Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа № 4

по дисциплине «Методы оптимизации»

Вариант: 3

Преподаватель: Селина Е. Г.

Выполнила: Вавилина Е. А.

Группа: P3230

Санкт-Петербург, 2025

Оглавление

Задание

Найти экстремум функции методами покоординатного спуска, градиентного спуска, наискорейшего спуска. Выполнить **3 итерации каждого метода вручную** и написать **программу** на одном из языков программирования. Точность: ε = 0.0001. Уравнение:

$$f(x_1, x_2) = x_1^2 + 3x_2^2 + 3x_1x_2 - x_1 - 2x_2 - 1$$

Начальная точка: $M_0 = (3; 3)$.

Подготовка

Заданная функция – эллиптический параболоид. Значение функции возрастает при отдалении аргументов от вершины (т.к. перед квадратами стоят положительные коэффициенты). См. график функции ниже

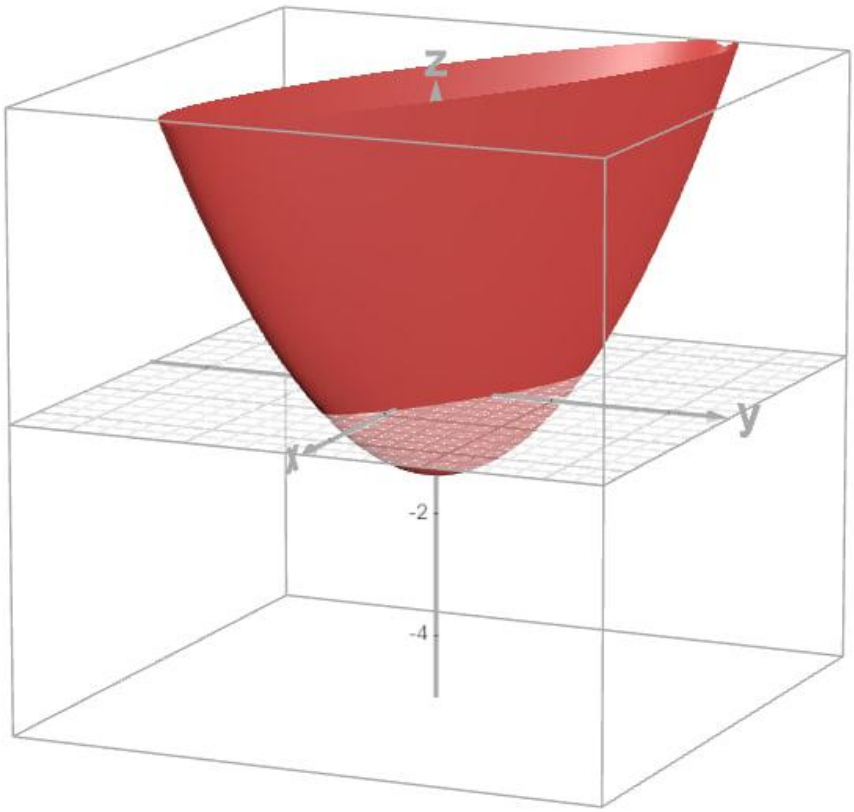
Следовательно, все методы будут искать не просто экстремум, а точку минимума.

Возьмем частные производные от каждой переменной:

$$\frac{\partial f}{\partial x_1} = 2x_1 + 3x_2 - 1$$

$$\frac{\partial f}{\partial x_2} = 6x_2 + 3x_1 - 2$$

$$f(3, 3) = 9 + 27 + 27 - 3 - 6 - 1 = 53$$



1. Покоординатный спуск

Цикл 1

Зафиксируем точку x_2. Тогда будем рассматривать $f(x_1, 3) = x_1^2 + 8x_1 - 20$

$\frac{df}{dx_1} = 2x_1 + 9 - 1 = 2x_1 + 8 \rightarrow x_1 = -4 \rightarrow M_1 = (-4;\ 3)$

Зафиксируем точку x_1. Тогда будем рассматривать $f(-4,\ x_2) = 3x_2^2 - 14x_2 + 19$

$\frac{df}{dx_2} = 6x_2 - 14 \rightarrow x_2 \approx 2.33 \rightarrow M_2 = (-4;\ 2.33)$

$f_{new}(x_1, x_2) \approx 2.66$

$|x - x_{new}| \approx 7.03 > 0.0001 \rightarrow$ продолжаем

Цикл 2

Зафиксируем точку x_2. Тогда будем рассматривать $f(x_1, 2.33)$

$\frac{df}{dx_1} = 2x_1 + 6 \rightarrow x_1 = -3 \rightarrow M_3 = (-3;\ 2.33)$

Зафиксируем точку x_1. Тогда будем рассматривать $f(-3,\ x_2)$

$\frac{df}{dx_2} = 6x_2 - 11 \rightarrow x_2 \approx 1.83 \rightarrow M_4 = (-3;\ 1.83)$

$f_{new}(x_1, x_2) \approx 0.92$

$|x - x_{new}| \approx 1.12 > 0.0001 \rightarrow$ продолжаем

Цикл 3

Зафиксируем точку x_2. Тогда будем рассматривать $f(x_1,\ 1.83\ )$

$\frac{df}{dx_1} = 2x_1 + 4.5 \rightarrow x_1 = -2.25 \rightarrow M_5 = (-2.25;\ 1.83)$

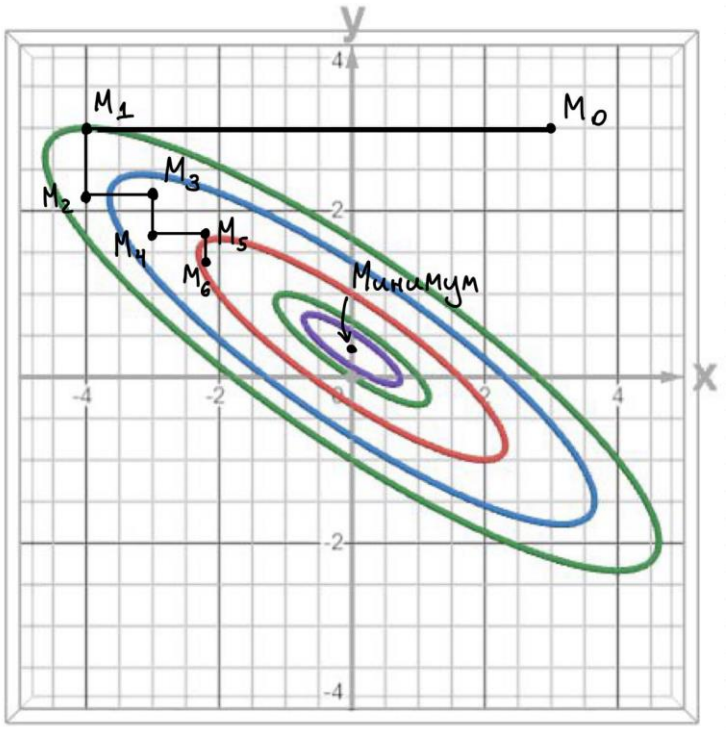
Зафиксируем точку x_1. Тогда будем рассматривать $f(-2.25,\ x_2)$

$\frac{df}{dx_2} = 6x_2 - 8.75 \rightarrow x_2 \approx 1.46 \rightarrow M_6 = (-2.25;\ 1.46)$

$f_{new}(x_1, x_2) \approx -0.07$

$|x - x_{new}| \approx 0.84 > 0.0001 \rightarrow$ продолжаем

Минимум не достигнут. Наложим точки на график функции, состоящий из линий уровня графика.



Результаты работы программы:

```
Метод покоординатного спуска
Начальная точка: [3. 3.]
Финальная точка: [-0.00022602  0.33344634]
Значение функции: -1.3333
Итераций выполнено: 35
++++++++++++++++++++++++++++++
```

2. Градиентный спуск

Т.к. мы ищем минимум – будем считать в направлении антиградиента.

На начальном этапе возьмем шаг $\lambda = 0.24$. Что бы максимально избавиться от риска того, что мы не сойдемся – возьмем коэффициент 0.8 и будем умножать на него каждый раз, когда функция перестанет убывать.

Шаг 1

Для точки M_0 рассчитаем шаг в направлении антиградиента

$$x_1 = 3 - 0.24 * \frac{\partial f}{\partial x_1}(3;3) = 3 - 0.24 * 14 = -0.36$$

$$x_2 = 3 - 0.24 * \frac{\partial f}{\partial x_2}(3;3) = 3 - 0.24 * 25 = -3$$

$M_1(-0.36;\ -3)$

$f - f_{new} = 53 - 35.73 = 17.27 > 0 \rightarrow$ двигаемся к минимуму и не достигли точности

Шаг 2

Для точки M_1 рассчитаем шаг в направлении антиградиента

$$x_1 = -0.36 - 0.24 * \frac{\partial f}{\partial x_1}(-0.36;\ -3) = -0.36 - 0.24 * -10.72 = 2.21$$

$$x_2 = -3 - 0.24 * \frac{\partial f}{\partial x_2}(-0.36;\ -3) = -3 - 0.24 * -21.08 = 2.06$$

$M_2(2.21;\ 2.06)$

$f - f_{new} = 35.73 - 23.96 = 11.77 > 0$
→ двигаемся к минимуму и не достигли точности

Шаг 3

Для точки M_1 рассчитаем шаг в направлении антиградиента

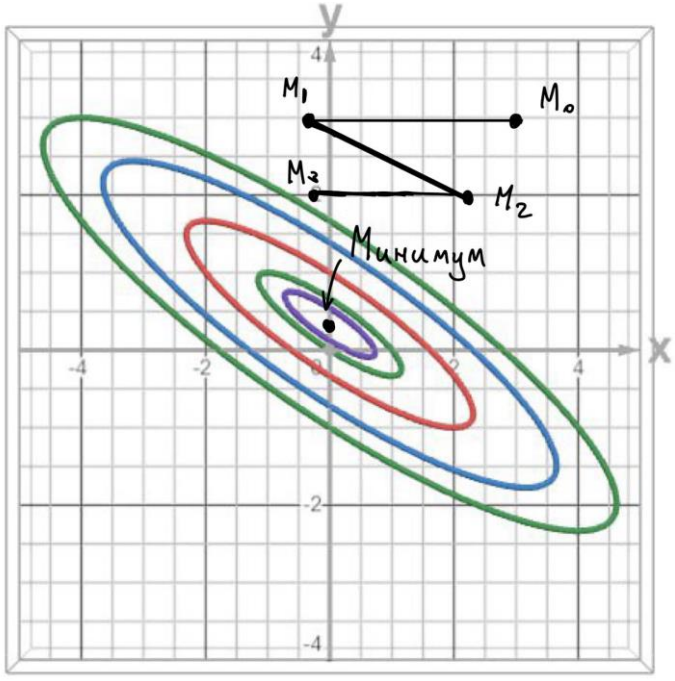
$$x_1 = 2.21 - 0.24 * \frac{\partial f}{\partial x_1}(-0.36;\ -3) = 2.21 - 0.24 * 9.6 = -0.09$$

$$x_2 = 2.06 - 0.24 * \frac{\partial f}{\partial x_2}(-0.36;\ -3) = 2.06 - 0.24 * 16.99 = -2.01$$

$M_3(-0.09;\ -2.01)$

$f - f_{new} = 23.96 - 15.93 = 8.03 > 0$ → двигаемся к минимуму и не достигли точности

Минимум не достигнут. Наложим точки на график функции, состоящий из линий уровня графика.



Результаты работы программы:

```
Метод градиентного спуска
Начальная точка: [3. 3.]
Финальная точка: [0.03584325 0.31840945]
Значение функции: -1.3330
Итераций выполнено: 36
++++++++++++++++++++++++++++
```

3. Наискорейший спуск

Попробуем составить универсальную формулу для расчета оптимального шага на каждом шаге. Пусть мы находимся в точке $M(x_1; x_2)$. В S уже учтем движение в направлении минимума и спрячем минус туда.

Тогда $f(\lambda) = (x_1 + \lambda * S_1)^2 + 3(x_2 + \lambda S_2)^2 + 3(x_1 + \lambda * S_1)(x_2 + \lambda S_2) - (x_1 + \lambda * S_1) - 2(x_2 + \lambda S_2) - 1$

И $\frac{df}{d\lambda} = (2S_1^2 + 6S_2^2 + 6S_1S_2)\lambda + (2x_1S_1 + 6x_2S_2 + 3x_1S_2 + 3x_2S_1 - S_1 - 2S_2) = 0$

Тогда $\lambda = -\frac{2x_1S_1+6x_2S_2+3x_1S_2+3x_2S_1-S_1-2S_2}{2S_1^2+6S_2^2+6S_1S_2}$

В дальнейшем будем пользоваться этой формулой.

Шаг 1

$$S = -\frac{grad(M_0)}{|grad(M_0)|} = -\frac{(14; 25)}{28.65} = (-0.49; -0.87)$$

Тогда: $x_1 = 3 - \lambda * 0.49$ и $x_2 = 3 - \lambda * 0.87$ и наименьшее значение функции при:

$$\lambda = -\frac{2 * 3 * -0.49 + 6 * 3 * -0.87 + 3 * 3 * -0.87 + 3 * 3 * -0.49 - (-0.49) - 2 * -0.87}{2 * (-0.49)^2 + 6 * (-0.87)^2 + 6 * -0.49 * -0.87} = 3.77$$

$x_{1_{new}} = 3 + 3.77 * -0.49 = 1.16$ и $x_{2_{new}} = 3 + 3.77 * -0.87 = -0.29 \rightarrow M_1(1.16;\ -0.29)$

$|grad(f_{old})| = 28.65 > 0.0001 \rightarrow$ продолжаем вычисление

Шаг 2

$$S = -\frac{grad(M_1)}{|grad(M_1)|} = -\frac{(0.45; -0.25)}{0.51} = (-0.87; 0.49)$$

Тогда: $x_1 = 3 - \lambda * 0.87$ и $x_2 = 3 + \lambda * 0.49$ и наименьшее значение функции при:

$\lambda = 1.31$

$x_{1_{new}} = 1.16 + 1.31 * -0.87 = 0.02$ и $x_{2_{new}} = -0.29 + 1.31 * 0.49 = 0.35 \rightarrow M_2(0.02;\ 0.35)$

$|grad(f_{old})| = 0.52 > 0.0001 \rightarrow$ продолжаем вычисление

Шаг 3

$$S = -\frac{grad(M_2)}{|grad(M_2)|} = -\frac{(0.09; 0.16)}{0.18} = (-0.49; -0.87)$$

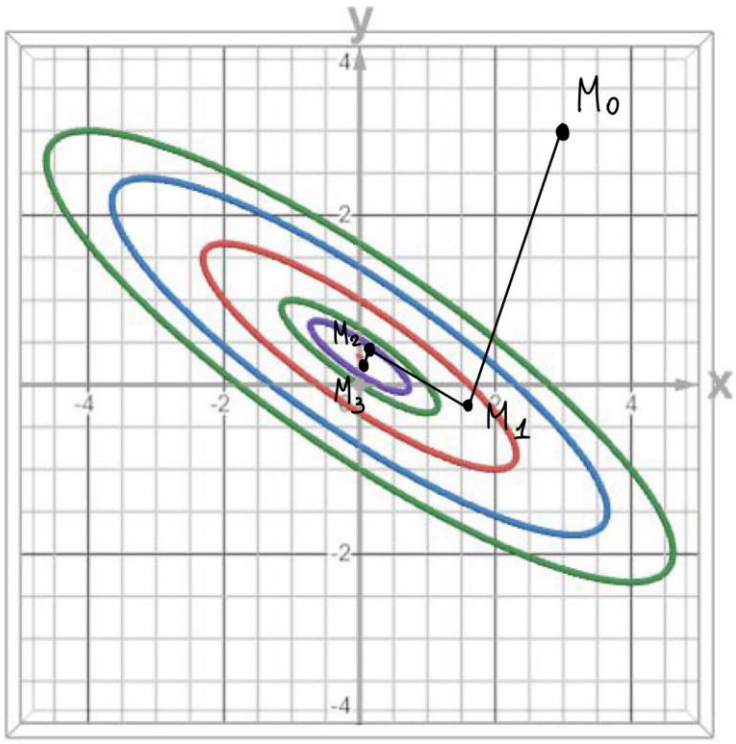
Тогда: $x_1 = 3 - \lambda * 0.49$ и $x_2 = 3 - \lambda * 0.87$ и наименьшее значение функции при:

$\lambda = 0.03$

$x_{1_{new}} = 0.02 + 0.03 * -0.49 = 0.007$ и $x_{2_{new}} = 0.35 + 0.03 * -0.87 = 0.33 \rightarrow$
$M_3(0.007;\ 0.33)$

$|grad(f_{old})| = 0.18 > 0.0001 \rightarrow$ продолжаем вычисление

Минимум не достигнут. Наложим точки на график функции, состоящий из линий уровня графика.



Результаты работы программы:

```
Метод наискорейшего спуска
Начальная точка: [3. 3.]
Финальная точка: [0.00004562 0.33330886]
Значение функции: -1.3333
Итераций выполнено: 5
++++++++++++++++++++++++++++++
```

4. Итоговый код

```
import numpy as np


def f(x):
    x1, x2 = x
    return x1 ** 2 + 3 * x2 ** 2 + 3 * x1 * x2 - x1 - 2 * x2 - 1


def df_dx1(x):
    x1, x2 = x
    return 2 * x1 + 3 * x2 - 1


def df_dx2(x):
    x1, x2 = x
    return 6 * x2 + 3 * x1 - 2


def gradient(x):
    return np.array([df_dx1(x), df_dx2(x)])


# Метод покоординатного спуска
def coordinate_descent(start, ep, max_iter=1000):
    x = np.array(start, dtype=float)
    history = [x.copy()]

    for _ in range(max_iter):
        x_old = x.copy()

        # Минимизация по x1
        x[0] = (1 - 3 * x[1]) / 2

        # Минимизация по x2
        x[1] = (2 - 3 * x[0]) / 6

        history.append(x.copy())

        if np.linalg.norm(x - x_old) < ep:
            break

    return np.array(history)


# Метод градиентного спуска
def gradient_descent(start, ep, max_iter=1000, alpha=0.24, decay=0.8):
    x = np.array(start, dtype=float)
    history = [x.copy()]
    current_alpha = alpha

    for _ in range(max_iter):
        x_old = x.copy()
        grad = gradient(x)

        # Делаем шаг
        x_new = x - current_alpha * grad

        # Проверяем улучшение
        if f(x_new) < f(x):
            x = x_new
            current_alpha = alpha
        else:
```

```
            current_alpha *= decay

        history.append(x.copy())

        # Проверка условия останова
        if abs(f(x_new)-f(x_old)) < ep:
            break

    return np.array(history)


# Метод наискорейшего спуска
def steepest_descent(start, ep, max_iter=1000):
    x = np.array(start, dtype=float)
    history = [x.copy()]

    for _ in range(max_iter):
        x_old = x.copy()
        grad = gradient(x)

        grad_norm = np.linalg.norm(grad)
        if grad_norm < ep:
            break

        S = -grad / grad_norm

        S1, S2 = S
        x1, x2 = x

        numerator = (2 * x1 * S1 + 6 * x2 * S2 + 3 * x1 * S2 + 3 * x2 * S1 -
S1 - 2 * S2)
        denominator = (2 * S1 ** 2 + 6 * S2 ** 2 + 6 * S1 * S2)

        if abs(denominator) < 1e-10:
            break

        lam = -numerator / denominator

        x = x + lam * S
        history.append(x.copy())

    return np.array(history)


# Вывод результатов
def print_results(method_name, history):
    np.set_printoptions(suppress=True)
    print(f"\n{method_name}")
    print(f"Начальная точка: {history[0].round(4)}")
    print(f"Финальная точка: {history[-1]}")
    print(f"Значение функции: {f(history[-1]):.4f}")
    print(f"Итераций выполнено: {len(history) - 1}")
    print("+" * 30)


start_point = [3.0, 3.0]
epsilon = 0.0001

# Покоординатный спуск
cd_history = coordinate_descent(start_point, epsilon)
print_results("Метод покоординатного спуска", cd_history)

# Градиентный спуск
gd_history = gradient_descent(start_point, epsilon)
```

```
print_results("Метод градиентного спуска", gd_history)

# Наискорейший спуск
sd_history = steepest_descent(start_point, epsilon)
print_results("Метод наискорейшего спуска", sd_history)
```

Вывод

В этой лабораторной работе я научилась вычислять экстремум многомерной функции методом покоординатного спуска, градиентного спуска и наискорейшего спуска.