

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 №1**

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) – 2017

Выполнил:

студент гр. ИП-712

«04» мая 2019 г.

\_\_\_\_\_

/Алексеев С.В./

Оценка « \_\_\_\_\_ »

Проверил:

доцент Кафедры ПМиК

« » мая 2019 г.

\_\_\_\_\_

/Рубан А.А./

Новосибирск, 2019

## Содержание

Введение	3
Постановка задачи	3
Основные идеи и характеристики применяемых методов	4
1. Метод Рунге-Кутты 4го порядка.	4
2. Метод стрельб	5
3. Оценка погрешности решения ДУ и СДУ методом двойного пересчета. Коррекция решения.	5
4. Пересчёт производной по свойствам векторов	7
Описание программы	7
Результаты	8
Листинг программы	9



## 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:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = y_0 \\ y(b) = y_1 \end{cases} \quad (1)$$

А мы умеем решать:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = y_0 \\ y'(a) = y'_0 = k \end{cases} \quad (2),$$

В (2) нам известно  $y'(a)$ , поэтому для решения задачи (1) мы будем подбирать  $y'(a)$  в (2), с тем, чтобы  $y(b) = y_1$ .

### Постановка задачи

1. Решить краевую задачу методом Рунге-Кутты IV

$$y'' = (y + e^x)/2$$

$$y(0) = 1;$$

$$y(1) = 2.7182818284590;$$

## Основные идеи и характеристики применяемых методов

### 1. Метод Рунге-Кутты 4го порядка.

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

Формулы метода Рунге-Кутты 4го порядка:

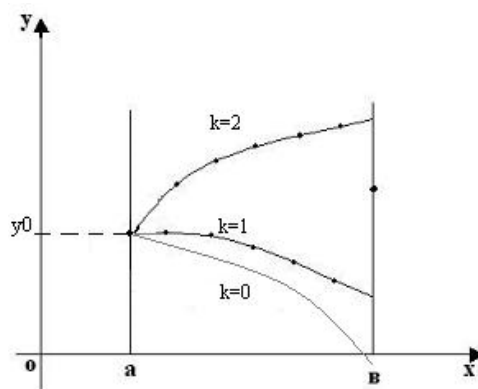
$$\begin{aligned}k_1 &= f(x_i, y_i) \\k_2 &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\k_3 &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\k_4 &= f(x_i + h, y_i + hk_3)\end{aligned}\quad (6.7)$$

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

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

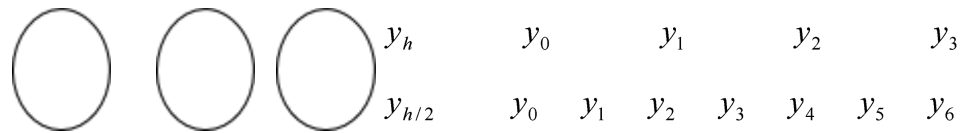
### 2. Метод стрельб

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  $q(k) = y_1$ , где  $q(k)$  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  $|y_{h(k)} - y_{h/2(k)}| < 3\varepsilon$  для метода 2го порядка,  $|y_{h(k)} - y_{h/2(k)}| < 15\varepsilon$  для метода 4го порядка, то в качестве точного решения берём  $y_{h/2}$ . Если же точность не достигнута, то шаг  $h$  уменьшаем вдвое и т.д., пока она не будет достигнута.

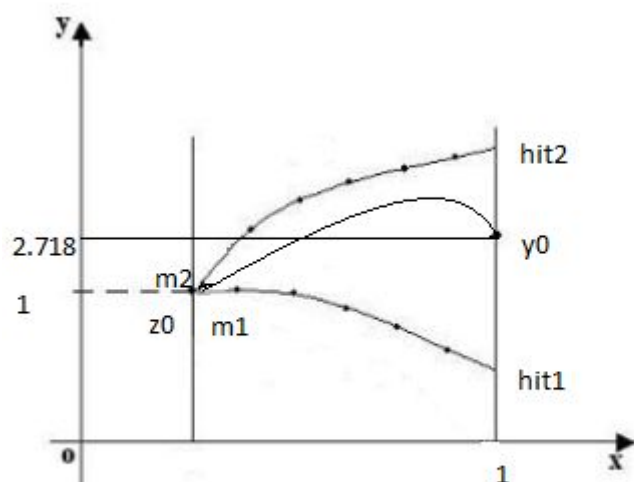
Метод двойного пересчёта при решении ДУ и СДУ практически единственный имеет возможность для оценки погрешностей, так как иные формулы очень сложны и требуют оценок различных производных.

Как и при ЧИ, при решении ДУ и СДУ после 2го пересчёта в качестве точного решения выгодно брать не  $y_{h/2}$ , а  $y_{кор}$ .

$$y_{кор} = y_{h/2} + \frac{1}{3}(y_{h/2} - y_h) \quad \text{ - для второго порядка}$$

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

4. *Пересчёт производной по свойствам вектора.* После первых двух выстрелов предполагается, что векторы  $(m_2, \text{hit}_2)$ ,  $(m_1, \text{hit}_1)$ ,  $(z_0, y_0)$  коллинеарны и в соответствии с этим считается уточнённая  $m_3$  для следующего “выстрела”.



## Описание программы

1. `void shooting()` - Ф у н к ц и я з а п у с к а е т  
“с т р е л ь б у” п о м е т о д у с т р е л ь б ,  
п о д б и р а я н у ж н ы е з н а ч е н и я п е р в о й  
п р о и з в о д н о й . П р е д п о л а г а е т с я , ч т о  
в е к т о р ы  $(m_2, hit_2)$ ,  $(m_1, hit_1)$ ,  $(z_0, y_0)$  к о л л и н е а р н ы  
и в с о о т в е т с т в и и с э т и м с ч и т а е т с я  
у т о ч н ё н н а я  $m_3$ .
2. `double RK4DoubleCounting(double m)` - ф у н к ц и я  
о с у щ е с т в л я е т д в о й н о й п е р е с ч ё т ,  
з а п у с к а я м е т о д Р у н г е - К у т т ы 4  
п о р я д к а с р а з н ы м ш а г о м д о  
п о л у ч е н и я н у ж н о й т о ч н о с т и .  
П р и н и м а е т о ч е р е д н о е  
п е р е с ч и т а н н о е з н а ч е н и е п е р в о й  
п р о и з в о д н о й
3. `double RK4(double h, int range, double[] yy, double z0)` -  
р е а л и з а ц и я м е т о д а Р у н г е - К у т т ы 4  
п о р я д к а
4. `double f(double x, double y, double z)` - ф у н к ц и я ,  
я в л я ю щ а я с я п р а в о й ч а с т ь ю  
д и ф ф е р е н ц и а л ь н о г о у р а в н е н и я  
в т о р о г о п о р я д к а  $y'' = (y + e^x)/2$
5. `double g(double x, double y, double z)` - ф у н к ц и я ,  
н е о б х о д и м а я д л я р е а л и з а ц и и м е т о д а  
Р у н г е - К у т т ы д л я у р а в н е н и я в т о р о г о  
п о р я д к а . В о з в р а щ а е т з н а ч е н и е  
п е р в о й п р о и з в о д н о й









## Результаты

```
0,8562500000000 2,3543154363285 2,3543154363391
0,8625000000000 2,3690759864761 2,3690759864867
0,8687500000000 2,3839290789557 2,3839290789663
0,8750000000000 2,3988752939680 2,3988752939786
0,8812500000000 2,4139152153515 2,4139152153622
0,8875000000000 2,4290494306051 2,4290494306158
0,8937500000000 2,4442785309110 2,4442785309216
0,9000000000000 2,4596031111577 2,4596031111684
0,9062500000000 2,4750237699637 2,4750237699744
0,9125000000000 2,4905411097005 2,4905411097112
0,9187500000000 2,5061557365160 2,5061557365267
0,9250000000000 2,5218682603587 2,5218682603694
0,9312500000000 2,5376792950010 2,5376792950117
0,9375000000000 2,5535894580634 2,5535894580741
0,9437500000000 2,5695993710387 2,5695993710495
0,9500000000000 2,5857096593162 2,5857096593270
0,9562500000000 2,6019209522061 2,6019209522169
0,9625000000000 2,6182338829640 2,6182338829748
0,9687500000000 2,6346490888159 2,6346490888267
0,9750000000000 2,6511672109828 2,6511672109936
0,9812500000000 2,6677888947060 2,6677888947169
0,9875000000000 2,6845147892721 2,6845147892830
0,9937500000000 2,7013455480386 2,7013455480495
1,0000000000000 2,7182818284590 2,7182818284699
```

The precise counted y is hitAvg: 2.718281828458998, m3 = 1.00000000000138682

n  6: TODO  9: Version Control  Terminal  0: Messages

```

import javafx.application.Application;

import javafx.stage.Stage;


import java.text.DecimalFormat;


public class Main extends Application {


    double x0 = 0, xn = 1, y0 = 1, z0 = 0.7, precY = 2.7182818284590, yn = precY,
        m1 = z0, m2 = 1.2, m3, z1, h = 0.1, H = 2 * h,
        yy[], YY[], hit1, hit2, hitAvg, eps = 10E-10;
    int range, RANGE;


    public void start(Stage stage) {
        shooting();
    }


    public void shooting() {
        hit1 = RK4DoubleCounting(m1);
        hit2 = RK4DoubleCounting(m2);
        if (Math.abs(hit1 - precY) < eps) System.out.println("The precise counted y is hit1: " + hit1);
        else if (Math.abs(hit2 - precY) < eps) System.out.println("The precise counted y is hit2: " + hit2);
        else {
            m3 = m2 + (((m2 - m1) * (precY - hit2)) / ((hit2 - hit1)));
            hitAvg = RK4DoubleCounting(m3);
        }
        if (Math.abs(hitAvg - precY) < eps) System.out.println("The precise counted y is hitAvg: " + hitAvg + ",
m3 = "+m3);
        else
            do {
                m1 = m2;
                m2 = m3;
            }
    }
}

```

```

        hit1 = hit2;
        hit2 = hitAvg;
        m3 = m2 + (((m2 - m1) * (precY - hit2)) / ((hit2 - hit1)));
        z0 = m3;
        hitAvg = RK4DoubleCounting(m3);
    } while (Math.abs(hitAvg - precY) < eps);
}

```

```

public double RK4DoubleCounting(double m) {
    System.out.println("X\t\t\t\tY\t\t\t\tY");
    double result;
    do {
        RANGE = (int) ((xn - x0) / h);
        range = (int) ((xn - x0) / H);
        YY = new double[range];
        System.out.println("RK4 launch with H = " + H);
        RK4(H, range, YY, m);
        yy = new double[RANGE];
        System.out.println("RK4 launch with h = " + h);
        result = RK4(h, RANGE, yy, m);
        System.out.println();
        h *= 0.5;
        H = 2 * h;
    } while (Math.abs(yy[RANGE - 1] - YY[range - 1]) > eps);
    h = 0.1; H = 2 * h;
    return result;
}

```

```

public double RK4(double h, int range, double[] yy, double z0) {
    double xc = x0, yc = y0, zc = z0, y1c = 1, k1, k2, k4, k3, k11, k22, k44, k33;

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

```

```

String x00 = new DecimalFormat("#0.000000000000000").format(xc);
String y11 = new DecimalFormat("#0.000000000000000").format(y1c);
String z11 = new DecimalFormat("#0.000000000000000").format(zc);
System.out.println(x00 + "\t" + y11 + "\t" + z11);

k1 = h * f(xc, yc, zc);
k11 = h * g(xc, yc, zc);
k2 = h * f(xc + h / 2.0, yc + k11 / 2.0, zc + k1 / 2.0);
k22 = h * g(xc + h / 2.0, yc + k11 / 2.0, zc + k1 / 2.0);
k3 = h * f(xc + h / 2.0, yc + k22 / 2.0, zc + k2 / 2.0);
k33 = h * g(xc + h / 2.0, yc + k22 / 2.0, zc + k2 / 2.0);
k4 = h * f(xc + h, yc + k33, zc + k3);
k44 = h * g(xc + h, yc + k33, zc + k3);
zc = zc + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0;
yy[i] = y1c = yc + (k11 + 2.0 * k22 + 2.0 * k33 + k44) / 6.0;
yc = y1c;
xc += h;
}

String x00 = new DecimalFormat("#0.000000000000000").format(xc);
String y11 = new DecimalFormat("#0.000000000000000").format(y1c);
String z11 = new DecimalFormat("#0.000000000000000").format(zc);
System.out.println(x00 + "\t" + y11 + "\t" + z11);
return y1c;
}

public static void main(String[] args) {
    launch(args);
}

public double f(double x, double y, double z) {
    double result = (Math.exp(x) + y) / 2;
    return result; //return Math.pow(x, 4) / 7;
}

```

```
public double g(double x, double y, double z) {  
    return (z);  
}  
}
```