МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

**

**ОТЧЁТ**

**по лабораторной работе №6**

**по дисциплине: «** *Вычислительная математика* **»**

**Вариант 2**

Выполнил(а):Проверил:

Студенты гр. *АП-227* *Ландовский В.В.*

*Бузмаков А.И.*

*Шестаков К.Д.*

*Федотов И.В.*

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

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

(подпись) (подпись)

Новосибирск

2024

**Цель работы**

Изучение методов решения дифференциальных уравнений различных типов и их применение для анализа различных физических и инженерных задач, а также освоение компьютерных инструментов для численного решения дифференциальных уравнений и визуализации их решений.

**Постановка задачи**

1. Аналитическими методами или сторонними программными средствами

найти точное решение задачи Коши для дифференциального уравнения y'=f(x,y).

Правая часть уравнения, начальная и конечная точки заданы в таблице 6.1.

1. Разработать программную реализацию вычисления решения задачи Коши

методами, заданными преподавателем. Входной информацией для разработанной

программы должны быть: координаты начальной точки, ордината конечной

точки, шаг интегрирования. Правая часть уравнения и точное решение жестко

задаются в программе. Выходная информация: графики численных решений,

полученных заданными методами, и график точного решения.

1. Провести эксперименты с различными значениями шага: сравнить

результаты с точным решением. Экспериментально подобрать максимальные

значения шага, при которых приближенные решения сходятся к точному,

минимальные значения шага, при котором приближенные решения расходятся.

Подобрать наибольшие значения шага, при которых результаты методов

становятся визуально неотличимыми от точного решения.

**Исходные данные**

|  |  |  |  |
| --- | --- | --- | --- |
| **f(x, y)** | **x0** | **y0** | **xn** |
| sin(x) – 2y | 0 | 5 | 10 |

Методы: метод Эйлера, метод Рунге–Кутты–Мерсона, модифицированный метод Эйлера, метод Адамса 4-го порядка.

**Ход работы**

Сначала вручную посчитали точное решение задачи Коши для уравнения y’ = sin(x) – 2y с начальными точками x0 = 0, y0 = 5 и конечной точкой xn = 10.

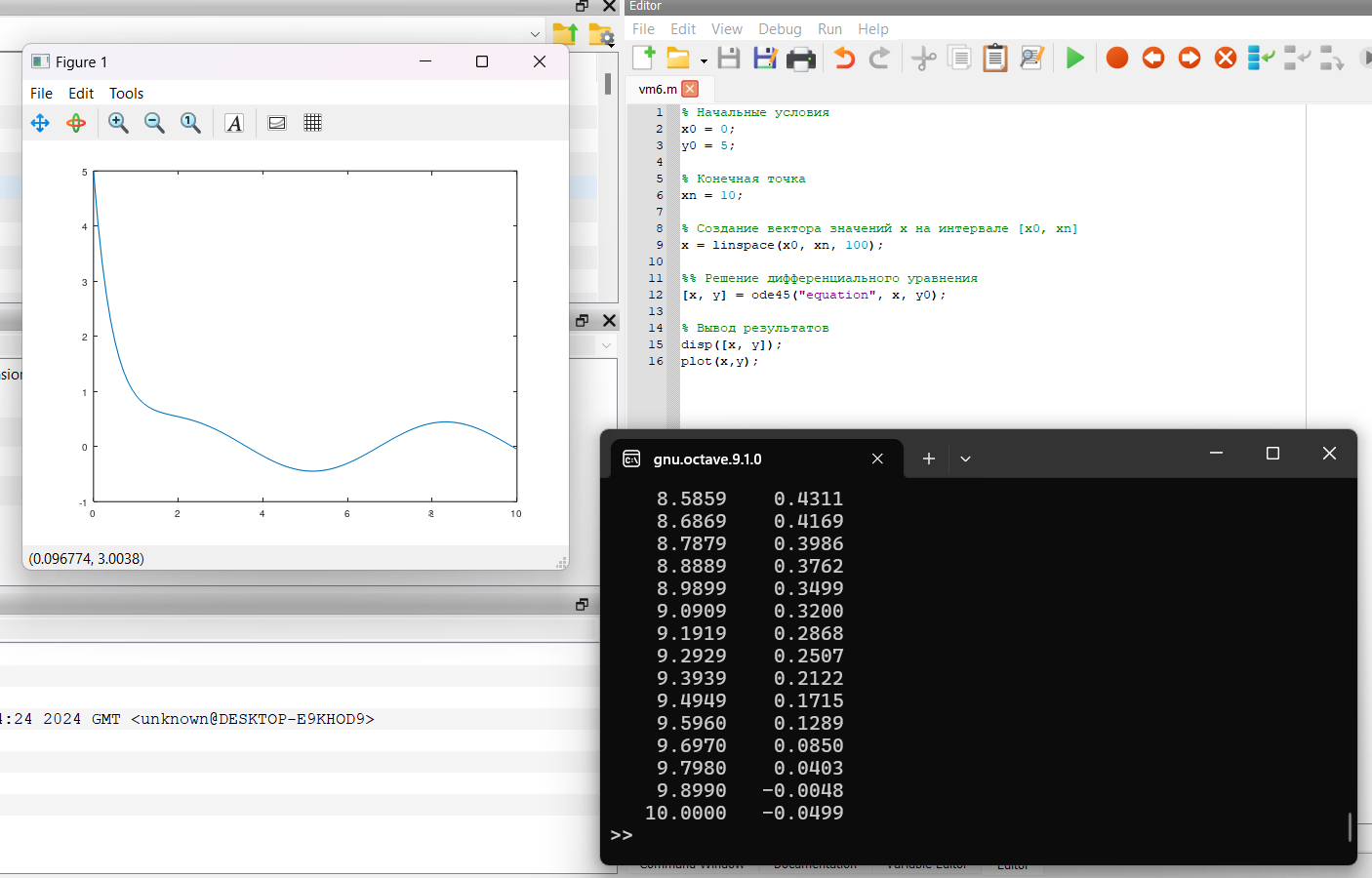


Рис. 1 - Решение диф. уравнения в программе GNU Octave

Попробуем найти точное решение данного уравнения аналитически.

Наше уравнение: y’ = sin(x) – 2y

Перенесем неизвестное в левую часть: y’ + 2y = sin(x)

Определим функции P(x) и Q(x): P(x) = 2, Q(x) = sin(x)

Подставим P(x) = 2 в формулу для интегрирующего множителя u(x):

u(x) = eꭍ2dx

Q(x) = sin(x)

Вычислим интеграл:

u(x) = e2x

Q(x) = sin(x)

Подставим интегрирующий множитель u(x) и функцию Q(x) в формулу общего для общего решения:

Вычислим интеграл:

# ∈R

Получим:

Это формула точного решения для нашего уравнения. За константу С возьмем начальное значение y0, так как с этим значением получается наиболее точное решение, судя по эмпирическому опыту.

Реализуем программу:

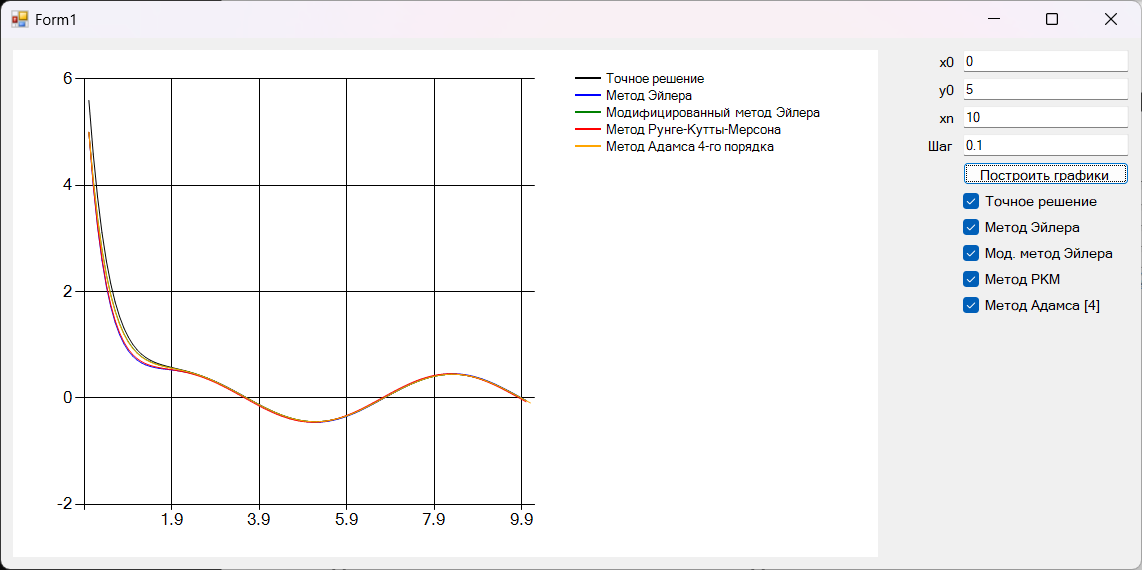


Рис. 2 – Решение уравнения в разработанной программе с теми же значениями

Оба метода решают одно и то же дифференциальное уравнение и, следовательно, дают схожие результаты, хотя их точность немного различается из-за разных численных методов.

Проведем эксперимент с различными значениями шага:

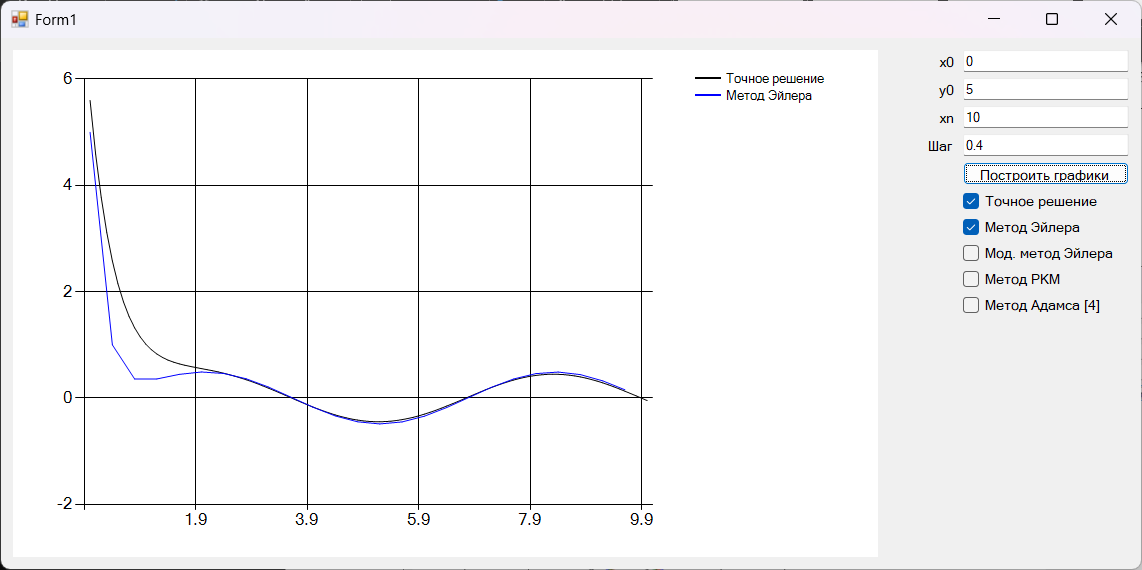


Рис. 3 - Максимальный шаг для метода Эйлера

Максимальным шагом для метода Эйлера, при котором график имеет приемлемый вид, будет 0.4.

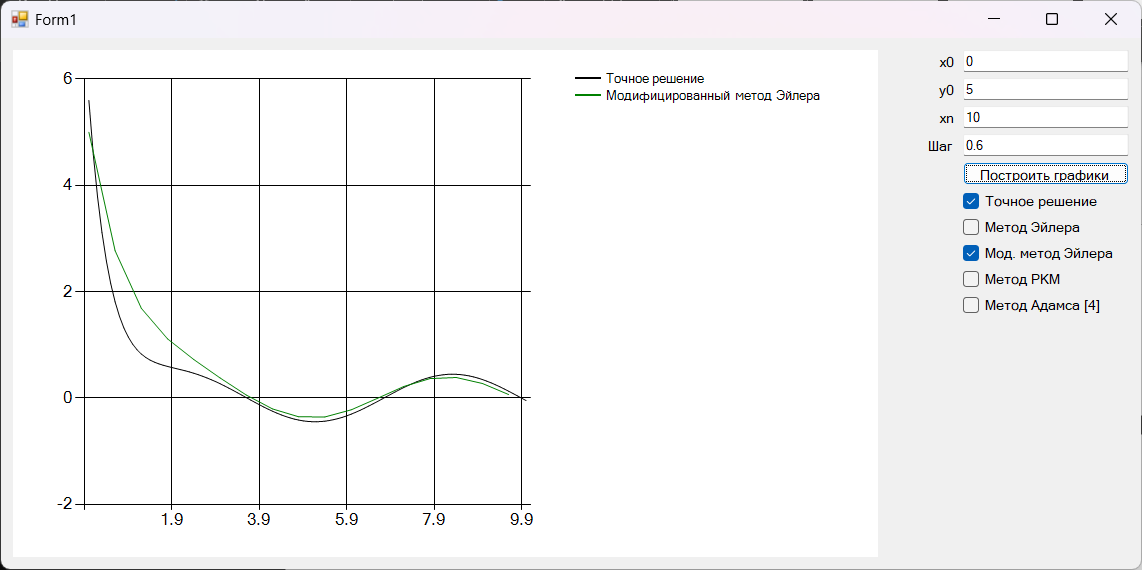


Рис. 4 - Максимальный шаг для модифицированного метода Эйлера

Максимальным шагом для модифицированного метода Эйлера будет 0.6. Дальше график будет «уходить вверх» относительно точного решения.

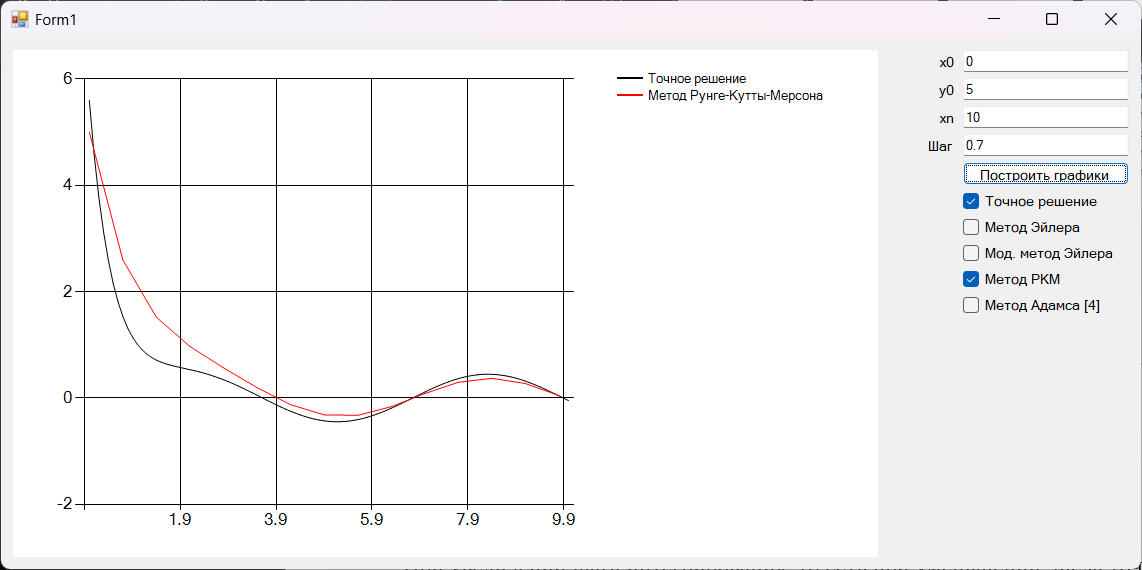


Рис. 5 - Максимальный шаг для метода Рунге-Кутты-Мерсона

Максимальный шаг для метода Рунге-Кутты-Мерсона 0.7. Дальше график будет иметь аналогичное с модифицированным методом Эйлера поведение.

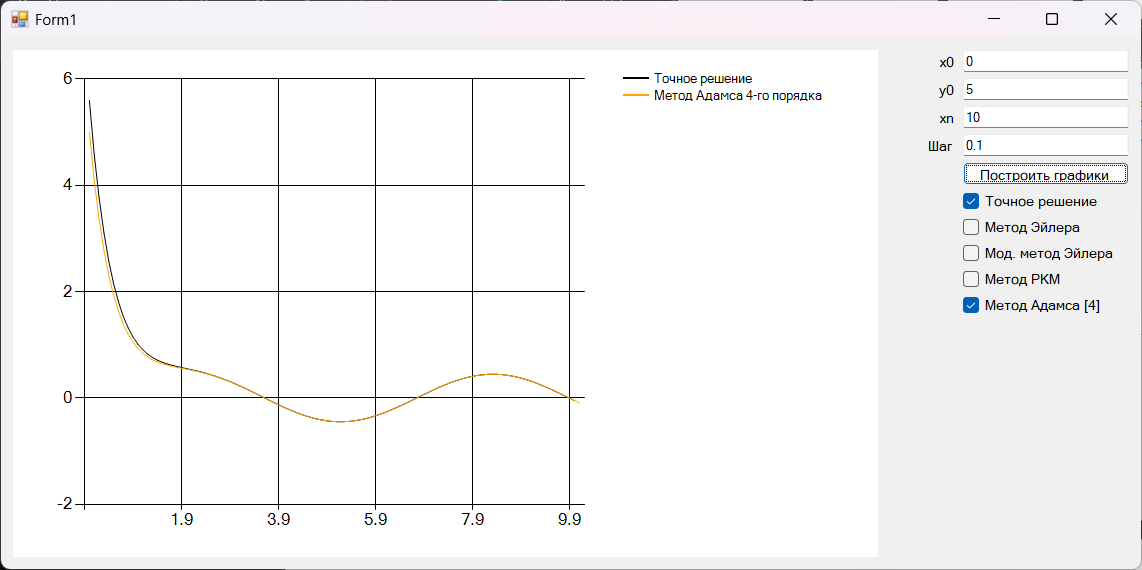


Рис. 6 - Максимальный шаг для метода Адамса 4-го порядка

Максимальным шагом для метода Адамса 4-го порядка будет 0.1. Дальше график будет «ломаться».

При увеличении шага интегрирования, то есть при уменьшении числа точек, в которых вычисляются значения функции, численные методы становятся менее точными. Это приводит к тому, что графики начинают "ломаться" или отклоняться от точного решения.

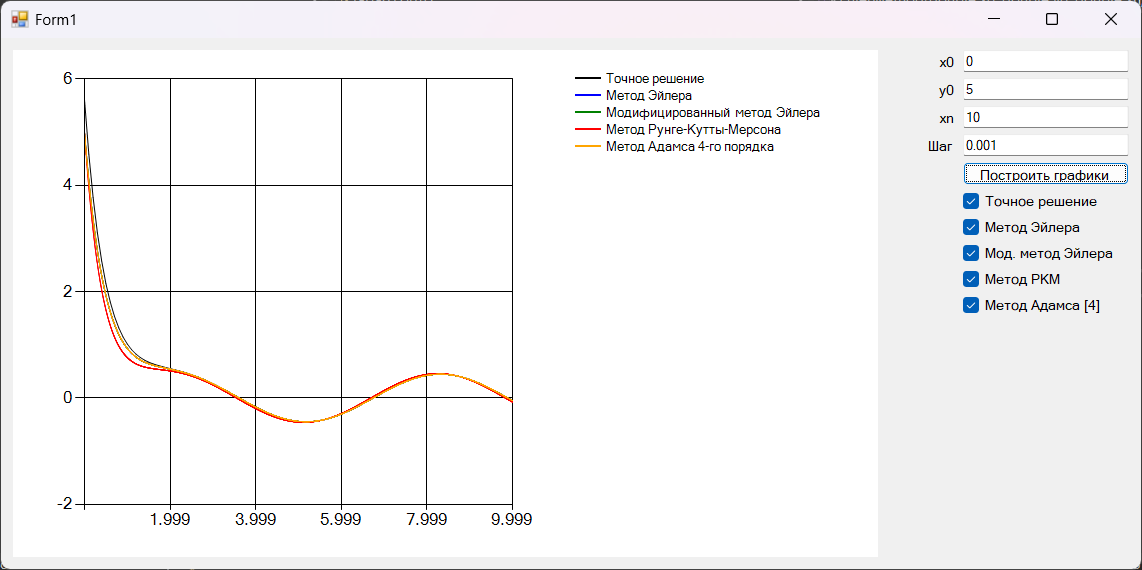


Рис. 7 - Минимальный шаг

Минимальным значением шага для каждого метода будет любое значение > 0, не превышающее ограничения языка программирования.

При уменьшении шага интегрирования, графики численных методов сходятся к графику точного решения. Это происходит потому, что чем меньше шаг, тем ближе приближение численного метода к истинному значению.

**Вывод**

В ходе выполнения данной работы была разработана программа на C#, которая решает задачу Коши для дифференциального уравнения y′=sin(x)−2y с использованием различных численных методов, таких как метод Эйлера, метод Рунге-Кутты-Мерсона, модифицированный метод Эйлера и метод Адамса четвертого порядка. Каждый из этих методов представлен в виде отдельной функции, принимающей начальные условия, координаты начальной и конечной точек, а также шаг интегрирования.

Также было реализовано вычисление точного решения данного дифференциального уравнения. Для этого было найдено программное решение и реализовано соответствующее вычисление в программе.

Сравнение результатов, полученных в разработанной программе, с результатами, полученными в сторонней программе, показало существенное сходство между графиками, полученными обоими методами. Оба решения показывают схожую динамику изменения функции y(x), что подтверждает корректность реализации численных методов в разработанной программе.

Было отмечено, что изменение шага интегрирования существенно влияет на точность численных методов. При увеличении шага интегрирования графики становятся менее точными и могут начать отклоняться от точного решения, в то время как при уменьшении шага интегрирования графики приближаются к точному решению.

Таким образом, данная работа продемонстрировала эффективность численных методов для решения задачи Коши и их зависимость от выбора шага интегрирования.

**Текст разработанной программы**

|  |
| --- |
| Form1.cs |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Windows.Forms;  using System.Windows.Forms.DataVisualization.Charting;  namespace laba6  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  // Дифференциальное уравнение: y' = sin(x) - 2y  private double DifferentialEquation(double x, double y)  {  return Math.Sin(x) - 2 \* y;  }  // Точное решение  private double ExactSolution(double x, double y0)  {  return (2 \* Math.Sin(x) - Math.Cos(x)) / 5 + y0 / Math.Exp(2 \* x);  }  // Метод Эйлера  private void EulerMethod(double x0, double y0, double xn, double h)  {  List<double> xValues = new List<double>();  List<double> yValues = new List<double>();  double x = x0;  double y = y0;  while (x <= xn)  {  xValues.Add(x);  yValues.Add(y);  y += h \* DifferentialEquation(x, y);  x += h;  }  PlotGraph(xValues, yValues, Color.Blue, "Метод Эйлера");  }  // Метод Рунге-Кутты-Мерсона  private void RungeKuttaMersonMethod(double x0, double y0, double xn, double h)  {  List<double> xValues = new List<double>();  List<double> yValues = new List<double>();  double x = x0;  double y = y0;  while (x <= xn)  {  xValues.Add(x);  yValues.Add(y);  double k1 = h \* DifferentialEquation(x, y);  double k2 = h \* DifferentialEquation(x + h / 3, y + k1 / 3);  double k3 = h \* DifferentialEquation(x + h / 3, y + k1 / 3 + k2 / 6);  double k4 = h \* DifferentialEquation(x + h / 2, y + k1 / 3 + k3 / 3);  double k5 = h \* DifferentialEquation(x + h, y + k1 / 2 - k3 + 2 \* k4);  double localErr = Math.Abs((k1 / 6 + (2 \* k4 + k5) / 3) - (k1 / 6 + k4 / 2 + k5 / 3));  if (localErr <= h \* 32)  {  y += k1 / 6 + (2 \* k4 + k5) / 3;  x += h;  }  else  {  h /= 2; // уменьшение шага интегрирования  }  }  PlotGraph(xValues, yValues, Color.Red, "Метод Рунге-Кутты-Мерсона");  }  // Модифицированный метод Эйлера  private void ModifiedEulerMethod(double x0, double y0, double xn, double h)  {  List<double> xValues = new List<double>();  List<double> yValues = new List<double>();  double x = x0;  double y = y0;  while (x <= xn)  {  xValues.Add(x);  yValues.Add(y);  double k1 = h \* DifferentialEquation(x, y);  double k2 = h \* DifferentialEquation(x + h, y + k1);  y += (k1 + k2) / 2;  x += h;  }  PlotGraph(xValues, yValues, Color.Green, "Модифицированный метод Эйлера");  }  // Метод Адамса 4-ого порядка  private void AdamsMethod(double x0, double y0, double xn, double h)  {  List<double> xValues = new List<double>();  List<double> yValues = new List<double>();  List<double> fValues = new List<double>(); // Список для хранения значений производной  // Используем метод Рунге-Кутты для вычисления первых четырех точек  double x = x0;  double y = y0;  double k1, k2, k3, k4;  xValues.Add(x);  yValues.Add(y);  fValues.Add(DifferentialEquation(x, y)); // Значение производной в начальной точке  for (int i = 0; i < 3; i++)  {  k1 = h \* fValues[i];  k2 = h \* DifferentialEquation(x + h / 2, y + k1 / 2);  k3 = h \* DifferentialEquation(x + h / 2, y + k2 / 2);  k4 = h \* DifferentialEquation(x + h, y + k3);  y += (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;  x += h;  xValues.Add(x);  yValues.Add(y);  fValues.Add(DifferentialEquation(x, y)); // Сохраняем значение производной  }  // Применяем метод Адамса для последующих точек  while (x <= xn)  {  double f0 = fValues[fValues.Count - 1];  double f1 = fValues[fValues.Count - 2];  double f2 = fValues[fValues.Count - 3];  double f3 = fValues[fValues.Count - 4];  double nextY = y + h \* (55 \* f0 - 59 \* f1 + 37 \* f2 - 9 \* f3) / 24;  x += h;  xValues.Add(x);  yValues.Add(nextY);  fValues.Add(DifferentialEquation(x, nextY)); // Сохраняем значение производной  y = nextY;  }  PlotGraph(xValues, yValues, Color.Orange, "Метод Адамса 4-го порядка");  }  // Построение графика  private void PlotGraph(List<double> xValues, List<double> yValues, Color color, string methodName)  {  if (xValues.Count != yValues.Count)  {  return;  }  Series series = new Series();  series.ChartType = SeriesChartType.Line;  series.Color = color;  series.Name = methodName;  for (int i = 0; i < xValues.Count; i++)  {  series.Points.AddXY(xValues[i], yValues[i]);  }  chart1.Series.Add(series);  }  private void btnPlot\_Click(object sender, EventArgs e)  {  chart1.Series.Clear();  double x0 = double.Parse(txtX0.Text);  double y0 = double.Parse(txtY0.Text);  double xn = double.Parse(txtXn.Text);  double h = double.Parse(txtStepSize.Text);  // График точного решения  List<double> exactXValues = new List<double>();  List<double> exactYValues = new List<double>();  for (double x = x0; x <= xn; x += 0.1)  {  exactXValues.Add(x);  exactYValues.Add(ExactSolution(x, y0));  }  if (checkBox1.Checked)  {  PlotGraph(exactXValues, exactYValues, Color.Black, "Точное решение");  }  if (checkBox2.Checked)  {  EulerMethod(x0, y0, xn, h);  }  if (checkBox3.Checked)  {  ModifiedEulerMethod(x0, y0, xn, h);  }  if (checkBox4.Checked)  {  RungeKuttaMersonMethod(x0, y0, xn, h);  }  if (checkBox5.Checked)  {  AdamsMethod(x0, y0, xn, h);  }  }  }  } |