Дисциплина: Численные методы

Лабораторное задание №2

Отчет

Тема: «Численное решение задачи Коши для обыкновенных

дифференциальных уравнений 1-го порядка методами Рунге-Кутта»

Выполнил:

студент 3 курса 8 группы

Крутько А.С.

Проверила:

преподаватель

Махинова О.А.

[Задача 1 3](#_Toc101009838)

[Постановка задачи 3](#_Toc101009839)

[Используемые формулы 4](#_Toc101009840)

[Реализация формул в коде 5](#_Toc101009841)

[Численные эксперименты 7](#_Toc101009842)

[Задача 2 8](#_Toc101009843)

[Постановка задачи 8](#_Toc101009844)

[Используемые формулы 9](#_Toc101009845)

[Реализация формул в коде 10](#_Toc101009846)

[Численные эксперименты 12](#_Toc101009847)

[Вывод 13](#_Toc101009848)

Задача 1

Постановка задачи

Найти численное решение задачи Коши для ОДУ 1-го порядка вида (\*)

Методом типа Рунге-Кутта третьего порядка с заданной точностью .

Используемые формулы

Формулы, использованные для получения решений задачи Коши:

Формулы вида

где

Образуют три семейства формул типа Рунге-Кутта третьего порядка. Одно семейство – двухпараметрическое со свободными параметрами

Где , определяются из системы двух линейных уравнений:

Коэффициенты вычисляются простым пересчетом:

В случае получаются следующие формулы:

Реализация формул в коде

// K1

**private** **double** K1(**double** x, **double** y, **double** h)

=> h \* CurrentFunc(x, y);

// К2

**private** **double** K2(**double** x, **double** y, **double** h, **double** k1)

=> h \* CurrentFunc(x + 1.0 / 3 \* h, y + 1.0 / 3 \* k1);

// K3

**private** **double** K3(**double** x, **double** y, **double** h, **double** k2)

=> h \* CurrentFunc(x + 2.0 / 3 \* h, y + 2.0 / 3 \* k2);

**public** **int** GetResult()

{

// Значения текущей погрешности и предыдущей для проверки на уменьшении погрешности при уменьшении шага

**double** currentError = 10.0;

// Решаемо ли уравнение методом Рунге-Кутта или нет

// Размерности

**int** n = N0;

**int** n2 = 2 \* N0;

**double** tempH = 0.0;

**double** temp2H = 0.0;

**do**

{

// Задаем равномерную сетку

GridN = BuildGrid(n, **ref** tempH);

Grid2N = BuildGrid(n2, **ref** temp2H);

HN = tempH;

H2N = temp2H;

**double** previousError = currentError;

ResultN = SolveCauchy(n, GridN, HN);

Result2N = SolveCauchy(n2, Grid2N, H2N);

**bool** isSolved = ResultN.Length != 0 || Result2N.Length != 0;

ResultLast2Iterations.Enqueue(ResultN);

**while** (ResultLast2Iterations.Count > 3)

{

ResultLast2Iterations.Dequeue();

}

currentError = FindError();

// Удваиваем количество точек на сетке

n \*= 2;

n2 \*= 2;

// Процесс решения прекращен, т.к. шаг стал меньше возможного

**if** (MinimalH > HN) **return** 2;

// Процесс решения прекращен, т.к. с уменьшением шага погрешность не уменьшается

**if** (currentError > previousError) **return** 1;

// Решение не получено, двухсторонний метод Рунге-Кутта с данным начальным шагом не применим

**if** (!isSolved) **return** 4;

} **while** (currentError > Epsilon0);

// Завершение в соответствии с назначенным условием о достижении заданной точности

**return** 0;

}

// Метод для построения равномерной сетки с количеством подотрезков n

**private** **double**[] BuildGrid(**int** n, **ref** **double** h)

{

**return**

Distribution.EvenNodes(A, B, n, **ref** h);

}

// Получение решения задачи Коши методом Рунге-Кутта порядка p

**private** **double**[] SolveCauchy(**int** n, IReadOnlyList<**double**> x, **double** h)

{

**double**[] y = **new** **double**[n];

y[0] = Func0;

**for** (**var** index = 1; index < n; index++)

{

**double** k1 = K1(x[index - 1], y[index - 1], h);

**double** k2 = K2(x[index - 1], y[index - 1], h, k1);

**double** k3 = K3(x[index - 1], y[index - 1], h, k2);

y[index] = y[index - 1] + 1.0 / 4.0 \* (k1 + 3.0 \* k3);

}

**return** y;

}

Численные эксперименты

При проведении численных экспериментов все значения для задачи Коши для системы ОДУ будут указаны на скриншотах программы.

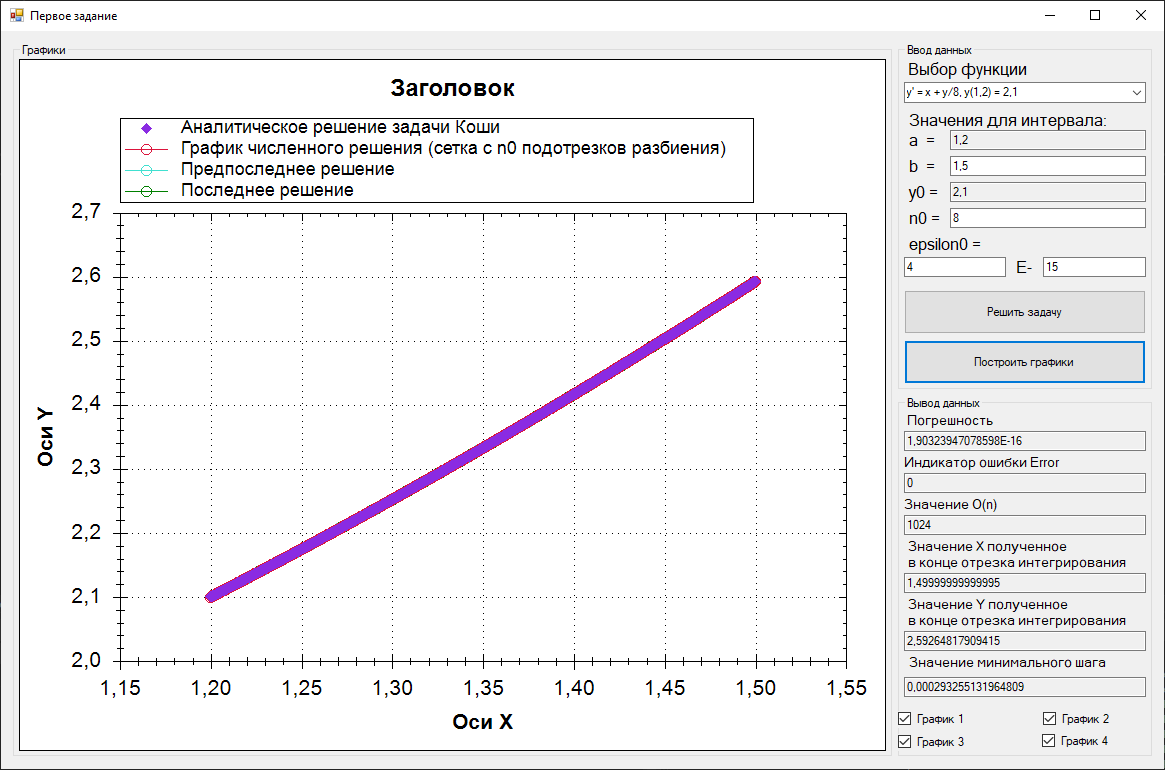


Рисунок 1 Пример работы программы для ОДУ 1

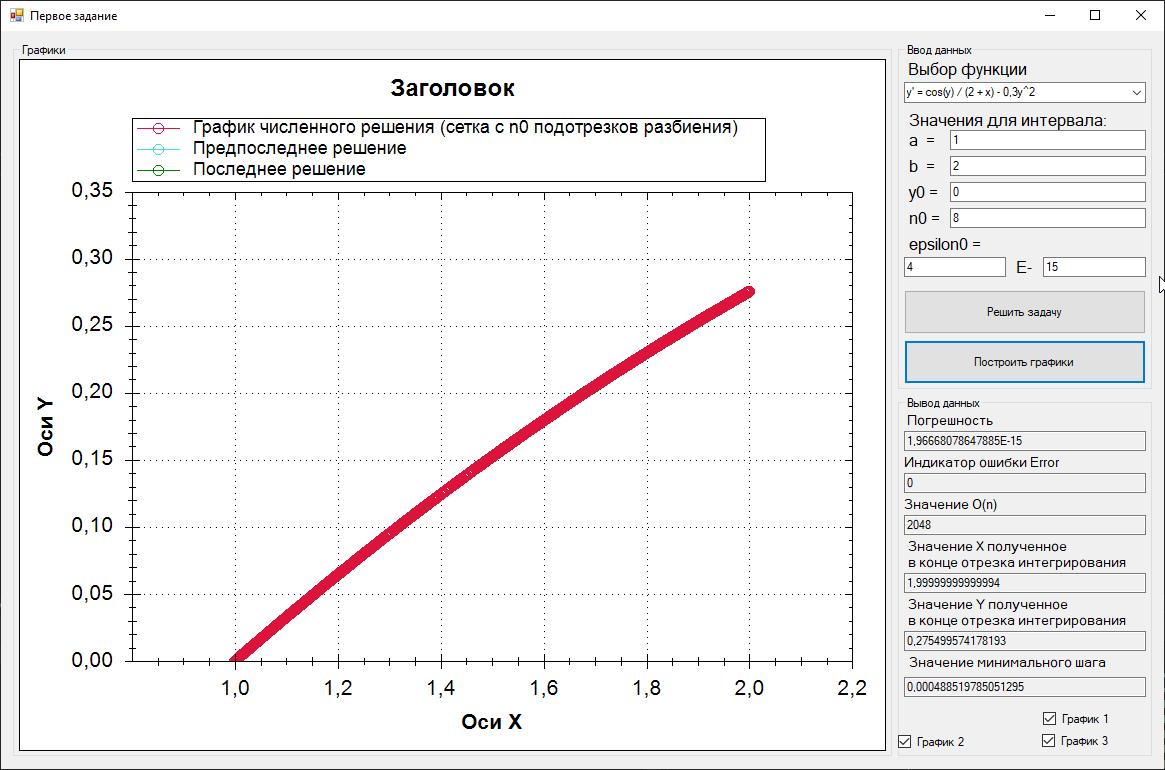


Рисунок 2 Пример работы программы для ОДУ 2

Задача 2

Постановка задачи

Найти численное решение задачи Коши для системы ОДУ 1-ого порядка вида (\*\*)

Методом типа Рунге-Кутта четвертого порядка с заданной точностью .

Используемые формулы

В случае метода типа Рунге – Кутта четвертого порядка формулы типа Рунге-Кутта содержат 13 неизвестных параметров; условия, обеспечивающие четвертый порядок точности метода на шаге, дают 11 нелинейных уравнений.

При решении задачи Коши для системы ОДУ (\*\*) мною была использована формула трёх восьмых:

Для того, чтобы получить решение задачи Коши для системы ОДУ нужно представить формулы Рунге-Кутта в векторном виде и расписать полученное в покомпонентно векторную формулу:

Реализация формул в коде

// Получение решения задачи Коши методом Рунге-Кутта порядка p

**private** **void** SolveCauchy(

**int** n,

IReadOnlyList<**double**> x,

**double** h,

**ref** **double**[] y,

**ref** **double**[] z

)

{

y = **new** **double**[n];

y[0] = Y0;

z = **new** **double**[n];

z[0] = Z0;

**for** (**var** i = 1; i < n; i++)

{

**var** k11 = Func1(z[i - 1]) \* h;

**var** k12 = h \* Func2(

x[i - 1],

y[i - 1],

z[i - 1]

);

**var** k21 = Func1(z[i - 1] \* k12) \* h;

**var** k22 = h \* Func2(

x[i - 1] + 1.0 / 3 \* h,

y[i - 1] + 1.0 / 3 \* k11,

z[i - 1] \* k12

);

**var** k31 = h \* Func1(z[i - 1] - 1.0 / 3 \* k11 + k12);

**var** k32 = h \* Func2(

x[i - 1] + 1.0 / 3 \* h,

y[i - 1] - 1.0 / 3 \* k11 + k21,

z[i - 1] - 1.0 / 3 \* k12 + k22

);

**var** k41 = h \* Func1(

z[i - 1] + k12 - k22 + k32

);

**var** k42 = h \* Func2(

x[i - 1] + h,

y[i - 1] + k11 - k21 + k31,

z[i - 1] + k12 - k22 + k32

);

y[i] = y[i - 1] + 1.0 / 8 \* (k11 + 3 \* k21 + 3 \* k31 + k41);

z[i] = z[i - 1] + 1.0 / 8 \* (k12 + 3 \* k22 + 3 \* k32 + k42);

}

}

**private** **int** GetResult()

{

// Значения текущей погрешности и предыдущей для проверки на уменьшении погрешности при уменьшении шага

**double** currentErrorY = 10;

**double** currentErrorZ = 10;

// Решаемо ли уравнение методом Рунге-Кутта или нет

// Размерности

**var** n = N0;

**var** n2 = 2 \* N0;

**double** h = 0;

**double** h2 = 0;

**do**

{

// Задаем равномерную сетку

**var** x = BuildGrid(n, **ref** h);

**var** x2 = BuildGrid(n2, **ref** h2);

**var** previousErrorY = currentErrorY;

**var** previousErrorZ = currentErrorZ;

**double**[] yn = **null**;

**double**[] zn = **null**;

**double**[] y2N = **null**;

**double**[] z2N = **null**;

SolveCauchy(n, x, h, **ref** yn, **ref** zn);

SolveCauchy(n2, x2, h2, **ref** y2N, **ref** z2N);

**var** isSolved =

yn.Length != 0

|| y2N.Length != 0

|| zn.Length != 0

|| z2N.Length != 0;

LastNQueue.Enqueue(n);

**while** (LastNQueue.Count > 3)

{

LastNQueue.Dequeue();

}

LastN2Queue.Enqueue(n2);

**while** (LastN2Queue.Count > 3)

{

LastN2Queue.Dequeue();

}

YError = currentErrorY = FindError(yn, y2N);

ZError = currentErrorZ = FindError(zn, z2N);

// Удваиваем количество точек на сетке

n \*= 2;

n2 \*= 2;

// Процесс решения прекращен, т.к. шаг стал меньше возможного

**if** (MinH > h) **return** 2;

// Процесс решения прекращен, т.к. с уменьшением шага погрешность не уменьшается

**if** (!(currentErrorY < previousErrorY) && !(currentErrorZ < previousErrorZ)) **return** 1;

// Решение не получено, двухсторонний метод Рунге-Кутта с данным начальным шагом не применим

**if** (!isSolved) **return** 4;

} **while** (!(currentErrorY < Epsilon0) && !(currentErrorZ < Epsilon0));

// Завершение в соответствии с назначенным условием о достижении заданной точности

**return** 0;

}

// Метод для построения равномерной сетки с количеством подотрезков n

**private** **double**[] BuildGrid(**int** n, **ref** **double** h)

{

**return**

Distribution.EvenNodes(A, B, n, **ref** h);

}

Численные эксперименты

При проведении численных экспериментов все значения для задачи Коши для системы ОДУ будут указаны на скриншотах программы.

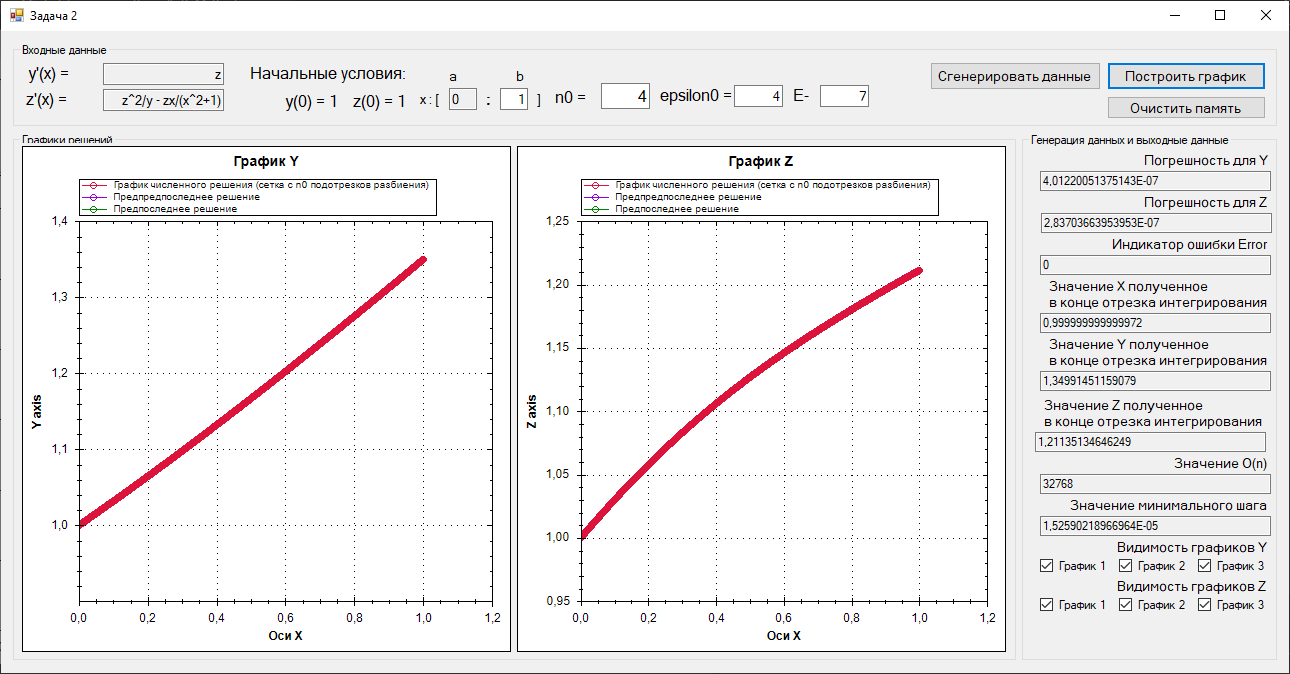


Рисунок 3 Пример работы программы для заданных значений

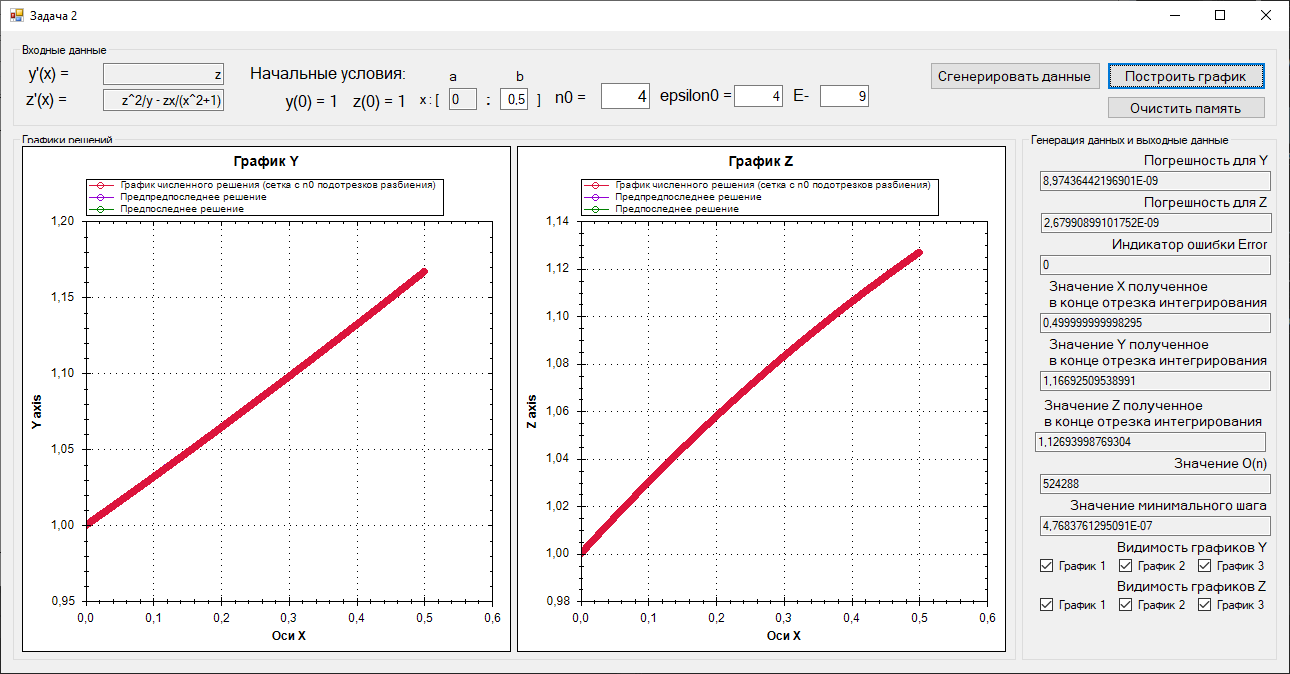


Рисунок 4 Пример работы программы для заданных значений

Вывод

В результате выполнения лабораторной работы у меня получилось найти численные решения для задач Коши, представленных в Задаче №1 и Задаче №2 данной лабораторной работы