Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

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 Пауэлла.

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>

/// Функция для одномерной минимизации

/// </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.Y + t \* d.Y, 2) + 6 \* Math.Pow(p.X + t \* d.X, 2) + 5 \* (p.X + t \* d.X) + 2 \* (p.Y + t \* d.Y) + 1;

/// <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 = 0.01;

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>

/// Function

/// </summary>

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

/// <returns></returns>

static double Func(Point p) => Math.Pow(p.Y, 2) + 6 \* Math.Pow(p.X, 2) + 5 \* p.X + 2 \* p.Y + 1;

/// <summary>

/// Ранг матрицы

/// </summary>

/// <param name="n"></param>

/// <param name="m"></param>

/// <param name="matrix"></param>

/// <returns></returns>

static int Rang(List<Point> listMatrix)

{

// Конвертируем список в матрицу

int n = listMatrix.Count - 1;

int m = 2;

double[,] matrix = new double[n, m];

for (int l = 0; l < n; l++)

{

matrix[l, m - 2] = listMatrix[l + 1].X;

matrix[l, m - 1] = listMatrix[l + 1].Y;

}

// Ищем ранг с помощью модифицированного метода Гаусса

const double EPS = 1E-9;

int rang = Math.Max(n, m);

bool[] line\_used = new bool[n];

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

{

int j;

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

if (!line\_used[j] && Math.Abs(matrix[j, i]) > EPS)

break;

if (j == n)

--rang;

else

{

line\_used[j] = true;

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

matrix[j, p] /= matrix[j, i];

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

if (k != j && Math.Abs(matrix[k, i]) > EPS)

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

matrix[k, p] -= matrix[j, p] \* matrix[k, i];

}

}

return rang;

}

/// <summary>

/// Метод Пауэлла

/// </summary>

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

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

static void Run(Point x0, double eps = 0.01)

{

Point resultPoint;

// Step 1

int n = 2;

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

d.Add(null);

d.Add(new Point(1, 0));

d.Add(new Point(0, 1));

d[0] = d[n];

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

x.Add(x0);

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

y.Add(x[0]); // y[0] = x[0];

int i = 0;

int k = 0;

while (true)

{

// Step 2

y.Add(null); // y[i + 1]

double t = 0;

t = FindExtremum(y[i], d[i]); // Поиск параметра t методом Фибоначчи

y[i + 1] = y[i] + t \* d[i];

// Step 3

// a

if (i < n - 1)

{

i++;

continue; // goto Step 2

}

// b

else if (i == n - 1)

{

if (y[n] == y[0])

{

resultPoint = y[n];

break; // y[n]

}

else

{

i++;

continue; // goto Step 2

}

}

// c

else if (i == n)

{

if (y[n + 1] == y[1])

{

resultPoint = y[n + 1];

break; // y[n + 1]

}

// goto Step 4

}

// Step 4

x.Add(null);

x[k + 1] = y[n + 1];

// a

if ((x[k + 1] - x[k]).GetLength() < eps)

{

resultPoint = x[k + 1];

break; // x[k + 1]

}

// b

else

{

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

for (int j = 0; j < d.Count; j++)

d\_v.Add(null);

d\_v[0] = d\_v[n] = y[n + 1] - y[1];

for (int j = 1; j < n; j++) // !!! n - 1

d\_v[j] = d[j + 1];

if (Rang(d\_v) == n)

{

for (int j = 0; j < d\_v.Count; j++)

d[j] = d\_v[j];

k++;

i = 0;

y.Clear();

y.Add(x[1]); //y[0] = x[1];

continue; // goto Step 2

}

else

{

// d[j] = d[j]

k++;

i = 0;

y.Clear();

y.Add(x[1]); //y[0] = x[1];

continue; // goto Step 2

}

}

}

double result = Func(resultPoint);

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

Console.WriteLine("Точность: {0:F10}", Math.Abs(Func(new Point(-0.4166666666, -1)) - result));

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

}

/// <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(0, 0));

sw.Stop();

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

Console.WriteLine("Тест 2");

Console.WriteLine("Ухудшаем точность в 10 раз");

sw.Reset();

sw.Start();

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

sw.Stop();

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

Console.WriteLine("Тест 3");

Console.WriteLine("Улучшаем точность в 10 раз");

sw.Reset();

sw.Start();

Run(new Point(0, 0), 0.001);

sw.Stop();

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

Console.WriteLine("Тест 4");

Console.WriteLine("Берем другую точку");

sw.Reset();

sw.Start();

Run(new Point(8, 9));

sw.Stop();

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

Console.ReadKey();

}

}

}

# Скриншоты результата выполнения программы

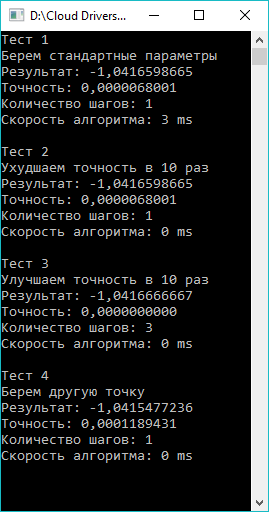


Рисунок 1 – Пример работы программы

# Выводы

В результате проделанной работы была разработана программа, которая выполнила поиск безусловного экстремума (минимума) заданной функции методом Пауэлла. В ходе тестов были установлены следующие зависимости при изменениях параметров (таблица 1): при увеличении/уменьшении параметра точности уменьшается/увеличивается количество итераций и понижается/повышается точность результата; изменение точки на более удаленную от локального экстремума привело к ухудшению точности результата выполнения программы. Также в 1 тесте замечено неправдоподобное значение скорости алгоритма, но это скорее относится к особенности работы таймеров в среде Visual Studio, а во всех остальных случаях скорость алгоритма <1 мс.

Таблица 1 – Исследование алгоритма путем изменения параметров

|  |  |
| --- | --- |
| Берем стандартные параметры  Результат: -1,0416598665  Точность: 0,0000068001  Количество шагов: 1  Скорость алгоритма: 3 ms | Берем другую точку  Результат: -1,0415477236  Точность: 0,0001189431  Количество шагов: 1  Скорость алгоритма: 0 ms |
| Увеличиваем точность  Результат: -1,0246516994375  Точность: 0,0170149672  Количество шагов: 2  Скорость алгоритма: 0 ms | Уменьшаем точность  Результат: -1,04166550290998  Точность: 0,0000011638  Количество шагов: 10  Скорость алгоритма: 0 ms |
| Улучшаем точность в 10 раз  Результат: -1,0416666667  Точность: 0,0000000000  Количество шагов: 3  Скорость алгоритма: 0 ms | Ухудшаем точность в 10 раз  Результат: -1,0416598665  Точность: 0,0000068001  Количество шагов: 1  Скорость алгоритма: 0 ms |

В качестве сравнения данного алгоритма с другими использованы в каждом методе одинаковые точки старта алгоритма, одинаковые параметры (уникальные параметры взяты по рекомендациям авторов алгоритмов) и сравнивались результаты при изменении параметра точности. Результаты приведены на таблице 2.

Таблица 2 – Сравнение различных алгоритмов при различных значениях параметра точности (0.1 / 0.01 / 0.001)

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Время** | **Точность результата** | **Шаги** |
| Хука-Дживса | 1 мс / 1 мс / 1 мс | 0.0065 / 0.001 / 0.0000016 | 8 / 14 / 20 |
| Нелдера-Мида | 1 мс / 2 мс / 3 мс | 0.026 / 0.0163 / 0.0004 | 7 / 9 / 10 |
| Розенброка | <1 мс / 1 мс / 1 мс | 0.017 / 0.000137 / 0.00000116 | 2 / 6 / 10 |
| Пауэлла | <1 мс / <1 мс / <1 мс | 0.0000068 / 0.0000068 / 0.0000000000 | 1 / 1 / 3 |