Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизированных систем (АСУ)

**Оптимизация функций двух переменных**

Отчет по лабораторной работе № 3 по дисциплине

«Методы оптимизации»

|  |
| --- |
| Выполнили: |
| Студенты гр. 439-1 |
| Зозуля Е.Д.  Рахматуллин Т.Т. |
| 18.10.2021 г. |
|  |

|  |
| --- |
| Руководитель: |
| А.А. Шелестов |
| « »\_\_\_\_\_\_\_\_\_\_2021 г. |

2021

**Задание**

Найти минимум функции двух переменных. Использовать следующие методы:

А) два прямых метода (симплексный метод*,* метод Хука-Дживса);

Точность .

**Вариант №9**

**Описание методов оптимизации**

**Симплексный метод:**

Метод поиска по симплексу был предложен в 1962г. Спендли (W.Spendley), Хекстом (G. R. Hext) и Химсвортом (F. R. Himsworth). Этот метод называют последовательным симплекс-методом (ПСМ). Следует отметить, что указанный метод и другие подобные методы не имеют отношения к симплекс-методу линейного программирования, а сходство названий носит чисто случайный характер.

В одномерном пространстве симплекс есть отрезок прямой, в двумерном пространстве – треугольник; в трехмерном пространстве – треугольная пирамида (тетраэдр) и т. д.

Суть метода: приближение к минимальной точке с помощью изменения координат вершин симплекса.

**Алгоритм:**

**Шаг 1.** Задается исходная вершина симплекса

**Шаг 2.** В вершинах симплекса вычисляется ЦФ

**Шаг 3.** Проверяем условия:

Если «да», то конец; если «нет», то переходим на Шаг 4.

**Шаг 4.** Находится «наихудшая» вершина симплекса (при поиске минимума

«наихудшая» вершина – та, в которой значение функции максимально).

**Шаг 5**. Осуществляется расчет координат новой вершины (вершина

отражения ):

**Шаг 6.** Если точка оказывается «хуже» всех остальных точек симплекса, то осуществляется возврат к исходному симплексу с последующим его сжатием относительно «лучшей» из вершин Переход на Шаг 2. Если не является «худшей» в новом симплексе, то перейти на шаг 3.

**Метод поиска Хука – Дживса:**

Симплекс-метод и метод Хука – Дживса относятся к категории эвристических методов. В методе поиска по симплексу основное внимание было уделено геометрическому расположению пробных точек. Основная цель построения множества таких точек – определение направления, в котором должен вестись поиск. Расположение пробных точек влияет лишь на чувствительность направления поиска к изменениям топологических свойств целевой функции. Стратегию поиска по симплексу можно усовершенствовать путем введе- ния множества векторов, задающих направление поиска. Суть метода: нахождение в окрестности текущей точки наилучшей и движение в этом направлении. Если значение в окрестных точках больше чем в текущей, то происходит уменьшение шага. Процедура Хука–Дживса представляет собой комбинацию двух поисков: "Исследующий" поиск и Поиск по образцу

**Алгоритм:**

**Шаг 1.** Определить начальную точку ; приращения (шаги) ; коэффициент уменьшения шага ; параметр окончания поиска .

**Шаг 2.** Провести исследующий поиск.

**Шаг 3.** Был ли исследующий поиск удачным (найдена ли точка с меньшим значением ЦФ)? Да: переход на Шаг 5. Нет: продолжить, т.е. переход на Шаг 4.

**Шаг 4.** Проверка на окончание поиска. Выполняется ли неравенство ? Да: окончание поиска, т.е. текущая точка аппроксимирует точку экстремума x\*.

Нет: уменьшить приращение . Переход на Шаг 2.

**Шаг 5**. Провести поиск по образцу:

**Шаг 6.** Провести исследующий поиск, используя точку : в качестве временной базовой точки. Пусть в результате получена точка :

**Шаг 7.** Выполняется ли неравенство: ? Да: положить . ;Переход на Шаг 5. Нет: переход на Шаг 4.

Листинг с реализацией методов приведен ниже.

Листинг 1

#include <iostream>

#include <fstream>

#include <math.h>

#include "PolStr.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

char f[] = "x^4 - 12\*x^3 + 47\*x^2 - 60\*x";

char\* fx = CreatePolStr(f, 0);

double x = 0.000001, eps = 0.0001, xi = 0.000001;

while (abs(EvalPolStr(fx, xi, 1)) > eps)

{

xi = x - EvalPolStr(fx, x, 1) / EvalPolStr(fx, x, 2);

x = xi;

}

cout << "Минимальное значение функции = " << EvalPolStr(fx, xi, 1) << endl;

cout << "Значение аргумента = " << xi << endl;

delete[] fx;

}

Листинг 2

#include <iostream>

#include <fstream>

#include <math.h>

#include "PolStr.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

char f[] = "x^4 - 12\*x^3 + 47\*x^2 - 60\*x";

char\* fx = CreatePolStr(f, 0);

double a = 0, b = 5, eps = 0.0001, xi = 0, fxi = 1;

while (abs(fxi) > eps)

{

xi = (a + b) / 2;

fxi = EvalPolStr(fx, xi, 1);

if (fxi > 0)

b = xi;

else if (fxi < 0)

a = xi;

}

cout << "Минимальное значение функции = " << fxi << endl;

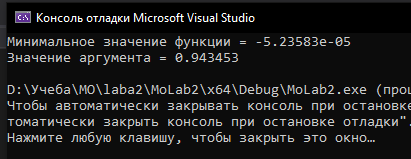
cout << "Значение аргумента = " << xi << endl;

delete[] fx;

}

**Результат работы**

**Метод Ньютона:**



**Метод средней точки:**

