Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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»

Отчет по лабораторной работе № 5 «Метод наискорейшего спуска поиска минимума функции многих переменных»

по курсу
«Численные методы»

Выполнил:

Студент группы ИУ9-62Б

Караник А.А.

Проверила:

Домрачева А.Б.

2024 г.

Цель работы

Целью данной работы является реализация программы, осуществляющая поиск минимума функции многих переменных методом наискорейшего спуска.

Постановка задачи

1) Найти минимум функции $f(x,y) = x^4 + y^2 + \ln(1 + 0.1x^2y^2) + 0.15x$ двух переменных с точностью $\varepsilon = 0.001$, начиная итерации из точки $X^0 = (0,1)$.
2) Найти минимум аналитичности.
3) Сравнить полученные результаты.

Теоретические сведения

Метод наискорейшего спуска является итерационным. Пусть для функции $f(x_1, x_2, \dots, x_n)$ (или в векторной форме $f(x)$) на k-м шаге имеем некоторое приближение к минимуму $x^k = \left(x_1^k, \dots, x_n^k\right)$. Рассмотрим функции одной переменной $\varphi_k(t) = f\left(x^k - t grad f(x^k)\right)$.

Минимум функции $\varphi_k(t)$ можно найти любым методом одномерной оптимизации. Обозначим эту точку минимума через t^*. Теперь для следующего приближения к точке экстремума полагаем

$$x^{k+1} = x^k - t^* grad f(x^k)$$

Процесс поиска минимума продолжается до тех пор, пока $\left|\left|grad f(x^k)\right|\right| = \max\limits_{i \le i \le n} \left|\frac{\partial f}{\partial x_i}(x^k)\right|$ не станет меньше допустимой погрешности ε.

В большинстве случаев точно искать минимум функции $\varphi_k(t)$ не нужно и достаточно ограничиться лишь одним приближением к t построенным, например, по методу парабол. Тогда особенно простым вид итерации будет в двумерном случае:

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

где $t^* = -\frac{\varphi_k'(0)}{\varphi_k''(0)}$; $\varphi_k'(0) = -\left(\frac{\partial f}{\partial x}\right)^2 - \left(\frac{\partial f}{\partial y}\right)^2$; $\varphi_k''(0) = \frac{\partial f^2}{\partial x^2}\left(\frac{\partial f}{\partial x}\right)^2 + 2\frac{\partial f^2}{\partial x \partial y}\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial f^2}{\partial y^2}\left(\frac{\partial f}{\partial y}\right)^2$; все производные берутся в точке (x_k, y_k)

Реализация

Исходный код программы:

```
/**
 * Laboratory work: 5
 * Discipline: Numerical methods
 * Copyright (c) 2024, Andrey Karanik
*/

#include <iostream>
#define _USE_MATH_DEFINES
#include <cmath>
#include <iomanip>
#include <vector>

double F(double x1, double x2) {
    return x1 * x1 * x1 * x1 + x2 * x2 + log(1 + 0.1 * x1 * x1 * x2 * x2) + 0.15 * x1;
}

double derivativeX1(double x1, double x2) {
    return 4 * x1 * x1 * x1 + (2 * x2 * x2 * x1) / (10 + x2 * x2 * x1 * x1) + 0.15;
}

double derivativeX2(double x1, double x2) {
    return 2 * x2 + (2 * x1 * x1 * x2) / (10 + x1 * x1 * x2 * x2);
}

double derivativeX1X1(double x1, double x2) {
   return (0.2 * x2 * x2) / (1 + 0.1 * x2 * x2 * x1 * x1) + x1 * x1 * (12 - (0.04 * x2 * x2 * x2 * x2) / ((1 + 0.1 * x1 * x1
* x2 * x2) * (1 + 0.1 * x1 * x1 * x2 * x2)));
}

double derivativeX1X2(double x1, double x2) {
   return (40 * x1 * x2) / ((x1 * x1 * x2 * x2 + 10) * (x1 * x1 * x2 * x2 + 10));
}

double derivativeX2X2(double x1, double x2) {
   return (0.2 * x1 * x1) / (1 + 0.1 * x1 * x1 * x2 * x2) - (0.04 * x1 * x1 * x1 * x1 * x2 * x2) / ((1 + 0.1 * x1 * x1 * x2 *
x2) * (1 + 0.1 * x1 * x1 * x2 * x2)) + 2;
}

int main(int argc, char *argv[]) {
   double eps = 0.001;

   int k = 0;
   double x1k = 0;
   double x2k = 1;

   double max;

   do {
      double phi1 = -(derivativeX1(x1k, x2k) * derivativeX1(x1k, x2k)) - (derivativeX2(x1k, x2k) * derivativeX2(x1k, x2k));
      double phi2 = derivativeX1X1(x1k, x2k) * derivativeX1(x1k, x2k) * derivativeX1(x1k, x2k) +
                    2 * derivativeX1X2(x1k, x2k) * derivativeX1(x1k, x2k) * derivativeX2(x1k, x2k) +
                    derivativeX2X2(x1k, x2k) * derivativeX2(x1k, x2k) * derivativeX2(x1k, x2k);
      double t = -(phi1 / phi2);
      double temp1 = x1k;
      double temp2 = x2k;
      x1k = temp1 - t * derivativeX1(temp1, temp2);
      x2k = temp2 - t * derivativeX2(temp1, temp2);
      k++;
```

```
        if (fabs(derivativeX1(x1k, x2k)) > fabs(derivativeX2(x1k, x2k))) {
            max = fabs(derivativeX1(x1k, x2k));
        } else {
            max = fabs(derivativeX2(x1k, x2k));
        }

    } while (max >= eps);

    std::cout << "minimum:" << std::endl;

    std::cout << x1k << " " << x2k << std::endl;

    std::cout << "analytical minimum:" << std::endl;

    std::cout << -0.334716 << " " << 0 << std::endl;

    std::cout << "step: " << std::endl;

    std::cout << k;

    return 0;
};
```

Тестирование

Вывод программы:

minimum:

-0.335077 0.000207822

analytical minimum:

-0.334716 0

step:

9

Вывод

В процессе выполнения лабораторной работы была разработана программа, осуществляющая поиск минимума функции многих переменных методом наискорейшего спуска. Также был найден минимум аналитичности, и было воспроизведено сравнение полученных результатов.