Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности

электронно-вычислительных систем (КИБЭВС)

ЧИСЛЕННАЯ ОПТИМИЗАЦИЯ.

Отчет по лабораторной работе №5

по дисциплине «Численные методы»

Выполнил

Студент гр. 728-2

\_\_\_\_\_\_\_ Д.Р. Геворгян

\_\_.12.2019

Принял

Доцент кафедры КИБЭВС

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ С. С. Катаева

оценка подпись

\_\_.12.2019

1 Введение

Цель работы: Необходимо найти минимум индивидуальной функции  следующими методами:

1. градиентным спуском;
2. наискорейшим спуском.

2 Ход работы

Необходимо найти методом градиентного и наискорейшего спуска минимум индивидуальной функции:

*.*

2.1 Метод градиентного спуска

Градиентный спуск – метод нахождения локального экстремума (минимума или максимума) функции с помощью движения вдоль градиента. Для минимизации функции в направлении градиента используются методы одномерной оптимизации, например, метод золотого сечения. Также можно искать не наилучшую точку в направлении градиента, а какую-либо лучше текущей.

,

где  ( – максимальный коэффициент при переменной). В данной работе *t* = 0.012.

Выберем начальные точки: x1 = 8; x2 = 8.

Выведем на экран результаты первых пяти итераций.

Результаты работы программы представлены на рисунке 2.1.

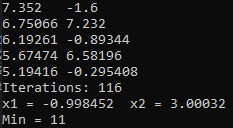


Рисунок 2.1 – Результаты работы программы

График метода градиентного спуска представлен на рисунке 2.2.



Рисунок 2.2 – График метода градиентного спуска

2.2 Метод наискорейшего спуска

Формула итерационного процесса здесь почти не отличается от формулы для метода градиентного спуска, за исключением того, что шаг необходимо рассчитывать на каждой итерации отдельно. Формула имеет вид:

.

При применении наискорейшего спуска для расчета шага t на каждом шаге использовать метод золотого сечения (без использования вычисления производной): для этого определить отрезок, на котором будет лежать оптимальное значение длины шага, и на выбранном отрезке найти минимум функции  с помощью метода золотого сечения. Вычисление оптимальной длины шага происходит на каждой итерации.

На i-й итерации поиска шага вычисляем значения  по формулам:



,

где .

Также рассчитываем значения . Если , то , иначе . Итерационный процесс завершается, когда , где  = 0.01. В таком случае за вычисленное значение шага принимается значение .

Выберем начальные точки: = 2; = 2.

Результаты работы программы представлены на рисунке 2.3.

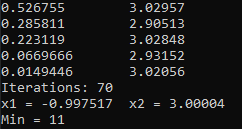


Рисунок 2.3 – Результаты работы программы

График метода наискорейшего спуска представлен на рисунке 2.4.



Рисунок 2.4 – График метода наискорейшего спуска

3 Заключение

В ходе лабораторной работы был найден минимум индивидуальной функции методами градиентного и наискорейшего спуска. При применении наискорейшего спуска для расчета шага на каждом шаге был использован метод «золотого сечения».

Код программ представлен в приложениях А, Б. Язык программирования C++.

Отчёт был написан согласно ОС ТУСУР 2013.

Таблица 3.1 - Результаты вычислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Начальная точка | Количество итераций | t | Точка минимума | Минимум функции |
| Градиентного спуска | (8;8) | 116 | 0.012 | (-0.0998452;3.00032) | 11 |
| Метод наискорейшего спуска | (2;2) | 70 | - | (-0.997517;3.00004) | 11 |

Приложение А

(обязательное)

Код программы:

# include <iostream>

# include <math.h>

using namespace std;

double f(double x1, double x2) {

return 3 \* pow(x1 + 1, 2) + 80 \* pow(x2 - 3, 2) + 11;

}

double gr1(double x) {

return 6 \* x + 6;

}

double gr2(double x) {

return 160 \* x - 480;

}

int main() {

double t = 0.012;

double x1 = 8;

double x2 = 8;

int c = 0;

while (abs(gr1(x1)) > 0.01 && abs(gr2(x2)) > 0.01) {

x1 = x1 - t \* gr1(x1);

x2 = x2 - t \* gr2(x2);

if (c < 5)

cout << x1 << "\t" << x2 << endl;

++c;

}

cout << "Iterations: " << c << endl;

cout << "x1 = " << x1 << "\t" << "x2 = " << x2 << endl;

cout << "Min = " << f(x1, x2) << endl;

system("pause");

return 0;

}

Приложение Б

(обязательное)

Код программы:

#include <iostream>

#include <math.h>

using namespace std;

double f(double x1, double x2) {

return 3 \* pow(x1 + 1, 2) + 80 \* pow(x2 - 3, 2) + 11;

}

double gr1(double x) {

return 6 \* x + 6;

}

double gr2(double x) {

return 160 \* x - 480;

}

double golden\_ratio(double a, double b, double x1, double x2) {

double al = 0.618;

double be = 0.382;

double xb, xa;

while (fabs(a - b) > 0.01) {

xa = b - be \* fabs(b - a);

xb = b - al \* fabs(b - a);

if (f(x1 - xb \* gr1(x1), x2 - xb \* gr2(x2)) >= f(x1 - xa \* gr1(x1), x2 - xa \*

gr2(x2)))

a = xb;

else

b = xa;

}

if (f(x1 - xb \* gr1(x1), x2 - xb \* gr2(x2)) < f(x1 - xa \* gr1(x1), x2 - xa \* gr2(x2)))

return xb;

else

return xa;

}

int main() {

double t = 0;

double x1 = 2;

double x2 = 2;

double a = 0;

double b = 1;

int c = 0;

while (fabs(gr1(x1)) > 0.01 && fabs(gr2(x2)) > 0.01) {

t = golden\_ratio(a, b, x1, x2);

x1 = x1 - t \* gr1(x1);

x2 = x2 - t \* gr2(x2);

if (c > 6 && c < 12)

cout << x1 << "\t" << x2 << endl;

++c;

}

cout << "Iterations: " << c << endl;

cout << "x1 = " << x1 << "\t" << "x2 = " << x2 << endl;

cout << "Min = " << f(x1, x2) << endl;

system("pause");

return 0;

}