|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  Учреждение образования «Полоцкий государственный университет»  Факультет информационных технологий  Кафедра технологий программирования  ОТЧЕТ по  ЛАБОРАТОРНОЙ РАБОТЕ №1  по дисциплине  **«Методы численного анализа»**  На тему: «Метод Эйлера, усовершенствованный метод Эйлера, метод Эйлера-Коши для решения ОДУ»  ВЫПОЛНИЛ студент группы 15-КБ  Осипенко Вадим  ПРОВЕРИЛА ассистент кафедры ТП  Рудькова Т.С.  Полоцк, 2017 г. |

**Название**: «Метод Эйлера, усовершенствованный метод Эйлера, метод Эйлера-Коши для решения ОДУ»

**Цель работы**: Изучить основные методы решения задачи Коши для ОДУ первого порядка: Метод Эйлера, усовершенствованный метод Эйлера, метод Эйлера-Коши.

**Теоретическая часть**:

**Постановка задачи**: Пусть задано дифференциальное уравнение где f(x,u(x)) – заданная непрерывная функция двух аргументов. Требуется найти функцию u=u(x), непрерывную при 0≤x<X, удовлетворяющую этому уравнению и некоторому начальному условию u(0)=u0. Решение задачи Коши любым численным методом может быть найдено только в виде таблицы. Чаще всего используются разностные методы. Поэтому построим сетку (дискретное множество точек) ω={0=<<…, - шаг сетки. В каждом узле сетки i = 0,1,2,..,n будем рассматривать сеточную функцию как приближенную (аппроксимирующую) к точному решение на данном множестве точек. Далее заменим значения производной функции u(x) в первом уравнение отношением конечных разностей. Таким образом, переходим от непрерывного дифференциального уравнения относительно функции u(x) к разностной задаче относительно сеточной функции y.

y()=

**Разностная схема Эйлера (метод ломаных).**

Простейшим численным методом решения задачи Коши для ОДУ первого порядка, является метод Эйлера. Метод Эйлера может быть построен, исходя из понятий теории разностных схем. Введем на отрезке [a,b] равномерную сетку и соответствующую ей сеточную функцию для аппроксимации искомого решения Для аппроксимации производной используем правую разностную схему, т.е. . И получим, h = (). Метод Эйлера обладает малой точностью, т.е имеет первый порядок точности. Погрешность каждого нового шага вообще говоря систематически возрастает. Теоретически для оценки погрешности метода Эйлера имеет место неравенство: Здесь М1, М2, М3 - верхние границы для функции f (x, u) и ее частных производных. Однако для практики наиболее приемлемым способом оценки погрешности решения является метод двойного пересчета с шагами h и h/2. Совпадающие десятичные знаки решения в соответствующих узлах при различных пересчетах считаются верными.

**Усовершенствованный метод Эйлера.** Усовершенствованным методом Эйлера называется метод, который использует формулу следующего вида Для использования этого метода нам необходима формула

**Метод Эйлера-Коши.**

Рассмотрим еще одну модификацию метода Эйлера, если аппроксимацию функции f приравнять среднему арифметическому значений этой функции на концах элементарного отрезка: Данный алгоритм можно записать в виде одного соотношения вида: . Формула является модификацией метода Эйлера и ее называют методом Эйлера-Коши. Этот метод может быть получен путем разложения функции u(x) в ряд Тейлора. На практике же оценку погрешности полученного решения принято получать с помощью двойного пересчета с шагами h и h/2. С помощью метода Эйлера-Коши можно проводить контроль точности решения путем сравнения промежуточного значения функции в i+1 узле с ее окончательным значением в этом узле , т.е. значений и На основании этого сравнения выбирается величина шага h в каждом узле. Если модуль разности этих значений сравним с погрешностью вычислений, т.е. выполняется неравенство , то шаг можно увеличить.

**Вариант №9:**

**x**∈[1.2;2.2]

Решить данное дифференциальное уравнение. Составить таблицу приближенных значений интеграла дифференциального уравнения y'=f(x,y), удовлетворяющего начальным условиям y(x0)=y0 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 Эйлера;

2. Усовершенствованный метод Эйлера;

3. Метод Эйлера-Коши.

Перед написанием и тестированием программы решим данную задачу с помощью MathCad

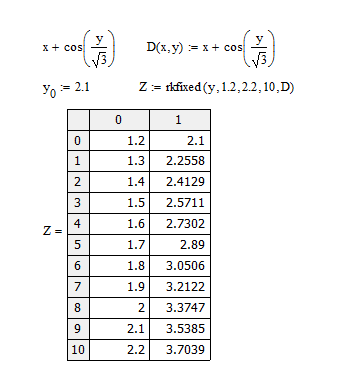


Рисунок 1 – решение задачи с помощью MathCad



Рисунок 2 – графическое представление результатов

Теперь зная решение нашего ОДУ напишем 3 программы решающие уравнение методом: Эйлера, Усовершенствованным методом Эйлера и Методом Эйлера-Коши.

1. Метод Эйлера.

Метод Эйлера самый простой в реализации, однако он является самым неточным. Напишем программу для данного метода.

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

**public class** main1 {  
 **public static double** f(**double** x, **double** y){  
 **return** x+Math.*cos*(y/Math.*sqrt*(3));  
 }*//функция y штрих* **public static double** fn(**double** x, **double** y, **double** h){  
 **return** y+h\*(*f*(x,y));  
 }*//функция вычисляющая следующий y* **public static void** main(String[] args){  
 **double** xb=1.2;*//начальное значение x* **double** xe=2.2;*//конечное значение x* **double** y=2.1;*//значение y нулевое* **double** h=0.1;*//шаг* **for**(**double** i=1.2;i<=2.2;i+=h){  
 //System.***out***.println(**"x = "** + i + **" y = "** + y);  
 System.***out***.println(**"y("** + (i+0.1) + **") = "** + *fn*(i,y,h));  
 y=*fn*(i,y,h);  
 //System.***out***.println();  
 }  
  
 }  
}

В данной программе цикл находит последующие решения задачи Коши для y с помощь данных формул.

Результатом работы программы является:

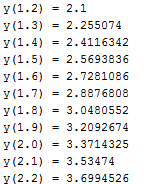


Рисунок 3 – результат работы первой программы

Составим таблицу демонстрирующую изменения

Таблица 1 – результат работы первой программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I | x(i) | y(i) | Точное решение | Погрешность  % |
| 0 | 1.2 | 2.1 | 2.1 | 0 |
| 1 | 1.3 | 2.255 | 2.2558 | 3.548\*10^-4 |
| 2 | 1.4 | 2.4116 | 2.4129 | 5.391\*10^-4 |
| 3 | 1.5 | 2.5693 | 2.5711 | 7.006\*10^-4 |
| 4 | 1.6 | 2.7281 | 2.7302 | 7.698\*10^-4 |
| 5 | 1.7 | 2.8877 | 2.89 | 7.965\*10^-4 |
| 6 | 1.8 | 3.048 | 3.0506 | 8.53\*10^-4 |
| 7 | 1.9 | 3.2092 | 3.2122 | 9.348\*10^-4 |
| 8 | 2.0 | 3.3714 | 3.3747 | 9.788\*10^-4 |
| 9 | 2.1 | 3.5347 | 3.5385 | 1.075\*10^-3 |
| 10 | 2.2 | 3.6994 | 3.7039 | 1.216\*10^-3 |

Погрешность метода составила: 7.149\*10^-4 %

2. Усовершенствованный метод Эйлера.

Усовершенствованный метод Эйлера является более точным чем обычный метод Эйлера. Он не сильно отличается в реализации от обычного метода и основан на усредненном значение y и x.

Листинг 2 – реализация программы вычисляющей ОДУ усовершенствованным методом Эйлера на языке Java

**public class** main2 {  
 **public static double** f(**double** x, **double** y){  
 **return** x+Math.*cos*(y/Math.*sqrt*(3));  
 }  
 **public static double** fn(**double** x, **double** y, **double** h){  
 **return** y+h\*(*f*(x+h/2,*f12*(x,y,h)));  
 }  
 **public static double** f12(**double** x, **double** y, **double** h){  
 **return** y+(h/2)\**f*(x,y);  
 }  
 **public static void** main(String[] args){  
 **double** xb=1.2;  
 **double** xe=2.2;  
 **double** y=2.1;  
 **double** h=0.1;  
 System.***out***.println(**"y("** + xb + **") = "** + y);  
 *//System.out.println();* **for**(**double** i=1.2;i<=2.2;i+=h){  
 *//System.out.println("x = " + i + " y = " + y);* System.***out***.println(**"y("** + (**float**)(i+h) + **") = "** + (**float**)*fn*(i,y,h));  
 y=*fn*(i,y,h);  
 *//System.out.println();* }  
  
 }  
}

В данной программе цикл находит последующие решения задачи Коши для y с помощью изменённых формул.

Результатом работы программы является:

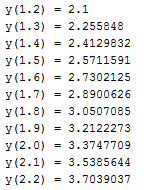


Рисунок 4 – результат работы второй программы

Составим таблицу демонстрирующую изменения

Таблица 2 – результат работы второй программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I | x(i) | y(i) | Точный результат | Погрешность  % |
| 0 | 1.2 | 2.1 | 2.1 | 0 |
| 1 | 1.3 | 2.2558 | 2.2558 | 0 |
| 2 | 1.4 | 2.413 | 2.4129 | 4.144\*10^-5 |
| 3 | 1.5 | 2.5711 | 2.5711 | 0 |
| 4 | 1.6 | 2.7302 | 2.7302 | 0 |
| 5 | 1.7 | 2.89 | 2.89 | 0 |
| 6 | 1.8 | 3.0507 | 3.0506 | 3.278\*10^-5 |
| 7 | 1.9 | 3.2122 | 3.2122 | 0 |
| 8 | 2.0 | 3.3748 | 3.3747 | 2.963\*10^-5 |
| 9 | 2.1 | 3.5385 | 3.5385 | 0 |
| 10 | 2.2 | 3.7039 | 3.7039 | 0 |

В данном случае погрешность составила: 9.441\*10^-6

3. Метод Эйлера-Коши.

Данный метод является более точным в сравнение с обычным методом и основывается на разложение в ряд Коши.

Листинг 2 – реализация программы вычисляющей ОДУ метод Эйлера-Коши на языке Java

**public class** main3 {  
 **public static double** f(**double** x, **double** y){*//* **return** x+Math.*cos*(y/Math.*sqrt*(3));  
 }  
 **public static double** y(**double** y, **double** dy){*//* **return** y+dy;  
 }  
 **public static double** yv(**double** x, **double** y, **double** h){*//* **return** y+h\**f*(x,y);  
 }  
 **public static double** dy(**double** f, **double** fv, **double** h){  
 **return** (h/2)\*(f+fv);  
 }  
 **public static void** main(String[] args){  
 **double** xb=1.2;  
 **double** xe=2.2;  
 **double** y=2.1;  
 **double** h=0.1;  
 *//System.out.println("x = " + xb + " y(" + xb + ") = " + y);* **for**(**double** i=1.2;i<=2.2;i+=h){  
 **double** f=*f*(i,y);  
 **double** yv=*yv*(i,y,h);  
 **double** fv=*f*(i+h,yv);  
 **double** dy=*dy*(f, fv, h);  
 *//System.out.println("x = " + h + " y = " + y);* y+=dy;  
 System.***out***.println( **"y("** + (**float**)(i+h) + **") = "** + y);  
 }  
 }  
}

Как мы видим последующие значения находятся с помощью суммы предыдущего и найденного с помощью формул значения

Результатом работы программы является:

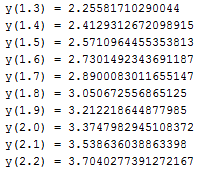


Рисунок 3 – результат работы программы

Составим таблицу демонстрирующую изменения

Таблица 3 – результат работы первой программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I | x(i) | y(i) | Точный результат | Погрешность  % |
| 0 | 1.2 | 2.1 | 2.1 | 0 |
| 1 | 1.3 | 2.2558 | 2.2558 | 0 |
| 2 | 1.4 | 2.4129 | 2.4129 | 0 |
| 3 | 1.5 | 2.5711 | 2.5711 | 0 |
| 4 | 1.6 | 2.7301 | 2.7302 | 3.663\*10^-5 |
| 5 | 1.7 | 2.89 | 2.89 | 0 |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 1.8 | 3.0507 | 3.0506 | 3.278\*10^-5 |
| 7 | 1.9 | 3.2122 | 3.2122 | 0 |
| 8 | 2.0 | 3.3748 | 3.3747 | 2.963\*10^-5 |
| 9 | 2.1 | 3.5387 | 3.5385 | 5.652\*10^-5 |
| 10 | 2.2 | 3.704 | 3.7039 | 2.7\*10^-5 |

Погрешность в данном случае составила: 1.66\*10^-5

Составим график на котором будут изображены значения всех методов:

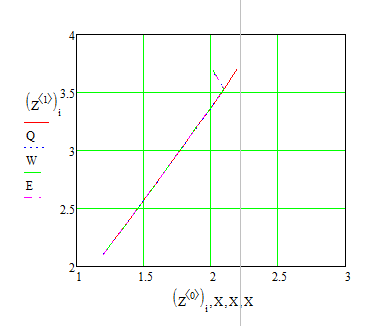


Рисунок 4 – график значений x и y

На данном графике мы видим четыре линии, которые расположены близко к друг другу здесь Q - Метод Эйлера; W - Усовершенствованный метод Эйлера; E - Метод Эйлера-Коши; а Z – решение MathCad. Поскольку все линии расположены достаточно близко оценить погрешность графически нет возможности и мы должны сравнить их другим методом а именно сравнив погрешность в каждой точке.

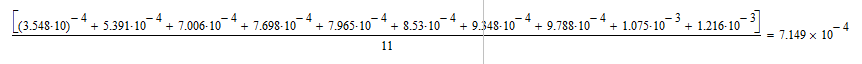


Рисунок 5 – подсчет погрешности первого метода



Рисунок 6 – подсчет погрешности второго метода



Рисунок 7 – подсчет погрешности третьего метода

**Выводы о проделанной работе.**

В ходе данной лабораторной работы мы решили задачу Коши с помощью трех методов: Метод Эйлера, усовершенствованный метод Эйлера и метод Эйлера-Коши. Решив нашу задачу мы определили что самым не точным методом является первый погрешность которого составила: 7.149\*10^-4; после которого идет третий с погрешностью: 1.66\*10^-5; самым точным оказался второй метод - усовершенствованный метод Эйлера его погрешность: 9.441\*10^-6.