Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №6**

по «Вычислительной математике»

Решение СЛАУ

Вариант 4

Выполнил:

Студент группы P3213

Ваховский П.А.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2022

**Цель работы**: Цель лабораторной работы: решить задачу Коши численными методами.

Для исследования использовать:

* Одношаговые методы;
* Многошаговые методы.

**Задание**

1. Программная реализация задачи:

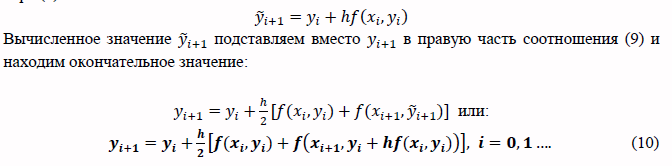
2.1. Исходные данные: ОДУ вида , начальные условия , интервал дифференцирования [*a, b*], шаг *h*, точность .

2.2. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям. Для оценки точности использовать правило Рунге.

2.3. Построить графики точного решения и полученного численного решения (разными цветами).

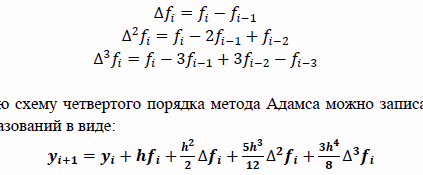
**Описание методов, рабочие формулы:**

**Модифицированный метод Эйлера:**





Метод Адамса:



**Листинг класса ImprovedEulerMethod:**

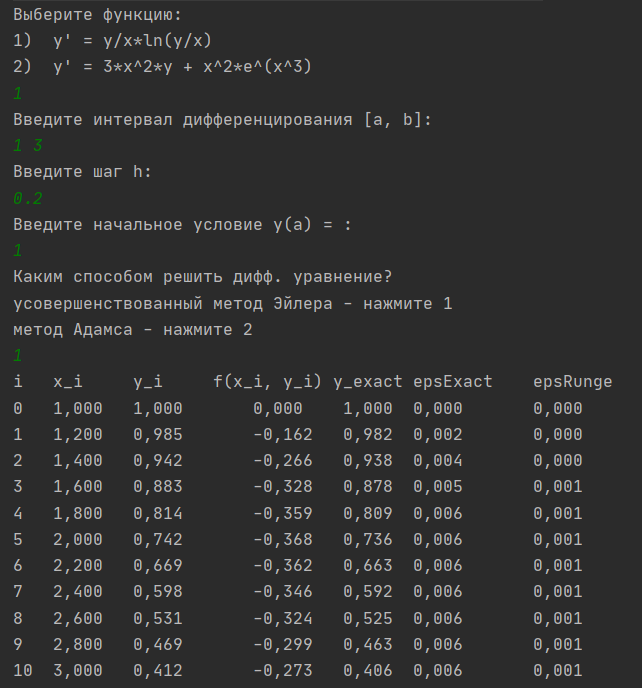
package main.java.solvers;  
  
import main.java.DiffusionEquation;  
  
import java.util.ArrayList;  
  
public class ImprovedEulerMethod extends SolveMethod{  
  
 public ImprovedEulerMethod(DiffusionEquation equation, double a, double b, double h) {  
 super(equation, a, b, h);  
 }  
  
 private double eulerFormul(double x\_prev, double y\_prev, double h){  
 return y\_prev+h\*equation.getEquationFunction()  
 .calcFunction(x\_prev, y\_prev);  
 }  
  
 public ArrayList<Solution> calc(boolean hHalf){  
 double h;  
 ArrayList<Solution> solution;  
 if(hHalf){  
 h = this.h/2;  
 solution = hHalfSolutions;  
 }  
 else{  
 h = this.h;  
 solution = hSolution;  
 }  
  
 double yExact = equation.getExactFunction().calcFuntion(a);  
 double yMethod = yExact;  
 double eqFunction = equation.getEquationFunction().calcFunction(a, yMethod);  
 solution.add(new Solution(0, a, yMethod, eqFunction, yExact, Math.*abs*(yExact - yMethod)));  
 for(int i = 1; i <= (b-a)/h; i++){  
 double x = a+(double)i\*h;  
 yExact = equation.getExactFunction().calcFuntion(x);  
 yMethod = yMethod+h/2\* (eqFunction +  
 equation.getEquationFunction().calcFunction(x, eulerFormul(x-h, yMethod, h)));  
 eqFunction = equation.getEquationFunction().calcFunction(x, yMethod);  
 solution.add(new Solution(i, x, yMethod, eqFunction, yExact, Math.*abs*(yExact - yMethod)));  
 }  
 return solution;  
 }  
  
}

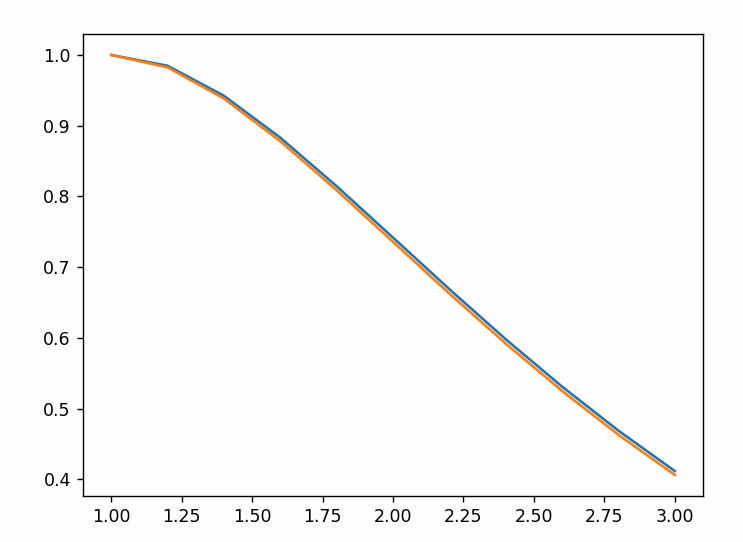
}

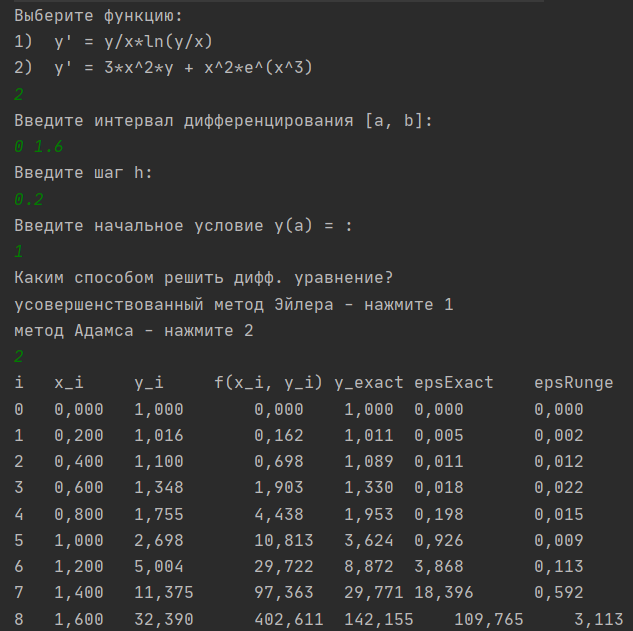
**AdamsMethod**

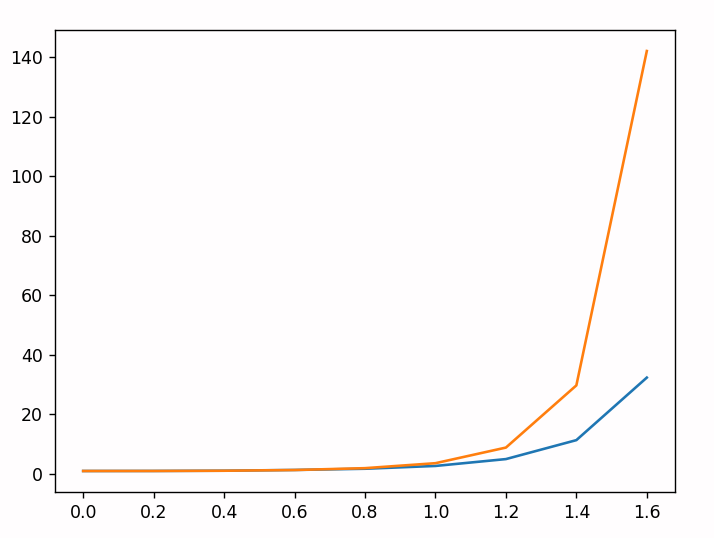
package main.java.solvers;  
  
import main.java.DiffusionEquation;  
  
import java.util.ArrayList;  
  
public class AdamsMethod extends SolveMethod{  
  
 public AdamsMethod(DiffusionEquation equation, double a, double b, double h) {  
 super(equation, a, b, h);  
 }  
  
 public ArrayList<Solution> calc(boolean hHalf){  
 double h;  
 ArrayList<Solution> solution;  
 if(hHalf){  
 h = this.h/2;  
 hHalfSolutions = new ImprovedEulerMethod(equation, a, b, h).calc(true);  
 solution = hHalfSolutions;  
 }  
 else{  
 h = this.h;  
 hSolution = new ImprovedEulerMethod(equation, a, b, h).calc(false);  
 solution = hSolution;  
 }  
  
 for(int i = 4; i <= (b-a)/h+0.001; i++){  
 double df = solution.get(i-1).f-solution.get(i-2).f;  
 double ddf = solution.get(i-1).f-2\*solution.get(i-2).f+solution.get(i-3).f;  
 double dddf = solution.get(i-1).f-3\*solution.get(i-2).f+3\*solution.get(i-3).f+solution.get(i-4).f;  
  
 double y\_i = solution.get(i-1).y\_i + h\*solution.get(i-1).f  
 + h\*h/2\*df + 5\*h\*h\*h/12\*ddf + 3\*h\*h\*h\*h/8\*dddf;  
 double f\_i = equation.getEquationFunction().calcFunction(a+(double)i\*h, y\_i);  
  
 solution.get(i).y\_i = y\_i;  
 solution.get(i).f = f\_i;  
 solution.get(i).epsExact = Math.*abs*(y\_i-solution.get(i).yExact);  
 }  
 return solution;  
 }  
  
  
}

**Примеры:**









**Вывод: приближенные методы вычисления дают хорошую точность в случае, если скорость изменения функции решения между двумя узлами небольшая.**