Отчёт ЛР 4 методы оптимизации

Чураков А А P3231, B-19

22 апреля 2025 г.

1 Постановка задачи

Найти минимум функции

$$f(x_1, x_2) = 2x_1^2 + 4x_2^2 - 5x_1x_2 + 11x_1 + 8x_2 - 3$$

методами:

- покоординатного спуска,
- градиентного спуска с фиксированным шагом,
- наискорейшего спуска (с поиском оптимального шага).

Точность остановки: $\varepsilon = 10^{-4}$. Начальное приближение:

$$x^{(0)} = (1, 1).$$

2 Метод покоординатного спуска

2.1 Ручные вычисления

Итерация 1. $M^0 = (1, 1)$.

Шаг 1: минимизация по x_1, при $x_2 = 1$.

$$f(x_1, 1) = 2x_1^2 + 6x_1 + 9, \quad \frac{df}{dx_1} = 4x_1 + 6 = 0 \Rightarrow x_1 = -1.5.$$

Промежуточная точка: $M^{1/2} = (-1.5,\ 1)$.

$$f(1, 1) = 17.0, \quad f(-1.5, 1) = 4.5 \quad \Rightarrow \text{уменьшение на } 12.5.$$

Шаг 2: минимизация по x_2, при $x_1 = -1.5$.

$$f(-1.5, x_2) = 4x_2^2 + 15.5x_2 - 15, \quad \frac{df}{dx_2} = 8x_2 + 15.5 = 0 \Rightarrow x_2 = -1.9375.$$

Новая точка: $M^1 = (-1.5,\ -1.9375)$.

$$f(-1.5, 1) = 4.5, \quad f(-1.5, -1.9375) \approx -30.0938 \quad \Rightarrow \text{уменьшение на } 34.5938.$$

Итерация 2. $M^1 = (-1.5,\ -1.9375)$.

Шаг 1: минимизация по x_1, при $x_2 = -1.9375$.

$$f(x_1, -1.9375) = 2x_1^2 + 20.6875x_1 - 3.4844, \quad \frac{df}{dx_1} = 4x_1 + 20.6875 = 0 \Rightarrow x_1 = -5.171875.$$

Промежуточная точка: $M^{2/2} = (-5.171875,\ -1.9375)$.

$$f(-1.5, -1.9375) \approx -30.0938, \quad f(-5.171875, -1.9375) \approx -140.8817 \quad \Rightarrow \text{уменьшение на } 110.7879.$$

Шаг 2: минимизация по x_2, при $x_1 = -5.171875$.

$$f(-5.171875, x_2) = 4x_2^2 + 33.8594x_2 - 6.3750, \quad \frac{df}{dx_2} = 8x_2 + 33.8594 = 0 \Rightarrow x_2 = -4.23242.$$

Новая точка: $M^2 = (-5.171875,\ -4.23242)$.

$$f(-5.171875, -1.9375) \approx -140.8817, \quad f(-5.171875, -4.23242) \approx -315.0434 \quad \Rightarrow \text{уменьшение на } 174.1617.$$

Итерация 3. $M^2 = (-5.171875,\ -4.23242)$.

Шаг 1: минимизация по x_1, при $x_2 = -4.23242$.

$$\frac{df}{dx_1} = 4x_1 + 32.1621 = 0 \Rightarrow x_1 \approx -8.04053.$$

Промежуточная точка: $M^{3/2} \approx (-8.04053,\ -4.23242)$.

$$f(-5.171875, -4.23242) \approx -315.0434, \quad f(-8.04053, -4.23242) \approx -354.7012 \quad \Rightarrow \text{уменьшение на } 39.6578.$$

Шаг 2: минимизация по x_2, при $x_1 = -8.04053$.

$$\frac{df}{dx_2} = 8x_2 + 48.2027 = 0 \Rightarrow x_2 \approx -6.02534.$$

Новая точка: $M^3 \approx (-8.04053,\ -6.02534)$.

$$f(-8.04053, -4.23242) \approx -354.7012, \quad f(-8.04053, -6.02534) \approx -390.8752 \quad \Rightarrow \text{уменьшение на } 36.174.$$

Вывод. После трёх итераций метод покоординатного спуска дал:

$$x^* \approx (-8.04053,\ -6.02534), \quad f(x^*) \approx -390.8752.$$

3 Метод градиентного спуска с фиксированным шагом

3.1 Ручные вычисления

Начальная точка: $M^0 = (1, 1)$, шаг $\lambda = 0.1$.

Итерация 1. $x^{(0)} = (1, 1)$.

$$\nabla f(1,1) = \begin{pmatrix} 4 \cdot 1 - 5 \cdot 1 + 11 \\ 8 \cdot 1 - 5 \cdot 1 + 8 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \end{pmatrix}, \quad x^{(1)} = (1,1) - 0.1 \cdot (10, 11) = (0,\ -0.1).$$

$$f(1,1) = 2 + 4 - 5 + 11 + 8 - 3 = 17.0, \quad f(0, -0.1) = 0 + 0.04 + 0 + 0 - 0.8 - 3 = -3.76.$$

Уменьшение: $17.0 - (-3.76) = 20.76$.

Итерация 2. $x^{(1)} = (0,\ -0.1)$.

$$\nabla f(0, -0.1) = \begin{pmatrix} 4 \cdot 0 - 5 \cdot (-0.1) + 11 \\ 8 \cdot (-0.1) - 5 \cdot 0 + 8 \end{pmatrix} = \begin{pmatrix} 11.5 \\ 7.2 \end{pmatrix}, \quad x^{(2)} = (0,\ -0.1) - 0.1 \cdot (11.5, 7.2) = (-1.15,\ -0.82).$$

$$f(0, -0.1) = -3.76, \quad f(-1.15,\ -0.82) \approx -46.428. \quad \text{Уменьшение: } 42.668.$$

Итерация 3. $x^{(2)} \approx (-1.15,\ -0.82)$.

$$\nabla f(-1.15, -0.82) \approx \begin{pmatrix} 4 \cdot (-1.15) - 5 \cdot (-0.82) + 11 \\ 8 \cdot (-0.82) - 5 \cdot (-1.15) + 8 \end{pmatrix} \approx \begin{pmatrix} 10.5 \\ 7.19 \end{pmatrix},$$

$$x^{(3)} \approx (-1.15,\ -0.82) - 0.1 \cdot (10.5, 7.19) \approx (-2.20,\ -1.539).$$

$$f(-1.15,\ -0.82) \approx -46.428, \quad f(-2.20,\ -1.539) \approx -120.969. \quad \text{Уменьшение: } 74.541.$$

$$x^* \approx (-6.37,\ -4.02), \quad f(x^*) \approx -303.627.$$

4 Метод наискорейшего спуска

4.1 Ручные вычисления

Начальная точка: $M^0 = (1, 1)$.

Итерация 1.

$$\nabla f(1,1) = \begin{pmatrix} 4 \cdot 1 - 5 \cdot 1 + 11 \\ 8 \cdot 1 - 5 \cdot 1 + 8 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \end{pmatrix}, \quad d^{(0)} = -\nabla f = (-10,\ -11).$$

Поиск оптимального шага:

$$\varphi(\lambda) = f(1 - 10\lambda,\ 1 - 11\lambda) = 134\lambda^2 - 221\lambda + 17,$$

$$\varphi'(\lambda) = 268\lambda - 221 = 0 \Rightarrow \lambda_0 = \frac{221}{268} \approx 0.825.$$

Новое приближение:

$$x^{(1)} = (1, 1) + 0.825 \cdot (-10,\ -11) \approx (-7.25,\ -8.08).$$

$$f(1, 1) = 17.0, \quad f(-7.25,\ -8.08) \approx -353.38, \quad \text{уменьшение: } 370.38.$$

Итерация 2. $M^1 \approx (-7.25,\ -8.08)$.

$$\nabla f(-7.25, -8.08) \approx (-2.65,\ 3.89), \quad d^{(1)} = (2.65,\ -3.89)$$
$$\varphi(\lambda) = f(-7.25 + 2.65\lambda,\ -8.08 - 3.89\lambda)$$

Подставим в исходную функцию:

$$\begin{aligned} \varphi(\lambda) &= 2(-7.25 + 2.65\lambda)^2 + 4(-8.08 - 3.89\lambda)^2 \\ &\quad - 5(-7.25 + 2.65\lambda)(-8.08 - 3.89\lambda) \\ &\quad + 11(-7.25 + 2.65\lambda) + 8(-8.08 - 3.89\lambda) - 3 \\ &= A\lambda^2 + B\lambda + C \end{aligned}$$

Рассчитанные коэффициенты:

$$A \approx 20.36, \quad B \approx -3.01, \quad C \approx f(-7.25,\ -8.08) \approx -353.38$$

Находим минимум:

$$\lambda_1 = -\frac{B}{2A} = \frac{3.01}{2 \cdot 20.36} \approx 0.074$$

$$x^{(2)} = (-7.25,\ -8.08) + 0.074 \cdot (2.65,\ -3.89) \approx (-7.05,\ -8.37)$$

$$f(x^{(2)}) \approx -355.20, \quad \text{уменьшение на } 1.82$$

Итерация 3. $M^2 \approx (-7.05,\ -8.37)$.

$$\nabla f(-7.05,\ -8.37) \approx (24.65,\ -23.71), \quad d^{(2)} = (-24.65,\ 23.71)$$

$$\varphi(\lambda) = f(-7.05 - 24.65\lambda,\ -8.37 + 23.71\lambda)$$

Подставим в исходную функцию:

$$\begin{aligned}\varphi(\lambda) = {} & 2(-7.05 - 24.65\lambda)^2 + 4(-8.37 + 23.71\lambda)^2 \\ & - 5(-7.05 - 24.65\lambda)(-8.37 + 23.71\lambda) \\ & + 11(-7.05 - 24.65\lambda) + 8(-8.37 + 23.71\lambda) - 3 \\ = {} & A\lambda^2 + B\lambda + C\end{aligned}$$

Рассчитанные коэффициенты:

$$A \approx 1214.13, \quad B \approx -17.49, \quad C \approx f(-7.05,\ -8.37) \approx -355.20$$

Минимум:

$$\lambda_2 = -\frac{B}{2A} = \frac{17.49}{2 \cdot 1214.13} \approx 0.0072$$

$$x^{(3)} = (-7.05,\ -8.37) + 0.0072 \cdot (-24.65,\ 23.71) \approx (-7.23,\ -8.20)$$

$$f(x^{(3)}) \approx -355.202, \quad \text{уменьшение на } 0.002$$

5 Программная реализация на Python

```
    def coordinate_descent(epsilon=0.0001, max_iter=100):
    x1, x2 = 2, -2
    iter_num = 0

    print(f"Iteration {iter_num}: x1 = {x1:.6f}, x2 = {x2:.6f}, f = {f(x1, x2):.6f}")

    while iter_num < max_iter:
        x1_old, x2_old = x1, x2

        x1 = (3 - 0.5 * x2) / 14.0

        x2 = (5 - 0.5 * x1) / 6.0

        iter_num += 1
        diff = abs(x1 - x1_old) + abs(x2 - x2_old)

        print(f"Iteration {iter_num}: x1 = {x1:.6f}, x2 = {x2:.6f}, f = {f(x1, x2):.6f}, diff = {
            diff:.6f}")

        if diff < epsilon:
            break

    return x1, x2, iter_num

def f(x1, x2):
    return 7*x1**2 + 3*x2**2 + 0.5*x1*x2 - 3*x1 - 5*x2 + 2

if __name__ == "__main__":
    final_x1, final_x2, iterations = coordinate_descent()
    print("\nFinal result:")
    print(f"x1 = {final_x1:.6f}")
    print(f"x2 = {final_x2:.6f}")
    print(f"f(x1, x2) = {f(final_x1, final_x2):.6f}")
    print(f"Total iterations: {iterations}")


    import numpy as np

def function_value(x1, x2):
    return 7*x1**2 + 3*x2**2 + 0.5*x1*x2 - 3*x1 - 5*x2 + 2
```

```
def gradient(x1, x2):
    grad_x1 = 14*x1 + 0.5*x2 - 3
    grad_x2 = 6*x2 + 0.5*x1 - 5
    return np.array([grad_x1, grad_x2])

def gradient_descent(eta, epsilon, max_iterations):
    x = np.array([2.0, -2.0])
    print(f" : ({x[0]}, {x[1]})")

    min_eta = 1e-6

    for i in range(max_iterations):
        grad = gradient(x[0], x[1])

        if np.linalg.norm(grad) < epsilon:
            print(f"   {i+1} .")
            break

        current_val = function_value(x[0], x[1])
        new_x = x - eta * grad
        new_val = function_value(new_x[0], new_x[1])

        retry_count = 0
        while new_val > current_val and eta > min_eta:
            eta /= 2
            retry_count += 1
            new_x = x - eta * grad
            new_val = function_value(new_x[0], new_x[1])

        if eta <= min_eta:
            print(f"   (eta={eta}).  .")
            break

        x = new_x
        print(f" {i+1}:  ({x[0]:.6f}, {x[1]:.6f}),  : {new_val:.6f},  : {eta:.6f}")

    else:
        print("   .")

    return x[0], x[1]

#
eta = 0.1
epsilon = 0.0001
max_iterations = 1000

final_x1, final_x2 = gradient_descent(eta, epsilon, max_iterations)
print(f" : ({final_x1:.6f}, {final_x2:.6f})")
print(f"   : {function_value(final_x1, final_x2):.6f}")

import numpy as np

def f(x):
    x1, x2 = x
    return 7*x1**2 + 3*x2**2 + 0.5*x1*x2 - 3*x1 - 5*x2 + 2

def gradient(x):
    x1, x2 = x
    grad_x1 = 14*x1 + 0.5*x2 - 3
    grad_x2 = 6*x2 + 0.5*x1 - 5
    return np.array([grad_x1, grad_x2])

def golden_section_search(f, a, b, tol=1e-5, max_iter=100):
    ratio = (np.sqrt(5) - 1) / 2
    c = b - ratio*(b - a)
```

```
    d = a + ratio*(b - a)

    for _ in range(max_iter):
        if f(c) < f(d):
            b = d
        else:
            a = c

        c = b - ratio*(b - a)
        d = a + ratio*(b - a)

        if abs(b - a) < tol:
            break
    return (a + b) / 2

def steepest_descent(f, grad, x0, epsilon=1e-4, max_iter=100):
    x = np.array(x0, dtype=float)
    history = []

    for k in range(max_iter):
        g = grad(x)
        grad_norm = np.linalg.norm(g)
        history.append((x.copy(), f(x), grad_norm))

        if grad_norm < epsilon:
            break

        S = -g / grad_norm

        def f_alpha(alpha):
            return f(x + alpha * S)

        alpha = golden_section_search(f_alpha, 0, 1)
        x = x + alpha * S

    return x, history

x0 = [2.0, -2.0]
epsilon = 0.0001

solution, history = steepest_descent(f, gradient, x0, epsilon)

print("  :")
print(f" : ({x0[0]}, {x0[1]})")
print("============================================")

for i, (x, f_val, grad_norm) in enumerate(history[:3]):
    print(f" {i+1}:")
    print(f": ({x[0]:.4f}, {x[1]:.4f})")
    print(f" : {f_val:.4f}")
    print(f" : {grad_norm:.4f}")
    print("--------------------------------------------")

final_x, final_f, final_grad_norm = history[-1]
print(f"\n : ({final_x[0]:.6f}, {final_x[1]:.6f})")
print(f" : {final_f:.6f}")
print(f" : {final_grad_norm:.6f}")
if final_grad_norm < epsilon:
    print("  !")
else:
    print("   .")
```