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 № 2

Тема «Методы Рунге-Кутты для решения задачи Коши»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с-ИБС 31  очной формы обучения  Савельев С.А.  Проверил: доцент кафедры ИБС Кожанова Е.Р. |

Саратов 2021

Целью данной работы является формирование практических навыков решения задачи Коши для линейных обыкновенных дифференциальных уравнений первого и второго порядков методом Рунге-Кутта с их программной реализацией.

**ЗАДАНИЕ НА ПРАКТИЧЕКСУЮ РАБОТУ**

1. Изучить основные понятия по данной теме.

2. Выполнить расчет по заданному варианту № 21.

3. Разработать алгоритм решения задачи.

4. Написать программу для реализации алгоритма.

5. Выполнить контрольный тест работы программы и сравнить с результатами, полученными в п.2.

6. Выводы.

**1. ОСНОВНЫЕ ПОНЯТИЯ**

*Методы Рунге-Кутта* — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. К классу методов Рунге-Кутта относятся явный метод Эйлера и модифицированный метод Эйлера с пересчётом, которые представляют собой соответственно методы первого и второго порядка точности. Существуют стандартные явные методы третьего порядка точности, не получившие широкого распространения. Наиболее часто используется и реализован в различных математических пакетах классический метод Рунге-Кутта, имеющий четвёртый порядок точности.

Пусть дано дифференциальное уравнение первого порядка

с начальным условием

В окрестностях точки функцию разложим в ряд Тейлора:

который можно применить для приближенного определения искомой функции . Для уменьшения погрешности метода интегрирования дифференциального уравнения необходимо учитывать большее количество членов ряда. Однако при этом возникает необходимость аппроксимации производных от правых частей дифференциального уравнения.

Основная идея методов Рунге-Кутты заключается в том, что производные аппроксимируются через значения функции в точках на интервале которые выбираются из условия наибольшей близости алгоритма к ряду Тейлора. В зависимости от старшей степени *h*, с которой учитываются члены ряда, построены вычислительные схемы Рунге-Кутты разных порядков точности.

Так, например, общая форма записи метода Рунге-Кутты второго порядка следующая:

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

где

Решение ОДУ, полученное по этой схеме, имеет погрешность Для параметра а наиболее часто используют значения

Рассмотрим первый вариант метода Рунге-Кутта второго порядка. При формула (1) примет вид:

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

Формулу (2) можно представить в виде следующей схемы:

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

Это метод Рунге-Кутты второго порядка (1-й вариант), или исправленный метод Эйлера.

Геометрически процесс нахождения точки можно проследить по рис.1.

По методу Эйлера находится точка лежащая на прямой . В этой точке снова вычисляется тангенс угла наклона касательной (прямая ).

Усреднение двух тангенсов дает прямую . Проводим через точку прямую *,* параллельную *.* Точка, в которой прямая  пересечется с ординатой  будет искомой точкой

Тангенс угла наклона прямой равен

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

Уравнение прямой запишется в виде:

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

тогда в точке с учетом (4) получим решение:

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

Формула описывает метод Рунге-Кутта второго порядка при

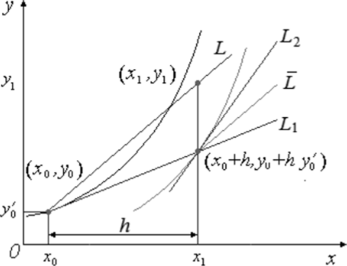


Рис.1

В случае второго варианта метода Рунге-Кутта второго порядка принимают при Тогда формула (1) принимает вид:

|  |  |
| --- | --- |
|  | (7) |

Представим формулу (7) в виде схемы:

|  |  |
| --- | --- |
|  | (8) |

Это метод Рунге-Кутта второго порядка (2-й вариант), или модифицированный метод Эйлера. Геометрическая интерпретация метода Рунге-Кутта при представлена на рис.2.

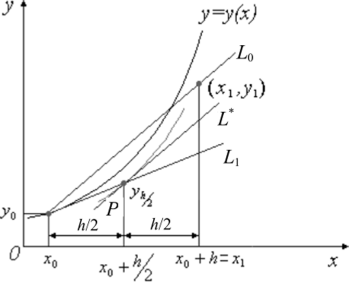


Рис.2

Формула описывает метод Рунге-Кутта второго порядка при

Метод Рунге-Кутта четвертого порядка описывается системой следующих соотношений:

|  |  |
| --- | --- |
|  | (9) |

где

|  |  |
| --- | --- |
|  | (10) |

Геометрическая интерпретация метода представлена на рис.3.

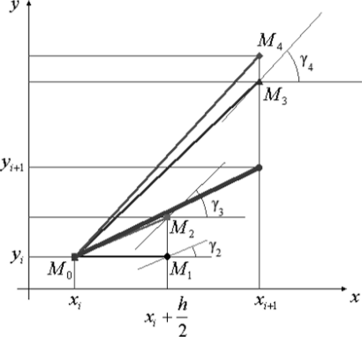


Рис.3

Задача Коши для дифференциального уравнения второго порядка сводится к системе двух уравнений первого порядка.

Получается, метод Рунге-Кутта второго порядка для дифференциального уравнения второго порядка примет вид:

|  |  |
| --- | --- |
|  | (11) |

Метод Рунге-Кутта четвертого порядка для дифференциального уравнения второго порядка примет вид:

|  |  |
| --- | --- |
|  | (12) |

**2. Расчет по заданному варианту №21**

Задание:

Решить задачу Коши для обыкновенного дифференциальное уравнения первого порядка для заданного варианта № 21 методом Рунге – Кутта второго и четвертого порядков точности. Сравнить результаты между собой и с результатами, полученными методом Эйлера.

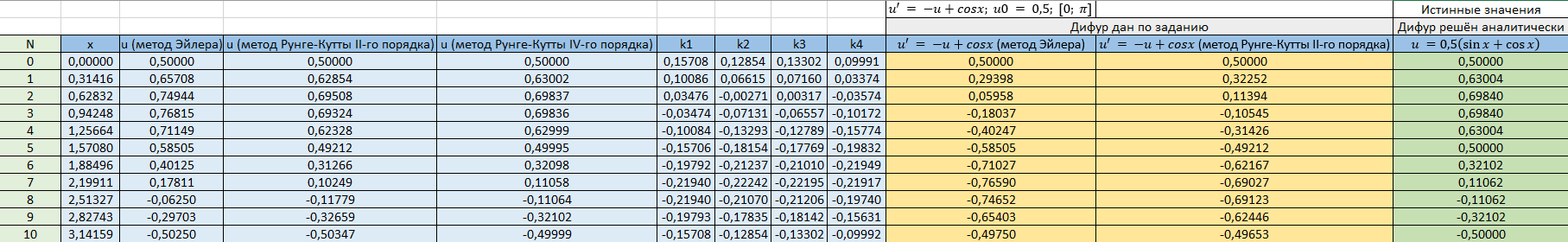
Дана следующая функция:

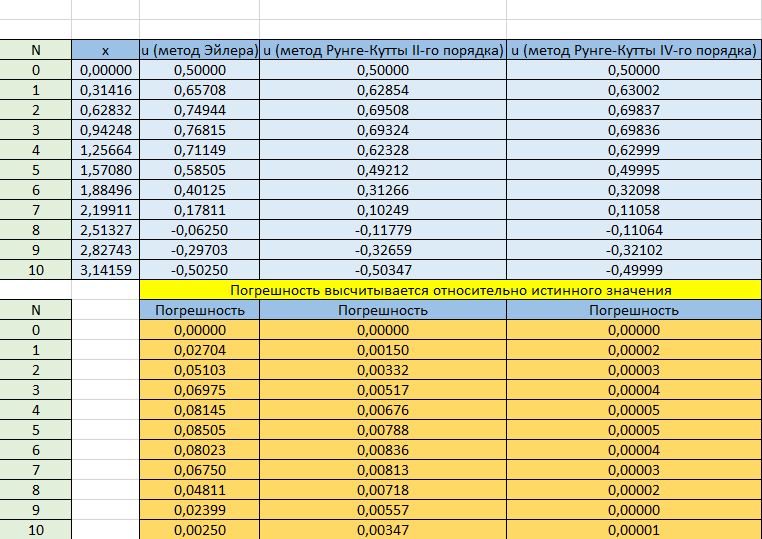
определённая на интервале с начальным условием

Аналитическое решение дифференциального уравнения представляет собой выражение

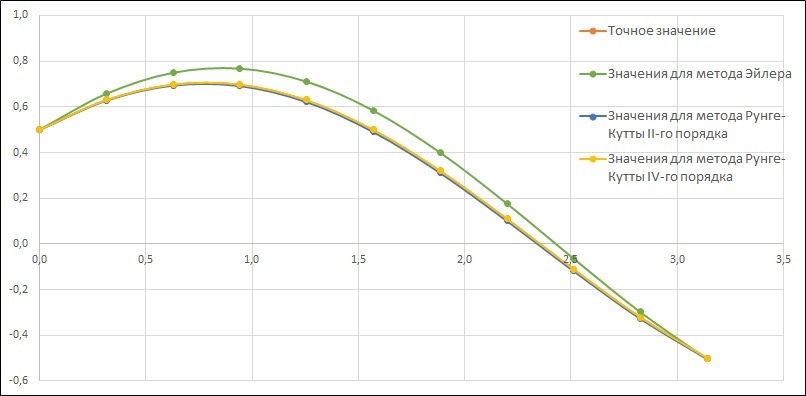
Вычисления производились в программе Microsoft Excel. Для задания была составлена таблица, где по формулам (8) считаются значения функции при разном значении аргумента для метода Рунге-Кутты II-го порядка и по формулам (10) – для метода Рунге-Кутты IV-го порядка.

В таблице приведены значения функции которые высчитываются при помощи метода Эйлера, метода Рунге-Кутты II-го порядка точности и метода Рунге-Кутты IV-го порядка точности. Также были посчитаны промежуточные коэффициенты для метода Рунге-Кутты IV-го порядка точности:

Значения функции посчитанные методом Эйлера нужны для сравнения с другими методами. Отдельно для наглядности были посчитаны погрешности методов, относительно истинного решения ОДУ, найденного аналитическим путём:



Также ниже приведены графики, которые наглядно показывают, как графики численных методов отклоняются от графика точных значений решения ОДУ:



Стоит обратить внимание на то, что всего было сделано 10 шагов разбиения отрезка.

Задание 2:

Решить задачу Коши для обыкновенного дифференциальное уравнения второго порядка для заданного варианта № 21 методом Рунге – Кутта второго и четвертого порядков точности. Сравнить результаты между собой и с результатами, полученными методом Эйлера.

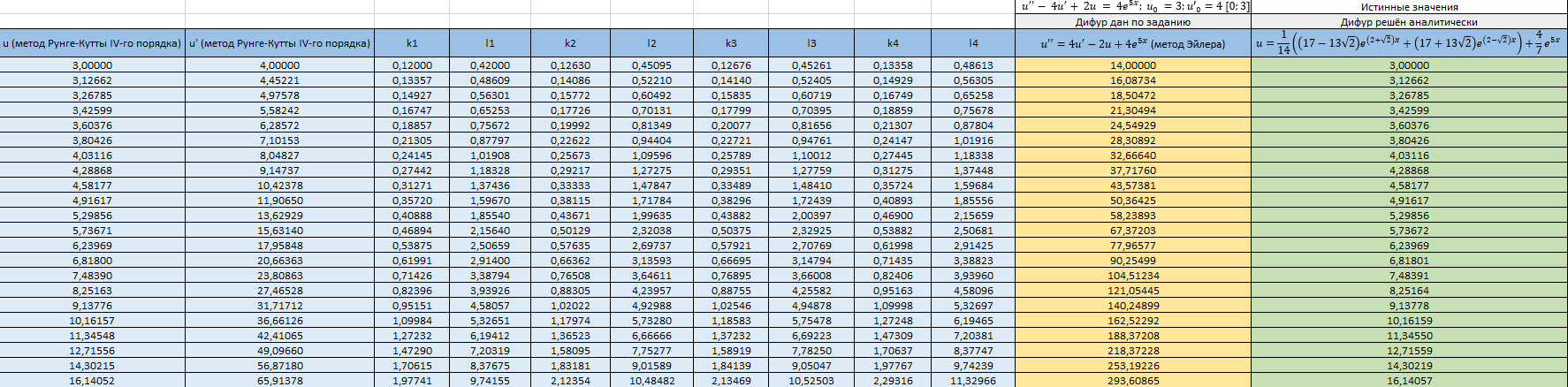
Дана следующая функция:

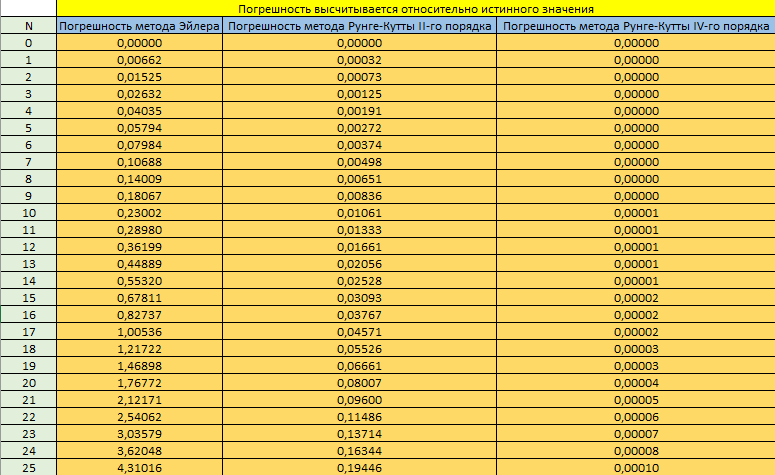
определённая на интервале с начальными условиями

Аналитическое решение дифференциального уравнения представляет собой выражение

Вычисления производились в программе Microsoft Excel. Для задания была составлена таблица, где по формулам (11) считаются значения функции при разном значении аргумента для метода Рунге-Кутты II-го порядка и по формулам (12) – для метода Рунге-Кутты IV-го порядка.

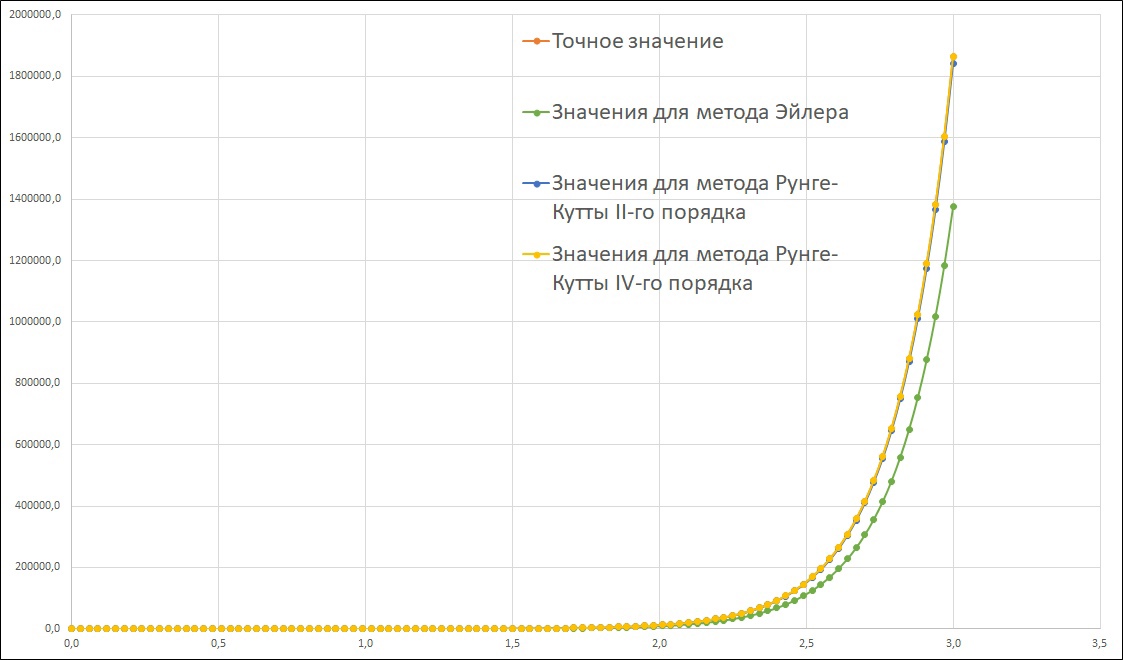
В таблице приведены значения функции которые высчитываются при помощи метода Эйлера, метода Рунге-Кутты II-го порядка точности и метода Рунге-Кутты IV-го порядка точности. Также были посчитаны промежуточные коэффициенты для методов Рунге-Кутты II-го и IV-го порядка точности:

Значения функции посчитанные методом Эйлера нужны для сравнения с другими методами. Отдельно для наглядности были посчитаны погрешности методов, относительно истинного решения ОДУ, найденного аналитическим путём:



Стоит обратить внимание на то, что было всего было сделано 100 шагов разбиения отрезка.

Также ниже приведены графики, которые наглядно показывают, как графики численных методов отклоняются от графика точных значений решения ОДУ:



**3. Алгоритм решения задачи**

Для решения задачи Коши для линейных ОДУ I-го и II-го порядков методом Рунге-Кутты II-го порядка точности используются следующие формулы:

Для I-го порядка:

Формулу можно упростить следующим образом:

Для II-го порядка:

Для решения задачи Коши для линейных ОДУ I-го и II-го порядков методом Рунге-Кутты IV-го порядка точности используются следующие формулы:

Для I-го порядка:

где

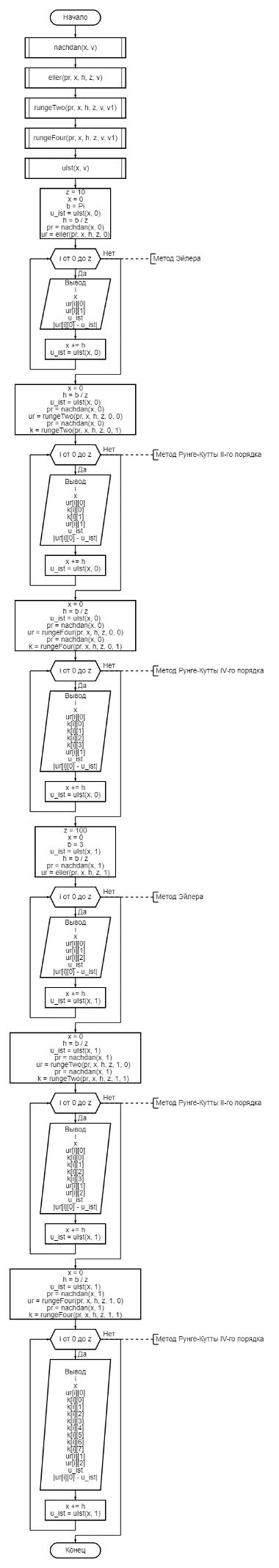
Для II-го порядка:

шаг разбиения отрезка количество разбиений.

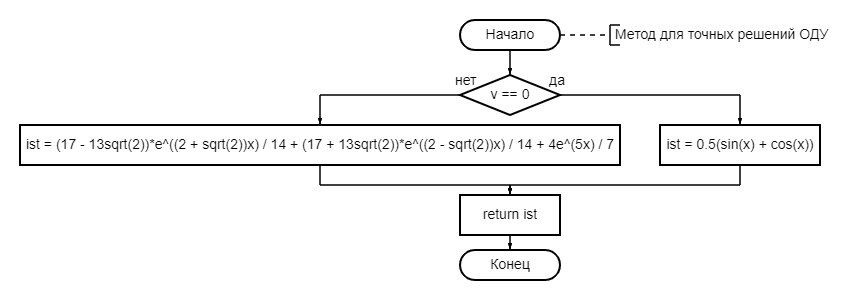
Чем больше разбиенией (и соответственно меньше шаг), тем больше точность вычисления.

При этом важно, чтобы были заданы начальные условия как для ОДУ I-го порядка, так и для ОДУ II-го порядка.

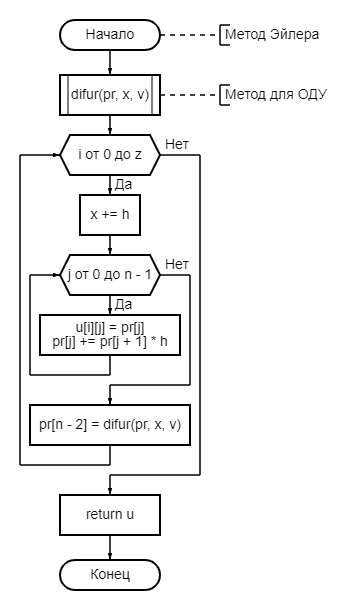
Для основной программы была составлена блок-схема:

****

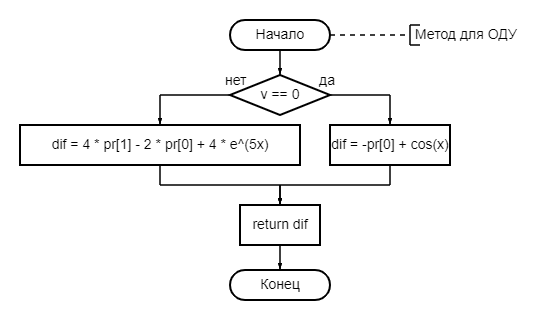
Для программы был составлен метод uIst(x, v):



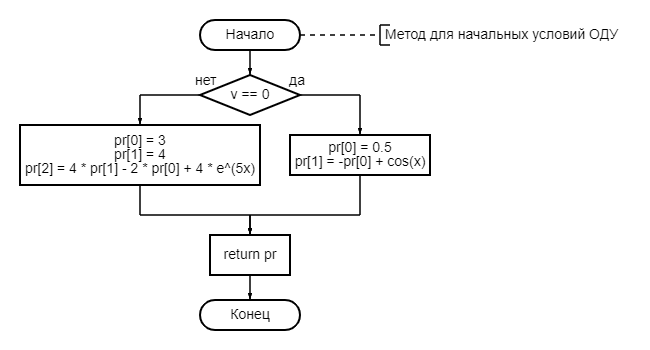
Для программы был составлен метод eiler(pr, x, h, z, v):

****

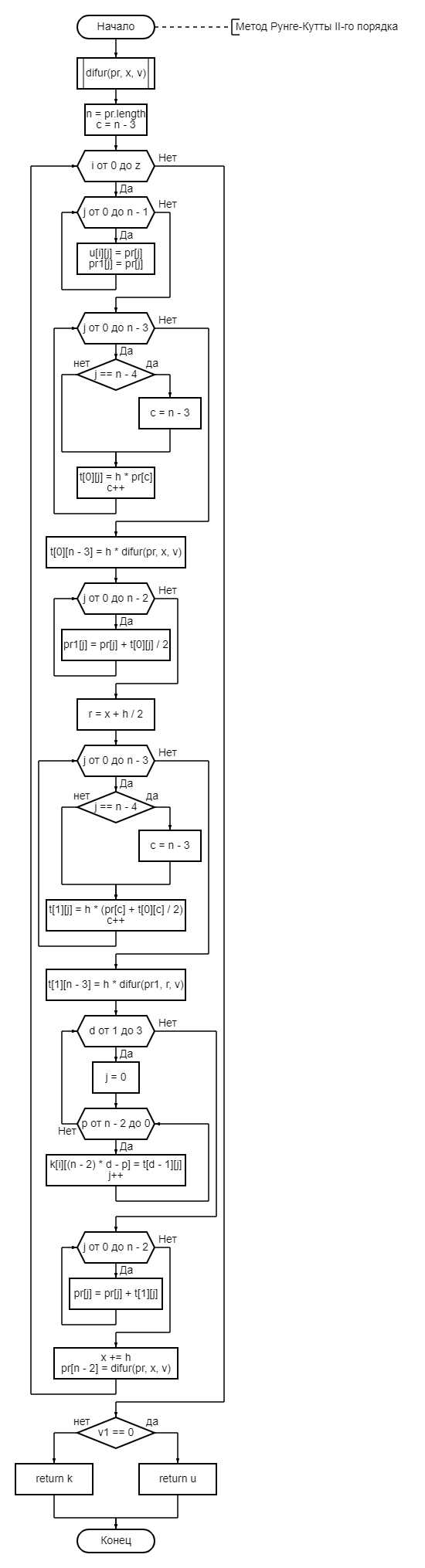
Для программы был составлен метод difur(pr, x, v):

****

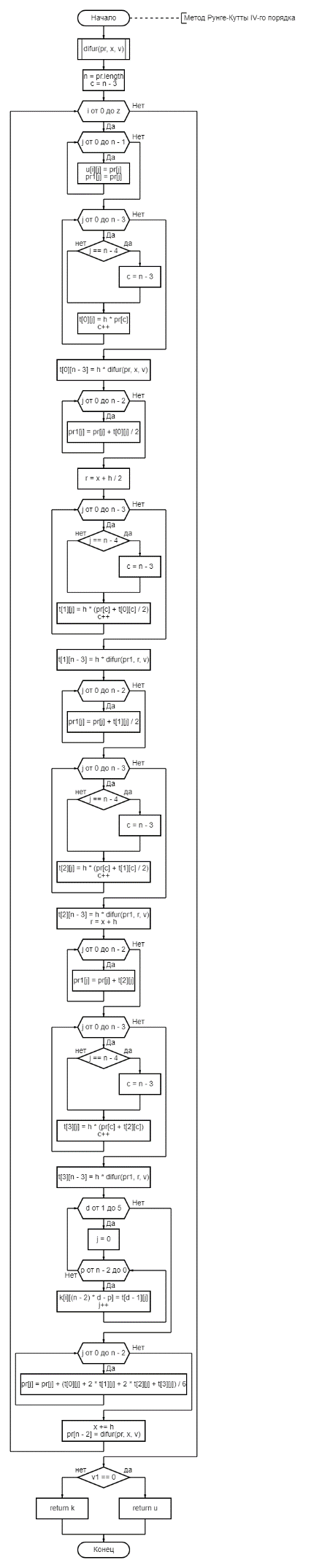
Для программы был составлен метод nachdan(x, v):

****

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

****

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

****

**4. 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 = Math.PI, u\_ist = uIst(x, 0), h = b / z;

double[] pr;

double[][] ur, k;

pr = nachdan(x, 0);

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

System.out.println("Программа высчитывает приближённые значения в узлах задачи Коши для следующих условий: u' = -u + cos(x); a = 0, b = pi; u0 = 0,5");

System.out.println("Метод Эйлера:");

System.out.println("N\t|x\t\t\t|u\t\t\t|u' = -u + cos(x)\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", ur[i][1]) + "\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, 0);

}

x = 0; h = b / z;

u\_ist = uIst(x, 0);

pr = nachdan(x, 0);

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

pr = nachdan(x, 0);

k = rungeTwo(pr, x, h, z, 0, 1);

System.out.println("\n\nМетод Рунге-Кутты II-го порядка:");

System.out.println("N\t|x\t\t\t|u\t\t\t|k1\t\t\t|k2\t\t\t|u' = -u + cos(x)\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", ur[i][1]) + "\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, 0);

}

x = 0; h = b / z;

u\_ist = uIst(x, 0);

pr = nachdan(x, 0);

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

pr = nachdan(x, 0);

k = rungeFour(pr, x, h, z, 0, 1);

System.out.println("\n\nМетод Рунге-Кутты IV-го порядка:");

System.out.println("N\t|x\t\t\t|u\t\t\t|k1\t\t\t|k2\t\t\t|k3\t\t\t|k4\t\t\t|u' = -u + cos(x)\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", ur[i][1]) + "\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, 0);

}

z = 100;

x = 0; b = 3; u\_ist = uIst(x, 1); h = b / z;

pr = nachdan(x, 1);

ur = eiler(pr, x, h, z, 1);

System.out.println("\n\nПрограмма высчитывает приближённые значения в узлах задачи Коши для следующих условий: u'' - 4u' + 2u = 4e^(5x); a = 0, b = 3; u(0) = 3, u'(0) = 4");

System.out.println("Метод Эйлера:");

System.out.println("N\t|x\t\t\t|u\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\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, 1);

}

x = 0; h = b / z;

u\_ist = uIst(x, 1);

pr = nachdan(x, 1);

ur = rungeTwo(pr, x, h, z, 1, 0);

pr = nachdan(x, 1);

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

System.out.println("\n\nМетод Рунге-Кутты II-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\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\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, 1);

}

x = 0; h = b / z;

u\_ist = uIst(x, 1);

pr = nachdan(x, 1);

ur = rungeFour(pr, x, h, z, 1, 0);

pr = nachdan(x, 1);

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

System.out.println("\n\nМетод 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, 1);

}

}

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

double ist;

if (v == 0)

ist = 0.5 \* (Math.sin(x) + Math.cos(x));

else

ist = (17 - 13 \* Math.sqrt(2)) \* Math.exp((2 + Math.sqrt(2)) \* x) / 14 + (17 + 13 \* Math.sqrt(2)) \* Math.exp((2 - Math.sqrt(2)) \* x) / 14 + 4 \* Math.exp(5 \* x) / 7;

return ist;

}

public static double[][] eiler(double[] pr, double x, double h, int z, int v){ //Метод Эйлера для ОДУ (любого порядка)

int n = pr.length;

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

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

x += h;

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

u[j][k] = pr[k];

pr[k] += pr[k + 1] \* h;

}

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

}

return u;

}

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

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

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

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

double[][] k = new double[z + 1][2 \* n - 4];

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, v); //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, v); //l2

for (int d = 1; d < 3; 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[1][j];

}

x += h;

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

}

if (v1 == 0)

return u;

else

return k;

}

public static double[][] rungeFour(double[] pr, double x, double h, int z, int v, 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, v); //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, v); //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, v); //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, v); //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, v);

}

if(v1 == 0)

return u;

else

return k;

}

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

double dif;

if (v == 0)

dif = -pr[0] + Math.cos(x);

else

dif = 4 \* pr[1] - 2 \* pr[0] + 4 \* Math.exp(5 \* x);

return dif;

}

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

double pr[];

if (v == 0){

pr = new double[3];

pr[0] = 0.5;

pr[1] = -pr[0] + Math.cos(x);

}

else{

pr = new double[4];

pr[0] = 3;

pr[1] = 4;

pr[2] = 4 \* pr[1] - 2 \* pr[0] + 4 \* Math.exp(5 \* x);

}

return pr;

}

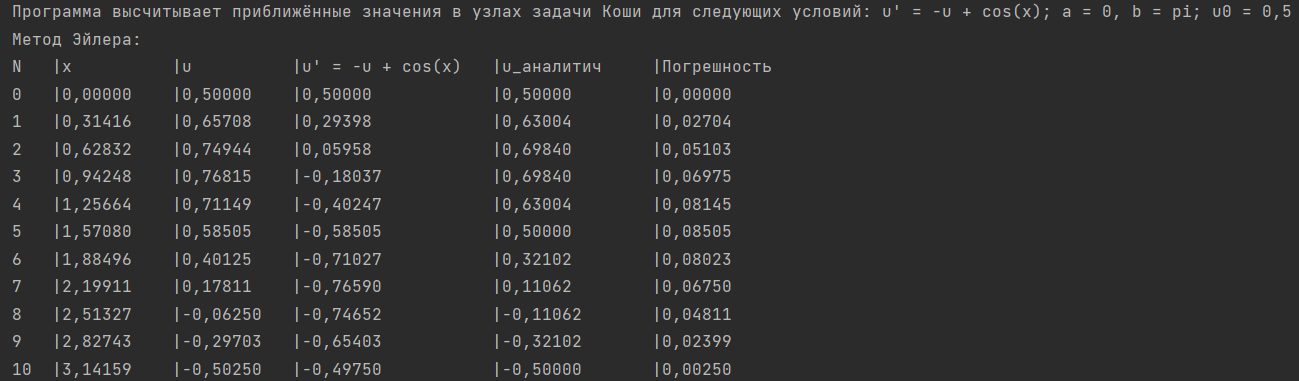
}

**5. Контрольный тест**

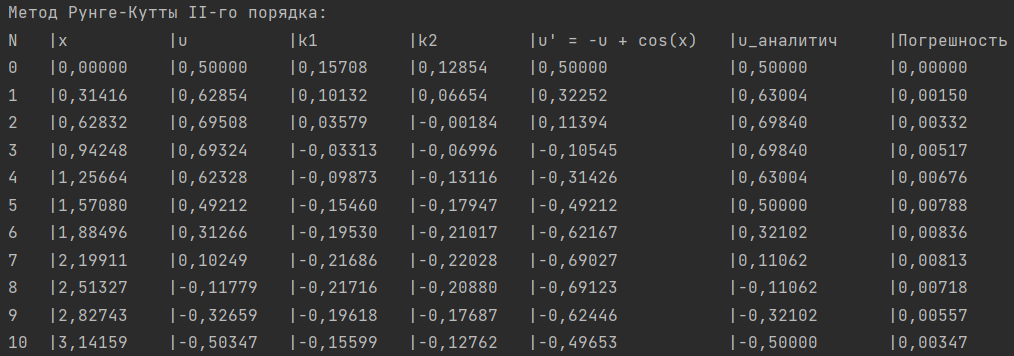
Результаты работы программы:

Программа высчитывает значения для ОДУ I-го порядка. Число разбиений равно 10.

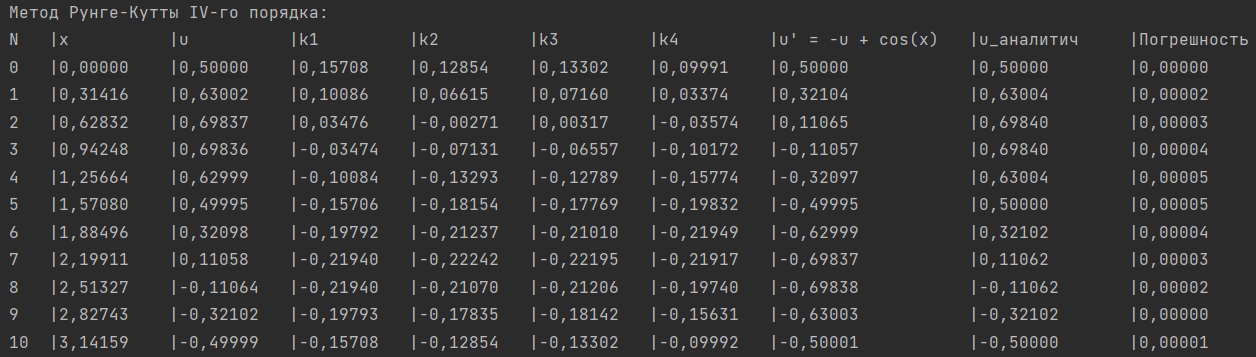
Метод Эйлера (приводится для сравнения с другими методами):



Метод Рунге-Кутты II-го порядка:

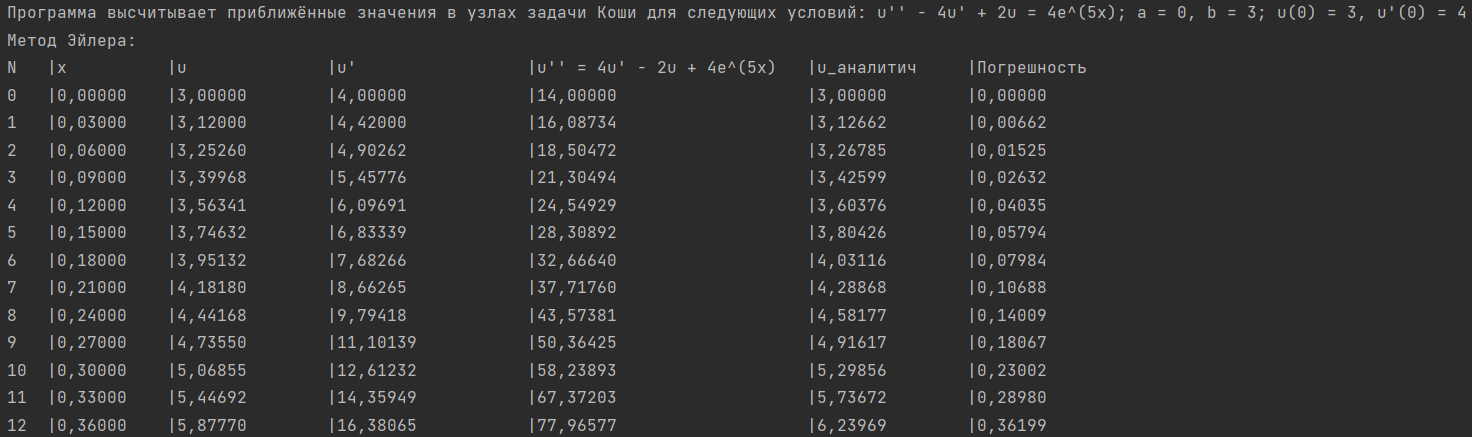


Метод Рунге-Кутты IV-го порядка:

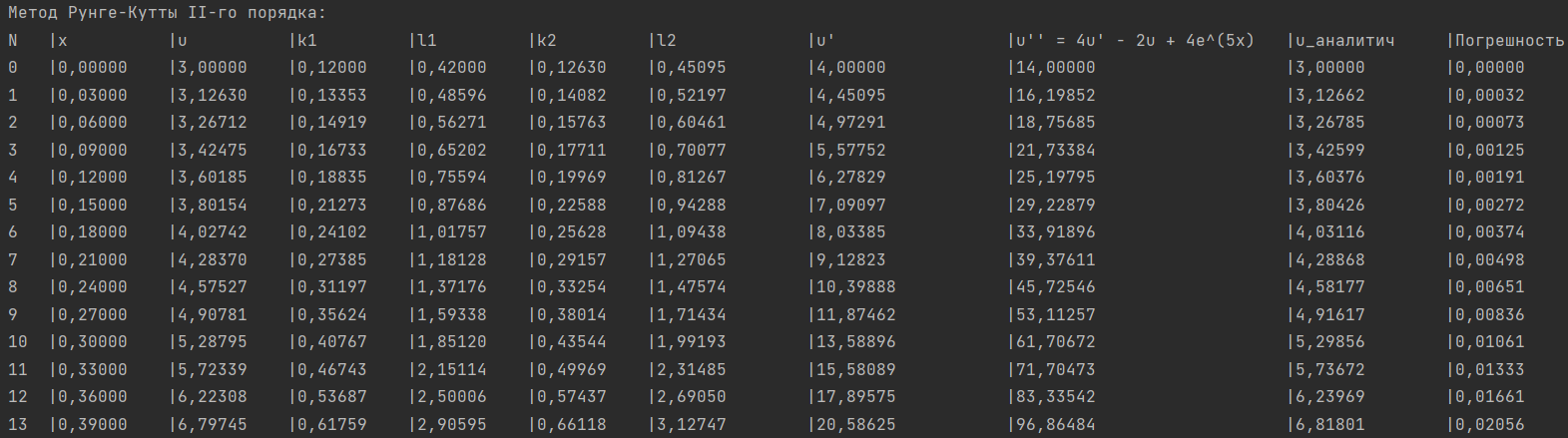


Программа высчитывает значения для ОДУ II-го порядка. Число разбиений равно 100.

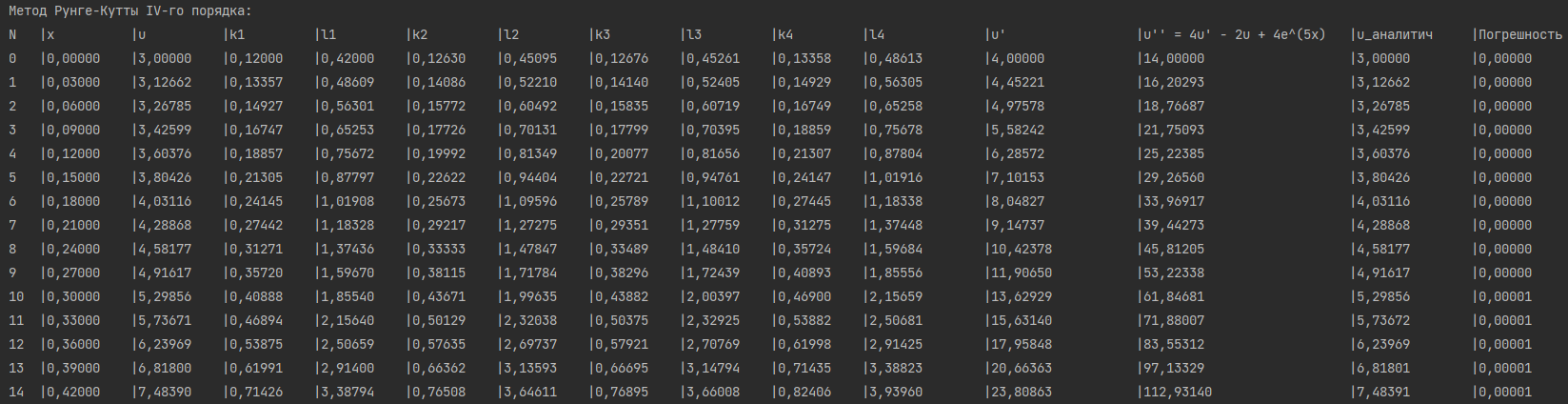
Метод Эйлера:



Метод Рунге-Кутты II-го порядка:



Метод Рунге-Кутты IV-го порядка:



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.

**6. ВЫВОДЫ**

В рамках данной работы были применены основные навыки решения задачи Коши для линейных обыкновенных дифференциальных уравнений первого и второго порядков методом Рунге-Кутта, а также была написана программа, реализующая решение задания на языке Java. Результаты программы и ручного расчета совпали.