Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Практическая работа № 4

Тема «Решение краевой задачи методом стрельбы»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с-ИБС 32  очной формы обучения  Солодилов В.В..  Проверил: доцент кафедры ИБС Кожанова Е.Р. |

Саратов 2022

**Цель:** формирование практических навыков решения краевой задачи линейного обыкновенного дифференциального уравнения методом стрельбы с их программной реализацией.

**1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

*Краевая задача (граничная задача)* – задача о нахождении решения заданного дифференциального уравнения (системы дифференциальных уравнений), удовлетворяющего [краевым (граничным) условиям](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%B8%D1%8F) в концах интервала или на границе области.

Одним из методов приближенного решения краевых задач для линейного обыкновенного дифференциального уравнения второго порядка является *метод прогонки.*

Постановка задачи. Требуется найти функцию , которая является решением следующей краевой задачи:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

Задачу (1), (2) называют краевой, поскольку дополнительные условия (2) задаются на концах отрезка

Разностная аппроксимация производных. Введем на отрезке равномерную сетку Записывая уравнение (1) во внутренних узлах сетки получим -но уравнение для определения неизвестных

|  |  |
| --- | --- |
|  | (3) |

где

Для того чтобы получить замкнутую систему уравнений относительно , необходимо первые и вторые проивзодные функции в узловых точках выразить через значения в этих точках. Будем предполагать, что функция имеет все необходимые по ходу рассуждения непрерывные производные. Выразим значения по формуле Тейлора, беря точку в качестве точки разложения:

|  |  |
| --- | --- |
|  | (4) |

Отсюда можно получить следующие выражения для точного значения первой производной функции в точке

|  |  |
| --- | --- |
|  | (5) |

а также выражение для точного значения второй производной функции в той же точке

|  |  |
| --- | --- |
|  | (6) |

Отношения

в формуле (5) называются правой разностной производной, левой разностной проиводной и центральной разностной производной, соответственно. Отношение

в (6) называется второй разностной производной.

Из формулы (5) следует, что левая и правая разностные производные аппроксимируют производную с первым порядком точности относительно шага а центральная разностная производная – со вторым порядком точности относительно Из (6) следует, что вторая разностная производная аппроксимирует производную со вторым порядком точности.

*Решение задачи методом стрельбы*. Метод стрельбы сводит решение краевой задачи для ОДУ к решению итерационной последовательности задач Коши.

Этот метод рассмотрим на примере следующей первой краевой задачи для ОДУ второго порядка:

|  |  |
| --- | --- |
|  | (7) |
|  | (8) |
|  | (9) |

Вместо краевой задачи (7) – (9) рассматривается следующая задача Коши:

|  |  |
| --- | --- |
|  | (10) |
|  | (11) |
|  | (12) |

в которой интегральная кривая зависит не только от переменной но и от параметра который называется *углом стрельбы*.

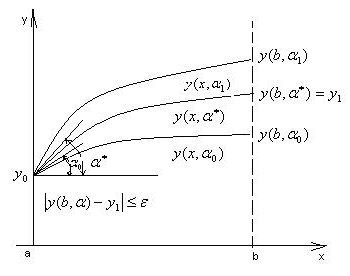


Рисунок 1. Геометрическая интерпретация метода стрельбы

Он выбирается из условия равенства значения интегральной кривой на правой границе  значению  с наперед заданной точностью

|  |  |
| --- | --- |
|  | (13) |

Угол пристрелки, удовлетворяющий неравенству (13), обозначим через Интегральная кривая, полученная из решения задачи Коши (10) – (12) с углом, близким к этому значению, в соответствии с неравенством (13) и будет решением краевой задачи (7) – (9) с точностью

Таким образом, алгоритм метода стрельбы, следующий:

1. Выбирается например из условия:
2. С этим значением  одним из методов решается задача Коши (10) – (12) с получением  и  если при этом выполняется условие (13), то краевая задача (7) – (9) решена с точностью
3. В противном случае могут быть следующие два варианта:
   1. тогда угол стрельбы каким-либо способом уменьшается и решается задача Коши (10) – (12) тем же методом до тех пор, пока не выполнится условие
   2. тогда угол стрельбы каким-либо способом увеличивается и решается задача Коши до тех пор, пока не выполнится условие
4. Таким образом, угол стрельбы находится внутри интервала  после чего истинное значение угла стрельбы определяется методом половинного деления с реализацией следующей цепочки:
   1. анализируется неравенство если оно выполнено, то и истинная интегральная кривая; если неравенство не выполнено, то итерационный процесс повторяется начиная с пункта 4 a.

**2. ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ**

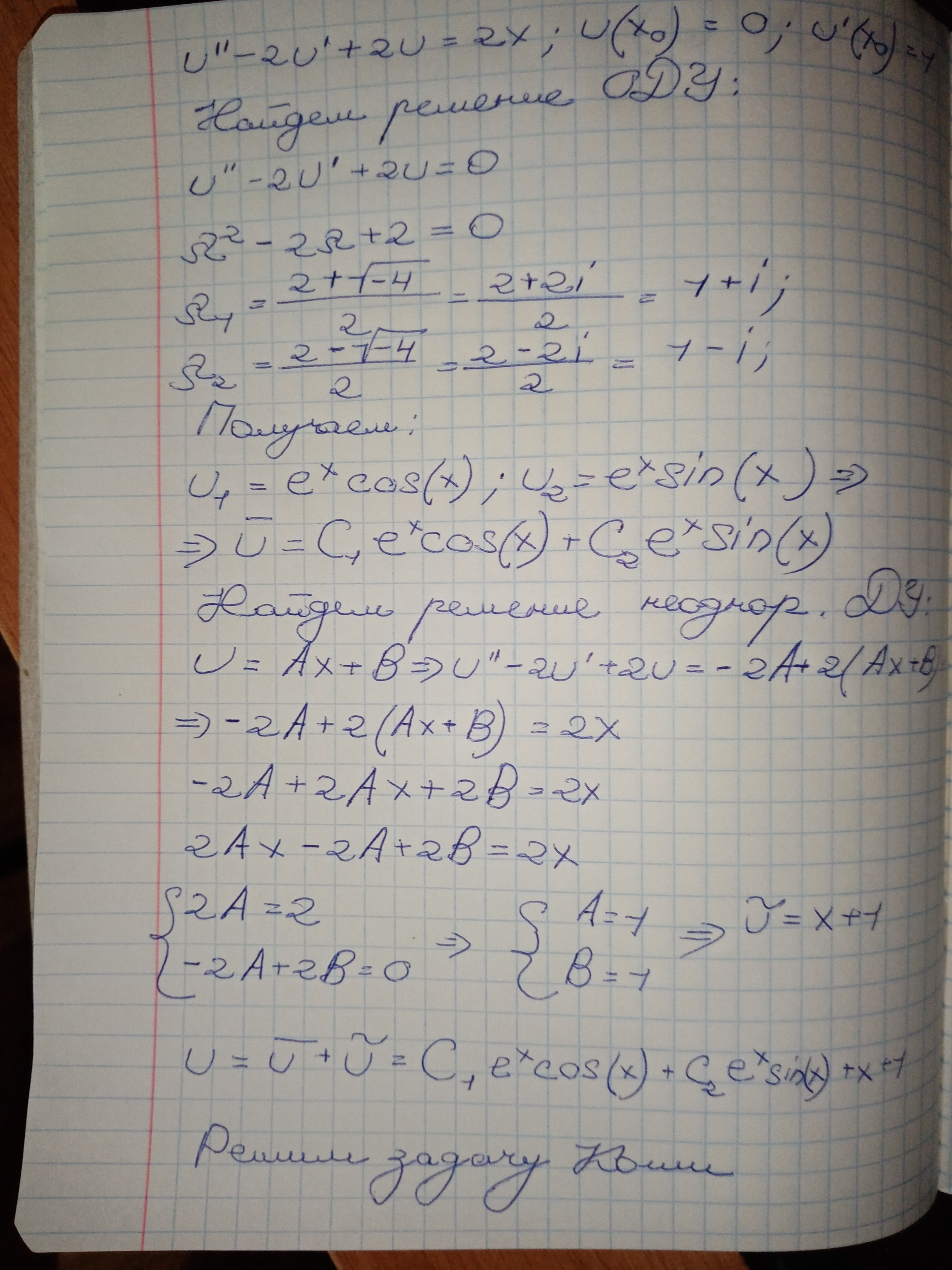
Задание:

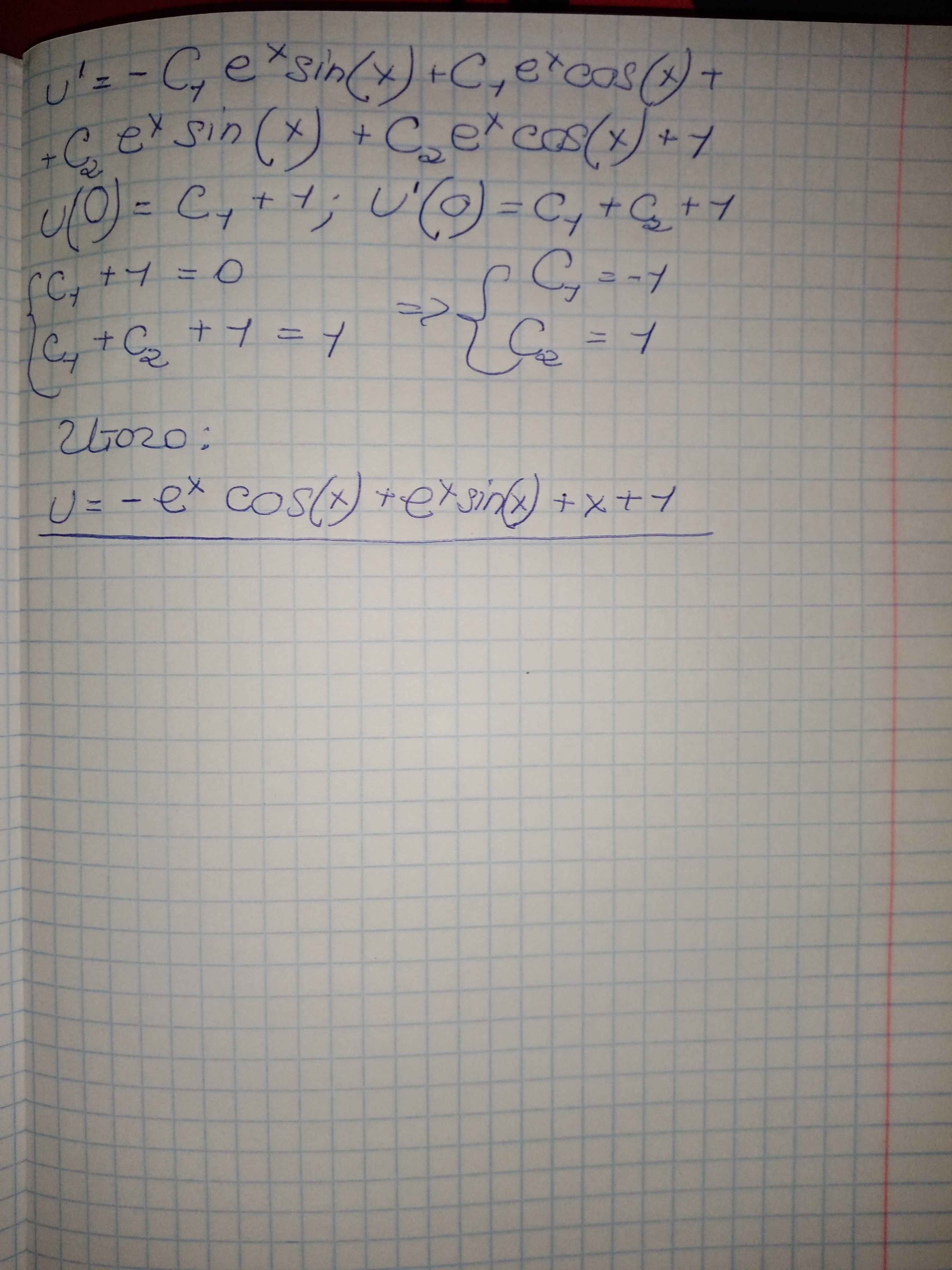
Решить краевую задачу методом стрельбы. Для задания краевого условия в точке предварительно необходимо решить аналитически соответствующую задачу Коши:

Значение находится постановкой точки в точное решение задачи Коши:

Здесь

**3. РАСЧЁТ РЕШЕНИЯ**

Аналитическое решение задачи Коши для ОДУ II-го порядка:

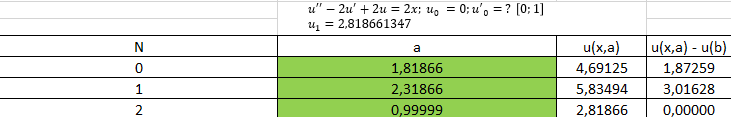


Итак, аналитическим решение ОДУ II-го порядка является выражение

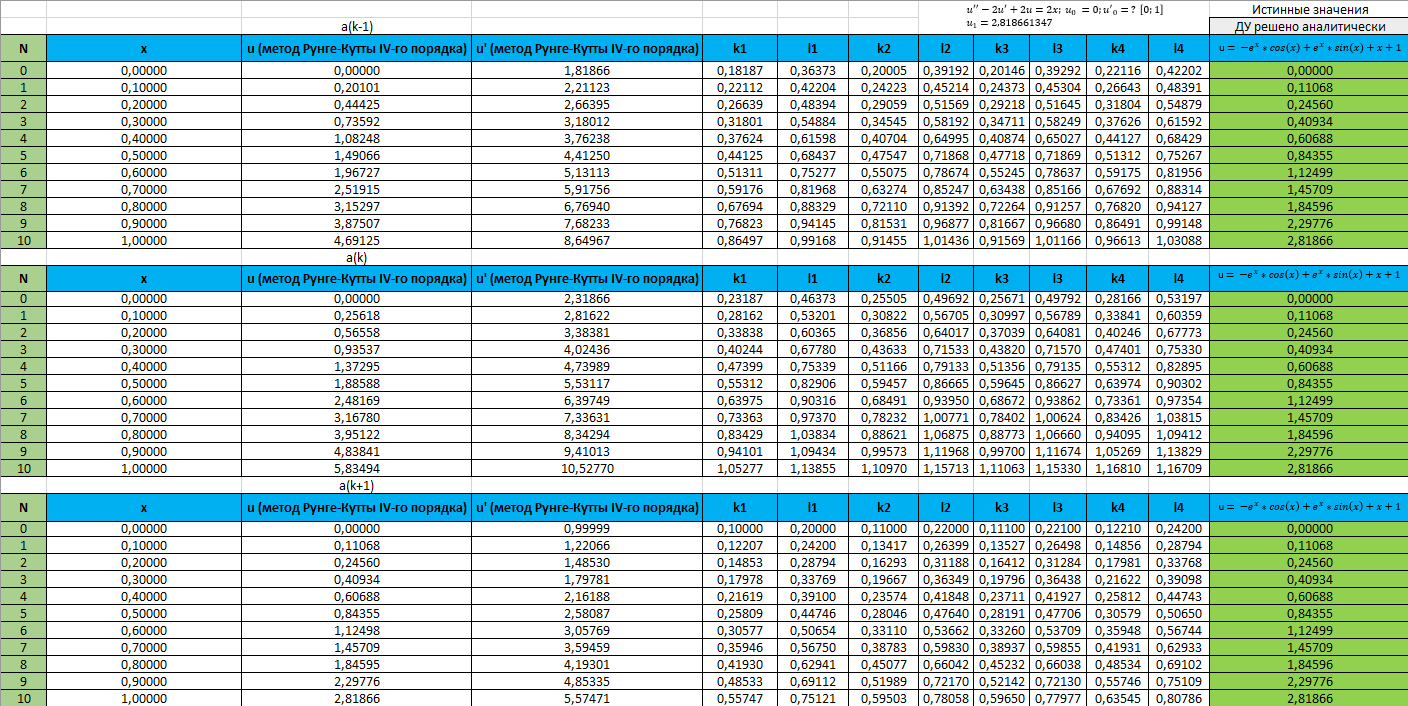
Путём подстановки точки в точное решение задачи Коши находится значение

Вычисления производились в программе Microsoft Excel. Для задания была составлена таблица, где было указано количество разбиений отрезка соответственно был рассчитан шаг разбиения Чтобы решить краевую задачу методом стрельбы, требуется неоднократно решить задачу Коши, например, методом Рунге-Кутты IV-го порядка точности, при разных значениях коэффициента где Коэффициент подбирается одним из вариантов метода Ньютона – методом секущих.

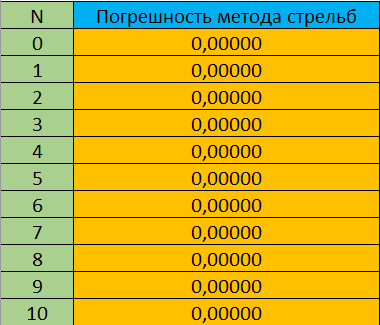
Нахождение угла «пристрелки» (нахождение угла останавливается на моменте, когда ):



Решение задачи Коши при разных значениях угла «пристрелки» (нужно обратить внимание на то, что последняя таблица – конечное решение задачи Коши, с уже подобранным коэффициентом ):



Также точные значения сравниваются со значениями, вычисленными методом стрельб, и показываются погрешности данного метода:



**4. Алгоритм решения задачи**

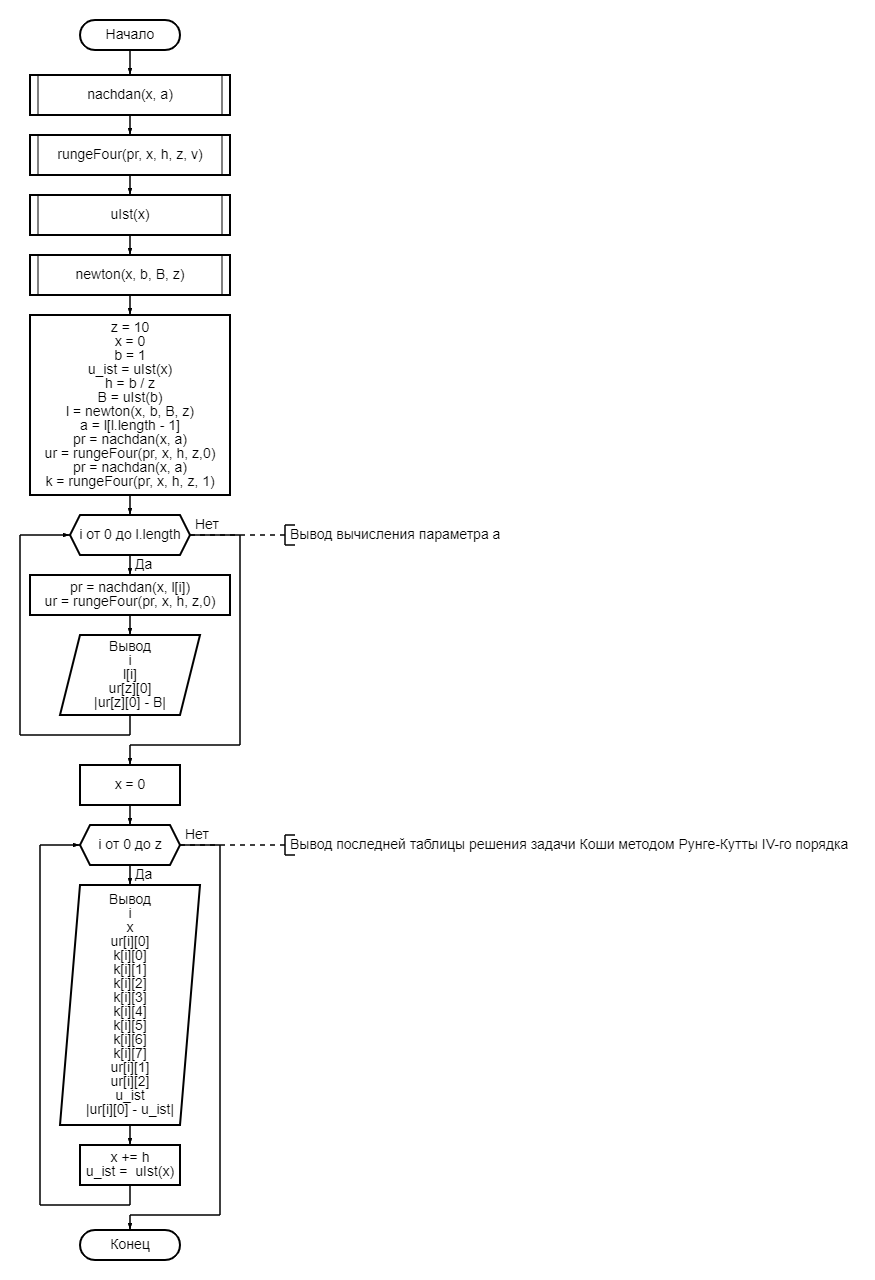
Для решения краевой задачи методом стрельбы для линейного ОДУ II-го порядка:

сначала нужно найти угол «пристрелки» можно использовать простейший метод дихотомии, то есть метод деления отрезка пополам, но этот метод применим, если решение задачи Коши не слишком чувствительно к изменению параметра поэтому лучше использовать один из вариантов метода Ньютона: метод секущих:

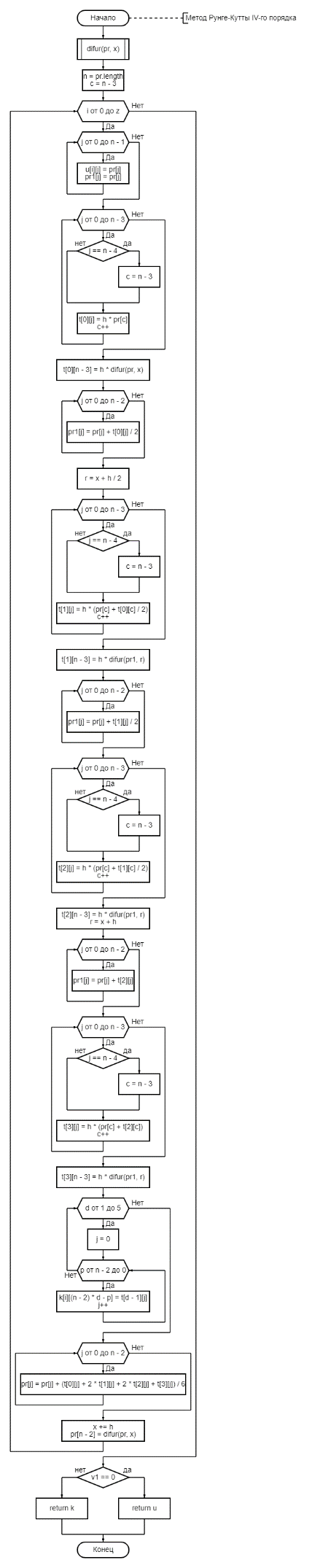
где решения задачи Коши при разных значениях параметра Для начала вычислений нужно задать первые значения параметра и Пусть и внутри формулы – границы заданного промежутка. Саму задачу Коши можно решить любым известным методом, например методом Рунге-Кутты IV-го порядка. Вычисление параметра останавливается тогда, когда выполняется условие где какая-либо заданная точность.

1. **БЛОК-СХЕМЫ**

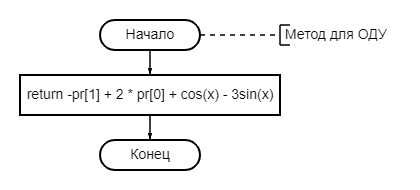
Для основной программы была составлена блок-схема:

****

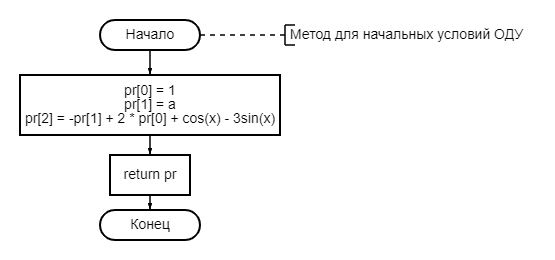
Для программы был составлен метод rungeFour(pr, x, h, z, v1):

****

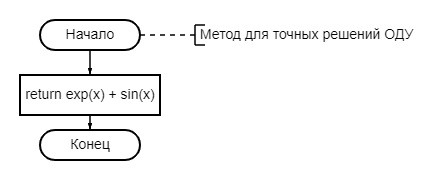
Для программы был составлен метод difur(pr, x):

****

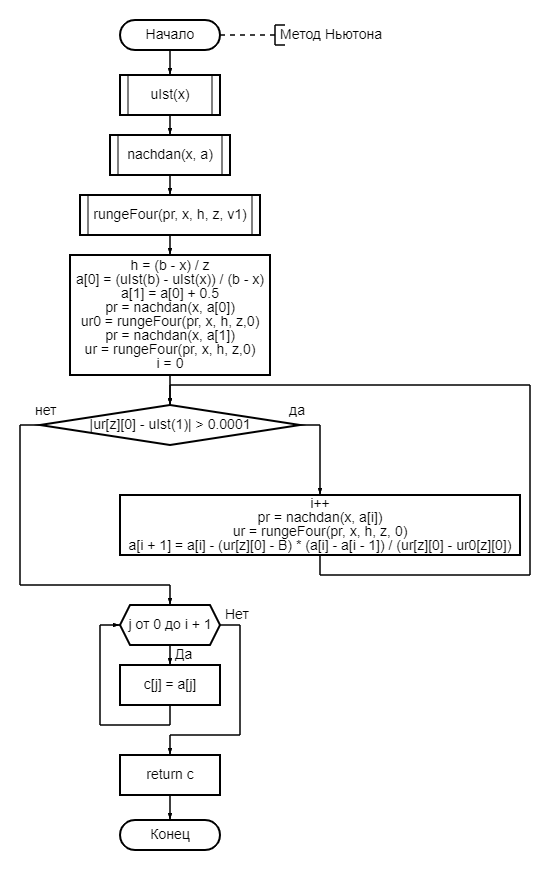
Для программы был составлен метод nachdan(x, a):

****

Для программы был составлен метод uIst(x):

****

Для программы был составлен метод newton(x, b, B, z):

****

**6. Программа для реализации алгоритма**

Данная программа была реализована на языке Java:

public class Main {

public static void main(String[] args){

int z = 10;

double x = 0, b = 1, u\_ist = uIst(x), h = b / z, B = uIst(b);

double[] pr;

double[][] ur, k;

double[] l = newton(x, b, B, z);

double a = l[l.length - 1];

pr = nachdan(x, a);

ur = rungeFour(pr, x, h, z,0);

pr = nachdan(x, a);

k = rungeFour(pr, x, h, z, 1);

System.out.println("Нахождение a - угла \"пристрелки\":");

System.out.println("N\t|a\t\t\t|u(x,a)\t\t||u(x,a) - u(b)|");

for(int i = 0; i < l.length; i++){

pr = nachdan(x, l[i]);

ur = rungeFour(pr, x, h, z,0);

System.out.println(i + "\t|" + String.format("%.5f", l[i]) + "\t|" + String.format("%.5f", ur[z][0]) + "\t|" + String.format("%.5f", Math.abs(ur[z][0] - B)));

}

x = 0;

System.out.println("\nПрограмма высчитывает приближённые значения в узлах задачи Коши для следующих условий: u'' + u' - 2u = cos(x) - 3sin(x); a = 0, b = 1; u(0) = 1, u(b) = " + String.format("%.5f", B) + ", u'(0) = " + String.format("%.5f", a) + " - подобранный угол \"пристрелки\"");

System.out.println("Метод Рунге-Кутты IV-го порядка:");

System.out.println("N\t|x\t\t\t|u\t\t\t|k1\t\t\t|l1\t\t\t|k2\t\t\t|l2\t\t\t|k3\t\t\t|l3\t\t\t|k4\t\t\t|l4\t\t\t\t|u'\t\t\t\t\t|u'' = 4u' - 2u + 4e^(5x)\t|u\_аналитич\t\t|Погрешность");

for (int i = 0; i <= z; i++){

System.out.println(i + "\t|" + String.format("%.5f", x) + "\t|" + String.format("%.5f", ur[i][0]) + "\t|" + String.format("%.5f", k[i][0]) + "\t|" + String.format("%.5f", k[i][1]) + "\t|" + String.format("%.5f", k[i][2]) + "\t|" + String.format("%.5f", k[i][3]) + "\t|" + String.format("%.5f", k[i][4]) + "\t|" + String.format("%.5f", k[i][5]) + "\t|" + String.format("%.5f", k[i][6]) + "\t|" + String.format("%.5f", k[i][7]) + "\t\t|" + String.format("%.5f", ur[i][1]) + "\t\t\t|" + String.format("%.5f", ur[i][2]) + "\t\t\t\t\t|" + String.format("%.5f", u\_ist) + "\t\t|" + String.format("%.5f", Math.abs(ur[i][0] - u\_ist)));

x += h;

u\_ist = uIst(x);

}

}

public static double[][] rungeFour(double[] pr, double x, double h, int z, int v1){ //Метод Рунге-Кутты IV-го порядка для ОДУ (любого порядка)

int n = pr.length, c = n - 3;

double[][] u = new double[z + 1][n];

double[][] t = new double[4][n - 2]; // t[0][n - 2] - k1, t[1][n - 2] - k2, t[2][n - 2] - k3, t[3][n - 2] - k4

double[][] k = new double[z + 1][(int) Math.pow(2, n - 1)];

double[] pr1 = new double[n];

double r;

for (int i = 0; i <= z; i++) {

for (int j = 0; j < n - 1; j++) {

u[i][j] = pr[j];

pr1[j] = pr[j];

}

for (int j = 0; j < n - 3; j++, c++) {

if (j == n - 4)

c = n - 3;

t[0][j] = h \* pr[c]; //k1

}

t[0][n - 3] = h \* difur(pr, x); //l1

for (int j = 0; j < n - 2; j++) {

pr1[j] = pr[j] + t[0][j] / 2;

}

r = x + h / 2;

for (int j = 0; j < n - 3; j++, c++) {

if (j == n - 4)

c = n - 3;

t[1][j] = h \* (pr[c] + t[0][c] / 2); //k2

}

t[1][n - 3] = h \* difur(pr1, r); //l2

for (int j = 0; j < n - 2; j++) {

pr1[j] = pr[j] + t[1][j] / 2;

}

for (int j = 0; j < n - 3; j++, c++) {

if (j == n - 4)

c = n - 3;

t[2][j] = h \* (pr[c] + t[1][c] / 2); //k3

}

t[2][n - 3] = h \* difur(pr1, r); //l3

r = x + h;

for (int j = 0; j < n - 2; j++) {

pr1[j] = pr[j] + t[2][j];

}

for (int j = 0; j < n - 3; j++, c++) {

if (j == n - 4)

c = n - 3;

t[3][j] = h \* (pr[c] + t[2][c]); //k4

}

t[3][n - 3] = h \* difur(pr1, r); //l4

for (int d = 1; d < 5; d++) {

int j = 0;

for (int p = n - 2; p > 0; p--, j++) {

k[i][(n - 2) \* d - p] = t[d - 1][j];

}

}

for (int j = 0; j < n - 2; j++){

pr[j] = pr[j] + (t[0][j] + 2 \* t[1][j] + 2 \* t[2][j] + t[3][j]) / 6;

}

x += h;

pr[n - 2] = difur(pr, x);

}

if(v1 == 0)

return u;

else

return k;

}

public static double difur(double[] pr, double x){ //Метод для ОДУ I-го и II-го порядков

return -pr[1] + 2 \* pr[0] + Math.cos(x) - 3 \* Math.sin(x);

}

public static double[] nachdan(double x, double a){ //Метод для начальных условий ОДУ I-го и II-го порядков

double pr[];

pr = new double[4];

pr[0] = 1;

pr[1] = a;

pr[2] = -pr[1] + 2 \* pr[0] + Math.cos(x) - 3 \* Math.sin(x);

return pr;

}

public static double uIst(double x){ //Метод для точных решений ОДУ

return Math.exp(x) + Math.sin(x);

}

public static double[] newton(double x, double b, double B, int z){ //Метод Ньютона (используется разностная производная)

double[] pr;

double[][] ur0, ur;

double h = (b - x) / z;

double[] a = new double[5];

a[0] = (uIst(b) - uIst(x)) / (b - x);

a[1] = a[0] + 0.5;

pr = nachdan(x, a[0]);

ur0 = rungeFour(pr, x, h, z,0);

pr = nachdan(x, a[1]);

ur = rungeFour(pr, x, h, z,0);

int i = 0;

while (Math.abs(ur[z][0] - uIst(1)) > 0.00001) {

i++;

pr = nachdan(x, a[i]);

ur = rungeFour(pr, x, h, z,0);

a[i + 1] = a[i] - (ur[z][0] - B) \* (a[i] - a[i - 1]) / (ur[z][0] - ur0[z][0]);

}

double[] c = new double[i + 1];

for (int j = 0; j < i + 1; j++){

c[j] = a[j];

}

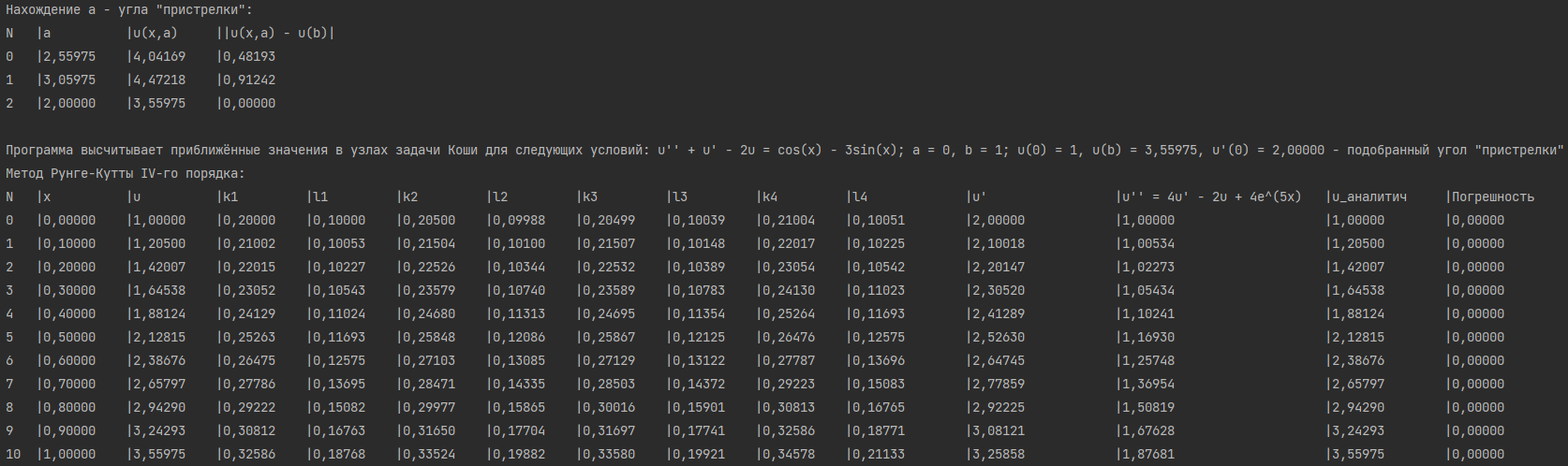
return c;

}

}

**7. Контрольный тест**

Результаты работы программы:



Если сравнить полученные программой данные, с данными, которые были получены при расчёте в Microsoft Excel (п. 2), то можно увидеть, что значения совпадают друг с другом.

**8. ВЫВОДЫ**

В рамках данной работы были применены основные навыки решения краевой задачи для линейного обыкновенного дифференциального уравнения II-го порядка методом стрельбы, а также была написана программа, реализующая решение задания на языке Java. Результаты программы и ручного расчета совпали.