

**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: Т. В. Мохнач
Преподаватель: Д. Е. Пивоваров
Группа: М8О-303Б-21
Дата:
Оценка:
Подпись:

Москва, 2024

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: 15

$$xy'' + y' = 0,$$

$$y(1) = 1,$$

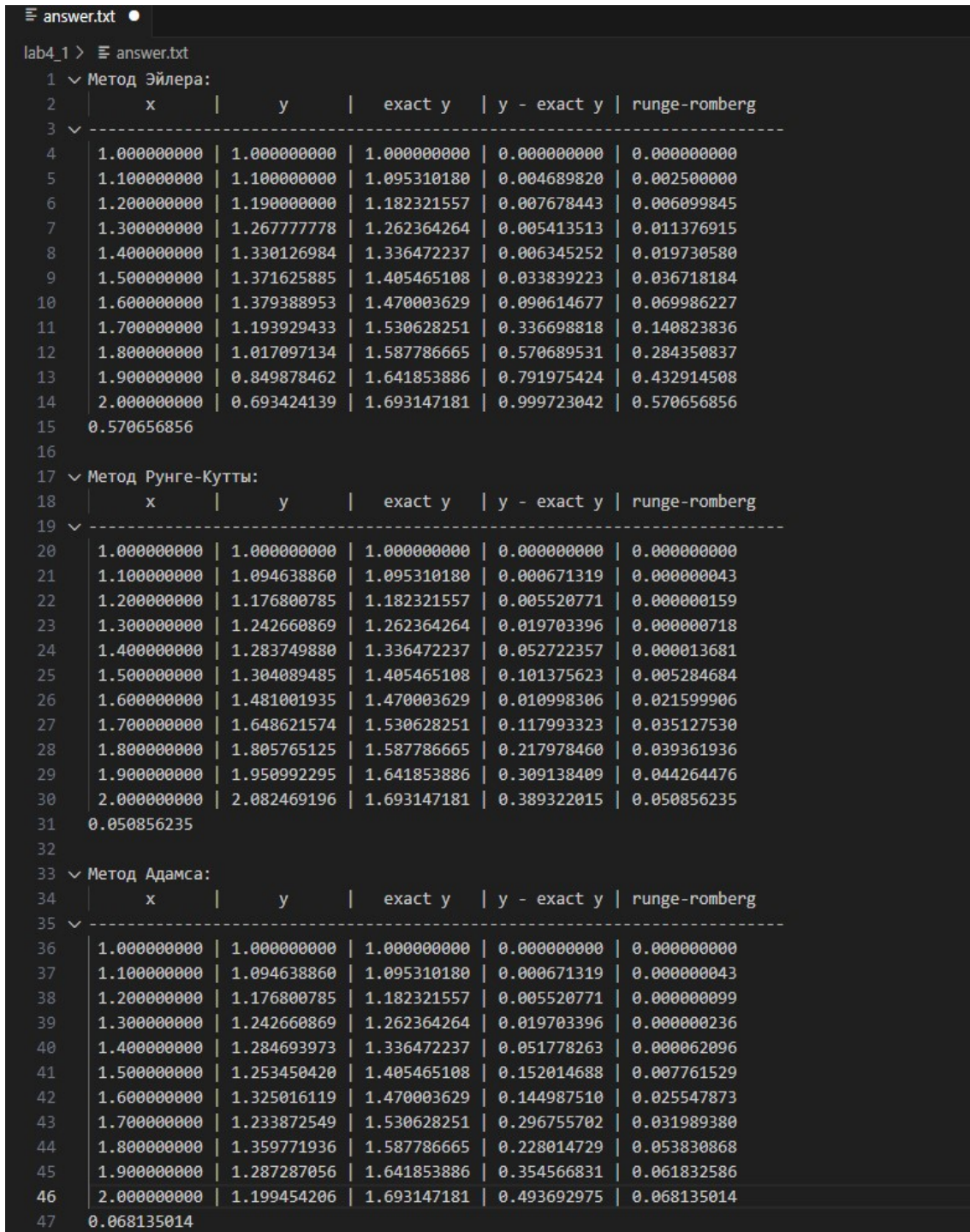
$$y'(1) = 1,$$

$$x \in [1, 2], h = 0.1$$

$$y = 1 + \ln|x|$$

Рис. 1: Входные данные

2 Результаты работы



```
lab4_1 > ≡ answer.txt
1  ▾ Метод Эйлера:
2  |   x   |   y   | exact y | y - exact y | runge-romberg
3  ▾ -----
4  | 1.00000000 | 1.00000000 | 1.00000000 | 0.00000000 | 0.00000000
5  | 1.10000000 | 1.10000000 | 1.095310180 | 0.004689820 | 0.00250000
6  | 1.20000000 | 1.19000000 | 1.182321557 | 0.007678443 | 0.006099845
7  | 1.30000000 | 1.267777778 | 1.262364264 | 0.005413513 | 0.011376915
8  | 1.40000000 | 1.330126984 | 1.336472237 | 0.006345252 | 0.019730580
9  | 1.50000000 | 1.371625885 | 1.405465108 | 0.033839223 | 0.036718184
10 | 1.60000000 | 1.379388953 | 1.470003629 | 0.090614677 | 0.069986227
11 | 1.70000000 | 1.193929433 | 1.530628251 | 0.336698818 | 0.140823836
12 | 1.80000000 | 1.017097134 | 1.587786665 | 0.570689531 | 0.284350837
13 | 1.90000000 | 0.849878462 | 1.641853886 | 0.791975424 | 0.432914508
14 | 2.00000000 | 0.693424139 | 1.693147181 | 0.999723042 | 0.570656856
15 | 0.570656856
16
17 ▾ Метод Рунге-Кутты:
18 |   x   |   y   | exact y | y - exact y | runge-romberg
19 ▾ -----
20 | 1.00000000 | 1.00000000 | 1.00000000 | 0.00000000 | 0.00000000
21 | 1.10000000 | 1.094638860 | 1.095310180 | 0.000671319 | 0.000000043
22 | 1.20000000 | 1.176800785 | 1.182321557 | 0.005520771 | 0.000000159
23 | 1.30000000 | 1.242660869 | 1.262364264 | 0.019703396 | 0.000000718
24 | 1.40000000 | 1.283749880 | 1.336472237 | 0.052722357 | 0.000013681
25 | 1.50000000 | 1.304089485 | 1.405465108 | 0.101375623 | 0.005284684
26 | 1.60000000 | 1.481001935 | 1.470003629 | 0.010998306 | 0.021599906
27 | 1.70000000 | 1.648621574 | 1.530628251 | 0.117993323 | 0.035127530
28 | 1.80000000 | 1.805765125 | 1.587786665 | 0.217978460 | 0.039361936
29 | 1.90000000 | 1.950992295 | 1.641853886 | 0.309138409 | 0.044264476
30 | 2.00000000 | 2.082469196 | 1.693147181 | 0.389322015 | 0.050856235
31 | 0.050856235
32
33 ▾ Метод Адамса:
34 |   x   |   y   | exact y | y - exact y | runge-romberg
35 ▾ -----
36 | 1.00000000 | 1.00000000 | 1.00000000 | 0.00000000 | 0.00000000
37 | 1.10000000 | 1.094638860 | 1.095310180 | 0.000671319 | 0.000000043
38 | 1.20000000 | 1.176800785 | 1.182321557 | 0.005520771 | 0.000000099
39 | 1.30000000 | 1.242660869 | 1.262364264 | 0.019703396 | 0.000000236
40 | 1.40000000 | 1.284693973 | 1.336472237 | 0.051778263 | 0.000062096
41 | 1.50000000 | 1.253450420 | 1.405465108 | 0.152014688 | 0.007761529
42 | 1.60000000 | 1.325016119 | 1.470003629 | 0.144987510 | 0.025547873
43 | 1.70000000 | 1.233872549 | 1.530628251 | 0.296755702 | 0.031989380
44 | 1.80000000 | 1.359771936 | 1.587786665 | 0.228014729 | 0.053830868
45 | 1.90000000 | 1.287287056 | 1.641853886 | 0.354566831 | 0.061832586
46 | 2.00000000 | 1.199454206 | 1.693147181 | 0.493692975 | 0.068135014
47 | 0.068135014
```

Рис. 2: Вывод программы

3 Исходный код

```
1 #include <functional>
2 #include <iostream>
3 #include <fstream>
4 #include <vector>
5 #include <tuple>
6 #include <iomanip>
7
8 using namespace std;
9 using tddd = tuple<double, double, double>;
10
11 /* f(x, y, z) */
12 using func = function<double(double, double, double)>;
13 using vect = vector<tddd>;
14 using vec = vector<double>;
15
16 ofstream fout("answer.txt");
17 const double EPS = 1e-9;
18
19
20 bool leq(double a, double b) {
21     return (a < b) || (abs(b - a) < EPS);
22 }
23
24
25 double g(double x, double y, double z) {
26     return (-x/z);
27 }
28
29 double f(double x, double y, double z) {
30     (void)x;
31     (void)y;
32     return z;
33 }
34
35 vect solve_euler(double l, double r, double y0, double z0, double h){
36     vect res;
37     double xk = l;
38     double yk = y0;
39     double zk = z0;
40     res.push_back(make_tuple(xk, yk, zk));
41     while (leq(xk + h, r)) {
42         double dy = h * f(xk, yk, zk);
43         double dz = h * g(xk, yk, zk);
44         xk += h;
45         yk += dy;
46         zk += dz;
47         res.push_back(make_tuple(xk, yk, zk));
48     }
49     return res;
50 }
51 vect solve_runge(double l, double r, double y0, double z0, double h) { // ый Порядок
52     vect res;
53     double xk = l;
54     double yk = y0;
55     double zk = z0;
56     res.push_back(make_tuple(xk, yk, zk));
57     while (leq(xk + h, r)) {
58         double K1 = h * f(xk, yk, zk);
59         double L1 = h * g(xk, yk, zk);
60         double K2 = h * f(xk + 0.5 * h, yk + 0.5 * K1, zk + 0.5 * L1);
```

```

61     double L2 = h * g(xk + 0.5 * h, yk + 0.5 * K1, zk + 0.5 * L1);
62     double K3 = h * f(xk + 0.5 * h, yk + 0.5 * K2, zk + 0.5 * L2);
63     double L3 = h * g(xk + 0.5 * h, yk + 0.5 * K2, zk + 0.5 * L2);
64     double K4 = h * f(xk + h, yk + K3, zk + L3);
65     double L4 = h * g(xk + h, yk + K3, zk + L3);
66     double dy = (K1 + 2.0 * K2 + 2.0 * K3 + K4) / 6.0;
67     double dz = (L1 + 2.0 * L2 + 2.0 * L3 + L4) / 6.0;
68     xk += h;
69     yk += dy;
70     zk += dz;
71     res.push_back(make_tuple(xk, yk, zk));
72 }
73 return res;
74 }
75
76 double calc_tuple(func f, tddd xyz) {
77     return f(get<0>(xyz), get<1>(xyz), get<2>(xyz));
78 }
79
80 vect solve_adams(double l, double r, double y0, double z0, double h) { //ый4 порядок
81     if (1 + 3.0 * h > r) {
82         throw invalid_argument("h is too big"); //Многошаговый метод: решениезависитнеотданныхвходномуэле,
            аотнескольких
83     } // Черезинтеграл
84     vect res = solve_runge(1, 1 + 3.0 * h, y0, z0, h); // Первыеточкичерезrunge
85     size_t cnt = res.size();
86     double xk = get<0>(res.back());
87     double yk = get<1>(res.back());
88     double zk = get<2>(res.back());
89     while (leq(xk + h, r)) {
90         /* Предиктор*/
91         double dy = (h / 24.0) * (55.0 * calc_tuple(f, res[cnt - 1])
92             - 59.0 * calc_tuple(f, res[cnt - 2])
93             + 37.0 * calc_tuple(f, res[cnt - 3])
94             - 9.0 * calc_tuple(f, res[cnt - 4]));
95         double dz = (h / 24.0) * (55.0 * calc_tuple(g, res[cnt - 1])
96             - 59.0 * calc_tuple(g, res[cnt - 2])
97             + 37.0 * calc_tuple(g, res[cnt - 3])
98             - 9.0 * calc_tuple(g, res[cnt - 4]));
99         double xk1 = xk + h;
100        double yk1 = yk + dy;
101        double zk1 = zk + dz;
102        res.push_back(make_tuple(xk1, yk1, zk1));
103        ++cnt;
104        /* Корректор*/
105        dy = (h / 24.0) * (9.0 * calc_tuple(f, res[cnt - 1])
106            + 19.0 * calc_tuple(f, res[cnt - 2])
107            - 5.0 * calc_tuple(f, res[cnt - 3])
108            + 1.0 * calc_tuple(f, res[cnt - 4]));
109        dz = (h / 24.0) * (9.0 * calc_tuple(g, res[cnt - 1])
110            + 19.0 * calc_tuple(g, res[cnt - 2])
111            - 5.0 * calc_tuple(g, res[cnt - 3])
112            + 1.0 * calc_tuple(g, res[cnt - 4]));
113        xk += h;
114        yk += dy;
115        zk += dz;
116        res.pop_back();
117        res.push_back(make_tuple(xk, yk, zk));
118    }
119    return res;
120 }
121

```

```

122 double max_runge_romberg(const vect & y_2h, const vect & y_h, double p) {
123     double coef = 1.0 / (pow(2, p) - 1.0);
124     double res = 0.0;
125     for (size_t i = 0; i < y_2h.size(); ++i) {
126         res = max(res, coef * abs(get<1>(y_2h[i]) - get<1>(y_h[2 * i])));
127     }
128     return res;
129 }
130
131
132 double y(double x) {
133     return 1 + log(abs(x));
134 }
135
136 double runge_romberg(double y1, double y2, int64_t p) {
137     return abs((y1 - y2) / (pow(2, p) - 1));
138 }
139
140
141 void print_data(vector<tddd> &sol_h1, vector<tddd> &sol_h2, int64_t p) {
142     fout << " x |" << " y\t |" << " exact y |" << " y - exact y | runge-romberg\n";
143     -----\n";
144     for (int i = 0; i < sol_h1.size(); ++i) {
145         double exact_y = y(get<0>(sol_h1[i]));
146         fout << fixed << setprecision(9) << " " << get<0>(sol_h1[i]) << " | " << get<1>(sol_h1[i]) << " | "
147             << exact_y << " | " << abs(exact_y - get<1>(sol_h1[i])) << " | " << runge_romberg(get<1>(
148                 sol_h1[i]), get<1>(sol_h2[2*i]), p) << endl;
149     }
150     fout << endl;
151 }
152
153 int main() {
154     double l = 1, r = 2, y0 = 1, z0 = 1, h = 0.1;
155
156     vector<tddd> sol_euler_h1 = solve_euler(l, r, y0, z0, h), sol_euler_h2 = solve_euler(l, r, y0, z0, h/2)
157     ;
158     fout << "Метод Эйлера:" << endl;
159     print_data(sol_euler_h1, sol_euler_h2, 1);
160     fout << max_runge_romberg(sol_euler_h1, sol_euler_h2, 1);
161     fout << endl;
162     fout << endl;
163
164     vector<tddd> sol_runge_h1 = solve_runge(l, r, y0, z0, h), sol_runge_h2 = solve_runge(l, r, y0, z0, h/2)
165     ;
166     fout << "Метод РунгеКутты-:" << endl;
167     print_data(sol_runge_h1, sol_runge_h2, 4);
168     fout << max_runge_romberg(sol_runge_h1, sol_runge_h2, 4);
169     fout << endl;
170     fout << endl;
171
172     vector<tddd> sol_adams_h1 = solve_adams(l, r, y0, z0, h), sol_adams_h2 = solve_adams(l, r, y0, z0, h/2)
173     ;
174     fout << "Метод Адамса:" << endl;
175     print_data(sol_adams_h1, sol_adams_h2, 4);
176     fout << max_runge_romberg(sol_adams_h1, sol_adams_h2, 4);
177     fout << endl;
178     fout << endl;
179 }

```

2 Метод стрельбы и конечно-разностный метод

1 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

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

Рис. 3: Входные данные

2 Результаты работы

```
≡ answer.txt X
lab4_2 > ≡ answer.txt
1  Метод стрельбы:
2  | x | y | exact y | y - exact y | runge-romberg
3  -----
4  | 0.000000000 | 1.323117585 | 2.000000000 | 0.676882415 | 0.000000017
5  | 0.100000000 | 1.509552891 | 2.079833372 | 0.570280481 | 0.000000003
6  | 0.200000000 | 1.667529128 | 2.118666453 | 0.451137325 | 0.000000016
7  | 0.300000000 | 1.795054853 | 2.115486948 | 0.320432095 | 0.000000020
8  | 0.400000000 | 1.890128749 | 2.069227506 | 0.179098757 | 0.000000015
9  | 0.500000000 | 1.950693856 | 1.978676971 | 0.027983115 | 0.000000000
10
11  Погрешность вычислений:
12  0.000000020
13
14  Конечно-разностный метод:
15  | x | y | exact y | y - exact y | runge-romberg
16  -----
17  | 0.000000000 | 2.000000000 | 2.000000000 | 0.000000000 | 0.000000000
18  | 0.100000000 | 2.074000727 | 2.079833372 | 0.005832645 | 0.000001371
19  | 0.200000000 | 2.107058524 | 2.118666453 | 0.011607930 | 0.000002967
20  | 0.300000000 | 2.098220631 | 2.115486948 | 0.017266317 | 0.000004314
21  | 0.400000000 | 2.046481390 | 2.069227506 | 0.022746116 | 0.000004112
22  | 0.500000000 | 1.950693856 | 1.978676971 | 0.027983115 | 0.000000000
23
24  Погрешность вычислений:
25  0.000004314
26
```

Рис. 4: Вывод программы

3 Исходный код

```
1  #include <cmath>
2  #include <iostream>
3  #include <vector>
4  #include <tuple>
5  #include <iostream>
6  #include <vector>
7  #include <functional>
8  #include <fstream>
9  #include <iomanip>
10
11 using namespace std;
12 using tddd = tuple<double, double, double>;
13 /* f(x, y, z) */
14 using func = function<double(double, double, double)>;
15 using vect = vector<tddd>;
16 using vec = vector<double>;
17
18 const double PI = acos(-1.0);
19 ofstream fout("answer.txt");
20
21
22 double exact_y (double x) {
23     return sin(x) + 2 - sin(x)*log((1+sin(x))/(1-sin(x)));
24 }
25
26 double g(double x, double y, double z) {
27     return tan(x)*z - 2*y;
28 }
29
30 double f(double x, double y, double z) {
31     (void)x;
32     (void)y;
33     return z;
34 }
35
36 double px(double x) {
37     return -tan(x);
38 }
39
40 double qx(double x) {
41     return 2.0;
42 }
43
44 double fx(double x) {
45     (void)x;
46     return 0.0;
47 }
48
49 template <class T>
50 class Trid {
51 private:
52     const T EPS = 1e-6;
53
54     int n;
55     vector<T> a, b, c;
56 public:
57     Trid(const int &size) : n(size), a(n), b(n), c(n) {}
58     Trid(vector<double> &_a, vector<double> &_b, vector<double> &_c) : n(_a.size()), a(_a), b(_b), c(_c)
59         {};
```

```

60     vector<T> Solve(const vector<T> &d) {
61         vector<T> p(n);
62         p[0] = -c[0] / b[0];
63         vector<T> q(n);
64         q[0] = d[0] / b[0];
65         for (int i = 1; i < n; ++i) {
66             p[i] = -c[i] / (b[i] + a[i]*p[i-1]);
67             q[i] = (d[i] - a[i]*q[i-1])/(b[i] + a[i]*p[i-1]);
68         }
69         vector<T> x(n);
70         x.back() = q.back();
71         for (int i = n - 2; i >= 0; --i) {
72             x[i] = p[i] * x[i+1] + q[i];
73         }
74         return x;
75     }
76
77     friend istream & operator >> (istream &in, Trid<T> &t) {
78         in >> t.b[0] >> t.c[0];
79         for (int i = 1; i < t.n - 1; ++i) {
80             in >> t.a[i] >> t.b[i] >> t.c[i];
81         }
82         in >> t.a.back() >> t.b.back();
83         return in;
84     }
85
86     ~Trid() = default;
87 };
88
89
90 const double EPS = 1e-9;
91
92 bool leq(double a, double b) {
93     return (a < b) || (abs(b - a) < EPS);
94 }
95
96
97 vect runge_solve(double l, double r, double y0, double z0, double h) {
98     vect res;
99     double xk = l;
100    double yk = y0;
101    double zk = z0;
102    res.push_back(make_tuple(xk, yk, zk));
103    while (leq(xk + h, r)) {
104        double K1 = h * f(xk, yk, zk);
105        double L1 = h * g(xk, yk, zk);
106        double K2 = h * f(xk + 0.5 * h, yk + 0.5 * K1, zk + 0.5 * L1);
107        double L2 = h * g(xk + 0.5 * h, yk + 0.5 * K1, zk + 0.5 * L1);
108        double K3 = h * f(xk + 0.5 * h, yk + 0.5 * K2, zk + 0.5 * L2);
109        double L3 = h * g(xk + 0.5 * h, yk + 0.5 * K2, zk + 0.5 * L2);
110        double K4 = h * f(xk + h, yk + K3, zk + L3);
111        double L4 = h * g(xk + h, yk + K3, zk + L3);
112        double dy = (K1 + 2.0 * K2 + 2.0 * K3 + K4) / 6.0;
113        double dz = (L1 + 2.0 * L2 + 2.0 * L3 + L4) / 6.0;
114        xk += h;
115        yk += dy;
116        zk += dz;
117        res.push_back(make_tuple(xk, yk, zk));
118    }
119    return res;
120 }
121

```

```

122 double runge_romberg_max(const vect & y_2h, const vect & y_h, double p) {
123     double coef = 1.0 / (pow(2, p) - 1.0);
124     double res = 0.0;
125     for (size_t i = 0; i < y_2h.size(); ++i) {
126         res = max(res, coef * abs(get<1>(y_2h[i]) - get<1>(y_h[2 * i])));
127     }
128     return res;
129 }
130
131 double get_eta_next(double eta_prev, double eta, const vect sol_prev, const vect sol, double delta, double
    gamma, double y1) {
132     double yb_prev = get<1>(sol_prev.back());
133     double zb_prev = get<2>(sol_prev.back());
134     double phi_prev = delta * yb_prev + gamma * zb_prev - y1; //  $\phi = D * y + g * z - y1$ 
135     double yb = get<1>(sol.back());
136     double zb = get<2>(sol.back());
137     double phi = delta * yb + gamma * zb - y1;
138     return eta - (eta - eta_prev) / (phi - phi_prev) * phi; // Метод секущих для решения уравнения  $\phi(\eta) = 0$ ;
139 }
140
141
142 vect shooting_solve(double a, double b, double alpha, double beta, double y0, double delta, double gamma,
    double y1, double h, double eps) { // четвертый порядок
143     double eta_prev = 0.9;
144     double eta = 0.8;
145     while (1) {
146         vect sol_prev = runge_solve(a, b, eta_prev, y0, h);
147         sol = runge_solve(a, b, eta, y0, h);
148
149         double eta_next = get_eta_next(eta_prev, eta, sol_prev, sol, delta, gamma, y1);
150         if (abs(eta_next - eta) < eps) {
151             return sol;
152         } else {
153             eta_prev = eta;
154             eta = eta_next;
155         }
156     }
157 }
158
159 class fin_dif {
160 private:
161
162     double a, b;
163     func p, q, f;
164     double alpha, beta, y0;
165     double delta, gamma, y1;
166
167 public:
168     fin_dif(const double _a, const double _b,
169         const func _p, const func _q, const func _f,
170         const double _alpha, const double _beta, const double _y0,
171         const double _delta, const double _gamma, const double _y1)
172         : a(_a), b(_b), p(_p), q(_q), f(_f),
173         alpha(_alpha), beta(_beta), y0(_y0),
174         delta(_delta), gamma(_gamma), y1(_y1) {}
175
176 };
177
178 using tridiag = Trid<double>;
179 vect fin_dif_solve(double a, double b,
180     double alpha, double beta, double y0,
181     double delta, double gamma, double y1, double h) {

```

```

182     size_t n = (b - a) / h;
183     vec xk(n + 1);
184     for (size_t i = 0; i <= n; ++i) {
185         xk[i] = a + h * i; //Разностная сетка
186     }
187     vec A(n + 1);
188     vec B(n + 1);
189     vec C(n + 1);
190     vec D(n + 1);
191     B[0] = (alpha - beta/h);
192     C[0] = beta/h;
193     D[0] = y0;
194     A.back() = -gamma/h;
195     B.back() = delta + gamma/h;
196     D.back() = y1;
197     for (size_t i = 1; i < n; ++i) { // Составляем систему трехдиагональной матрица
198         A[i] = 1.0 - px(xk[i]) * h * 0.5;
199         B[i] = -2.0 + h * h * qx(xk[i]);
200         C[i] = 1.0 + px(xk[i]) * h * 0.5;
201         D[i] = h * h * fx(xk[i]);
202     }
203     tridiag sys_eq(A, B, C);
204     vec yk = sys_eq.Solve(D);
205     vect res;
206     for (size_t i = 0; i <= n; ++i) {
207         res.push_back(make_tuple(xk[i], yk[i], NAN));
208     }
209     return res;
210 }
211
212 double runge_romberg(double y1, double y2, int64_t p) {
213     return abs((y1 - y2) / (pow(2, p) - 1));
214 }
215
216 void print_data(vector<tddd> &sol_h1, vector<tddd> &sol_h2, int64_t p) {
217     fout << " x | " << " y\t | " << " exact y | " << " y - exact y | runge-romberg\n";
218     -----\n";
219     for (int i = 0; i < sol_h1.size(); ++i) {
220         double ex_y = exact_y(get<0>(sol_h1[i]));
221         fout << fixed << setprecision(9) << " " << get<0>(sol_h1[i]) << " | " << get<1>(sol_h1[i]) << " | "
222             << ex_y << " | " << abs(ex_y - get<1>(sol_h1[i])) << " | " << runge_romberg(get<1>(sol_h1[i]),
223             get<1>(sol_h2[2*i]), p) << endl;
224     }
225     fout << endl;
226 }
227
228 int main() {
229     cout.precision(6);
230     cout << fixed;
231     double h = 0.1, eps = 0.001;
232
233     double a = 0, b = PI/6;
234     double alpha = 1, beta = 0, y0 = 2;
235     double delta = 1, gamma = 0, y1 = 2.5 - 0.5*log(3);
236     fout << "Метод стрельбы:" << endl;
237     vector<tddd> sol_shooting_h1 = shooting_solve(a, b, alpha, beta, y0, delta, gamma, y1, h, eps),
238     sol_shooting_h2 = shooting_solve(a, b, alpha, beta, y0, delta, gamma, y1, h / 2, eps);
239     print_data(sol_shooting_h1, sol_shooting_h2, 4);
240     fout << "Погрешность вычислений:" << endl;
241     double shooting_err = runge_romberg_max(sol_shooting_h1, sol_shooting_h2, 4);
242     fout << shooting_err << endl << endl;
243 }

```

```

241 | vector<tddd> sol_fin_dif_h1 = fin_dif_solve(a, b, alpha, beta, y0, delta, gamma, y1, h),
242 | sol_fin_dif_h2 = fin_dif_solve(a, b, alpha, beta, y0, delta, gamma, y1, h / 2);
243 | fout << "Конечноразностный- метод:" << endl;
244 | print_data(sol_fin_dif_h1, sol_fin_dif_h2, 2);
245 | fout << "Погрешность вычислений:" << endl;
246 | double fin_dif_err = runge_romberg_max(sol_fin_dif_h1, sol_fin_dif_h2, 2);
247 | fout << fin_dif_err << endl;
248 | }

```