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:	И. С. Своеволин
Преподаватель:	Д. Е. Пивоваров
Группа:	М8О-303Б-21
Дата:	
Оценка:	
Подпись:	

Москва, 2024

1 Методы решения начальных и краевых задач для обыкновенных дифференциальных уравнений (ОДУ) и систем ОДУ

1 Постановка задачи

4.1. Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки h. 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: 20

$$x(x-1)y'' + \frac{1}{2}y' - \frac{3}{4}y = 0 \quad y(2) = 2\sqrt{2}, \quad y'(2) = \frac{3}{2}\sqrt{2}, \quad x \in [2,3], h = 0.1 \tag{1}$$

$$y = |x|^{3/2} \tag{2}$$

2 Результаты работы

```
Явный Эйлер Улучшенный  Рунге-Кутта Адамс       Точный ответ
2.82842712  2.82842712  2.82842712  2.82842712  2.82842712
3.04055916  3.04321081  3.04966758  3.04966758  3.04318912
3.25799449  3.26316620  3.27607947  3.27607947  3.26312733
3.48059538  3.48817496  3.50758831  3.50758831  3.48812270
3.70823601  3.71812643  3.74411656  3.73675304  3.71806401
3.94080053  3.95291687  3.98558593  3.97112540  3.95284708
4.17818158  4.19244878  4.23191880  4.20994369  4.19237403
4.42027915  4.43663030  4.48303904  4.45363257  4.43655272
4.66699967  4.68537469  4.73887247  4.70185715  4.68529615
4.91825529  4.93859987  4.99934714  4.95462338  4.93852205

Погрешности методом Рунге-Ромберга
Явный Эйлер Улучшенный  Рунге-Кутта Адамс
-0.00000000 -0.00000000 -0.00000000 -0.00000000
0.00176777  -0.00001762 -0.00353478 -0.00353478
0.00344613  -0.00003157 -0.00705750 -0.00887126
0.00504883  -0.00004246 -0.01059461 -0.01599983
0.00658649  -0.00005073 -0.01416599 -0.01534794
0.00806751  -0.00005675 -0.01778683 -0.01507458
0.00949867  -0.00006082 -0.02146899 -0.01447272
0.01088553  -0.00006317 -0.02522180 -0.01409484
0.01223272  -0.00006400 -0.02905272 -0.01367768
0.01354414  -0.00006349 -0.03296774 -0.01331641
```

Рис. 1: Вывод в консоли

3 Исходный код

`matrix.h`

```
#pragma once
#include <iostream>
#include <vector>
#include <ccomplex>
#include <fstream>

using namespace std;

using cmd = complex <double>;
const double pi = acos(-1);

struct matrix
{
```

```cpp
    int rows = 0, cols = 0;
    vector <vector <double>> v;

    matrix() {}
    matrix(int _rows, int _cols)
    {
        rows = _rows;
        cols = _cols;
        v = vector <vector <double>>(rows, vector <double>(cols));
    }

    vector <double>& operator[](int row)
    {
        return v[row];
    }

    operator double()
    {
        return v[0][0];
    }
};

matrix operator*(matrix lhs, matrix rhs)
{
    if (lhs.cols != rhs.rows)
        return matrix(0, 0);
    matrix res(lhs.rows, rhs.cols);
    for (int i = 0; i < res.rows; i++)
    {
        for (int j = 0; j < res.cols; j++)
        {
            res[i][j] = 0;
            for (int k = 0; k < lhs.cols; k++)
                res[i][j] += lhs[i][k] * rhs[k][j];
        }
    }
    return res;
}

matrix operator*(double lhs, matrix rhs)
{
    for (int i = 0; i < rhs.rows; i++)
    {
        for (int j = 0; j < rhs.cols; j++)
            rhs[i][j] *= lhs;
    }
    return rhs;
}

```

```
matrix operator+(matrix lhs, matrix rhs)
{
    if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
        return matrix(0, 0);
    matrix res(lhs.rows, lhs.cols);
    for (int i = 0; i < rhs.rows; i++)
    {
        for (int j = 0; j < res.cols; j++)
            res[i][j] = lhs[i][j] + rhs[i][j];
    }
    return res;
}

matrix operator-(matrix lhs, matrix rhs)
{
    if (lhs.rows != rhs.rows || rhs.cols != lhs.cols)
        return matrix(0, 0);
    matrix res(lhs.rows, lhs.cols);
    for (int i = 0; i < rhs.rows; i++)
    {
        for (int j = 0; j < res.cols; j++)
            res[i][j] = lhs[i][j] - rhs[i][j];
    }
    return res;
}

ostream& operator<<(ostream& stream, matrix a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream << a[i][j] << ' ';
        stream << '\n';
    }
    return stream;
}

istream& operator>>(istream& stream, matrix& a)
{
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            stream >> a[i][j];
    }
    return stream;
}

matrix transposition(matrix a)
{
```

```
    matrix res(a.cols, a.rows);
    for (int i = 0; i < a.rows; i++)
    {
        for (int j = 0; j < a.cols; j++)
            res[j][i] = a[i][j];
    }
    return res;
}

vector <int> swp;

pair <matrix, matrix> lu_decomposition(matrix a)
{
    int n = a.rows;
    matrix l(n, n);
    swp = vector <int>(0);
    for (int k = 0; k < n; k++)
    {
        matrix prev = a;
        int idx = k;
        for (int i = k + 1; i < n; i++)
        {
            if (abs(prev[idx][k]) < abs(prev[i][k]))
                idx = i;
        }
        swap(prev[k], prev[idx]);
        swap(a[k], a[idx]);
        swap(l[k], l[idx]);
        swp.push_back(idx);
        for (int i = k + 1; i < n; i++)
        {
            double h = prev[i][k] / prev[k][k];
            l[i][k] = h;
            for (int j = k; j < n; j++)
                a[i][j] = prev[i][j] - h * prev[k][j];

        }
    }
    for (int i = 0; i < n; i++)
        l[i][i] = 1;
    return { l, a };
}

matrix solve_triag(matrix a, matrix b, bool up)
{
    int n = a.rows;
    matrix res(n, 1);
    int d = up ? -1 : 1;
    int first = up ? n - 1 : 0;
```

```
    for (int i = first; i < n && i >= 0; i += d)
    {
        res[i][0] = b[i][0];
        for (int j = 0; j < n; j++)
        {
            if (i != j)
                res[i][0] -= a[i][j] * res[j][0];
        }
        res[i][0] = res[i][0] / a[i][i];
    }
    return res;
}

matrix solve_gauss(pair <matrix, matrix> lu, matrix b)
{
    for (int i = 0; i < swp.size(); i++)
        swap(b[i], b[swp[i]]);
    matrix z = solve_triag(lu.first, b, false);
    matrix x = solve_triag(lu.second, z, true);
    //for (int i = 0; i < swp.size(); i++)
        //swap(x[i], x[swp[i]]);
    return x;
}

matrix inverse(matrix a)
{
    int n = a.rows;
    matrix b(n, 1);
    pair <matrix, matrix> lu = lu_decomposition(a);
    matrix res(n, n);
    for (int i = 0; i < n; i++)
    {
        b[max(i - 1, 0)][0] = 0;
        b[i][0] = 1;
        matrix col = solve_gauss(lu, b);
        for (int j = 0; j < n; j++)
            res[j][i] = col[j][0];
    }
    return res;
}

double determinant(matrix a)
{
    int n = a.rows;
    pair <matrix, matrix> lu = lu_decomposition(a);
    double det = 1;
    for (int i = 0; i < n; i++)
        det *= lu.second[i][i];
    return det;
```

```
}

matrix solve_tridiagonal(matrix& a, matrix& b)
{
    int n = a.rows;
    vector <double> p(n), q(n);
    p[0] = -a[0][1] / a[0][0];
    q[0] = b[0][0] / a[0][0];
    for (int i = 1; i < n; i++)
    {
        if (i != n - 1)
            p[i] = -a[i][i + 1] / (a[i][i] + a[i][i - 1] * p[i - 1]);
        else
            p[i] = 0;
        q[i] = (b[i][0] - a[i][i - 1] * q[i - 1]) / (a[i][i] + a[i][i - 1] * p[i - 1]);
    }
    matrix res(n, 1);
    res[n - 1][0] = q[n - 1];
    for (int i = n - 2; i >= 0; i--)
        res[i][0] = p[i] * res[i + 1][0] + q[i];
    return res;
}

double abs(matrix a)
{
    double mx = 0;
    for (int i = 0; i < a.rows; i++)
    {
        double s = 0;
        for (int j = 0; j < a.cols; j++)
            s += abs(a[i][j]);
        mx = max(mx, s);
    }
    return mx;
}

matrix solve_iteration(matrix a, matrix b, double eps)
{
    int n = a.rows;
    matrix alpha(n, n), beta(n, 1);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            alpha[i][j] = -a[i][j] / a[i][i];
        alpha[i][i] = 0;
    }
    for (int i = 0; i < n; i++)
        beta[i][0] = b[i][0] / a[i][i];
    matrix x = beta;
```

```
    double m = abs(a);
    double epsk = 2 * eps;
    while (epsk > eps)
    {
        matrix prev = x;
        x = beta + alpha * x;
        if (m < 1)
            epsk = m / (1 - m) * abs(x - prev);
        else
            epsk = abs(x - prev);
    }
    return x;
}

matrix solve_seidel(matrix a, matrix b, double eps)
{
    int n = a.rows;
    matrix alpha(n, n), beta(n, 1);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            alpha[i][j] = -a[i][j] / a[i][i];
        alpha[i][i] = 0;
    }
    for (int i = 0; i < n; i++)
        beta[i][0] = b[i][0] / a[i][i];
    matrix x = beta;
    double m = abs(alpha);
    double epsk = 2 * eps;
    while (epsk > eps)
    {
        matrix prev = x;
        for (int i = 0; i < n; i++)
        {
            double cur = beta[i][0];
            for (int j = 0; j < n; j++)
                cur += alpha[i][j] * x[j][0];
            x[i][0] = cur;
        }
        if (m < 1)
            epsk = m / (1 - m) * abs(x - prev);
        else
            epsk = abs(x - prev);
    }
    return x;
}

pair <matrix, matrix> method_jacobi(matrix a, double eps)
{
```

```
    int n = a.rows;
    double epsk = 2 * eps;
    matrix vec(n, n);
    for (int i = 0; i < n; i++)
        vec[i][i] = 1;
    while (epsk > eps)
    {
        int cur_i = 1, cur_j = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < i; j++)
            {
                if (abs(a[cur_i][cur_j]) < abs(a[i][j]))
                {
                    cur_i = i;
                    cur_j = j;
                }
            }
        }
        matrix u(n, n);
        double phi = pi / 4;
        if (abs(a[cur_i][cur_i] - a[cur_j][cur_j]) > 1e-7)
            phi = 0.5 * atan((2 * a[cur_i][cur_j]) / (a[cur_i][cur_i] - a[cur_j][cur_j
                ]));
        for (int i = 0; i < n; i++)
            u[i][i] = 1;
        u[cur_i][cur_j] = -sin(phi);
        u[cur_i][cur_i] = cos(phi);
        u[cur_j][cur_i] = sin(phi);
        u[cur_j][cur_j] = cos(phi);
        vec = vec * u;
        a = transposition(u) * a * u;
        epsk = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < i; j++)
                epsk += a[i][j] * a[i][j];
        }
        epsk = sqrt(epsk);
    }
    matrix val(n, 1);
    for (int i = 0; i < n; i++)
        val[i][0] = a[i][i];
    return { val, vec };
}

double sign(double x)
{
    return x > 0 ? 1 : -1;
```

```
}

pair <matrix, matrix> qr_decomposition(matrix a)
{
    int n = a.rows;
    matrix e(n, n);
    for (int i = 0; i < n; i++)
        e[i][i] = 1;
    matrix q = e;
    for (int i = 0; i < n - 1; i++)
    {
        matrix v(n, 1);
        double s = 0;
        for (int j = i; j < n; j++)
            s += a[j][i] * a[j][i];
        v[i][0] = a[i][i] + sign(a[i][i]) * sqrt(s);
        for (int j = i + 1; j < n; j++)
            v[j][0] = a[j][i];
        matrix h = e - (2.0 / double(transposition(v) * v)) * (v * transposition(v));
        q = q * h;
        a = h * a;
    }
    return { q, a };
}

vector <cmd> qr_eigenvalues(matrix a, double eps)
{
    int n = a.rows;
    vector <cmd> prev(n);
    while (true)
    {
        pair <matrix, matrix> p = qr_decomposition(a);
        a = p.second * p.first;
        vector <cmd> cur;
        for (int i = 0; i < n; i++)
        {
            if (i < n - 1 && abs(a[i + 1][i]) > 1e-7)
            {
                double b = -(a[i][i] + a[i + 1][i + 1]);
                double c = a[i][i] * a[i + 1][i + 1] - a[i][i + 1] * a[i + 1][i];
                double d = b * b - 4 * c;
                cmd sgn = (d > 0) ? cmd(1, 0) : cmd(0, 1);
                d = sqrt(abs(d));
                cur.push_back(0.5 * (-b - sgn * d));
                cur.push_back(0.5 * (-b + sgn * d));
                i++;
            }
            else
                cur.push_back(a[i][i]);
```

```
        }
        bool ok = true;
        for (int i = 0; i < n; i++)
            ok = ok && abs(cur[i] - prev[i]) < eps;
        if (ok)
            break;
        prev = cur;
    }
    return prev;
}
```

4-1.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <functional>
#include <algorithm>
#include "matrix.h"

using namespace std;
using equation = function <double(vector <double>)>;
using func = function <double(double)>;

double runge_romberg(double i1, double i2, double h1, double h2, double p)
{
    double k = h2 / h1;
    return (i1 - i2) / (pow(k, p) - 1);
}

vector <equation> make_system(equation dy, int k)
{
    vector <equation> res(k);
    for (int i = 0; i < k - 1; i++)
    {
        auto f = [=](vector <double> x) mutable
        {
            return x[i + 2];
        };
        res[i] = f;
    }
    res[k - 1] = dy;
    return res;
}

vector <vector <double>> explicit_euler(vector <equation> dy, vector <double> yk,
    double a, double b, double h)
{
```

```
    int n = dy.size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
        res[i].push_back(yk[i]);
    for (double x = a; x <= b - h; x += h)
    {
        vector <double> args;
        args.push_back(x);
        for (int i = 0; i < n; i++)
            args.push_back(res[i].back());
        for (int i = 0; i < n; i++)
            res[i].push_back(res[i].back() + h * dy[i](args));
    }
    return res;
}

vector <vector <double>> improved_euler(vector <equation> dy, vector <double> yk, double a, double b, double h)
{
    int n = dy.size();
    vector <vector <double>> tmp(n);
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
    {
        tmp[i].push_back(yk[i]);
        res[i].push_back(yk[i]);
    }
    int k = 0;
    for (double x = a; x <= b - h / 2; x += h / 2)
    {
        vector <double> args;
        args.push_back(x);
        for (int i = 0; i < n; i++)
            args.push_back(tmp[i].back());
        for (int i = 0; i < n; i++)
        {
            tmp[i].push_back(tmp[i][tmp[i].size() - 1 - k % 2] + 0.5 * (k % 2 + 1) * h * dy[i](args));
            if (k % 2)
                res[i].push_back(tmp[i].back());
        }
        k++;
    }
    return res;
}

vector <double> make_args(double x, const vector <double>& y, double add_x, double add_y)
{
```

```
    vector <double> res;
    res.push_back(x + add_x);
    for (int i = 0; i < y.size(); i++)
        res.push_back(y[i] + add_y);
    return res;
}

vector <vector <double>> runge_kutta(vector <equation> dy, vector <double> yk, double l, double r, double h)
{
    // для4 порядкаточности
    int p = 4;
    vector <double> a = { 0, 0, 0.5, 0.5, 1 };
    vector <vector <double>> b = { {}, {0}, {0, 0.5}, {0, 0, 0.5}, {0, 0, 0, 0.5} };
    vector <double> c = { 0, 1. / 6, 1. / 3, 1. / 3, 1. / 6 };
    //
    int n = dy.size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
        res[i].push_back(yk[i]);
    vector <double> K(p + 1);
    for (double x = l; x <= r - h; x += h)
    {
        vector <double> y;
        for (int idx = 0; idx < n; idx++)
            y.push_back(res[idx].back());
        for (int idx = 0; idx < n; idx++)
        {
            K[1] = h * dy[idx](make_args(x, y, 0, 0));
            for (int i = 2; i <= p; i++)
            {
                double add = 0;
                for (int j = 1; j <= i - 1; j++)
                    add += b[i][j] * K[j];
                K[i] = h * dy[idx](make_args(x, y, a[i] * h, add));
            }
            double delta = 0;
            for (int i = 1; i <= p; i++)
                delta += c[i] * K[i];
            res[idx].push_back(res[idx].back() + delta);
        }
    }
    return res;
}

vector <vector <double>> adams(vector <equation> dy, vector <double> yk, double a, double b, double h)
{
    int n = dy.size();
```

```cpp
    vector <vector <double>> y = runge_kutta(dy, yk, a, b, h);
    int m = y[0].size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 4; j++)
            res[i].push_back(y[i][j]);
    }
    for (int k = 4; k < m; k++)
    {
        vector <vector <double>> args(4);
        for (int j = 0; j < 4; j++)
        {
            args[j].push_back(a + h * (k - j - 1));
            for (int i = 0; i < n; i++)
                args[j].push_back(res[i][k - j - 1]);
        }
        for (int i = 0; i < n; i++)
        {
            double delta = 55 * dy[i](args[0]) - 59 * dy[i](args[1]) + 37 * dy[i](args[2]) - 9 * dy[i](args[3]);
            res[i].push_back(res[i].back() + (h / 24) * delta);
        }
    }
    return res;
}

int main()
{
    setlocale(LC_ALL, "Rus");
    ofstream fout("answer4-1.txt");
    fout.precision(8);
    fout << fixed;

    auto ddy = [](vector <double> x)
        {
            return ( -1./2 * x[2] + 3./4 * x[1] ) / ( x[0] * (x[0] - 1) );
        };
    double h1 = 0.1, h2 = 0.05;
    vector <vector <double>> y1 = explicit_euler(make_system(ddy, 2), { 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h1);
    vector <vector <double>> y2 = improved_euler(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h1);
    vector <vector <double>> y3 = runge_kutta(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h1);
    vector <vector <double>> y4 = adams(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h1);
    fout << "Явный Эйлер" << "\t" << "Улучшенный" << "\t" << "РунгеКутта-" << "\t" << "Адамс " << "\t" << "Точный ответ" << endl;
```

```
    for (int i = 0; i < y1[0].size(); i++)
    {
        double x = 2 + h1 * i;
        fout << y1[0][i] << '\t' << y2[0][i] << '\t' << y3[0][i] << '\t' << y4[0][i] << '\t' << pow(abs(x), 3./2) << endl;
    }
    vector <vector <double>> y12 = explicit_euler(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h2);
    vector <vector <double>> y22 = improved_euler(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h2);
    vector <vector <double>> y32 = runge_kutta(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h2);
    vector <vector <double>> y42 = adams(make_system(ddy, 2),{ 2 * sqrt(2), 3. / 2 * sqrt(2) }, 2, 3, h2);
    fout << "\Погрешностиn методомРунгеРомберга-\n";
    fout << "Явный Эйлер" << "\t" << "Улучшенный" << "\t" << "РунгеКутта-" << "\t" << "Адамс " << endl;
    for (int i = 0; i < y1[0].size(); i++)
    {
        fout << runge_romberg(y1[0][i], y12[0][2 * i], h1, h2, 2) << '\t'
            << runge_romberg(y2[0][i], y22[0][2 * i], h1, h2, 4) << '\t'
            << runge_romberg(y3[0][i], y32[0][2 * i], h1, h2, 4) << '\t'
            << runge_romberg(y4[0][i], y42[0][2 * i], h1, h2, 4) << endl;
    }
}
```

4 Постановка задачи

4.2. 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: 20 (Сделан 21, поскольку 20 вариант прописан с ошибкой в файле с ТЗ)

$$x(2x+1)y'' + 2(x+1)y' - 2y = 0, \qquad y'(1) = 0, \qquad y(3) - y'(3) = \frac{31}{9} \tag{3}$$

$$y(x) = x + 1 + \frac{1}{x} \tag{4}$$

5 Результаты работы

```
Стрельба    Разности    Точное
3.05909138  3.05755377  3.00000000
3.05909138  3.05755377  3.00909091
3.07819541  3.07394816  3.03333333
3.11159727  3.10296242  3.06923077
3.15598466  3.14189780  3.11428571
3.20899988  3.18877364  3.16666667
3.26891863  3.24210348  3.22500000
3.33445037  3.30075003  3.28823529
3.40460984  3.36382830  3.35555556
3.47863168  3.43063937  3.42631579
3.55591222  3.50062390  3.50000000
3.63596885  3.57332900  3.57619048
3.71841103  3.64838400  3.65454545
3.80291934  3.72548265  3.73478261
3.88922989  3.80436968  3.81666667
3.97712278  3.88483063  3.90000000
4.06641323  3.96668403  3.98461538
4.15694483  4.04977525  4.07037037
4.24858426  4.13397173  4.15714286
4.34121719  4.21915921  4.24482759

Погрешности методом Рунге-Ромберга
Стрельба    Разности
-0.07136471 -0.03817839
-0.06622140 -0.03216416
-0.06357545 -0.02695730
-0.06260307 -0.02238041
-0.06279289 -0.01830404
-0.06381482 -0.01463170
-0.06544857 -0.01129005
-0.06754294 -0.00822237
-0.06999159 -0.00538417
-0.07271807 -0.00274004
-0.07566640 -0.00026147
-0.07879487 0.00207475
-0.08207193 0.00428769
-0.08547337 0.00639318
-0.08898038 0.00840446
-0.09257818 0.01033267
-0.09625501 0.01218729
-0.10000144 0.01397639
-0.10380980 0.01570692
-0.10767384 0.01738487
```

Рис. 2: Вывод в консоли

6 Исходный код

4-2.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <functional>
#include <algorithm>
#include "matrix.h"

using namespace std;
using equation = function <double(vector <double>)>;
using func = function <double(double)>;

double runge_romberg(double i1, double i2, double h1, double h2, double p)
{
    double k = h2 / h1;
    return (i1 - i2) / (pow(k, p) - 1);
}

vector <equation> make_system(equation dy, int k)
{
    vector <equation> res(k);
    for (int i = 0; i < k - 1; i++)
    {
        auto f = [=](vector <double> x) mutable
        {
            return x[i + 2];
        };
        res[i] = f;
    }
    res[k - 1] = dy;
    return res;
}

vector <vector <double>> explicit_euler(vector <equation> dy, vector <double> yk,
    double a, double b, double h)
{
    int n = dy.size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
        res[i].push_back(yk[i]);
    for (double x = a; x <= b - h; x += h)
    {
        vector <double> args;
        args.push_back(x);
        for (int i = 0; i < n; i++)
```

```
            args.push_back(res[i].back());
        for (int i = 0; i < n; i++)
            res[i].push_back(res[i].back() + h * dy[i](args));
    }
    return res;
}

vector <vector <double>> improved_euler(vector <equation> dy, vector <double> yk, double a, double b, double h)
{
    int n = dy.size();
    vector <vector <double>> tmp(n);
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
    {
        tmp[i].push_back(yk[i]);
        res[i].push_back(yk[i]);
    }
    int k = 0;
    for (double x = a; x <= b - h / 2; x += h / 2)
    {
        vector <double> args;
        args.push_back(x);
        for (int i = 0; i < n; i++)
            args.push_back(tmp[i].back());
        for (int i = 0; i < n; i++)
        {
            tmp[i].push_back(tmp[i][tmp[i].size() - 1 - k % 2] + 0.5 * (k % 2 + 1) * h * dy[i](args));
            if (k % 2)
                res[i].push_back(tmp[i].back());
        }
        k++;
    }
    return res;
}

vector <double> make_args(double x, const vector <double>& y, double add_x, double add_y)
{
    vector <double> res;
    res.push_back(x + add_x);
    for (int i = 0; i < y.size(); i++)
        res.push_back(y[i] + add_y);
    return res;
}

vector <vector <double>> runge_kutta(vector <equation> dy, vector <double> yk, double l, double r, double h)
```

```
{
    // для4 порядкаточности
    int p = 4;
    vector <double> a = { 0, 0, 0.5, 0.5, 1 };
    vector <vector <double>> b = { {}, {0}, {0, 0.5}, {0, 0, 0.5}, {0, 0, 0, 0.5} };
    vector <double> c = { 0, 1. / 6, 1. / 3, 1. / 3, 1. / 6 };
    //
    int n = dy.size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
        res[i].push_back(yk[i]);
    vector <double> K(p + 1);
    for (double x = l; x <= r - h; x += h)
    {
        vector <double> y;
        for (int idx = 0; idx < n; idx++)
            y.push_back(res[idx].back());
        for (int idx = 0; idx < n; idx++)
        {
            K[1] = h * dy[idx](make_args(x, y, 0, 0));
            for (int i = 2; i <= p; i++)
            {
                double add = 0;
                for (int j = 1; j <= i - 1; j++)
                    add += b[i][j] * K[j];
                K[i] = h * dy[idx](make_args(x, y, a[i] * h, add));
            }
            double delta = 0;
            for (int i = 1; i <= p; i++)
                delta += c[i] * K[i];
            res[idx].push_back(res[idx].back() + delta);
        }
    }
    return res;
}

vector <vector <double>> adams(vector <equation> dy, vector <double> yk, double a, double b, double h)
{
    int n = dy.size();
    vector <vector <double>> y = runge_kutta(dy, yk, a, b, h);
    int m = y[0].size();
    vector <vector <double>> res(n);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 4; j++)
            res[i].push_back(y[i][j]);
    }
    for (int k = 4; k < m; k++)
```

```
    {
        vector <vector <double>> args(4);
        for (int j = 0; j < 4; j++)
        {
            args[j].push_back(a + h * (k - j - 1));
            for (int i = 0; i < n; i++)
                args[j].push_back(res[i][k - j - 1]);
        }
        for (int i = 0; i < n; i++)
        {
            double delta = 55 * dy[i](args[0]) - 59 * dy[i](args[1]) + 37 * dy[i](args
                [2]) - 9 * dy[i](args[3]);
            res[i].push_back(res[i].back() + (h / 24) * delta);
        }
    }
    return res;
}

vector <double> shooting(equation ddy, double a, double b, vector <double> alpha,
    vector <double> beta, double ya, double yb, double h)
{
    double eps = 0.00000001;
    vector <equation> v = make_system(ddy, 2);
    auto phi = [&](double n)
    {
        vector <double> args;
        if (abs(beta[0]) > eps)
            args = { n, (ya - alpha[0] * n) / beta[0] };
        else
            args = { ya / alpha[0], n };
        vector <vector <double>> yk = runge_kutta(v, args, a, b, h);
        return alpha[1] * yk[0].back() + beta[1] * yk[1].back() - yb;
    };
    double n0 = 10, n1 = -1;
    double phi0 = phi(n0);
    double phi1 = phi(n1);
    double n;
    while (true)
    {
        n = n1 - ((n1 - n0) / (phi1 - phi0)) * phi1;
        double phij = phi(n);
        if (abs(phij) < eps)
            break;
        n0 = n1;
        n1 = n;
        phi0 = phi1;
        phi1 = phij;
    }
    vector <double> args;
```

```
    if (abs(beta[0]) > eps)
        args = { n, (ya - alpha[0] * n) / beta[0] };
    else
        args = { ya / alpha[0], n };
    vector <vector <double>> res = runge_kutta(v, args, a, b, h);
    return res[0];
}

vector <double> finite_difference(func f, func p, func q, double a, double b, vector <
    double> alpha, vector <double> beta, double ya, double yb, double h)
{
    vector <double> x;
    for (int i = 0; a + i * h < b; i++)
        x.push_back(a + i * h);
    int n = x.size();
    matrix A(n + 1, n + 1);
    matrix B(n + 1, 1);
    A[0][0] = alpha[0] * h - beta[0];
    A[0][1] = beta[0];
    B[0][0] = h * ya;
    for (int i = 1; i <= n - 1; i++)
    {
        A[i][i + 1] = 1 + p(x[i]) * h / 2;
        A[i][i] = -2 + h * h * q(x[i]);
        A[i][i - 1] = 1 - p(x[i]) * h / 2;
        B[i][0] = h * h * f(x[i]);
    }
    A[n][n - 1] = -beta[1];
    A[n][n] = alpha[1] * h + beta[1];
    B[n][0] = h * yb;
    matrix sol = solve_tridiagonal(A, B);
    vector <double> res;
    for (int i = 0; i < n + 1; i++)
        res.push_back(sol[i][0]);
    return res;
}

int main()
{
    setlocale(LC_ALL, "Rus");
    ofstream fout("answer4-2.txt");
    fout.precision(8);
    fout << fixed;

    auto ddy = [](vector <double> x)
    {
        return (2 * x[1] - 2 * (x[0] + 1) * x[2]) / (x[0] * (2 * x[0] + 1));
    };
    double h1 = 0.1;
```

```
    vector <double> y1 = shooting(ddy, 1, 3, { 0, 1 }, { 1, -1 }, 0, 31. / 9, h1);

    auto f = [](double x) { return 0; };
    auto p = [](double x) { return 2 * (x + 1) / (x * (2 * x + 1)); };
    auto q = [](double x) { return -2 / (x * (2 * x + 1)); };
    vector <double> y2 = finite_difference(f, p, q, 1, 3, { 0, 1 }, { 1, -1 }, 0, 31. / 9, h1);
    fout << "Стрельба" << "\t" << "Разности" << '\t' << "Точное" << endl;
    for (int i = 0; i < y1.size(); i++)
    {
        double x = 1 + i * h1;
        fout << y1[i] << '\t' << y2[i] << '\t' << (x + 1 + 1 / x) << endl;
    }
    double h2 = 0.05;
    vector <double> y12 = shooting(ddy, 1, 3, { 0, 1 }, { 1, -1 }, 0, 31. / 9, h2);
    vector <double> y22 = finite_difference(f, p, q, 1, 3, { 0, 1 }, { 1, -1 }, 0, 31. / 9, h2);
    fout << "\Погрешностиn методомРунгеРомберга-\n";
    fout << "Стрельба" << "\t" << "Разности" << endl;
    for (int i = 0; i < y1.size(); i++)
    {
        fout << runge_romberg(y1[i], y12[2 * i], h1, h2, 4) << '\t'
            << runge_romberg(y2[i], y22[2 * i], h1, h2, 2) << endl;
    }
}
```