Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Численные методы»

Студент:	Ю. В. Кон
Преподаватель:	Д. Е. Пивоваров
Группа:	М8О-303Б-21
Дата:	
Оценка:	
Подпись:	

Москва, 2024

4.1 Методы Эйлера, Рунге-Кутты и Адамса

1 Постановка задачи

Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Вариант: 12

$$(x^2+1)y''-2xy'+2y=0,$$
$$y(0)=1,$$
$$y'(0)=1,$$
$$x\in[0,1],\ h=0.1$$

$$y = x - x^2 + 1$$

Рис. 1: Входные данные

2 Результаты работы

```
Метод Эйлера:
x: 0, решение: 1, точное решение: 1, погрешность сравнением с точным решением: 0
x: 0.1, решение: 1.1, точное решение: 1.09, погрешность сравнением с точным решением: 0.01
x: 0.2, решение: 1.18, точное решение: 1.16, погрешность сравнением с точным решением: 0.02
x: 0.3, решение: 1.2398, точное решение: 1.21, погрешность сравнением с точным решением: 0.029802
x: 0.4, решение: 1.27921, точное решение: 1.24, погрешность сравнением с точным решением: 0.0392117
x: 0.5, решение: 1.29804, точное решение: 1.25, погрешность сравнением с точным решением: 0.0480422
x: 0.6, решение: 1.29612, точное решение: 1.24, погрешность сравнением с точным решением: 0.0561159
x: 0.7, решение: 1.27327, точное решение: 1.21, погрешность сравнением с точным решением: 0.0632668
x: 0.8, решение: 1.22934, точное решение: 1.16, погрешность сравнением с точным решением: 0.0693411
x: 0.9, решение: 1.1642, точное решение: 1.09, погрешность сравнением с точным решением: 0.0741973
x: 1, решение: 1.07771, точное решение: 1, погрешность сравнением с точным решением: 0.0777061

Погрешность методом Рунге-Ромберга-Ричардсона для метода Эйлера: 0.000399927

Метод Рунге-Кутты:
x: 0, решение: 1, точное решение: 1, погрешность сравнением с точным решением: 0
x: 0.1, решение: 1.09, точное решение: 1.09, погрешность сравнением с точным решением: 4.14591e-008
x: 0.2, решение: 1.16, точное решение: 1.16, погрешность сравнением с точным решением: 4.17591e-009
x: 0.3, решение: 1.21, точное решение: 1.21, погрешность сравнением с точным решением: 1.40984e-007
x: 0.4, решение: 1.24, точное решение: 1.24, погрешность сравнением с точным решением: 3.72462e-007
x: 0.5, решение: 1.25, точное решение: 1.25, погрешность сравнением с точным решением: 7.0078e-007
x: 0.6, решение: 1.24, точное решение: 1.24, погрешность сравнением с точным решением: 1.12655e-006
x: 0.7, решение: 1.21, точное решение: 1.21, погрешность сравнением с точным решением: 1.64904e-006
x: 0.8, решение: 1.16, точное решение: 1.16, погрешность сравнением с точным решением: 2.26652e-006
x: 0.9, решение: 1.09, точное решение: 1.09, погрешность сравнением с точным решением: 2.97671e-006
x: 1, решение: 1, точное решение: 1, погрешность сравнением с точным решением: 3.77712e-006

Погрешность методом Рунге-Ромберга-Ричардсона для метода Рунге-Кутты: 0.00316643

Метод Адамса:
x: 0, решение: 1, точное решение: 1, погрешность сравнением с точным решением: 0
x: 0.1, решение: 1.09, точное решение: 1.09, погрешность сравнением с точным решением: 4.14591e-008
x: 0.2, решение: 1.16, точное решение: 1.16, погрешность сравнением с точным решением: 4.17591e-009
x: 0.3, решение: 1.21, точное решение: 1.21, погрешность сравнением с точным решением: 1.40984e-007
x: 0.4, решение: 1.24, точное решение: 1.24, погрешность сравнением с точным решением: 3.72462e-007
x: 0.5, решение: 1.25, точное решение: 1.25, погрешность сравнением с точным решением: 9.95616e-007
x: 0.6, решение: 1.24, точное решение: 1.24, погрешность сравнением с точным решением: 1.56541e-006
x: 0.7, решение: 1.20185, точное решение: 1.21, погрешность сравнением с точным решением: 0.00814706
x: 0.8, решение: 1.15213, точное решение: 1.16, погрешность сравнением с точным решением: 0.00786829
x: 0.9, решение: 1.07347, точное решение: 1.09, погрешность сравнением с точным решением: 0.0165342
x: 1, решение: 0.975426, точное решение: 1, погрешность сравнением с точным решением: 0.0245741

Погрешность методом Рунге-Ромберга-Ричардсона для метода Адамса: 0.00319045
```

Рис. 2: Вывод программы в консоли

3 Исходный код

Файл с первым заданием четвертой лабораторной работы:

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>

using namespace std;

double exact_solution(double x){
    return x - x*x + 1;
}

void euler(double h, double x_0, double y1_0, double y2_0, double end, vector<double>&
     x_val, vector<double>& y1_val, vector<double>& y2_val) {
    double x = x_0;
    double y1 = y1_0;
    double y2 = y2_0;
    while (x <= end) {
        x_val.push_back(x);
        y1_val.push_back(y1);
        y2_val.push_back(y2);
        double y1_new = y1 + h * y2;
        double y2_new = y2 + h * ((2 * x * y2 - 2 * y1)/(x * x + 1));
        y1 = y1_new;
        y2 = y2_new;
        x += h;
    }
}

void runge_kutta(double h, double x_0, double y1_0, double y2_0, double end, vector<
    double>& x_val, vector<double>& y1_val, vector<double>& y2_val) {
    double x = x_0;
    double y1 = y1_0;
    double y2 = y2_0;
    while (x <= end) {
        x_val.push_back(x);
        y1_val.push_back(y1);
        y2_val.push_back(y2);
        double k1_1 = h * y2;
        double k1_2 = h * ((2 * x * y2 - 2 * y1)/(x * x + 1));

        double k2_1 = h * (y2 + 0.5 * k1_2);
        double k2_2 = h * ((2 * (x + 0.5 * h) * (y2 + 0.5 * k1_2) - 2 * (y1 + 0.5 *
            k1_1)) / ((x + 0.5 * h) * (x + 0.5 * h) + 1));

        double k3_1 = h * (y2 + 0.5 * k2_2);
```

```
        double k3_2 = h * ((2 * (x + 0.5 * h) * (y2 + 0.5 * k2_2) - 2 * (y1 + 0.5 *
            k2_1)) / ((x + 0.5 * h) * (x + 0.5 * h) + 1));

        double k4_1 = h * (y2 + k3_2);
        double k4_2 = h * ((2 * (x + h) * (y2 + k3_2) - 2 * (y1 + k3_1)) / ((x + h) * (
            x + h) + 1));

        y1 += (k1_1 + 2 * k2_1 + 2 * k3_1 + k4_1) / 6;
        y2 += (k1_2 + 2 * k2_2 + 2 * k3_2 + k4_2) / 6;
        x += h;
    }
}

void adam(double h, double x_0, double y1_0, double y2_0, double end, vector<double>&
    x_val, vector<double>& y1_val, vector<double>& y2_val) {
    double x = x_0;
    double y1 = y1_0;
    double y2 = y2_0;
    vector<double> f_1, f_2;
    for (int i = 0; i < 4; i++) {
        x_val.push_back(x);
        y1_val.push_back(y1);
        y2_val.push_back(y2);

        double k1_1 = h * y2;
        double k1_2 = h * ((2 * x * y2 - 2 * y1) / (x * x + 1));

        double k2_1 = h * (y2 + 0.5 * k1_2);
        double k2_2 = h * ((2 * (x + 0.5 * h) * (y2 + 0.5 * k1_2) - 2 * (y1 + 0.5 *
            k1_1)) / ((x + 0.5 * h) * (x + 0.5 * h) + 1));

        double k3_1 = h * (y2 + 0.5 * k2_2);
        double k3_2 = h * ((2 * (x + 0.5 * h) * (y2 + 0.5 * k2_2) - 2 * (y1 + 0.5 *
            k2_1)) / ((x + 0.5 * h) * (x + 0.5 * h) + 1));

        double k4_1 = h * (y2 + k3_2);
        double k4_2 = h * ((2 * (x + h) * (y2 + k3_2) - 2 * (y1 + k3_1)) / ((x + h) * (
            x + h) + 1));

        y1 += (k1_1 + 2 * k2_1 + 2 * k3_1 + k4_1) / 6;
        y2 += (k1_2 + 2 * k2_2 + 2 * k3_2 + k4_2) / 6;
        x += h;

        f_1.push_back(y2);
        f_2.push_back((2 * x * y2 - 2 * y1) / (x * x + 1));
    }

    while (x <= end) {
        x_val.push_back(x);
```

```
        y1_val.push_back(y1);
        y2_val.push_back(y2);
        double y1_new = y1 + h / 24 * (55 * f_1.back() - 59 * f_1[f_1.size() - 2] + 37 * f_1[f_1.size() - 3] - 9 * f_1[f_1.size() - 4]);
        double y2_new = y2 + h / 24 * (55 * f_2.back() - 59 * f_2[f_2.size() - 2] + 37 * f_2[f_2.size() - 3] - 9 * f_2[f_2.size() - 4]);
        f_1.push_back(y2_new);
        f_2.push_back((2 * x * y2_new - 2 * y1_new) / (x * x + 1));
        y1 = y1_new;
        y2 = y2_new;
        x += h;
    }
}

int main(){
    double x_0 = 0.0;
    double y1_0 = 1;
    double y2_0 = 1;
    double end = 1.0;
    double h = 0.1;
    vector<double> x_val, y1_val, y2_val;
    vector<double> x_val_half, y1_val_half, y2_val_half;
    euler(h, x_0, y1_0, y2_0, end, x_val, y1_val, y2_val);
    ofstream fout("output.txt");
    fout << " :" << endl;
    for (size_t i = 0; i < x_val.size(); i++) {
        double y_exact = exact_solution(x_val[i]);
        fout << "x: " << x_val[i] << ", : " << y1_val[i] << ",  : " << y_exact << ", : " << fabs(y1_val[i] - y_exact) << endl;
    }
    fout << endl;
    euler(h / 2, x_0, y1_0, y2_0, end, x_val_half, y1_val_half, y2_val_half);
    double error = fabs(y1_val_half[y1_val_half.size() - 1] - y1_val[y1_val.size() - 1]) / (pow(2, 4) - 1);
    fout << "  --   : " << error << endl;
    fout << endl;
    x_val.clear();
    y1_val.clear();
    y2_val.clear();
    x_val_half.clear();
    y1_val_half.clear();
    y2_val_half.clear();
    runge_kutta(h, x_0, y1_0, y2_0, end, x_val, y1_val, y2_val);
    fout << " -:" << endl;
    for (size_t i = 0; i < x_val.size(); i++) {
        double y_exact = exact_solution(x_val[i]);
        fout << "x: " << x_val[i] << ", : " << y1_val[i] << ",  : " << y_exact << ", : " << fabs(y1_val[i] - y_exact) << endl;
    }
```

```
    fout << endl;
    runge_kutta(h / 2, x_0, y1_0, y2_0, end, x_val_half, y1_val_half, y2_val_half);
    error = fabs(y1_val_half[y1_val_half.size() - 1] - y1_val[y1_val.size() - 1]) / (pow(2, 4) - 1);
    fout << "  --   -: " << error << endl;
    fout << endl;
    x_val.clear();
    y1_val.clear();
    y2_val.clear();
    x_val_half.clear();
    y1_val_half.clear();
    y2_val_half.clear();
    adam(h, x_0, y1_0, y2_0, end, x_val, y1_val, y2_val);
    fout << " :" << endl;
    for (size_t i = 0; i < x_val.size(); i++) {
        double y_exact = exact_solution(x_val[i]);
        fout << "x: " << x_val[i] << ", : " << y1_val[i] << ",  : " << y_exact << ",  : " << fabs(y1_val[i] - y_exact) << endl;
    }
    fout << endl;
    adam(h / 2, x_0, y1_0, y2_0, end, x_val_half, y1_val_half, y2_val_half);
    error = fabs(y1_val_half[y1_val_half.size() - 1] - y1_val[y1_val.size() - 1]) / (pow(2, 4) - 1);
    fout << "  --   : " << error << endl;
    return 0;
}
```

4.2 Метод стрельбы и конечно-разностный метод

4 Постановка задачи

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

$x(x-1)y''-xy'+y=0$ $y'(1)=3$ $y(3)-3y'(3)=-4$	$y(x)=2+x+2x\ln\lvert x\rvert$

Рис. 3: Входные данные

5 Результаты работы

```
Точное решение:
3.0003 3.63791 4.34249 5.10441 5.91645 6.77303 7.66967 8.60273 9.56915 10.5664 11.5922

Метод стрельбы:
3.0003 3.63169 4.33043 5.08802 5.89747 6.75321 7.6507 8.5862 9.55657 10.5592 11.5917
Метод difference:
3.0003 4.22716 4.84554 5.46875 6.09883 6.73901 7.39475 8.07546 8.79801 9.5938 10.8235

Погрешности методом Рунге-Ромберга-Ричардсона
Метод стрельбы: 7.0564e-012
Метод difference: 0.0316271

Погрешности сравнением с точным решением
Метод стрельбы:
0 0.00622174 0.0120579 0.0163902 0.0189788 0.019817 0.0189709 0.0165285 0.0125815 0.00721792 0.000519726
Метод difference:
0 0.589255 0.503046 0.364348 0.182377 0.0340174 0.274921 0.527265 0.771136 0.972574 0.7687
```

Рис. 4: Вывод программы в консоли

6 Исходный код

Файл со вторым заданием четвертой лабораторной работы:

```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>

using namespace std;

double f(double x, double y, double z){
    return (x*z-y)/(x*x-x);
}

double exact_solution(double x){
    return 2+x+2*x*log(fabs(x));
}

double p(double x){
    return -x;
}

double q(double x){
    return 1;
}

vector<double> runge_kutta(double a, double b, double h, vector<double> x, int n,
    double y0, double z0) {
    vector<double> y(n);
    vector<double> z(n);

    vector<double> K(4);
    vector<double> L(4);

    y[0] = y0;
    z[0] = z0;

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

        K[0] = h * z[i - 1];
        L[0] = h * f(x[i - 1], y[i - 1], z[i-1]);

        for (int j = 1; j < 4; ++j) {
            K[j] = h * (z[i - 1] + L[j - 1] / 2);
            L[j] = h * f(x[i - 1] + h / 2, y[i - 1] + K[j - 1] / 2, z[i - 1] + L[j - 1]
                / 2);
        }

        double dy = (K[0] + 2 * K[1] + 2 * K[2] + K[3]) / 6;
```

```
        double dz = (L[0] + 2 * L[1] + 2 * L[2] + L[3]) / 6;
        y[i] = y[i - 1] + dy;
        z[i] = z[i - 1] + dz;
    }

    return y;
}

vector<double> shooting_method(double a, double b, double h, vector<double> x, int n,
    double eps, double y0, double y1){
    double eta0 = 1;
    double eta = 0.8;

    double F0 = runge_kutta(a, b, h, x, n, y0, eta0)[n-1] - y1;
    double F = runge_kutta(a, b, h, x, n, y0, eta)[n-1] - y1;

    while(abs(F) > eps){
        double c = eta;
        eta = eta - F*(eta - eta0)/(F - F0);
        eta0 = c;
        F0 = F;
        F = runge_kutta(a, b, h, x, n, y0, eta)[n - 1] - y1;
    }
    return runge_kutta(a, b, h, x, n, y0, eta);
}

vector<double> difference_method(double a, double b, double h, vector<double> x, int n
    , double y0, double y1) {
    vector<double> A, B, C, D, P(n), Q(n), res(n);

    A.push_back(0);
    B.push_back(-2 + h * h * q(x[1]));
    C.push_back(1 + p(x[1]) * h / 2);
    D.push_back(-(1 - (p(x[1]) * h) / 2) * y0);
    for (int i = 2; i < n; ++i) {

        A.push_back(1 - p(x[i]) * h / 2);
        B.push_back(-2 + h * h * q(x[i]));
        C.push_back(1 + p(x[i]) * h / 2);
        D.push_back(0);
    }
    A.push_back(1 - p(x[n - 2]) * h / 2);
    B.push_back(-2 + h * h * q(x[n - 2]));
    C.push_back(0);
    D.push_back(-(1 + (p(x[n - 2]) * h) / 2) * y1);

    P[0] = (-C[0] / B[0]);
    Q[0] = (D[0] / B[0]);
    for (int i = 1; i <= n; ++i) {
```

```
        P[i] = (-C[i] / (B[i] + A[i] * P[i - 1]));
        Q[i] = ((D[i] - A[i] * Q[i - 1]) / (B[i] + A[i] * P[i - 1]));
    }

    res[n-1] = Q[n-1];
    for (int i = n - 2; i > 0; --i)
        res[i] = P[i] * res[i + 1] + Q[i];
    res[0] = y0;
    res[n] = y1;
    return res;
}

int main(){
    double a = 1.0001;
    double b = 3;

    double y0 = 2+a+2*a*log(a);
    double y1 = 2+b+2*b*log(b);
    double h = 0.2;
    double eps = 0.0001;

    ofstream fout("output.txt");

    int n = 11;

    vector<double> x(n), y(n);
    for (int i = 0; i < n; ++i){
        x[i] = h * i + a;
        y[i] = exact_solution(x[i]);
    }

    vector<double> shoot = shooting_method(a, b, h, x, n, eps, y0, y1);

    fout << endl << " :" << endl;
    for (int i = 0; i < n; ++i){
        fout << y[i] << " ";
    }
    fout << endl;
    fout << endl << " :" << endl;
    for (int i = 0; i < n; ++i){
        fout << shoot[i] << " ";
    }
    vector<double> diff = difference_method(a, b, h, x, n, y0, y1);

    fout << endl << " difference:" << endl;
    for (int i = 0; i < n; ++i){
        fout << diff[i] << " ";
    }
    fout << endl;
```

```

    fout << endl << "  --" << endl;
    vector<double> shoot_half = shooting_method(a, b, h/2, x, n, eps, y0, y1);
    double error = fabs(shoot_half[shoot_half.size() - 1] - shoot[shoot.size() - 1]) / (pow(2, 4) - 1);
    fout << " : " << error << endl;

    vector<double> diff_half = difference_method(a, b, h/2, x, n, y0, y1);
    error = fabs(diff_half[diff_half.size() - 1] - diff[diff.size() - 1]) / (pow(2, 4) - 1);

    fout << " difference: " << error << endl;
    fout << endl;

    fout << "    " << endl;
    fout << " :" << endl;
    for (int i = 0; i < n; ++i){
        fout << abs(shoot[i] - y[i]) << " ";
    }
    fout << endl << " difference:" << endl;
    for (int i = 0; i < n; ++i){
        fout << abs(diff[i] - y[i]) << " ";
    }
    return 0;
}
```