|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  Учреждение образования «Полоцкий государственный университет»  Факультет информационных технологий  Кафедра технологий программирования  ОТЧЕТ по  ЛАБОРАТОРНОЙ РАБОТЕ №2  по дисциплине  **«Методы численного анализа»**  На тему: «Методы Рунге-Кутта, методы Адамса для решения ОДУ»  ВЫПОЛНИЛ студент группы 15-КБ  Осипенко Вадим  ПРОВЕРИЛА ассистент кафедры ТП  Рудькова Т.С.  Полоцк, 2017 г. |

**Название**: «Методы Рунге-Кутта, методы Адамса для решения ОДУ»

**Цель работы**: Изучить наиболее распространенный одношаговый метод

Рунге-Кутта и многошаговый метод Адамса решения задачи Коши для ОДУ

первого порядка.

**Теоретическая часть**:

**Постановка задачи. Метод Рунге-Кутта**

Пусть имеем задачу Коши для ОДУ первого порядка

И будем решать её разностными методами. Среди других явных одношаговых методов

наибольшее распространение получил метод Рунге-Кутта.

Семейство явных одношаговых методов Рунге-Кутта для вычисления сеточного решения по уже известному значению этой функции в i-том узле, т.е. может быть выражено следующим образом.

Или в явном виде это семейство записать как:

где коэффициент

Коэффициенты

представляют собой константы, значение которых выбирается из соображений точности,

устойчивости и экономичности алгоритма.

Как правило, методы Рунге-Кутта со значениями k=5 не используются.

**Многошаговые методы Адамса.**

В методах Адамса аппроксимация u′(x) проводится только по двум точкам, то это означает, что коэффициенты левой части принимают значения 1; k.

Следовательно, методы Адамса можно записать следующим образом

Если = 0 получаем явные (экстраполяционные) методы Адамса, если ≠ 0, методы будут называться неявными (интерполяционными) методами Адамса. В зависимости от количества k предыдущих узлов, методы называются k –шаговыми.

**Вариант №9:**

Решить дифференциальное уравнение. Составить таблицу приближенных значений интеграла дифференциального уравнения y ′ = f (x, y) , удовлетворяющего начальным условиям на отрезке [a,b] с шагом h=0.1. Все вычисления вести с четырьмя десятичными знаками. Оценить погрешность вычислений. Для решения задачи Коши ОДУ применить:

1. Метод Рунге-Кутта: схема предиктор-корректор, метод 4 порядка, метод 3 порядка, схему третьего порядка уточнить по правилу Рунге. (на 7-8)

2. Продолжить таблицу на 3 шага методом Адамса (явным (8-9) и неявным (9-10)).

**Ход работы:**

Для начала найдем решение с помощью MathCad, которое будет выступать в качестве точного решения (Рисунок 1) а также построим график результатов(Рисунок 2).

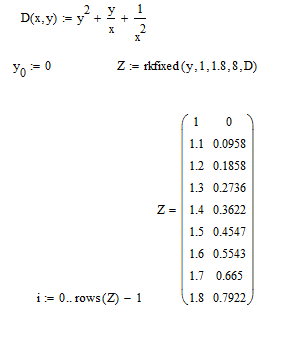


Рисунок 1 – Точное решение в MathCad.

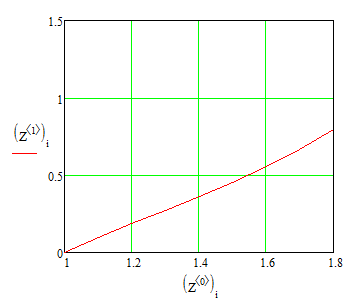


Рисунок 2 – График значений.

Напишем программы для первого задания реализующие метод Рунге-Кута.

Для метода – предиктор корректор.

Листинг 1 – реализация программы вычисляющей ОДУ методом Рунге-Кута 2-го порядка (Предиктор-корректор) на языке Java.

**public class** main3 {  
 **public static double** f(**double** x, **double** y){*//f(x,y)* **return** Math.*pow*(y,2) + (y/x) + (1/Math.*pow*(x,2));  
 }  
  
 **public static double** yh(**double** x, **double** y, **double** h){  
 **return** y + (h/2)\**f*(x,y);  
 }  
  
 **public static double** yn(**double** x, **double** y, **double** h){  
 **return** y + h\**f*(x+(h/2),*yh*(x,y,h));  
 }  
 **public static void** main(String[] args){*//предтктор-корректор* **double** xb = 1, xe = 1.5;  
 **double** y0 = 0;  
 **double** h = 0.1;  
 **double** y = y0;  
 System.***out***.println(**"Программа запущена."**);  
 System.***out***.println(**"h = "** + h + **"\nx(начальное) = "** + xb + **"\nx(конечное)"** + xe + **"\ny(начальное) = "** + y0);  
 **for**(**double** x = xb; x<=xe; x+=h){  
 y=*yn*(x,y,h);  
 System.***out***.println(**"y("**+(**float**)(x+h)+**") = "** + y);  
 }  
 }  
}

Результат работы программы:

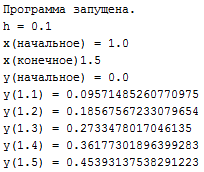


Рисунок 3 – результат работы программы.

Погрешность при округление решения до четырех чисел после запятой равна нулю.

Для метода – Рунге-Кута 3-го порядка.

Листинг 2 – реализация программы вычисляющей ОДУ методом Рунге-Кута 3-го порядка на языке Java.

**public class** main1 {  
 **public static double** f(**double** x, **double** y){*//f(x,y) или k1* **return** Math.*pow*(y,2) + (y/x) + (1/Math.*pow*(x,2));  
 }  
  
 **public static double** k1(**double** x, **double** y, **double** h){  
 **return** *f*(x,y);  
 }  
  
 **public static double** k2(**double** x, **double** y, **double** h){*//k2* **return** *f*(x+(h/3), y+((h/3)\**k1*(x,y,h)));  
 }  
  
 **public static double** k3(**double** x, **double** y, **double** h){*//k3* **return** *f*(x+(2\*h)/3, y + ((2\*h)/3)\**k2*(x,y,h));  
 }  
  
 **public static double** yn(**double** x, **double** y, **double** h){  
 **return** y + (h/4)\*(*k1*(x,y,h)+3\**k3*(x,y,h));  
 }  
  
 **public static void** main(String[] args){*//метод 3-его порядка (Q в маткаде)* **double** xb = 1, xe = 1.5;  
 **double** y0 = 0;  
 **double** h = 0.1;  
 **double** y = y0;  
 System.***out***.println(**"Программа запущена."**);  
 System.***out***.println(**"h = "** + h + **"\nx(начальное) = "** + xb + **"\nx(конечное)"** + xe + **"\ny(начальное) = "** + y0);  
 **for**(**double** x = xb; x<=xe; x+=h){  
 y = *yn*(x,y,h);  
 System.***out***.println(**"y("** + (**float**)(x+h) + **") = "** + y);  
 }  
 }  
}

Результат работы программы:

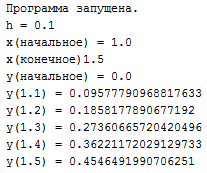


Рисунок 4 – результат работы программы.

Погрешность при округление решения до четырех чисел после запятой равна нулю.

Для метода – Рунге-Кута 4-го порядка.

Листинг 3 – реализация программы вычисляющей ОДУ методом Рунге-Кута 4-го порядка на языке Java.

**public class** main2 {  
 **public static double** f(**double** x, **double** y){*//f(x,y)* **return** Math.*pow*(y,2) + (y/x) + (1/Math.*pow*(x,2));  
 }  
 **public static double** k1(**double** x, **double** y, **double** h){*//k1* **return** *f*(x,y);  
 }  
 **public static double** k2(**double** x, **double** y, **double** h){*//k2* **return** *f*(x+(h/2), y+((h/2)\**k1*(x,y,h)));  
 }  
 **public static double** k3(**double** x, **double** y, **double** h){*//k3* **return** *f*(x+h/2, y + (h/2)\**k2*(x,y,h));  
 }  
 **public static double** k4(**double** x, **double** y, **double** h){  
 **return** *f*(x+h,y+h\**k3*(x,y,h));  
 }  
 **public static double** yn(**double** x, **double** y, **double** h){  
 **return** y + (h/6)\*(*k1*(x,y,h)+2\**k2*(x,y,h)+2\**k3*(x,y,h)+*k4*(x,y,h));  
 }  
 **public static void** main(String[] args){*//метод 4-его порядка (W в маткаде)* **double** xb = 1, xe = 1.5;  
 **double** y0 = 0;  
 **double** h = 0.1;  
 **double** y = y0;  
 System.***out***.println(**"Программа запущена."**);  
 System.***out***.println(**"h = "** + h + **"\nx(начальное) = "** + xb + **"\nx(конечное)"** + xe + **"\ny(начальное) = "** + y0);  
 **for**(**double** x = xb; x<=xe; x+=h){  
 y = *yn*(x,y,h);  
 System.***out***.println(**"y("** + (**float**)(x+h) + **") = "** + y);  
 }  
 }  
}

Результат работы программы:

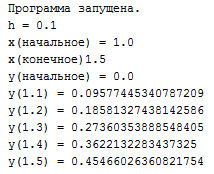


Рисунок 5 – результат работы программы.

Погрешность при округление решения до четырех чисел после запятой равна нулю.

Для метода – Рунге-Кута 3-го порядка по правилу уточнения.

Листинг 4 – реализация программы вычисляющей ОДУ методом Рунге-Кута 3-го порядка с уточнением на языке Java.

**public class** main4 {  
 **public static double** f(**double** x, **double** y){*//f(x,y) или k1* **return** Math.*pow*(y,2) + (y/x) + (1/Math.*pow*(x,2));  
 }  
  
 **public static double** k1(**double** x, **double** y, **double** h){  
 **return** *f*(x,y);  
 }  
  
 **public static double** k2(**double** x, **double** y, **double** h){*//k2* **return** *f*(x+(h/3), y+((h/3)\**k1*(x,y,h)));  
 }  
  
 **public static double** k3(**double** x, **double** y, **double** h){*//k3* **return** *f*(x+(2\*h)/3, y + ((2\*h)/3)\**k2*(x,y,h));  
 }  
  
 **public static double** yn(**double** x, **double** y, **double** h){  
 **return** y + (h/4)\*(*k1*(x,y,h)+3\**k3*(x,y,h));  
 }  
  
 **public static double** yh(**double** y1, **double** y2){  
 **return** ((Math.*pow*(2,3)\*y2-y1)/(Math.*pow*(2,3)-1));  
 }  
 **public static void** main(String[] args){*//метод 3-его порядка с уточнеием (R в маткаде)* **double** xb = 1, xe = 1.5;  
 **double** y0 = 0;  
 **double** h = 0.1;  
 **double** y = y0;  
 **double**[] y1 = **new double**[6];  
 **double**[] y2 = **new double**[11];  
 y1[0]=y0;  
 y2[0]=y0;  
 System.***out***.println(**"Программа запущена."**);  
 System.***out***.println(**"h = "** + h + **"\nx(начальное) = "** + xb + **"\nx(конечное)"** + xe + **"\ny(начальное) = "** + y0);  
 **int** i=1;  
 System.***out***.println(**"3-й порядок с шагом 0.1 :"**);  
 **for**(**double** x = xb; x<=xe; x+=h){  
 y = *yn*(x,y,h);  
 System.***out***.println(**"y("** + (**float**)(x+h) + **") = "** + y);  
 y1[i]=y;  
 i++;  
 }  
 i=1;  
 y = y0;  
 System.***out***.println(**"3-й порядок с шагом 0.05 :"**);  
 **for**(**double** x = xb; x<=xe; x+=(h/2)){  
 y = *yn*(x,y,h/2);  
 System.***out***.println(**"y("** + (**float**)(x+(h/2)) + **") = "** + y);  
 y2[i]=y;  
 i++;  
 }  
 i=0;  
 System.***out***.println(**"Окончательный результат"**);  
 **for** (i=0; i<6;i++){  
 System.***out***.println(**"y["** + i + **"] = "** + *yh*(y1[i],y2[((i\*2))]));  
 }  
 }  
}

Результат работы программы:

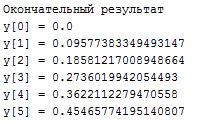


Рисунок 6 – результат работы программы.

Погрешность при округление решения до четырех чисел после запятой равна нулю.

Как мы видим методы Рунге-Кута являются достаточно точными.

Составим программу для второго задания – метода Адамса.

Листинг 4 – реализация программы вычисляющей ОДУ явным методом Адамса на языке Java.

**public class** adams1 {  
 **public static double** func(**double** x, **double** y){  
 **return** Math.*pow*(y,2) + (y/x) + (1/Math.*pow*(x,2));  
 }  
 **public static double** K1(**double** x, **double** y){  
 **double** K1 = *func*(x, y);  
 **return** K1;  
 }  
 **public static double** K2(**double** x, **double** y){  
 **double** K2 = *func*(x + 0.05, y + 0.05 \* *K1*(x,y));  
 **return** K2;  
 }  
 **public static double** K3(**double** x, **double** y){  
 **double** K3 = *func*(x + 0.05, y + 0.05 \* *K2*(x,y));  
 **return** K3;  
 }  
 **public static double** K4(**double** x, **double** y){  
 **double** K4 = *func*(x+0.1,y +0.1\**K3*(x,y) );  
 **return** K4;  
 }  
  
  
 **public static void** main (String[] args){*//Метод адамса явный* **double** xi[]={1,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8};  
 **double** yi[]=**new double**[10];  
 **double** f[]=**new double**[10];  
 **double** D1fi[]=**new double**[10];  
 **double** D2fi[]=**new double**[10];  
 **double** D3fi[]=**new double**[10];  
 **double** D4fi[]=**new double**[10];  
 **double** DYi[]=**new double**[10];  
 yi[0]=0;  
 **for** (**int** i=0; i<6; i++){  
 yi[i+1] = yi[i] + 0.1/6 \* (*K1*(xi[i],yi[i]) + 2\**K2*(xi[i],yi[i]) + 2\**K3*(xi[i],yi[i]) + *K4*(xi[i],yi[i]));  
 f[i]=*func*(xi[i],yi[i]);  
 }  
 **for** (**int** i=0;i<4; i++){  
 D1fi[i]=f[i+1] - f[i];  
 }  
 **for** (**int** i=0;i<3; i++){  
 D2fi[i]=D1fi[i+1] - D1fi[i];  
 }  
 **for** (**int** i=0;i<2; i++){  
 D3fi[i]=D2fi[i+1] - D2fi[i];  
 }  
 **for** (**int** i=0;i<1; i++){  
 D4fi[i]=D3fi[i+1] - D3fi[i];  
 }  
 **for** (**int** i=0;i<4; i++){  
 DYi[i]=yi[i+1] - yi[i];  
 }  
 **for**(**int** i=5; i<8; i++){  
 DYi[i]= 0.1 \* (*func*(xi[i],yi[i]) + (0.5\*D1fi[i-1])+(5/12\*D2fi[i-2])+ (3/18\*D3fi[i-3])+(251/720\*D4fi[i-4]));  
 yi[i+1]= yi[i] + DYi[i];  
 f[i+1] = *func*(xi[i+1],yi[i+1]);  
 D1fi[i]=f[i+1] - f[i];  
 D2fi[i-1]=D1fi[i] - D1fi[i-1];  
 D3fi[i-2]=D2fi[i-1] - D2fi[i-2];  
 D4fi[i-3]=D3fi[i-2] - D3fi[i-3];  
 **if**(i==7){  
 DYi[i]= 0.1 \* (*func*(xi[i],yi[i]) + (0.5\*D1fi[i-1])+(5/12\*D2fi[i-2])+ (3/18\*D3fi[i-3])+(251/720\*D4fi[i-4]));  
 yi[i+1]= yi[i] + DYi[i];  
 }  
 }  
  
 **for** (**int** i=0; i<9;i++){  
 System.***out***.print(String.*format*(**"x("**+ i +**")= "** + **"%.1f"**, xi[i]));  
 System.***out***.print(String.*format*(**". y("**+ i +**")="**+ **" %.4f"**,yi[i]));  
 System.***out***.print(String.*format*(**". f("**+ i +**")="**+ **" %.4f"**,f[i]));  
 System.***out***.print(String.*format*(**". Df("**+ i +**")="**+ **" %.4f"**,D1fi[i]));  
 System.***out***.print(String.*format*(**". D2f("**+ i +**")="**+ **" %.4f"**,D2fi[i]));  
 System.***out***.print(String.*format*(**". D3f("**+ i +**")="**+ **" %.4f"**,D3fi[i]));  
 System.***out***.print(String.*format*(**". D4f("**+ i +**")="**+ **" %.4f"**,D4fi[i]));  
 System.***out***.println(String.*format*(**". DY("**+ i +**")="**+ **" %.4f"**,DYi[i]));  
 }  
 }  
}

Результат работы программы.

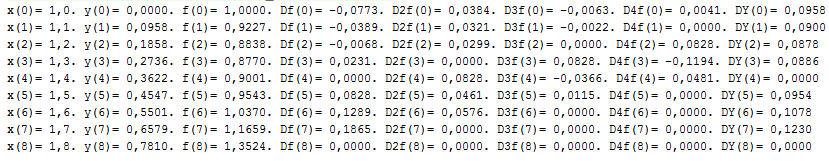


Рисунок 7 – Результат работы программы.

Максимальная погрешность метода равна 0.011%

Как мы видим самым точным являются методы Рунге-Кута.

**Выводы о проделанной работе.**

В ходе данной лабораторной работы мы решили задачу Коши с помощью 4 методов Рунге-Кута и явного метода Адамса. Решив нашу задачу мы определили что самым не точным методом является метод Адамса погрешность которого составила: 0.011%; погрешность остальных методов в нашем случае составила 0%.