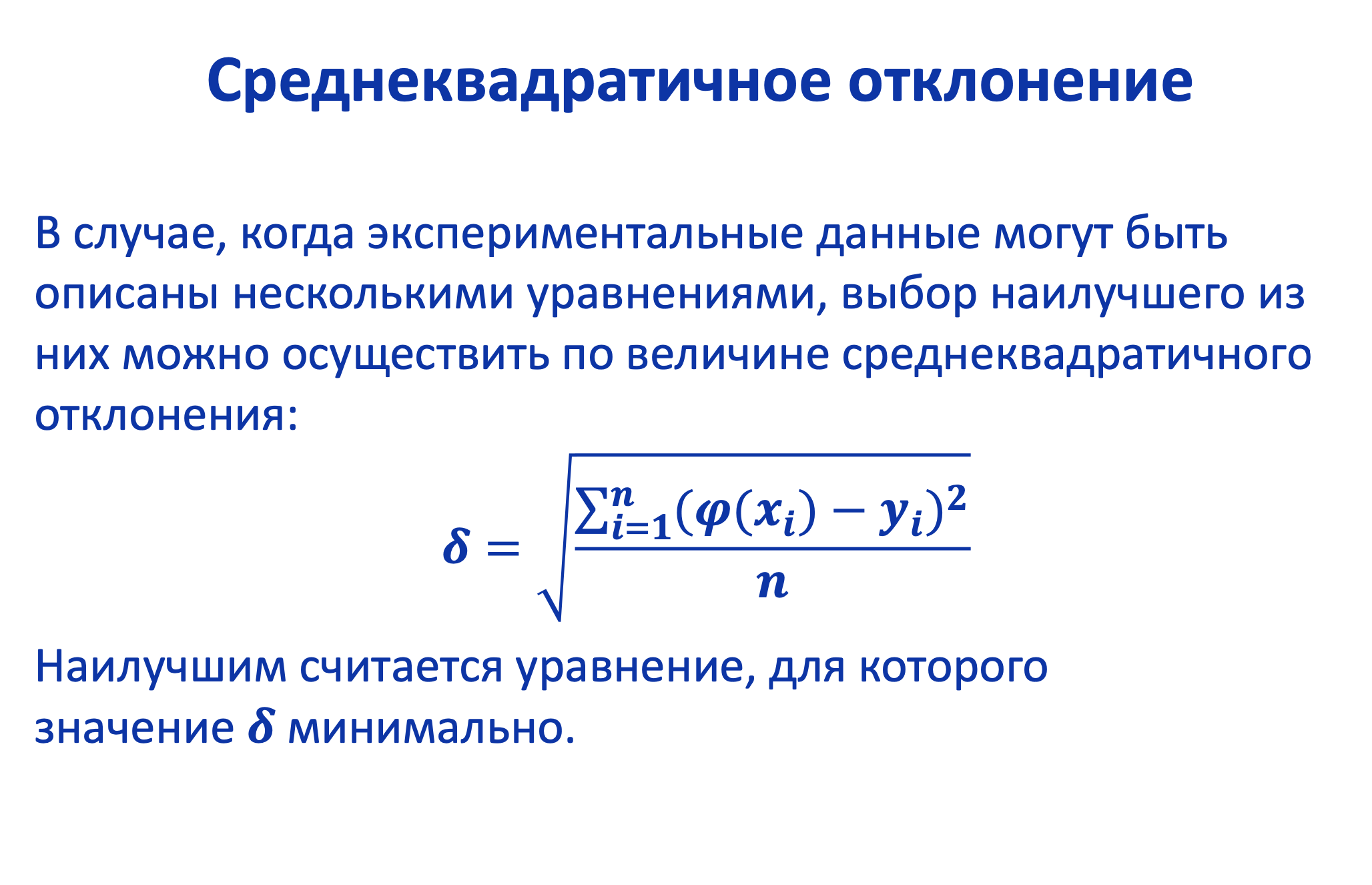
# Задание

Цель работы: найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

# Рабочие формулы

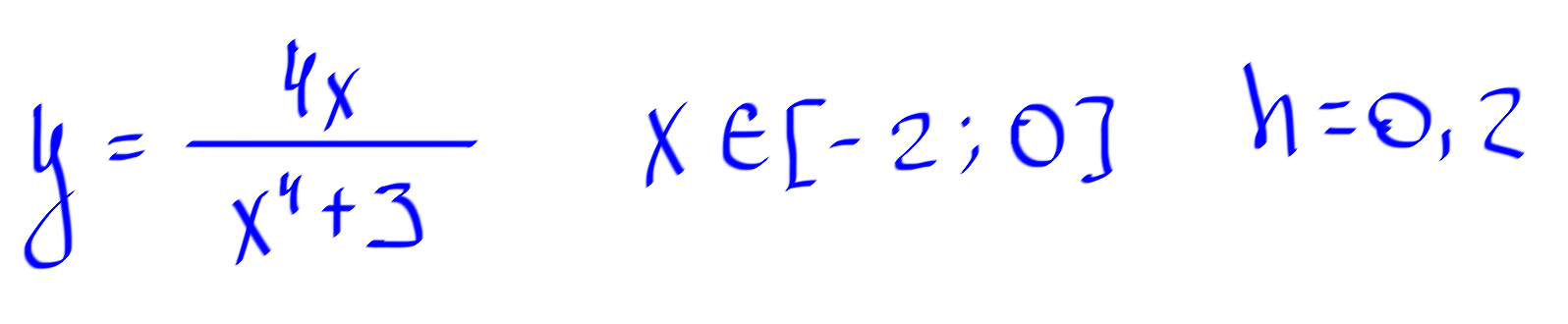






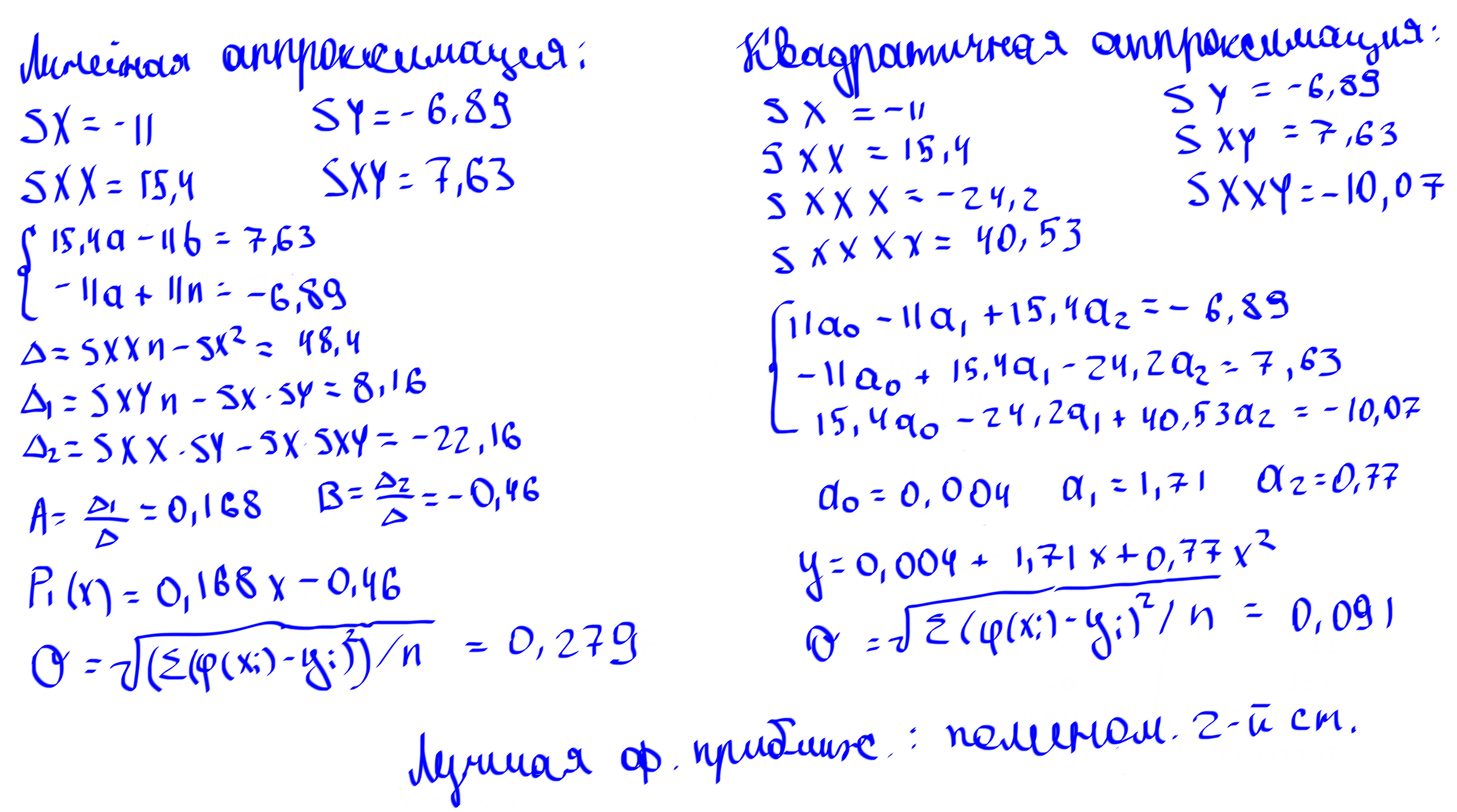
# Часть I: вычислительная

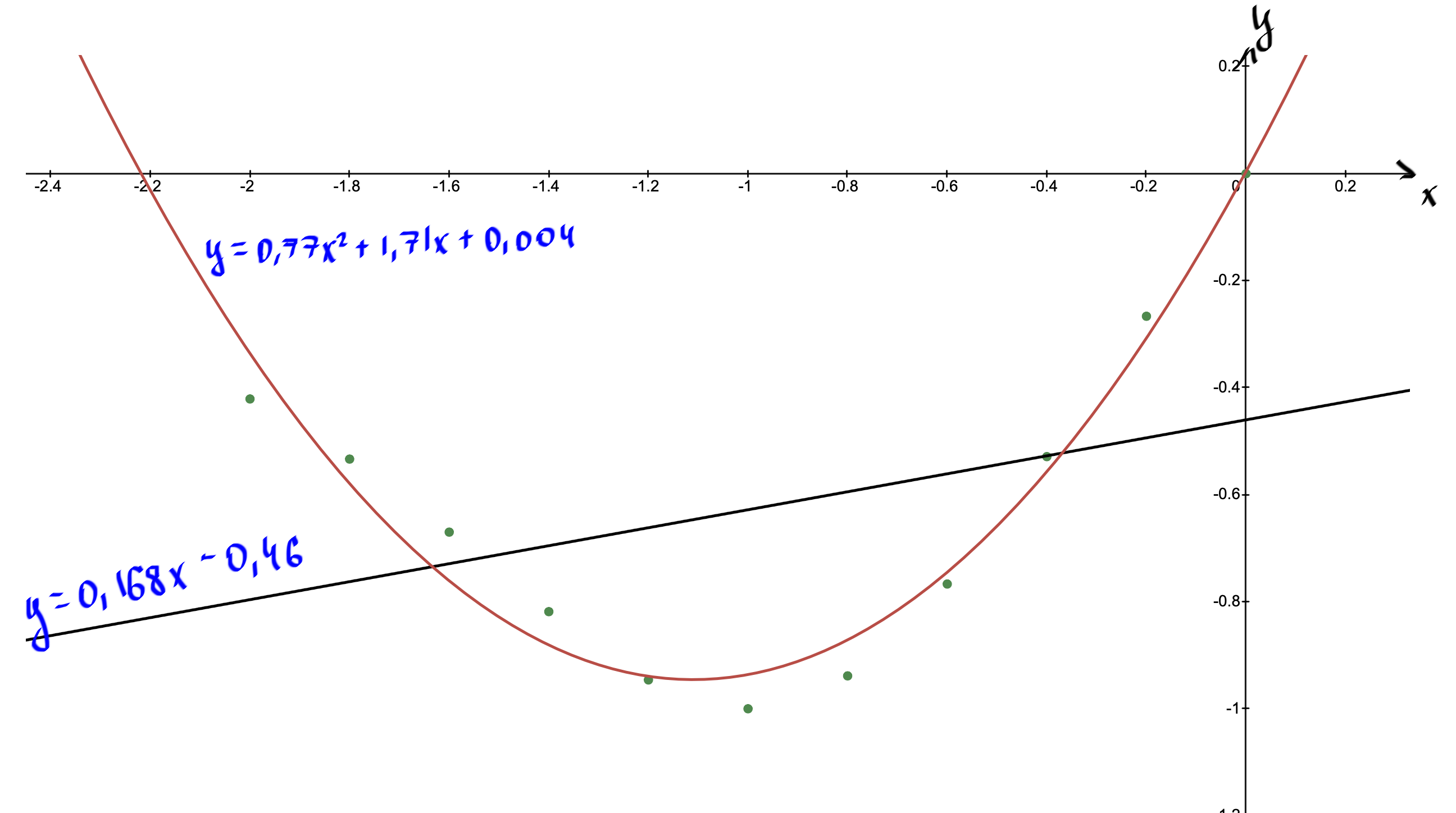
***Исходная функция:***



***Таблица табулирования заданной функции:***

|  |  |
| --- | --- |
| **X** | **Y** |
| -2 | -0,4210526 |
| -1,8 | -0,5334282 |
| -1,6 | -0,6699045 |
| -1,4 | -0,818522 |
| -1,2 | -0,9460738 |
| -1 | -1 |
| -0,8 | -0,9385265 |
| -0,6 | -0,7668712 |
| -0,4 | -0,5288207 |
| -0,2 | -0,2665245 |
| 0 | 0 |



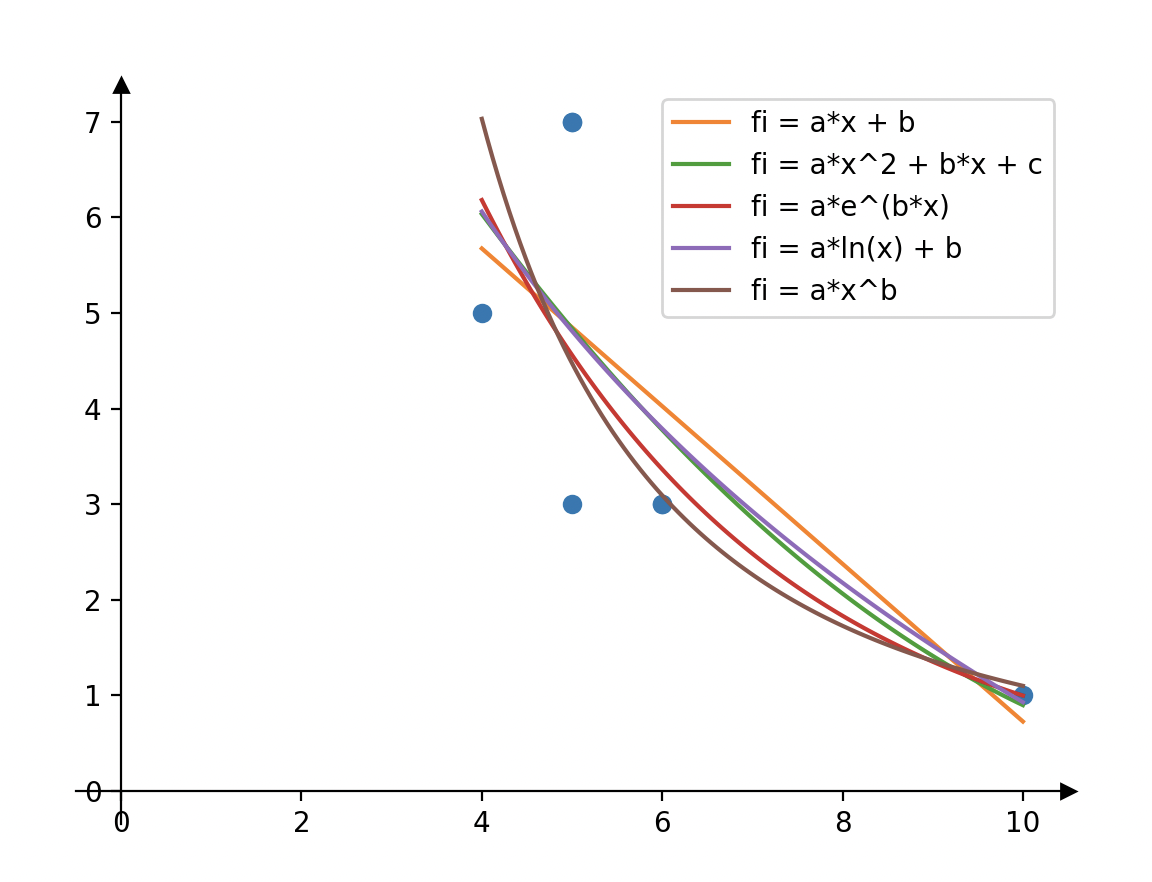


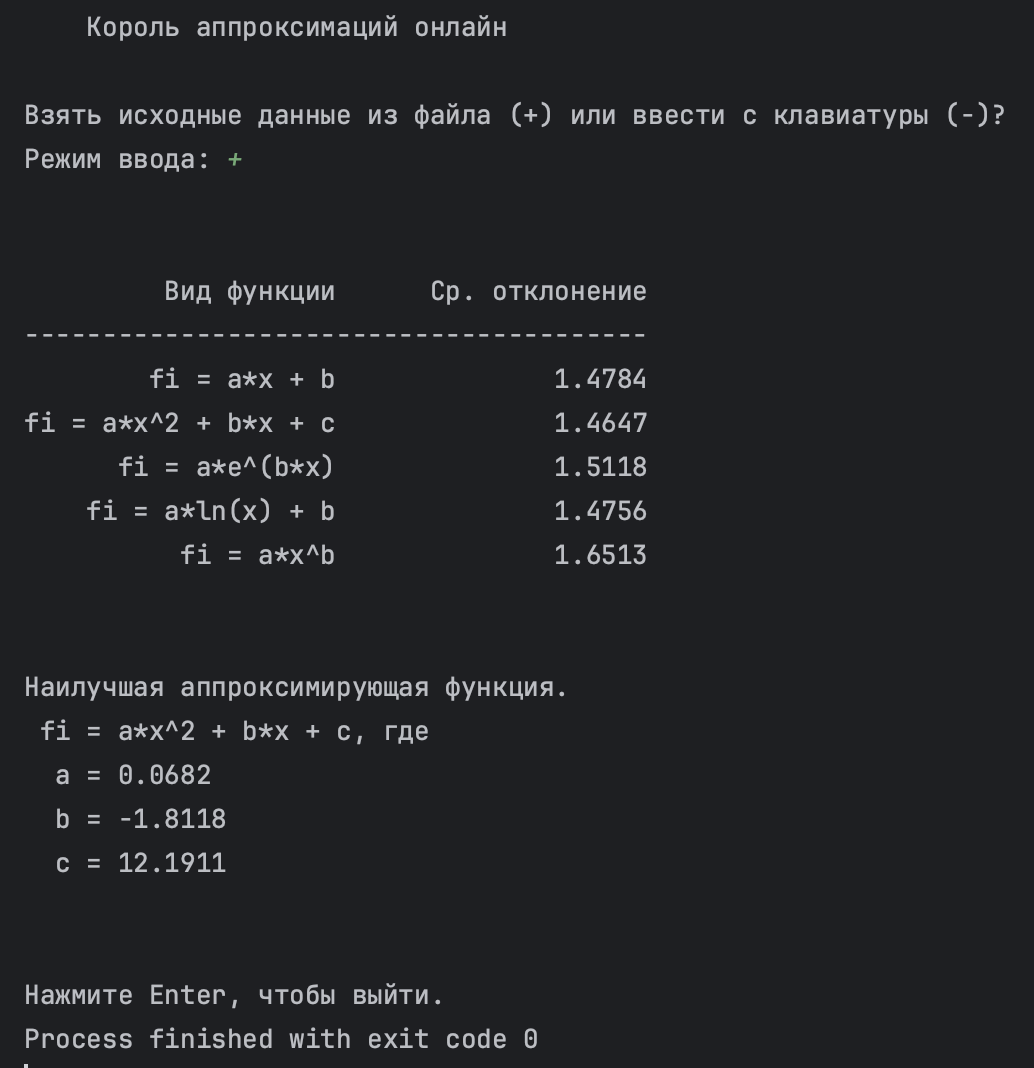
# Часть II: программная

***Исходный код:***

import numpy as np  
import matplotlib.pyplot as plt  
from math import log, exp, sqrt  
  
FILE\_IN = "input.txt"  
FILE\_OUT = "output.txt"  
  
  
# найти минор элемента матрицы  
def solve\_minor(matrix, i, j):  
 n = len(matrix)  
 return [[matrix[row][col] for col in range(n) if col != j] for row in range(n) if row != i]  
  
  
# найти определитель матрицы  
def solve\_det(matrix):  
 n = len(matrix)  
 if n == 1:  
 return matrix[0][0]  
 det = 0  
 sgn = 1  
 for j in range(n):  
 det += sgn \* matrix[0][j] \* solve\_det(solve\_minor(matrix, 0, j))  
 sgn \*= -1  
 return det  
  
  
# найти меру отклонения  
def calc\_s(dots, f):  
 n = len(dots)  
 x = [dot[0] for dot in dots]  
 y = [dot[1] for dot in dots]  
  
 return sum([(f(x[i]) - y[i]) \*\* 2 for i in range(n)])  
  
  
# найти среднеквадратичное отклонение  
def calc\_stdev(dots, f):  
 n = len(dots)  
  
 return sqrt(calc\_s(dots, f) / n)  
  
  
# линейная аппроксимация  
def lin\_func(dots):  
 data = {}  
  
 n = len(dots)  
 x = [dot[0] for dot in dots]  
 y = [dot[1] for dot in dots]  
  
 sx = sum(x)  
 sx2 = sum([xi \*\* 2 for xi in x])  
 sy = sum(y)  
 sxy = sum([x[i] \* y[i] for i in range(n)])  
  
 d = solve\_det([[sx2, sx],  
 [sx, n]])  
 d1 = solve\_det([[sxy, sx],  
 [sy, n]])  
 d2 = solve\_det([[sx2, sxy],  
 [sx, sy]])  
  
 try:  
 a = d1 / d  
 b = d2 / d  
 except ZeroDivisionError:  
 return None  
 data['a'] = a  
 data['b'] = b  
  
 f = lambda z: a \* z + b  
 data['f'] = f  
  
 data['str\_f'] = "fi = a\*x + b"  
  
 data['s'] = calc\_s(dots, f)  
  
 data['stdev'] = calc\_stdev(dots, f)  
  
 return data  
  
  
# квадратичная аппроксимация  
def sqrt\_func(dots):  
 data = {}  
  
 n = len(dots)  
 x = [dot[0] for dot in dots]  
 y = [dot[1] for dot in dots]  
  
 sx = sum(x)  
 sx2 = sum([xi \*\* 2 for xi in x])  
 sx3 = sum([xi \*\* 3 for xi in x])  
 sx4 = sum([xi \*\* 4 for xi in x])  
 sy = sum(y)  
 sxy = sum([x[i] \* y[i] for i in range(n)])  
 sx2y = sum([(x[i] \*\* 2) \* y[i] for i in range(n)])  
  
 d = solve\_det([[n, sx, sx2],  
 [sx, sx2, sx3],  
 [sx2, sx3, sx4]])  
 d1 = solve\_det([[sy, sx, sx2],  
 [sxy, sx2, sx3],  
 [sx2y, sx3, sx4]])  
 d2 = solve\_det([[n, sy, sx2],  
 [sx, sxy, sx3],  
 [sx2, sx2y, sx4]])  
 d3 = solve\_det([[n, sx, sy],  
 [sx, sx2, sxy],  
 [sx2, sx3, sx2y]])  
  
 try:  
 c = d1 / d  
 b = d2 / d  
 a = d3 / d  
 except ZeroDivisionError:  
 return None  
 data['c'] = c  
 data['b'] = b  
 data['a'] = a  
  
 f = lambda z: a \* (z \*\* 2) + b \* z + c  
 data['f'] = f  
  
 data['str\_f'] = "fi = a\*x^2 + b\*x + c"  
  
 data['s'] = calc\_s(dots, f)  
  
 data['stdev'] = calc\_stdev(dots, f)  
  
 return data  
  
  
# экспоненциальная аппроксимация  
def exp\_func(dots):6  
 data = {}  
  
 n = len(dots)  
 x = [dot[0] for dot in dots]  
 y = []  
 for dot in dots:  
 if dot[1] <= 0:  
 return None  
 y.append(dot[1])  
  
 lin\_y = [log(y[i]) for i in range(n)]  
 lin\_result = lin\_func([(x[i], lin\_y[i]) for i in range(n)])  
  
 a = exp(lin\_result['b'])  
 b = lin\_result['a']  
 data['a'] = a  
 data['b'] = b  
  
 f = lambda z: a \* exp(b \* z)  
 data['f'] = f  
  
 data['str\_f'] = "fi = a\*e^(b\*x)"  
  
 data['s'] = calc\_s(dots, f)  
  
 data['stdev'] = calc\_stdev(dots, f)  
  
 return data  
  
  
# логарифмическая аппроксимация  
def log\_func(dots):  
 data = {}  
  
 n = len(dots)  
 x = []  
 for dot in dots:  
 if dot[0] <= 0:  
 return None  
 x.append(dot[0])  
 y = [dot[1] for dot in dots]  
  
 lin\_x = [log(x[i]) for i in range(n)]  
 lin\_result = lin\_func([(lin\_x[i], y[i]) for i in range(n)])  
  
 a = lin\_result['a']  
 b = lin\_result['b']  
 data['a'] = a  
 data['b'] = b  
  
 f = lambda z: a \* log(z) + b  
 data['f'] = f  
  
 data['str\_f'] = "fi = a\*ln(x) + b"  
  
 data['s'] = calc\_s(dots, f)  
  
 data['stdev'] = calc\_stdev(dots, f)  
  
 return data  
  
  
# степенная аппроксимация  
def pow\_func(dots):  
 data = {}  
  
 n = len(dots)  
 x = []  
 for dot in dots:  
 if dot[0] <= 0:  
 return None  
 x.append(dot[0])  
 y = []  
 for dot in dots:  
 if dot[1] <= 0:  
 return None  
 y.append(dot[1])  
  
 lin\_x = [log(x[i]) for i in range(n)]  
 lin\_y = [log(y[i]) for i in range(n)]  
 lin\_result = lin\_func([(lin\_x[i], lin\_y[i]) for i in range(n)])  
  
 a = exp(lin\_result['b'])  
 b = lin\_result['a']  
 data['a'] = a  
 data['b'] = b  
  
 f = lambda z: a \* (z \*\* b)  
 data['f'] = f  
  
 data['str\_f'] = "fi = a\*x^b"  
  
 data['s'] = calc\_s(dots, f)  
  
 data['stdev'] = calc\_stdev(dots, f)  
  
 return data  
  
  
# отрисовать графики полученных функций  
def plot(x, y, plot\_x, plot\_ys, labels):  
 ax = plt.gca()  
 ax.spines['left'].set\_position('zero')  
 ax.spines['bottom'].set\_position('zero')  
 ax.spines['right'].set\_color('none')  
 ax.spines['top'].set\_color('none')  
 ax.plot(1, 0, marker=">", ms=5, color='k',  
 transform=ax.get\_yaxis\_transform(), clip\_on=False)  
 ax.plot(0, 1, marker="^", ms=5, color='k',  
 transform=ax.get\_xaxis\_transform(), clip\_on=False)  
  
 plt.plot(x, y, 'o')  
 for i in range(len(plot\_ys)):  
 plt.plot(plot\_x, plot\_ys[i], label=labels[i])  
  
 plt.legend()  
 plt.show()  
  
  
# получить данные из файла  
def getdata\_file():  
 data = {'dots': []}  
  
 with open(FILE\_IN, 'rt', encoding='UTF-8') as fin:  
 try:  
 for line in fin:  
 current\_dot = tuple(map(float, line.strip().split()))  
 if len(current\_dot) != 2:  
 raise ValueError  
 data['dots'].append(current\_dot)  
 if len(data['dots']) < 2:  
 raise AttributeError  
 except (ValueError, AttributeError):  
 return None  
  
 return data  
  
  
# Получить данные с клавиатуры  
def getdata\_input():  
 data = {'dots': []}  
  
 print("\nВводите координаты через пробел, каждая точка с новой строки.")  
 print("Чтобы закончить, введите 'END'.")  
 while True:  
 try:  
 current = input().strip()  
 if current == 'END':  
 if len(data['dots']) < 2:  
 raise AttributeError  
 break  
 current\_dot = tuple(map(float, current.split()))  
 if len(current\_dot) != 2:  
 raise ValueError  
 data['dots'].append(current\_dot)  
 except ValueError:  
 print("Введите точку повторно - координаты некорректны!")  
 except AttributeError:  
 print("Минимальное количество точек - 2!")  
  
 return data  
  
  
def main():  
 print("\tКороль аппроксимаций онлайн")  
  
 print("\nВзять исходные данные из файла (+) или ввести с клавиатуры (-)?")  
 inchoice = input("Режим ввода: ")  
 while (inchoice != '+') and (inchoice != '-'):  
 print("Введите '+' или '-' для выбора способа ввода.")  
 inchoice = input("Режим ввода: ")  
  
 if inchoice == '+':  
 data = getdata\_file()  
 if data is None:  
 print("\nДанные в файле некорректны!")  
 print("Режим ввода переключен на ручной.")  
 data = getdata\_input()  
 else:  
 data = getdata\_input()  
  
 answers = []  
 temp\_answers = [lin\_func(data['dots']),  
 sqrt\_func(data['dots']),  
 exp\_func(data['dots']),  
 log\_func(data['dots']),  
 pow\_func(data['dots'])]  
 for answer in temp\_answers:  
 if answer is not None:  
 answers.append(answer)  
 else:  
 print("Error computing a function.")  
  
 print("\n\n%20s%20s" % ("Вид функции", "Ср. отклонение"))  
 print("-" \* 40)  
 for answer in answers:  
 print("%20s%20.4f" % (answer['str\_f'], answer['stdev']))  
  
 x = np.array([dot[0] for dot in data['dots']])  
 y = np.array([dot[1] for dot in data['dots']])  
 plot\_x = np.linspace(np.min(x), np.max(x), 100)  
 plot\_y = []  
 labels = []  
 for answer in answers:  
 plot\_y.append([answer['f'](x) for x in plot\_x])  
 labels.append(answer['str\_f'])  
 plot(x, y, plot\_x, plot\_y, labels)  
  
 best\_answer = min(answers, key=lambda z: z['stdev'])  
 print("\nНаилучшая аппроксимирующая функция.")  
 print(f" {best\_answer['str\_f']}, где")  
 print(f" a = {round(best\_answer['a'], 4)}")  
 print(f" b = {round(best\_answer['b'], 4)}")  
 if 'c' in best\_answer:  
 print(f" c = {round(best\_answer['c'], 4)}")  
 else:  
 print(" c = -")  
  
 input("\n\nНажмите Enter, чтобы выйти.")  
  
  
main()

***Вывод программы:***

******



# Вывод

В ходе выполнения данной лабораторной работы я научился находить функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.