Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4
Задание №2 по курсу «Численные методы»
Вариант №12

Студент:	С. М. Власова
Преподаватель:	И. Э. Иванов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Задание №4.1

Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Вариант: 12

Краевая задача:

$$\begin{cases} x(x-1)y'' - xy' + y = 0 \\ y'(2) = 3 + 2ln2 \\ y(3) - 3y'(3) = -4. \end{cases} \tag{1}$$

Точное решение: $y(x) = 2 + x + 2x \cdot ln|x|$.

1 Описание метода решения

1 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

1.1 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ КРАЕВОЙ ЗАДАЧИ ДЛЯ ОДУ

Примером краевой задачи является двухточечная краевая задача для обыкновенного дифференциального уравнения второго порядка.

$$y'' = f(x, y, y') \tag{2}$$

с граничными условиями, заданными на концах отрезка $[a, b]$.

$$y(a) = y_0$$

$$y(b) = y_1.$$

Следует найти такое решение $y(x)$ на этом отрезке, которое принимает на концах отрезка значения y_0, y_1. Если функция $f(x, y, y')$ линейна по аргументам y и y', то задача является *линейной краевой*, в противном случае — нелинейной.

Кроме граничных условий, называемых *граничными условиями первого рода*, используются еще условия на на производные от решения на концах — *граничные условия второго рода:*

$$y'(a) = \hat{y}_0$$
$$y'(b) = \hat{y}_1.$$

или линейная комбинация решений и производных — *граничные условия третьего рода:*

$$\alpha y(a) + \beta y'(a) = \hat{y}_0$$
$$\delta y(b) + \gamma y'(b) = \hat{y}_1,$$

где $\alpha, \beta, \delta, \gamma$ — такие числа, что $|\alpha| + |\beta| \neq 0, \quad |\delta| + |\gamma| \neq 0$.
Возможно на разных концах отрезка использовать условия различных типов.
В данном пособии рассматриваются два приближенных метода решения краевой задачи:

1. **метод стрельбы (пристрелки);**

2. **конечно-разностный метод.**

Метод стрельбы.

Суть метода заключена в многократном решении задачи Коши для приближенного нахождения решения краевой задачи.
Пусть надо решить краевую задачу на отрезке $[a, b]$. Вместо исходной задачи формулируется задача Коши с уравнением (2) и с начальными условиями

$$y(a) = y_0$$

$$y'(a) = \eta,$$

где η — некоторое значение тангенса угла наклона касательной к решению в точке $x = a$.

Положим сначала некоторое начальное значение параметру $\eta = \eta_0$, после чего решим каким-либо методом задачу Коши. Пусть $y = y_0(x, y_0, \eta_0)$ решение этой задачи на интервале $[a, b]$, тогда сравнивая значение функции $y_0(b, y_0, \eta_0)$ со значением y_1 в правом конце отрезка можно получить информацию для корректировки угла наклона касательной к решению в левом конце отрезка. Решая задачу Коши для нового значения $\eta = \eta_1$, получим другое решение со значением $y_1(b, y_0, \eta_1)$ на правом конце.Таким образом, значение решения на правом конце $y(b, y_0, \eta)$ на правом конце будет являться функцией одной переменной η. Задачу можно сформулировать таким образом: требуется найти такое значение переменной $\eta^\star$, чтобы решение $y(b, y_0, ^\star)$ в правом конце отрезка совпало со значением y_1. Другими словами, решение исходной задачи эквивалентно нахождению корня уравнения

$$\Phi(\eta) = 0,$$

где

$$\Phi(\eta) = y(b, y_0, \eta) - y_1.$$

Это уравнение является "алгоритмическим" уравнением, так как левая часть его задается с помощью алгоритма численного решения соответствующей задачи Коши. Но методы решения уравнения аналогичны методам решения нелинейных уравнений, изложенным в разделе 2.

Следует заметить, что, так как невозможно вычислить производную функции $\Phi(\eta)$, то вместо метода Ньютона следует использовать метод секущих, в котором производная от функции заменена ее разностным аналогом. Данный разностный аналог легко вычисляется по двум приближениям, например η_k и η_{k+1}.
Следующее значение искомого корня определяется по соотношению

$$\eta_{j+2} = \eta_{j+1} - \frac{\eta_{j+1} - \eta_j}{\Phi(\eta_{j+1}) - \Phi(\eta_j)} \cdot \Phi(\eta_{j+1}).$$

Итерации по этой формуле выполняется до удовлетворения заданной точности.

Конечно-разностный метод.

Рассмотрим двухточечную краевую задачу для линейного дифференциального уравнения второго порядка на отрезке $[a, b]$

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

$$y(a) = y_0, \quad y(b) = y_1.$$

Введем разностную сетку на отрезке $[a, b]$: $\Sigma^{(h)} = \{x_k = x_0 + hk\}$, $k = 0, 1, \dots, N$, $h = |b - a|/N$. Решение краевой задачи будем искать в виде сеточной функции $y(h) = \{y_k, \quad k = 0, 1, \dots, N\}$, предлагая, что решение существует и единственно. Введем разностную аппроксимацию производных следующим образом:

$$y'_k = \frac{y_{k+1} - y_{k-1}}{2h} + O(h^2);$$

$$y''_k = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + O(h^2).$$

Подставляя аппроксимации производных, получим систему уравнений для нахождения y_k :

$$\begin{cases} y_0 = y_a \\ \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + p(x_k) \cdot \frac{y_{k+1} - y_{k-1}}{2h} + q(x_k) y_k = f(x_k), \qquad k = 1, \dots, N-1 \\ y_N = y_b. \end{cases} \tag{3}$$

Приводя подобные и учитывая, что при задании граничных условий первого рода два неизвестных $y_0, \quad y_N$ уже фактически определены, получим систему линейных алгебраических уравнений с трехдиагональной матрицей коэффициентов

$$\begin{cases} (-2 + h^2 \cdot q(x_1)) \cdot y_1 + (1 + \frac{p(x_1)h}{2}) \cdot y_2 = h^2 \cdot f(x_1) - (1 - \frac{p(x_1)h}{2}) y_a \\ (1 - \frac{p(x_k)h}{2}) y_{k-1} + (-2 + h^2 \cdot q(x_k)) y_k + (1 + \frac{p(x_k)h}{2}) \cdot y_{k+1} = h^2 \cdot f(x_k), \qquad k = 2, \dots, N-2 \\ (1 - \frac{p(x_{N-1})h}{2}) y_{N-1} + (-2 + h^2 \cdot q(x_{N-1})) y_{N-1} = h^2 \cdot f(x_{N-1}) - (1 + \frac{p(x_{N-1})h}{2}) y_b \end{cases} \tag{4}$$

Для системы при достаточно малых шагах сетки h и $q(x_k) < 0$ выполнены условия преобладания диагональных элементов

$$|-2 + h^2 \cdot q(x_k)| > |1 - \frac{p(x_k)h}{2}| + |1 + \frac{p(x_k)h}{2}|,$$

что гарантирует устойчивость счета и корректность применения метода прогонки для решения этой системы.

В случае использования граничных условий второго и третьего рода аппроксимация производных проводится с помощью односторонних разностей первого и второго порядков.

$$y'_0 = \frac{y_1 - y_0}{h} + O(h)$$

$$y'_N = \frac{y_N - y_{N-1}}{h} + O(h)$$

$$y'_0 = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2)$$

$$y'_N = \frac{y_{N-2} - 4y_{N-1} + 3y_N}{2h} + O(h^2).$$

В случае использования первых двух формул линейная алгебраическая система аппроксимирует дифференциальную задачу в целом только с первым порядком (из-за аппроксимации в граничных точках), однако сохраняется трех диагональная структура матрицы коэффициентов. В случае использования последних двух формул второй порядок аппроксимации сохраняется везде, но матрица линейной системы не трехдиагональная.

2 Моя задача

Краевая задача:

$$\begin{cases} x(x-1)y'' - xy' + y = 0 \\ y'(2) = 3 + 2ln2 \\ y(3) - 3y'(3) = -4. \end{cases} \tag{5}$$

Точное решение: $y(x) = 2 + x + 2x \cdot ln|x|$.
Решение краевой задачи методом стрельбы:
Сведем ОДУ второго порядка к решению системы ОДУ первого порядка. Для этого введем новые переменные.
Пусть $y = y_1$, тогда имеем:

$$\begin{cases} y_1' = y_2 \\ y_2' = \frac{xy_2 - y_1}{x(x-1)} \\ y_2(2) = 3 + 2ln2 \\ y_1(2) = \eta \end{cases} \tag{6}$$

Так же в качестве проверочного условия имеем граничное условие третьего рода:

$$y_2(3) = \frac{y_1(3) + 4}{3}.$$

Итак, необходимо подобрать такой параметр $\eta^\star$, чтобы значение функции $y_2(\eta^\star)$ в точке $x = 3$ совпало со значением $f = \frac{y_1(3)+4}{3}$.
В качестве первых двух значений я взяла $\eta_0 = 2$ и $\eta_1 = 2.8$ и, вычислив методом Рунге-Кутты значения функций $y_1^{(0)}$ и $y_1^{(1)}$, нашла значения функций

$$\Phi_0 = \left| y_2^{(i)}(3) - \frac{y_1^{(i)}(3) + 4}{3} \right|, \quad i = 0, 1.$$

Далее, по формуле вычисляется следующее значение η_2 :

$$\eta_2 = \eta_1 - \frac{\eta_1 - \eta_0}{\Phi(\eta_1) - \Phi(\eta_0)} \cdot \Phi(\eta_1).$$

и аналогично находится решение методом Рунге-Кутты $y_1^{(2)}$, соответствующая ему функция Φ_2. Поиск решения продолжается, пока не будет выполняться условие

$$|\Phi(\eta^\star)| \leq \varepsilon,$$

т.е. решение не достигнет заданной точности.

Решение краевой задачи конечно-разностным методом:
Представим уравнение в следующем виде

$$\begin{cases} y'' + p(x)y' + q(x)y = f(x) \\ \mu_0 y'(a) + \nu_0 y(a) = y_a \\ \mu_1 y'(b) + \nu_1 y(b) = y_b \end{cases} \tag{7}$$

Имеем

$$\begin{cases} y'' - \frac{1}{x-1} \cdot y' + \frac{1}{x(x-1)} \cdot y = 0 \\ y'(2) = 3 + 2ln2 \\ -3 \cdot y'(3) + y(3) = -4 \end{cases} \tag{8}$$

Итак,

$$p(x) = -\frac{1}{x-1}, \quad q(x) = \frac{1}{x(x-1)}, \quad f(x) = 0;$$

$$\nu_0 = 0, \quad \mu_0 = 1, y_a = 3 + 2ln2;$$

$$\nu_1 = 1, \quad \mu_1 = -3, y_b = -4.$$

Аппроксимируем производные с помощью односторонних разностей первого порядка

$$\begin{cases} (\nu_0 - \frac{\mu_0}{h}) \cdot y_0 + \frac{\mu_0}{h} \cdot y_1 = y_a \\ (\frac{1}{h^2} - \frac{p_i}{2h}) \cdot y_{i-1} - (\frac{2}{h^2} - q_i) \cdot y_i + (\frac{1}{h^2} + \frac{p_i}{2h}) \cdot y_{i+1} = f_i, \quad i = 1, \ldots, N-1 \\ -\frac{\mu_1}{h} \cdot y_{N-1} + (\nu_1 + \frac{\mu_1}{h}) \cdot y_N = x_b. \end{cases} \tag{9}$$

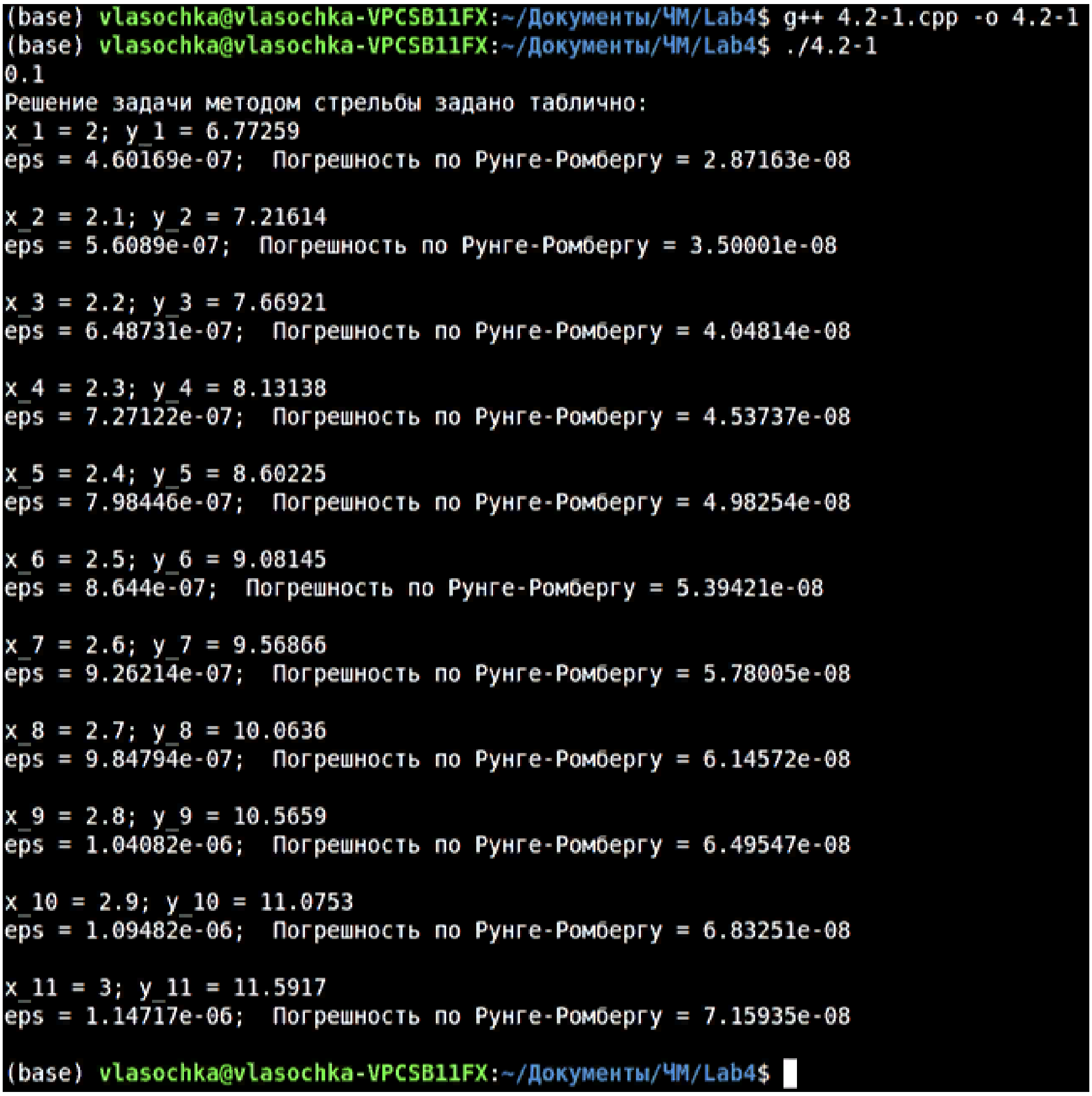
Эта система линейных уравнений соответствует трехдиагональной матрице, ее решение может быть найдено методом прогонки.
Вычислим матрицу для $h = 0.5, N = 2$:

$$\begin{cases} -2y_0 + 2y_1 = 3 + 2ln2 \\ \frac{14}{3}y_0 - \frac{116}{15}y_1 + \frac{10}{3}y_2 = 0 \\ -6y_1 - 5y_2 = -4. \end{cases} \tag{10}$$

3 Протокол

Входные данные я ввожу через консоль: значение шага h.
Выходные данные я вывожу на экран.
Скриншот консоли:

I. Метод стрельбы.



```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ g++ 4.2-1.cpp -o 4.2-1
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.2-1
0.1
Решение задачи методом стрельбы задано таблично:
x_1 = 2; y_1 = 6.77259
eps = 4.60169e-07;  Погрешность по Рунге-Ромбергу = 2.87163e-08

x_2 = 2.1; y_2 = 7.21614
eps = 5.6089e-07;  Погрешность по Рунге-Ромбергу = 3.50001e-08

x_3 = 2.2; y_3 = 7.66921
eps = 6.48731e-07;  Погрешность по Рунге-Ромбергу = 4.04814e-08

x_4 = 2.3; y_4 = 8.13138
eps = 7.27122e-07;  Погрешность по Рунге-Ромбергу = 4.53737e-08

x_5 = 2.4; y_5 = 8.60225
eps = 7.98446e-07;  Погрешность по Рунге-Ромбергу = 4.98254e-08

x_6 = 2.5; y_6 = 9.08145
eps = 8.644e-07;  Погрешность по Рунге-Ромбергу = 5.39421e-08

x_7 = 2.6; y_7 = 9.56866
eps = 9.26214e-07;  Погрешность по Рунге-Ромбергу = 5.78005e-08

x_8 = 2.7; y_8 = 10.0636
eps = 9.84794e-07;  Погрешность по Рунге-Ромбергу = 6.14572e-08

x_9 = 2.8; y_9 = 10.5659
eps = 1.04082e-06;  Погрешность по Рунге-Ромбергу = 6.49547e-08

x_10 = 2.9; y_10 = 11.0753
eps = 1.09482e-06;  Погрешность по Рунге-Ромбергу = 6.83251e-08

x_11 = 3; y_11 = 11.5917
eps = 1.14717e-06;  Погрешность по Рунге-Ромбергу = 7.15935e-08

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$
```

Решение краевой задачи методом стрельбы:

x	y	ε
2	6.77259	4.60169e-07
2.1	7.21614	5.6089e-07
2.2	7.66921	6.48731e-07
2.3	8.13138	7.27122e-07
2.4	8.60225	7.98446e-07
2.5	9.08145	8.644e-07
2.6	9.56866	9.26214e-07
2.7	10.0636	9.84794e-07
2.8	10.5659	1.04082e-06
2.9	11.0753	1.09482e-06
3	11.5917	1.14717e-06

Погрешность метода стрельбы небольшая.

II. Конечно-разностный метод.
Решение краевой задачи конечно-разностным методом с шагом $h = 0.5$:

x	y	ε
2	5.91545	0.857143
2.5	8.10859	0.972861
3	10.5303	1.06136

Решение краевой задачи конечно-разностным методом с шагом $h = 0.2$:

x	y	ε
2	6.4568	0.315789
2.2	7.33406	0.335154
2.4	8.24959	0.352655
2.6	9.20022	0.368441
2.8	10.1832	0.382637
3	11.1963	0.395354

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ g++ 4.2-2.cpp -o 4.2-2
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.2-2
0.5
Решение задачи конечно-разностным методом задано таблично:
x_1 = 2; y_1 = 5.91545
eps = 0.857143;  Погрешность по Рунге-Ромбергу = 0.0304762

x_2 = 2.5; y_2 = 8.10859
eps = 0.972861;  Погрешность по Рунге-Ромбергу = 0.0344268

x_3 = 3; y_3 = 10.5303
eps = 1.06136;  Погрешность по Рунге-Ромбергу = 0.0374334

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.2-2
0.2
Решение задачи конечно-разностным методом задано таблично:
x_1 = 2; y_1 = 6.4568
eps = 0.315789;  Погрешность по Рунге-Ромбергу = 0.0107962

x_2 = 2.2; y_2 = 7.33406
eps = 0.335154;  Погрешность по Рунге-Ромбергу = 0.0114474

x_3 = 2.4; y_3 = 8.24959
eps = 0.352655;  Погрешность по Рунге-Ромбергу = 0.0120358

x_4 = 2.6; y_4 = 9.20022
eps = 0.368441;  Погрешность по Рунге-Ромбергу = 0.0125661

x_5 = 2.8; y_5 = 10.1832
eps = 0.382637;  Погрешность по Рунге-Ромбергу = 0.0130426

x_6 = 3; y_6 = 11.1963
eps = 0.395354;  Погрешность по Рунге-Ромбергу = 0.013469

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$
```

Я аппроксимировала производные односторонними разностями первого порядка, так что погрешность получилось больше, чем в методе стрельбы.

4 Исходный код

Листинг 1: Метод стрельбы

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>

double ExactVal(double x)
{
    double val = 2 + x + 2*x*log(fabs(x));
    return val;
}

double F(double x, double y1, double y2)
{
    double value = (x*y2 - y1)/(x*(x-1));
    return value;
}

void Runge_Kutta(std::vector<double>& y, std::vector<double>& y2, std::vector<double>&
    eps, double h, double nu, int a, int b)
{
    int n = (b-a)/h + 1;
    double y_h = 1;
    int p = 4;
    std::vector<double> K(p);
    std::vector<double> L(p);
    y[0] = nu;
    y2[0] = 3 + 2*log(2);
    eps[0] = fabs(ExactVal(2) - y[0]);
    for(int i = 0; i < n-1; i++)
    {
        K[0] = h*y2[i];
        L[0] = h*F(a + h*i, y[i], y2[i]);
        K[1] = h*(y2[i] + L[0]/2);
        L[1] = h*F(a + h*i + h/2, y[i] + K[0]/2, y2[i] + L[0]/2);
        K[2] = h*(y2[i] + L[1]/2);
        L[2] = h*F(a + h*i + h/2, y[i] + K[1]/2, y2[i] + L[1]/2);
        K[3] = h*(y2[i] + L[2]);
        L[3] = h*F(a + h*i + h, y[i] + K[2], y2[i] + L[2]);

```

```
        y[i+1] = y[i] + (K[0] + 2*K[1] + 2*K[2] + K[3])/6;
        y2[i+1] = y2[i] + (L[0] + 2*L[1] + 2*L[2] + L[3])/6;
        eps[i+1] = fabs(ExactVal(a + h*(i+1)) - y[i+1]);
    }
}

void Shoot(std::vector<double>& y, std::vector<double>& eps, double h, int a, int b)
{
    int n = (b-a)/h + 1;
    double epsilon = 0.1;

    std::vector<double> y2(n);
    std::vector<double> Phi(3);
    std::vector<double> nu(3);

    nu[0] = 2;
    nu[1] = 2.8;
    Runge_Kutta(y, y2, eps, h, nu[0], a, b);
    Phi[0] = fabs(y2[n-1] - (y[n-1] + 4)/3);

    Runge_Kutta(y, y2, eps, h, nu[1], a, b);
    Phi[1] = fabs(y2[n-1] - (y[n-1] + 4)/3);
    nu[2] = nu[1] - (nu[1] - nu[0])/(Phi[1] - Phi[0])*Phi[1];

    {
        nu[2] = nu[1] - (nu[1] - nu[0])/(Phi[1] - Phi[0])*Phi[1];
        Runge_Kutta(y, y2, eps, h, nu[2], a, b);
        Phi[2] = fabs(y2[n-1] - (y[n-1] + 4)/3);

        nu[0] = nu[1];
        nu[1] = nu[2];

        Phi[0] = Phi[1];
        Phi[1] = Phi[2];

    }while(Phi[2] > epsilon);
}

int main()
{
    double k = 0.5;
    double h, h_r, x;
    int n, m;
    double y_h;
```

```
    int a = 2;
    int b = 3;

    std::cin >> h;
    h_r = h*k;
    n = (b-a)/h + 1;
    m = (b-a)/h_r + 1;

    std::vector<double> y(n);
    std::vector<double> eps(n);
    std::vector<double> rung(m);
    std::vector<double> eps2(m);

    Shoot(y, eps, h, a, b);
    Shoot(rung, eps2, h_r, a, b);

    std::cout << "The solution of the problem using the shooting method is set in a table:" << std::endl;
    for(int i = 0; i < n; i++)
    {
        std::cout << "x_" << i + 1 << " = " << a + i*h << "; y_" << i + 1 << " = " << y[i] << std::endl;
        std::cout << "eps = " << eps[i] << "; The error in the Runge-Romberg =" << fabs(y[i]-rung[i/k])/15 << std::endl << std::endl;
    }
    return 0;
}
```

Листинг 2: Конечно-разностный метод

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>

double ExactVal(double x)
{
    double val = 2 + x + 2*x*log(fabs(x));
    return val;
}

double func(double x)
{
```

```
    return 0;
}

double q_func(double x)
{
    double val = 1/(x*(x-1));
    return val;
}

double p_func(double x)
{
    double val = -1/(x-1);
    return val;
}

void FDM(std::vector<double>& y, std::vector<double>& eps, double h, int l, int r)
{
    int n = (r-l)/h + 1;
    double y_l = 3 + 2*log(2);
    double y_r = -4;
    double nu_0 = 0;
    double nu_1 = 1;
    double mu_0 = 1;
    double mu_1 = -3;
    double a, b, c, d, p, q, f;

    std::vector<double> P(n);
    std::vector<double> Q(n);

    for( int i = 0; i < n; i++)
    {
        p = p_func(l + h*i);
        q = q_func(l + h*i);
        f = func(l + h*i);
        if( i == 0)
        {
            b = nu_0 - mu_0/h;
            c = mu_0/h;
            d = y_l;

            P[i] = -c/b;
            Q[i] = d/b;
        }
        else if( i != n-1)
```

```
        {
            a = 1/pow(h, 2) - p/(2*h);
            b = -2/pow(h, 2) + q;
            c = 1/pow(h, 2) + p/(2*h);
            d = f;
            P[i] = -c/(b + a * P[i-1]);
            Q[i] = (d - a*Q[i-1])/(b + a*P[i-1]);
        }
        else
        {
            a = -mu_1/h;
            b = nu_1 + mu_1/h;
            d = y_r;
            P[i] = 0;
            Q[i] = (d - a*Q[i-1])/(b + a*P[i-1]);
        }
    }

    for( int i = n-1; i >= 0; i--)
    {
        if( i == n-1)
            y[i] = Q[i];
        else
            y[i] = P[i]*y[i + 1] + Q[i];
        eps[i] = fabs(ExactVal(l + h*i) - y[i]);
    }
}

int main()
{
    double k = 0.5;
    double h, h_r, x;
    int n, m;
    double y_h;
    int a = 2;
    int b = 3;

    std::cin >> h;
    h_r = h*k;
    n = (b-a)/h + 1;
    m = (b-a)/h_r + 1;

    std::vector<double> y(n);
    std::vector<double> eps(n);
```

```
    std::vector<double> rung(m);
    std::vector<double> eps2(m);

    FDM(y, eps, h, a, b);
    FDM(rung, eps2, h_r, a, b);

    std::cout << "The solution of the problem by the finite-difference method is given in a table: " << std::endl;
    for(int i = 0; i < n; i++)
    {
        std::cout << "x_" << i + 1 << " = " << a + i*h << "; y_" << i + 1 << " = " << y[i] << std::endl;
        std::cout << "eps = " << eps[i] << "; The error in the Runge-Romberg =" << fabs(y[i]-rung[i/k])/15 << std::endl << std::endl;
    }
    return 0;
}
```

5 Выводы

Выполнив второе задание четвертой лабораторной работы, я познакомилась с двумя методами решения краевой задачи для ОДУ. Суть метода стрельбы заключена в многократном решении задачи Коши для приближенного решения краевой задачи. Строится система ОДУ первого порядка с начальными условиями на левом конце отрезка — одно условие подобрано искусственно, равно параметру η. Поиск решения заключается в подборе такого значения параметра η, чтобы значение функции на правом конце отрезка совпадало с данным в задаче с заданной точностью. Конечно-разностный метод базируется на замене производных их конечно-разностными аппроксимациями — первого и второго порядка. Это позволяет получить СЛАУ с трехдиагональной матрицей, которая решается методом прогонки.