Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №5

«Метод наискорейшего спуска поиска минимума функции многих переменных»

по курсу «Численные методы»

Студент: Шемякин В.А.

Группа: ИУ9-62Б

Преподаватель: Домрачева А.Б.

Москва 2025

1 Цель

Целью данной работы является изучение метода наискорейшего спуска поиска минимума функции двух переменных и сравнение с результатом, найденным аналитически.

2 Постановка задачи

Дано: функция многих переменных

$$f(x) = (x_1, x_2, ..., x_n)\text{и точка } X^0$$

Задание:

1. Найти минимум функции двух переменных с точностью 0.001, начиная итерации с точки X^0
2. Найти минимум аналитичности
3. Сравнить полученные результаты

Индивидуальный вариант(№ 5): $f(x) = (x_1 - 1)^2 + (x_2 - 2)^4 + e^{\frac{1}{x_1^2 + x_2^2}}, X^0 = (1, 2)$

3 Основные теоретические сведения

Метод наискорейшего спуска является итерационным. Пусть для заданной функции $f(x_1, x_2, \ldots, x_n)$ на k-том шаге имеется приближение к минимуму

$$\mathbf{X}^k = \big(x_1^k, x_2^k, \ldots, x_n^k\big).$$

Рассмотрим функцию одной переменной

$$\varphi_k(t) = f\big(\mathbf{X}^k - t \operatorname{grad} f(\mathbf{X}^k)\big),$$

где

$$\operatorname{grad} f(\mathbf{X}^k) = \left(\frac{\partial f}{\partial x_1}(\mathbf{X}^k), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{X}^k)\right)$$

— градиент функции f в точке $\mathbf{X}^k$.

Функция $\varphi_k(t)$ представляет собой ограничение f на прямую градиентного спуска, проходящую через точку $\mathbf{X}^k$.

Следующее приближение к точке минимума принимаем в виде

$$\boxed{\mathbf{X}^{k+1} = \mathbf{X}^k - t^* \operatorname{grad} f(\mathbf{X}^k)},$$

где t^* — минимум функции $\varphi_k(t)$.

Процесс поиска минимума продолжается, пока

$$\left\|\operatorname{grad} f(\mathbf{X}^k)\right\| = \max_{1\le i\le n}\left|\tfrac{\partial f}{\partial x_i}(\mathbf{X}^k)\right| < \varepsilon$$

не станет меньше заданной погрешности ε.

Двумерный случай

При $n = 2$ формула итерации принимает вид

$$(x_{k+1}, y_{k+1}) = \left(x_k - t^* \tfrac{\partial f}{\partial x},\ y_k - t^* \tfrac{\partial f}{\partial y}\right),$$

где

$$t^* = -\frac{\varphi_k'(0)}{\varphi_k''(0)}, \qquad \varphi_k'(0) = -\left(\left(\tfrac{\partial f}{\partial x}\right)^2 + \left(\tfrac{\partial f}{\partial y}\right)^2\right),$$

$$\varphi_k''(0) = \tfrac{\partial^2 f}{\partial x^2}\left(\tfrac{\partial f}{\partial x}\right)^2 + 2\tfrac{\partial^2 f}{\partial x\,\partial y}\tfrac{\partial f}{\partial x}\tfrac{\partial f}{\partial y} + \tfrac{\partial^2 f}{\partial y^2}\left(\tfrac{\partial f}{\partial y}\right)^2,$$

а все производные берутся в точке (x_k, y_k).

4 Реализация

Listing 1: main.py

```
import math

eps = 1e-3
max_iter = 10000

def f(x, y):
    return (x - 1) ** 2 + (y - 2) ** 4 + math.exp(1 / (x * x + y * y))
```

```
def f_x(x, y):
    r2 = x * x + y * y
    return 2 * (x - 1) - (2 * x / r2 ** 2) * math.exp(1 / r2)

def f_y(x, y):
    r2 = x * x + y * y
    return 4 * (y - 2) ** 3 - (2 * y / r2 ** 2) * math.exp(1 / r2)

def f_x_x(x, y):
    r2 = x * x + y * y
    exp1 = math.exp(1 / r2)
    q = 2 * x / r2 ** 2
    dq_dx = 2 / r2 ** 2 - 8 * x * x / r2 ** 3
    return 2 - exp1 * (dq_dx - q * q)

def f_y_y(x, y):
    r2 = x * x + y * y
    exp1 = math.exp(1 / r2)
    s = 2 * y / r2 ** 2
    ds_dy = 2 / r2 ** 2 - 8 * y * y / r2 ** 3
    return 12 * (y - 2) ** 2 - exp1 * (ds_dy - s * s)

def f_x_y(x, y):
    r2 = x * x + y * y
    exp1 = math.exp(1 / r2)
    return exp1 * (8 * x * y / r2 ** 3 + 4 * x * y / r2 ** 4)

def analytical_min():
    return 1.029180, 2.319519
def solve(xk, yk):
    k = 0

    while max(abs(f_x(xk, yk)), abs(f_y(xk, yk))) >= eps and k < max_iter:
        gx, gy = f_x(xk, yk), f_y(xk, yk)
        phi1 = - (gx * gx + gy * gy)
        gHg  = (f_x_x(xk, yk) * gx * gx + 2 * f_x_y(xk, yk) * gx * gy + f_y_y(xk, yk) * gy * gy)
        t_star = - phi1 / gHg
        xk -= t_star * gx
        yk -= t_star * gy
        k += 1
    return k, xk, yk



def main():
    xk, yk = 1.0, 2.0
```

```
    it, xk, yk = solve(xk, yk)

    print(f'Количество итераций: {it}')
    print(f"Метод: ({xk:.6f}, {yk:.6f})")
    print(f'Аналитически: {analytical_min()}')
    print(f'Разность: ({abs(xk - analytical_min()[0]):.6e}, {abs(yk - analytical_min()[1]):.6e})')

if __name__ == "__main__":
    main()
```

5 Результаты

Количество итераций: 4

Метод: (1.028994, 2.319510)

Аналитически: (1.02918, 2.319519)

Разность: (1.862377e-04, 9.155482e-06)

6 Вывод

В ходе выполнения лабораторной работы был реализован метод наискорейшего спуска. Полученные численные результаты мало отличаются от результатов, полученных аналитически, что подтверждает эффективность метода.