Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

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 Нелдера-Мида.

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

# Листинг программы

Point.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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;

}

}

}

Program.cs

using System;

using System.Diagnostics;

namespace ConsoleApp1

{

class Program

{

/// <summary>

/// Function

/// </summary>

/// <param name="x1"></param>

/// <param name="x2"></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;

static double Run(Point[] points, double alpha = 1, double beta = 0.5, double gamma = 2, double eps = 0.01)

{

// Step 1

int n = points.Length - 1;

int k = 0;

while(true)

{

// Step 2

Point min = points[0]; // x[l]

Point max = points[0]; // x[h]

Point secondMax = points[0]; // x[s]

for (int i = 0; i < n + 1; i++)

{

if (Func(points[i]) < Func(min))

{

min = points[i];

}

else if (Func(points[i]) > Func(max))

{

secondMax = max;

max = points[i];

}

else if (Func(points[i]) > Func(secondMax) && Func(points[i]) < Func(max))

{

secondMax = points[i];

}

}

// Step 3

double sumX = 0;

double sumY = 0;

for (int i = 0; i < n + 1; i++)

{

if (points[i] == max)

continue;

sumX = sumX + points[i].X;

sumY = sumY + points[i].Y;

}

double cgX = sumX / n;

double cgY = sumY / n;

Point cg = new Point(cgX, cgY); // x[n+2]

// Step 4

double sum = 0;

for (int i = 0; i < n + 1; i++)

sum = sum + Math.Pow(Func(points[i]) - Func(cg), 2);

double sigma = Math.Pow(((double)1 / (n + 1)) \* sum, 0.5);

if (sigma <= eps)

{

double result = Func(min);

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

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

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

return result;

}

else

{

// Step 5

double reflX = cg.X + alpha \* (cg.X - max.X);

double reflY = cg.Y + alpha \* (cg.Y - max.Y);

Point refl = new Point(reflX, reflY); // x[n+3]

// Step 6

if (Func(refl) <= Func(min))

{

double stretchX = cg.X + gamma \* (refl.X - cg.X);

double stretchY = cg.Y + gamma \* (refl.Y - cg.Y);

Point stretch = new Point(stretchX, stretchY); // x[n+4]

if (Func(stretch) < Func(max))

{

points[Array.IndexOf(points, max)] = stretch; // x[h] = x[n+4]

k++;

}

else

{

points[Array.IndexOf(points, max)] = refl; // x[h] = x[n+3]

k++;

}

}

else if (Func(secondMax) < Func(refl) && Func(refl) <= Func(max))

{

double squeezeX = cg.X + beta \* (max.X - cg.X);

double squeezeY = cg.Y + beta \* (max.Y - cg.Y);

Point squeeze = new Point(squeezeX, squeezeY); // x[n+5]

points[Array.IndexOf(points, max)] = squeeze; // x[h] = x[n+5]

k++;

}

else if (Func(min) < Func(refl) && Func(refl) <= Func(secondMax))

{

points[Array.IndexOf(points, max)] = refl; // x[h] = x[n+3]

k++;

}

else // Func(refl) > Func(max)

{

for (int i = 0; i < n + 1; i++)

{

points[i].X = min.X + 0.5 \* (points[i].X - min.X);

points[i].Y = min.Y + 0.5 \* (points[i].Y - min.Y);

}

k++;

}

}

}

}

/// <summary>

/// Main method

/// </summary>

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

static void Main(string[] args)

{

Stopwatch sw = new Stopwatch(); // Timer for algorithm's speed

double result;

Console.WriteLine("Тест 1");

Console.WriteLine("Берем стандартные параметры (коэф. Нелдера и Мида)");

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) } );

sw.Stop();

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

Console.WriteLine("Тест 2");

Console.WriteLine("Увеличим размер многоугольника в 100 раз");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-100, -50), new Point(0, 100), new Point(100, -50) });

sw.Stop();

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

Console.WriteLine("Тест 3");

Console.WriteLine("Уменьшим размер многоугольника в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-0.5, -0.25), new Point(0, 0.5), new Point(0.5, -0.25) });

sw.Stop();

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

Console.WriteLine("Тест 4");

Console.WriteLine("Уменьшим отражение в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 0.5);

sw.Stop();

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

Console.WriteLine("Тест 5");

Console.WriteLine("Увеличим отражение в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 2);

sw.Stop();

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

Console.WriteLine("Тест 6");

Console.WriteLine("Уменьшим сжатие в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.25);

sw.Stop();

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

Console.WriteLine("Тест 7");

Console.WriteLine("Увеличим сжатие в 1.5 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.75); //

sw.Stop();

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

Console.WriteLine("Тест 8");

Console.WriteLine("Уменьшим растяжение в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.5, 1);

sw.Stop();

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

Console.WriteLine("Тест 9");

Console.WriteLine("Увеличим растяжение в 2 раза");

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.5, 4);

sw.Stop();

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

Console.WriteLine("Тест 10");

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

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.5, 2, 0.001);

sw.Stop();

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

Console.WriteLine("Тест 11");

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

sw.Reset();

sw.Start();

result = Run(new Point[] { new Point(-1, -0.5), new Point(0, 1), new Point(1, -0.5) }, 1, 0.5, 2, 0.1);

sw.Stop();

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

Console.ReadKey();

}

}

}

# Скриншоты результата выполнения программы

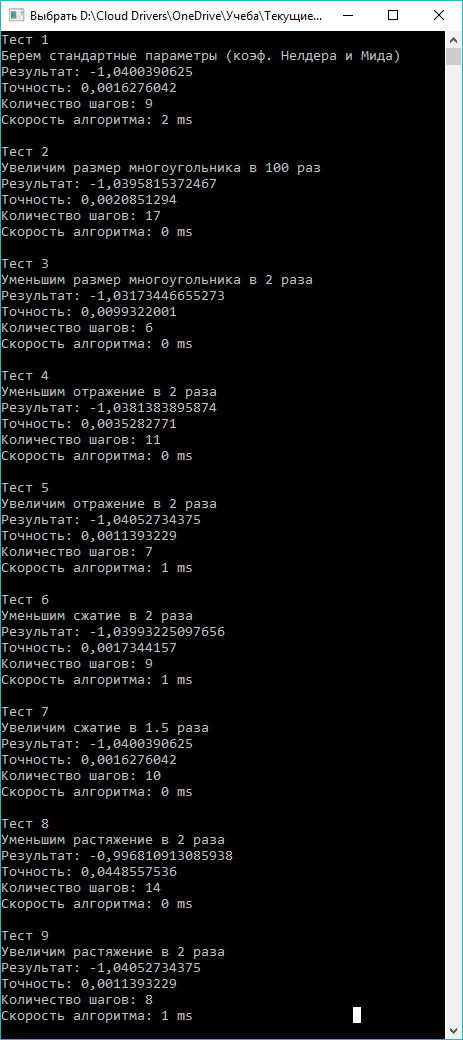


Рисунок 1 – Пример работы программы

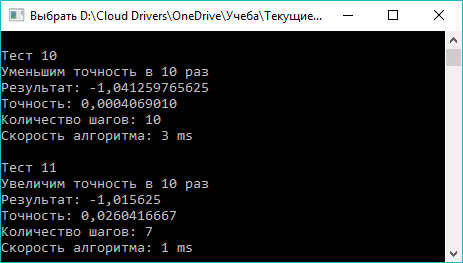


Рисунок 2 – Пример работы программы

# Выводы

В результате проделанной работы была разработана программа, которая выполнила поиск безусловного экстремума (минимума) заданной функции методом Нелдера-Мида. В ходе тестов были установлены следующие зависимости по отношению к стандартным значениям по Нелдеру и Миду: если мы изменяем размер многоугольника в большую сторону, то увеличивается количество шагов, а если уменьшаем, то и количество шагов уменьшается, однако в обоих случаях падает точность результатов; увеличение коэффициента отражения приводит к увеличению количества шагов и падению точности результатов, а уменьшение – наоборот; коэффициент сжатия существенного влияния на алгоритм не оказывает, однако его стоит брать в пределах 1, иначе программа не сможет произвести необходимые расчеты; уменьшение коэффициента растяжения приводит к ухудшению точности результата и увеличению количества шагов, а увеличение – наоборот; уменьшение точности приводит к более точным результатам, однако это увеличивает количество итераций и время выполнения алгоритма, а уменьшение – наоборот.

Если сравнить данный алгоритм с алгоритмом Хука-Дживса, то можно заметить, что данный алгоритм выигрывает в количестве итераций, однако уступает в точности результатов при одинаковых параметрах точности.