МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО «Брянский государственный технический университет»

Кафедра «Компьютерные технологии и системы»

Направление подготовки

10.05.04- «Информационно-аналитические системы безопасности»

Профиль – «Автоматизация информационно-аналитической деятельности»

**Курсовая работа**

по дисциплине: «численные методы»

на тему:

**«Автоматизация поиска экстремума методом градиентного спуска»**

|  |  |
| --- | --- |
|  | Выполнил:  Студент группы О-19-ИАС-аид-С  Колотвин Артём Павлович  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г |
|  | Принял:  к.т.н., доцент каф. «КТС»  Леонов Юрий Алексеевич  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**Брянск 2022**

## ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу по дисциплине «численные методы».

**Студент** Колотвин А.П. **Группа** О-19-ИАС-аид-С

### Наименование и область применения

Полное наименование системы: «автоматизация поиска экстремума методом градиентного спуска».

### Основания для разработки

Основанием для разработки является демонстрация теоретических и практических навыков, приобретенных за курс обучения, по дисциплине «численные методы».

### Назначение разработки

Назначением разработки является создание WPF приложения на языке программирования C#, предназначенной для нахождения экстремума функции методом градиентного спуска.

### Технические требования к программе или программному изделию

Программа должна включать в себя следующие функциональные характеристики:

* реализация метода градиентного спуска для функций вида y=f(x) и z=f(x,y);
* должны строиться графики исходных функций и точки градиентного спуска, наглядно отображающие принцип работы метода;
* помимо построения 2D функций, программа должна уметь строить графики для трехмерных функций (поверхностей) с помощью библиотеки Matlab;
* для новых пользователей должно быть реализовано руководство пользователя;
* пользовательский интерфейс должен быть простым, интуитивно понятным и комфортным для использования.

**Руководитель Леонов Ю.А.**

## СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc106662815)

[1. АНАЛИТИЧЕСКИЙ РАЗДЕЛ 5](#_Toc106662816)

[1.1. Обзор и анализ существующих программных решений 5](#_Toc106662817)

[1.2. Определение функциональных требований к разрабатываемой программной системе 6](#_Toc106662818)

[2. КОНСТРУКТОРСКИЙ РАЗДЕЛ 8](#_Toc106662819)

[2.1. Функциональная схема работы программы 8](#_Toc106662820)

[2.2. Диаграмма потоков данных 10](#_Toc106662821)

[2.3. Описание математических методов и алгоритмов предметной области 11](#_Toc106662822)

[2.4. Выбор графического и пользовательского интерфейса 12](#_Toc106662823)

[3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 16](#_Toc106662824)

[3.1. Описание классов и пользовательских методов 16](#_Toc106662825)

[3.2. Разработка алгоритмов отдельных подзадач 17](#_Toc106662826)

[3.3. Руководство пользователя 17](#_Toc106662827)

[4. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ 19](#_Toc106662828)

[4.1. Виды контроля качества разрабатываемого ПО 19](#_Toc106662829)

[4.2. Методика проведения и результаты тестирования 20](#_Toc106662830)

[ЗАКЛЮЧЕНИЕ 21](#_Toc106662831)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 22](#_Toc106662832)

[ПРИЛОЖЕНИЯ 23](#_Toc106662833)

[Приложение 1. Листинг программы 23](#_Toc106662834)

[Приложение 2. Интерфейс программы 35](#_Toc106662835)

## ВВЕДЕНИЕ

В настоящее время искусственные нейронные сети широко используются при решении самых разнообразных задач особенно там, где обычные алгоритмические решения оказываются неэффективными или вовсе невозможными. Решения на основе искусственных нейронных сетей становятся все более совершенными и популярными, поэтому можно предположить, что и в будущем искусственные нейронные сети будут широко использоваться за счет лучшего понимания их основополагающих принципов. Одной из основных характеристик нейронных сетей является их обучение, процесс получения новых знаний на основе уже имеющихся. И, пожалуй, самым распространенным методом обучения нейронных сетей является метод градиентного спуска.

Градиентный спуск — метод нахождения минимального значения функции потерь. Иначе можно сказать, что градиентный спуск - это итерационный алгоритм оптимизации для нахождения локального минимума дифференцируемой функции. Идея состоит в том, чтобы делать повторные шаги в направлении, противоположном градиенту функции в текущей точке, потому что это направление самого крутого спуска. Градиент же – это вектор, который своим направлением указывает на направление возрастания некоторой скалярной величины.

Таким образом, целью данной курсовой работы является разработка программного обеспечения, позволяющего автоматизировать процесс нахождения экстремума функции методом градиентного спуска.

Основными задачи разработки являются обзор существующих программных решений с целью выявления основных функциональных требований к разрабатываемому программному продукту, разработка диаграмм, функциональных схем работы программы и алгоритмов. А также разработка графического интерфейса, руководства пользователя и последующее проведение тестирования готового программного комплекса.

## АНАЛИТИЧЕСКИЙ РАЗДЕЛ

### Обзор и анализ существующих программных решений

Для определения основных функциональных возможностей требуемых для реализации и выявлений необходимых конкурентных преимуществ разрабатываемого программного комплекса, необходимо рассмотреть готовые программные решения и выявить их основные достоинства и недостатки.

#### Онлайн калькулятор градиентного спуска Math.semestr

Назначение этого сервиса заключается в нахождении минимума функции методом градиентного (наискорейшего) спуска (рис.1.1).

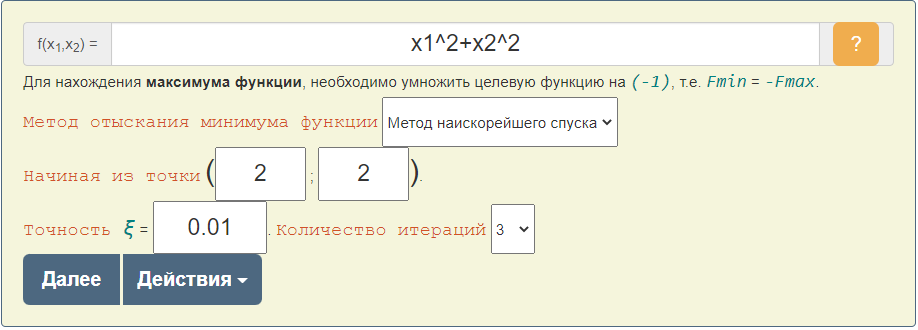


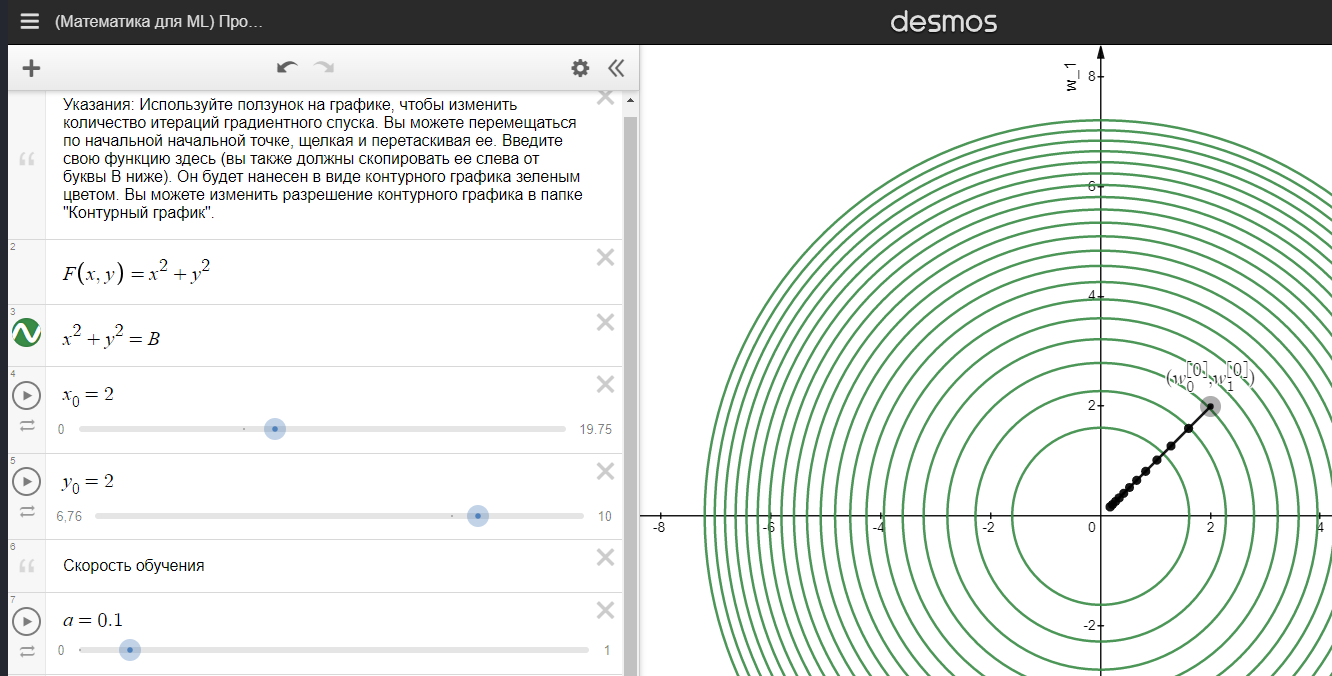
Рис.1.1. Сервис Math.semestr

Из достоинств данного решения можно подчеркнуть простой пользовательский интерфейс и возможность оформления решения задачи в формате MS WORD, однако, данный сервис решает только функции вида z = f(x,y), и не решает соответственно функции вида y=f(x). Еще одним весомым недостатком является отсутствие наглядного графика работы метода, что упростило бы понимание работы градиентного спуска.

#### Онлайн сервис Desmos

Следующий сервис содержит различные математические калькуляторы, одним из которых является калькулятор нахождения минимума функции методом градиентного спуска (рис.1.2).

Рис.1.2. Сервис Desmos



Данное программное решение исправляет недостатки прошлого рассматриваемого сервиса, однако не лишено и собственных недостатков. Плюсом данного решения является функция построения графика для наглядности, однако все графики отображаются в 2D формате, что не всегда дает полного понимания работы метода.

### Определение функциональных требований к разрабатываемой программной системе

На основе рассмотренных готовых программных решений и выявления их достоинств и недостатков, можно выделить набор функциональных требований к разрабатываемой программной системе. К таким требованиям относятся:

* программа должна уметь вычислять экстремум для функций вида y=f(x) и z=f(x,y);
* работа метода градиентного спуска должна быть наглядной пользователю, следовательно, требуется разработать алгоритм построения графиков функций;
* для большей наглядности работы метода при вычислении трехмерных функций, требуется строить поверхность (график в 3D пространстве);
* пользовательский интерфейс должен быть простым, понятным и комфортным для использования;
* для новых пользователей, еще не знакомых с программным обеспечением, требуется разработать руководство по работе с программой.

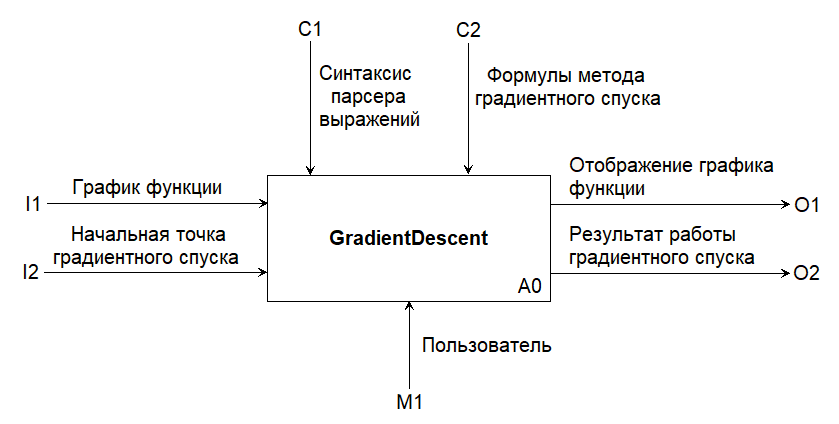
## КОНСТРУКТОРСКИЙ РАЗДЕЛ

### Функциональная схема работы программы

IDEF0 – это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Данный стандарт позволяет представить программный комплекс в виде набора функциональных блоков, каждый из которых может осуществлять взаимодействие с другими функциональными блоками посредством четырех видов интерфейсов (входа, управления, механизма и выхода).

Функциональная схема работы программы «Gradient Descent» первого уровня декомпозиции представлена на рис.2.1.

Рис.2.1. IDEF0 первого уровня декомпозиции



Интерфейс входа (input) описывает исходные данные или объекты для выполнения функции, результат выполнения которой описан в интерфейсе выхода (output). На данной диаграмме на вход поступают график функции и начальная точка градиентного спуска. На выходе – отображение графика функции и результат работы градиентного спуска. Сверху описывается интерфейс управления (control), который описывает правила и ограничения, которыми программа руководствуется. На диаграмме это синтаксис парсера математический выражений и формулы метода градиентного спуска. Последний интерфейс, который описывается снизу – это интерфейс механизма (mechanism). Он описывает ресурсы, используемые в процессе выполнения функции, при этом эти ресурсы не должны изменяться. На диаграмме этим механизмом является пользователь.

На диаграмме второго уровня декомпозиции (рис.2.2) эта система разбивается на отдельные компоненты.

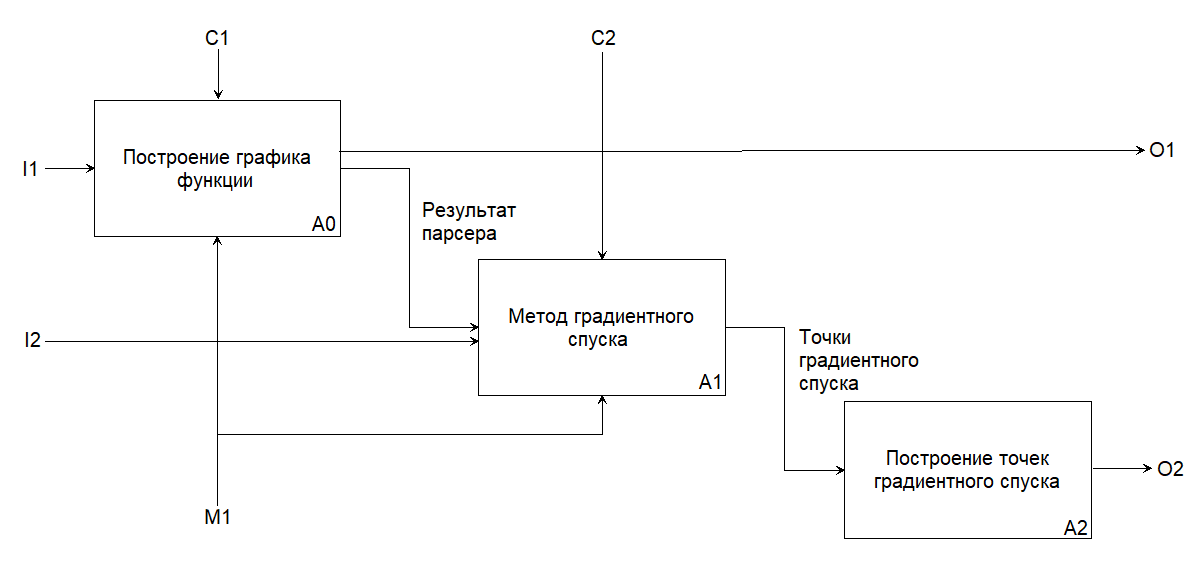


Рис.2.2. IDEF0 второй уровень декомпозиции

Для блока А0 на вход поступает график функции, управляется этот блок пользователем, а руководствуется он синтаксисом парсера математический выражений. На выходе – отображение графика функции и результат парсера, который поступает на вход в блок А1.

Для блока А1 входными данными являются функция, полученная в результате работы парсера и начальная точка градиентного спуска. Механизмом для данного блока также является пользователь. Интерфейс управления включается в себя формулы метода градиентного спуска. Выходными данными здесь являются точки градиентного спуска, поступающие на вход в блок А2.

Последний блок А2 получает на вход точки градиентного спуска, полученные в результате работы метода и на выходе отображает результат работы градиентного спуска.

### Диаграмма потоков данных

Диаграмма потоков данных – это один из основных инструментов структурного анализа и проектирования информационных систем. При создании диаграммы потоков данных применяется стандарт DFD (data flow diagrams). Стандарт предполагает использование четырёх элементов: функций, потоков данных, хранилищ данных и внешних сущностей.

Модель DFD, как и большинство других структурных моделей является иерархичной, то есть, каждый процесс может быть подвергнут декомпозиции. Первый уровень декомпозиции диаграммы потоков данных для программы «Gradient Descent» приведен на рис.2.3.

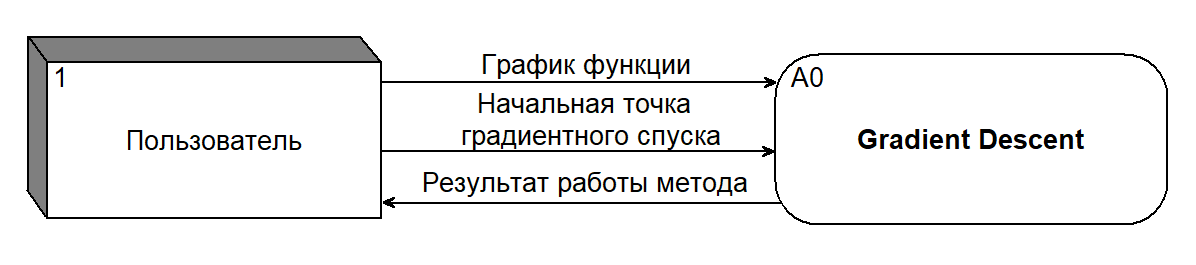


Рис.2.3. DFD первого уровня декомпозиции

На вход в блок А0 от пользователя поступают исходный график функции и начальная точка градиентного спуска, а на выходе возвращает пользователю результат работы метода градиентного спуска.

Диаграммы потоков данных обеспечивают удобный способ описания передаваемой информации, как между частями моделируемой системы, так и между системой и внешним миром.

На втором уровне декомпозиции (рис.2.4) блок А0 подвергается детализации.

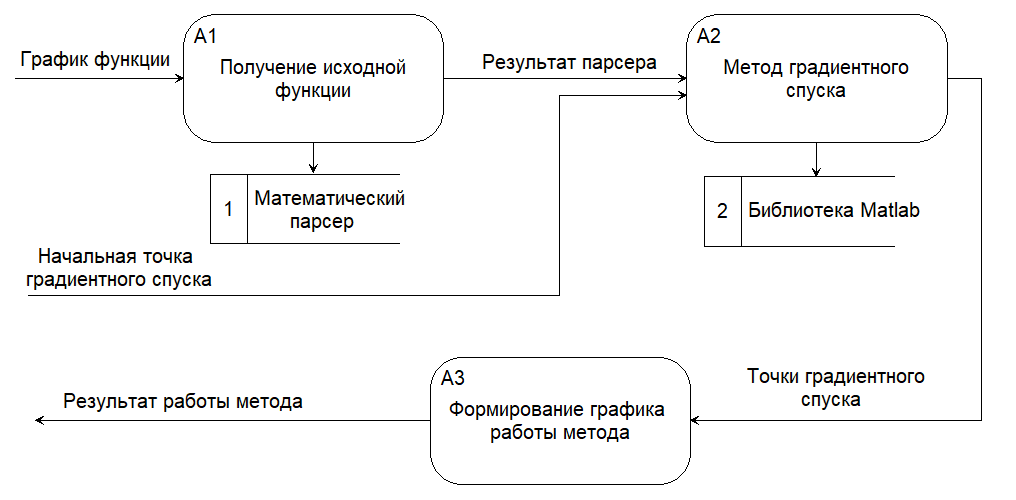


Рис.2.4. DFD второго уровня декомпозиции

### Описание математических методов и алгоритмов предметной области

Как уже было сказано ранее, градиентный спуск – это итерационный алгоритм нахождения локального минимума некоторой функции. Для определения направления спуска используется понятие градиента – вектора, показывающего направление наибольшее возрастания некоторой величины. Если это значение взять с противоположным знаком, то оно является антиградиентом – вектором, показывающим наискорейшее убывание функции. Суть метода градиентного спуска заключается в следующем:

1. На исходном графике некоторой функции берется начальная точка градиентного спуска, для которого вычисляется значение градиента в этой точке (градиент есть сумма частных производных некоторой функции).
2. Вычисляется следующая точка градиентного спуска в направлении наискорейшего убывания функции по формуле:

где – текущая точка, k – константа, отвечающая за скорость нахождения минимума функции (в литературе learning rate или «скорость обучения»).

1. Для найденной точки вычисляется значение градиента в этой точке.
2. Шаги 2 и 3 повторяются до тех пор, выполняется условие:

где – точность, задаваемая пользователем.

Блок-схема алгоритма градиентного спуска имеет следующий вид (рис.2.5):

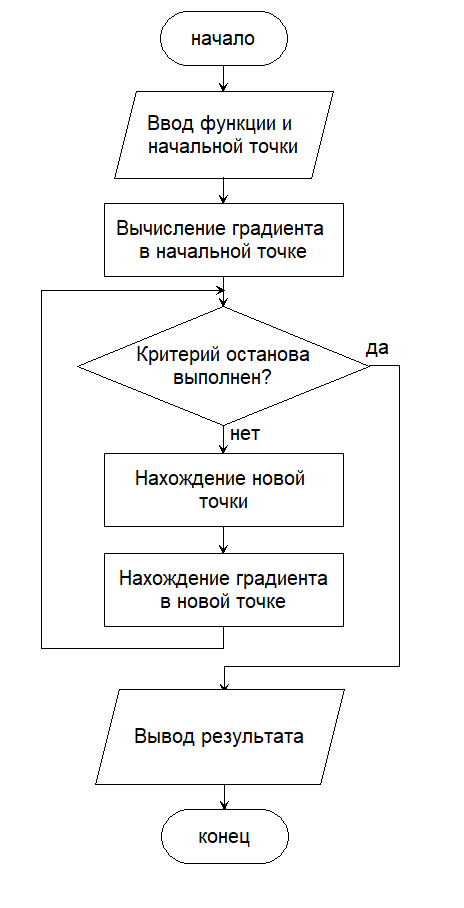


Рис.2.5. Блок-схема алгоритма градиентного спуска

### Выбор графического и пользовательского интерфейса

Разработка графического и пользовательского интерфейса является не менее важной частью создания программного продукта. Интерфейс пользователя (user interface или сокращенно UI) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. UI должны быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне.

Первый вариант пользовательского интерфейса представлял собой сырой вариант, который определял какие входные данные имеет программа (рис.2.6).

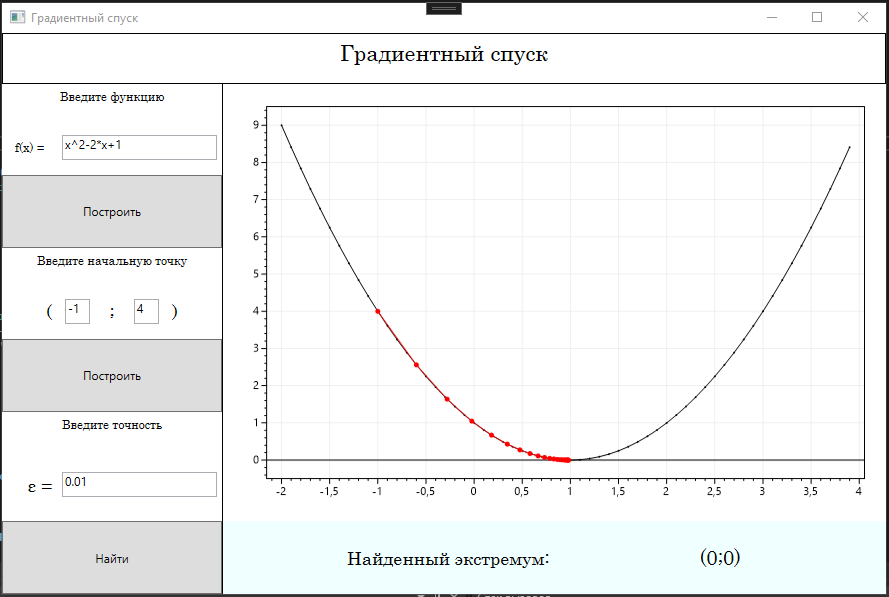


Рис.2.6. Первый вариант пользовательского интерфейса

Данный интерфейс был доработан, убраны лишние элементы, а также был разработан общий дизайн для остальных окон. В итоге было разработано меню, в котором пользователь выбирал функцию какого вида необходимо построить и рассчитать точки градиентного спуска. Финальный интерфейс программы представлен на рис.2.7-2.9.



Рис.2.7. Главное меню

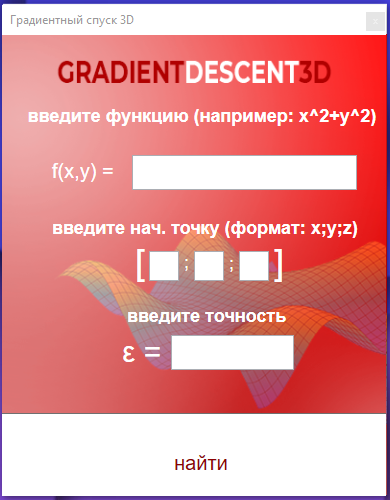
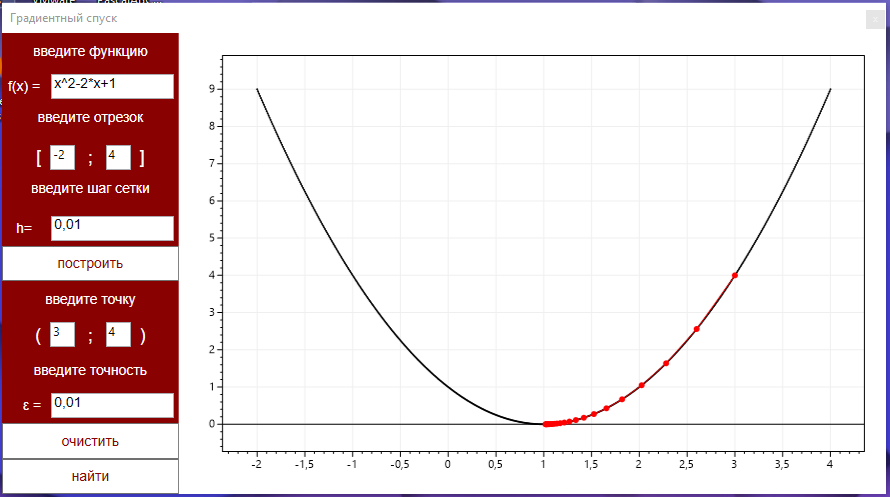


Рис.2.8. Построение функции f(x,y)

Рис.2.9. Построение функции f(x)



## ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

### Описание классов и пользовательских методов

Gradient() – класс, хранящий свойства и методы для определения градиента функции в указанной точке.

GradientFinder(string function, Point point) – метод для нахождения градиента функции в переданной точке. Возвращает значение градиента в точке с типом double.

Gradient3DFinder(string function, Point3D point) – метод для нахождения градиента трехмерной функции. Возвращает значение градиента в трехмерной точке с типом double.

GradientDescent – класс, хранящий точки, найденные с помощью метода градиентного спуска.

FindingNewGradientDescentPoint(string function, Point point) – метод нахождения новой точки градиентного спуска. Принимает в качестве параметров строковую функцию и текущую точку. Возвращает новую точку с типом структуры Point.

GradientDescentMethod(string function, Point point, double eps) – метод, реализующий градиентный спуск, принимающий в качестве параметров строковую функцию, начальную точку и точность.

GradientDescent3D – класс, хранящий трехмерные точки, найденные с помощью метода градиентного спуска для трехмерной функции.

FindingNewGradientDescent3DPoint(string function, Point3D point) – метод нахождения новой трехмерной точки градиентного спуска. Принимает в качестве параметров строковую функцию и текущую точку. Возвращает новую точку с типом структуры Point3D.

GradientDescent3DMethod(string function, Point3D point, double eps) – метод, реализующий градиентный спуск для трехмерной функции, принимающий в качестве параметров строковую функцию, трехмерную начальную точку и точность.

Структура Point, хранящая координаты X и Y точки.

Структура Point3D, хранящая координаты X, Y и Z трехмерной точки.

Структура GradientVector, хранящая координаты I и J вектора градиента.

### Разработка алгоритмов отдельных подзадач

Отдельной подзадачей в данной курсовой работе является построение трехмерных графиков функций вида z=f(x,y). Для реализации этой задачи требовалось связываться с Matlab через API.

Matlab – это пакет прикладных программ для решения задач технических вычислений. С помощью данного приложения необходимо создать функцию, которая принимает строковую функцию и точки градиентного спуска. Строковый формат функции преобразуется в дескриптор функции – специальный тип данных Matlab, с помощью которого затем строится поверхность.

function result = GradientDescent(str,x, y, z)

func = str2func(str);

fsurf(func);

hold on

result = plot3(x,y,z, 'red', 'linewidth', 3, 'marker', 'o');

hold

end

Данный скрипт необходимо сохранить как .m файл, затем вызвав команду «deploytool» создается библиотека, хранящая в себе класс, содержащий данный скрипт.

После успешного создания библиотеки подключаются два файла: созданный .dll файл и файл MWArray.dll, предназначенный для приведения типов данных C# к типу Matlab.

### Руководство пользователя

Важной частью создания программного продукта является разработка руководства для новых пользователей. Оно должно давать представление пользователю о том, для чего предназначен программный комплекс, что такое градиентный спуск и какой функционал он использует.

Руководство пользователя доступно из главного меню и имеет следующий вид (рис.3.1):

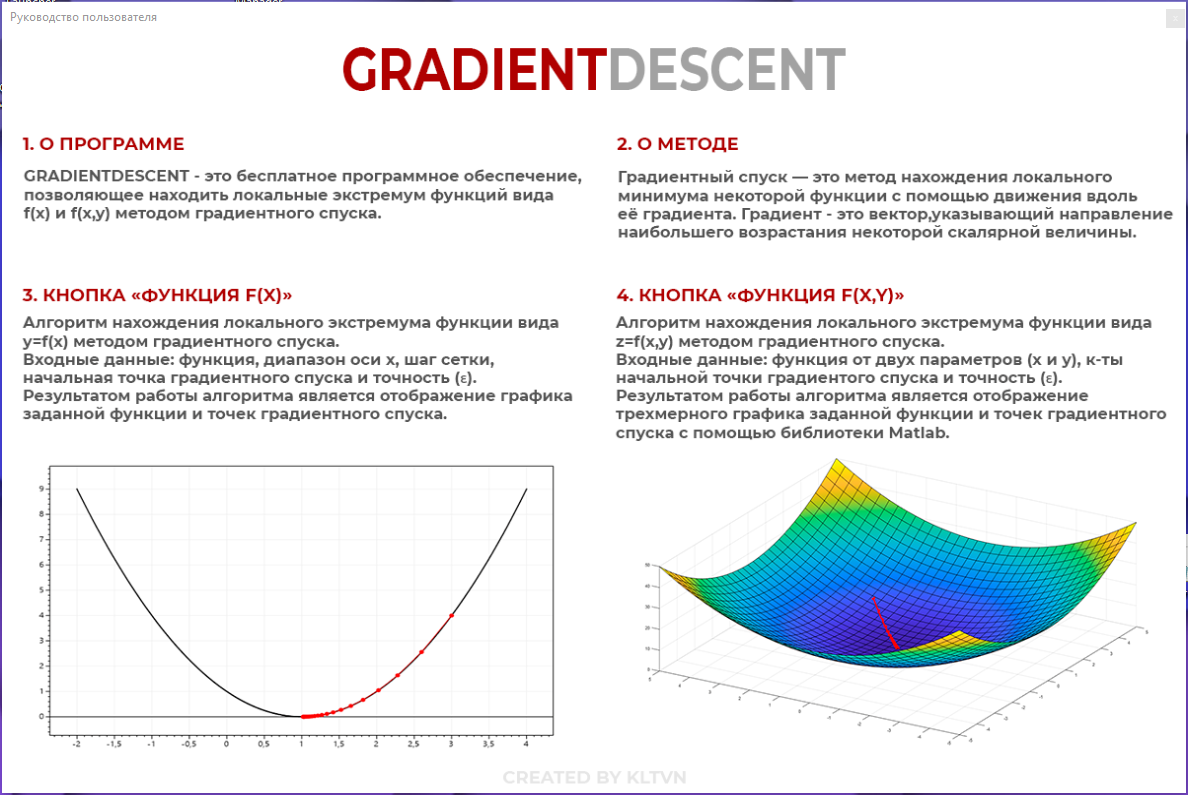


Рис.3.1. Руководство пользователя

## ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ

### Виды контроля качества разрабатываемого ПО

Тестирование – очень важный и трудоёмкий этап процесса разработки программного обеспечения, так как правильно тестирование позволяет выявить подавляющее большинство ошибок, допущенных при составлении программ.

Различают два подхода к формированию тестов: структурный и функциональный. Каждый из этих подходов имеет свои особенности и области применения.

Структурный подход базируется на том, что известна структура тестируемого программного обеспечения, в том числе и его алгоритмы. В этом случае тесты строят так, чтобы проверить правильность реализации заданной логики в коде программы.

Функциональный подход основывается на том, что структура программного обеспечения неизвестна, в этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также подходом, управляемым данными, так как при его использовании тесты строят на базе различных способов декомпозиции множества данных.

### Методика проведения и результаты тестирования

Во время тестирования программы были использованы следующие принципы:

* необходимо подбирать тесты не только для правильных, но и для неверных данных;
* предполагаемые результаты тестирования должны быть известны ещё до начала тестов;
* каждый тест должен быть досконально изучен и перепроверен, чтобы не пропустить малозаметную на первый взгляд ошибку в программе;
* по возможности, следует избегать тестирования самим автором программы;
* тестирование не должно превращаться в рутинную работу, требуется проявлять повышенный интерес и подходить к этому процессу творчески.

В результате тестирования было выявлено, что передаваемая функция в строковом формате методу Matlab не принимает этот параметр, а, следовательно, не может построить график функции. Выяснило, что для построения графиков функций Matlab использует собственный тип данных – дескриптор функции, к которому требовалось приводить функцию в строковом виде.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был разработан программный комплекс для автоматизации поиска экстремума методом градиентного спуска. При его разработке был проведен анализ предметной области с использованием литературных и информационных источников по методам машинного обучения нейронных сетей, в частности метода градиентного спуска.

Был проведен сравнительный анализ существующих программных решений с последующим выявлением функциональных требований к разрабатываемому программному продукту, затем были разработаны функциональные схемы работы программы и диаграммы потоков данных нескольких уровней декомпозиции. Описаны математические методы и алгоритмы, в частности была разработана блок-схема алгоритма нахождения экстремума функции методом градиентного спуска.

Были разработаны варианты пользовательского интерфейса программы с последующим выявлением общего дизайна и стиля графических элементов. Разработано руководство для новых пользователей, объясняющее назначение данного программного обеспечения и его функциональные возможности.

В ходе выполнения работы был получен опыт в интеграции пользовательских библиотек Matlab в приложение .NET.

Таким образом, была достигнута поставленная цель, заключающаяся в разработке программного комплекса для автоматизации поиска экстремума функции методом градиентного спуска, выполнены задачи по её достижению и учтены все функциональные требования, которыми должен обладать конечный программный продукт.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Буч, Г. Введение в UML от создателей языка / Г.Буч, Д.Рамбо, И.Якобсон. – ДМК Пресс, 2015. – 496 с.
2. Либерти, Д. Программирование C#. – 2-е изд.: ЛитРес, 2017. – 679 с.
3. Павловская, Т.А. C#. Программирование на языке высокого уровня. – СПБ.: Питер, 2016. – 432 с.
4. Современные численные методы оптимизации. Метод универсального градиентного спуска: учебное пособие / А. В. Гасников. – М.: МФТИ, 2018. – 166 с.
5. Троелсен, Э. Язык программирования C#и платформы .NET и .NET Core. – Диалектика, 2018. – 1328 с.
6. Документация Matlab [Электронный ресурс]. Режим доступа: https://www.mathworks.com/help/matlab/
7. Сервис Math.semestr [Электронный ресурс]. Режим доступа: https://math.semestr.ru/optim/steepest-descent.php
8. Сервис Desmos [Электронный ресурс]. Режим доступа: https://www.desmos.com/calculator
9. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Изд-во стандартов, 1991. – 20 с.
10. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам. – М.: Изд-во стандартов, 2007. – 31 с.
11. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления.- Стандартинформ, 2008.- 17 с.
12. Р 50.1.028-2001. «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования».- М.:Изд-во стандартов, 2002. - 54 с.

## ПРИЛОЖЕНИЯ

### Приложение 1. Листинг программы

Листинг класса Gradient

using MathNet.Symbolics;

using System;

using System.Collections.Generic;

using System.Linq;

namespace gradientDescent

{

public class Gradient

{

public List<double> GradientValuesList { get; private set; }

public List<double> Gradient3DValuesList { get; private set; }

public List<GradientVector> gradientVectorsList { get; private set; }

public Gradient()

{

GradientValuesList = new List<double>();

Gradient3DValuesList = new List<double>();

gradientVectorsList = new List<GradientVector>();

}

public double GradientFinder(string function, Point point)

{

var func = SymbolicExpression.Parse(function);

var variables = func.CollectVariables();

double result;

var x = func.Differentiate(variables.First());

var res\_der = x.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", point.X } }).RealValue;

GradientValuesList.Add(res\_der);

result = res\_der;

return result;

}

public double Gradient3DFinder(string function, Point3D point)

{

var func = SymbolicExpression.Parse(function);

var variables = func.CollectVariables();

double result;

double i, j;

var x = func.Differentiate(variables.First());

var y = func.Differentiate(variables.Last());

i = x.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", point.X } }).RealValue;

j = y.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.Last()}", point.Y } }).RealValue;

GradientVector gradientVector = new GradientVector();

gradientVector.I = i;

gradientVector.J = j;

gradientVectorsList.Add(gradientVector);

var gradientResult = Math.Sqrt(Math.Pow(x.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", point.X } }).RealValue, 2) +

Math.Pow(y.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.Last()}", point.Y } }).RealValue, 2));

Gradient3DValuesList.Add(gradientResult);

result = gradientResult;

return result;

}

}

}

Листинг класса GradientDescent

using System;

using System.Collections.Generic;

using System.Linq;

using MathNet.Symbolics;

namespace gradientDescent

{

public class GradientDescent

{

private Gradient gradient = new Gradient();

public List<Point> GradientDescentPointsList { get; } // Список точек градиентного спуска

public GradientDescent()

{

GradientDescentPointsList = new List<Point>();

}

// Метод градиентного спуска

public void GradientDescentMethod(string function, Point point, double eps)

{

gradient.GradientFinder(function, point);

GradientDescentPointsList.Add(point);

Point secondPoint = FindingNewGradientDescentPoint(function, point);

gradient.GradientFinder(function, secondPoint);

while (Math.Abs(gradient.GradientValuesList.Last() - gradient.GradientValuesList.ElementAt(gradient.GradientValuesList.Count() - 2)) > eps)

{

Point newPoint = FindingNewGradientDescentPoint(function, GradientDescentPointsList.Last());

gradient.GradientFinder(function, newPoint);

}

}

// Нахождение новой точки градиентного спуска

private Point FindingNewGradientDescentPoint(string function, Point point)

{

var func = SymbolicExpression.Parse(function);

var variables = func.CollectVariables();

Point newPoint;

newPoint.X = point.X - 0.1 \* gradient.GradientValuesList.Last();

newPoint.Y = func.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", newPoint.X } }).RealValue;

GradientDescentPointsList.Add(newPoint);

return newPoint;

}

}

}

Листинг класса GradientDescent3D

using System;

using System.Collections.Generic;

using System.Linq;

using MathNet.Symbolics;

namespace gradientDescent

{

public class GradientDescent3D

{

private Gradient gradient = new Gradient();

public List<Point3D> GradientDescent3DPointsList { get; } // Список точек градиентного спуска

public GradientDescent3D()

{

GradientDescent3DPointsList = new List<Point3D>();

}

// Метод градиентного спуска

public void GradientDescent3DMethod(string function, Point3D point, double eps)

{

gradient.Gradient3DFinder(function, point);

GradientDescent3DPointsList.Add(point);

Point3D secondPoint = FindingNewGradientDescent3DPoint(function, point);

gradient.Gradient3DFinder(function, secondPoint);

while (Math.Abs(gradient.Gradient3DValuesList.Last() - gradient.Gradient3DValuesList.ElementAt(gradient.Gradient3DValuesList.Count() - 2)) > eps)

{

Point3D newPoint = FindingNewGradientDescent3DPoint(function, GradientDescent3DPointsList.Last());

gradient.Gradient3DFinder(function, newPoint);

}

}

// Нахождение новой точки градиентного спуска

private Point3D FindingNewGradientDescent3DPoint(string function, Point3D point)

{

var func = SymbolicExpression.Parse(function);

var variables = func.CollectVariables();

Point3D newPoint;

newPoint.X = point.X - 0.1 \* gradient.gradientVectorsList.Last().I;

newPoint.Y = point.Y - 0.1 \* gradient.gradientVectorsList.Last().J;

newPoint.Z = func.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", newPoint.X }, { $"{variables.Last()}", newPoint.Y } }).RealValue;

GradientDescent3DPointsList.Add(newPoint);

return newPoint;

}

}

}

Листинг структуры Point

namespace gradientDescent

{

public struct Point

{

public double X;

public double Y;

public Point(double x, double y)

{

X = x;

Y = y;

}

}

}

Листинг структуры Point3D

namespace gradientDescent

{

public struct Point3D

{

public double X;

public double Y;

public double Z;

public Point3D(double x, double y, double z)

{

X = x;

Y = y;

Z = z;

}

}

}

Листинг структуры GradientVector

namespace gradientDescent

{

public struct GradientVector

{

public double I;

public double J;

public GradientVector(double i, double j)

{

I = i;

J = j;

}

}

}

Листинг MainWindow.xaml

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using ScottPlot.WPF;

using MathNet.Symbolics;

using System.Drawing;

using System.Windows.Input;

namespace gradientDescent

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

GradientDescent gradient = new GradientDescent();

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

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

public MainWindow()

{

InitializeComponent();

}

// Метод построения графика функции

private void GenerateFunction(string function, int startX, int endX, double step)

{

var func = SymbolicExpression.Parse(function);

var variables = func.CollectVariables();

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

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

for (double i = startX; i < endX; i += step)

{

dataX.Add(i);

double y = func.Evaluate(new Dictionary<string, FloatingPoint>() { { $"{variables.First()}", i } }).RealValue;

dataY.Add(y);

}

Plot.Plot.AddScatter(dataX.ToArray(), dataY.ToArray(), color: Color.Black, markerSize: 2);

Plot.Refresh();

Plot.Plot.AddHorizontalLine(0, color: Color.Black);

}

// Метод построения точек градиентного спуска

private void DrawingPointsOfGradientDescentMethod()

{

foreach (Point point in gradient.GradientDescentPointsList)

{

gradientDescentPointX.Add(point.X);

gradientDescentPointY.Add(point.Y);

}

Plot.Plot.AddScatter(gradientDescentPointX.ToArray(), gradientDescentPointY.ToArray(), color: Color.Red, markerSize: 6);

Plot.Refresh();

MessageBox.Show($"Экстремум в точке: ({Math.Round(gradient.GradientDescentPointsList.Last().X, 3)}; {Math.Round(gradient.GradientDescentPointsList.Last().Y, 3)})");

}

private void ExecuteButton\_Click(object sender, RoutedEventArgs e)

{

string function = InitialFunctionTextBox.Text;

try

{

double pointX = Convert.ToDouble(InitialPointX.Text);

double pointY = Convert.ToDouble(InitialPointY.Text);

double eps = Convert.ToDouble(AccuracyTextBox.Text);

Point point = new Point(pointX, pointY);

gradient.GradientDescentMethod(function, point, eps);

DrawingPointsOfGradientDescentMethod();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка!" + ex.Message);

}

}

private void BuildFunctionButton\_Click(object sender, RoutedEventArgs e)

{

string function = InitialFunctionTextBox.Text;

int startX = int.Parse(startXTextBox.Text);

int endX = int.Parse(endXTextBox.Text);

try

{

double step = Convert.ToDouble(stepTextBox.Text);

GenerateFunction(function, startX, endX, step);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка!" + ex.Message);

}

}

private void ClearAllButton\_Click(object sender, RoutedEventArgs e)

{

InitialFunctionTextBox.Clear();

startXTextBox.Clear();

endXTextBox.Clear();

stepTextBox.Clear();

InitialPointX.Clear();

InitialPointY.Clear();

AccuracyTextBox.Clear();

Plot.Plot.Clear();

gradientDescentPointX.Clear();

gradientDescentPointY.Clear();

}

private void textboxes\_previewTextInput(object sender, TextCompositionEventArgs e)

{

if (!(Char.IsDigit(e.Text, 0) || (e.Text == ",") && (!AccuracyTextBox.Text.Contains(',') && AccuracyTextBox.Text.Length != 0)

|| (!stepTextBox.Text.Contains(',') && stepTextBox.Text.Length != 0)

))

{

e.Handled = true;

}

}

}

}

Листинг GradientDescent3DWindow.xaml

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Input;

using MathWorks.MATLAB.NET.Arrays;

namespace gradientDescent

{

/// <summary>

/// Логика взаимодействия для GradientDescent3DWindow.xaml

/// </summary>

public partial class GradientDescent3DWindow : Window

{

GradientDescent3D gradientDescent3D = new GradientDescent3D();

public GradientDescent3DWindow()

{

InitializeComponent();

}

private void textboxes\_previewTextInput(object sender, TextCompositionEventArgs e)

{

if (!(Char.IsDigit(e.Text, 0) || (e.Text == ",") && (!AccuracyTextBox.Text.Contains(',') && AccuracyTextBox.Text.Length != 0)))

{

e.Handled = true;

}

}

private void ExecuteButton\_Click(object sender, RoutedEventArgs e)

{

string function = InitialFunctionTextBox.Text;

try

{

double eps = Convert.ToDouble(AccuracyTextBox.Text);

double xPoint = Convert.ToDouble(initialXTextBox.Text);

double yPoint = Convert.ToDouble(initialYTextBox.Text);

double zPoint = Convert.ToDouble(initialZTextBox.Text);

Point3D point = new Point3D(xPoint, yPoint, zPoint);

gradientDescent3D.GradientDescent3DMethod(function, point, eps);

string func = "@(x,y) " + function;

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

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

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

for (int i = 0; i < gradientDescent3D.GradientDescent3DPointsList.Count(); i++)

{

x.Add(gradientDescent3D.GradientDescent3DPointsList[i].X);

y.Add(gradientDescent3D.GradientDescent3DPointsList[i].Y);

z.Add(gradientDescent3D.GradientDescent3DPointsList[i].Z);

}

Gradient3D.Gradient3D gradient3D = new Gradient3D.Gradient3D();

gradient3D.GradientDescent(func, (MWNumericArray)x.ToArray(), (MWNumericArray)y.ToArray(), (MWNumericArray)z.ToArray());

}

catch (Exception ex)

{

MessageBox.Show("Ошибка!" + ex.Message);

}

}

}

}

Листинг GradientDescentMenu.xaml

using System.Windows;

namespace gradientDescent

{

/// <summary>

/// Логика взаимодействия для GradientDescentMenu.xaml

/// </summary>

public partial class GradientDescentMenu : Window

{

public GradientDescentMenu()

{

InitializeComponent();

}

private void gradientDescent3DButton\_Click(object sender, RoutedEventArgs e)

{

GradientDescent3DWindow gradient3DWindow = new GradientDescent3DWindow();

gradient3DWindow.Show();

}

private void gradientDescentButton\_Click(object sender, RoutedEventArgs e)

{

MainWindow gradientDescent = new MainWindow();

gradientDescent.Show();

}

private void aboutButton\_Click(object sender, RoutedEventArgs e)

{

TutorialWindow tutorialWindow = new TutorialWindow();

tutorialWindow.Show();

}

}

}

MainWindow XAML

<Window x:Class="gradientDescent.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:gradientDescent"

mc:Ignorable="d"

Title="Градиентный спуск" Height="500" Width="900" WindowStartupLocation="CenterScreen" ResizeMode="NoResize" WindowStyle="ToolWindow">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="4\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0" Background="#FF8A0101">

<Grid.RowDefinitions>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Label VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="14" FontFamily="Arial" Foreground="White">введите функцию</Label>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="3\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Label HorizontalAlignment="Center" FontSize="14" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">f(x) =</Label>

</Grid>

<Grid Grid.Column="1">

<TextBox Name="InitialFunctionTextBox" Height="25" Margin="5 0 5 0" FontFamily="Arial" FontSize="14"></TextBox>

</Grid>

</Grid>

<Grid Grid.Row="2">

<Label HorizontalAlignment="Center" FontSize="14" FontFamily="Arial" Foreground="White">введите отрезок</Label>

</Grid>

<Grid Grid.Row="3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"></ColumnDefinition>

<ColumnDefinition Width="1.5\*"></ColumnDefinition>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="1.5\*"></ColumnDefinition>

<ColumnDefinition Width="2\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Label HorizontalAlignment="Right" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">[</Label>

</Grid>

<Grid Grid.Column="1">

<TextBox Name="startXTextBox" Width="25" Height="25"></TextBox>

</Grid>

<Grid Grid.Column="2">

<Label HorizontalAlignment="Center" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">;</Label>

</Grid>

<Grid Grid.Column="3">

<TextBox Name="endXTextBox" Width="25" Height="25"></TextBox>

</Grid>

<Grid Grid.Column="4">

<Label HorizontalAlignment="Left" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">]</Label>

</Grid>

</Grid>

<Grid Grid.Row="4">

<Label HorizontalAlignment="Center" FontSize="14" FontFamily="Arial" Foreground="White">введите шаг сетки</Label>

</Grid>

<Grid Grid.Row="5">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="3\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Label HorizontalAlignment="Center" FontSize="14" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">h=</Label>

</Grid>

<Grid Grid.Column="1">

<TextBox Name="stepTextBox" Height="25" Margin="5 0 5 0" FontFamily="Arial" FontSize="14" PreviewTextInput="textboxes\_previewTextInput"/>

</Grid>

</Grid>

<Grid Grid.Row="6">

<Button Name="BuildFunctionButton" FontFamily="Arial" FontSize="14" Foreground="#FF8A0101" Background="White" Click="BuildFunctionButton\_Click">построить</Button>

</Grid>

<Grid Grid.Row="7">

<Label HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="14" FontFamily="Arial" Foreground="White">введите точку</Label>

</Grid>

<Grid Grid.Row="8">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"></ColumnDefinition>

<ColumnDefinition Width="1.5\*"></ColumnDefinition>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="1.5\*"></ColumnDefinition>

<ColumnDefinition Width="2\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Label HorizontalAlignment="Right" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">(</Label>

</Grid>

<Grid Grid.Column="1">

<TextBox Name="InitialPointX" Width="25" Height="25"></TextBox>

</Grid>

<Grid Grid.Column="2">

<Label HorizontalAlignment="Center" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">;</Label>

</Grid>

<Grid Grid.Column="3">

<TextBox Name="InitialPointY" Width="25" Height="25"></TextBox>

</Grid>

<Grid Grid.Column="4">

<Label HorizontalAlignment="Left" FontSize="18" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">)</Label>

</Grid>

</Grid>

<Grid Grid.Row="9">

<Label HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="14" FontFamily="Arial" Foreground="White">введите точность</Label>

</Grid>

<Grid Grid.Row="10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"></ColumnDefinition>

<ColumnDefinition Width="3\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Label HorizontalAlignment="Right" FontSize="14" FontFamily="Arial" VerticalContentAlignment="Center" Foreground="White">ε =</Label>

</Grid>

<Grid Grid.Column="1">

<TextBox Name="AccuracyTextBox" PreviewTextInput="textboxes\_previewTextInput" Height="25" Margin="5 0 5 0" FontFamily="Arial" FontSize="14"></TextBox>

</Grid>

</Grid>

<Grid Grid.Row="11">

<Button Name="ClearAllButton" FontFamily="Arial" FontSize="14" Foreground="#FF8A0101" Background="White" Click="ClearAllButton\_Click">очистить</Button>

</Grid>

<Grid Grid.Row="12">

<Button Name="ExecuteButton" Click="ExecuteButton\_Click" FontFamily="Arial" FontSize="14" Foreground="#FF8A0101" Background="White">найти</Button>

</Grid>

</Grid>

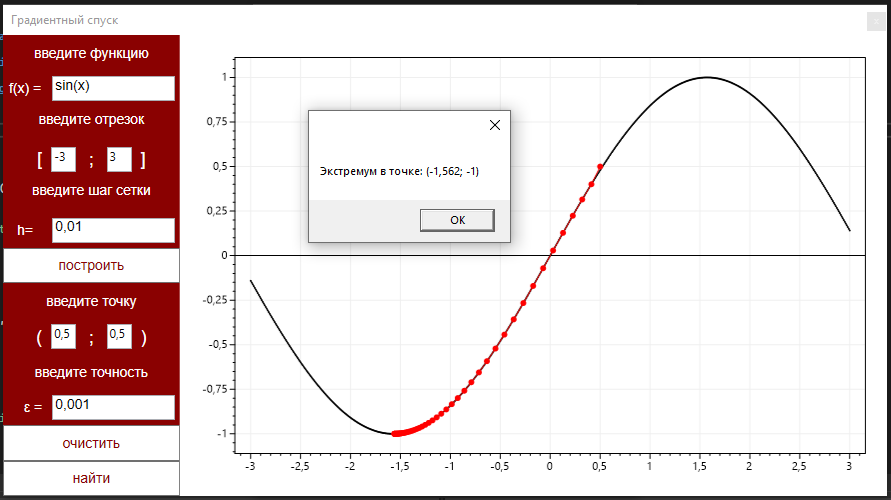
<Grid Grid.Column="1">

<WpfPlot Name="Plot"></WpfPlot>

</Grid>

</Grid>

</Window>



### Приложение 2. Интерфейс программы

