МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики Кафедра прикладной математики**

**ОТЧЕТ**

**по дисциплине**

**«Методы оптимизации»**

Работу выполнил М. А. Пузырёв

Работу принял преподаватель Е. С. Троценко

Краснодар

2024

1. **Постановка задачи**

Пусть дана функция , ограниченная снизу на множестве *R*n и имеющая непрерывные частные производные во всех его точках.

Требуется найти локальный минимум функции двух переменных  методом Ньютона при начальных значениях , т.е. такие точки , что .

1. **Стратегия поиска**

Стратегия решения задачи состоит в построении последовательности точек {, k = , таких, что . Точки последовательности {} вычисляются по правилу

(1)

где задается пользователем, а направление спуска определяется для каждого значения k по формуле

(2)

Выбор по формуле (2) гарантирует выполнение требования при условии, что . Формула (2) получена из следующих соображений:

1. Функция аппроксимируется в каждой точке последовательности {, квадратичной функцией

2. Направление определяется из необходимого условия экстремума первого порядка: . Таким образом, при выполнении требования последовательность является последовательностью точек минимумов квадратичных функций . Чтобы обеспечить выполнение требования , даже в тех случаях, когда для каких-либо значений матрица Гессе не окажется положительно определенной, рекомендуется для соответствующих значений вычислить точку по методу градиентного спуска c выбором величины шага из условия

Построение последовательности {}, заканчивается в точке , для которой , где – заданное малое положительное число, или при ( - предельное число итераций), или при двукратном одновременном выполнении неравенств , где - малое положительное число. Вопрос о том, может ли точка рассматриваться как найденное приближение искомой точки локального минимума , решается путем дополнительного исследования.

1. **Алгоритм**

*Шаг l*. Задать предельное число итераций . Найти градиент и матрицу Гессе .

*Шаг 2*. Положить

*Шаг 3*. Вычислить.

*Шаг 4*. Проверить выполнение критерия окончания :

а) если критерий выполнен, то ;

б) если критерий не выполнен, то перейти к шагу 5.

*Шаг 5*. Проверить выполнение неравенства :

а) если неравенство выполнено, то ;

б) если нет, то перейти к шагу 6.

*Шаг 6*. Вычислить матрицу .

*Шаг 7*. Вычислить матрицу .

*Шаг 8*. Проверить выполнение условия :

а) если , то перейти к шагу 9;

б) если нет, то перейти к шагу 10, положив

*Шаг 9.* Определить

*Шаг 10.* Найти точку ,

положив если

или выбрав из условия если

*Шаг 11.* Проверить выполнение условий

а) если оба условия выполнены при текущем значении и , то расчет окончен, ;

б) в противном случае положить и перейти к шагу 3.

**4 Код программы**

Реализация кода представлена на языке C++:

#include <iostream>

#include <cmath>

#include <math.h>

using namespace std;

double F(double x, double y) {

return pow(x, 2) + 4 \* pow(y, 2) + x \* y + x;}

double Fx(double x, double y) {

return 2 \* x + y + 1;}

double Fy(double x, double y) {

return x + 8 \* y;}

double L(double x, double y) {

double l = pow(x, 2) + pow(y, 2);

return sqrt(l);}

void M(double \*\*A, double\*\* B, int n) {

for (int k = 0; k < n; ++k){

double div = A[k][k];

for (int j = 0; j < n; ++j){

A[k][j] /= div;

B[k][j] /= div;}

for (int i = k + 1; i < n; ++i){

double multi = A[i][k];

for (int j = 0; j < n; ++j){

A[i][j] -= multi \* A[k][j];

B[i][j] -= multi \* B[k][j];}}}

for (int k = n - 1; k > 0; --k){

for (int i = k - 1; i >= 0; --i){

double multi = A[i][k];

for (int j = 0; j < n; ++j){

A[i][j] -= multi \* A[k][j];

B[i][j] -= multi \* B[k][j];}}}}

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

int m = 50, it = 0;

double eps1 = 0.005, eps2 = 0.01;

double x, y, t, x01, y01, x00, y00, d1, d2;

double\*\* H = new double\* [2];

for (int i = 0; i < 2; i++) {

H[i] = new double[2];}

double\*\* H1 = new double\* [2];

for (int j = 0; j < 2; j++) {

H1[j] = new double[2];}

H[0][0] = 2.0, H[0][1] = 1.0, H[1][0] = 1.0, H[1][1] = 8.0;

H1[0][0] = 1.0, H1[0][1] = 0.0, H1[1][0] = 0.0, H1[1][1] = 1.0;

M(H, H1, 2); // получение матрицы H-1

x00 = x01 = 3, y00 = y01 = 1;

if (H1[0][0] > 0 && (H1[0][1] < 0)) {

for (int i = 0; i < m; i++) {

if (L(Fx(x01, y01), Fy(x01, y01)) < eps1)break;

d1 = -H1[0][0] \* Fx(x01, y01) - H1[0][1] \* Fy(x01, y01);

d2 = -H1[1][0] \* Fx(x01, y01) - H1[1][1] \* Fy(x01, y01);

x = x01 + d1;

y = y01 + d2;

if (L(x - x01, y - y01) < eps2 && fabs(F(x, y) - F(x01, y01)) < eps2 && L(x01 - x00, y01 - y00) < eps2 && fabs(F(x01, y01) - F(x00, y00)) < eps2) {

x01 = x;

y01 = y;

it++;

break;}

x00 = x01, y00 = y01;

x01 = x, y01 = y;

it++;}}

cout << "Алгоритм завершился за " << it << " итераций.\n";

cout << "Точка минимума X = (" << x01 << "; " << y01 << ")\n";

cout << "Значение функции в минимуме: F = " << F(x01, y01) << endl;

return 0;}

**5 Сходимость**

Пусть дважды непрерывно дифференцируема сильно выпуклая функция с константой на и удовлетворяет условию

где , а начальная точка такова, что , т.е.

где Тогда последовательность {} сходится к точке минимума с квадратичной скоростью .

**6 Вывод**

Для решения задачи по поиску безусловного минимума функции методом Ньютона была написана программа на языке C++. Результатом работы программы является вычисленное минимальное значение функции, равное -0,266667 в точке минимума X = (-0,53333; 0,066667) (рис.1). При изменении параметра и , точка минимума, значение функции в ней и сходимость не меняются. При уменьшении и число итераций, необходимых для нахождения безусловного минимума остаётся неизменным, что иллюстрирует тот факт, что метод Ньютона для квадратичных, сильно выпуклых функций сходится за 1 итерацию (рис.2).

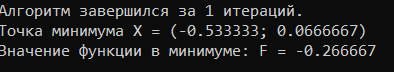


Рисунок 1 – Результат работы программы

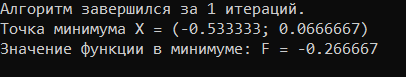


Рисунок 2 – Результат работы программы