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

по курсу «Численные методы»

Студент группы ИУ9-62Б Нащёкин Н.Д.

Преподаватель: Домрачева А.Б.

Москва, 2025

1 Задача

1. Найти минимум функции двух переменных с точностью $\varepsilon = 0.001$, начиная итерации из точки X^0.

2. Найти минимум аналитически.

3. Сравнить полученные результаты.

2 Основная теория

Метод наискорейшего спуска

Метод наискорейшего спуска – итерационный метод, позволяющий найти точку минимума заданной функции. Пусть для функции $f(x_1, ..., x_n)$ на k-ом шаге имеем приближение к минимуму $X^k \approx (x_1^k, ..., x_n^k)$. Рассмотрим функцию $\varphi_k(t) = f(x_1^k - t\frac{\partial f}{\partial x_1}(X^k), ..., x_n^k - t\frac{\partial f}{\partial x_n}(X^k)) = f(X^k - t * grad(f(X^k)))$.

Функция $\varphi_k(t)$ – это ограничение функции $f(X)$ на прямую градиентного (наискорейшего) спуска, проходящую через точку k-го приближения X^k. Пусть t^* – точка минимума этой функции. Тогда следующее приближение к точке минимума: $X^{k+1} = X^k - t^* * grad(f(X^k)) = x_1^k - t^*\frac{\partial f}{\partial x_1}(X^k), ..., x_n^k - t^*\frac{\partial f}{\partial x_n}(X^k)$.

Итерации продолжаются, пока не будет выполнено условие завершения счёта:

$$||grad(f(X^k))|| = \max_{1 \le i \le n} |\frac{\partial f}{\partial x_i}(X^k)| < \varepsilon$$

Чаще всего точно искать минимум функции $\varphi(t)$ не нужно и достаточно ограничиться лишь одним приближением и поиском t^k, например, по методу парабол. Тогда в двумерном случае приближения будут особенно простыми:

$$(x_{k+1}, y_{k+1}) = (x_k - t^k\frac{\partial f}{\partial x}, y_k - t^k\frac{\partial f}{\partial y}).$$

Здесь $t^k = -\frac{\varphi_k'(0)}{\varphi_k''(0)}$, где $\varphi_k'(0) = -(\frac{\partial f}{\partial x})^2 - (\frac{\partial f}{\partial y})^2$, $\varphi_k''(0) = \frac{\partial^2 f}{\partial x^2}(\frac{\partial f}{\partial x})^2 + 2\frac{\partial^2 f}{\partial x \partial y}\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2}(\frac{\partial f}{\partial y})^2$, все частные производные здесь берутся в точке (x_k, y_k).

3 Практическая реализация

Листинг 1 — реализация программы

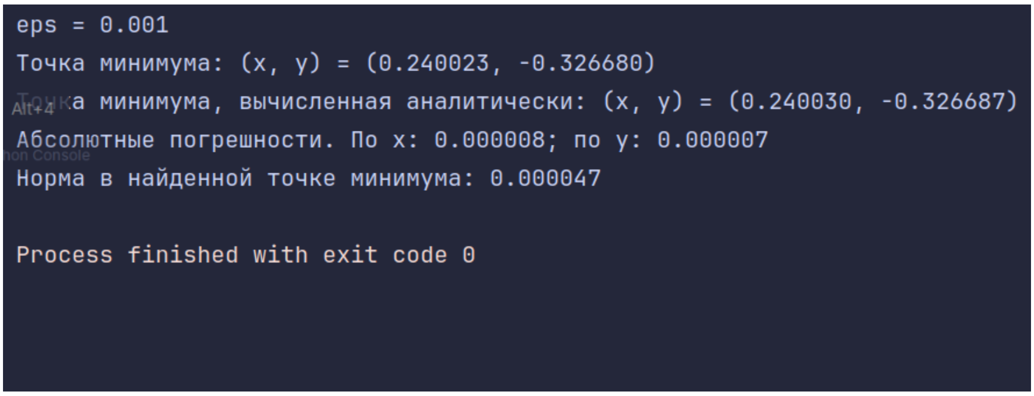
```python
from sympy import *


# f(x) = 2 * x_1^2 + 3 * x_2^2 - 2 * sin((x_1 - x_2) / 2) + x_2
# x0 = (0, 0)


eps = 0.001


x_1 = Symbol("x_1")
x_2 = Symbol("x_2")
f = 2 * x_1 ** 2 + 3 * x_2 ** 2 - 2 * sin((x_1 - x_2) / 2) + x_2


def norm(point: dict):
    df_x1 = diff(f, x_1).subs(point).evalf()
    df_x2 = diff(f, x_2).subs(point).evalf()
    return max(abs(df_x1), abs(df_x2))


def phi_k_first(x_k):
    return -(diff(f, x_1).subs(x_k) ** 2).evalf() - (diff(f, x_2).subs(x_k) ** 2).evalf()


def phi_k_second(x_k):
    return (
        diff(f, x_1, 2).subs(x_k).evalf() * diff(f, x_1).subs(x_k).evalf() ** 2
        + 2
        * diff(f, x_1, x_2).subs(x_k).evalf()
        * diff(f, x_1).subs(x_k).evalf()
        * diff(f, x_2).subs(x_k).evalf()
        + diff(f, x_2, 2).subs(x_k).evalf() * diff(f, x_2).subs(x_k).evalf() ** 2
    )


def t_k(x_k):
    return - phi_k_first(x_k) / phi_k_second(x_k)


def grad_down():
    x_k = {x_1: 0, x_2: 0}
    x_k1 = {x_1: 0, x_2: 0}
    while norm(x_k) >= eps:
        x_k = x_k1
        x_k1 = {
```

```
            x_1: x_k[x_1] - t_k(x_k) * diff(f, x_1).subs(x_k).evalf(),
            x_2: x_k[x_2] - t_k(x_k) * diff(f, x_2).subs(x_k).evalf()
        }
    return x_k1


def find_min():
    df_dx1 = diff(f, x_1)
    df_dx2 = diff(f, x_2)
    critical_point = nsolve([df_dx1, df_dx2], (x_1, x_2), [0, 0])
    return critical_point


def main():
    res_numeric = grad_down()
    res_analytic = find_min()

    print(f'eps = {eps}')
    print(f'Точка минимума: (x, y) = ({res_numeric[x_1]:.6f}, {res_numeric[x_2]:.6f})')
    print(f'Точка минимума, вычисленная аналитически: (x, y) = ({res_analytic[0]:.6f}, {res_analytic[1]:.6f})')
    print(f'Абсолютные погрешности. По x: {abs(res_numeric[x_1] - res_analytic[0]):.6f}; '
          f'по y: {abs(res_numeric[x_2] - res_analytic[1]):.6f}')
    print(f"Норма в найденной точке минимума: {norm(res_numeric):.6f}")


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

Мой вариант – 22. По условию требуется найти минимум функции $f = 2x_1^2 + 3x_2^2 - 2sin(\frac{x_1 - x_2}{2}) + x_2$, $X^0 = (0, 0)$. Метод наискорейшего спуска реализован в функции $grad\_down()$. Также по условию необходимо сравнить полученный результат с аналитическим решением. Для этого, например с использованием библиотеки символьных вычислений $SymPy$, нужно найти решение системы

$$\begin{cases} \frac{\partial f}{\partial x} = 0 \\ \frac{\partial f}{\partial y} = 0. \end{cases}$$

Решения этой системы – критические точки, которые нужно проверить на наличие экстремума с помощью достаточного условия его существования: $\frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y} > 0$, причём если $\frac{\partial^2 f}{\partial x^2} > 0$, то это точка минимума (нас интересует этот случай). Но в случае с заданной функцией критическая точка только одна, она и является точкой минимума, поэтому дополнительную проверку достаточного условия можно не проводить. Результат работы программы приведён на рисунке 1.



```
eps = 0.001
Точка минимума: (x, y) = (0.240023, -0.326680)
Точка минимума, вычисленная аналитически: (x, y) = (0.240030, -0.326687)
Абсолютные погрешности. По x: 0.000008; по y: 0.000007
Норма в найденной точке минимума: 0.000047

Process finished with exit code 0
```

Рис. 1 — Результат выполнения программы

4 Вывод

В данной лабораторной работе был реализован метод наискорейшего спуска для поиска точки минимума функции двух переменных. Кроме того, требуемая точка также была найдена аналитически. Результаты численного и аналитического поиска, а также норма градиента заданной функции в численно найденной точке выводятся на экран.