Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

Вариант 1

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 Флетчера-Ривса.

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;

using MathNet.Numerics;

namespace ConsoleApp1

{

class Program

{

/// <summary>

/// Функция

/// </summary>

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

/// <returns></returns>

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

/// <summary>

/// Градиент функции в точке

/// </summary>

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

/// <returns></returns>

static Point GradFunc(Point p)

{

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>

/// Функция для поиска экстремума

/// </summary>

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

/// <param name="t"></param>

/// <param name="d"></param>

/// <returns></returns>

static double FuncExtremum(Point p, double t, Point d) => Math.Pow(p.X + t \* d.X, 2) +

(p.X + t \* d.X) \* (p.Y + t \* d.Y) + 2 \* Math.Pow(p.Y + t \* d.Y, 2);

/// <summary>

/// Поиск экстремума методом одномерной минимизации (золотого сечения)

/// </summary>

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

/// <param name="d"></param>

/// <returns></returns>

static double FindExtremum(Point p, Point d)

{

double a = p.X + 100; // Левая граница по X начальной точки

double b = p.X - 100; // Правая граница по X начальной точки

double eps = 0.001;

double x = 0;

double y = 0;

double Fy = 0;

double z = 0;

double Fz = 0;

double length = 0;

int k = 0;

// Step 1

length = b - a;

// Step 2

k = 0;

// Step 3

y = a + ((3 - Math.Sqrt(5)) / 2) \* (b - a);

z = a + b - y;

while (true)

{

// Step 4

Fy = FuncExtremum(p, y, d);

Fz = FuncExtremum(p, z, d);

// Step 5

if (Fy <= Fz)

{

// a = a;

b = z;

z = y;

y = a + b - y;

}

else

{

a = y;

// b = b;

y = z;

z = a + b - z;

}

// Step 6

double delta = Math.Abs(a - b);

if (delta <= eps)

{

x = (a + b) / 2;

break;

}

else

{

k++;

}

}

return x;

}

/// <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 Флетчера-Ривса

/// </summary>

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

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

static void Run(Point x0, double eps1 = 0.1, double eps2 = 0.15, int M = 10)

{

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

x.Add(x0);

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

d.Add(null);

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

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

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

// Шаг 1

Point gradF = GradFunc(x0); // ???

// Шаг 2

int k = 0;

while (true)

{

// Шаг 3

gradF = GradFunc(x[k]);

// Шаг 4

if (gradF.GetLength() < eps1)

{

// а

PrintResult(x[k], k);

return;

}

// б

// Шаг 5

if (k >= M)

{

// а

PrintResult(x[k], k);

return;

}

// б

if (k == 0)

{

// Шаг 6

d[0] = -1 \* gradF; // !!! ошибка в алгоритме - нигде не сказано о переходе на шаг 9

}

else

{

// Шаг 7

beta.Add(0);

beta[k - 1] = Math.Pow(GradFunc(x[k]).GetLength(), 2) / Math.Pow(GradFunc(x[k - 1]).GetLength(), 2);

// Шаг 8

d.Add(null);

d[k] = -1 \* GradFunc(x[k]) + beta[k - 1] \* d[k - 1];

}

// Шаг 9

t.Add(0);

t[k] = FindExtremum(x[k], d[k]); // Поиск параметра t методом Фибоначчи

// Шаг 10

x.Add(null);

x[k + 1] = x[k] + t[k] \* d[k];

// Шаг 11

// а

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

{

reiteration++;

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

{

PrintResult(x[k], k);

return;

}

}

else

{

k++;

continue;

}

}

}

/// <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("Увеличим eps1");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 3");

Console.WriteLine("Уменьшим eps1");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 4");

Console.WriteLine("Увеличим eps2");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 5");

Console.WriteLine("Уменьшим eps2");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 6");

Console.WriteLine("Уменьшим M");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 7");

Console.WriteLine("Увеличим M");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.ReadKey();

}

}

}

# Скриншоты результата выполнения программы

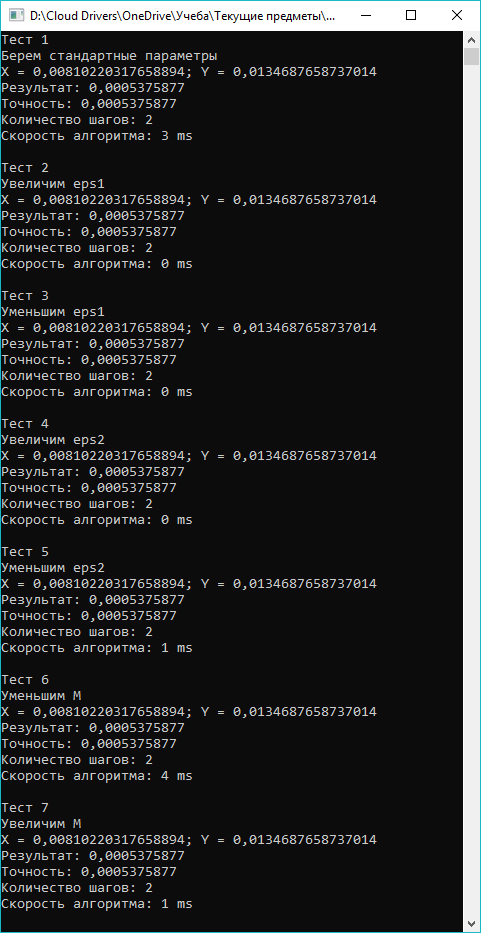


Рисунок 1 – Пример работы программы

# Выводы

В результате проделанной работы была разработана программа, которая выполнила поиск безусловного экстремума (минимума) заданной функции методом Флетчера-Ривса. В ходе тестов были установлены следующие зависимости при изменениях параметров (таблица 1): при изменении точности eps1 количество итераций и точность результата не изменяются (однако это не означает, что параметр ни на что не влияет, просто исследуемая функция не сложна и алгоритм даже при худшей точности выполняется успешно); при увеличении/уменьшении параметра точности eps2 наблюдаем тоже самое, что и для eps1; изменение параметра M также никак не повлияло для данной функции. Примечательно, что данный алгоритм довольно эффективен для простых функций с одним локальным минимумом: находит за минимальное число шагов с высокой точностью даже при плохой заданной точности. Скорость алгоритма в каждом тесте высокая и не нуждается в сравнении.

Таблица 1 – Исследование алгоритма путем изменения параметров

|  |  |
| --- | --- |
| Тест 1  Берем стандартные параметры  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 3 ms | Тест 2  Увеличим eps1  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 0 ms |
| Тест 3  Уменьшим eps1  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 0 ms | Тест 4  Увеличим eps2  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 0 ms |
| Тест 5  Уменьшим eps2  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 1 ms | Тест 6  Уменьшим M  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 4 ms |
| Тест 7  Увеличим M  X = 0,00810220317658894; Y = 0,0134687658737014  Результат: 0,0005375877  Точность: 0,0005375877  Количество шагов: 2  Скорость алгоритма: 1 ms |  |

Также стоит сравнить данный алгоритм с методом покоординатного спуска. Однако стоит отметить, что исследуемые функции хоть и различны, но их отличие лишь в угле поворота относительно осей X и Y. А значит можно сравнить эти два алгоритма, результаты приведены в таблице 2:

Таблица 2 – Сравнение различных алгоритмов при значениях x0 = (10,10) eps1=0.1, eps2 = 0.15, M = 10)

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Время** | **Точность результата** | **Шаги** |
| Покоординатного спуска | 3 мс | 0.0011 | 8 |
| Флетчера-Ривса | 3мс | 0.00053 | 2 |