Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

Вариант 15

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ В.В. Тынченко

подпись, дата инициалы, фамилия

Студент КИ15–16Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Т.В. Радионов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

# Задача лабораторной работы

Разработать программу, реализующую метод покоординатного спуска.

Найти безусловный экстремум функции, выбранной в соответствии с заданием, с использованием разработанной программы.

# Листинг программы

Point.cs

using System;

namespace ConsoleApp1

{

class Point

{

public double X { get; set; }

public double Y { get; set; }

/// <summary>

/// Конструктор

/// </summary>

/// <param name="x"></param>

/// <param name="y"></param>

public Point(double x, double y)

{

X = x;

Y = y;

}

/// <summary>

/// Длина

/// </summary>

/// <returns></returns>

public double GetLength()

{

return Math.Sqrt(X \* X + Y \* Y);

}

/// <summary>

/// Сложение

/// </summary>

/// <param name="p1"></param>

/// <param name="p2"></param>

/// <returns></returns>

public static Point operator +(Point p1, Point p2)

{

return new Point(p1.X + p2.X, p1.Y + p2.Y);

}

/// <summary>

/// Вычитание

/// </summary>

/// <param name="p1"></param>

/// <param name="p2"></param>

/// <returns></returns>

public static Point operator -(Point p1, Point p2)

{

return new Point(p1.X - p2.X, p1.Y - p2.Y);

}

/// <summary>

/// Скалярное произведение

/// </summary>

/// <param name="p1"></param>

/// <param name="p2"></param>

/// <returns></returns>

public static double operator \*(Point p1, Point p2)

{

return p1.X \* p2.X + p1.Y \* p2.Y;

}

/// <summary>

/// Произведение на число

/// </summary>

/// <param name="p1"></param>

/// <param name="c"></param>

/// <returns></returns>

public static Point operator \*(Point p1, double c)

{

return new Point (p1.X \* c, p1.Y \* c);

}

public static Point operator \*(double c, Point p1)

{

return new Point(c \* p1.X, c \* p1.Y);

}

/// <summary>

/// Деление на число

/// </summary>

/// <param name="p1"></param>

/// <param name="p2"></param>

/// <returns></returns>

public static Point operator /(double c, Point p1)

{

return new Point(c / p1.X, c / p1.Y);

}

public static Point operator /(Point p1, double c)

{

return new Point(p1.X / c, p1.Y / c);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace ConsoleApp1

{

class Program

{

/// <summary>

/// Function

/// </summary>

/// <param name="p"></param>

/// <returns></returns>

static double Func(Point p) => Math.Pow(p.X, 2) + p.X \* p.Y + Math.Pow(p.Y, 2);//Math.Pow(p.X, 2) + 5 \* p.X \* p.Y + Math.Pow(p.Y, 2);

/// <summary>

/// Градиент функции в точке

/// </summary>

/// <param name="p"></param>

/// <returns></returns>

static Point GradFunc(Point p)

{

//double derX = 2 \* p.X + 5 \* p.Y;

//double derY = 5 \* p.X + 2 \* p.Y;

double dx = 0.001;

double dy = 0.001;

double derX = (Func(new Point(p.X + dx, p.Y)) - Func(p)) / dx;

double derY = (Func(new Point(p.X, p.Y + dy)) - Func(p)) / dy;

return new Point(derX, derY);

}

/// <summary>

/// Вывести результат в консоль

/// </summary>

/// <param name="resultPoint"></param>

/// <param name="steps"></param>

static void PrintResult(Point resultPoint, int steps)

{

double result = Func(resultPoint);

Console.WriteLine("X = " + resultPoint.X + "; Y = " + resultPoint.Y);

Console.WriteLine("Результат: {0:F10}", result);

Console.WriteLine("Точность: {0:F10}", Math.Abs(Func(new Point(0, 0)) - result));

Console.WriteLine("Количество шагов: {0}", steps);

}

/// <summary>

/// Метод покоординатного спуска

/// </summary>

/// <param name="x0"></param>

/// <param name="eps"></param>

static void Run(Point x00, double eps = 0.0, double eps1 = 0.1, double eps2 = 0.15, int M = 10)

{

// !!! Параметр eps = 0 не нужен в алгоритме

int reiteration = 0; // для шага 11 отслеживание повторения условия для j и j - 1

int steps = 0;

// Шаг 1

int n = 2;

if (M % n == 1)

{

Console.WriteLine("Вы вабрали M не кратное n");

return;

}

List<List<Point>> x = new List<List<Point>>();

x.Add(new List<Point>(n + 1));

x[0].Add(x00); //x[0,0] = x00

Point gradF = GradFunc(x[0][0]);

// Шаг 2

int j = 0;

int k = 0; // k <= n - 1

List<Point> e = new List<Point> { null, new Point(1, 0), new Point(0, 1) };

List<double> t = new List<double>();

// Шаг 3

step3:

if (j >= M)

{

PrintResult(x[j][0], steps); // x[j][k]

return;

}

// Шаг 4

k = 0;

// Шаг 5

step5:

// б

if (k > n - 1)

{

x.Add(new List<Point> { null });

x[j + 1][0] = x[j][n]; //!!! ошибка в алгоритме, там написано x[j + 1][k] = x[j][n]

j++;

//steps++;

goto step3;

}

// Шаг 6

gradF = GradFunc(x[j][k]);

// Шаг 7

if (gradF.GetLength() <= eps1) // || gradF(x[j,k]) || <= eps1

{

PrintResult(x[j][k], steps);

return;

}

// Шаг 8

t.Add(0.5); // t[k] = 0.5

x[j].Add(null); // t[j][k+1] = null

while (true)

{

// Шаг 9

double der = 0;

if (k + 1 == 1) //dF(p)/dx

{

double dx = 0.001;

der = (Func(new Point(x[j][k].X + dx, x[j][k].Y)) - Func(x[j][k])) / dx;

//der = 2 \* x[j][k].X + 5 \* x[j][k].Y;

}

else if (k + 1 == n) // dF(p)/dy

{

double dy = 0.001;

der = (Func(new Point(x[j][k].X, x[j][k].Y + dy)) - Func(x[j][k])) / dy;

//der = 5 \* x[j][k].X + 2 \* x[j][k].Y;

}

x[j][k + 1] = x[j][k] - t[k] \* der \* e[k + 1];

// Шаг 10

if (Func(x[j][k + 1]) - Func(x[j][k]) < 0)

{

break;

}

else

{

t[k] = t[k] / 2;

continue;

}

}

// Шаг 11

if ((x[j][k + 1] - x[j][k]).GetLength() < eps2 && Math.Abs(Func(x[j][k + 1]) - Func(x[j][k])) < eps2)

{

reiteration++;

if (reiteration == 2) // j - 1 тоже соответствовало данному условию

{

PrintResult(x[j][k + 1], steps);

return;

}

}

else

{

reiteration = 0;

}

k++;

steps++;

goto step5;

}

/// <summary>

/// Main method

/// </summary>

/// <param name="args"></param>

static void Main(string[] args)

{

Stopwatch sw = new Stopwatch();

Console.WriteLine("Тест 1");

Console.WriteLine("Берем стандартные параметры");

sw.Reset();

sw.Start();

Run(new Point(10, 10));

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 2");

Console.WriteLine("Увеличиваем eps1");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.5);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 3");

Console.WriteLine("Уменьшаем eps1");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.01);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 4");

Console.WriteLine("Увеличиваем eps2");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.1, 0.5);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 5");

Console.WriteLine("Уменьшаем eps2");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.1, 0.01);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 6");

Console.WriteLine("Уменьшаем M");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.1, 0.15, 4);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 7");

Console.WriteLine("Увеличиваем M");

sw.Reset();

sw.Start();

Run(new Point(10, 10), 0, 0.1, 0.15, 20);

sw.Stop();

Console.WriteLine("Скорость алгоритма: {0} ms\n", sw.ElapsedMilliseconds);

Console.ReadKey();

}

}

}

# Скриншоты результата выполнения программы

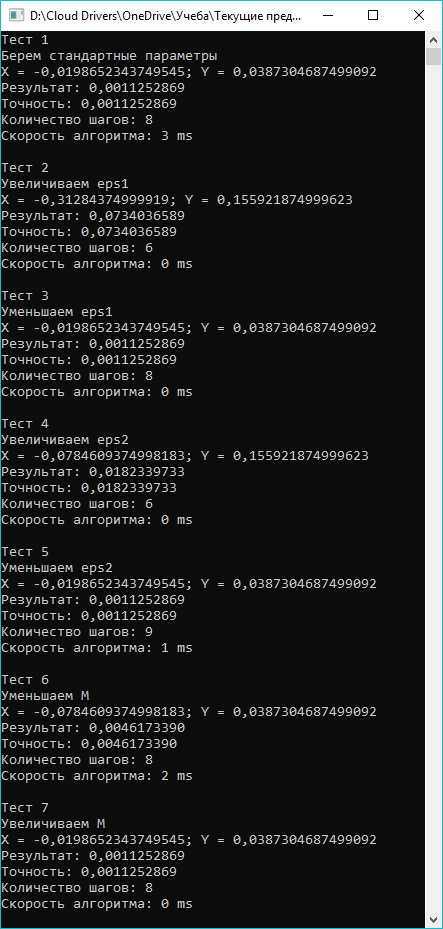


Рисунок 1 – Пример работы программы

# Выводы

В результате проделанной работы была разработана программа, которая выполнила поиск безусловного экстремума (минимума) заданной функции методом покоординатного спуска. В ходе тестов были установлены следующие зависимости при изменениях параметров (таблица 1): при увеличении параметра точности eps1 уменьшается количество итераций и понижается точность результата, а при уменьшении eps1 изменений не происходит; при увеличении/уменьшении параметра точности eps2 уменьшается/увеличивается количество итераций и понижается/повышается точность результата; изменение параметра M привело к ухудшению точности результата выполнения программы, увеличение никак не повлияло. Во всех тестах скорость алгоритма наибыстрейшая и в сравнении не нуждается. Стоит отметить, что в некоторых функциях увеличение параметра M может привести к более точному результату.

Таблица 1 – Исследование алгоритма путем изменения параметров

|  |  |
| --- | --- |
| Тест 1  Берем стандартные параметры  X = -0,0198652343749545; Y = 0,0387304687499092  Результат: 0,0011252869  Точность: 0,0011252869  Количество шагов: 8  Скорость алгоритма: 3 ms | Тест 2  Увеличиваем eps1  X = -0,31284374999919; Y = 0,155921874999623  Результат: 0,0734036589  Точность: 0,0734036589  Количество шагов: 6  Скорость алгоритма: 0 msms |
| Тест 3  Уменьшаем eps1  X = -0,0198652343749545; Y = 0,0387304687499092  Результат: 0,0011252869  Точность: 0,0011252869  Количество шагов: 8  Скорость алгоритма: 0 ms | Тест 4  Увеличиваем eps2  X = -0,0784609374998183; Y = 0,155921874999623  Результат: 0,0182339733  Точность: 0,0182339733  Количество шагов: 6  Скорость алгоритма: 0 ms |
| Тест 5  Уменьшаем eps2  X = -0,0198652343749545; Y = 0,0387304687499092  Результат: 0,0011252869  Точность: 0,0011252869  Количество шагов: 9  Скорость алгоритма: 1 ms | Тест 6  Уменьшаем M  X = -0,0784609374998183; Y = 0,0387304687499092  Результат: 0,0046173390  Точность: 0,0046173390  Количество шагов: 8  Скорость алгоритма: 2 ms |
| Тест 7  Увеличиваем M  X = -0,0198652343749545; Y = 0,0387304687499092  Результат: 0,0011252869  Точность: 0,0011252869  Количество шагов: 8  Скорость алгоритма: 0 ms |  |