|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 1**

|  |  |
| --- | --- |
| **Тема Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ**  **Студент Якуба Д. В.**  **Группа ИУ7-63Б**  **Оценка (баллы)**  **Преподаватель Градов В. М.** |  |

Москва.

2021 г.

Лабораторная работа по теме «Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ»

Тема:

Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ.

Цель работы:

Получение навыков решения задачи Коши для ОДУ методами Пикара и явными методами первого порядка точности (Эйлера) второго порядка точности (Рунге-Кутта).

Задание:

Задано ОДУ, не имеющее аналитического решения:

Требуется построить таблицу, содержащую значения аргумента с заданным шагом в интервале [0, ] и результаты расчета функции в приближениях Пикара (от 1-го до 4-го), а также численными методами. Границу интервала выбирать максимально возможной из условия, чтобы численные методы обеспечивали точность вычисления решения уравнения до второго знака после запятой.

Входные данные:

Конечное значение , шаг.

Выходные данные:

Таблица, содержащая значения аргумента с заданным шагом в интервале [0, ] и результаты расчета функции в приближениях Пикара (от 1-го до 4-го), а также численными методами.

Описание

Обыкновенными дифференциальными уравнениями (ОДУ) называются уравнения с одной независимой переменной. Если независимых переменных больше, чем одна, то уравнение называется дифференциальным уравнением с частными производными.

Задача Коши

Общее решения ДУ n-го порядка зависит от констант общего решения. Для выделения частного решения требуется задать n условий.

В задаче Коши все дополнительные условия задаются в одной точке:

Можно выделить три метода решения обыкновенных дифференциальных уравнений в задаче Коши: аналитические, аналитические приближенные и численные.

Метод Пикара

Данный метод является представителем приближенных методов решения рассматриваемого класса задач. Идея метода чрезвычайно проста и сводится к процедуре последовательных приближений для решения интегрального уравнения, к которому приводится исходное дифференциальное уравнение. Пусть поставлена задача Коши:

Проинтегрируем выписанное уравнение:

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме:

причём

Метод Эйлера

Метод Эйлера — простейший численный метод решения систем обыкновенных дифференциальных уравнений.

Метод Эйлера в явном виде представлен формулой:

*где*

Метод Рунге-Кутта 2-го порядка

Методы Рунге-Кутта являются численными. На практике применяются методы Рунге-Кутта, обеспечивающие построение разностных схем различного порядка точности. Наиболее употребительны схемы второго и четвертого порядков.

Используя формулу Тейлора, решение дифференциального уравнения можно представить в виде:

где обозначено

Согласно . Далее удерживаем только выписанные члены ряда. Представим вторую производную следующим образом:

Пусть

Обозначим приближенное значение решения в узле с номером через .

Имеем:

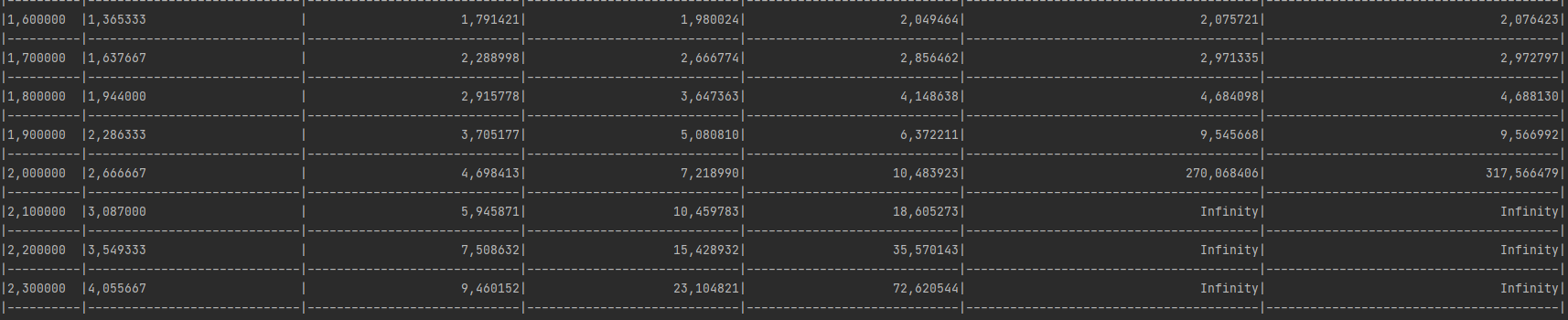
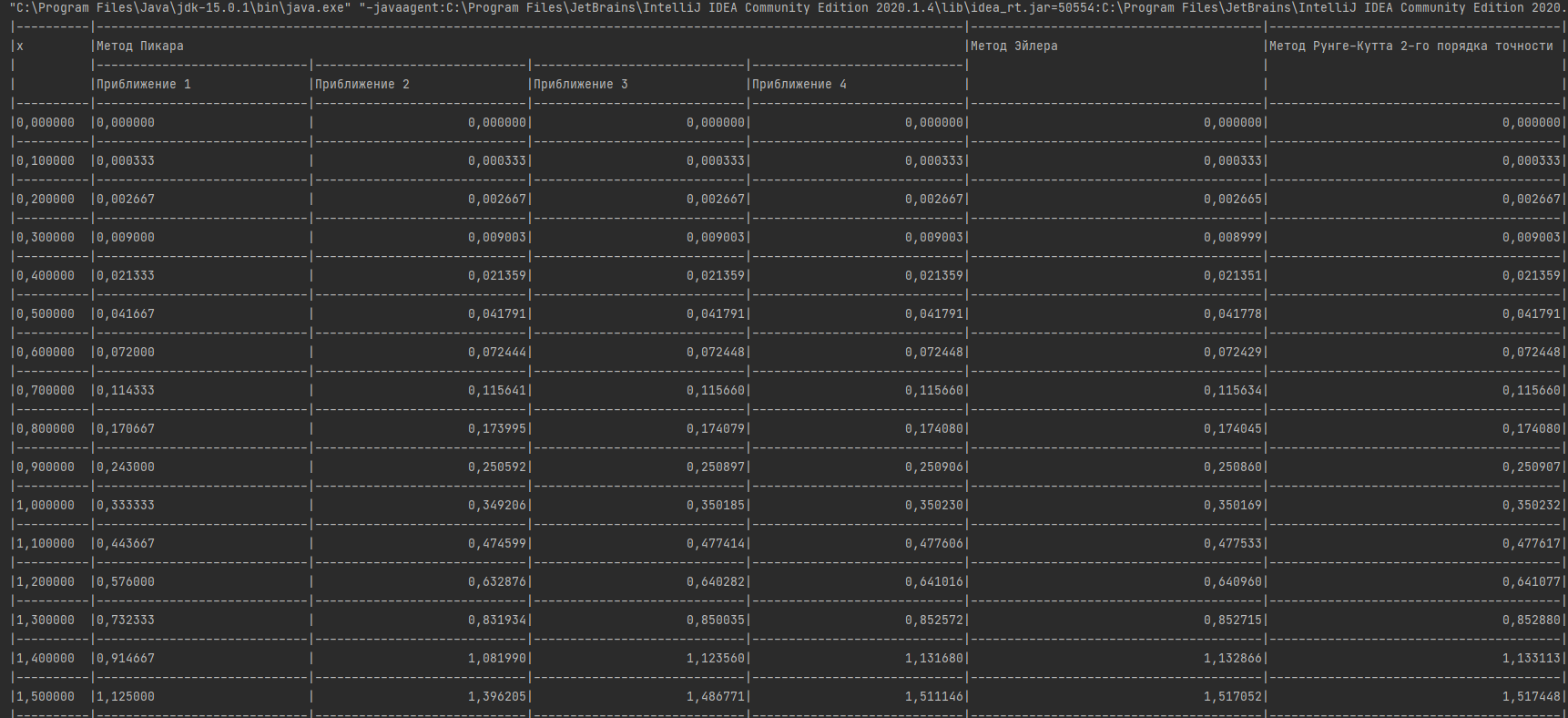
Введенные здесь параметры подлежат определению. Разлагая правую часть в ряд Тейлора до линейных членом и приводя подобные члены, получим последовательно:

Условием выбора параметров поставим близость выражения (2) ряду (1), тогда:

Один параметр остается свободным. Пусть это будет , тогда:

Таким образом получим:

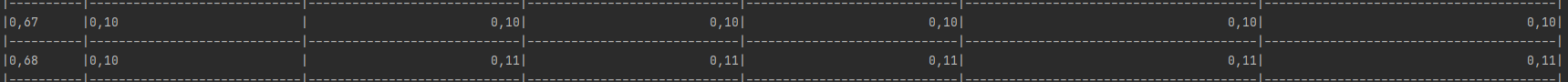
Результат

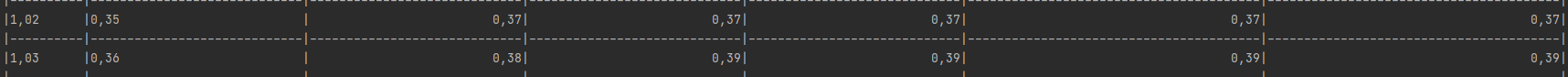
Вычисления производились с шагом , для демонстрации выводится каждый 1000-ый результат.

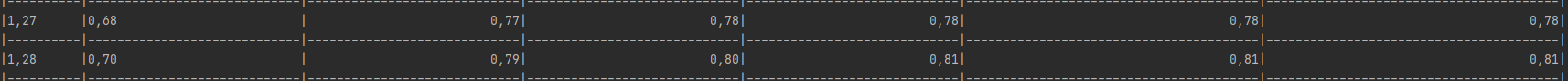
Контрольные вопросы

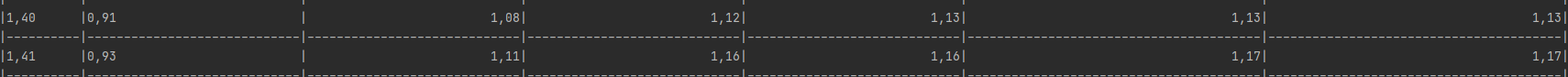
1. Укажите интервалы значений аргумента, в которых можно считать решением заданного уравнения каждое из первых 4-х приближений Пикара. Точность результата оценивать до второй цифры после запятой. Объяснить свой ответ.

Ответ: для нахождения интервалов значений аргумента, в которых можно считать решение заданного уравнения каждое из первых 4-х приближений Пикара потребуется уменьшать шаг и проводить сравнительный анализ для разных приближений. Интервал для некоторого приближения будет включать в себя все значения аргумента, при которых полученные решения методом Пикара будут совпадать со значениями (до второй цифры, как это указано в вопросе) более высоких порядков приближения и рассмотренных численных методов при шаге меньшим или равным .

Имеем:

Интервал для первого приближения:

Интервал для второго приближения:

Интервал для третьего приближения:

Интервал для четвёртого приближения:

2. Пояснить, каким образом можно доказать правильность полученного результата при фиксированном значении аргумента в численных методах.

Ответ: доказать правильность полученного результата при фиксированном значении аргумента в численных методах можно посредством постепенного уменьшения шага. Если при уменьшении шага полученный результат изменится незначительно (относительно предыдущих изменений), полученный результат можно считать правильным.

3. Каково значение функции при x=2, то есть привести значение u(2).

Ответ: опираясь на вопрос №2, привести значение можно посредством уменьшения шага и анализом изменения полученного численным методом значения.

Для ;

Для ;

Разница с предыдущим значением: 161,294611.

Для ;

Разница с предыдущим значением: 12,358452.

Для ;

Разница с предыдущим значением: 0,154397.



Для ;

Разница с предыдущим значением: 0,001568.

Таким образом:

Код программы

import kotlin.math.abs  
import kotlin.math.ceil  
import kotlin.math.pow  
  
const val *xF* = 10  
const val *picF* = 120  
const val *picFfo* = *picF* / 4 - 1  
const val *eulF* = 40  
const val *rkF* = 40  
  
fun firstApprox(x: Double) : Double  
{  
 return x \* x \* x / 3  
}  
  
fun secondApprox(x: Double) : Double  
{  
 return *firstApprox*(x) + x.*pow*(7) / 63  
}  
  
fun thirdApprox(x: Double) : Double  
{  
 return *secondApprox*(x) + x.*pow*(11) \* 2 / 2079 + x.*pow*(15) / 59535  
}  
  
fun fourthApprox(x: Double) : Double  
{  
 return *secondApprox*(x) + 2 \* x.*pow*(11) / 2079 + 13 \* x.*pow*(15) / 218295 + 82 \* x.*pow*(19) / 37328445 +  
 662 \* x.*pow*(23) / 10438212015 + 4 \* x.*pow*(27) / 3341878155 +  
 x.*pow*(31) / 109876902975  
}  
  
fun picardMet(xStart: Double, step: Double, numOfIters: Int) : MutableList<MutableList<Double>>  
{  
 val outList: MutableList<MutableList<Double>> = *mutableListOf*()  
  
 var curX = xStart  
  
 outList.add(*mutableListOf*(curX, *firstApprox*(curX), *secondApprox*(curX), *thirdApprox*(curX), *fourthApprox*(curX)))  
  
 for (i in 1..numOfIters)  
 {  
 curX += step  
 outList.add(*mutableListOf*(curX, *firstApprox*(curX), *secondApprox*(curX), *thirdApprox*(curX), *fourthApprox*(curX)))  
 }  
  
 return outList  
}  
  
fun curMathFun(x: Double, y: Double) : Double { return x \* x + y \* y}  
  
fun eulerMet(xStart: Double, yStart: Double, step: Double, numOfIters: Int) : MutableList<Double>  
{  
 val outList: MutableList<Double> = *mutableListOf*()  
  
 var curX = xStart  
 var curY = yStart  
  
 outList.add(curY)  
 for (i in 1..numOfIters)  
 {  
 curY += step \* *curMathFun*(curX, curY)  
  
 outList.add(curY)  
  
 curX += step  
 }  
  
 return outList  
}  
  
fun rungeKutMet(xStart: Double, yStart: Double, alpha: Double, step: Double, numOfIters: Int) : MutableList<Double>  
{  
 val outList: MutableList<Double> = *mutableListOf*()  
  
 var curX = xStart  
 var curY = yStart  
  
 outList.add(curY)  
 for (i in 1..numOfIters)  
 {  
 curY += step \* ((1 - alpha) \* *curMathFun*(curX, curY) +  
 alpha \* *curMathFun*(curX + step / (2 \* alpha),curY + step \* *curMathFun*(curX, curY) / (2 \* alpha)))  
  
 outList.add(curY)  
  
 curX += step  
 }  
  
 return outList  
}  
  
fun printHead()  
{  
 *print*(String.*format*("|%-${*xF*}s|%-${*picF* - 1}s|%-${*eulF*}s|%-${*rkF*}s|\n", "", "", "", "").*replace*(' ', '-'))  
 *print*(String.*format*("|%-${*xF*}s|%-${*picF* - 1}s|%-${*eulF*}s|%-${*rkF*}s|\n", "x", "Метод Пикара", "Метод Эйлера", "Метод Рунге-Кутта 2-го порядка точности"))  
  
 *print*(String.*format*("|%${*xF*}s|", " "))  
 for (i in 1..4)  
 *print*(String.*format*("%-${*picFfo*}s|", "").*replace*(' ', '-'))  
 *println*(String.*format*("%${*eulF*}s|%${*rkF*}s|", "", ""))  
  
 *print*(String.*format*("|%${*xF*}s|", " "))  
 for (i in 1..4)  
 *print*(String.*format*("%-${*picFfo*}s|", "Приближение $i"))  
 *print*(String.*format*("%${*eulF*}s|%${*rkF*}s|\n", "", ""))  
  
 *print*(String.*format*("|%${*xF*}s|", " ").*replace*(' ', '-'))  
 for (i in 1..4)  
 *print*(String.*format*("%-${*picFfo*}s|", "").*replace*(' ', '-'))  
 *println*(String.*format*("%${*eulF*}s|%${*rkF*}s|", "", "").*replace*(' ', '-'))  
}  
  
fun printAnswers(picAnsw: MutableList<MutableList<Double>>, eulAnsw: MutableList<Double>, runAnsw: MutableList<Double>)  
{  
 for (i in 0 *until* picAnsw.size *step* 100)  
 {  
 val curPic = picAnsw[i]  
 *println*(  
 String.*format*(  
 "|%-${*xF*}f|%-${*picFfo*}f|%${*picFfo*}f|%${*picFfo*}f|%${*picFfo*}f|%${*eulF*}f|%${*rkF*}f|",  
 curPic[0], curPic[1], curPic[2], curPic[3], curPic[4], eulAnsw[i], runAnsw[i]  
 )  
 )  
 *print*(String.*format*("|%${*xF*}s|", " ").*replace*(' ', '-'))  
 for (i in 1..4)  
 *print*(String.*format*("%-${*picFfo*}s|", "").*replace*(' ', '-'))  
 *println*(String.*format*("%${*eulF*}s|%${*rkF*}s|", "", "").*replace*(' ', '-'))  
 }  
}  
  
fun main()  
{  
 val xStart = 0.0  
 val xStop = 2.0  
 val yStart = 0.0  
 val step = 10e-5  
  
 val iters = *ceil*(*abs*(xStop - xStart) / step).toInt()  
  
 val picAnsw = *picardMet*(xStart, step, iters)  
 val eulAnsw = *eulerMet*(xStart, yStart, step, iters)  
 val runAnsw = *rungeKutMet*(xStart, yStart, 0.5, step, iters)  
  
 *printHead*()  
 *printAnswers*(picAnsw, eulAnsw, runAnsw)  
}