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:	Я. А. Хайруллина
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: 26

26	$x^4 y'' + 2x^3 y' + y = 0$, $y(1) = 1$, $y'(1) = 1$, $x \in [1,2], h = 0.1$	$y = (\sin 1 + \cos 1)\cos\frac{1}{x} + (\sin 1 - \cos 1)\sin\frac{1}{x}$

Рис. 1: Входные данные

2 Результаты работы

```
X: 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2
Y: 1 1.08665 1.15204 1.20222 1.2413 1.27215 1.29678 1.31664 1.33281 1.34606 1.35701

Euler method: 1 1.1 1.17 1.21976 1.25558 1.28163 1.30068 1.31466 1.32491 1.33238 1.33775
Runge-Kutte method: 1 1.08876 1.15497 1.20522 1.24394 1.27416 1.29802 1.31705 1.33236 1.34476 1.35488
Adams method: 1 1.08876 1.15497 1.20522 1.24806 1.28173 1.3084 1.32985 1.34717 1.36131 1.37291

With Runge-Romberg-Richardson method

Euler method: 1 1.09 1.15656 1.20662 1.24482 1.27434 1.29741 1.3156 1.33006 1.34163 1.35094
Runge-Kutte method: 1 1.08723 1.15281 1.20297 1.2419 1.27253 1.29689 1.31647 1.33235 1.34532 1.35599
Adams method: 1 1.08876 1.15497 1.20522 1.24806 1.28173 1.3084 1.32985 1.34717 1.36131 1.37291

Delta of exact value

Euler method: 0 0.00334546 0.00452163 0.0044013 0.00351564 0.00219133 0.000630817 0.00103998 0.00274286 0.00442975 0.006
07174
Runge-Kutte method: 0 0.000570788 0.000770289 0.000749147 0.000598656 0.00037461 0.00011136 0.000169869 0.000455984 0.00
0738994 0.00101413
Adams method: 0 0.00210235 0.00293314 0.0030072 0.00675603 0.00958042 0.0116214 0.0132061 0.0143666 0.0152505 0.0158987
```

Рис. 2: Вывод программы в консоли

3 Исходный код

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

using namespace std;

double f(double x, double y, double z) {
    return -2 * z / x - y / pow(x, 4);
}

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

vector<double> Euler_method(double a, double b, double h) {
    int s = (b - a) / h + 1;
    vector<double> x(s);
    vector<double> y(s, 1.0);
    vector<double> z(s, 1.0);
    x[0] = a;
    for (int i = 1; i < s; ++i) {
        x[i] = x[i - 1] + h;
        z[i] = z[i - 1] + h * f(x[i - 1], y[i - 1], z[i - 1]);
        y[i] = y[i - 1] + h * z[i - 1];
    }
    return y;
}

vector<vector <double>> RK_method(double a, double b, double h) {
    int s = (b - a) / h + 1;
    vector<double> x(s);
    vector<double> y(s, 1);
    vector<double> z(s, 1);
    vector<double> K(4, 0.0);
    vector<double> L(4, 0.0);
    x[0] = a;
    for (int i = 1; i < s; ++i) {
        x[i] = x[i - 1] + h;
        for (int j = 1; j < K.size(); ++j) {
            K[0] = h * z[i - 1];
            L[0] = h * f(x[i - 1], y[i - 1], z[i - 1]);
            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 deltay = (K[0] + 2 * K[1] + 2 * K[2] + K[3]) / 6;
```

```
        double deltaz = (L[0] + 2 * L[1] + 2 * L[2] + L[3]) / 6;
        y[i] = y[i - 1] + deltay;
        z[i] = z[i - 1] + deltaz;
    }
    return { y, z };
}


vector<double> Adams_method(double a, double b, double h) {
    int s = (b - a) / h + 1;
    vector<double> x;
    vector<double> y(s, 0);
    vector<double> z(s, 0);
    for (double i = a; i < b + h; i += h) {
        x.push_back(i);
    }
    vector<double> y_start = RK_method(a, a + 3 * h, h)[0];
    vector<double> z_start = RK_method(a, a + 3 * h, h)[1];
    for (int i = 0; i < y_start.size(); ++i) {
        y[i] = y_start[i];
        z[i] = z_start[i];
    }
    for (int i = 4; i < s; ++i) {
        z[i] = (z[i - 1] + h * (
            55 * f(x[i - 1], y[i - 1], z[i - 1]) - 59 * f(x[i - 2], y[i - 2], z[i - 2])
            + 37 * f(x[i - 3], y[i - 3], z[i - 3]) - 9 * f(x[i - 4], y[i - 4], z[i -
                4])) / 24);
        y[i] = y[i - 1] + h * z[i - 1];
    }
    return y;
}

vector<vector<double>> RRR_method(double a, double b, double h) {
    vector<double> Euler1, RK1, Adams1;
    vector<double> Euler_norm = Euler_method(a, b, h);
    vector<double> Euler_half = Euler_method(a, b, h / 2);
    vector<double> RK_norm = RK_method(a, b, h)[0];
    vector<double> RK_half = RK_method(a, b, h / 2)[0];
    vector<double> Adams_norm = Adams_method(a, b, h);
    vector<double> Adams_half = Adams_method(a, b, h / 2);
    int s = (b - a) / h + 1;
    Euler1.resize(s);
    RK1.resize(s);
    Adams1.resize(s);
    for (int i = 0; i < s; ++i) {
        Euler1[i] = Euler_norm[i] + (Euler_half[i * 2] - Euler_norm[i]) / (1 - 0.5 *
            0.5);
        RK1[i] = RK_norm[i] + (RK_half[i * 2] - RK_norm[i]) / (1 - 0.5 * 0.5);
        Adams1[i] = Adams_norm[i] + (Adams_norm[i] - Adams_norm[i]) / (1 - 0.5 * 0.5);
```

```
    }
    return { Euler1, RK1, Adams1 };
}


int main() {
    double h = 0.1;
    double a = 1;
    double b = 2;
    vector<double> x, y;
    for (double i = a; i < b + h; i += h) {
        x.push_back(i);
        y.push_back(exact_solution(i));
    }
    std::cout << "X: ";
    for (int i = 0; i < x.size(); ++i) {
        std::cout << x[i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Y: ";
    for (int i = 0; i < y.size(); ++i) {
        std::cout << y[i] << " ";
    }
    std::cout << endl << std::endl;
    std::cout << "Euler method: ";
    for (int i = 0; i < Euler_method(a, b, h).size(); ++i) {
        std::cout << Euler_method(a, b, h)[i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Runge-Kutte method: ";
    for (int i = 0; i < RK_method(a, b, h)[0].size(); ++i) {
        std::cout << RK_method(a, b, h)[0][i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Adams method: ";
    vector <double> res_adams = Adams_method(a, b, h);
    for (int i = 0; i < res_adams.size(); ++i) {
        std::cout << res_adams[i] << " ";
    }
    std::cout << endl << std::endl;
    std::cout << "With Runge-Romberg-Richardson method \n" << std::endl;
    vector<vector<double>> result = RRR_method(a, b, h);
    std::cout << "Euler method: ";
    for (int i = 0; i < result[0].size(); ++i) {
        std::cout << result[0][i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Runge-Kutte method: ";
    for (int i = 0; i < result[1].size(); ++i) {
```

```
        std::cout << result[1][i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Adams method: ";
    for (int i = 0; i < result[2].size(); ++i) {
        std::cout << result[2][i] << " ";
    }
    std::cout << endl << std::endl;
    std::cout << "Delta of exact value \n" << std::endl;
    std::cout << "Euler method: ";
    for (int i = 0; i < result[0].size(); ++i) {
        std::cout << abs(result[0][i] - y[i]) << " ";
    }
    std::cout << std::endl;
    std::cout << "Runge-Kutte method: ";
    for (int i = 0; i < result[1].size(); ++i) {
        std::cout << abs(result[1][i] - y[i]) << " ";
    }
    std::cout << std::endl;
    std::cout << "Adams method: ";
    for (int i = 0; i < result[2].size(); ++i) {
        std::cout << abs(result[2][i] - y[i]) << " ";
    }
    std::cout << std::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: 26

26	$x(x+1)y''+(x+2)y'-y=x+\frac{1}{x}$, $y'(1)=\frac{3}{2}$, $4y'(2)+y(2)=13+4\ln 2$	$y(x)=x+\frac{7}{2}+\frac{1}{x}+\left(\frac{x}{2}+1\right)\ln\lvert x\rvert$

Рис. 3: Входные данные

5 Результаты работы

```
Real: 5 Shooting: 7.14538 Difference: 5
Real: 5.15682 Shooting: 7.2548 Difference: 5.28814
Real: 5.32505 Shooting: 7.37993 Difference: 5.44703
Real: 5.50213 Shooting: 7.51713 Difference: 5.61313
Real: 5.68629 Shooting: 7.66382 Difference: 5.78511
Real: 5.87623 Shooting: 7.81811 Difference: 5.96203
Real: 6.07101 Shooting: 7.97856 Difference: 6.14313
Real: 6.2699 Shooting: 8.14409 Difference: 6.32785
Real: 6.47235 Shooting: 8.31386 Difference: 6.51576
Real: 6.67793 Shooting: 8.48719 Difference: 6.70649
Real: 6.88629 Shooting: 8.66356 Difference: 6.88629
Error shooting: 2.14538Runge shooting: 8.64075e-06Error difference: 0Runge difference : 0
Error shooting: 2.09798Runge shooting: -6.69623e-06Error difference: 0.131316Runge difference : -0.0326808
Error shooting: 2.05488Runge shooting: -1.10775e-05Error difference: 0.121982Runge difference : -0.0277617
Error shooting: 2.015Runge shooting: -1.22756e-05Error difference: 0.110994Runge difference : -0.0229065
Error shooting: 1.97753Runge shooting: -1.24455e-05Error difference: 0.0988256Runge difference : -0.0182013
```

Рис. 4: Вывод программы в консоли

6 Исходный код

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

const double e = 0.00001;

double func(double x) {
    return x + 7 / 2 + 1 / x + (x / 2 + 1) * log(abs(x));
}

double dfunc(double x, double y, double z) {
    return (-(x + 2) * z + y) / (x * (x + 1));
}

double g(double y, double z) {
    return 4 * z + y - 13 - 4 * log(2);
}

double p(double x) {
    return (x + 2) / (x * (x + 1));
}

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

double f(double x) {
    return (x + 1 / x) / (x * (x + 1));
}

double K(double x, double y, double z, double h, int i);

double L(double x, double y, double z, double h, int i) {
    if (i == 0) {
        return h * z;
    }
    else if (i == 3) {
        return h * (z + K(x, y, z, h, i - 1));
    }
    else {
        return h * (z + K(x, y, z, h, i - 1) / 2);
    }
}

double K(double x, double y, double z, double h, int i) {
    if (i == 0) {
        return h * dfunc(x, y, z);
```

```
    }
    else if (i == 3) {
        return h * dfunc(x + h, y + L(x, y, z, h, i - 1), z + K(x, y, z, h, i - 1));
    }
    else {
        return h * dfunc(x + h / 2, y + L(x, y, z, h, i - 1) / 2, z + K(x, y, z, h, i - 1) / 2);
    }
}

double delta_z(double x, double y, double z, double h) {
    double d = 0;
    for (int i = 0; i < 4; ++i) {
        if (i == 0 || i == 3) {
            d += K(x, y, z, h, i);
        }
        else {
            d += 2 * K(x, y, z, h, i);
        }
    }
    return d / 6;
}

double delta_y(double x, double y, double z, double h) {
    double d = 0;
    for (int i = 0; i < 4; ++i) {
        if (i == 0 || i == 3) {
            d += L(x, y, z, h, i);
        }
        else {
            d += 2 * L(x, y, z, h, i);
        }
    }
    return d / 6;
}


std::vector<std::vector<double>> Runge_method(double l, double r, double h, double y0, double z0) {
    double x = l;
    int n = (int)((r - l) / h);
    double z = z0;
    double y = y0;
    std::vector<double> res(n + 1);
    std::vector<double> res_z(n + 1);
    res[0] = y0;
    res_z[0] = z0;
    for (int i = 1; i <= n; ++i) {
        double z1 = z + delta_z(x, y, z, h);
```

```
        y = y + delta_y(x, y, z, h);
        res[i] = y;
        res_z[i] = z1;
        z = z1;
        x += h;
    }
    return { res, res_z };
}

std::vector<double> Shooting(double l, double r, double h, double z0) {
    double n1 = 1, n2 = 0.8, n3;
    double g1, g2, g3;

    std::vector<std::vector<double>> res1 = Runge_method(l, r, h, n1, z0);
    double res1y = res1[0][res1[0].size() - 1];
    double res1z = res1[1][res1[1].size() - 1];
    std::vector<std::vector<double>> res2 = Runge_method(l, r, h, n2, z0);
    double res2y = res2[0][res2[0].size() - 1];
    double res2z = res2[1][res2[1].size() - 1];
    g1 = g(res1y, res1z);
    g2 = g(res2y, res2z);
    std::vector<std::vector<double>> res;
    while (std::abs(g2) > e) {
        n3 = n2 - (n2 - n1) / (g2 - g1) * g2;
        res = Runge_method(l, r, h, n3, z0);
        double resy = res[0][res[0].size() - 1];
        double res_z = res[1][res[1].size() - 1];
        g3 = g(resy, res_z);
        n1 = n2;
        n2 = n3;
        g1 = g2;
        g2 = g3;
    }
    return res[0];
}

std::vector<double> Difference(double l, double r, double h, double y0, double y1) {
    int n = (int)((r - l) / h);

    std::vector<std::vector<double>> A(n, std::vector<double>(n));
    std::vector<double> d(n);
    double x = l + h;

    for (int i = 0; i < n; ++i) {
        A[i][i] = -2 + h * h * q(x);
        if (i > 0) A[i][i - 1] = 1 - p(x) * h / 2;
        if (i < n - 1) A[i][i + 1] = (1 + p(x) * h / 2);
        x += h;
    }
```

```
    d[0] = h * h * f(l + h) - (1 - p(l + h) * h / 2) * y0;
    d[n - 1] = h * h * f(r - h) - (1 + p(r - h) * h / 2) * y1;
    x = l + 2 * h;
    for (int i = 1; i < n - 1; ++i) {
        d[i] = h * h * f(x);
        x += h;
    }

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

    for (int i = 0; i < n; ++i) {
        if (i == 0) {
            P[i] = -A[i][i + 1] / A[i][i];
            Q[i] = d[i] / A[i][i];
        }
        else if (i == n - 1) {
            P[i] = 0;
            Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1])
                ;
        }
        else {
            P[i] = -A[i][i + 1] / (A[i][i] + A[i][i - 1] * P[i - 1]);
            Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1])
                ;
        }
    }

    std::vector<double> y(n + 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];
        }
    }
    y[0] = y0;
    y[n] = y1;


    return y;
}



std::vector<double> RR_method(std::vector<double> Y1, std::vector<double> Y2, int n,
    int p) {
```

```
    std::vector<double> R(n);
    for (int i = 0; i < n; ++i) {
        R[i] = (Y1[i * 2] - Y2[i]) / (std::pow(2, p) - 1);
    }
    return R;
}

std::vector<double> Error(std::vector<double> Yt, std::vector<double> Y, int n) {
    std::vector<double> eps(n);
    for (int i = 0; i < n; ++i) {
        eps[i] = std::abs(Yt[i] - Y[i]);
    }
    return eps;
}

int main() {
    double l = 1, r = 2;
    double z0 = 3 / 2;
    double h = 0.1;
    double y0 = func(l);
    double y1 = func(r);

    std::vector<double> shooting = Shooting(l, r, h, z0);
    std::vector<double> difference = Difference(l, r, h, y0, y1);

    std::vector<double> real(shooting.size());
    double x = l;
    for (int i = 0; i < real.size(); i++) {
        real[i] = func(x);
        x += h;

        std::cout << "Real: " << real[i] << " Shooting: " << shooting[i] << "
            Difference: " << difference[i] << std::endl;
    }

    std::vector<double> shooting2 = Shooting(l, r, h * 2, z0);
    std::vector<double> difference2 = Difference(l, r, h * 2, y0, y1);

    std::vector<double> RShooting = RR_method(shooting, shooting2, real.size() / 2, 2);
    std::vector<double> RDifference = RR_method(difference, difference2, real.size() /
        2, 2);

    std::vector<double> EShooting = Error(shooting, real, real.size() / 2);
    std::vector<double> EDifference = Error(difference, real, real.size() / 2);

    for (int i = 0; i < real.size() / 2; i++) {
        std::cout << "Error shooting: " << EShooting[i] << "Runge shooting: " <<
            RShooting[i] << "Error difference: " << EDifference[i] << "Runge difference
             : " << RDifference[i] << std::endl;
```

```
    }
}
```