

**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: Батулин Е.А.
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: 1

$$1 \quad \left| \begin{array}{l} y'' + y - \sin 3x = 0, \\ y(0) = 1, \\ y'(0) = 1, \\ x \in [0, 1], h = 0.1 \end{array} \right| \quad \left| \begin{array}{l} y = \cos x + \frac{11}{8} \sin x - \frac{\sin 3x}{8} \end{array} \right|$$

Рис. 1: Входные данные

2 Результаты работы

```
0.000000 0.100000 0.200000 0.300000 0.400000 0.500000 0.600000 0.700000 0.800000 0.900000 1.000000
1.000000 1.100000 1.190000 1.271955 1.347657 1.418472 1.485131 1.547581 1.604917 1.655410 1.696608
1.000000 1.095336 1.182658 1.263762 1.340008 1.412108 1.479990 1.542742 1.598640 1.645263 1.679686
1.000000 1.095336 1.182658 1.263762 1.340091 1.412215 1.480077 1.542751 1.598506 1.644932 1.679113
0.000000 0.000001 0.000001 0.000001 0.000002 0.000002 0.000002 0.000002 0.000002 0.000001 0.000001
```

Рис. 2: Вывод программы в консоли

3 Исходный код

```
1  #include <iostream>
2  #include<iomanip>
3  #include <cmath>
4
5  using namespace std;
6
7  double init(double x, double y, double z){
8      return -y + sin(3 * x);
9  }
10
11 double exsol(double x){
12     return cos(x) + 11. / 8. * sin(x) - sin(3 * x) / 8.;
13 }
14
15 double* E(double a, double b, double h, double y0, double z0){
16     int n = (b - a) / h;
17     double x[n + 1];
18     double* y = new double[n + 1];
19     double z[n + 1];
20     x[0] = a;
21     y[0] = y0;
22     z[0] = z0;
23     for (int i = 1; i <= n; ++i){
24         y[i] = y[i - 1] + h * z[i - 1];
25         z[i] = z[i - 1] + h * init(x[i - 1], y[i - 1], z[i - 1]);
26         x[i] = x[i - 1] + h;
27     }
28     return y;
29 }
30
31 double K(double x, double y, double z, double h, int i);
32
33 double L(double x, double y, double z, double h, int i){
34     if (i == 0){
35         return h * z;
36     }
37     else if (i == 3){
38         return h * (z + K(x, y, z, h, i - 1));
39     }
40     else{
41         return h * (z + K(x, y, z, h, i - 1) / 2);
42     }
43 }
44
45 double K(double x, double y, double z, double h, int i){
46     if (i == 0){
47         return h * init(x, y, z);
```

```

48     }
49     else if (i == 3){
50         return h * init(x + h, y + L(x, y, z, h, i - 1), z + K(x, y, z, h, i - 1));
51     }
52     else{
53         return h * init(x + h / 2, y + L(x, y, z, h, i - 1) / 2, z + K(x, y, z, h, i -
54             1) / 2);
55     }
56 }
57 double dz(double x, double y, double z, double h){
58     double d = 0;
59     for (int i = 0; i < 4; ++i){
60         if (i == 0 or i == 3){
61             d += K(x, y, z, h, i);
62         }
63         else{
64             d += 2 * K(x, y, z, h, i);
65         }
66     }
67     return d / 6;
68 }
69
70 double dy(double x, double y, double z, double h){
71     double d = 0;
72     for (int i = 0; i < 4; ++i){
73         if (i == 0 or i == 3){
74             d += L(x, y, z, h, i);
75         }
76         else{
77             d += 2 * L(x, y, z, h, i);
78         }
79     }
80     return d / 6;
81 }
82
83 pair<double*, double*> RK4(double a, double b, double h, double y0, double z0){
84     int n = (b - a) / h;
85     double x[n + 1];
86     double* y = new double[n + 1];
87     double* z = new double[n + 1];
88     x[0] = a;
89     y[0] = y0;
90     z[0] = z0;
91     for (int i = 1; i <= n; ++i){
92         y[i] = y[i - 1] + dy(x[i - 1], y[i - 1], z[i - 1], h);
93         z[i] = z[i - 1] + dz(x[i - 1], y[i - 1], z[i - 1], h);
94         x[i] = x[i - 1] + h;
95     }

```

```

96     return pair<double*, double*>(y, z);
97 }
98
99 double* A(double a, double b, double h, double y0, double z0){
100     int n = (b - a) / h;
101     double x[n + 1];
102     double* y = new double[n + 1];
103     double* z = new double[n + 1];
104     pair<double*, double*> yz = RK4(a, b, h, y0, z0);
105     for (int i = 0; i < 4; ++i){
106         x[i] = a + h * i;
107         y[i] = yz.first[i];
108         z[i] = yz.second[i];
109     }
110     for (int i = 4; i <= n; ++i){
111         y[i] = y[i - 1] + h / 24 * (55 * z[i - 1] - 59 * z[i - 2] + 37 * z[i - 3] - 9 *
            z[i - 4]);
112         z[i] = z[i - 1] + h / 24 * (55 * init(x[i - 1], y[i - 1], z[i - 1]) - 59 * init
            (x[i - 2], y[i - 2], z[i - 2]) + 37 * init(x[i - 3], y[i - 3], z[i - 3]) -
            9 * init(x[i - 4], y[i - 4], z[i - 4]));
113         x[i] = x[i - 1] + h;
114     }
115     return y;
116 }
117
118 double* RR(double Y_1[], double Y_2[], int n){
119     double* R = new double[n];
120     for (int i = 0; i < n; ++i){
121         R[i] = (Y_1[i * 2] - Y_2[i]) / (pow(2, 4) - 1);
122     }
123     return R;
124 }
125
126 double* Error(double Y_t[], double Y[], int n){
127     double* eps = new double[n];
128     for (int i = 0; i < n; ++i){
129         eps[i] = abs(Y_t[i] - Y[i]);
130     }
131     return eps;
132 }
133
134 int main(){
135     double a = 0;
136     double b = 1;
137     double y0 = 1;
138     double z0 = 1;
139     double h = 0.1;
140
141     double* Ans[5];

```

```

142     int n = (b - a) / h;
143
144     double* X = new double[n + 1];
145     for (int i = 0; i <= n; ++i){
146         X[i] = a + h * i;
147     }
148     Ans[0] = X;
149
150     double* Y_E = E(a, b, h, y0, z0);
151     Ans[1] = Y_E;
152
153     auto [Y_R, Z1] = RK4(a, b, h, y0, z0);
154     Ans[2] = Y_R;
155
156     double* Y_A = A(a, b, h, y0, z0);
157     Ans[3] = Y_A;
158
159     double Y_t[n + 1];
160     for (int i = 0; i <= n; ++i){
161         Y_t[i] = exsol(X[i]);
162     }
163
164     double* eps = Error(Y_t, Y_R, n + 1);
165     Ans[4] = eps;
166
167     for (int i = 0; i < 5; ++i){
168         for (int j = 0; j <= n; ++j){
169             cout << fixed << Ans[i][j] << " ";
170         }
171         cout << "\n";
172     }
173 }

```

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: 1

$$1 \quad \left| \begin{array}{l} xy'' + 2y' - xy = 0, \\ y'(1) = 0, \\ 1.5y(2) + y'(2) = e^2 \end{array} \right| \quad y(x) = \frac{e^x}{x}$$

Рис. 3: Входные данные

5 Результаты работы

```
1.000000 1.100000 1.200000 1.300000 1.400000 1.500000 1.600000 1.700000 1.800000 1.900000 2.000000
2.718280 2.731061 2.766766 2.822539 2.896574 2.987795 3.095648 3.219971 3.360917 3.518894 3.694530
2.826843 2.826843 2.852756 2.901015 2.969318 3.056228 3.160926 3.283057 3.422624 3.579925 3.755504
0.000002 0.000001 0.000002 0.000003 0.000003 0.000003 0.000003 0.000002 0.000002 0.000002 0.000002
0.108561 0.095783 0.085992 0.078479 0.072747 0.068435 0.065281 0.063088 0.061709 0.061033 0.060976
```

Рис. 4: Вывод программы в консоли

6 Исходный код

```
1  #include <iostream>
2  #include<iomanip>
3  #include <cmath>
4
5  using namespace std;
6
7  class Matrix {
8  public:
9      int rows, cols;
10     double **a;
11
12     Matrix (){
13         rows = 0;
14         cols = 0;
15         a = new double*[rows];
16         for(int i = 0; i < rows; i++)
17             a[i] = new double[cols];
18     }
19
20     Matrix(int n, int m, bool identity = 0)
21     {
22         rows = n;
23         cols = m;
24         a = new double *[rows];
25         for (int i = 0; i < rows; i++){
26             a[i] = new double[cols];
27             for (int j = 0; j < cols; j++){
28                 a[i][j] = identity * (i == j);
29             }
30         }
31     }
32
33     double* operator[](int i)
34     {
35         return a[i];
36     }
37
38     void RowSwap(int row1, int row2)
39     {
40         swap(a[row1], a[row2]);
41     }
42
43     void scan()
44     {
45         for(int i = 0; i < rows; i++)
46         {
47             for(int j = 0; j < cols; j++)
```



```

48     {
49         scanf ("%lf", &a[i][j]);
50     }
51 }
52 }
53
54 void print()
55 {
56     for(int i = 0; i < rows; i++)
57     {
58         for(int j = 0; j < cols; j++)
59         {
60             cout <<"\t"<< a[i][j] << "\t";
61         }
62         cout << endl;
63     }
64 }
65 };
66
67 double init(double x, double y, double z){
68     return -2 * z / x + y;
69 }
70
71 double exsol(double x){
72     return exp(x) / x;
73 }
74
75 double Phi(double y, double z){
76     return 1.5 * y + z - exp(2);
77 }
78
79 double p(double x){
80     return 2 / x;
81 }
82
83 double q(double x){
84     return -1;
85 }
86
87 double K(double x, double y, double z, double h, int i);
88
89 double L(double x, double y, double z, double h, int i){
90     if (i == 0){
91         return h * z;
92     }
93     else if (i == 3){
94         return h * (z + K(x, y, z, h, i - 1));
95     }
96     else{

```

```

97         return h * (z + K(x, y, z, h, i - 1) / 2);
98     }
99 }
100
101 double K(double x, double y, double z, double h, int i){
102     if (i == 0){
103         return h * init(x, y, z);
104     }
105     else if (i == 3){
106         return h * init(x + h, y + L(x, y, z, h, i - 1), z + K(x, y, z, h, i - 1));
107     }
108     else{
109         return h * init(x + h / 2, y + L(x, y, z, h, i - 1) / 2, z + K(x, y, z, h, i -
110             1) / 2);
111     }
112 }
113
114 double dz(double x, double y, double z, double h){
115     double d = 0;
116     for (int i = 0; i < 4; ++i){
117         if (i == 0 or i == 3){
118             d += K(x, y, z, h, i);
119         }
120         else{
121             d += 2 * K(x, y, z, h, i);
122         }
123     }
124     return d / 6;
125 }
126
127 double dy(double x, double y, double z, double h){
128     double d = 0;
129     for (int i = 0; i < 4; ++i){
130         if (i == 0 or i == 3){
131             d += L(x, y, z, h, i);
132         }
133         else{
134             d += 2 * L(x, y, z, h, i);
135         }
136     }
137     return d / 6;
138 }
139
140 pair<double*, double*> RK4(double a, double b, double h, double y0, double z0){
141     int n = (b - a) / h;
142     double x[n + 1];
143     double* y = new double[n + 1];
144     double* z = new double[n + 1];
145     x[0] = a;

```

```

145     y[0] = y0;
146     z[0] = z0;
147     for (int i = 1; i <= n; ++i){
148         y[i] = y[i - 1] + dy(x[i - 1], y[i - 1], z[i - 1], h);
149         z[i] = z[i - 1] + dz(x[i - 1], y[i - 1], z[i - 1], h);
150         x[i] = x[i - 1] + h;
151     }
152     return pair<double*, double*>(y, z);
153 }
154
155 double* S(double a, double b, double h, double z0, double eps){
156     double nu[3] = {1, 0.8, 0};
157     double phi[3] = {0, 0, 0};
158     int n = (b - a) / h;
159
160     pair<double*, double*> R;
161     R = RK4(a, b, h, nu[0], z0);
162     phi[0] = Phi(R.first[n], R.second[n]);
163     R = RK4(a, b, h, nu[1], z0);
164     phi[1] = Phi(R.first[n], R.second[n]);
165     phi[2] = phi[1];
166     while (abs(phi[1]) > eps){
167         nu[2] = nu[1] - (nu[1] - nu[0]) / (phi[1] - phi[0]) * phi[1];
168         R = RK4(a, b, h, nu[2], z0);
169         phi[2] = Phi(R.first[n], R.second[n]);
170         nu[0] = nu[1];
171         nu[1] = nu[2];
172         phi[0] = phi[1];
173         phi[1] = phi[2];
174     }
175     return R.first;
176 }
177
178 double* FD(double a, double b, double h, double z0, double eps){
179     int n = (b - a) / h;
180     double X[n + 1];
181     for (int i = 0; i <= n; ++i){
182         X[i] = a + h * i;
183     }
184     Matrix A = Matrix(n + 1, 3);
185     double B[n + 1];
186     A[0][0] = 0;
187     A[0][1] = -1;
188     A[0][2] = 1;
189     B[0] = h * z0;
190     for (int i = 1; i < n; ++i){
191         A[i][0] = 1 - p(X[i]) * h / 2;
192         A[i][1] = -2 + pow(h, 2) * q(X[i]);
193         A[i][2] = 1 + p(X[i]) * h / 2;

```

```

194     B[i] = 0;
195 }
196 A[n][0] = -1;
197 A[n][1] = 1 + 1.5 * h;
198 A[n][2] = 0;
199 B[n] = exp(2) * h;
200
201 double P[n + 1], Q[n + 1];
202 double* Y = new double[n + 1];
203 P[0] = -A[0][2] / A[0][1];
204 Q[0] = B[0] / A[0][1];
205 for (int i = 1; i < n + 1; ++i){
206     P[i] = -A[i][2] / (A[i][1] + A[i][0] * P[i - 1]);
207     Q[i] = (B[i] - A[i][0] * Q[i - 1]) / (A[i][1] + A[i][0] * P[i - 1]);
208 }
209
210 Y[n] = Q[n];
211 for (int i = n - 1; i > -1; --i){
212     Y[i] = P[i] * Y[i + 1] + Q[i];
213 }
214 return Y;
215 }
216
217 double* Error(double Y_t[], double Y[], int n){
218     double* eps = new double[n];
219     for (int i = 0; i < n; ++i){
220         eps[i] = abs(Y_t[i] - Y[i]);
221     }
222     return eps;
223 }
224
225
226 int main(){
227     double a = 1;
228     double b = 2;
229     double z0 = 0;
230     double h = 0.1;
231     double eps = 0.001;
232
233     double* Ans[5];
234     int n = (b - a) / h;
235
236     double* X = new double[n + 1];
237     for (int i = 0; i <= n; ++i){
238         X[i] = a + h * i;
239     }
240     Ans[0] = X;
241
242     double* Y_S = S(a, b, h, z0, eps);

```

```

243     Ans[1] = Y_S;
244
245     double* Y_FD = FD(a, b, h, z0, eps);
246     Ans[2] = Y_FD;
247
248     double Y_t[n + 1];
249     for (int i = 0; i <= n; ++i){
250         Y_t[i] = exsol(X[i]);
251     }
252
253     double* err = Error(Y_t, Y_S, n + 1);
254     Ans[3] = err;
255
256     err = Error(Y_t, Y_FD, n + 1);
257     Ans[4] = err;
258
259     for (int i = 0; i < 5; ++i){
260         for (int j = 0; j <= n; ++j){
261             cout << fixed << Ans[i][j] << " ";
262         }
263         cout << "\n";
264     }
265 }

```