|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  Учреждение образования «Полоцкий государственный университет»  Факультет информационных технологий  Кафедра технологий программирования  ОТЧЕТ по  ЛАБОРАТОРНОЙ РАБОТЕ №3  по дисциплине  **«Методы численного анализа»**  На тему: «Решение систем ОДУ»  ВЫПОЛНИЛ студент группы 15-КБ  Осипенко Вадим  ПРОВЕРИЛА ассистент кафедры ТП  Рудькова Т.С.  Полоцк, 2017 г. |

**Название**: «Решение систем ОДУ».

**Цель работы**: Изучить основные методы для нахождения решения системы обыкновенных дифференциальных уравнений.

**Теоретическая часть**:

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений.

Здесь U(x) – искомый вектор решения задачи

f(x,u(x)) - заданный вектор правой части.

U0 – вектор начальных условий.

Задача Коши для одного обыкновенного дифференциального уравнения является частным случаем задачи Коши для системы ОДУ. Рассмотрим случай системы, состоящей из двух дифференциальных уравнений.

Пусть вектор U(x) и система имеет вид:

; y(a)= z(a)=

Метод Эйлера для решения задачи (5)-(6) является конечно-разностным методом. Поэтому введем равномерную сетку с постоянным шагом h>0

Введем сеточные функции и и функции правой части.

Тогда приближенное решение системы в узлах будем вычислять последовательно по формулам:

Найти решение систем дифференциальных уравнений методом Эйлера и Рунге-Кутта на отрезке [0;1] при заданных начальных условиях с шагом h. Оценить погрешность полученного решения. В случае если решение имеет погрешность больше 5%-7% уменьшить шаг h.

**Ход работы:**

Для начала найдем точное решение в качестве которого будет выступать решение MathCad(Рисунок 1) и построим график.

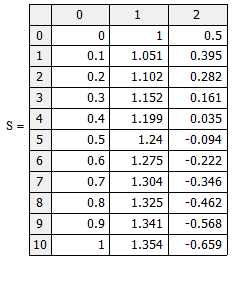


Рисунок 1 – Решение с помощью MathCad.

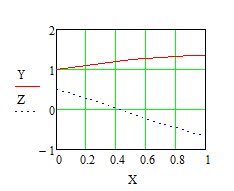


Рисунок 2 – График решений.

Напишем программу которая будет находить решения нашей функции с шагом h=0.1 методом Эйлера.

Листинг 1 – реализация программы вычисляющей системы ОДУ методом Эйлера на языке Java.

**public class** main1 {  
 **public static double** y(**double** x, **double** y, **double** z){  
 **return** y\*z+x;  
 }  
  
 **public static double** z(**double** x, **double** y, **double** z){  
 **return** Math.*pow*(x,2)-Math.*pow*(y,2);  
 }  
  
 **public static void** main(String[] args){  
 System.***out***.println(**"Программа запущена"**);  
 **double** xb = 0, xe = 1, h=0.1;  
 **double** y = 1;  
 **double** z = 0.5;  
 System.***out***.println(**""**);  
 **for** (**double** x = xb; x <= xe;){  
 **double** yt = y;  
 **double** zt = z;  
  
 *//System.out.println("y(" + (float)(x) + ") = " + (float)(y = (yt+h\*y(x,yt,zt))));  
 //System.out.println("z(" + (float)(x) + ") = " + (float)(z = (zt+h\*z(x,yt,zt))));* System.***out***.println(**"f("** + (**float**)(x+h) + **") = "** + (y += h \* *y*(x, yt, zt)));  
 System.***out***.println(**"f("** + (**float**)(x+h) + **") = "** + (z += h \* *z*(x, yt, zt)));  
 x+=h;  
 }  
 }  
}

Результатом будет:

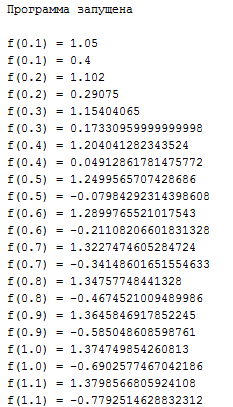


Рисунок 3 – Решение программы с шагом h=0.1

Как мы видим в данном случае погрешность слишком большая поэтому изменим шаг.

Результатом будет:

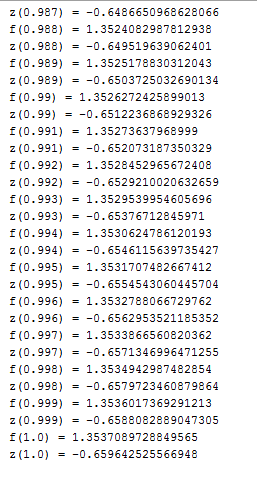


Рисунок 4 – Результат программы с шагом h=0.001

Как мы видим при уменьшении шага погрешность уменьшается когда шаг равен 0.001 погрешность приближается к нулю.

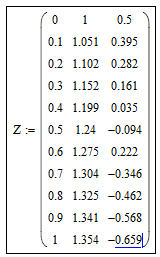


Рисунок 5 – Результат вычисления в узловых точках

Погрешность данного метода близка к нулю.

Напишем программу которая будет находить решения нашей функции с шагом h=0.1 методом Рунге-Кута.

Листинг 1 – реализация программы вычисляющей системы ОДУ методом Рунге-Кута на языке Java.

**public class** main2 {  
 **public static double** f(**double** x, **double** y, **double** z){*//f(x,y)* **return** y\*z+x;  
 }  
 **public static double** k1(**double** x, **double** y, **double** z, **double** h){*//k1* **return** *f*(x,y,z);  
 }  
 **public static double** k2(**double** x, **double** y, **double** z, **double** h){*//k2* **return** *f*(x+(h/2), y+((h/2)\**k1*(x,y,z,h)),z + h/2\**l1*(x,y,z,h));  
 }  
 **public static double** k3(**double** x, **double** y, **double** z, **double** h){*//k3* **return** *f*(x+h/2, y + (h/2)\**k2*(x,y,z,h),z+(h/2)\**l2*(x,y,z,h));  
 }  
 **public static double** k4(**double** x, **double** y, **double** z, **double** h){  
 **return** *f*(x+h,y+h\**k3*(x,y,z,h),z+h\**l3*(x,y,z,h));  
 }  
 **public static double** yn(**double** x, **double** y, **double** z, **double** h){  
 **return** y + (h/6)\*(*k1*(x,y,z,h)+2\**k2*(x,y,z,h)+2\**k3*(x,y,z,h)+*k4*(x,y,z,h));  
 }  
 **public static double** z(**double** x, **double** y, **double** z){*//f(x,y)* **return** Math.*pow*(x,2)-Math.*pow*(y,2);  
 }  
 **public static double** l1(**double** x, **double** y, **double** z, **double** h){*//k1* **return** *z*(x,y,z);  
 }  
 **public static double** l2(**double** x, **double** y, **double** z, **double** h){*//k2* **return** *z*(x+(h/2), y+((h/2)\**k1*(x,y,z,h)),z + h/2\**l1*(x,y,z,h));  
 }  
 **public static double** l3(**double** x, **double** y, **double** z, **double** h){*//k3* **return** *z*(x+h/2, y + (h/2)\**k2*(x,y,z,h),z+(h/2)\**l2*(x,y,z,h));  
 }  
 **public static double** l4(**double** x, **double** y, **double** z, **double** h){  
 **return** *z*(x+h,y+h\**k3*(x,y,z,h),z+h\**l3*(x,y,z,h));  
 }  
 **public static double** zn(**double** x, **double** y, **double** z, **double** h){  
 **return** z + (h/6)\*(*l1*(x,y,z,h)+2\**l2*(x,y,z,h)+2\**l3*(x,y,z,h)+*l4*(x,y,z,h));  
 }  
 **public static void** main(String[] args){*//метод 4-его порядка (W в маткаде)* **double** xb = 0, xe = 1;  
 **double** y0 = 1;  
 **double** z0 = 0.5;  
 **double** h = 0.01;  
 **double** y = y0;  
 **double** z = z0;  
 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){  
 **double** ty;  
 **double** tz;  
 ty = *yn*(x,y,z,h);  
 tz = *zn*(x,y,z,h);  
 y=ty;  
 z=tz;  
 System.***out***.println(**"y("** + (**float**)(x+h) + **") = "** + y);  
 System.***out***.println(**"z("** + (**float**)(x+h) + **") = "** + z);  
 }  
 }  
}

Результатом будет:

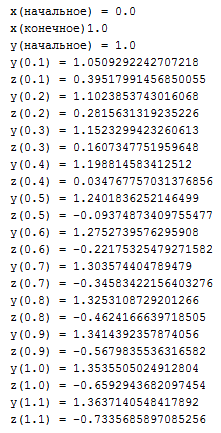


Рисунок 5 – Решение программы с шагом h=0.1

Как мы видим в данном случае погрешность слишком большая поэтому изменим шаг.

Результатом будет:

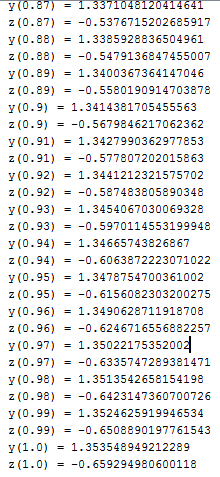


Рисунок 6 – Результат программы с шагом h=0.01

Как мы видим при уменьшении шага погрешность уменьшается когда шаг равен 0.01 погрешность приближается к нулю.

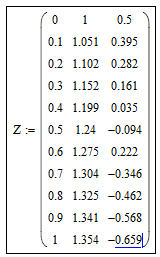


Рисунок 7 – Результат вычисления в узловых точках

График будет иметь вид:

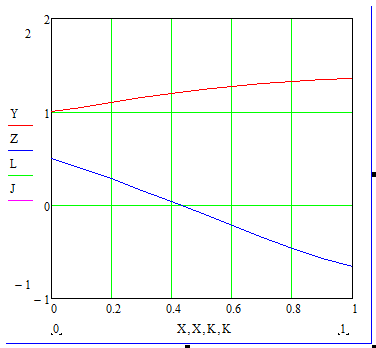


Рисунок 8 – Конечный график

**Выводы о проделанной работе.**

В данной лабораторной работе мы решили систему ОДУ и определили что численный метод решения может обладать высокой сходимость однако его точность будет зависеть от вида функции и шага решения. В нашем случае метод Эйлера оказался менее точным чем метод Рунге-Кута 4-го порядка т.к. для достаточной точности методу Эйлера понадобилось использовать шаг 0.001, а методу Рунге-Кута 0.01.