Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

09.03.01 Информатика и вычислительная техника

**Курсовая работа**

**по дисциплине «Вычислительная математика»**

**Вариант №9**

по направлению 09.03.01 «Информатика и вычислительная техника»,

направленность (профиль) – «Программное обеспечение средств вычислительной техники и автоматизированных систем», квалификация – бакалавр,

программа академического бакалавриата,

форма обучения – очная, год начала подготовки (по учебному плану) – 2016

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИП-611  «26» мая 2018 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Кияница А.П./ |
| Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_» |  |  |
| Проверил:  доцент Кафедры ПМиК  «30» мая 2018 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Рубан А.А./ |

Новосибирск, 2018

Содержание

[Введение 3](#_Toc515437661)

[Постановка задачи 3](#_Toc515437662)

[Основные идеи и характеристики применяемых методов 4](#_Toc515437663)

[*1. Метод Рунге-Кутта 4го порядка.* 4](#_Toc515437664)

[2. *Метод стрельб* 4](#_Toc515437665)

[3. *Оценка погрешности решения ДУ и СДУ методом двойного пересчета. Коррекция решения*. 4](#_Toc515437666)

[4. *Метод простого деления (МПД).* 5](#_Toc515437667)

[Описание программы 7](#_Toc515437668)

[Результаты 8](#_Toc515437669)

[Листинг программы 9](#_Toc515437670)

# Введение

**Краевые задачи для дифференциальных уравнений.**

Для ДУ высших порядков часто бывает необходимо решить так называемую краевую задачу, т.е. начальные условия, которые заданы в разных точках.

Рассмотрим простейшую краевую задачу для ДУ 2го порядка:

 (1)

А мы умеем решать:

 (2),

В (2) нам известно , поэтому для решения задачи (1) мы будем подбирать  в (2), с тем, чтобы у(b) = у1.

# Постановка задачи

1. Решить краевую задачу методом Рунге-Кутта IV

y’’=y’+ex

y(0)=1;

y(1)=e;

# Основные идеи и характеристики применяемых методов

## *1. Метод Рунге-Кутта 4го порядка.*

Наиболее применяемым методом решения ДУ и СДУ является метод Рунге-Кутта 4го порядка.

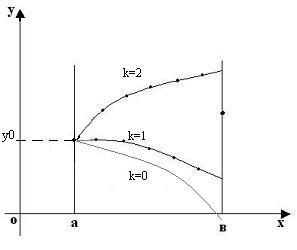
Формулы метода Рунге-Кутта 4го порядка:

 (6.7)



в векторной форме данной формулы, величины y, f, k заменяют на Y, F, K.

## 2. *Метод стрельб*

После пристрелки и определения интервала [a,b], где идёт смена знака, запускаем МПД или МХ. На практике это выглядит так, как будто мы решаем уравнение , где возвращает решение задачи Коши (6.11) в точке b при заданном k. 

## 3. *Оценка погрешности решения ДУ и СДУ методом двойного пересчета. Коррекция решения*.

Используя такую же идею, как и в численном интегрировании, находим решение ДУ на [a,b] дважды с шагом h и с шагом h/2. Получим следующую картину:



Сравниваем попарно, если расхождение между  для метода 2го порядка,  для метода 4го порядка, то в качестве точного решения берём . Если же точность не достигнута, то шаг h уменьшаем вдвое и т.д., пока она не будет достигнута.

Метод двойного пересчёта при решении ДУ и СДУ практически единственный имеет возможность для оценки погрешностей, так как иные формулы очень сложны и требуют оценок различных производных.

Как и при ЧИ, при решении ДУ и СДУ после 2го пересчёта в качестве точного решения выгодно брать не , а .

- для второго порядка

Метод двойного пересчёта применим не только лишь при ЧИ, при решении ДУ и СДУ, но и при решении других численных методов.

## 4. *Метод простого деления (МПД).*

1. Находим интервал a, b на котором функция меняет свой знак:

f(a)\*f(b)<0 (имеет хотя бы один корень)

2. Делим интервал пополам точкой С:



3. Из 2-х полученных интервалов([a,c] и [c,b]) выбираем тот, на котором происходит смена знака:

f(a)\*f(с)<0 - [a,c]

f(с)\*f(b)<0 - [c,b]

4. Повторить пункт 2, если не достигли наперед заданной точности |b-a|>, иначе, если , то идем на пункт 5.

5. В качестве точного решения берём  (середина последнего интервала). От этой точки х расстояние до любой другой точки отрезка не превосходит .

***Замечание:***

В предложенном выше методе мы контролируем точность по х (). Иногда, вместо этого требуется достигнуть заданной точности по y, т.е. , но обычно, под точным понимается точное по x.

# Описание программы

**1.**Функция *double\* F(double x,double\* Y)* – вычисляем производную вектора.

**2.**Функция *double\* summ(double\* Y,double\* k,double h)* – вычисляет значение Yi.

**3.**Функция *double\* Yi(double\* Y,double x,double h)* –

возвращает значение первой производной, полученное при решении ДУ методом Рунге-Кутта с одним шагом.

**4.**Функция *bool comparison(double\*\* y, double\*\* yy,int N,double e)* – осуществляет сравнение двух массивов значений производных для двойного пересчета.

**5.**Функция *double comp(double X)* – вычисляет значения, для построения кубического сплайна.

6. Функция double\* Yi(double\* Y1,double x,double h, int b,double e)-вычисляет значение производной на интервале с двойным пересчетом

7. Функция double searchCor(double\* Y,double x,double h, int b,double e)-реализует метод пристрелки

# Результаты

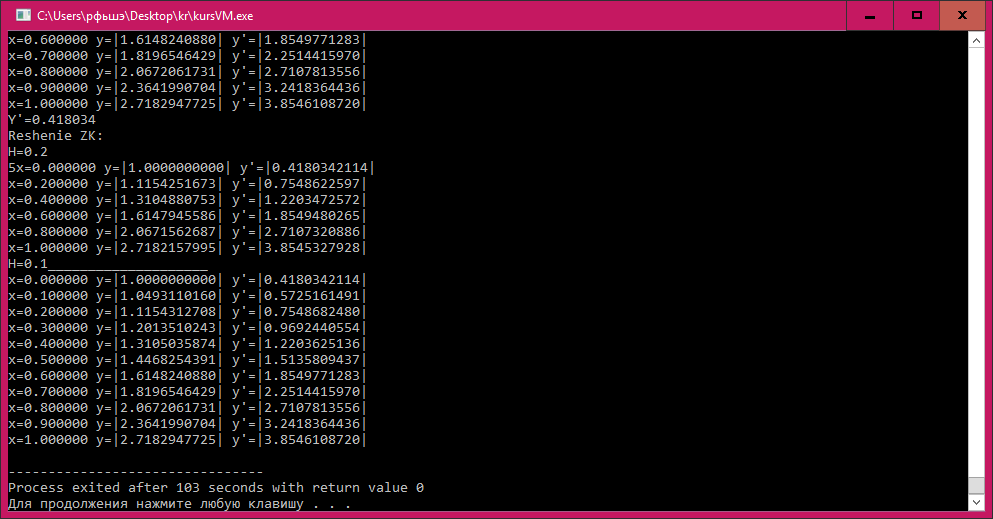


Рисунок 1. Решение краевой задачи

# Листинг программы

#include <stdio.h>

#include <math.h>

#include <conio.h>

#include <iostream>

//#define y2 (exp(x)+Y[0])/2

//#define y2 x\*Y[0]+Y[1]

#define y2 exp(x)+Y[1]

const int n=2;

using namespace std;

double\* F(double x,double\* Y){

double \*y=new double[n];

y[0]=Y[1];

y[1]=y2;

//cout<<endl<<Y[0]<<Y[1]<<endl;

return y;

}

double\* summ(double\* Y,double\* k,double h){

double \*y=new double[n];

for(int i=0;i<n;i++){

y[i]=Y[i]+(h\*k[i]);

}

return y;

}

double\* Yi(double\* Y,double x,double h){

double \*\*k=new double\*[4];

//double \*K=new double[n];

for(int i=0;i<n;i++){

k[i]= new double[n];

}

k[0]=F(x,Y);

k[1]=F(x+(h/2),summ(Y,k[0],h/2));

k[2]=F(x+(h/2),summ(Y,k[1],h/2));

k[3]=F(x+(h),summ(Y,k[2],h));

// for(int i=0;i<4;i++){

// cout<<"|"<<k[i][0]<<"|"<<endl<<"|"<<k[i][1]<<"|"<<endl<<endl;

// }

return summ(Y,summ(summ(k[0],k[1],2),summ(k[3],k[2],2),1),h/6);

}

bool comparison(double\*\* y, double\*\* yy,int N,double e){

for(int i=0;i<=N;i+=2){

if(fabs((y[i/2][1]-yy[i][1]))>=15\*e)

return false;

}

return true;

}

double\* Yi(double\* Y1,double x,double h, int b,double e){

double c=0,\*Y=new double[n], \*nY=new double[n],\*\*y,\*\*yy;

int nn;

do{

Y[0]=nY[0]=Y1[0];

Y[1]=nY[1]=Y1[1];

cout<<"H="<<h<<""<<endl;

nn=(int)((b-x)/h);

cout<<nn;

y=new double \*[nn+1];

for(int i=0;i<=nn;i++){

y[i]= new double[n];

}

y[0]=Y;

printf("x=%f y=|%10.10f| y'=|%10.10f|\n",x,y[0][0],y[0][1]);

for(int i=1;i<=nn;i++){

Y=Yi(Y,x+(h\*i)-h,h);

y[i]=Y;

// cout<<"x="<<x+(h\*i)<<"\t|"<<Y[0]<<" |"<<"\t|"<<Y[1]<<"|"<<endl<<endl;

printf("x=%f y=|%10.10f| y'=|%10.10f|\n",x+(h\*i),y[i][0],y[i][1]);

}

h=h/2;

cout<<"H="<<h<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

nn=(int)((b-x)/h);

yy=new double \*[nn+1];

for(int i=0;i<nn;i++){

yy[i]= new double[n];

}

yy[0]=nY;

printf("x=%f y=|%10.10f| y'=|%10.10f|\n",x,yy[0][0],yy[0][1]);

for(int i=1;i<=nn;i++){

nY=Yi(nY,x+(h\*i)-h,h);

yy[i]=nY;

// cout<<"x1="<<x+(h\*i)<<"\t|"<<nY[0]<<" |"<<"\t|"<<nY[1]<<"|"<<endl<<endl;

printf("x=%f y=|%10.10f| y'=|%10.10f|\n",x+(h\*i), yy[i][0],yy[i][1]);

}

c++;

h=h/2;

}while(!comparison(y,yy,nn,e));

return yy[nn];

}

double searchCor(double\* Y,double x,double h, int b,double e){

double l,ll,c,temp, \*y=new double[n],\*yy=new double[n];

ll=l=atan((Y[1]-Y[0])/(b-x));

//cout<<ll<<endl;

do{

y[1]=ll;

y[0]=Y[0];

//cout<<ll<<";"<<Y[0]<<endl;

y=Yi(y, x, h,b,e);

if(y[0]<Y[1])

ll+=e;

else if(y[0]>Y[1])

ll-=e;

// cout<<y[1]<<";"<<Y[1]<<endl;

}while(fabs(y[0]-Y[1])>e);

if(l>ll){

temp=l;

l=ll;

ll=temp;

}

cout<<l<<";"<<ll<<endl;

do{

cout<<l<<";"<<ll<<endl;

c=(l+ll)/2;

yy[1]=l;

y[1]=c;

yy[0]=y[0]=Y[0];

if((Yi(y, x, h,b,e)[0]-Y[1])\*(Yi(yy, x, h,b,e)[0]-Y[1]) < 0)

ll=c;

else

l=c;

}while((yy[0]-Y[1])>=e);

return yy[1];

}

int main()

{

double y=2,y1=4,x=1 ,e, h , a=0, \*Y=new double[n], \*nY=new double[n], \*nnY=new double[n], \*nnnY=new double[n];

int b;

x=0;

h=0.2;

e=0.00001;

b=1;

Y[0]=nY[0]=1;

Y[1]=2.71828;

nY[1]=searchCor(Y,x,h, b,e);

cout<<"Y'="<<nY[1]<<endl<<"Reshenie ZK:"<<endl;

nY=Yi(nY, x, h,b,e);

// cout<<"\t|"<<nY[0]<<"|"<<endl<<"\t|"<<nnnY[1]<<"|"<<endl<<endl;

}