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 № 1

Тема «Метод Эйлера для решения задачи Коши»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с-ИБС32  очной формы обучения  Солодилов В.В.  Проверил: доцент кафедры ИБС Кожанова Е.Р. |

Саратов 2021

**Цель работы**: сформировать практические навыки решения задачи Коши для линейных обыкновенных дифференциальных уравнения первого и второго порядков методом Эйлера с их программной реализацией.

**КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

*Метод Эйлера* - наиболее простой численный метод решения (систем) обыкновенных дифференциальных уравнений. Метод Эйлера является явным, одношаговым методом первого порядка точности, основанном на аппроксимации интегральной кривой кусочно-линейной функцией, т. н. ломаной Эйлера.

Пусть дано дифференциальное уравнение первого порядка

(1)

и начальное условие , причем функция непрерывна. Требуется на отрезке приближенно построить интегральную кривую уравнения (1), проходящую через заданную точку .

Разобьем отрезок на *n* равных частей:

Длина каждого отрезка определяется выражением , причем H называют шагом разбиения.

Вычислим в начальной точке угловой коэффициент касательной к искомой интегральной кривой:

Так как функция непрерывна , то можно считать, что на малом участке интегральной кривой ее наклон постоянен, то есть эту кривую приближенно можно заменить ломаной линией.

Заменим на отрезке интегральную кривую отрезком касательной (рис.1). Вычислим приближенно значение в точке .

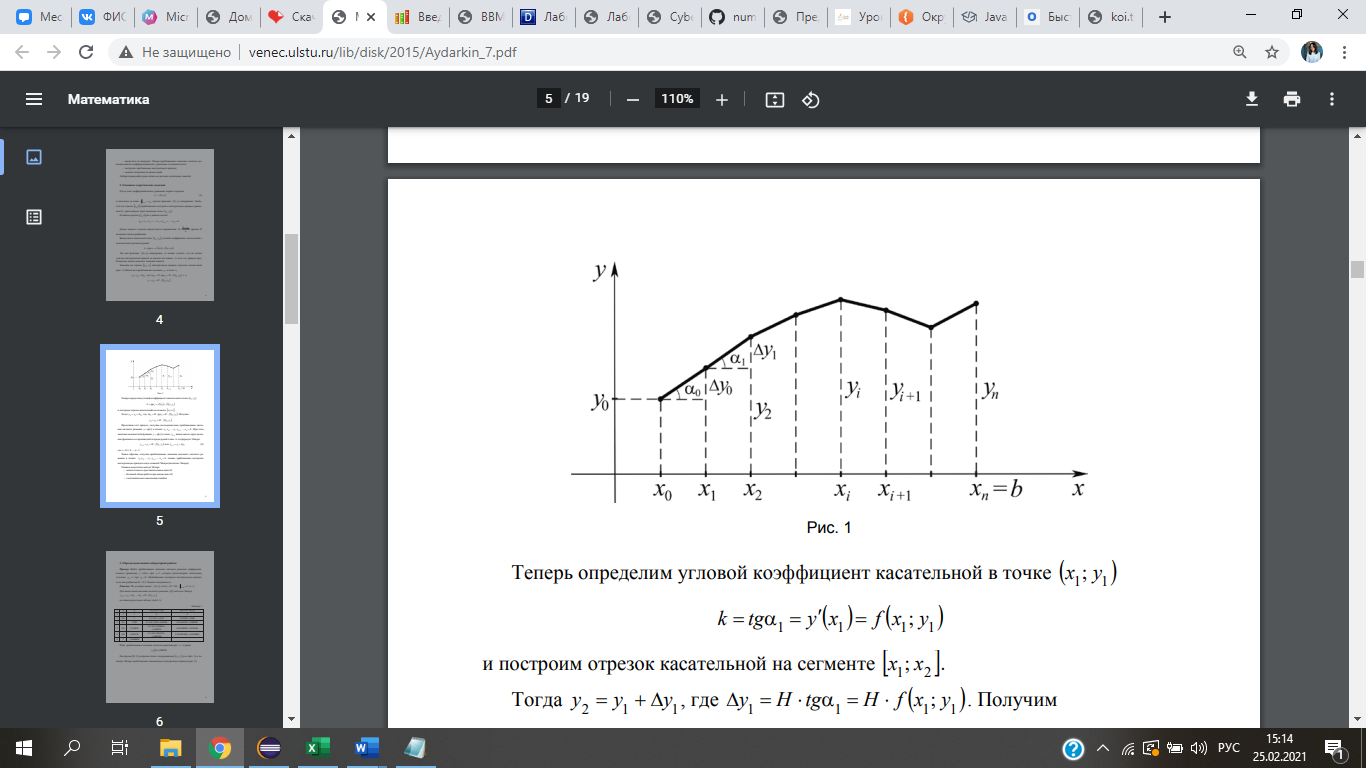


Рис.1 Графическая интерпретация метода Эйлера

Теперь определим угловой коэффициент касательной в точке

и построим отрезок касательной на сегменте .

Тогда

Получим

Продолжая этот процесс, получим последовательно приближенные значения частного решения в точках .

При этом значение неизвестной функции в точке вычисляется через значения функции и ее производной в предыдущей точке по формуле Эйлера:

или (2)

где

Таким образом, получив приближенные значения искомого частного решения в точках , можно приближенно построить интегральную кривую в виде ломаной Эйлера (полигона Эйлера).

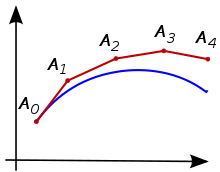


Рис.2 Ломаная Эйлера (красная линия) — приближённое решение в пяти узлах задачи Коши и точное решение этой задачи (выделено синим цветом)

Недостатки метода Эйлера:

* малая точность при значительном шаге H;
* большой объем работы при малом шаге H;
* систематическое накопление ошибок.

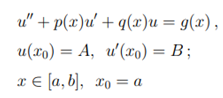
Задача Коши для дифференциального уравнения второго порядка сводится к системе двух уравнений первого порядка.

Получается,

*Задание 1*. Решить задачу Коши для обыкновенного дифференциального уравнения первого порядка методом Рунге – Кутта второго и четвертого порядков точности. Сравнить результаты между собой и с результатами, полученными методом Эйлера:

*Задание 2*. Решить задачу Коши для обыкновенного дифференциального уравнения второго порядка (таблица 2) методом Рунге – Кутта второго и четвертого порядков точности. Сравнить результаты между собой и с результатами, полученными методом Эйлера:

Метод решения:

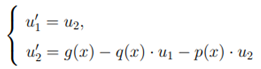


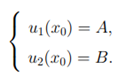
Пусть





Тогда задачу Коши запишем:





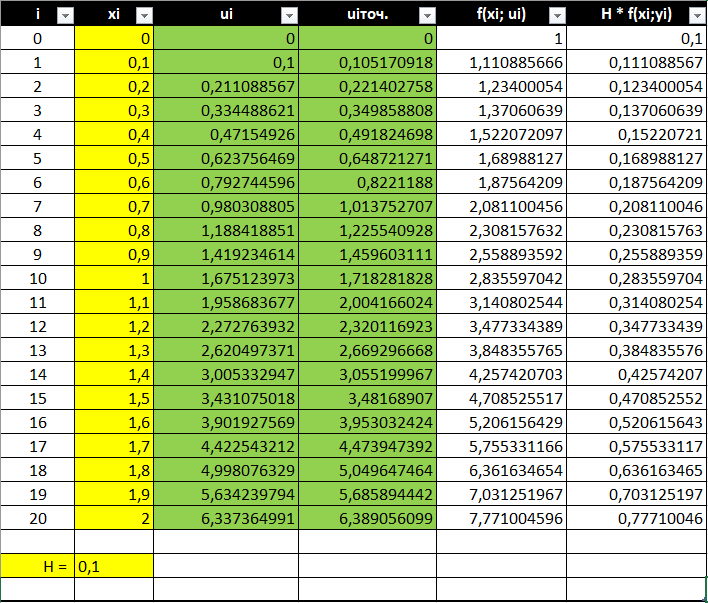
**Расчёт заданий**

*Расчёт Задания 1.*

Для выполнения задания 1 был использован Microsoft Office Excel. Уравнение решено с помощью двух методов:

1. Метод Эйлера

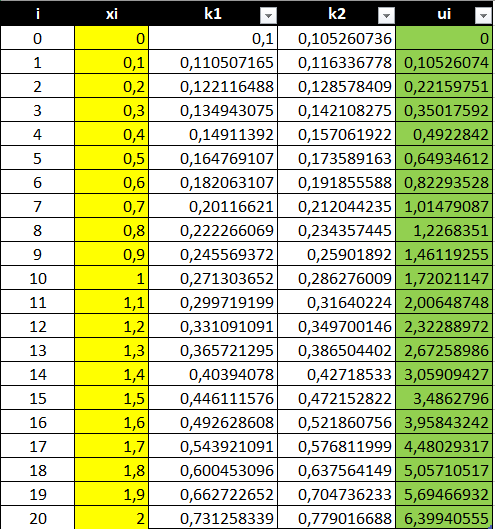
По известным значениям аргумента функции была построена таблица для нахождения значений функции:

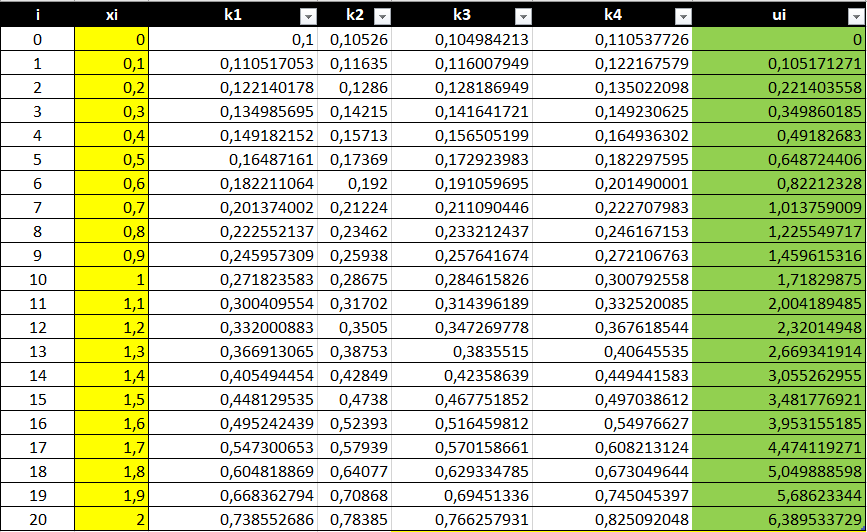


По полученным значениям был построен график функции, найденной методом Эйлера, а также точной аналитической функции:

1. Метод Рунге-Кутта

Для решения ДУ методом Рунге-Кутта второго и четвертого порядков на основе заданных значений аргумента были найдены значения функции, которые были оформлены в виде таблицы:





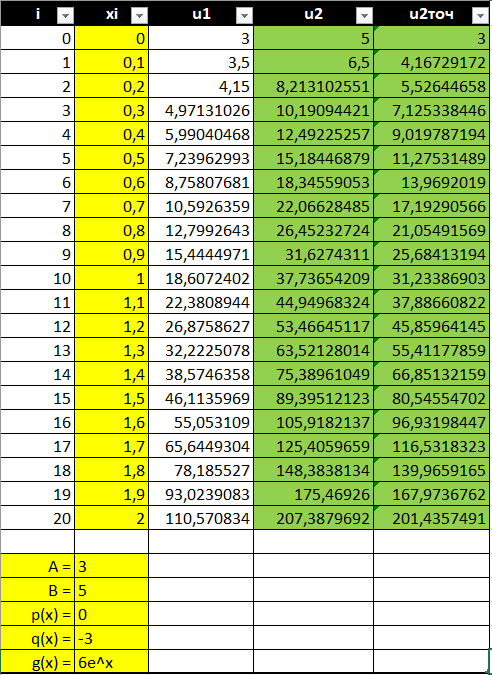
По полученным и ранее известным значениям функции и аргумента были построены графики функции:

На основе полученных результатов можно сказать, что наиболее точным является метод Рунге-Кутта четвертого порядка точности.

*Задание 2.*

1. *Метод Эйлера*

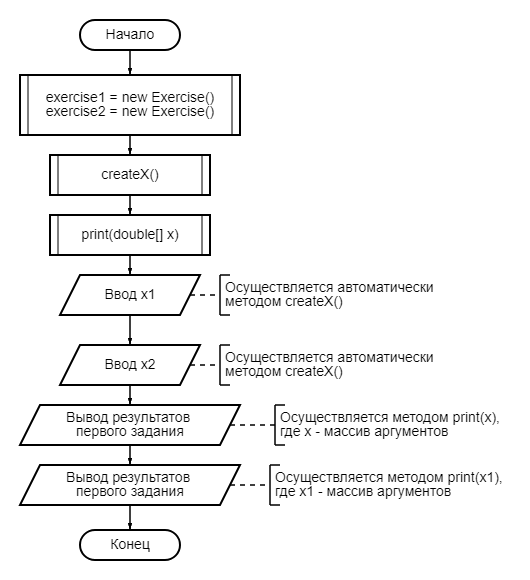
Аналогично, для задания 2 методом Эйлера было произведено численное дифференцирование, что было оформлено в виде таблицы:



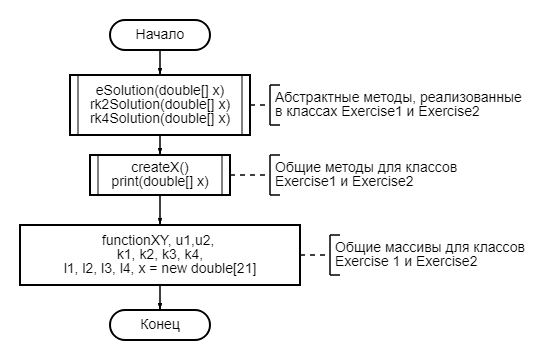
Результат вычислений был вынесен на график вместе с аналитически найденной исходной функцией:

1. Метод Рунге-Кутта

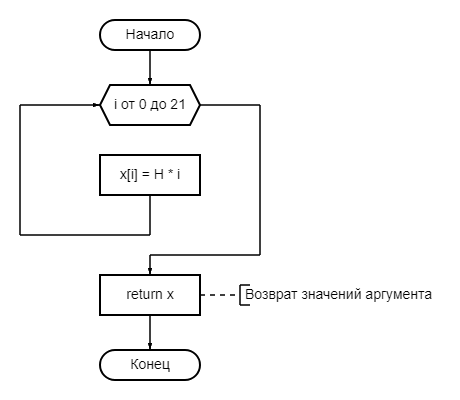
**Блок-схемы**



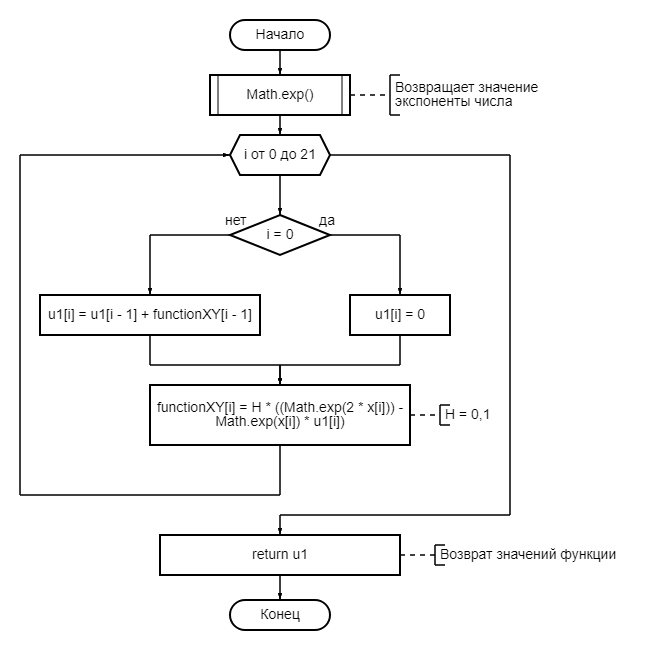
Класс Main()



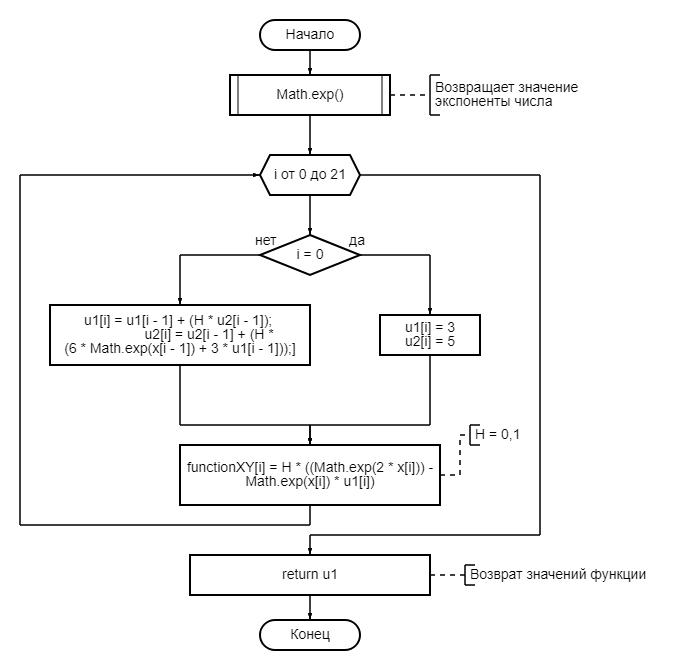
Класс Exercise()



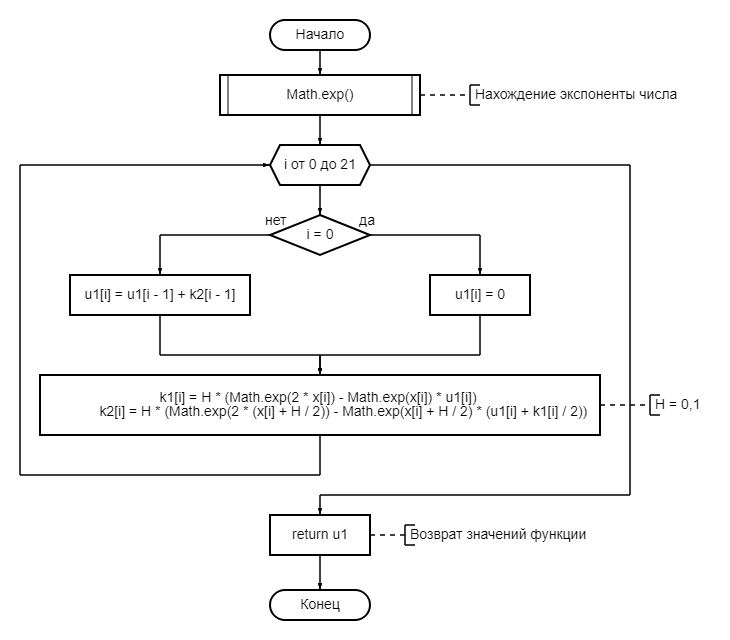
Метод createX()



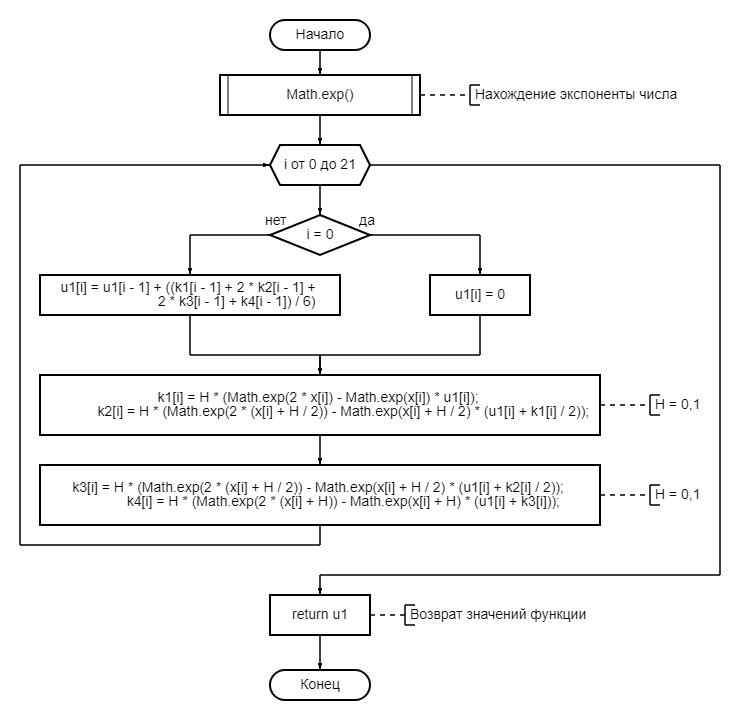
Метод eSolution() класса Exercise1()



Метод eSolution() класса Exercise2()



Метод rk2Solution() класса Exercise1()



Метод rk4Solution() класса Exercise1()

Листинг программы

*//Main.java*

package com.company;  
  
public class Main {  
 public static void main(String[] args) {  
 Exercise exercise1 = new Exercise1();  
 double[] x = exercise1.createX();  
 Exercise exercise2 = new Exercise2();  
 double[] x1 = exercise2.createX();  
  
 System.out.println("Задание 1");  
 exercise1.print(x);  
 System.out.println("\nЗадание 2");  
 exercise2.print(x1);  
 }  
}

*//Exercise.java*

package com.company;  
  
public abstract class Exercise {  
 protected final double H = 0.1;  
 protected double[] functionXY = new double[21];  
 protected double[] u1 = new double[21];  
 protected double[] u2 = new double[21];  
 protected double[] k1 = new double[21];  
 protected double[] k2 = new double[21];  
 protected double[] k3 = new double[21];  
 protected double[] k4 = new double[21];  
 protected double[] l1 = new double[21];  
 protected double[] l2 = new double[21];  
 protected double[] l3 = new double[21];  
 protected double[] l4 = new double[21];  
 protected double[] x = new double[21];  
  
 abstract double[] eSolution(double[] x);  
 abstract double[] rk2Solution(double[] x);  
 abstract double[] rk4Solution(double[] x);  
  
 public double[] createX() {  
 for (int i = 0; i < 21; i++) {  
 x[i] = H \* i;  
 }  
 return x;  
 }  
  
 public void print(double[] x) {  
 System.out.println("\tX\t\t\tМетод Эйлера \t\t\tМетод Рунге-Кутта 2-го порядка\t\tМетод Рунге-Кутта 4-го порядка");  
 for (int i = 0; i < 21; i++) {  
 System.out.format("%5f\t\t ", x[i]);  
 System.out.format("%5f\t\t\t\t\t\t", eSolution(x)[i]);  
 System.out.format("%5f\t\t\t\t\t\t\t", rk2Solution(x)[i]);  
 System.out.format("%5f\t\t\t\t\n", rk4Solution(x)[i]);  
 }  
 }  
}

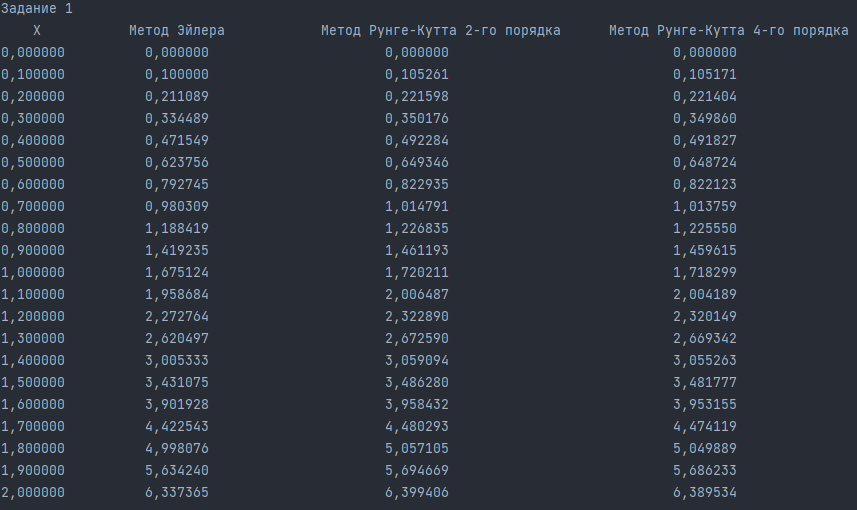
*//Exercise1.java*

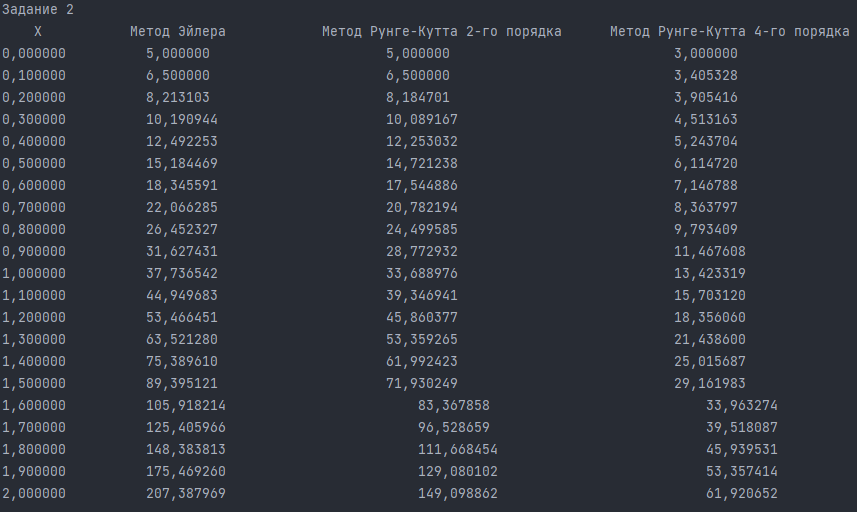
package com.company;  
  
public class Exercise1 extends Exercise {  
 @Override  
 public double[] eSolution(double[] x) {  
 for (int i = 0; i < 21; i++) {  
 if (i == 0) {  
 u1[i] = 0;  
 } else {  
 u1[i] = u1[i - 1] + functionXY[i - 1];  
 }  
 functionXY[i] = H \* (Math.exp(2 \* x[i]) - Math.exp(x[i]) \* u1[i]);  
 }  
 return u1;  
 }  
  
 @Override  
 public double[] rk2Solution(double[] x) {  
 for (int i = 0; i < 21; i++) {  
 if (i == 0) {  
 u1[i] = 0;  
 } else {  
 u1[i] = u1[i - 1] + k2[i - 1];  
 }  
  
 k1[i] = H \* (Math.exp(2 \* x[i]) - Math.exp(x[i]) \* u1[i]);  
 k2[i] = H \* (Math.exp(2 \* (x[i] + H / 2)) -  
 Math.exp(x[i] + H / 2) \* (u1[i] + k1[i] / 2));  
 }  
 return u1;  
 }  
  
 @Override  
 public double[] rk4Solution(double[] x) {  
 for (int i = 0; i < 21; i++) {  
 if (i == 0) {  
 u1[i] = 0;  
 } else {  
 u1[i] = u1[i - 1] + ((k1[i - 1] + 2 \* k2[i - 1] +  
 2 \* k3[i - 1] + k4[i - 1]) / 6);  
 }  
  
 k1[i] = H \* (Math.exp(2 \* x[i]) - Math.exp(x[i]) \* u1[i]);  
 k2[i] = H \* (Math.exp(2 \* (x[i] + H / 2)) -  
 Math.exp(x[i] + H / 2) \* (u1[i] + k1[i] / 2));  
 k3[i] = H \* (Math.exp(2 \* (x[i] + H / 2)) -  
 Math.exp(x[i] + H / 2) \* (u1[i] + k2[i] / 2));  
 k4[i] = H \* (Math.exp(2 \* (x[i] + H)) -  
 Math.exp(x[i] + H) \* (u1[i] + k3[i]));  
 }  
 return u1;  
 }  
}

*//Exercise2.java*

package com.company;  
  
public class Exercise2 extends Exercise {  
 @Override  
 double[] eSolution(double[] x) {  
 for (int i = 0; i < 21; i++) {  
 if (i == 0) {  
 u1[i] = 3;  
 u2[i] = 5;  
 } else {  
 u1[i] = u1[i - 1] + (H \* u2[i - 1]);  
 u2[i] = u2[i - 1] + (H \* (6 \* Math.exp(x[i - 1]) + 3 \* u1[i - 1]));  
 }  
 }  
 return u2;  
 }  
  
 @Override  
 double[] rk2Solution(double[] x) {  
 for (int i = 0; i < 21; i++) {  
 if (i == 0) {  
 u1[i] = 3;  
 u2[i] = 5;  
 } else {  
 u1[i] = u1[i - 1] + ((k1[i - 1] + k2[i - 1]) / 2);  
 u2[i] = u2[i - 1] + (H \* (6 \* Math.exp(x[i - 1]) +  
 3 \* u1[i - 1]));  
 }  
 k1[i] = H \* u2[i];  
 l1[i] = H \* (6 \* Math.exp(x[i]) + 3 \* u1[i]);  
 l2[i] = H \* (6 \* Math.exp(x[i] + H) + 3 \* (u1[i] + k1[i]));  
 k2[i] = H \* (((l1[i] + l2[i]) / 2) + l1[i]);  
 }  
 return u2;  
 }  
  
 @Override  
 double[] rk4Solution(double[] x) {  
  
 return u1;  
 }  
}

Результат программы





Выводы