МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тульский государственный университет»

Институт прикладной математики и компьютерных наук

Кафедра вычислительной техники

Курсовая работа по дисциплине

ЧИСЛЕННЫЕ МЕТОДЫ

на тему: ЧИСЛЕННОЕ РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Студент группы 220681 Северин Д.В. \_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись, дата)

Руководитель работы к.т.н., доцент каф. ВТ Волошко А.Г. \_\_\_\_\_\_\_\_\_

(Ф.И.О., должность) (подпись, дата)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тула 2020

МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «Тульский государственный университет»

Институт прикладной математики и компьютерных наук

Кафедра «Вычислительная техника»

**Задание**

На курсовую работу по дисциплине «Численные методы»

студенту группы 220681 Северину Денису Валерьевичу

Тема работы:

«Методы решения дифференциальных уравнений»

Входные данные Вариант №19:

задача: решение дифференциальных уравнений методом Эйлера и методом Адамса;

уравнение: на отрезке [a,b], c шагом h;

приложение №1: приложение для решения численными методами дифференциальных уравнений.

Задание получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «20» февраля 2020 г.

(подпись студента)

Срок представления задания «19» июня 2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка подписи)

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

К защите. Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка подписи)

Замечания руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

*При защите курсового проекта (работы) наличие рецензии обязательно.*

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc42040451)

[1 Постановка задачи 5](#_Toc42040452)

[2 Математическое описание методов 6](#_Toc42040453)

[2.1 Метод Эйлера 6](#_Toc42040454)

[2.2 Метод Адамса 8](#_Toc42040455)

[3 Описание входных и выходных данных 12](#_Toc42040456)

[3.1 Метод Эйлера 12](#_Toc42040457)

[3.2 Метод Адамса 12](#_Toc42040458)

[4 Алгоритмы решения дифференциальных уравнений 13](#_Toc42040459)

[4.1 Метод Эйлера 13](#_Toc42040460)

[4.2 Метод Адамса 14](#_Toc42040461)

[4.3 Тестовая функция 16](#_Toc42040462)

[5 Программная реализация 17](#_Toc42040463)

[6 Тестирование работы программы 19](#_Toc42040464)

[7 Проверка корректности работы программы 22](#_Toc42040465)

[ЗАКЛЮЧЕНИЕ 26](#_Toc42040466)

[СПИСОК ЛИТЕРАТУРЫ 27](#_Toc42040467)

ВВЕДЕНИЕ

Целью выполнения курсовой работы является закрепление теоретических знаний по курсу «Численные методы» и получения практических навыков:

* анализа различных численных методов решения дифференциальных уравнений;
* разработки программных средств для решения численными методами дифференциальных уравнений.

Задачами курсовой работы являются:

* приобретение навыков анализа различных численных методов решения дифференциальных уравнений;
* приобретение навыков разработки программных средств для решения численными методами дифференциальных уравнений;
* сравнительный анализ вычислительной эффективности программ.

1 Постановка задачи

Необходимо разработать программное обеспечение, которое решает дифференциальное уравнение и сравнить методы по вычислительной эффективности. Приложение решает одну и ту же задачу методом Эйлера и методом Адамса.

Вариант задания №3:

Заданное уравнение: на отрезке [a,b], c шагом h.

Приложение №1 – приложение для решения численными методами дифференциальных уравнений.

2 Математическое описание методов

2.1 Метод Эйлера

Решить дифференциальное уравнение  численным методом - это значит для заданной последовательности аргументов  и числа , не определяя функцию *у = F(x),* найти такие значения  что  (*i = 1,2,...,n*) и .

Таким образом, численные методы позволяют вместо нахождения функции *у = F(x)* получить таблицу значений этой функции для заданной последовательности аргументов. Величина  называется *шагом интегрирования*. Рассмотрим некоторые из численных методов.

Метод Эйлера является сравнительно грубым и применяется в основном для ориентировочных расчетов.

Пусть дано дифференциальное уравнение первого порядка

 (1)

с начальным условием

 (2)

Требуется найти решение уравнения (1) на отрезке [а, b].

Разобьем отрезок [а,b] на n равных частей и получим последова­тельность , где  (*i = 1, 2,..., n*), a  - шаг интегрирования.

Выберем *k*-й участок  и проинтегрируем уравнение (1):

. (3)

Тогда формула (3) примет вид

 (4)

Обозначив,  т.е. , получим

 (5)

Продолжая этот процесс и каждый раз принимая подынтегральную функцию на соответствующем участке постоянной и равной ее значению в начале участка, получим таблицу решений дифференциального уравнения на заданном отрезке [а, b].

Если функция *f(x,y)* в некотором прямоугольнике



удовлетворяет условию

 (*N = const*) (6)

и, кроме того.

 (М = const) (7)

то имеет место следующая оценка погрешности:

, (8)

где  - значение точного решения уравнения (5.61) при ,а  - приближенное значение, полученное на *n*-м шаге.

Формула (8) имеет в основном теоретическое применение. На практике, как правило, применяют "двойной просчет". Сначала расчет ведется с шагом *h*, затем шаг дробят и повторный расчет ведется с шагом . Погрешность более точного значения  оценивается формулой

 (9)

Метод Эйлера может быть применен к решению систем дифференциальных уравнений и дифференциальных уравнений высших порядков. Однако в последнем случае дифференциальные уравнения должны быть приведены к системе дифференциальных уравнений первого порядка.

Пусть задана система двух уравнений первого порядка

 (10)

с начальными условиями

,  (11)

Приближенные значения  и  находятся по формулам

,  (12)

где ,  (*i = 0,1,2,…*). (13)

2.2 Метод Адамса

При решении дифференциального уравнения методом Рунге-Кутта необходимо производить много вычислений для нахождения каждого. В том случае, когда правая часть уравнения сложное аналитическое выражение, решение такого уравнения методом Рунге-Кутта вызывает большие трудности. Поэтому на практике применяется метод Адамса, который не требует многократного подсчета правой части уравнения.

Пусть дано дифференциальное уравнение

, (16)

с начальным условием

, . (17)

Требуемся найти решение этого уравнения на отрезке [a.b].

Разобьем отрезок [a,b] на n равных частей точками    
(i = 1, 2,..., n), a  – проинтегрируем дифференциальное уравнение). Выберем участок  и проинтегрируем дифференциальное уравнение (18); тогда получим

, (19)

или

. (20)

Для нахождения производной воспользуемся второй интерполяцион­ной формулой Ньютона (ограничиваясь при этом разностями третьего по­рядка):

. (21)

или

. (22)

Подставляя выражение для  из формулы (22) в соотношение (20) и учитывая, что , имеем

 (23)

Обозначим в дальнейшем  (*i = 0,1,2,…,n*).

Тогда для любой разности имеем  и

. (24)

По формуле  получаем решение уравнения. Формула (24) носит название экстраполяционной формулы Адамса.

Для начала процесса нужны четыре начальных значения  - так называемый начальный отрезок, который может быть найден, исходя из начального условия (19) с использованием одного из известных методов. Обычно начальный отрезок решения находится методом Рунге-Кутта.

Зная  можно определить

; ;

; .

Далее составляется таблица разностей величины q (табл. 1).

Таблица 1. Таблица разностей величины *q*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I |  |  |  |  |  |  |  |  |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) |
| 0 |  |  | - |  |  |  |  |  |
| 1 |  |  | - |  |  |  |  | - |
| 2 |  |  | - |  |  |  | - | - |
| 3 |  |  |  |  |  | - | - | - |
| 4 |  |  | - | - | - | - | - | - |
| 5 |  | - | - | - | - | - | - | - |
| 6 |  | - | - | - | - | - | - | - |

Метод Адамса заключается в продолжении диагональной таблицы разностей с помощью формулы (24). Используя числа , которые располагаются в таблице по диагонали, полагая в формуле   
 n = 3 (известное последнее значение у есть ), получаем:

.

Полученное значение  вносят и таблицу и находят . Затем, используя значения  и , находят  т.е. получается новая диагональ. По этим данным вычисляют

 .

Таким образом, продолжают таблицу решения, вычисляя правую часть дифференциального уравнения (18) на каждом этапе только один раз.

Для грубой оценки погрешности применяют принцип Рунге, который состоит в следующем:

1. Находят решение дифференциального уравнения при шаге *h*.
2. Значение шага удваивают и находят решение при шаге *Н = 2h*.

3. Вычисляют погрешность метода по формуле

,

где  - значение приближенного вычисления при двойном шаге H=2h;  - значение приближенного вычисления при шаге h.

Метод Адамса применяется также и для решения систем дифференциальных уравнений и дифференциальных уравнений n-гo порядка.

Пусть задана система двух уравнений



Тогда экстраполяционные формулы Адамса для этой системы имеют вид



где   и ,

.

3 Описание входных и выходных данных

3.1 Метод Эйлера

Объявление:

public List<List<double>> EulerMethod(double leftBorder, double rightBorder, double step, double initialY, bool fromAdams = false)

Входные данные:

Начальное значение y – переменная initialY тип данных double.

Левая граница отрезка – переменная leftBorder тип данных double.

Правая граница отрезка – переменная rightBorder тип данных double.

Шаг – переменная step тип данных double.

Выходные дынные:

Двумерный динамический массив данных, содержащий X и Y – тип данных List<List<double>>

3.2 Метод Адамса

Объявление:

public List<List<double>> AdamsMethod(double leftBorder, double rightBorder, double step, double initialY)

Входные данные:

Начальное значение y – переменная initialY тип данных double.

Левая граница отрезка – переменная leftBorder тип данных double.

Правая граница отрезка – переменная rightBorder тип данных double.

Шаг – переменная step тип данных double.

Выходные дынные:

Двумерный динамический массив данных, содержащий X и Y – тип данных List<List<double>>

4 Алгоритмы решения дифференциальных уравнений

4.1 Метод Эйлера

На первом этапе происходит вычисление y по формуле:

deltaY = step \* F(x, y);

y = deltaY + y;

На втором этапе происходит вычисление x:

x += step;

И так, в цикле для всех точек на данном промежутке с заданным шагом.

Блоксхема алгоритма метода Эйлера представлена на рисунке 1.

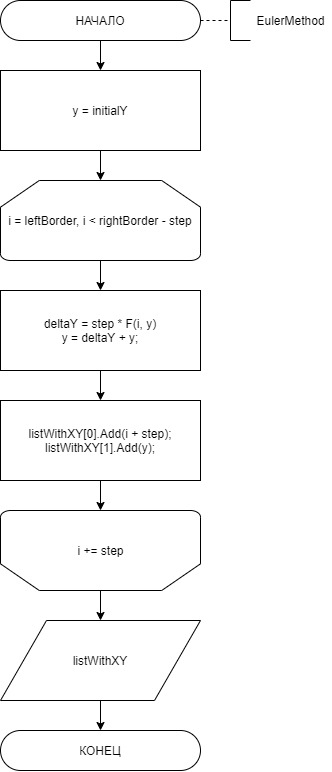


Рисунок 1 – Блок-схема метода Эйлера

4.2 Метод Адамса

- Получаем начальный отрезок с помощью метода Эйлера;

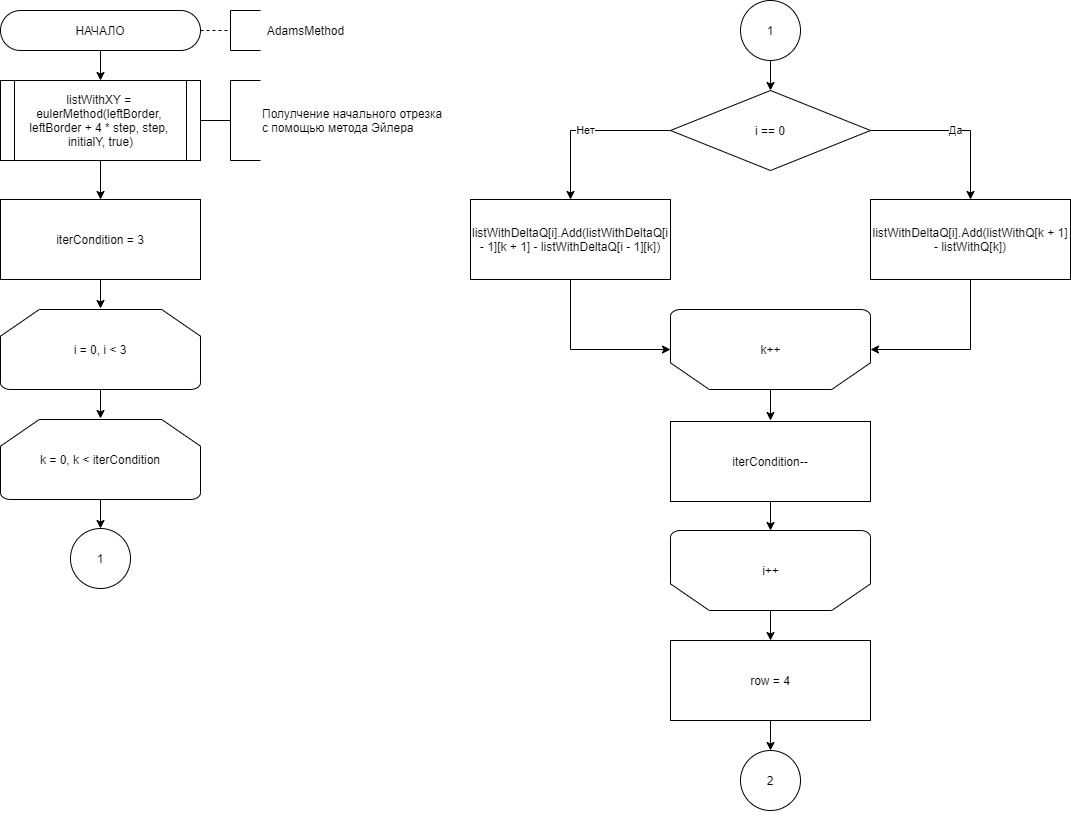
В цикле от начального x0 до конечного xn с шагом h:

- Вычисляем значения , , , 

- Вычисляется значение Δyn­

- Вычисляется значение yn

Блок-схема алгоритма метода Адамса представлена на рисунках 2 - 3.

  
Рисунок 2 – 1 часть блок-схемы алгоритма метода Адамса

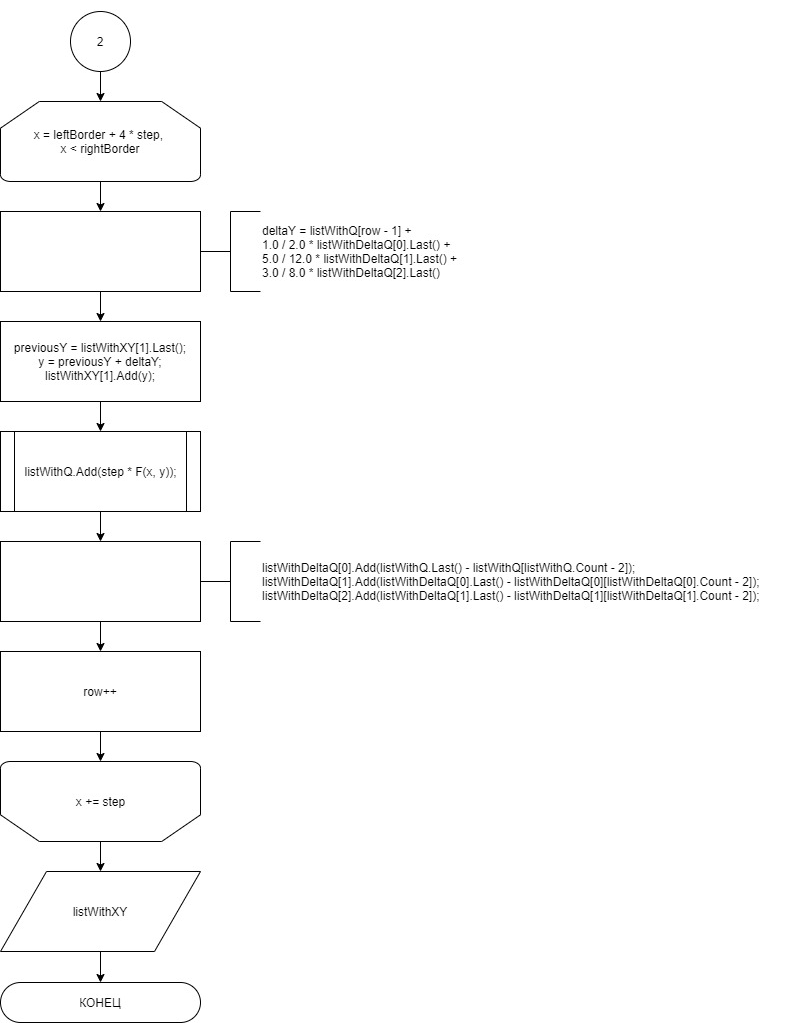


Рисунок 3 – 2 часть блок-схемы алгоритма метода Адамса

4.3 Тестовая функция

Блок-схема алгоритма тестовой функции представлена на рисунке 4.

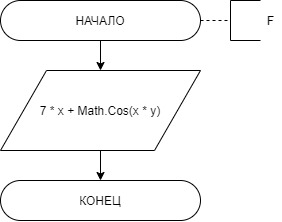


Рисунок 4 – Блок-схема алгоритма тестовой функции

5 Программная реализация

В данной работе были реализованы следующие методы:

* Метод Эйлера:

public List<List<double>> eulerMethod(double leftBorder, double rightBorder, double step, double initialY, bool fromAdams = false)

* Метод Адамса:

public List<List<double>> adamsMethod(double leftBorder, double rightBorder, double step, double initialY)

На вход обоих методов подаются следующие параметры:

leftBorder – левая граница интервала x;

leftBorder – правая граница интервала x;

step – шаг изменения x на промежутке [a,b];

initialY – начальное значение функции y0.

На вход в метод Эйлера дополнительно может передаваться булево значение fromAdams, изначально инициализированное как false, которое влияет на запуск таймера.

В данных методах происходит решение дифференциального уравнения на заданном интервале, с учётом начального значения y0 и шага step. Происходит вывод решения в каждой точке на экран, а также время выполнения каждого из методов.

Решение дифференциального уравнения начинается после заполнения всех полей входных данных и нажатия на кнопку. Если пользователь ввел некорректное значение какого-то из полей, то всплывет окно с информацией о том, в каком поле пользователь допустил ошибку. Все возможные ошибки, которые может вывести программа представлены на рисунке 5.

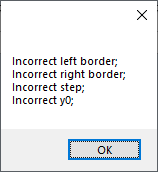


Рисунок 5 – Окно предупреждения

Блок-схема обработчика события нажатия на кнопку представлена на рисунке 6.

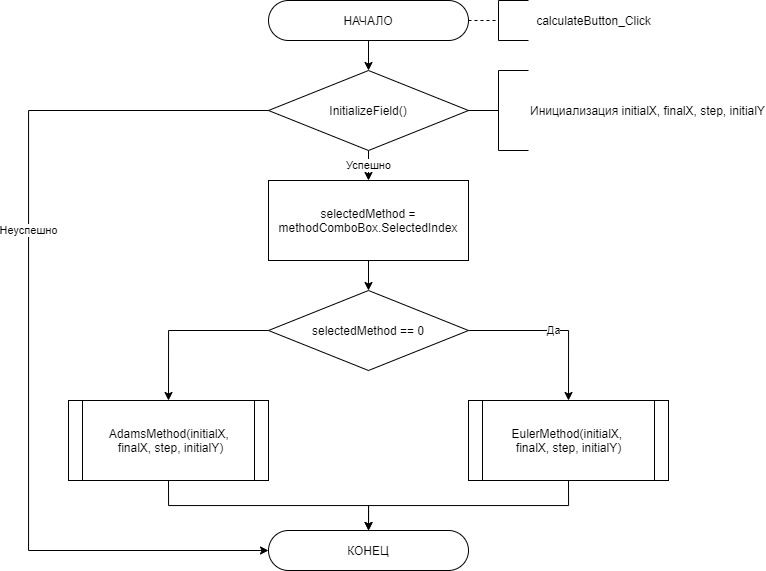


Рисунок 6 – Блок-схема обработчика события нажатия на кнопку

6 Тестирование работы программы

Форма программы выглядит следующим образом – рисунок 7.

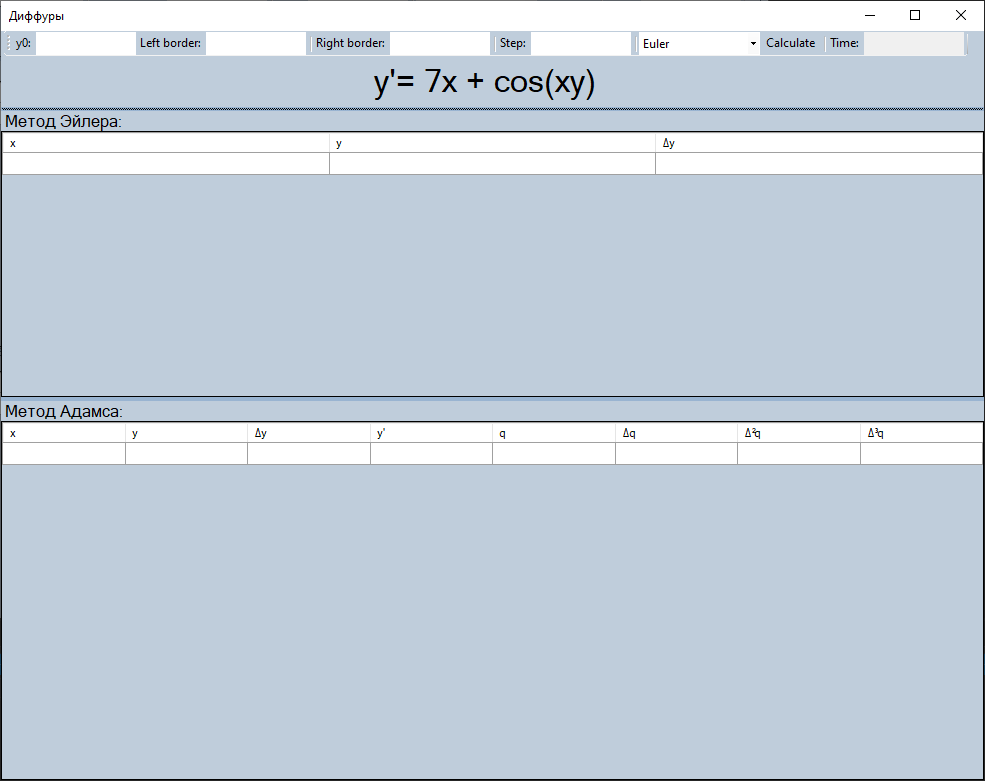


Рисунок 7 – Форма программы

Для тестирования использовались следующие входные данные – рисунок 8.



Рисунок 8 – Входные данные для тестирования

Выбираем метод Эйлера в выпадающем меню и получаем заполненную таблицу значений, рисунок 9.

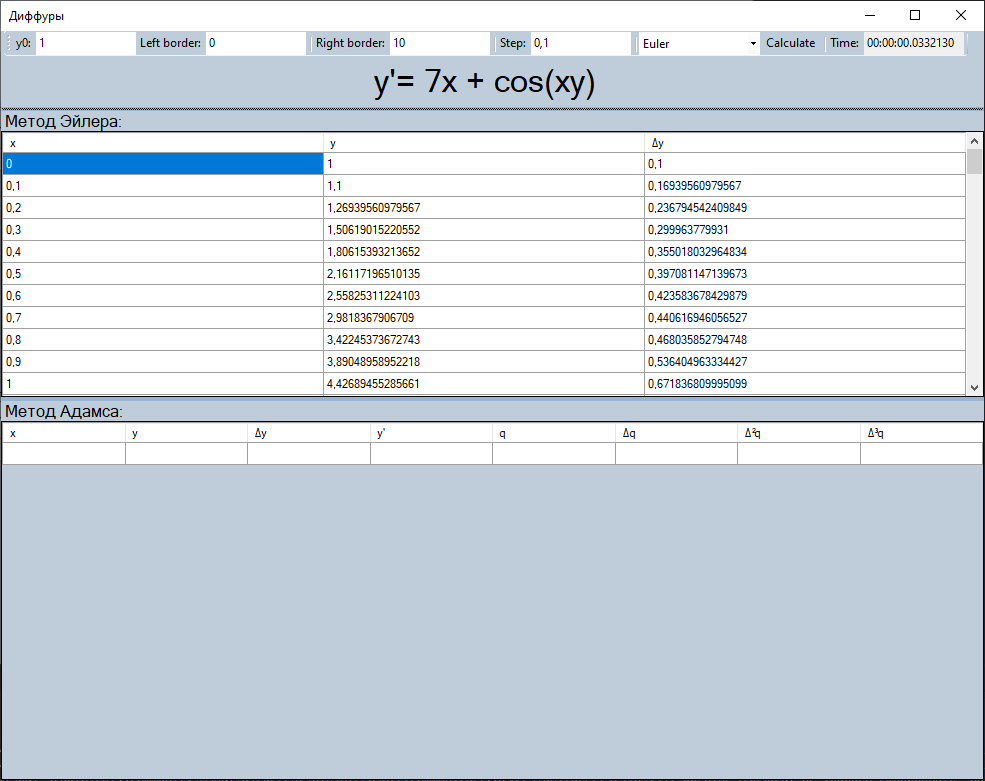


Рисунок 9 – Вывод метода Эйлера

Далее выбираем метод Адамса и получаем следующий вывод – рисунок 10.

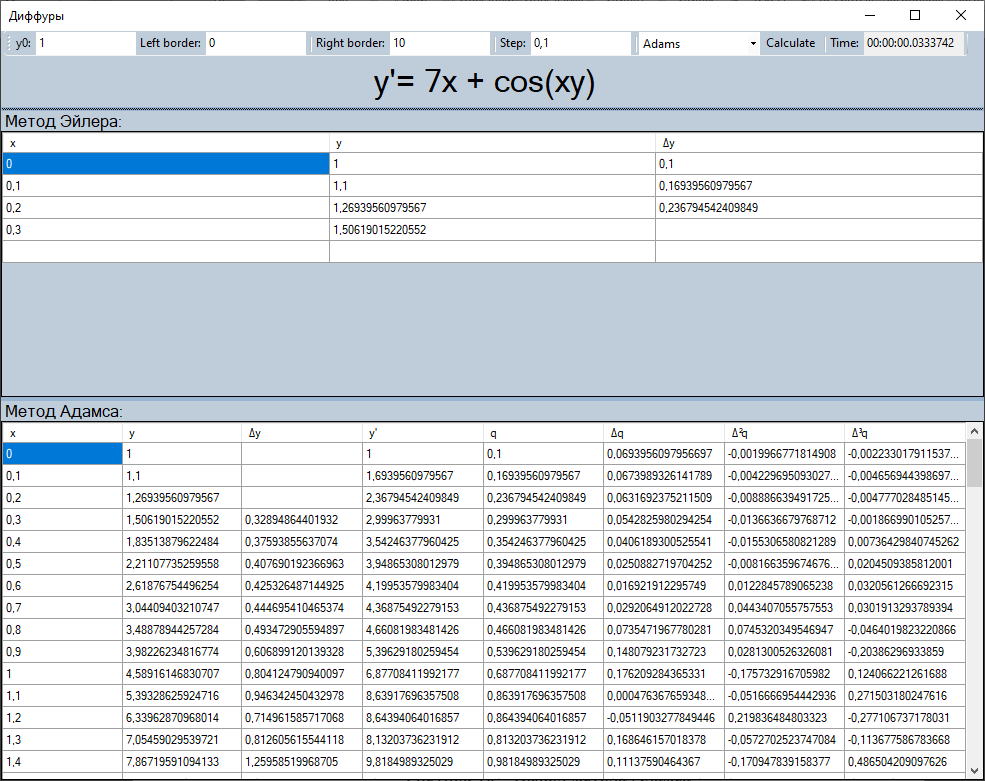


Рисунок 10 – Вывод метода Адамса

7 Проверка корректности работы программы

Для проверки корректности работы метода Эйлера в разработанной программе воспользуемся онлайн калькулятором на сайте www.mathstools.com, рисунок 11.

Входные данные для проверки:

* y0 – 1;
* Левая граница – 0;
* Правая граница – 10;
* Шаг – 0,1.

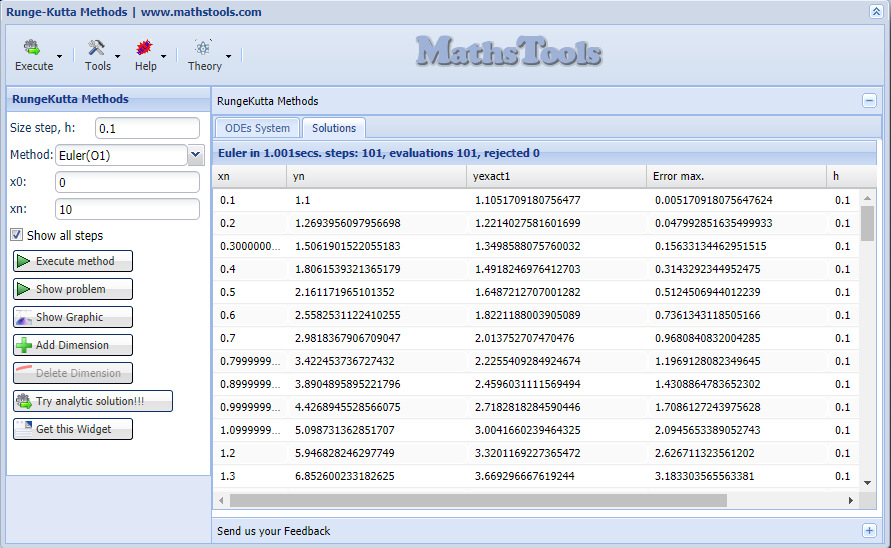


Рисунок 11 – Результат работы онлайн-калькулятора mathstools

Результат работы программы представлен на рисунке 12. Сравнивая выводы данных двух приложений, можно сделать вывод, что разработанная программа работает корректно.

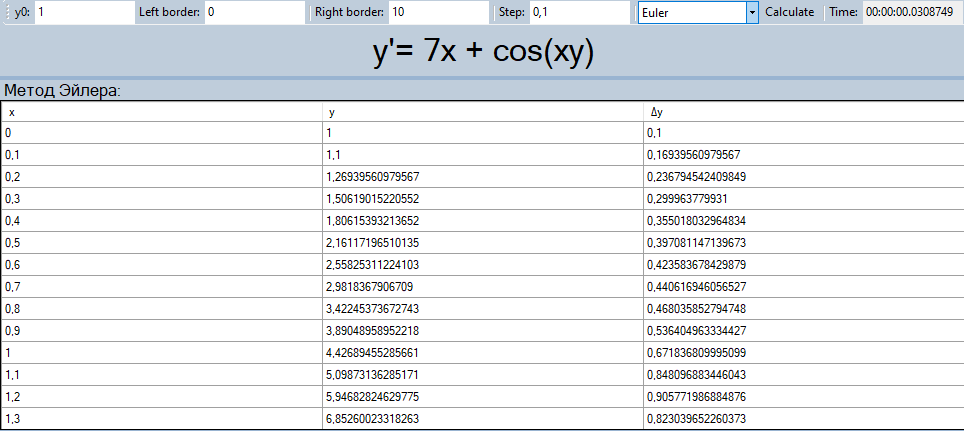


Рисунок 12 – Результат работы метода Эйлера

Для проверки корректности работы метода Адамса в разработанной программе воспользуемся онлайн калькулятором на сайте www.atozmath.com, рисунок 13.

Входные данные для проверки:

* y­0 = 1;
* x0 = 0;
* x = 4.

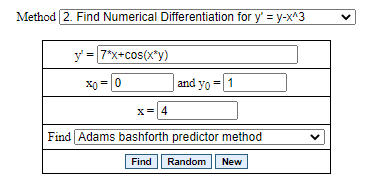
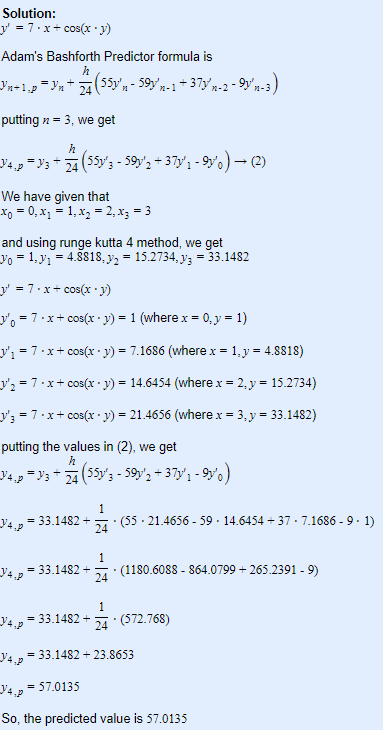


Рисунок 13 – Входные данные

В результате получаем значение в точке 4 равное 57.0135, рисунок 14.

  
Рисунок 14 – Результат работы онлайн-калькулятора atozmath

Результат работы разработанной программы представлен на рисунке 15. Значение в точке 4 равно 56.8849, что близко к значению, полученному в онлайн-калькуляторе. Следовательно, программа работает корректно.



Рисунок 15 – Результат работы метода Адамса

Для тестирования использовались следующие входные данные – рисунок 16. В ходе тестирования изменялся только шаг. Значения шагов указаны в таблице 2.



Рисунок 16 – Входные данные для тестирования

Таблица 2. Сравнение вычислительной эффективности методов

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | № Теста | Метод Эйлера | Метод Адамса |
| Время | |
| 0,1 | 1 | 00:00:00.0406746 | 00:00:00.0383048 |
| 2 | 00:00:00.0383014 | 00:00:00.0551301 |
| 3 | 00:00:00.0427775 | 00:00:00.0380591 |
| 4 | 00:00:00.0416503 | 00:00:00.0354085 |
| 5 | 00:00:00.0408466 | 00:00:00.0388186 |
| **Среднее время:** | **00:00:00.0408500** | **00:00:00.0371442** |
| 0,01 | 1 | 00:00:00.3989898 | 00:00:00.3467972 |
| 2 | 00:00:00.3625528 | 00:00:00.3921496 |
| 3 | 00:00:00.4219585 | 00:00:00.3739906 |
| 4 | 00:00:00.3951218 | 00:00:00.3613257 |
| 5 | 00:00:00.3806603 | 00:00:00.3582752 |
| **Среднее время:** | **00:00:00.3918566** | **00:00:00.3665076** |
| 0,001 | 1 | 00:00:03.5941100 | 00:00:03.4706993 |
| 2 | 00:00:03.5717281 | 00:00:03.4233186 |
| 3 | 00:00:03.7405126 | 00:00:03.4720503 |
| 4 | 00:00:03.5996789 | 00:00:03.4112429 |
| 5 | 00:00:03.5542403 | 00:00:03.4856182 |
| **Среднее время:** | **00:00:03.6120539** | **00:00:03.4525858** |

Из таблицы 2 можно сделать вывод, что метод Адамса работает быстрее, чем метод Эйлера. Следовательно вычислительная эффективность метода Адамса выше, чем метода Эйлера.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы было разработано программное обеспечение для решения численными методами дифференциальных уравнений. Разработанное программное обеспечение решает дифференциальное уравнение методами Эйлера и Адамса.

Было проведено сравнение времени работы методов. Исследования эффективности методов показали, что метод Адамса работает быстрее, чем метод Эйлера.

Все поставленные задачи были выполнены, а именно:

* приобретены навыки анализа различных численных методов решения дифференциальных уравнений;
* приобретены навыки разработки программных средств для решения численными методами дифференциальных уравнений;
* приобретены навыки сравнительного анализа вычислительной эффективности программ.

СПИСОК ЛИТЕРАТУРЫ

1. Пирумов, У.Г. Численные методы / У.Г. Перумов. – М.: Дрофа, 2007. – 222 с.
2. Математика : практикум по численным методам / Белорус. нац. техн. ун-т, Каф. "Высшая математика №1"; сост. :А.В.Грекова [и др.] .— Минск, 2006 .— 127с. — Библиогр.в конце кн. — ISBN 985-479-453-9
3. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
4. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
5. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
6. ГОСТ 19.701-90 (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем.
7. Шилдт Г. - C# 4.0 полное руководство – Москва: Вильямс, 2011. – 1056 с.
8. Метод Эйлера // Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Метод_Эйлера>
9. Метод Адамса // Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Метод_Адамса>

**ПРИЛОЖЕНИЕ**

**Листинг 1. Файл Form1.cs**

using System;

using System.Windows.Forms;

namespace differencialCourse

{

using static Convert;

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

methodComboBox.SelectedIndex = 0;

}

private void calculateButton\_Click(object sender, EventArgs e)

{

if (InitializeField(out var initialX, out var finalX, out var step, out var initialY) == false) return;

var methods = new Methods(eulerDataGrid, adamsDataGrid, timeTextBox);

var selectedMethod = methodComboBox.SelectedIndex;

if (selectedMethod == 0)

methods.EulerMethod(initialX, finalX, step, initialY);

else if (selectedMethod == 1)

methods.AdamsMethod(initialX, finalX, step, initialY);

else if (selectedMethod == 2) methods.AdamsPredictorMethod(initialX, finalX, step, initialY);

}

private bool InitializeField(out double initialX, out double finalX, out double step, out double initialY)

{

var ifValidValues = true;

var errorString = "";

try

{

initialX = ToDouble(leftBorderBox.Text.Replace('.', ','));

}

catch

{

initialX = 0;

errorString += "Incorrect left border;\n";

ifValidValues = false;

}

try

{

finalX = ToDouble(rightBorderBox.Text.Replace('.', ','));

}

catch

{

finalX = 0;

errorString += "Incorrect right border;\n";

ifValidValues = false;

}

try

{

step = ToDouble(stepBox.Text.Replace('.', ','));

}

catch

{

step = 0;

errorString += "Incorrect step;\n";

ifValidValues = false;

}

try

{

initialY = ToDouble(yTextBox.Text.Replace('.', ','));

}

catch

{

initialY = 0;

errorString += "Incorrect y0;\n";

ifValidValues = false;

}

if (initialX > finalX)

{

errorString += "Left border must be lesser than right;\n";

ifValidValues = false;

}

if (!ifValidValues) MessageBox.Show(errorString);

return ifValidValues;

}

}

}

**Листинг 2. Файл Methods.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Windows.Forms;

namespace differencialCourse

{

internal class Methods

{

private readonly DataGridView \_eulerDataGridView;

private readonly DataGridView \_adamsDataGridView;

private readonly ToolStripTextBox \_time;

public Methods(DataGridView eulerDataGridView, DataGridView adamsDataGridView, ToolStripTextBox time)

{

\_eulerDataGridView = eulerDataGridView;

\_adamsDataGridView = adamsDataGridView;

\_time = time;

}

public List<List<double>> EulerMethod(double leftBorder, double rightBorder, double step, double initialY, bool fromAdams = false)

{

var stopWatch = new Stopwatch();

if (!fromAdams)

{

stopWatch.Start();

}

\_eulerDataGridView.Rows.Clear();

\_eulerDataGridView.Rows.Add();

\_eulerDataGridView.Rows[0].Cells[0].Value = leftBorder;

\_eulerDataGridView.Rows[0].Cells[1].Value = initialY;

var listWithXY = new List<List<double>> { new List<double>(), new List<double>() };

listWithXY[0].Add(leftBorder);

listWithXY[1].Add(initialY);

var row = 0;

var y = initialY;

for (var i = leftBorder; i < rightBorder - step; i += step)

{

\_eulerDataGridView.Rows.Add();

var deltaY = step \* F(i, y);

\_eulerDataGridView.Rows[row].Cells[2].Value = deltaY;

row++;

\_eulerDataGridView.Rows[row].Cells[0].Value = i + step;

\_eulerDataGridView.Rows[row].Cells[1].Value = deltaY + y;

y = deltaY + y;

listWithXY[0].Add(i + step);

listWithXY[1].Add(y);

}

if (fromAdams) return listWithXY;

stopWatch.Stop();

var ts = stopWatch.Elapsed;

\_time.Text = ts.ToString();

stopWatch.Reset();

return listWithXY;

}

public List<List<double>> AdamsMethod(double leftBorder, double rightBorder, double step, double initialY)

{

var stopWatch = new Stopwatch();

stopWatch.Start();

\_adamsDataGridView.Rows.Clear();

List<List<double>> listWithXY = EulerMethod(leftBorder, leftBorder + 4 \* step, step, initialY,true);

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

List<List<double>> listWithDeltaQ = new List<List<double>> { new List<double>(), new List<double>(), new List<double>() };

listWithQ.Add(step \* F(listWithXY[0][0], listWithXY[1][0]));

listWithQ.Add(step \* F(listWithXY[0][1], listWithXY[1][1]));

listWithQ.Add(step \* F(listWithXY[0][2], listWithXY[1][2]));

listWithQ.Add(step \* F(listWithXY[0][3], listWithXY[1][3]));

\_adamsDataGridView.Rows.Add(listWithXY[0][0], listWithXY[1][0], null, F(listWithXY[0][0], listWithXY[1][0]), listWithQ[0]);

\_adamsDataGridView.Rows.Add(listWithXY[0][1], listWithXY[1][1], null, F(listWithXY[0][1], listWithXY[1][1]), listWithQ[1]);

\_adamsDataGridView.Rows.Add(listWithXY[0][2], listWithXY[1][2], null, F(listWithXY[0][2], listWithXY[1][2]), listWithQ[2]);

\_adamsDataGridView.Rows.Add(listWithXY[0][3], listWithXY[1][3], null, F(listWithXY[0][3], listWithXY[1][3]), listWithQ[3]);

var iterCondition = 3;

for (var i = 0; i < 3; i++)

{

for (var k = 0; k < iterCondition; k++)

{

if (i == 0)

{

listWithDeltaQ[i].Add(listWithQ[k + 1] - listWithQ[k]);

\_adamsDataGridView.Rows[k].Cells[i + 5].Value = listWithDeltaQ[i].Last();

continue;

}

listWithDeltaQ[i].Add(listWithDeltaQ[i - 1][k + 1] - listWithDeltaQ[i - 1][k]);

\_adamsDataGridView.Rows[k].Cells[i + 5].Value = listWithDeltaQ[i].Last();

}

iterCondition--;

}

var row = 4;

for (var x = leftBorder + 4 \* step; x < rightBorder; x += step)

{

\_adamsDataGridView.Rows.Add(x);

var deltaY = listWithQ[row - 1] +

1.0 / 2.0 \* listWithDeltaQ[0].Last() +

5.0 / 12.0 \* listWithDeltaQ[1].Last() +

3.0 / 8.0 \* listWithDeltaQ[2].Last();

\_adamsDataGridView.Rows[row - 1].Cells[2].Value = deltaY;

var previousY = listWithXY[1].Last();

var y = previousY + deltaY;

listWithXY[1].Add(y);

\_adamsDataGridView.Rows[row].Cells[1].Value = y;

\_adamsDataGridView.Rows[row].Cells[3].Value = F(x, y);

listWithQ.Add(step \* F(x, y));

\_adamsDataGridView.Rows[row].Cells[4].Value = listWithQ.Last();

listWithDeltaQ[0].Add(listWithQ.Last() - listWithQ[listWithQ.Count - 2]);

listWithDeltaQ[1].Add(listWithDeltaQ[0].Last() - listWithDeltaQ[0][listWithDeltaQ[0].Count - 2]);

listWithDeltaQ[2].Add(listWithDeltaQ[1].Last() - listWithDeltaQ[1][listWithDeltaQ[1].Count - 2]);

\_adamsDataGridView.Rows[row - 1].Cells[5].Value = listWithDeltaQ[0].Last();

\_adamsDataGridView.Rows[row - 2].Cells[6].Value = listWithDeltaQ[1].Last();

\_adamsDataGridView.Rows[row - 3].Cells[7].Value = listWithDeltaQ[2].Last();

row++;

}

stopWatch.Stop();

var ts = stopWatch.Elapsed;

\_time.Text = ts.ToString();

stopWatch.Reset();

return listWithXY;

}

public void AdamsPredictorMethod(double leftBorder, double rightBorder, double step, double initialY)

{

\_adamsDataGridView.Rows.Clear();

List<List<double>> listWithXY = EulerMethod(leftBorder, leftBorder + 4 \* step, step, initialY, true);

\_adamsDataGridView.Rows.Add(listWithXY[0][0], listWithXY[1][0], null, F(listWithXY[0][0], listWithXY[1][0]), step \* F(listWithXY[0][0], listWithXY[1][0]));

\_adamsDataGridView.Rows.Add(listWithXY[0][1], listWithXY[1][1], null, F(listWithXY[0][1], listWithXY[1][1]), step \* F(listWithXY[0][1], listWithXY[1][1]));

\_adamsDataGridView.Rows.Add(listWithXY[0][2], listWithXY[1][2], null, F(listWithXY[0][2], listWithXY[1][2]), step \* F(listWithXY[0][2], listWithXY[1][2]));

\_adamsDataGridView.Rows.Add(listWithXY[0][3], listWithXY[1][3], null, F(listWithXY[0][3], listWithXY[1][3]), step \* F(listWithXY[0][3], listWithXY[1][3]));

int iterCondition = 3;

for (int i = 0; i < 3; i++)

{

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

{

\_adamsDataGridView.Rows[k].Cells[i + 5].Value = (double)\_adamsDataGridView.Rows[k + 1].Cells[i + 4].Value - (double)\_adamsDataGridView.Rows[k].Cells[i + 4].Value;

}

iterCondition--;

}

int row = 4;

double deltaY;

double previousY;

for (var x = leftBorder + 4 \* step; x < rightBorder; x += step)

{

\_adamsDataGridView.Rows.Add(x);

deltaY = (double)\_adamsDataGridView.Rows[row - 1].Cells[4].Value +

1.0 / 2.0 \* (double)\_adamsDataGridView.Rows[row - 2].Cells[5].Value +

3.0 / 8.0 \* (double)\_adamsDataGridView.Rows[row - 3].Cells[6].Value;

\_adamsDataGridView.Rows[row - 1].Cells[2].Value = deltaY;

previousY = (double)\_adamsDataGridView.Rows[row - 1].Cells[1].Value;

double y = previousY + deltaY;

\_adamsDataGridView.Rows[row].Cells[1].Value = y;

for (int k = 0; k < 4; k++)

{

\_adamsDataGridView.Rows[row].Cells[3].Value = F(x, y);

\_adamsDataGridView.Rows[row].Cells[4].Value = step \* (double)\_adamsDataGridView.Rows[row].Cells[3].Value;

\_adamsDataGridView.Rows[row - 1].Cells[5].Value = (double)\_adamsDataGridView.Rows[row].Cells[4].Value - (double)\_adamsDataGridView.Rows[row - 1].Cells[4].Value;

\_adamsDataGridView.Rows[row - 2].Cells[6].Value = (double)\_adamsDataGridView.Rows[row - 1].Cells[5].Value - (double)\_adamsDataGridView.Rows[row - 2].Cells[5].Value;

\_adamsDataGridView.Rows[row - 3].Cells[7].Value = (double)\_adamsDataGridView.Rows[row - 2].Cells[6].Value - (double)\_adamsDataGridView.Rows[row - 3].Cells[6].Value;

}

row++;

}

}

private static double F(double x, double y)

{

return 7 \* x + Math.Cos(x \* y); // y'= 7x + cos(xy)

}

}

}