Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

Отчёт по лабораторной работе № 4

«Метод Рунге-Кутта численного решения задачи Коши для системы обыкновенных дифференциальных уравнений»
«Численное решение краевой задачи для линейного дифференциального уравнения второго порядка методом прогонки»

по курсу

«Численные методы»

Студент группы ИУ9-61Б Бакланова А.Д.

Преподаватель Домрачева А.Б.

Москва, 2020 г.

Цель работы

Целью данной работы является реализация *метода Рунге-Кутта* численного решения задачи Коши для системы ОДУ и численного решения краевой задачи для линейного ДУ второго порядка *методом прогонки*.

Задание

Метод Рунге-Кутта:

Найти численно с погрешностью $\varepsilon = 0.001$ решение задачи Коши для ДУ второго порядка, привести его к СОДУ первого порядка, используя метод Рунге-Кутта. Найти точное решение ДУ.

Метод прогонки:

Написать и отладить процедуру для решения трехдиагональной линейной системы методом прогонки. Решить задачу Коши.

Дано:

$y'' + py' + qy = f(x)$

$y(0) = y_0, \quad y'(0) = y'_0$

Теоретические сведения:

Метод Рунге-Кутта применяется для решения задачи КОши для системы ОДУ первого порядка

$y'_1 = f_1(x, y_1, y_2, \ldots, y_n)$

$y'_2 = f_2(x, y_1, y_2, \ldots, y_n)$

. . . .

$y'_n = f_n(x, y_1, y_2, \ldots, y_n)$

на отрезке $[x_0, x_{end}]$ с начальными условиями

$y_1(x_0) = y_{01}, . . , y_n(x_0) = y_{0n}$

Сам метод заключается в последовательном нахождении вектор-коэффициентов $k_1,\ k_2,\ k_3$ и k_4 по формулам

$$k_1 = f(x, y)$$

$$k_2 = f(x + h/2,\quad y + hk_1/2)$$

$$k_3 = f(x + h/2,\quad y + hk_2/2)$$

$$k_4 = f(x + h/2,\quad y + hk_3/2)$$

и построении приближения к решению СОДУ в точке $x + h$:

$$y(x + h) = y_h = y + \frac{h}{6}(k_1 + 2(k_2 + k_3) + k_4)$$

$$err = \frac{1}{2^p - 1}||y_h - y_2h||$$

Метод прогонки:

Краевая задача для линейного ДУ второго порядка имеет вид

$$y'' + p(x)y' + q(x)y = f(x)$$

$$y(0) = a,\ \ y(1) = b$$

Требуется найти частное решение уравнения, отвечающее краевым условиям.

Приближенным численным решением задачи называется <u>сеточная функция</u> $(x_i, y_i),\ i = 0,..,n$ заданная в $(n + 1)$-й точке $x_i = ih$, $h = 1/n$

$$p_i = p(x_i)$$

$$q_i = q(x_i)$$

$$f_i = f(x_i)$$

Из этого следует система уравнений относительно ординат сеточной функции y_i :

$$\frac{y_{i+1} - 2y_i + 2y_{i-1}}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = f_i$$

или после преобразований

$$y_{i-1}(1 - \frac{h}{2} p_i) + y_i(h^2 q_i - 2) + y_{i+1}(1 + \frac{h}{2} p_i) = h^2 f_i$$

$i = 1,..,n-1$

с краевыми условиями $y_0 = a$ и $y_n = b$

Входные данные:

$y'' - 7y' + 12y = 5$

$y(0) = 1, \quad y'(0) = 2$

Метод Рунге-Кутта:

Lab4.java

```java
public class Lab4 {
    //Методы Рунге-Кутта численного решения задачи Коши...

    public static void main(String[] args) {
        double x0, y0, y1, y0_1, z1;
        double k1, k2, k4, k3;
        double p, q, h, n, a, b;

        x0 = 0;
        y0 = 1;
        y0_1 = 2; //y'

        a = 0.0;
        b = 1.0;
        n = 10;

        h = (b - a)/n; // шаг

        //2 вариант p = -7; q = 12
        p = -7;
        q = 12;

        System.out.println("\t   X\t\t   Y\t\t   Y'");
        for (; x0 < b-h; x0 += h) {

            k1 = h * f(x0, y0, y0_1, p, q);

            k2 = h * f( x: x0 + h/2.0,  y: y0 + (h * y0_1)/2.0,  z: y0_1 + k1/2.0, p, q);

            k3 = h * f( x: x0 + h / 2.0,  y: y0 + (h * (y0_1 + k1 / 2.0)) / 2.0,  z: y0_1 + k2 / 2.0, p, q);

            k4 = h * f( x: x0 + h,  y: y0 + (h * (y0_1 + k2 / 2.0)),  z: y0_1 + k3, p, q);

            z1 = y0_1 + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0;
            y1 = y0 + ((h * y0_1) + 2.0 * (h * (y0_1 + k1 / 2.0)) + 2.0 * (h * (y0_1 + k2 / 2.0))+ (h * (y0_1 + k3))) / 6.0;

            System.out.printf("\t %.4f \t %.4f \t %.4f \n", (x0+h), y1, z1);
            y0 = y1;
            y0_1 = z1;
        }
    //2 вариант f(x) = 5
    public static double f(double x, double y, double z, double p, double q) { return (5 - p*z - q*y); }

}
```

Результат:

```
        X           Y           Y'
     0.1000      1.2395      2.8416
     0.2000      1.5803      4.0473
     0.3000      2.0664      5.7790
     0.4000      2.7613      8.2717
     0.5000      3.7571      11.8681
     0.6000      5.1878      17.0682
     0.7000      7.2478      24.6029
     0.8000      10.2207     35.5423
     0.9000      14.5204     51.4557
     1.0000      20.7520     74.6471

Process finished with exit code 0
```

Метод прогонки:

```cpp
#include <iostream>
#include <vector>
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <iomanip>
#include <limits>
#include <algorithm>
using namespace std;
double n = 10, a = 0, b = 1, A = 0, B = 1;
double p = -7;
double q = 12;
double f = 5;

double differential(double x) {
    double c1 = (A - (1/12 * (4*exp(3*a)) + (3*exp(4*a)) + 5) - B + (1/12 * (4*exp(3*b)) + (3*exp(4*b)) + 5)) / (a - b),
    c2 = B - (1/12 * (4*exp(3*b)) + (3*exp(4*b)) + 5) - c1 * b;

    return (1/12 * (4*exp(3*x)) + (3*exp(4*x)) + 5) + c1 * x + c2;
}

double h = (b - a) / n;
double xi = a + h;
vector<double> y1, y, di, ai, bi, ci;
bi.push_back(q(xi) * h * h - 2);
ci.push_back(1 + p(xi) * h / 2);
di.push_back(f(xi) * h * h - A * (1 - p(xi) * h / 2));

for (int i = 2; i < n - 1; i++) {
    xi = a + i * h;
    ai.push_back(1 - p(xi) * h / 2);
    bi.push_back(q(xi) * h * h - 2);
    ci.push_back(1 + p(xi) * h / 2);
    di.push_back(f(xi) * h * h);
}

xi = a + h * (n - 1);

bi.push_back(q(xi) * h * h - 2);
ai.push_back(1 - p(xi) * h / 2);
di.push_back(f(xi) * h * h - B * (1 + p(xi) * h / 2));
ci.push_back(0);
tridiag(y1, ai, bi, ci, di);
```

Результат:

```
0 0
0.10463394 0.1
0.16459716 0.2
0.22140758 0.3
0.28595756 0.4
0.36488384 0.5
0.46461467 0.6
0.59271875 0.7
0.75876719 0.8
0.22571048 0.9
1 1
```

Вывод:

В результате выполнения лабораторной работы были реализованы метод Рунге-Кутта численного решения задачи Коши для СОДУ и метод прогонки для решения краевой задачи. Метод прогонки работает чуть медленнее метода Рунге-Кутта на данных входных данных. Также метод Рунге-Кутта является более экономичным и простым в настройке, имеет 4 порядок точности. Метод прогонки не универсален.