**ФПИиКТ**

**Рабочий протокол и отчет по**

**домашней работе №6**

**Вариант №6**

Ибадуллаев Алибаба Эльбрус оглы

Группа: P3215

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург

2022г.

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

**Задание:**

Для исследования использовать:

* Одношаговые методы
* Многошаговые методы

1. Программная реализация задачи:

𝑦(𝑥")

𝑦! = 𝑓(𝑥, 𝑦), начальные условия

1.1. Исходные данные: ОДУ вида

, интервал дифференцирования [*a, b*], шаг *h*, точность 𝜀.

1.2. Составить таблицу приближенных значений интеграла

дифференциального уравнения, удовлетворяющего начальным

условиям. Для оценки точности использовать правило Рунге.

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

**Процесс выполнения работы:**

**Код программы (Java):**

**Main:**

public static void main(String[] args) throws IOException {  
 RandomAccessFile accessFile = new RandomAccessFile(new File("index.html"), "rw");  
 IO io = new IO();  
 //<a> <b> <h> <y0> <epsilon>  
 double[] input\_data = io.readData();  
 if (input\_data == null) {  
 System.*err*.println("Input Error!");  
 System.*exit*(-1);  
 }  
  
 double a = input\_data[0];  
 double b = input\_data[1];  
 double h = input\_data[2];  
 double y0 = input\_data[3];  
 double epsilon = input\_data[4];  
 int after\_point = *digitsAfterPoint*(epsilon);  
 *eilerMethod*(a,b,y0,h,after\_point,epsilon,accessFile);  
 *milnMethod*(a,b,y0,h,after\_point,epsilon,accessFile);  
 String text =  
 " </script>\n" +  
 "</body>\n" +  
 "</html>";  
 accessFile.seek(accessFile.length());  
 accessFile.write(text.getBytes());  
  
 File htmlFile = new File("index.html");  
 Desktop.*getDesktop*().browse(htmlFile.toURI());  
  
}

**Eiler method:**

private static void eilerMethod(double a, double b, double y0, double h, int after\_point, double epsilon, RandomAccessFile accessFile) throws IOException {  
 ArrayList<Double> y = new ArrayList<>();  
 ArrayList<Double> y1 = new ArrayList<>();  
 ArrayList<Double> x = new ArrayList<>();  
 while (Math.*abs*(*forRunge*(a, y0, h, 1) - *forRunge*(a,y0,2\*h,1))/(Math.*pow*(2,*ТОЧНОСТЬ\_ЭЙЛЕР*)-1) >= epsilon) {  
 int n = (int) ((b - a) / h);  
 double temp\_y = y0;  
 double temp\_x = a;  
 System.*out*.println("h = " + h);  
  
 y = new ArrayList<>();  
 y1 = new ArrayList<>();  
 x = new ArrayList<>();  
  
 System.*out*.format("%."+after\_point+"f >= %."+after\_point+"f\n", Math.*abs*(*forRunge*(a, y0, h, 1) - *forRunge*(a,y0,2\*h,1))/(Math.*pow*(2,*ТОЧНОСТЬ\_ЭЙЛЕР*)-1), epsilon);  
// System.out.println("x \t\t y \t\t f(x,y)");  
 for (int i = 0; i <= n; i++) {  
 x.add(temp\_x);  
 y.add(temp\_y);  
 y1.add(*func*(temp\_x, temp\_y));  
// System.out.format(new StringBuilder().append("%.").append(after\_point).append("f \t %.").append(after\_point).append("f\t %.").append(after\_point).append("f\n").toString(), temp\_x, temp\_y, func(temp\_x, temp\_y));  
 temp\_y = temp\_y + h \* *func*(temp\_x, temp\_y);  
 temp\_x = temp\_x + h;  
 }  
 h=h/2;  
 System.*out*.println("--------------------------");  
 }  
 String x\_text\_1 = Arrays.*toString*(x.toArray());  
 String y\_text\_1 = Arrays.*toString*(y1.toArray());  
  
 String text =  
 "\tcalculator.setExpression({\n" +  
 " type: 'table',\n" +  
 " columns: [\n" +  
 " {\n" +  
 " latex: 'x',\n" +  
 " values: "+x\_text\_1+",\n" +  
 " },\n" +  
 " {\n" +  
 " latex: 'y',\n" +  
 " values: "+y\_text\_1+",\n" +  
 " color: Desmos.Colors.RED,\n" +  
 " columnMode: Desmos.ColumnModes.LINES\n" +  
 " }\n" +  
 " ]\n" +  
 "});\n";  
 accessFile.seek(accessFile.length());  
 accessFile.write(text.getBytes());  
 }

**Miln method:**

private static void milnMethod(double a, double b, double y0, double h, int after\_point, double epsilon, RandomAccessFile accessFile) throws IOException {  
 ArrayList<ArrayList<Double>> pointsMiln = new ArrayList<ArrayList<Double>>(2);  
  
 int n = (int) ((b - a) / h);  
 System.*out*.println("h = " + h);  
 System.*out*.format("%."+after\_point+"f >= %."+after\_point+"f\n", Math.*abs*(*forRunge*(a, y0, h, 1) - *forRunge*(a,y0,2\*h,1))/(Math.*pow*(2,*ТОЧНОСТЬ\_МИЛНА*)-1), epsilon);  
 System.*out*.println("x \t\t y \t\t f(x,y)");  
 pointsMiln = *getKoshiSolution*(a,y0,h,3, after\_point);  
 ArrayList<Double> y = pointsMiln.get(0);  
 ArrayList<Double> y1 = pointsMiln.get(1);  
 ArrayList<Double> x = pointsMiln.get(2);  
 double temp\_x = a + 3\*h;  
 double A = 0;  
 double B = 0;  
 for (int i = 3; i < n; i++){  
 // predict  
 y.add(i+1, y.get(i-3) + 4d/3d \* h \* (2 \* y1.get(i) - y1.get(i-1) + 2 \* y1.get(i-2)));  
 temp\_x += h;  
 x.add(i+1, temp\_x);  
 B = y.get(i+1);  
 // correct  
 while (Math.*abs*(A-B) >= epsilon){  
 A = B;  
 y1.add(i+1, *func*(x.get(i+1), A));  
 B = y.get(i-1)+h\*(y1.get(i+1)+4\*y1.get(i)+y1.get(i-1))/3;  
 }  
 y.add(i+1, B);  
 System.*out*.format(new StringBuilder().append("%.").append(after\_point).append("f \t %.").append(after\_point).append("f\t %.").append(after\_point).append("f\n").toString(), temp\_x, y.get(i), *func*(temp\_x, y.get(i)));  
 }  
// h /= 2;  
 System.*out*.println("--------------------------");  
  
 String x\_text\_1 = Arrays.*toString*(pointsMiln.get(2).toArray());  
 String y\_text\_1 = Arrays.*toString*(pointsMiln.get(1).toArray());  
  
 String text = "calculator.setExpression({ id: 'f', latex: 'cos(x)' });\n" +  
 "\tcalculator.setExpression({\n" +  
 " type: 'table',\n" +  
 " columns: [\n" +  
 " {\n" +  
 " latex: 'x',\n" +  
 " values: "+x\_text\_1+",\n" +  
 " },\n" +  
 " {\n" +  
 " latex: 'y',\n" +  
 " values: "+y\_text\_1+",\n" +  
 " color: Desmos.Colors.BLUE,\n" +  
 " columnMode: Desmos.ColumnModes.POINTS\n" +  
 " }\n" +  
 " ]\n" +  
 "});\n";  
 accessFile.seek(accessFile.length());  
 accessFile.write(text.getBytes());  
 }

**Output**

**Eiler method:**

h = 0.0012

0,001 >= 0,001

x y f(x,y)

1,000 -1,000 0,540

1,001 -0,999 0,539

1,002 -0,999 0,538

1,004 -0,998 0,537

1,005 -0,997 0,536

1,006 -0,997 0,535

1,007 -0,996 0,534

1,009 -0,995 0,533

1,010 -0,995 0,532

1,011 -0,994 0,531

…

49,990 -2,113 0,962

49,992 -2,112 0,963

49,993 -2,111 0,963

49,994 -2,110 0,963

49,995 -2,109 0,964

49,997 -2,107 0,964

49,998 -2,106 0,964

49,999 -2,105 0,965

**Miln Method**

h = 5.0

0,137 >= 0,001

x y f(x,y)

1,000 -1,000 0,540

6,000 1,702 0,960

11,000 6,502 0,004

16,000 6,524 -0,958

21,000 6,524 -0,548

26,000 -0,788 0,647

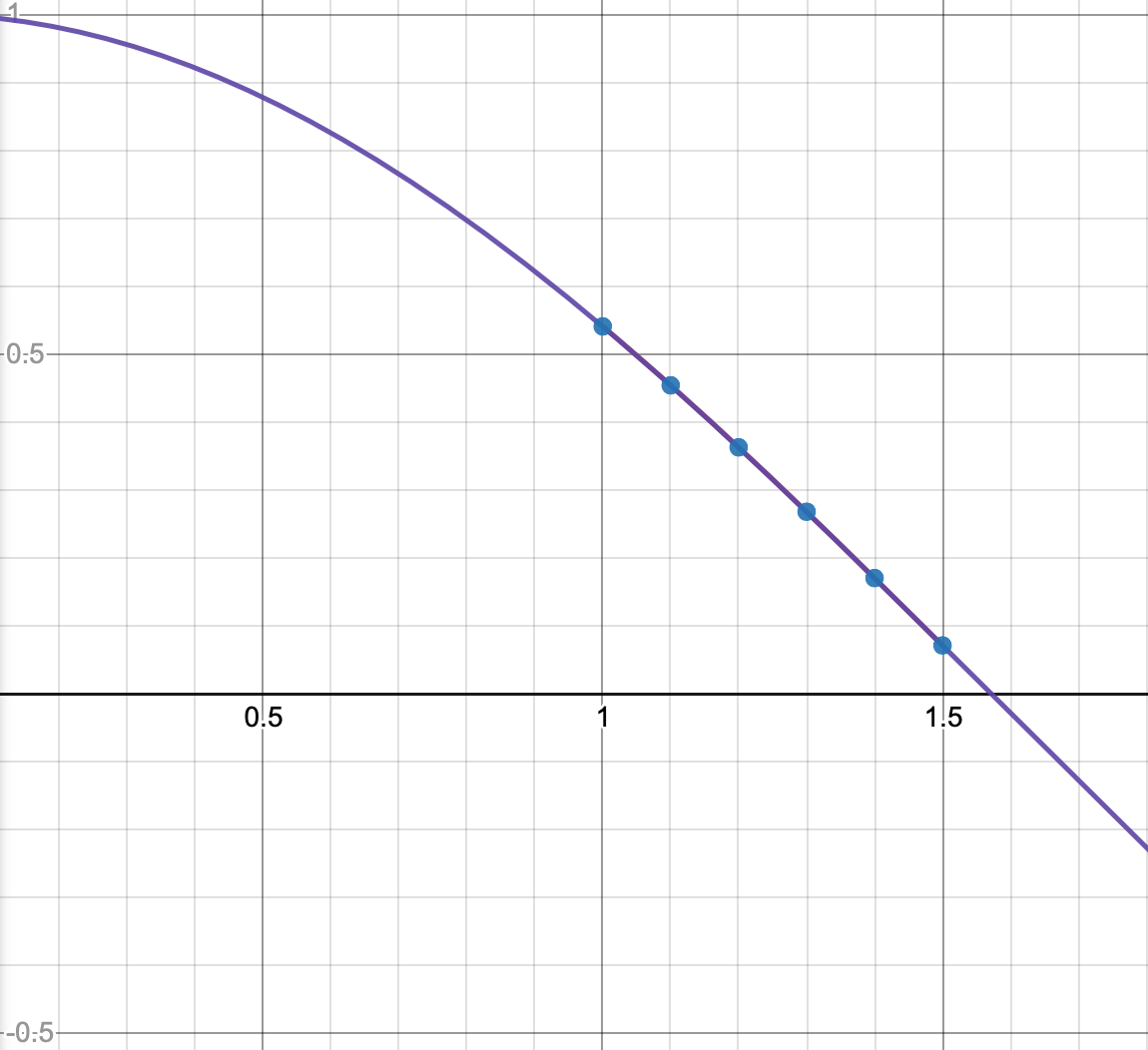
31,000 2,355 0,915

36,000 4,137 -0,128

41,000 9,318 -0,987

46,000 3,163 -0,432

Графики:



**Вывод:** реализована программа, решающая задачу Коши одношаговым модифицированным методом Эйлера (с делением шага и использованием правила Рунге), многошаговым методом Милна с поиском первых 3-х значений при помощи модифицированного метода Эйлера и правилом Рунге для коррекции решения. Также программа выводит график уравнения и полученного решения.