МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по лабораторной работе №6**

**по теме:**

**Решение дифференциальных уравнений**

**по дисциплине:** Вычислительная математика.

Вариант №4

Выполнили работу:

Студенты гр. АП-127, АВТФ

Болотников М. С,

Чайка П.АЛилия Александровна

«16» мая 2023 г.

Проверил работу:

Ландовский В.В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

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

(подпись)

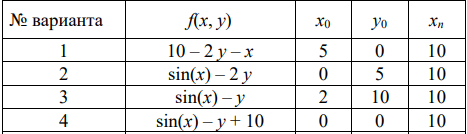
### Задание

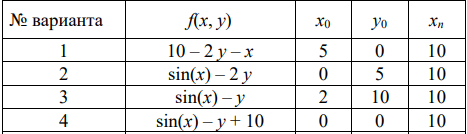
### 1. Аналитическими методами или сторонними программными средствами найти точное решение задачи Коши для дифференциального уравнения y'=f(x,y). Правая часть уравнения, начальная и конечная точки заданы в таблице.

### 2. 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.

### 3. 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





Методы для 4 варианта:

Метод Эйлера;

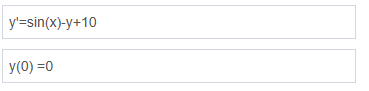
Метод Рунге-Кутты-Мерсона;

Исправленный Эйлера (из таблицы);

Адамса 2-го порядка (из таблицы).

**Нахождение точного решения хадачи Коши**

Используя сайт https://math24.biz/, нашли точное решение задачи Коши для дифференциального уравнения y'=f(x,y)





**Листинг программы с комментариями**

|  |
| --- |
| Содержимое файла “ Graphics.cs” |
| 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;  using System.Windows.Forms.DataVisualization.Charting;  using VM6;  namespace CountMathWf6  {  public partial class Graphics : Form  {  // количество точек  int n;  // значения задаваемые в программе  double x0,xn,H,y0;  // Получение всех X  List<double> X\_RES = new List<double>();  List<double> y\_Eyler = new List<double>();  // построение графика по исправленному методу Эйлера  private void butFixEiler\_Click(object sender, EventArgs e)  {  Count();  y\_Eyler = new List<double>();  Eiler(ref y\_Eyler);  List<double> y\_Isp\_Eyler = new List<double>();  y\_Isp\_Eyler.Add(y0);  for (int i = 1; x0 <= xn; x0 += H, i++)  {  y\_Isp\_Eyler = Realiz.IspEyler(x0, y\_Isp\_Eyler, y\_Eyler, i, H);  }  x0 = double.Parse(tbx0.Text);  // Левая граница.  double a = x0;  // Правая граница.  double b = xn;  // Шаг графика.  double x, y;  this.chartGraph.Series[2].Points.Clear();  x = a;  int j = 0;  while (x <= b)  {  y = y\_Isp\_Eyler[j];  this.chartGraph.Series[2].Points.AddXY(x, y);  x += H;  j++;  }  }  public Graphics()  {  InitializeComponent();  }  private void butClear\_Click(object sender, EventArgs e)  {  for (int i = 0; i < chartGraph.Series.Count; i++)  {  chartGraph.Series[i].Points.Clear();  }  }  // построение графика по методу Рунге - Кутты - Мерсона  private void butRKM\_Click(object sender, EventArgs e)  {  Count();  List<double> y\_RKM = new List<double>();  y\_RKM.Add(y0);  y\_RKM = Realiz.RKM(y\_RKM, ref X\_RES, H, n);  // Левая граница.  double a = x0;  // Правая граница.  double b = xn;  // Шаг графика.  double x, y;  this.chartGraph.Series[3].Points.Clear();  int j = 0;  while (j < X\_RES.Count)  {  x = X\_RES[j];  y = y\_RKM[j];  this.chartGraph.Series[3].Points.AddXY(x, y);  j++;  }  }  // построение графика по методу Адамса 2-го порядка  private void butAdamson\_Click(object sender, EventArgs e)  {  Count();  y\_Eyler = new List<double>();  Eiler(ref y\_Eyler);  List<double> y\_Adams = new List<double>();  y\_Adams = Realiz.Adams1(y\_Adams, y\_Eyler, H, y0, x0);  for (int i = 2; x0 <= xn; x0 += H, i++)  {  y\_Adams = Realiz.Adams2(x0, y\_Adams, H, i);  }  x0 = double.Parse(tbx0.Text);  // Левая граница.  double a = x0;  // Правая граница.  double b = xn;  // Шаг графика.  double x, y;  this.chartGraph.Series[4].Points.Clear();  x = a;  int j = 0;  while (x <= b)  {  y = y\_Adams[j];  this.chartGraph.Series[4].Points.AddXY(x, y);  x += H;  j++;  }  }  //построение графика точного решения  private void butClearSolve\_Click(object sender, EventArgs e)  {  Count();  List<double> Y\_toch = new List<double>();  for (; x0 <= xn; x0 += H)  {  Y\_toch = Realiz.f(Y\_toch, x0);  }  x0 = double.Parse(tbx0.Text);  // Левая граница.  double a = x0;  // Правая граница.  double b = xn;  // Шаг графика.  double x, y;  this.chartGraph.Series[0].Points.Clear();  x = a;  int j = 0;  while (x <= b )  {  y = Y\_toch[j];  this.chartGraph.Series[0].Points.AddXY(x, y);  x += H;  j++;  }  }  // Получение входных данных  void Count()  {  x0 = double.Parse(tbx0.Text);  xn = double.Parse(tbxn.Text);  H = double.Parse(tbh.Text);  X\_RES = new List<double>();  y0 = double.Parse(tby0.Text);  n = 0;  for (; x0 <= xn; x0 += H)  {  n++;  }  x0 = double.Parse(tbx0.Text);  for (; x0 <= xn; x0 += H)  {  X\_RES.Add(x0);  }  x0 = double.Parse(tbx0.Text);  }  // Расчет по методу Эйлера.  void Eiler(ref List<double> yEiler)  {  yEiler.Add(y0);  for (int i = 1; x0 <= xn; x0 += H, i++)  {  yEiler = Realiz.Eyler(x0, yEiler, H);  }  x0 = double.Parse(tbx0.Text);  }  // построение графика по методу Эйлера  private void butEiler\_Click(object sender, EventArgs e)  {  Count();  y\_Eyler = new List<double>();  Eiler(ref y\_Eyler);  // Левая граница.  double a = x0;  // Правая граница.  double b = xn;  // Шаг графика.  double x, y;  this.chartGraph.Series[1].Points.Clear();  x = a;  int j = 0;  while (x <= b)  {  y = y\_Eyler[j];  this.chartGraph.Series[1].Points.AddXY(x, y);  x += H;  j++;  }  }  }  } |
| Содержимое файла “ Functions.cs” |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Linq;  using System.Runtime.CompilerServices;  using System.Security.Cryptography.X509Certificates;  using System.Windows;  namespace VM6  {  public static class Realiz  {  const double eps = 0.001;  // Точное решение  public static List<double> f(List<double> y, double x)  {  y.Add(0.5 \* (-19 \* Math.Exp(-x) - Math.Cos(x) + Math.Sin(x) + 20));  return y;  }  // Метод Эйлера  public static List<double> Eyler(double x, List<double> y, double h)  {  y.Add(y.Last() + h \* (Math.Sin(x) - y.Last() + 10));  return y;  }  // Метод Рунге-Кутты-Мерсона  public static List<double> RKM(List<double> y, ref List<double> X\_RES, double h, int n)  {  int i = 0;  while (i < n)  {  if ( i != 0 ) X\_RES[i] = X\_RES[i-1] + h;  double k1, k2, k3, k4, k5;  k1 = h \* (Math.Sin(X\_RES[i]) - y[i] + 10);  k2 = h \* (Math.Sin(X\_RES[i] + (h / 3)) - (y[i] + k1 / 3) + 10);  k3 = h \* (Math.Sin(X\_RES[i] + (h / 3)) - (y[i] + (k1 + k2) / 6) + 10);  k4 = h \* (Math.Sin(X\_RES[i] + (h / 2)) - (y[i] + (k1 + 3 \* k3) / 8) + 10);  k5 = h \* (Math.Sin(X\_RES[i] + h) - (y[i] + (k1 - 3 \* k3 + 4 \* k4) / 2) + 10);  double sigma = (2 \* k1 - 9 \* k3 + 8 \* k4 - k5) / 30;  if (Math.Abs(sigma) >= eps)  {  h = h / 2;  continue;  }  y.Add(y[i] + (k1 + 4 \* k2 + k5) / 6);  if (Math.Abs(sigma) <= eps / 32)  {  h = h \* 2;  }  i++;  }  return y;  }  // Метод Эйлера  public static List<double> IspEyler(double x, List<double> y, List<double> y\_Eyler, int i, double h)  {  y.Add(y.Last() + h \* ((Math.Sin(x) - y.Last() + 10) + (Math.Sin(x + h) - y\_Eyler[i] + 10)) / 2);  return y;  }  // Метод Адамса(первые 2 приближения)  static double F0, F1;  public static List<double> Adams1(List<double> y, List<double> y\_Eyler, double h, double y0, double x0)  {  F0 = y0 + h \* ((Math.Sin(x0) - y0 + 10) + (Math.Sin(x0 + h) - y\_Eyler[1] + 10)) / 2;  F1 = F0 + h \* ((Math.Sin(x0 + h) - F0 + 10) + (Math.Sin(x0 + 2 \* h) - y\_Eyler[2] + 10)) / 2;  y.Add(F0);  y.Add(F1);  return y;  }  // Метод Адамса  public static List<double> Adams2(double x, List<double> y, double h, int i)  {  y.Add(y.Last() + h \* (3 \* F1 - F0) / 2);  F0 = F1;  F1 = Math.Sin(x + h) - y.Last() + 10;  return y;  }  }  } |

### Результаты экспериментов

### https://sun9-53.userapi.com/impg/uEknBBM1-CG7CZAZgBIorBUuhzNoEaFd8zBb-w/QoxK_CRF2n8.jpg?size=1280x614&quality=96&sign=0b39f3755c6b4117bfdc584330e6cfd6&type=album

### https://sun4-19.userapi.com/impg/nmb1N4DMpgIy_RKW-iBVM7raowW1FIWq1N2gew/a8pnmJinBUc.jpg?size=1280x617&quality=96&sign=3153fe16dbf133e95395f4e8edbf06f4&type=album

### https://sun4-20.userapi.com/impg/qgaikRmL8H1His0_7TvOPUjyE5z3FtvgmdUxUQ/t-5VfEdcekE.jpg?size=1280x616&quality=96&sign=8e466660dfb84a10b135f65e6294efa4&type=album

### Анализ результатов

Анализ шага для метода Эйлера:

* Наибольшее значение шага, при котором результаты визуально не отличаются: 0.01

### 

Анализ шага для исправленного метода Эйлера:

* Наибольшее значение шага, при котором результаты визуально не отличаются: 0.01

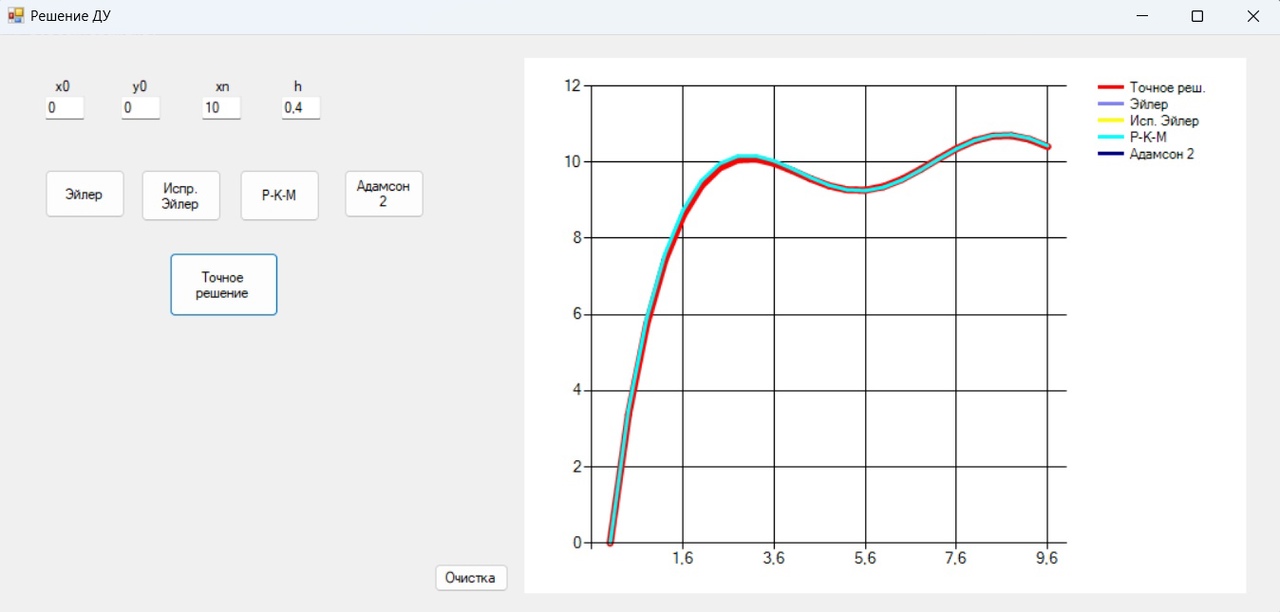
### 

### При увеличении шага метод Эйлера расходится быстрее, чем метод исправленного метода Эйлера:

### 

Анализ шага для исправленного метода Рунге-Кутты-Мерсона:

* Наибольшее значение шага, при котором результаты визуально не отличаются: 0.4



Анализ шага для исправленного метода Адамсона:

* Наибольшее значение шага, при котором результаты визуально не отличаются: 0.01

### 