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:	Наумов Г.К.
Преподаватель:	Пивоваров Д.Е.
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: 16

16	$(x^2-1)y''-2xy'+2y=0$, $y(2)=7$, $y'(2)=5$, $x\in[2,3], h=0.1$	$y=x^2+x+1$

Рис. 1: Входные данные

2 Результаты работы

```
Euler method:
x: 2, my solution: 7, exact solution: 10.3891, error: 3.38906
x: 2.1, my solution: 7.5, exact solution: 11.2662, error: 3.76617
x: 2.2, my solution: 8.02, exact solution: 12.225, error: 4.20501
x: 2.3, my solution: 8.56006, exact solution: 13.2742, error: 4.71412
x: 2.4, my solution: 9.12023, exact solution: 14.4232, error: 5.30295
x: 2.5, my solution: 9.70056, exact solution: 15.6825, error: 5.98194
x: 2.6, my solution: 10.3011, exact solution: 17.0637, error: 6.76266
x: 2.7, my solution: 10.9218, exact solution: 18.5797, error: 7.65788
x: 2.8, my solution: 11.5629, exact solution: 20.2446, error: 8.68176
x: 2.9, my solution: 12.2242, exact solution: 22.0741, error: 9.84991

Euler method error:
Exact solution in x = 3: 24.0855
My solution in x = 3: 12.2242
Error: 0.0486225

Runge-Kutta method:
x: 2, my solution: 7, exact solution: 10.3891, error: 3.38906
x: 2.1, my solution: 7.51, exact solution: 11.2662, error: 3.75617
x: 2.2, my solution: 8.04, exact solution: 12.225, error: 4.18501
x: 2.3, my solution: 8.59, exact solution: 13.2742, error: 4.68418
```

```
x: 2.4, my solution: 9.16, exact solution: 14.4232, error: 5.26318
x: 2.5, my solution: 9.75, exact solution: 15.6825, error: 5.93249
x: 2.6, my solution: 10.36, exact solution: 17.0637, error: 6.70374
x: 2.7, my solution: 10.99, exact solution: 18.5797, error: 7.58973
x: 2.8, my solution: 11.64, exact solution: 20.2446, error: 8.60465
x: 2.9, my solution: 12.31, exact solution: 22.0741, error: 9.76414

Runge-Kutta method error:
Exact solution in x = 3: 24.0855
My solution in x = 3: 12.31
Error: 0.0459999

Adams method:
x: 2, my solution: 7, exact solution: 10.3891, error: 3.38906
x: 2.1, my solution: 7.51, exact solution: 11.2662, error: 3.75617
x: 2.2, my solution: 8.04, exact solution: 12.225, error: 4.18501
x: 2.3, my solution: 8.59, exact solution: 13.2742, error: 4.68418
x: 2.4, my solution: 9.16, exact solution: 14.4232, error: 5.26318
x: 2.5, my solution: 9.75, exact solution: 15.6825, error: 5.93249
x: 2.6, my solution: 10.36, exact solution: 17.0637, error: 6.70374
x: 2.7, my solution: 10.9876, exact solution: 18.5797, error: 7.59216
x: 2.8, my solution: 11.6376, exact solution: 20.2446, error: 8.60702
x: 2.9, my solution: 12.305, exact solution: 22.0741, error: 9.76916

Adams method error:
Exact solution in x = 3: 24.0855
My solution in x = 3: 12.305
Error: 0.0457717
```

3 Исходный код

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


void eulerMethod(double h, double x0, double y10, double y20, double x_end, std::vector<double>& x_vals, std::vector<double>& y1_vals, std::vector<double>& y2_vals) {
    double x = x0;
    double y1 = y10;
    double y2 = y20;

    while (x <= x_end) {
        x_vals.push_back(x);
        y1_vals.push_back(y1);
        y2_vals.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 rungeKuttaMethod(double h, double x0, double y10, double y20, double x_end, std::vector<double>& x_vals, std::vector<double>& y1_vals, std::vector<double>& y2_vals) {
    double x = x0;
    double y1 = y10;
    double y2 = y20;

    while (x <= x_end) {
        x_vals.push_back(x);
        y1_vals.push_back(y1);
        y2_vals.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 adamsMethod(double h, double x0, double y10, double y20, double x_end, std::vector<double>& x_vals, std::vector<double>& y1_vals, std::vector<double>& y2_vals) {
    double x = x0;
    double y1 = y10;
    double y2 = y20;

    std::vector<double> f1, f2;
```

```cpp

    //   -
    for (int i = 0; i < 4; i++) {
        x_vals.push_back(x);
        y1_vals.push_back(y1);
        y2_vals.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;

        f1.push_back(y2);
        f2.push_back((2 * x * y2 - 2 * y1) / (x * x - 1));
    }

    while (x <= x_end) {
        x_vals.push_back(x);
        y1_vals.push_back(y1);
        y2_vals.push_back(y2);

        double y1_new = y1 + h / 24 * (55 * f1.back() - 59 * f1[f1.size() - 2] + 37 * f1[f1.size() - 3] - 9 * f1[f1.size() - 4]);
        double y2_new = y2 + h / 24 * (55 * f2.back() - 59 * f2[f2.size() - 2] + 37 * f2[f2.size() - 3] - 9 * f2[f2.size() - 4]);

        f1.push_back(y2_new);
        f2.push_back((2 * x * y2_new - 2 * y1_new) / (x * x - 1));

        y1 = y1_new;
        y2 = y2_new;
        x += h;
    }
}

```

```
double exactSolution(double x) {
    return x + 1 + exp(x);
}

int main() {
    double h = 0.1;
    double x0 = 2.0;
    double y10 = 7.0;
    double y20 = 5.0;
    double x_end = 3.0;
    std::vector<double> x_vals, y1_vals, y2_vals;
    std::vector<double> x_vals_half, y1_vals_half, y2_vals_half;

    std::cout << "Euler method: " << std::endl;
    eulerMethod(h, x0, y10, y20, x_end, x_vals, y1_vals, y2_vals);
    for (size_t i = 0; i < x_vals.size(); i++) {
        double y_exact = exactSolution(x_vals[i]);
        std::cout << "x: " << x_vals[i] << ", my solution: " << y1_vals[i] << ", exact solution: " << y_exact << ", error: " << fabs(y1_vals[i] - y_exact) << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Euler method error: " << std::endl;
    eulerMethod(h / 2, x0, y10, y20, x_end, x_vals_half, y1_vals_half, y2_vals_half);
    double error = fabs(y1_vals_half[y1_vals_half.size() - 1] - y1_vals[y1_vals.size() - 1]) / (pow(2, 4) - 1);
    std::cout << "Exact solution in x = " << x_end << ": " << exactSolution(x_end) << std::endl;
    std::cout << "My solution in x = " << x_end << ": " << y1_vals[y1_vals.size() - 1] << std::endl;
    std::cout << "Error: " << error << std::endl;
    std::cout << std::endl;
    x_vals.clear();
    y1_vals.clear();
    y2_vals.clear();
    x_vals_half.clear();
    y1_vals_half.clear();
    y2_vals_half.clear();

    std::cout << "Runge-Kutta method: " << std::endl;
    rungeKuttaMethod(h, x0, y10, y20, x_end, x_vals, y1_vals, y2_vals);
    for (size_t i = 0; i < x_vals.size(); i++) {
        double y_exact = exactSolution(x_vals[i]);
        std::cout << "x: " << x_vals[i] << ", my solution: " << y1_vals[i] << ", exact solution: " << y_exact << ", error: " << fabs(y1_vals[i] - y_exact) << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Runge-Kutta method error: " << std::endl;
```

```
    rungeKuttaMethod(h / 2, x0, y10, y20, x_end, x_vals_half, y1_vals_half, y2_vals_half);
    error = fabs(y1_vals_half[y1_vals_half.size() - 1] - y1_vals[y1_vals.size() - 1]) / (pow(2, 4) - 1);
    std::cout << "Exact solution in x = " << x_end << ": " << exactSolution(x_end) << std::endl;
    std::cout << "My solution in x = " << x_end << ": " << y1_vals[y1_vals.size() - 1] << std::endl;
    std::cout << "Error: " << error << std::endl;
    std::cout << std::endl;
    x_vals.clear();
    y1_vals.clear();
    y2_vals.clear();
    x_vals_half.clear();
    y1_vals_half.clear();
    y2_vals_half.clear();

    std::cout << "Adams method: " << std::endl;
    adamsMethod(h, x0, y10, y20, x_end, x_vals, y1_vals, y2_vals);
    for (size_t i = 0; i < x_vals.size(); i++) {
        double y_exact = exactSolution(x_vals[i]);
        std::cout << "x: " << x_vals[i] << ", my solution: " << y1_vals[i] << ", exact solution: " << y_exact << ", error: " << fabs(y1_vals[i] - y_exact) << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Adams method error: " << std::endl;
    adamsMethod(h / 2, x0, y10, y20, x_end, x_vals_half, y1_vals_half, y2_vals_half);
    error = fabs(y1_vals_half[y1_vals_half.size() - 1] - y1_vals[y1_vals.size() - 1]) / (pow(2, 4) - 1);
    std::cout << "Exact solution in x = " << x_end << ": " << exactSolution(x_end) << std::endl;
    std::cout << "My solution in x = " << x_end << ": " << y1_vals[y1_vals.size() - 1] << std::endl;
    std::cout << "Error: " << error << std::endl;
    std::cout << std::endl;
    x_vals.clear();
    y1_vals.clear();
    y2_vals.clear();
    x_vals_half.clear();
    y1_vals_half.clear();
    y2_vals_half.clear();

    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: 16

16	$y'' - \operatorname{tg}x\, y' + 2y = 0$, $y(0) = 2$, $y(\frac{\pi}{6}) = 2.5 - 0.5 \cdot \ln 3$	$y(x) = \sin x + 2 - \sin x \cdot \ln\left(\frac{1 + \sin x}{1 - \sin x}\right)$

Рис. 2: Входные данные

5 Результаты работы

```
Shooting method:
x: 0, my solution: 2,| exact solution: 2,| error: 0
x: 0.05, my solution: 2.04206,| exact solution: 2.04498,| error: 0.0029172
x: 0.1, my solution: 2.07401,| exact solution: 2.07983,| error: 0.0058271
x: 0.15, my solution: 2.09572,| exact solution: 2.10444,| error: 0.00872244
x: 0.2, my solution: 2.10707,| exact solution: 2.11867,| error: 0.011596
x: 0.25, my solution: 2.10795,| exact solution: 2.12239,| error: 0.0144405
x: 0.3, my solution: 2.09824,| exact solution: 2.11549,| error: 0.017249
x: 0.35, my solution: 2.0778,| exact solution: 2.09781,| error: 0.0200143
x: 0.4, my solution: 2.0465,| exact solution: 2.06923,| error: 0.0227296
x: 0.45, my solution: 2.00419,| exact solution: 2.02957,| error: 0.0253881
x: 0.5, my solution: 1.95069,| exact solution: 1.97868,| error: 0.0279831

Shooting method error:
Exact solution in x = 0.523599: 1.95069
My solution in x = 0.523599: 1.95069
Error: 2.66454e-16

Finite Difference method:
x: 0, my solution: 2,| exact solution: 2,| error: 0
x: 0.05, my solution: 2.04154,| exact solution: 2.04498,| error: 0.00344151
x: 0.1, my solution: 2.07278,| exact solution: 2.07983,| error: 0.0070568
```

```
x: 0.15, my solution: 2.09352,| exact solution: 2.10444,| error: 0.0109164
x: 0.2, my solution: 2.10368,| exact solution: 2.11867,| error: 0.014985
x: 0.25, my solution: 2.10327,| exact solution: 2.12239,| error: 0.019119
x: 0.3, my solution: 2.09242,| exact solution: 2.11549,| error: 0.0230651
x: 0.35, my solution: 2.07135,| exact solution: 2.09781,| error: 0.0264586
x: 0.4, my solution: 2.04041,| exact solution: 2.06923,| error: 0.0288224
x: 0.45, my solution: 2.00001,| exact solution: 2.02957,| error: 0.0295658
x: 0.5, my solution: 1.95069,| exact solution: 1.97868,| error: 0.0279831

Finite Difference method error:
Exact solution in x = 0.523599: 1.95069
My solution in x = 0.523599: 1.95069
Error: 0
```

6 Исходный код

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


void rungeKutta(double h, double x0, double y0, double dy0, double x_end, std::vector<double>& x_vals, std::vector<double>& y_vals) {
    double x = x0;
    double y = y0;
    double dy = dy0;

    while (x <= x_end) {
        x_vals.push_back(x);
        y_vals.push_back(y);

        double k1 = h * dy;
        double l1 = h * (tan(x) * dy - 2 * y);
        double k2 = h * (dy + 0.5 * l1);
        double l2 = h * (tan(x + 0.5 * h) * (dy + 0.5 * l1) - 2 * (y + 0.5 * k1));
        double k3 = h * (dy + 0.5 * l2);
        double l3 = h * (tan(x + 0.5 * h) * (dy + 0.5 * l2) - 2 * (y + 0.5 * k2));
        double k4 = h * (dy + l3);
        double l4 = h * (tan(x + h) * (dy + l3) - 2 * (y + k3));

        y += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        dy += (l1 + 2 * l2 + 2 * l3 + l4) / 6;
        x += h;
    }
}

double shootingMethod(double h, double x0, double y0, double x_end, double y_end, double initial_guess) {
    double tolerance = 1e-6;
```

```cpp
    double guess1 = initial_guess;
    double guess2 = initial_guess + 0.1;

    double f1, f2;

    while (true) {
        std::vector<double> x_vals, y_vals1, y_vals2;
        rungeKutta(h, x0, y0, guess1, x_end, x_vals, y_vals1);
        rungeKutta(h, x0, y0, guess2, x_end, x_vals, y_vals2);

        f1 = y_vals1.back() - y_end;
        f2 = y_vals2.back() - y_end;

        if (fabs(f1) < tolerance) {
            return guess1;
        }
        if (fabs(f2) < tolerance) {
            return guess2;
        }

        double guess_new = guess1 - f1 * (guess2 - guess1) / (f2 - f1);
        guess1 = guess2;
        guess2 = guess_new;
    }
}

void finiteDifferenceMethod(double h, double x0, double y0, double x_end, double y_end
    , std::vector<double>& x_vals, std::vector<double>& y_vals) {
    int n = (x_end - x0) / h;
    std::vector<double> a(n + 1), b(n + 1), c(n + 1), d(n + 1), y(n + 1);

    for (int i = 0; i <= n; ++i) {
        x_vals.push_back(x0 + i * h);
    }

    a[0] = 0;
    b[0] = 1;
    c[0] = 0;
    d[0] = y0;

    for (int i = 1; i < n; ++i) {
        double x = x0 + i * h;
        a[i] = 1 / (h * h) - tan(x) / (2 * h);
        b[i] = -2 / (h * h) + 2;
        c[i] = 1 / (h * h) + tan(x) / (2 * h);
        d[i] = 0;
    }

    a[n] = 0;
```

```
    b[n] = 1;
    c[n] = 0;
    d[n] = y_end;

    for (int i = 1; i <= n; ++i) {
        double m = a[i] / b[i - 1];
        b[i] -= m * c[i - 1];
        d[i] -= m * d[i - 1];
    }

    y[n] = d[n] / b[n];
    for (int i = n - 1; i >= 0; --i) {
        y[i] = (d[i] - c[i] * y[i + 1]) / b[i];
    }

    for (int i = 0; i <= n; ++i) {
        y_vals.push_back(y[i]);
    }
}

double exactSolution(double x) {
    return sin(x) + 2 - sin(x) * log((1 + sin(x)) / (1 - sin(x)));
}

int main() {
    double h = 0.05;
    double x0 = 0.0;
    double y0 = 2.0;
    double x_end = M_PI / 6;
    double y_end = 2.5 - 0.5 * log(3.0);
    std::vector<double> x_vals, y_vals;
    std::vector<double> x_vals_half, y_vals_half;

    std::cout << "Shooting method: " << std::endl;
    double initial_guess = 0.0;
    double dy0 = shootingMethod(h, x0, y0, x_end, y_end, initial_guess);
    rungeKutta(h, x0, y0, dy0, x_end, x_vals, y_vals);
    for (size_t i = 0; i < x_vals.size(); i++) {
        double y_exact = exactSolution(x_vals[i]);
        std::cout << "x: " << x_vals[i] << ", my solution: " << y_vals[i] << ",| exact solution: " << y_exact << ",| error: " << fabs(y_vals[i] - y_exact) << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Shooting method error: " << std::endl;
    dy0 = shootingMethod(h/2, x0, y0, x_end, y_end, initial_guess);
    rungeKutta(h/2, x0, y0, dy0, x_end, x_vals_half, y_vals_half);
    double error = fabs(y_vals_half[y_vals_half.size() - 1] - y_vals[y_vals.size() - 1]) / (pow(2, 4) - 1);
```

```cpp
    std::cout << "Exact solution in x = " << x_end << ": " << exactSolution(x_end) << std::endl;
    std::cout << "My solution in x = " << x_end << ": " << y_vals[y_vals.size() - 1] << std::endl;
    std::cout << "Error: " << error << std::endl;
    std::cout << std::endl;
    x_vals.clear();
    y_vals.clear();
    x_vals_half.clear();
    y_vals_half.clear();


    std::cout << "Finite Difference method: " << std::endl;
    finiteDifferenceMethod(h, x0, y0, x_end, y_end, x_vals, y_vals);
    for (size_t i = 0; i < x_vals.size(); i++) {
        double y_exact = exactSolution(x_vals[i]);
        std::cout << "x: " << x_vals[i] << ", my solution: " << y_vals[i] << ",| exact solution: " << y_exact << ",| error: " << fabs(y_vals[i] - y_exact) << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Finite Difference method error: " << std::endl;
    finiteDifferenceMethod(h/2, x0, y0, x_end, y_end, x_vals_half, y_vals_half);
    error = fabs(y_vals_half[y_vals_half.size() - 1] - y_vals[y_vals.size() - 1]) / (pow(2, 4) - 1);
    std::cout << "Exact solution in x = " << x_end << ": " << exactSolution(x_end) << std::endl;
    std::cout << "My solution in x = " << x_end << ": " << y_vals[y_vals.size() - 1] << std::endl;
    std::cout << "Error: " << error << std::endl;
    std::cout << std::endl;
    x_vals.clear();
    y_vals.clear();
    x_vals_half.clear();
    y_vals_half.clear();
    return 0;
}
```