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**

Даны дважды непрерывно дифференцируемые целевая функция

и функции ограничений определяющие множество допустимых решений Х.

Требуется найти локальный минимум целевой функции на множестве Х, т.е. такую точку что

,

где

1. **Стратегия поиска**

Идея метода заключается в сведении задачи на условный минимум к решению последовательности задач поиска безусловного минимума вспомогательной функции:

где – штрафная функция, – параметр штрафа, задаваемый на каждой

k-й итерации. Это связано с возможностью применения эффективных и надежных методов поиска безусловного экстремума, изложенных в гл. 2.

Штрафные функции конструируются, исходя из условий:

причем, при невыполнении ограничений и , справедливо . Чем больше , тем больше штраф за невыполнение ограничений. Как правило, для ограничений типа равенств используется квадратичный штраф, а для ограничений типа неравенств – квадрат срезки:

где – срезка функции:

Начальная точка поиска задается обычно вне множества допустимых решений . На каждой итерации ищется точка минимума вспомогательной функции при заданном параметре с помощью одного из методов безусловной минимизации. Полученная точка используется в качестве начальной на следующей итерации, выполняемой при возрастающем значении параметра штрафа. При неограниченном возрастании последовательность точек стремится к точке условного минимума .

1. **Алгоритм**

*Шаг 1.* Задать начальную точку ; начальное значение параметра штрафа ; число для увеличения параметра; малое число для остановки алгоритма. Положить .

*Шаг 2.* Составить вспомогательную функцию

*Шаг 3.* Найти точку безусловного минимума функции по с помощью какого-либо метода (нулевого, первого или второго порядка):

.

При этом задать все требуемые выбранным методом параметры. В качестве начальной точки взять Вычислить .

*Шаг 4.* Проверить условие окончания:

а) если процесс поиска закончить:

б) если , положить и перейти к шагу 2.

**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, double r) {

return pow(x, 2) + 4 \* pow(y, 2) + x \* y + x + (r/2) \* pow((x + y - 2), 2);

}

double P(double x, double y, double r) {

return (2 / r) \* pow((x + y - 2), 2);

}

double Fx(double x, double y, double r) {

return 2 \* x + y + 1 + r \* (x + y - 2);

}

double Fy(double x, double y, double r) {

return x + 8 \* y + r \* (x + y - 2);

}

double L(double x, double y) {

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

return sqrt(l);

}

double G(double x, double y, double r, double t) { // функция f(t)

return F(x - t \* Fx(x, y, r), y - t \* Fy(x, y, r), r);

}

double Zol(double x, double y, double r) { // метод нахождения минимума функции f(t)

double a = -1;

double b = 1;

double k = 0.38196;

double l = 0.01;

double t1 = a + k \* (b - a);

double t2 = a + b - t1;

do {

if (G(x, y, r, t1) <= G(x, y, r, t2)) {

b = t2;

t2 = t1;

t1 = a + b - t2;

}

else {

a = t1;

t1 = t2;

t2 = a + b - t1;

}

} while (fabs(b - a) > l);

return (b + a) / 2;

}

int main() {

setlocale(LC\_ALL, "rus");

int it = 0, m = 20;

double eps1 = 0.01, eps2 = 0.015;

double r = 0.1; double C = 5;

double x, y, t, x01, y01, x00, y00, d1, d2, x0, y0, b, d10, d20;

x00 = x01 = 3.0, y00 = y01 = 1.0;

do {

r = r \* C;

t = Zol(x00, y00, r);

x0 = x00 - t \* Fx(x00, y00, r);

y0 = y00 - t \* Fy(x00, y00, r);

d10 = -Fx(x0, y0, r);

d20 = -Fy(x0, y0, r);

for (int i = 1; i < m; i++) {

if (L(Fx(x0, y0, r), Fy(x0, y0, r)) < eps1)

break;

b = pow(L(Fx(x0, y0, r), Fy(x0, y0, r)), 2) / pow(L(Fx(x01, y01, r), Fy(x01, y01, r)), 2);

d1 = -Fx(x0, y0, r) + b \* d10;

d2 = -Fy(x0, y0, r) + b \* d20;

t = Zol(x0, y0, r);

x = x0 + t \* d1;

y = y0 + t \* d2;

if (L(x - x0, y - y0) < eps2 && fabs(F(x, y, r) - F(x0, y0, r)) < eps2 && L(x0 - x01, y0 - y01) < eps2 && fabs(F(x0, y0, r) - F(x01, y01, r)) < eps2) {

x0 = x;

y0 = y;

break;

}

d10 = d1;

d20 = d2;

x01 = x0, y01 = y0;

x0 = x, y0 = y;

}

x00 = x0, y00 = y0;

cout << x00 << "\t" << y00 << endl;

it++;

} while (P(x0, y0, r) > eps1);

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

cout << "r = " << r << endl;

cout << "Точка минимума X = (" << x0 << "; " << y0 << ")\n";

cout << "Значение функции в минимуме: F = " << F(x0, y0, r) << endl;

cout << "Аналитическое решение функции: F = " << F(1.625, 0.375, 0) << endl;

return 0;

}

**5 Сходимость**

Пусть - локально единственное решение задачи поиска условного минимума, а функции и непрерывно дифференцируемы в окрестности точки . Тогда для достаточно больших найдется точка локального минимума функции в окрестности и  при

**6 Вывод**

Для решения задачи по поиску безусловного минимума функции методом штрафов была написана программа на языке C++. Для реализации поиска безусловного минимума функции был использован метод Флетчера-Ривса с . При выборе , результатом работы программы является вычисленное минимальное значение функции, равное 5,0462 в точке минимума X = (1,42781; 0,393937) (рис.1). При изменении параметра C с 2 на 5, точка минимума, значение функции в ней. При увеличении число итераций, необходимых для нахождения безусловного минимума уменьшается с 8 до 4, а вычисленное значение минимума функции F стало равно 5,29066, при этом X = (1,49393; 0,419365) (рис.2). В обоих случаях условие сходимости было выполнено, так как основная и вспомогательная функции всюду дифференцируемы.

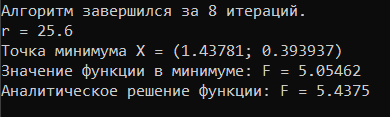


Рисунок 1 – Результат работы программы

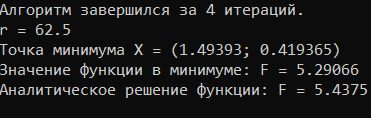


Рисунок 2 – Результат работы программы