

**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

**КУРСОВАЯ РАБОТА**

по дисциплине «Математическое программирование»

Разработка программы, реализующей методы прямого поиска:

метод Хука-Дживса, метод Розенброка

Студент

Мурзахожин А.Д.

Группа АИ-18

Руководитель

Качановский Ю.П.

Липецк 2020 г.

## ЗАДАНИЕ КАФЕДРЫ

Студенту: Мурзахожину А.Д. группы АИ-18.

Тема: Реализация методов прямого поиска: метод Хука-Дживса, метод Розенброка.

### Требования к программе:

В программе необходимо предусмотреть ввод данных с клавиатуры. Помимо функций решения задачи темы курсовой работы, программа должна в себя включать: систему меню, строго определяющую диалог пользователя с программой; систему помощи, позволяющую пользователю получить справку о задаче на текущем этапе ее решения, текущем состоянии программы, вариантах продолжения работы; обработчик ошибок: пользователя (выдача рекомендаций при неверных действиях); выполнения (предупреждение деления на ноль, потери точности и т.п. с выдачей диагностического сообщения).

### Содержание расчетно-пояснительной записки:

1. Титульный лист.
2. Задание кафедры.
3. Аннотация.
4. Оглавление.
5. Введение.
6. Описание задачи.
7. Этапы кодирования.
8. Результаты отладки программы.
9. Тестирование.
10. Описание программы.
11. Описание применения.
12. Руководство программиста.
13. Заключение.
14. Приложение: текст программы.

Дата выдачи задания: \_\_\_\_\_. Срок сдачи курсовой работы: \_\_\_\_\_.

Р

У

а

к

д

н

в

и

о

д

п

## Аннотация

С. 37. Ил.10. Табл.1. Литература 3. Прил.2;

В ходе выполнения курсовой работы была реализована программа, решающая поставленную задачу, на языке С#. Графический интерфейс приложения реализован с помощью технологии .NET Framework и подсистемы проектирования графических интерфейсов WPF. Представлено описание и сопутствующие сведения.

# Оглавление

О

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

В

[illegible]

## Введение

В данной работе необходимо реализовать методы прямого поиска: Хука-Дживса, Розенброка.

Основной алгоритм для методов:

Шаг 1: Проверка соблюдения условия останова.

Если условие останова выполняется, то вычисления прекратить и взять в качестве искомого решения, в противном случае перейти на шаг 2

Шаг 2: Расчет направления поиска. Вычислить не нулевой вектор  $S^{(k)}$  - направление поиска.

Шаг 3: Расчет длины шага. Вычислить положительное число  $\lambda^{(k)}$ , обеспечивающее выполнение неравенства:

$$f(x^{(k)} + \lambda^{(k)} S^{(k)}) < f(x^{(k)})$$

Шаг 4: Пересчет оценки решения.

$$x^{(k+1)} \leftarrow x^{(k)} + \lambda^{(k)} S^{(k)}$$

$$k \leftarrow k + 1$$

Переход на шаг 1.

Рассмотрим каждый из методов.

1. Метод Хука-Дживса:

Данный алгоритм делится на два этапа. Исследующий поиск вокруг базисной точки и поиск по образцу.

**Исследующий поиск.** Делаем пробный шаг по переменной  $x_1$ , т.е.

о  
п  
р  
е  
д  
е  
л  
я  
е  
м

функции меньше исходного. Из вновь полученной точки делаем пробные шаги по оставшимся координатам, используя тот же самый алгоритм.

Если в процессе исследующего поиска не удастся сделать ни одного удачного пробного шага, то  $\Delta \bar{x}$  необходимо скорректировать (уменьшить). После чего вновь переходим к исследующему поиску.

Если в процессе исследующего поиска сделан хотя бы один удачный пробный шаг, то переходим к поиску по образцу.

**Поиск по образцу.** После исследующего поиска мы получаем точку  $\bar{x}^{01}$ .

Н

а

п В поиске по "образцу" величина шага по каждой переменной пропорциональна величине шага на этапе исследующего поиска. Если удастся сделать удачный шаг в поиске по "образцу", то в результате находим новое

в

р Дальше если  $\lambda > \varepsilon$  из новой точки возвращаемся к исследующему поиску.

я 2. Метод Розенброка

б Суть метода состоит во вращении системы координат в соответствии с изменением скорости убывания целевой функции. Новые направления координатных осей определяются таким образом, чтобы одна из них соответствовала направлению наиболее быстрого убывания целевой функции, остальные находятся из условия ортогональности.

р Из начальной точки  $x[0]$  осуществляют спуск в точку  $x[1]$  по направлениям, параллельным координатным осям. На следующей итерации одна из осей должна проходить в направлении  $y1 = x[1] - x[0]$ , а другая - в направлении  $y2 = x[0] - x[1]$ . Спуск в точку  $x[2]$  по этим направлениям приводит к построению нового базиса. Спуск вдоль этих осей приводит в точку  $x[2]$ , что дает возможность построить новый вектор  $x[2] - x[1]$  и на его базе новую систему направлений поиска. В общем случае данный метод

м

и

н

и

эффективен при минимизации овражных функций, так как результирующее направление поиска стремится расположиться вдоль оси оврага.



## 1. Описание задачи

Необходимо разработать программу, реализующую методы прямого поиска.

Начнём работу с того, что создадим отдельные функции, которые будут описывать алгоритмы прямого поиска. Необходимо разработать процедуру вывода результатов на экран.

Следующим этапом соединим написанные алгоритмы и функции в единую программу. Так же сделаем пользовательское представление, оно будет заключаться в создании главного окна. Оно будет отвечать за ввод изначальной функции, ограничений, начальных данных и настроек для задания метода прямого поиска и вывода результата.

## 2. Этапы кодирования

### 2.1. Кодирование функций, используемых в программе

`public double comp(double[] xnew)` – функция подсчета функции в точке.

`public String MethodHookeJeeves(double dx, double a, double b, double e, int count)` – метод Хука-Дживса

`public double[] projection(double[] a, double[] b, int n)` – функция проекции одного вектора на другой.

`public String MethodRosenbrock(double dx, double a, double b, double e, int count)` – функция Розенброка.

`private void Form1_Load(object sender, EventArgs e)` – функция загрузки формы.

`private void button1_Click(object sender, EventArgs e)` – функция обработки кнопки размерности задачи.

`private void методToolStripMenuItem_Click(object sender, EventArgs e)` – функция обработки вызова метода Хука-Дживса.

`private void методРозенброкаToolStripMenuItem_Click(object sender, EventArgs e)` - функция обработки вызова метода Розенброка.

### 3. Результат отладки программы

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

При разработке программы использовались следующие методы отладки:

- использование точек останова, проход фрагмента программы «по проблемным местам» и просмотр значений переменных в процессе выполнения.

Все это позволяет выполнить используемая среда разработки (в нашем случае Visual Studio);

- контрольный вывод результатов выполнения на экран в виде информационных сообщений.

На этапе отладки были устранены и исправлены различные ошибки в программном коде.

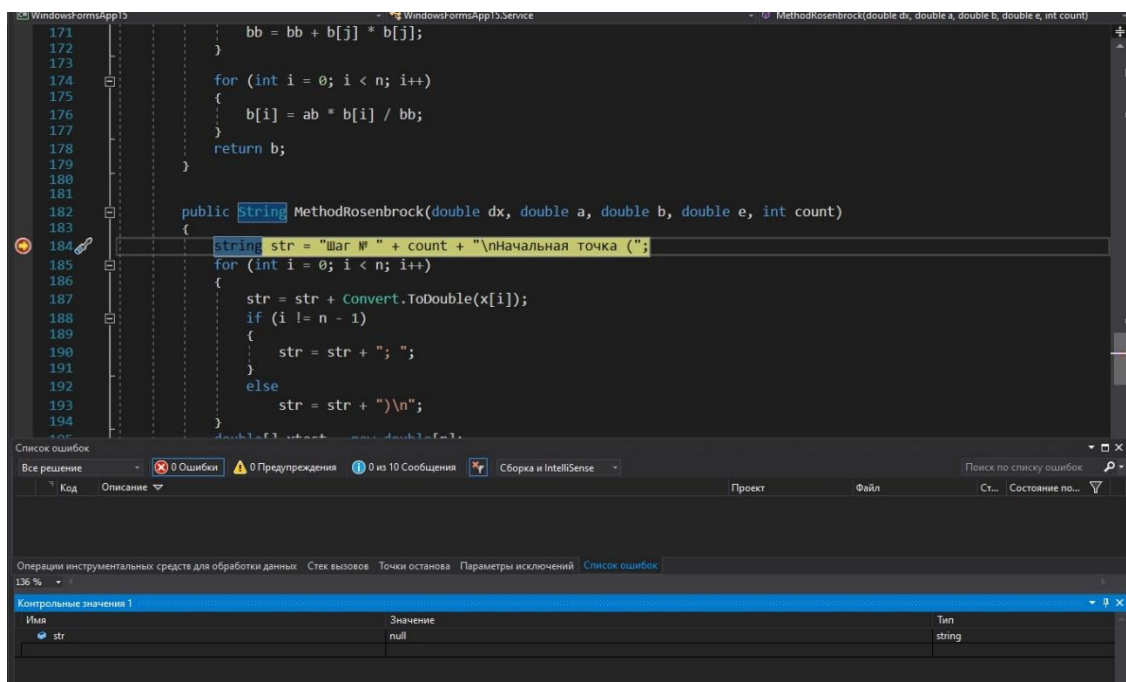


Рисунок 1 – Результат отладки

#### 4. Тестирование

На этапе тестирования мы проверим программу на реальных задачах. Были протестированы следующие функции программы:

- вычисление минимума функции различными методами прямого поиска;
- преобразование математических выражений;
- вывод результата на экран.

Программа корректно отработала и вернула правильный результат.

The screenshot shows a window titled 'Form1' with a menu bar containing 'Файл', 'Расчет', and 'Помощь'. The main area contains several input fields and a large empty rectangular box on the right. The inputs are: 'Размерность задачи' (2), 'Начальное приращение  $\Delta x$ ' (1), 'Коэффициент уменьшения шага: a' (2), 'Коэффициент усиления: b' (2), 'Точность  $\epsilon$ ' (0,01), 'Координаты начальной точки' with  $x_1 = -4$ ,  $x_2 = 4$ ,  $x_3 =$ ,  $x_4 =$ ,  $x_5 =$ ,  $x_6 =$ , and 'Уравнение:  $F(\bar{x}) = 8*x_1*x_1+4*x_1*x_2+5*x_2*x_2$ '.

Рисунок 2 – Поиск минимума для заданной функции

This screenshot shows the same program interface as Figure 2, but with a text box on the right displaying the search process. The text box contains the following information:

```
Шаг № 1
Начальная точка (-4; 4) Шаг dx = 1
Новая точка (-3; 3)
Поиск по образцу (-1; 1)
Шаг № 2
Начальная точка (-1; 1) Шаг dx = 1
Новая точка (0; 0)
Поиск по образцу неудачен
Шаг № 3
Начальная точка (0; 0) Шаг dx = 0,5
Новая точка (0; 0)
Шаг № 4
Начальная точка (0; 0) Шаг dx = 0,25
Новая точка (0; 0)
Шаг № 5
Начальная точка (0; 0) Шаг dx = 0,125
Новая точка (0; 0)
Шаг № 6
Начальная точка (0; 0) Шаг dx = 0,0625
Новая точка (0; 0)
Шаг № 7
Начальная точка (0; 0) Шаг dx = 0,03125
Новая точка (0; 0)
Шаг № 8
Начальная точка (0; 0) Шаг dx = 0,015625
Новая точка (0; 0)
Шаг № 9
Начальная точка (0; 0) Шаг dx = 0,0078125
Новая точка (0; 0)
```

Рисунок 3 – Результат поиска минимума методом Хука-Дживса

Файл

Расчет

Помощь

Размерность задачи

2

Начальное приращение  $\Delta x$  =

1

Коэффициент уменьшения шага: a =

0,5

Коэффициент усиления: b =

2

Точность  $\varepsilon$  =

0,01

Координаты начальной точки

X1 = -4

X2 = 4

X3 =

X4 =

X5 =

X6 =

Уравнение:  $F(\bar{x}) =$

$8 \cdot x_1 \cdot x_1 + 4 \cdot x_1 \cdot x_2 + 5 \cdot x_2 \cdot x_2$

Шаг № 18

Начальная точка (0,0577643237598485; -0,044438753641966)

Новая точка (0,122589475570029; 0,0219586215904565)

Шаг № 19

Начальная точка (0,122589475570029; 0,0219586215904565)

Новая точка (0,0751166624517767; 0,00346355983630877)

Шаг № 20

Начальная точка (0,0751166624517767; 0,00346355983630877)

Новая точка (-0,0267871400600636; -0,0156664339598623)

Шаг № 21

Начальная точка (-0,0267871400600636; -0,0156664339598623)

Новая точка (0,0206892187040433; 0,0124124735793308)

Шаг № 22

Начальная точка (0,0206892187040433; 0,0124124735793308)

Новая точка (-0,0120617599657095; 0,0136120347840326)

Шаг № 23

Начальная точка (-0,0120617599657095; 0,0136120347840326)

Новая точка (0,00485906223454627; 0,0279011860443924)

Шаг № 24

Начальная точка (0,00485906223454627; 0,0279011860443924)

Новая точка (0,0248480253416645; -0,012932360602897)

Шаг № 25

Начальная точка (0,0248480253416645; -0,012932360602897)

Новая точка (0,0124419737350685; 0,00630389354525783)

Шаг № 26

Начальная точка (0,0124419737350685; 0,00630389354525783)

Новая точка (0,0164041327738106; -0,00475943738766193)

Шаг № 27

Начальная точка (0,0164041327738106; -0,00475943738766193)

Новая точка (0,0119781882556313; -0,000103362103965923)

Рисунок 4 – Результат поиска минимума методом Розенброка

## 5. Описание программы

### 5.1 Общие сведения

#### 5.1.1 Обозначение и наименование программы

Название программного продукта: «Методы прямого поиска».

#### 5.1.2 Программное обеспечение, необходимое для функционирования программы

Для функционирования программы необходим установленный Visual Studio 2017 и выше.

#### 5.1.3 Языки программирования, на которых написана программа

Программа написана на языке программирования C#. В качестве среды разработки использована среда Visual Studio 2017.

#### 5.1.4 Функциональное назначение

Программа предназначена для выполнения задач, связанных с нахождением оптимума.

### 5.2 Описание логической структуры

#### 5.2.1 Алгоритм программы

Ход действия программы:

- пользователь заполняет поля с начальными данными, а именно выбирает количество переменных (2 или 3), задаёт начальное приращение, коэффициент уменьшения шага, коэффициент усиления, точность и начальную точку, вводит исходную функцию;
- происходит преобразование математического выражения;
- выполняется метод прямого поиска;
- вывод результата.

### 5.3 Используемые технические средства

#### 5.3.1 Для работы программы необходим ПК с установленной операционной системы Windows 7 и старше.

5.3.2 Технические характеристики ПК должны быть достаточными для запуска указанной операционной системы. Программа тестировалась на процессоре Intel Core i5, 4 ядра. Операционная система – Windows 10.

## 5.4 Вызов и загрузка

### 5.4.1 Способ вызова программы с соответствующего носителя данных

Для запуска программы необходимо запустить исполняемый файл

## 5.5 Входные точки в программу

Точка входа в программу зависит от метода, который выбрал пользователь: `MethodRosenbrock()` или `MethodHookeJeeves()`, в классе `class`

## 5.6 Входные данные

В программе предусмотрен способ передачи входных данных – ввод данных с клавиатуры.

Типы входных данных:

- число переменных (обязательно);
- начальное приращение (обязательно);
- коэффициент уменьшения шага (обязательно);
- коэффициент усиления (обязательно);
- точность прямого поиска (обязательно);
- начальная точка (обязательна);
- минимизируемая функция (обязательна).

## 5.7 Выходные данные

В программе предусмотрены выходные данные:

- отчет о проведенных вычислениях, а именно результат работы программы;

## 6. Описание применения

### 6.1 Назначение программы

Программа предназначена для нахождения точки минимума с использованием различных методов прямого поиска

### 6.2 Возможности программы

Основные функции программы:

- отображение меню и ожидание действий пользователя;

- ввод данных с клавиатуры для начала исследования;
- поиск точек оптимума заданным алгоритмом;
- вывод результата;

### 6.3 Ограничения, накладываемые на область применения

Возникают трудности при работе со сложными функциями, например, функции Розенброка, у которых линии уровня вытянуты. Для исследования таких функции следует попробовать различные значения параметров алгоритма.

### 6.4 Условия применения

#### 6.4.1 Используемые технические средства

6.4.1.1 Для работы программы необходим ПК с установленной операционной системой Windows 7 и старше.

.2 Технические характеристики ПК должны быть достаточными для запуска указанной операционной системы.

6.4.1.3 Программное обеспечение, необходимое для функционирования программы: для функционирования программы необходим установленный

### 6.5 Описание задачи

Данная программа предназначена для нахождения точки минимума с использованием различных методов прямого поиска.

Программа обладает следующими функциями:

- отображение меню и ожидание действий пользователя;
- ввод данных для начала исследования;
- поиск точек оптимума заданным алгоритмом;
- вывод результата;

### 6.6 Входные данные

В программе предусмотрен способ передачи входных данных — ввод данных с клавиатуры.

Типы входных данных:

- число переменных (обязательно);



- начальное приращение (обязательно);
- коэффициент уменьшения шага (обязательно);
- коэффициент усиления (обязательно);
- точность прямого поиска (обязательно);
- начальная точка (обязательна);
- минимизируемая функция (обязательна).

## 6.7 Выходные данные

В программе предусмотрены выходные данные:

- отчет о проведенных вычислениях в элементе RichBox;

## 7. Руководство программиста

### 7.1 Назначение и условие применения программы

#### 7.1.1 Назначение программы

Программа предназначена для нахождения точки минимума с использованием различных методов прямого поиска.

#### 7.1.2 Возможности программы

Основные функции программы:

- отображение меню и ожидание действий пользователя;
- ввод данных для начала исследования;
- поиск точек оптимума заданным алгоритмом;
- вывод результата;

#### 7.1.3 Используемые технические средства

7.1.3.1 Аппаратные требования: клавиатура, мышь. Системные требования: Windows 7 и выше. Для работы требуется 2 Мб дискового пространства, 31 Мб ОЗУ и установленный Visual Studio 2017 или выше.

7.1.3.2 Технические характеристики ПК должны удовлетворять следующим минимальным характеристикам: 1 Гб оперативной памяти, 2 ядерный процессор с частотой от 1 ГГц, наличие монитора, клавиатуры, мыши, жесткого диска.

7.2 Программное обеспечение, необходимое для функционирования программы.

Для функционирования программы требуется установленный Visual Studio 2017 и выше.

### 7.3 Характеристика программы

Программа состоит из одного исполняемого файла WindowsFormApp15.exe, содержит графический интерфейс с одним окном, предназначена для операционных систем семейства Windows. В программе предусмотрен ввод данных с клавиатуры.

7.3.2 Все взаимодействия с пользователем происходят через графический интерфейс программы.

Обращение к программе

7.4.1 Способ вызова программы с соответствующего носителя данных

Для запуска программы необходимо запустить исполняемый файл

### 7.5. Входные точки в программу

Точка входа в программу зависит от метода, который выбрал пользователь: MethodRosenbrock() или MethodHookeJeeves(), в классе class

### 7.6. Входные данные

В программе предусмотрены два способа передачи входных данных – ввод данных с клавиатуры.

- число переменных (обязательно);
- начальное приращение (обязательно);
- коэффициент уменьшения шага (обязательно);
- коэффициент усиления (обязательно);
- точность прямого поиска (обязательно);
- начальная точка (обязательна);
- минимизируемая функция (обязательна).

## 7.7. Выходные данные

В программе предусмотрены выходные данные:

- отчет о проведенных вычислениях в элементе RichBox;

## 8. Руководство пользователя

Работа с программой начинается с ввода данных для начала исследования.

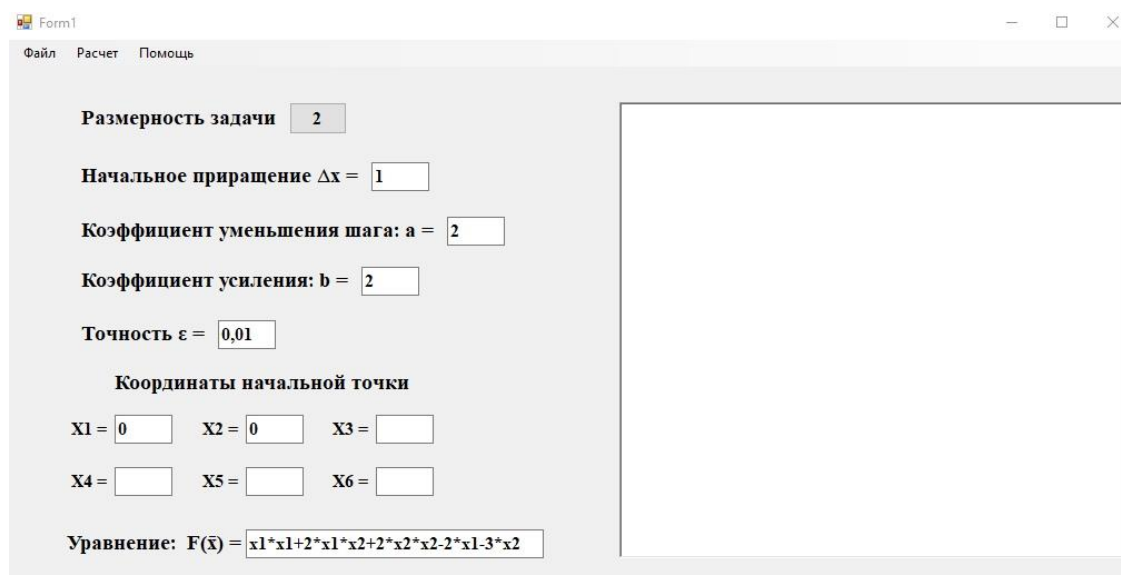
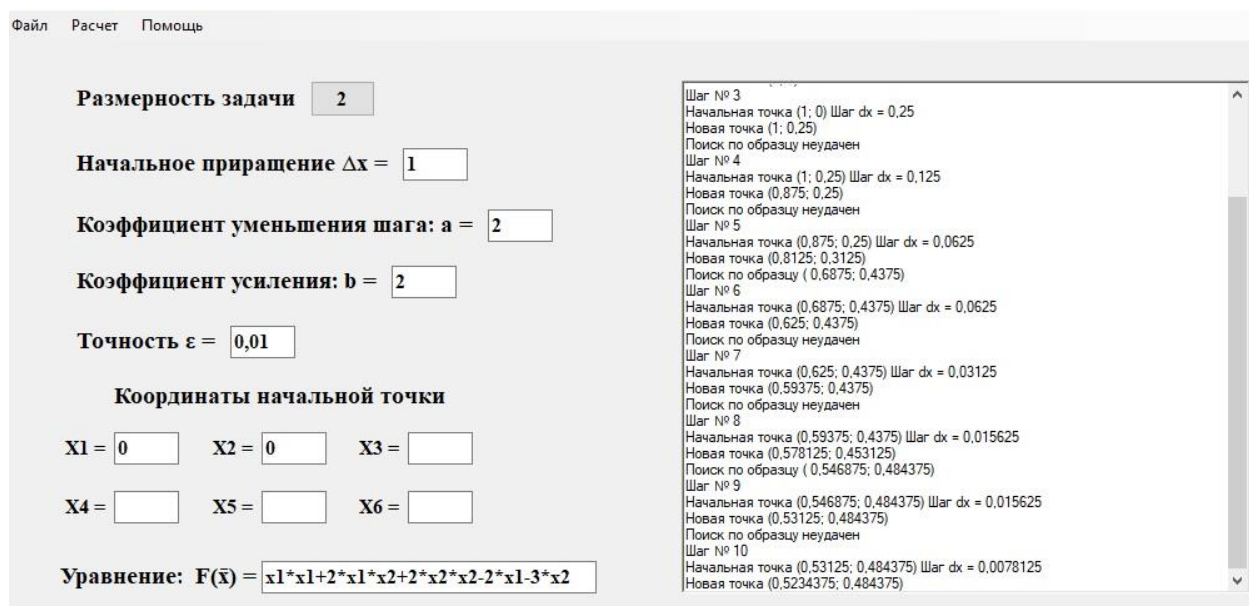


Рисунок 5 – Главное окно программы

Для получения результата нужно нажать на кнопку “Расчет” и выбрать метод для расчета. После нахождения условного минимума в боковом поле появится отчет, содержащий начальные условия, все точки, полученные во время поиска и результат.



Шар № 3  
Начальная точка (1; 0) Шар dx = 0,25  
Новая точка (1, 0,25)  
Поиск по образцу неудачен  
Шар № 4  
Начальная точка (1; 0,25) Шар dx = 0,125  
Новая точка (0,875; 0,25)  
Поиск по образцу неудачен  
Шар № 5  
Начальная точка (0,875; 0,25) Шар dx = 0,0625  
Новая точка (0,8125; 0,3125)  
Поиск по образцу ( 0,6875; 0,4375)  
Шар № 6  
Начальная точка (0,6875; 0,4375) Шар dx = 0,0625  
Новая точка (0,625; 0,4375)  
Поиск по образцу неудачен  
Шар № 7  
Начальная точка (0,625; 0,4375) Шар dx = 0,03125  
Новая точка (0,59375; 0,4375)  
Поиск по образцу неудачен  
Шар № 8  
Начальная точка (0,59375; 0,4375) Шар dx = 0,015625  
Новая точка (0,578125; 0,453125)  
Поиск по образцу ( 0,546875; 0,484375)  
Шар № 9  
Начальная точка (0,546875; 0,484375) Шар dx = 0,015625  
Новая точка (0,53125; 0,484375)  
Поиск по образцу неудачен  
Шар № 10  
Начальная точка (0,53125; 0,484375) Шар dx = 0,0078125  
Новая точка (0,5234375; 0,484375)

Рисунок 6 – Окно с отчётом результата метода Хука-Дживса

Файл

Расчет

Помощь

Размерность задачи

2

Начальное приращение  $\Delta x =$

1

Коэффициент уменьшения шага:  $a =$

0.5

Коэффициент усиления:  $b =$

2

Точность  $\varepsilon =$

0,01

Координаты начальной точки

X1 =

0

X2 =

0

X3 =

X4 =

X5 =

X6 =

Уравнение:  $F(\bar{x}) =$

$x1*x1+2*x1*x2+2*x2*x2-2*x1-3*x2$

Шаг № 10

Начальная точка (0,483261716600365; 0,550769854716441)

Новая точка (0,510933923736137; 0,48271378083299)

Шаг № 11

Начальная точка (0,510933923736137; 0,48271378083299)

Новая точка (0,52172495655448; 0,526755431260673)

Шаг № 12

Начальная точка (0,52172495655448; 0,526755431260673)

Новая точка (0,432658130873477; 0,525235650751972)

Шаг № 13

Начальная точка (0,432658130873477; 0,525235650751972)

Новая точка (0,477285061138743; 0,520514946853468)

Шаг № 14

Начальная точка (0,477285061138743; 0,520514946853468)

Новая точка (0,454399300764005; 0,517465135316583)

Шаг № 15

Начальная точка (0,454399300764005; 0,517465135316583)

Новая точка (0,466540586161939; 0,513749214201082)

Шаг № 16

Начальная точка (0,466540586161939; 0,513749214201082)

Новая точка (0,489416362513506; 0,501720855436605)

Шаг № 17

Начальная точка (0,489416362513506; 0,501720855436605)

Новая точка (0,486770952495117; 0,52445674788402)

Шаг № 18

Начальная точка (0,486770952495117; 0,52445674788402)

Новая точка (0,485444987100653; 0,512780618498492)

Шаг № 19

Начальная точка (0,485444987100653; 0,512780618498492)

Новая точка (0,483527574968577; 0,51891171816327)

Рисунок 7 – Окно с отчётом результата метода Розенброка

## 9. Сравнение работы программы с программами из ЛБ№2

Для сравнения работы программ возьмем уравнение:

$$f(x_1, x_2) = x_1^2 + 2x_1x_2 + 2x_2^2 - 2x_1 - 3x_2$$

Начальная точка (0;0)

Form1

Файл Расчет Помощь

Размерность задачи: 2

Начальное приращение  $\Delta x =$  1

Коэффициент уменьшения шага: a = 0,5

Коэффициент усиления: b = 2

Точность  $\varepsilon =$  0,01

Координаты начальной точки

X1 = 0 X2 = 0 X3 =

X4 = X5 = X6 =

Уравнение:  $F(\bar{x}) = x1*x1+2*x1*x2+2*x2*x2-2*x1-3*x2$

Шаг № 10  
Начальная точка (0,483261716600365; 0,550769854716441)  
Новая точка (0,510933923736137; 0,48271378083299)  
Шаг № 11  
Начальная точка (0,510933923736137; 0,48271378083299)  
Новая точка (0,52172495655448; 0,526755431260673)  
Шаг № 12  
Начальная точка (0,52172495655448; 0,526755431260673)  
Новая точка (0,432658130873477; 0,525235650751972)  
Шаг № 13  
Начальная точка (0,432658130873477; 0,525235650751972)  
Новая точка (0,477285061138743; 0,520514946853468)  
Шаг № 14  
Начальная точка (0,477285061138743; 0,520514946853468)  
Новая точка (0,454399300764005; 0,517465135316583)  
Шаг № 15  
Начальная точка (0,454399300764005; 0,517465135316583)  
Новая точка (0,466540586161939; 0,513749214201082)  
Шаг № 16  
Начальная точка (0,466540586161939; 0,513749214201082)  
Новая точка (0,489416362513506; 0,501720855436605)  
Шаг № 17  
Начальная точка (0,489416362513506; 0,501720855436605)  
Новая точка (0,486770952495117; 0,52445674788402)  
Шаг № 18  
Начальная точка (0,486770952495117; 0,52445674788402)  
Новая точка (0,485444987100653; 0,512780618498492)  
Шаг № 19  
Начальная точка (0,485444987100653; 0,512780618498492)  
Новая точка (0,483527574968577; 0,51891171816327)

Рисунок 8 – Введенные в мою программу данные для проверки

Методы оптимизации нулевого порядка

Данные Справка Выход

Оптимизация Исходные данные

Целевая функция (переменные: x1, x2 и т.д.)

$x1*x1+2*x1*x2+2*x2*x2-2*x1-3*x2$

Параметры задачи

Размерность: 2 Точность: 0,01 Шаг: 1

Начальная точка

X1	X2
0	0

Рисунок 9 – Введенные в программу 1 из ЛБ№2 данные

Рисунок 10 – Введенные в программу 2 из ЛБ№2 данные

Таблица 1 – Сравнение результатов вычислений программ

Метод	Программа 1 из ЛБ№2	Программа 2 из ЛБ№2	Разработанная программа
Хука-Дживса	<p>/-----/</p> <p>-----/</p> <p>Оптимум найден на итерации: 22</p> <p>Точка оптимума X_opt:</p> <p>x1_opt= -0,0001220703125</p> <p>x2_opt= 0,75</p> <p>Значение функции в X_opt:</p> <p>F(X_opt)= -1,12493894994259</p> <p>Точность: 6,103515625E-5</p> <p>Длина шага: 6,103515625E-5</p>	<p>Оптимум найден на итерации: 21</p> <p>Точка оптимума:</p> <p>Длина шага:</p> <p>Точность: 6,103515625E-5</p> <p>Значение функции: -</p>	<p>Шаг № 10</p> <p>Начальная точка (0,53125; 0,484375)</p> <p>Шаг dx = 0,0078125</p> <p>Новая точка (0,5234375; 0,484375)</p>
Розенброка	<p>/-----/</p> <p>-----/</p> <p>Оптимум найден на итерации: 10</p> <p>Точка оптимума X_opt:</p> <p>x1_opt= 0,500047824011675</p> <p>x2_opt= 0,49998100363273</p> <p>Значение функции в X_opt:</p> <p>F(X_opt)= -1,2499999988081</p> <p>Точность: 1,19033050083317E-5</p>	<p>Оптимум найден на итерации 10</p> <p>Значение функции в точке оптимума: -</p> <p>Точность, с которой был вычислен оптимум</p> <p>Точка оптимума:</p>	<p>Шаг № 19</p> <p>Начальная точка (0,485444987100653; 0,512780618498492)</p> <p>Новая точка (0,483527574968577; 0,51891171816327)</p>

При сравнении программ было выявлено, что программы из ЛБ№2 неверно находят точку минимума функции для метода Хука-Дживса. Программы не дают возможности настройки коэффициентов а и b. Метод Розенброка в этих программах находит точку оптимума за меньшее количество итераций и точнее. На это могут влиять несколько причин:

- 1) Коэффициенты  $a$  и  $b$  не известны в программах из ЛБ№2;
- 2) Норма сходимости в этих программах определяется как норма разницы значения функции в новой и старой точке; в разработанной мною программе норма определяется как норма вектора, проведенного из старой точки в новую.

## Заключение

В результате выполнения данного курсового проекта была разработана программа “Методы прямого поиска”. При помощи данной программы можно найти точку оптимума методом Хука-Дживса и методом Розенброка.

Было проведено сравнение разработанной программы с программами из ЛБ№2, что показало более гибкую настройку методов в разработанной программе.

Также было проведено исследование полученного программного продукта. Достоинства:

- понятный интерфейс;
- корректная работа.

Для достижения поставленной цели был изучен теоретический материал по данному вопросу, разработан алгоритм решения задачи, выполнено кодирование на языке C# проведена отладка и тестирование программы.

На разработанную программу была составлена документация в соответствии с ЕСПД: «Описание программы», «Описание применения» и «Руководство пользователя».



Приложение А  
(Код программы)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace WindowsFormsApp15
{
    public class Service
    {
        public String F;
        int n;
        double[] x;
        double[][] p;
        public Service(double[] xnew, int num)
        {
            this.n = num;
            this.x = new double[n];
            this.p = new double[n][];
            for (int i = 0; i < n; i++)
            {
                x[i] = xnew[i];
                this.p[i] = new double[n];
                for (int j = 0; j < n; j++)
                {
                    p[i][j] = 0;
                    if (i == j)
                    {
                        p[i][j] = 1;
                    }
                }
            }
        }
    }
}
```

```

        }
    }
}
//Подсчет выражения
public double comp(double[] xnew)
{
    String sF = F;
    for (int i = 0; i < n; i++)
    {
        sF = sF.Replace("x" + ((i + 1).ToString()), Convert.ToString(xnew[i]));
    }
    sF = sF.Replace("sqrt", "Math.Sqrt");
    sF = sF.Replace("sin", "Math.Sin");
    sF = sF.Replace("cos", "Math.Cos");
    sF = sF.Replace("log", "Math.Log");
    sF = sF.Replace("exp", "Math.Exp");
    sF = sF.Replace("pow", "Math.Pow");
    sF = sF.Replace(",", ".");
    sF = sF.Replace("-", "-");
    sF = sF.Replace("--", "+");
    sF = sF.Replace("--", "+");
    sF = sF.Replace(",", ".");
    sF = sF.Replace("*-", "*(-1)*");
    Calculator cs = new Calculator();
    Double res = cs.Calc(sF);
    return res;
}

public String MethodHookeJeeves(double dx, double a, double b, double e,
int count)

```

```

{
    String res = "\nШаг № " + count.ToString() + "\nНачальная точка (";
    for (int i = 0; i < n; i++)
    {
        res = res + x[i].ToString();
        if (i != n - 1)
        {
            res = res + "; ";
        }
    }
    res = res + ") Шаг dx = " + dx.ToString();
    double[] xtest = new double[n];
    double[] xold = new double[n];
    for (int i = 0; i < n; i++)
    {
        xtest[i] = x[i];
        xold[i] = x[i];
    }
    bool flag = false;
    for (int i = 0; i < n; i++)
    {
        xtest[i] = x[i] + dx;
        if (comp(xtest) < comp(x))
        {
            x[i] = x[i] + dx;
            flag = true;
        }
        else
        {
            xtest[i] = x[i] - dx;

```

```

        if (comp(xtest) < comp(x))
        {
            x[i] = x[i] - dx;
            flag = true;
        }
    }
    xtest[i] = x[i];
}
res = res + "\nНовая точка (";
for (int i = 0; i < n; i++)
{
    res = res + x[i].ToString();
    if (i != n - 1)
    {
        res = res + "; ";
    }
}
res = res + ")";
if (dx < e)
{
    return res;
}
if (flag == false)
{
    return res + MethodHookeJeeves(dx / a, a, b, e, count + 1);
}
else
{
    for (int i = 0; i < n; i++)
    {

```

```

        xtest[i] = x[i] + b * (x[i] - xold[i]);
    }
    if (comp(xtest) < comp(x))
    {
        for (int i = 0; i < n; i++)
        {
            x[i] = xtest[i];
        }
    }

    for (int i = 0; i < n; i++)
    {
        res = res + x[i].ToString();
        if (i != n - 1)
        {
            res = res + "; ";
        }
    }
    res = res + ")";
    return res + MethodHookeJeeves(dx, a, b, e, count + 1);

```

```

        res = res + "\nПоиск по образцу неудачен";
    return res + MethodHookeJeeves(dx / a, a, b, e, count + 1);
    }
}

```

```

public double[] projection(double[] a, double[] b, int n)
r    {
e        double ab = 0;
s
r
e

```

```

double bb = 0;
for (int j = 0; j < n; j++)
{
    ab = ab + a[j] * b[j];
    bb = bb + b[j] * b[j];
}
for (int i = 0; i < n; i++)
{
    b[i] = ab * b[i] / bb;
}
return b;
}

public String MethodRosenbrock(double dx, double a, double b, double e, int
count)
{
    string str = "Шаг № " + count + "\nНачальная точка (";
    for (int i = 0; i < n; i++)
    {
        str = str + Convert.ToDouble(x[i]);
        if (i != n - 1)
        {
            str = str + "; ";
        }
        else
            str = str + ")\n";
    }
    double[] xtest = new double[n];
    double[] xold = new double[n];
    for (int i = 0; i < n; i++)
    {

```

```

    xtest[i] = x[i];
    xold[i] = x[i];
}
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        xtest[j] = x[j] + dx * p[j][i];
    }
    if (comp(xtest) < comp(x))
    {
        for (int j = 0; j < n; j++)
        {
            xtest[j] = x[j] + b * dx * p[j][i];
        }
    }
    else
    {
        for (int j = 0; j < n; j++)
        {
            xtest[j] = x[j] - a * dx * p[j][i];
        }
    }
    for (int j = 0; j < n; j++)

}

str = str + "Новая точка (";
for (int i = 0; i < n; i++)

```

```

{
    str = str + Convert.ToDouble(x[i]);
    if (i != n - 1)
    {
        str = str + "; ";
    }
    else
        str = str + ")\n";
}

double r = 0;
for (int i = 0; i < n; i++)
{
    r = r + Math.Pow(x[i] - xold[i], 2);
}

if (Math.Sqrt(r) < e)
    return str;

//ортогонализация
for (int j = 0; j < n; j++)
{
    p[j][0] = x[j] - xold[j];
}

int count2 = 1;
while (count2 < n)
{
    double[] a1 = new double[n];
    double[] b1 = new double[n];
    for (int i2 = 0; i2 < n; i2++)
    {
        a1[i2] = p[i2][count2];
        b1[i2] = p[i2][count2 - 1];
    }
}

```



```
    }  
    double[] a1b1 = projection(a1, b1, n);  
    for (int j = 0; j < n; j++)  
    {  
        p[j][count2] = p[j][count2] - a1b1[j];  
    }  
    count2++;  
}  
return str + MethodRosenbrock(dx, a, b, e, count + 1);
```

Приложение В  
(Код программы)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp15
{
    public partial class Form1 : Form
    {
        TextBox[] textBoxes;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            textBoxes = new TextBox[6];
            textBoxes[0] = textBox1;
            textBoxes[1] = textBox2;
            textBoxes[2] = textBox3;
            textBoxes[3] = textBox4;
            textBoxes[4] = textBox5;
            textBoxes[5] = textBox6;
        }
    }
}
```

```

private void button1_Click(object sender, EventArgs e)
{
    if(Convert.ToInt32(button1.Text) == 6)
    {
        button1.Text = "1";
    }
    else
    {
        button1.Text = (Convert.ToInt32(button1.Text) + 1).ToString(); }
}

private void методToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        int n = Convert.ToInt32(button1.Text);
        double[] x = new double[n];
        for (int i = 0; i < n; i++)
        {
            x[i] = Convert.ToDouble(textBoxes[i].Text.Replace(".", ","));
        }
        double dx = Convert.ToDouble(textBox8.Text.Replace(".", ","));
        double a = Convert.ToDouble(textBox9.Text.Replace(".", ","));
        double b = Convert.ToDouble(textBox11.Text.Replace(".", ","));
        double eps = Convert.ToDouble(textBox10.Text.Replace(".", ","));
        Service methods = new Service(x, n);
        methods.F = textBox7.Text;
        String Result = methods.MethodHookeJeeves(dx, a, b, eps, 1);
        richTextBox1.Text = Result;
    }
}

```

```

        catch(Exception ex)
        {
            MessageBox.Show("Проверьте размерность "+ex.Message);
        }
    private void методРозенброкаToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        try
        {
            int n = Convert.ToInt32(button1.Text);
            double[] x = new double[n];
            for (int i = 0; i < n; i++)
            {
                x[i] = Convert.ToDouble(textBoxes[i].Text.Replace(".", ",")); }
            double dx = Convert.ToDouble(textBox8.Text.Replace(".", ","));
            double a = Convert.ToDouble(textBox9.Text.Replace(".", ","));
            double b = Convert.ToDouble(textBox11.Text.Replace(".", ","));
            double eps = Convert.ToDouble(textBox10.Text.Replace(".", ","));
            Service methods = new Service(x, n);
            methods.F = textBox7.Text;
            String Result = methods.MethodRosenbrock(dx, a, b, eps, 1);
            richTextBox1.Text = Result;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Проверьте размерность " + ex.Message);
        }
    }
}

```

### Список источников

1. Дегтярев Ю.И. Методы оптимизации: Учеб. пособие для спец. 0646 - "Автоматизир. системы упр."; 0647 - "Прикл. математика". М.: Сов. радио, 1980. 270 с.
2. [https://ami.nstu.ru/~headrd/seminar/publik\\_html/MO\\_conspect.pdf](https://ami.nstu.ru/~headrd/seminar/publik_html/MO_conspect.pdf)
3. Лесин В.В. Основы методов оптимизации: Учеб. пособие для втузов М.: МАИ , 1998. 340 с.