МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«**Кубанский государственный университет»

Факультет компьютерных технологий и прикладной математики

Кафедра математического моделирования

**Отчет о научно-исследовательской работе**

(практике по получению первичных навыков научно-исследовательской работы)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Е Гиренко

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Курс 2

Руководитель учебной практики

канд. физ.-мат. наук, доцент кафедры

математического моделирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Е. Рубцов

Краснодар

2022г.

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc104308234)

[2 Описание метода Эйлера 4](#_Toc104308235)

[3 Аналитическое решение задачи Коши 5](#_Toc104308236)

[4 Описание программы для численного решения задачи Коши 8](#_Toc104308237)

[5 Результаты вычислений 10](#_Toc104308238)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 13](#_Toc104308239)

[ПРИЛОЖЕНИЕ. Текст программы 14](#_Toc104308240)

**1 Постановка задачи**

Дано дифференциальное уравнение:

, .

* Методом ломаных Эйлера получить приближенное решение задачи Коши для заданного дифференциального уравнения. Начальное условие .
* Вычисления произвести при помощи программы, разработанной лично Вами на языке высокого уровня для различных значений N (например, при N=5, 20, 100). В программе предусмотреть ввод N.
* Получить аналитически точное решение задачи Коши.
* В одной системе координат построить графики точного и приближенного решений. Вычислить максимальную невязку (наибольшую по абсолютной величине разность между точными приближенным решениями для различных значений ).

**2 Описание метода Эйлера**

Метод Эйлера – простейший численный метод решения систем обыкновенных дифференциальных уравнений. Впервые описан Леонардом Эйлером в 1768 году в работе «Интегральное исчисление». Метод Эйлера является явным, одношаговым методом первого порядка точности. Он основан на аппроксимации интегральной кривой кусочно-линейной функцией, так называемой ломаной Эйлера.

Пусть дана задача Коши для уравнения первого порядка:

, ,

где функция определена на некоторой области . Решение ищется на интервале . На этом интервале введем узлы: . Приближенное решение в узлах , которое обозначим через , определяется по формуле:

Эти формулы непосредственно обобщаются на случай систем обыкновенных дифференциальных уравнений.

Для данного в постановке задачи дифференциального уравнения уравнение численного решения имеет вид:

**3 Аналитическое решение задачи Коши**

Решим дифференциальное уравнение и задачу Коши для него:

,

Умножим на :

Пусть , тогда . Сделаем замену и получим:

Запишем соответствующее ЛОДУ и преобразуем для интегрирования:

Умножим обе части на :

Проинтегрируем обе части:

,

где

– неизвестная функция.

Потенцируем:

.

Продифференцируем полученное выражение:

.

Подставим в изначальное ЛНДУ:

Сократим одинаковые части:

Преобразуем, умножим обе части на и проинтегрируем:

Заметим, что . Воспользуемся методом внесения под знак дифференциала и получим результат интегрирования:

где

– константа.

Получим зависимость :

Сделаем обратную замену :

При подстановке начального условия во второе выражение становиться ясно, что оно не имеет решений. Получим значение для первого выражения. Подставим

Получим Откуда следует, что Подставим в получим ответ –

**4 Описание программы для численного решения задачи Коши**

Программа реализована с использованием среды разработки Visual Studio 2019. В программе задействован набор библиотек “Windows Forms” для отображения пользовательского интерфейса, а также библиотека ZedGraph для отображения графиков функций. Работа программы начинается с загрузки всех элементов формы и рендеринга осей. Для дальнейшей работы пользователю необходимо ввести в специально указанное поле ввести значение n, отвечающее за количество точек, которые будут нарисованы для функции.

В программе реализованы функции f и realF. Каждой на вход подаётся значение x типа double (точка x функции) и на выходе возвращается значение y типа double (точка y(x) функции). В realF записан результат аналитического решения задачи Коши, в то время как в f записан изначальный вид данной функции. Также в программе реализованы вспомогательные функции, которые отвечают за взаимодействие программы и пользователя.

Основной функцией в программе является функции built, которая при вызове вычисляет приближённое и точное значения данной функции, а затем точка за точкой строит изображение графиков.

Метод Эйлера и вычисление максимальной невязки реализованы следующим образом:

x = 0;

while (x < 0 + h \* n)

{

list.Add(x, y);

// вычисление максимальной невязки

nowE = Math.Abs(y - realF(x));

if (nowE > maxE)

maxE = nowE;

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

x += h;

}

x – текущее значение функции по оси X, y – текущее значение функции по оси Y, nowE – нынешняя невязка, maxE – максимальная невязка, list – список точек, который необходим для построения графика, h – шаг, с которым идёт приращение, который вычисляется в соответствии с формулой h = (maxX - minX) / (n-1), где maxX = 1, minX = 0 (границы, данные в задаче), а n – количество отрезков, введённое пользователем. Все эти переменные объявлены ранее в программе.

Цикл при первой итерации вставляет в список list точки из начального условия, а также записывает первое полученное значение невязки, вычисленное как модуль разности точной функции и полученной методом Эйлера. Далее пока условие цикла выполняется в соответствии с методом Эйлера находит остальные точки функции и также вставляет в список list, после чего вычисляет невязку для этой точки и, если значение больше, записывает это значение в переменную.

Построение точного графика данной функции реализовано следующим образом:   
 while (x < 0+h1\*1000D)

{

reallist.Add(x, realF(x));

x += h1;

}  
 Все используемые здесь переменные имеют такой же смысл, что и в реализации метода Эйлера. Точки оси X считаются суммированием шага h. Точки оси Y считаются вызовом функции realF.

**5 Результаты вычислений**

На рисунках 2–6 изображены графики точного и приблизительного решения данной задачи Каши при

Точное изображение графика изображено на рисунке 1.

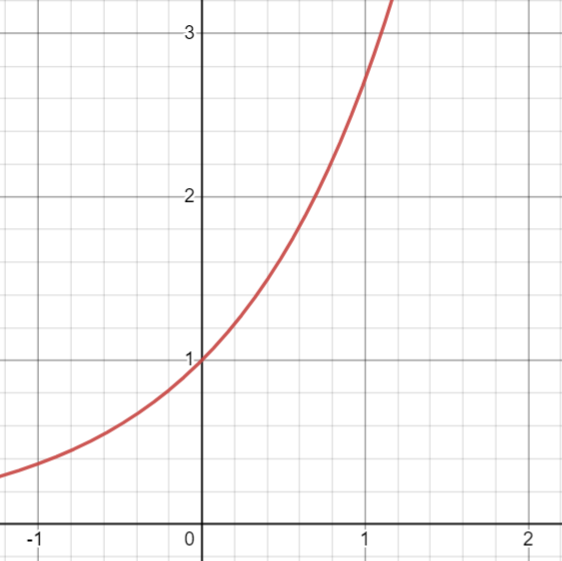


Рисунок 1 – График полученной функции, построенный с помощью Desmos

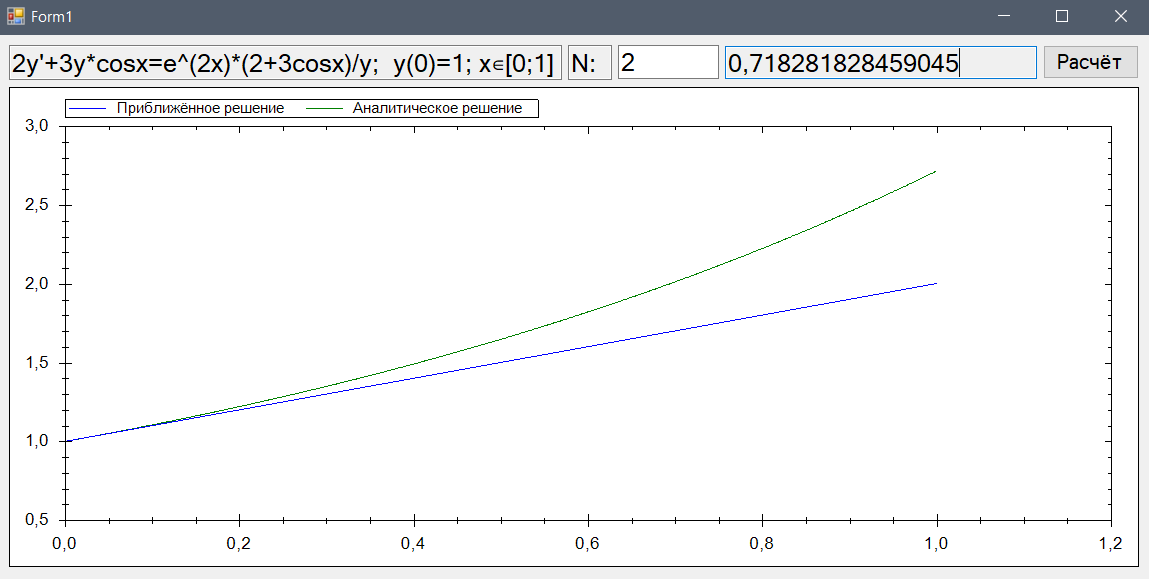


Рисунок 2 – Результат и исходная функция для

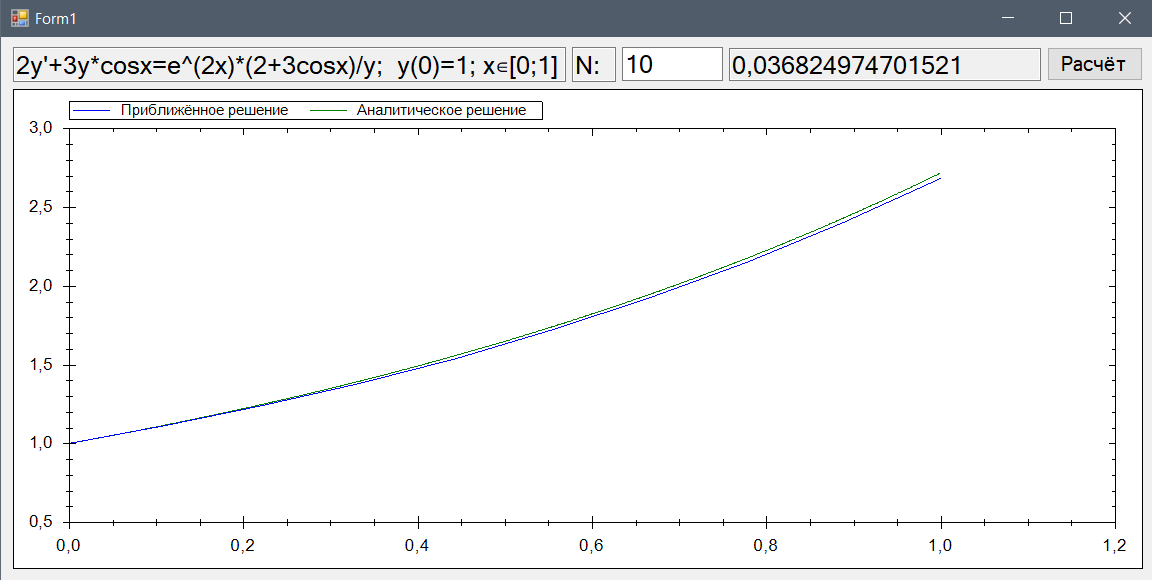


Рисунок 3 – Результат и исходная функция для

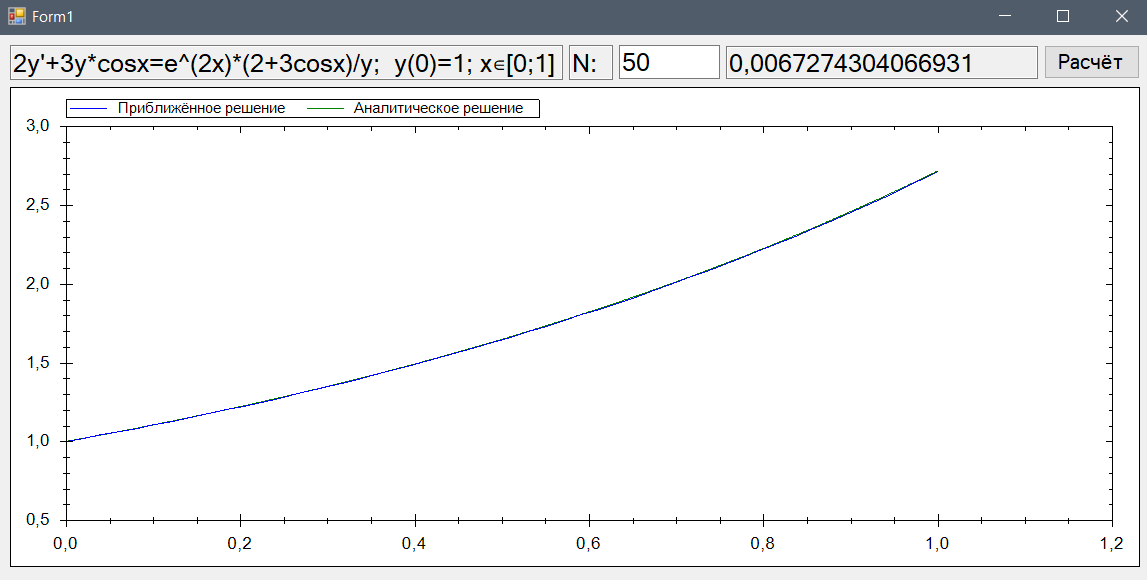


Рисунок 4 – Результат и исходная функция для

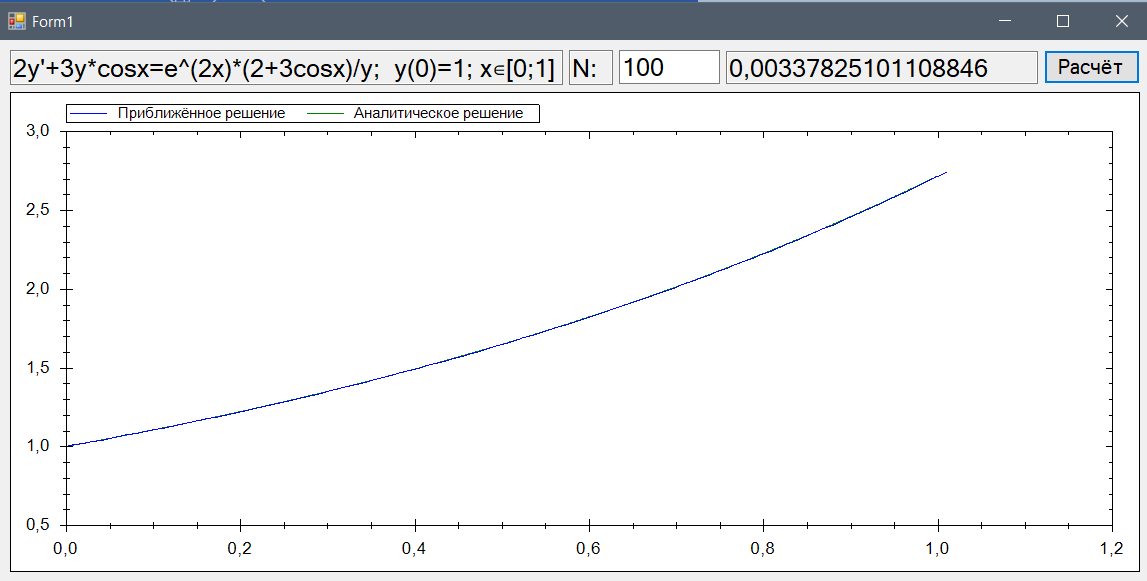


Рисунок 5 – Результат и исходная функция для

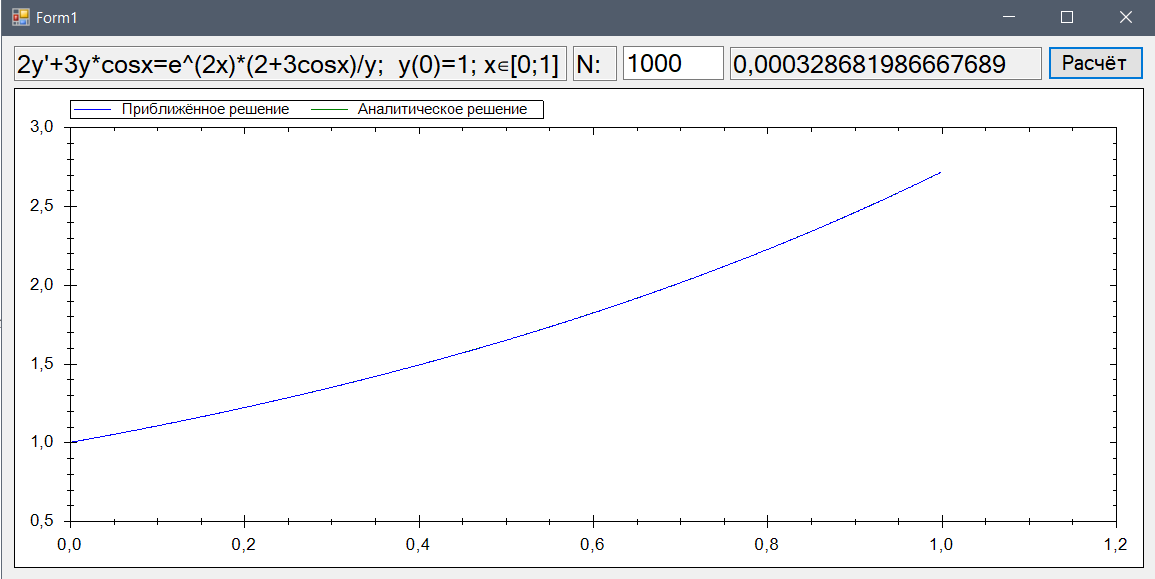


Рисунок 6 – Результат и исходная функция для

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пантелеев, А.В. Обыкновенные дифференциальные уравнения в примерах и задачах / А.В. Пантелеев. – М.: Высшая школа, 2001. – 376 c. ISBN 5-06-004134-4
2. Усов, В.А. Swift. Основы разработки приложений под iOS, iPadOS и macOS / Усов В.А – СПб.: Питер, 2021 – 544 c. –ISBN 978-5-4461-1796-3
3. Несен, А.В. Microsoft Word 2010. От новичка к профессионалу / А.В. Несен. – М.: Солон-Пресс, 2016. – 320 c. ISBN 978-5-91359-096-1

ПРИЛОЖЕНИЕ

Текст программы

using System;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using ZedGraph;

namespace metodEiler

{

public partial class Form1 : Form

{

double n, e;

// условия Каши и задание отрезка

const double x0=0, y0=1, minX = 0, maxX = 1;

// дифференциальное уравнение

static public double f(double x, double y)

{

return Math.Exp(2 \* x) \* (2 + 3 \* Math.Cos(x)) / (2 \* y) - 3 \* y \* Math.Cos(x) / 2;

}

//точное решение задачи Коши

static public double realF(double x)

{

return Math.Exp(x);

}

// метод очистки и настройки координатной плоскости

private void Clear(ZedGraphControl Zed\_GraphControl)

{

zedGraphControl1.GraphPane.CurveList.Clear();

zedGraphControl1.GraphPane.GraphObjList.Clear();

zedGraphControl1.GraphPane.XAxis.Type = AxisType.Linear;

zedGraphControl1.GraphPane.XAxis.Scale.TextLabels = null;

zedGraphControl1.GraphPane.XAxis.MajorGrid.IsVisible = false;

zedGraphControl1.GraphPane.YAxis.MajorGrid.IsVisible = false;

zedGraphControl1.GraphPane.YAxis.MinorGrid.IsVisible = false;

zedGraphControl1.GraphPane.XAxis.MinorGrid.IsVisible = false;

zedGraphControl1.GraphPane.XAxis.Title.IsVisible = false;

zedGraphControl1.GraphPane.YAxis.Title.IsVisible = false;

zedGraphControl1.GraphPane.Title.IsVisible = false;

zedGraphControl1.RestoreScale(zedGraphControl1.GraphPane);

zedGraphControl1.AxisChange();

zedGraphControl1.Invalidate();

}

// построение графиков

public void build(ZedGraphControl Zed\_GraphControl)

{

PointPairList list = new PointPairList();

PointPairList reallist = new PointPairList();

double h, h1, x=x0, y=y0;

double nowE, maxE = 0;

h = (maxX - minX) / (n-1);

h1 = (double)(maxX - minX) / 1000D;

// нахождение точек точной функции

while (x < x0+h1\*1000D)

{

reallist.Add(x, realF(x));

x += h1;

}

// нахождение точек приближённой функции

x = x0;

while (x < x0 + h \* n)

{

list.Add(x, y);

// вычисление максимальной невязки

nowE = Math.Abs(y - realF(x));

if (nowE > maxE)

maxE = nowE;

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

x += h;

}

textBox3.Text = Convert.ToString(maxE);

// отрисовка графиков функций

GraphPane my\_Pane = Zed\_GraphControl.GraphPane;

LineItem myCircle = my\_Pane.AddCurve("Приближённое решение", list, Color.Blue, SymbolType.None);

LineItem myCircle2 = my\_Pane.AddCurve("Аналитическое решение", reallist, Color.Green, SymbolType.None);

zedGraphControl1.AxisChange();

zedGraphControl1.Invalidate();

}

public Form1()

{

InitializeComponent();

Clear(zedGraphControl1);

}

// при нажатии на кнопку расчёт

private void button1\_Click(object sender, EventArgs ee)

{

if (textBox6.Text != "")

try

{ n = Convert.ToDouble(textBox6.Text); }

catch { return; }

Clear(zedGraphControl1);

build(zedGraphControl1);

}

}

}