Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

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 Хука-Дживса.

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

# Листинг программы

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(double x1, double x2) => Math.Pow(x2, 2) + 6 \* Math.Pow(x1, 2) + 5 \* x1 + 2 \* x2 + 1;

/// <summary>

/// Запуск алгоритма

/// </summary>

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

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

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

/// <param name="lambda"></param>

/// <param name="alpha"></param>

/// <returns></returns>

static double Run(double x = 0, double y = 0, double eps = 0.01, double lambda = 1, double alpha = 2)

{

// Step 1

// x = 0;

// y = x;

// eps = 0.1;

double delta = 1; // Направление шага

// lambda = 1;

// alpha = 2;

int k = 0; // Счетчик

// Step 2

double newX = x;

double newY = y;

while (true)

{

// Функция принимаем наименьшее значение при шаге вправо или влево?

if (Func(newX + delta, newY) < Func(newX, newY))

{

newX = newX + delta;

}

else

{

newX = newX - delta;

}

// Функция принимаем наименьшее значение при шаге вверх или вниз?

if (Func(newX, newY + delta) < Func(newX, newY))

{

newY = newY + delta;

}

else

{

newY = newY - delta;

}

// Step 3

// Шаг оказался успешным? (значение функции меньше стало, чем было?)

if (Func(newX, newY) < Func(x, y))

{

// Step 4

double oldX = x;

double oldY = y;

// Делаем текущей точкой наш шаг

x = newX;

y = newY;

// Задаем следующую точку

newX = newX + lambda \* (newX - oldX);

newY = newY + lambda \* (newY - oldY);

k++;

}

else

{

// Step 5

// Величина шага меньше области окончания (точности)?

if (delta <= eps)

{

Console.WriteLine("Total steps: {0}", k);

Console.WriteLine("X={0}; Y={1}; F(x,y)={2}", x, y, Func(x, y));

return Func(x, y);

}

else

{

delta = delta / alpha;

// Задаем следующую точку текущей

newX = x;

newY = 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");

sw.Start();

result = Run();

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 2");

Console.WriteLine("Берем другую точку");

sw.Start();

result = Run(8, 9);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 3");

Console.WriteLine("Берем более высокую точность");

sw.Start();

result = Run(0, 0, 0.001);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 4");

Console.WriteLine("Берем менее высокую точность");

sw.Start();

result = Run(0, 0, 0.1);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 5");

Console.WriteLine("Берем больше ускоряющий множитель");

sw.Start();

result = Run(0, 0, 0.01, 2);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 6");

Console.WriteLine("Берем меньше ускоряющий множитель");

sw.Start();

result = Run(0, 0, 0.01, 0.5);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 7");

Console.WriteLine("Берем больше коэффициент уменьшения шага");

sw.Start();

result = Run(0, 0, 0.01, 1, 4);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.WriteLine("Тест 8");

Console.WriteLine("Берем меньше коэффициент уменьшения шага");

sw.Start();

result = Run(0, 0, 0.01, 1, 1.2);

sw.Stop();

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

Console.WriteLine("Algorithm's speed: {0} ms\n", sw.ElapsedMilliseconds);

Console.ReadKey();

}

}

}

# Скриншоты результата выполнения программы

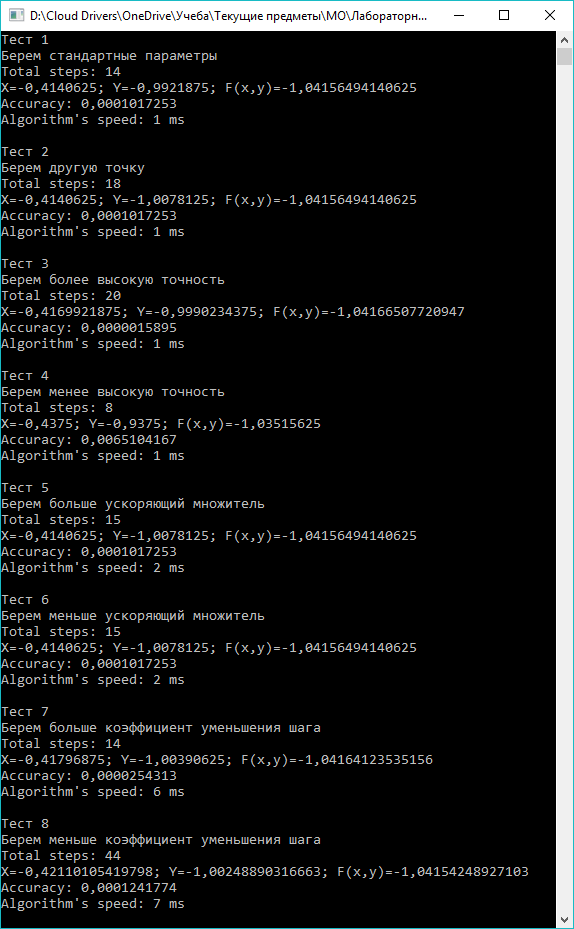


Рисунок 1 – Пример работы программы

# Выводы

В результате проделанной работы была разработана программа, которая выполнила поиск безусловного экстремума заданной функции методом Хука-Дживса. В ходе тестов были установлены следующие зависимости: если мы изменяем начальную точку, то изменяется только количество итераций; параметр точности также влияет на количество итераций программы, причем линейно, однако также напрямую зависит и точность результатов; изменение ускоряющего множителя в большую сторону ухудшает точность результатов, а в меньшую наоборот, причем на время и количество шагов этот параметр существенно не влияет; коэффициент изменения шага влияет абсолютно на все результаты измерений, в том числе и время выполнения программы: при уменьшении от 2 до 1 все характеристики ухудшаются, при увеличении до 10 и больше падает точность результатов и количество шагов. В целом алгоритм эффективен и быстр для поиска минимума функции.