МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет информационных технологий и робототехники***

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 7**

по дисциплине: ”Разработка приложений в визуальных средах ”

# **на тему: *”* Создание приложения, реализующего решение дифференциального уравнения первого порядка*”***

Вариант 1

Выполнил**:** студент группы 10701322 Бородкин Д.В.

Принял**:** доц. Гурский Н. Н.

Минск 2024

**Лабораторная работа № 7.**

**Цель лабораторной работы:** изучить принципы создания класса решения дифференциальных уравнений (ДУ) первого порядка.

**Задание:** Разработать класс решения ДУ первого порядка, а именно:

1.Разработать родительский класс решения ДУ первого порядка методом Эйлера, методом трапеций и методами Рунге-Кутта третьего и четвертого порядка.

2. Разработать класс-наследник, в котором предусмотреть накопление результатов вычислений в динамическом массиве.

3. Разработать класс-наследник, в котором предусмотреть накопление результатов вычислений в списке указателей.

4. Разработать класс-наследник, в котором предусмотреть накопление результатов вычислений в строковом списке.

Головная программа должна предоставлять возможность выбора метода решения дифференциального уравнения, графический вид полученного решения.

**Текст программы:**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Lab\_7

{

// Абстрактный родительский класс

public abstract class Differential\_equations

{

private double a; // Переменная для начала отрезка

private double b; // Переменная для конца отрезка

private double h; // Переменная для шага

public double A

{

get

{

return a;

}

set

{

a = value;

}

}

public double B

{

get

{

return b;

}

set

{

b = value;

}

}

public double H

{

get

{

return h;

}

set

{

h = value;

}

}

// Абстрактный метод класса для функции

public abstract double Function(double x, double y);

// Абстрактный метод класса для выходных данных на график и таблицу

public abstract void Output(int m, Chart chart = null, DataGridView table = null);

// Реализация метода Эйлера

protected double[,] Eiler()

{

int n = Convert.ToInt32(((b - a) / h) + 1);

double[,] array = new double[n, 2]; // двумерный массив типа double размером n строк на 2 столбца.

double x = a, y = 1;

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

{

array[i, 0] = x;

array[i, 1] = y;

y += h \* Function(x, y);

x += h;

}

return array;

}

// Реализация метода Трапеций

protected double[,] Trapeze()

{

int n = Convert.ToInt32(((b - a) / h) + 1);

double[,] array = new double[n, 2]; // двумерный массив типа double размером n строк на 2 столбца.

double x = a, y = 1;

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

{

array[i, 0] = x;

array[i, 1] = y;

y += 0.5 \* h \* (Function(x, y) + Function(x + h, y + h \* Function(x, y)));

x += h;

}

return array;

}

// Реализация метода Рунге-Кутты 3

protected double[,] RK3()

{

int n = Convert.ToInt32(((b - a) / h) + 1);

double[,] array = new double[n, 2]; // двумерный массив типа double размером n строк на 2 столбца.

double x = a, y = 1, k1, k2, k3;

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

{

array[i, 0] = x;

array[i, 1] = y;

k1 = h \* Function(x, y);

k2 = h \* Function(x + 0.5 \* h, y + 0.5 \* k1);

k3 = h \* Function(x + h, y + 2 \* k2 + k1);

y += (k1 + 4 \* k2 + k3) / 6;

x += h;

}

return array;

}

// Реализация метода Рунге-Кутты 4

protected double[,] RK4()

{

int n = Convert.ToInt32(((b - a) / h) + 1);

double[,] array = new double[n, 2]; // двумерный массив типа double размером n строк на 2 столбца.

double x = a, y = 1, k1, k2, k3, k4;

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

{

array[i, 0] = x;

array[i, 1] = y;

k1 = h \* Function(x, y);

k2 = h \* Function(x + 0.5 \* h, y + 0.5 \* k1);

k3 = h \* Function(x + 0.5 \* h, y + 0.5 \* k2);

k4 = h \* Function(x + h, y + k3);

y += (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

x += h;

}

return array;

}

}

// Производный класс

public class MethodOne : Differential\_equations

{

// Переопределение абстрактного метода родительского класса для функции

public override double Function(double x, double y)

{

return 3 \* x - 2 \* y + 5;

}

// Переопределение абстрактного метода родительского класса для выходных данных

public override void Output(int m, Chart chart = null, DataGridView table = null)

{

double[,] array;

if (m == 1)

{

array = Eiler();

}

else if (m == 2)

{

array = Trapeze();

}

else if (m == 3)

{

array = RK3();

}

else if (m == 4)

{

array = RK4();

}

else

{

return;

}

if (chart != null)

{

for (int i = 0; i < array.Length / 2; i++)

{

chart.Series[0].Points.AddXY(array[i, 0], array[i, 1]);

}

}

if (table != null)

{

table.RowCount = 2;

table.ColumnCount = array.Length / 2;

for (int i = 0; i < array.Length / 2; i++)

{

table.Columns[i].HeaderCell.Value = "n-" + (i + 1).ToString();

table.Rows[0].HeaderCell.Value = "X";

table.Rows[1].HeaderCell.Value = "Y";

// Преобразование значений из строки в число с плавающей точкой и вывод с округлением до 4 знаков после запятой

table.Rows[0].Cells[i].Value = Math.Round(double.Parse(array[i, 0].ToString()), 4);

table.Rows[1].Cells[i].Value = Math.Round(double.Parse(array[i, 1].ToString()), 4);

}

}

}

}

// Производный класс

public class MethodTwo : Differential\_equations

{

// Переопределение абстрактного метода родительского класса для функции

public override double Function(double x, double y)

{

return Math.Cosh(x) - y;

}

// Переопределение абстрактного метода родительского класса для выходных данных

public override void Output(int m, Chart chart = null, DataGridView table = null)

{

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

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

double[,] arr;

if (m == 1)

{

arr = Eiler();

}

else if (m == 2)

{

arr = Trapeze();

}

else if (m == 3)

{

arr = RK3();

}

else if (m == 4)

{

arr = RK4();

}

else

{

return;

}

for (int i = 0; i < arr.Length / 2; i++)

{

listx.Add(arr[i, 0]);

listy.Add(arr[i, 1]);

}

if (chart != null)

{

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

{

chart.Series[0].Points.AddXY(listx[i], listy[i]);

}

}

if (table != null)

{

table.RowCount = 2;

table.ColumnCount = listx.Count;

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

{

table.Columns[i].HeaderCell.Value = "n-" + (i + 1).ToString();

table.Rows[0].HeaderCell.Value = "X";

table.Rows[1].HeaderCell.Value = "Y";

// Преобразование значений из строки в число с плавающей точкой и вывод с округлением до 4 знаков после запятой

table.Rows[0].Cells[i].Value = Math.Round(double.Parse(listx[i].ToString()), 4);

table.Rows[1].Cells[i].Value = Math.Round(double.Parse(listy[i].ToString()), 4);

}

}

}

}

// Производный класс

public class MethodThree : Differential\_equations

{

// Переопределение абстрактного метода родительского класса для функции

public override double Function(double x, double y)

{

return Math.Sin(x) - y;

}

// Переопределение абстрактного метода родительского класса для выходных данных

public override void Output(int m, Chart chart = null, DataGridView table = null)

{

List<string> listx = new List<string>();

List<string> listy = new List<string>();

double[,] arr;

if (m == 1)

{

arr = Eiler();

}

else if (m == 2)

{

arr = Trapeze();

}

else if (m == 3)

{

arr = RK3();

}

else if (m == 4)

{

arr = RK4();

}

else

{

return;

}

for (int i = 0; i < arr.Length / 2; i++)

{

listx.Add(arr[i, 0].ToString());

listy.Add(arr[i, 1].ToString());

}

if (chart != null)

{

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

{

chart.Series[0].Points.AddXY(Convert.ToDouble(listx[i]), Convert.ToDouble(listy[i]));

}

}

if (table != null)

{

table.RowCount = 2;

table.ColumnCount = listx.Count;

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

{

table.Columns[i].HeaderCell.Value = "n-" + (i + 1).ToString();

table.Rows[0].HeaderCell.Value = "X";

table.Rows[1].HeaderCell.Value = "Y";

// Преобразование значений из строки в число с плавающей точкой и вывод с округлением до 4 знаков после запятой

table.Rows[0].Cells[i].Value = Math.Round(double.Parse(listx[i].ToString()), 4);

table.Rows[1].Cells[i].Value = Math.Round(double.Parse(listy[i].ToString()), 4);

}

}

}

}

}

**Результаты выполнения программы:**

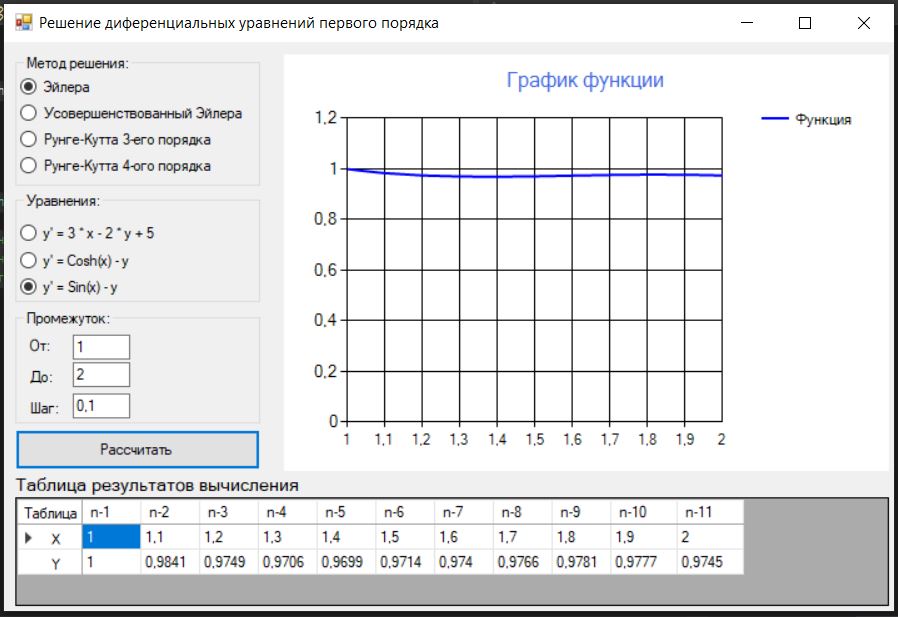


Рисунок 1. Результат работы программы.

**Вывод:** изучил принципы создания класса решения дифференциальных уравнений (ДУ) первого порядка.