МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт "Радиоэлектроники интеллектуальных технических систем"

Кафедра "Информатика и управление в технических системах"

### **ОТЧЁТ**

О выполнении практической работы №2

«Решение задачи Коши»

Выполнил: **Рябый А.И.**

группы **УТС-21-1-о**

Направление / специальность 27.03.04

«Управление в технических системах»

Проверил: ст. преп. Липко И.Ю.

Севастополь, 2022

**Цель работы**

Изучение способов численного решения систем обыкновенных дифференциальных уравнений посредством реализации алгоритмов интегрирования и сравнения их качества с точным решением.

**Ход работы**

Целью лабораторной работы является решение задачи Коши.

Постановка задачи Коши:

*–* дифференциальное уравнение

– начальные условия

Обыкновенное дифференциальное уравнение в нормальной форме (форме Коши) имеет вид:

Численное решение ОДУ представляет собой:

При выполнении лабораторной работы были программно реализованы численные методы решения дифференциальных уравнений, формулы которых представлены ниже:

1. Метод Эйлера
2. Метод трапеций
3. Метод Рунге – Кутта , где

Начальные условия y0 иy'0, отрезок интегрирования [t, **tf**] и уравнения для решения по вариантам представлены в таблицах ниже. Для того, чтобы уравнение стало возможным решить с помощью численных методов, их необходимо привести к форме Коши.

Значение решения дифференциального уравнения в заданный момент времени. (начальное условие). Точка, через которую проходит решение дифференциального уравнения.

Уравнение №1 вариант 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Уравнение** | **t0** | **tf** | **y0** | **y'0** | **Точное решение** |
|  | 0 | 10 | 1.8 | -0.5 |  |

1)

2)

Уравнение №2 вариант 16

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Уравнение** | **t0** | **tf** | **y0** | **y'0** | **Точное решение** |
|  | 1 | 30 | 5 | -1 | 5-ln(t) |

1)

2)

Реализуем запись численный метод трапеции:

𝑥∗(𝑡𝑘+1) = 𝑥(𝑡𝑘) + ℎ𝑓(𝑥(𝑡𝑘+1), 𝑡𝑘+1)

Численный метод Рунге-Кутта 4 порядка точности:

𝑥(𝑡𝑘+1) = 𝑥(𝑡𝑘) + ℎ (𝑘1 + 2𝑘2 + 2𝑘3 + 𝑘4) , где

𝑘1 = 𝑓(𝑥(𝑡𝑘), 𝑡𝑘)

𝑘2 = 𝑓 (𝑥(𝑡𝑘) + , 𝑡𝑘 + )

𝑘3 = 𝑓 (𝑥(𝑡𝑘) + , 𝑡𝑘 + ))

𝑘4 = 𝑓(𝑥(𝑡𝑘) + 𝑘3, 𝑡𝑘 + ℎ)

Векторы 𝑘𝑖, 𝑖 = 1,2,3,4 – суть значения вектор-функции правых частей 𝑓(𝑥, t), вычисленные при разных значениях аргументов. Зависимость каждого вектора от предыдущего обуславливает необходимость последовательного вычисления в порядке нумерации.

**Исследование графиков:**

Согласно графикам, изображенным на рисунке 1, можно сказать, что максимально близким к точному решению, в данной ситуации, является решение методом трапеции.

Рис.1 – Сравнение численных методов для уравнения №1

Согласно графикам, изображенным на рисунке 2, можно сказать, что максимально близким к точному решению, в данной ситуации, является решение методом Рунге-Кутта.

Рис.2 – Сравнение численных методов для уравнения №2

Согласно графику, изображенному на рисунке 3, можно сказать, что при решении уравнения методом трапеции ошибка меньше чем при решении методом Рунге-Кутта.

Рис.3 – Оценка ошибок для уравнения № 1

Проанализировав графики, изображенные на рисунке 4, можно сказать, что при решении уравнения методом Рунге-Кутта ошибка меньше чем при решении методом трапеции.

Рис.4 – Оценка ошибок для уравнения № 2

Проанализировав рисунок 5, можно сказать, что при использовании метода трапеции для уменьшения ошибки необходимо брать шаг 0,001.

Для метода Рунге-Кутта нужно брать шаг 1. Чем больше шаг, тем меньше погрешность вычислений.

Погрешность при каждом шаге:

Рис.5 – Оценка зависимости ошибки метода от шага для уравнения №1

Метод Рунге-Кутта оказался более точным. Для метода Рунге-Кутта нужно брать шаг 0,01. Чем меньше шаг, тем более точное вычисление.

Проанализировав рисунок 6, можно сказать, что при использовании метода трапеции для уменьшения ошибки необходимо брать шаг 0,1.

Рис.6 – Оценка зависимости ошибки метода от шага для уравнения №2

**Заключение**

В ходе лабораторной работы были изучены способы численного решения систем обыкновенных дифференциальных уравнений посредством реализации алгоритмов интегрирования и сравнения их качества с точным решением. Каждый метод подходит для вычисления различных уравнения по-разному. Для решения функций мы находили уравнение Коши вручную. После мы программировали метод Трапеций и метод Рунге-Кутта. Далее мы исследовали эти методы, сравнивали их между собой и с точным решением, полученным на калькуляторе.

На основе анализа построенных графиков, сделан вывод о зависимости точности решения численными методами от шага интегрирования и выявлен самый точный метод.

Для первого уравнения метод трапеции оказался более точным чем метод Рунге-Кутта. Ошибка была минимальна при шаге 0,001, но при значениях шага выше 0,001 ошибка возрастала. Однако, для второго уравнения метод Рунге-Кутта более точен. При шаге меньше 0,1 достигается хорошая точность, но при шаге от 0,1 и до 1 ошибка возрастала, как для метода трапеции, так и для метода Рунге-Кутта.

Метод Рунге – Кутта оказался менее точным, хотя согласно теории, должен был обладать наибольшей точностью. Возможно, на результат повлияла погрешность вычислений.

**Приложение:**

#include <iostream>

#include <fstream>

#include <math.h>

using namespace std;

/\*Для варианта 6

//отрезок интегрирования// отрезок изменения независимой переменной

float t = 0.0;

float tf = 10.0;

float h = 0.001;

float x0[2] = { 1.8, -0.5 };

float xEul[2];

float solution(float t) {

return exp(t)+exp(-t)-0.5\*sin(t)-0.2\*cos(2\*t);

}

void func(float t, const float\* x, float\* f) {

f[0] = x[1];

f[1] =(sin(t)+cos(2\*t)+x[0]) ;

}

\*/

/\*Для варианта 16

float t = 1.0;

float tf = 30.0;

float h = 0.001;

float x0[2] = { 5, -1 };

float xEul[2];

float solution(float t) {

return 5-log(t);

}

void func(float t, const float\* x, float\* f) {

f[0] = x[1];

f[1] =((-x[0]\*t)/(t\*t)) ;

}

\*/

float t = 1.0;

float tf = 30.0;

float h = 0.001;

float x0[2] = { 5, -1 };

float xEul[2];

float **solution**(float t) {

return 5-log(t);

}

void **func**(float t, const float\* x, float\* f) {

f[0] = x[1];

f[1] =((-x[0]\*t)/(t\*t)) ;

}

void **prymoug**(float\* x, float t, float h) {

float f[2];

func(t, x, *f*);

x[0] = x[0] + h \* f[0];

x[1] = x[1] + h \* f[1];

}

void **trapez**(float\* x, float t, float h) {

float f[2];

func(t, x, *f*);

float fh[2];

float xz[1];

xz[0]=x[0] + h\*f[0];

//xz[1]= x[1] + h\*f[1];

func(t+h,xz,*fh*);

x[0] = x[0] + h \*0.5\*(f[0]+fh[0]);

x[1] = x[1] + h \*0.5\*(f[1]+fh[1]);

}

void **rungeCutt**(float \*x,float t, float h){

float f[2];

func (t,x,*f*);

float k1[2],k2[2],k3[2],k4[2],sum[2],xv[2];

k1[0] = f[0];

k1[1] = f[1];

xv[0] = x[0] + k1[0] / 2.0;

xv[1] = x[1] + k1[1] / 2.0;

func(t+h/2.0,xv,*f*);

k2[0] = f[0];

k2[1] = f[1];

xv[0] = x[0] +k2[0] /2.0;

xv[1] = x[1] +k2[1] /2.0;

func(t+h/2.0,xv,*f*);

k3[0] = f[0];

k3[1] = f[1];

xv[0] = x[0] +k3[0] /2.0;

xv[1] = x[1] +k3[1] /2.0;

func(t+h,xv,*f*);

k4[0] = f[0];

k4[1] = f[1];

sum[0] = k1[0] +2 \*k2[0] + 2\*k3[0] +k4[0];

sum[1] = k1[1] +2 \*k2[1] + 2\*k3[1] +k4[1];

x[0]= x[0] +1.0 /6.0 \* h \* sum[0];

x[1]= x[1] +1.0 /6.0 \* h \* sum[1];

}

float **err**(float a, float b) {

return (a - b) \* (a - b);

}

int **main**()

{

ofstream f;

f.open("data.csv");

xEul[0] = x0[0];

xEul[1] = x0[1];

while (t < tf) {

trapez(*xEul*, t, h);

f << t << "," << solution(t)

<< "," << xEul[0] <<","<< xEul[1]

<< "," << err(solution(t), xEul[0])

<< endl;

t += h;

}

f.close();

}