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: М. Д. Жилин
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: 9

$$y''-(\frac{1}{x^{1/2}})y'+(\frac{1}{4x^2})(x+x^{1/2}-8)y=0$$
$$y(1)=2e,$$
$$y'(1)=2e,$$
$$x\in[1,2], h=0.1$$

$$y=\left(x^2+\frac{1}{x}\right)e^{x^{1/2}}$$

Рис. 1: Входные данные

2 Результаты работы

```
eiler_algo: 0 runge_algo: 0 adams: 0
eiler_algo: 0.219934 runge_algo: 0.288741 adams: 0.288741
eiler_algo: 0.48023 runge_algo: 0.624174 adams: 0.624174
eiler_algo: 0.795335 runge_algo: 1.02517 adams: 1.02517
eiler_algo: 1.18029 runge_algo: 1.51073 adams: 1.51087
eiler_algo: 1.65098 runge_algo: 2.10061 adams: 2.10058
eiler_algo: 2.22444 runge_algo: 2.81579 adams: 2.81561
eiler_algo: 2.91904 runge_algo: 3.67889 adams: 3.67843
eiler_algo: 3.75472 runge_algo: 4.7145 adams: 4.71366
eiler_algo: 15.9969 runge_algo: 17.1931 adams: 17.1919
```

Рис. 2: Вывод программы в консоли

3 Исходный код

```
#include <bits/stdc++.h>

using namespace std;
using d_vect = vector<double >;

double K(double x, double y, double dy, double h, int i);
double L(double x, double y, double dy, double h, int i);
double d_f(double x, double y, double dy);
double correct_f(double x);
double diff_dy(double x, double y, double dy, double h);
double diffY(double x, double y, double dy, double h);
d_vect eiler_algo(double l, double r, double h, double y0, double dy0);
vector<d_vect> runge_algo(double l, double r, double h, double y0, double dy0);
d_vect adams(double l, double r, double h, double y0, double dy0);


double d_f(double x, double y, double dy) {
    return dy/pow(x, 1/2) - (x + pow(x, 1/2) - 8)*y/(4*x*x);
}

double correct_f(double x) {
    return (x*x + 1/x)*exp(pow(x, 1/2));
}

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

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

double diffY(double x, double y, double dy, double h){
    double d = 0;
    for (int i = 0; i < 4; i++)
        if (i == 0 || i == 3)
```

```
47              d += L(x, y, dy, h, i);
48          else
49              d += 2 *L(x, y, dy, h, i);
50      return d / 6;
51  }
52
53  double diff_dy(double x, double y, double dy, double h){
54      double d = 0;
55      for (int i = 0; i < 4; i++)
56          if (i == 0 || i == 3)
57              d += K(x, y, dy, h, i);
58          else
59              d += 2 * K(x, y, dy, h, i);
60      return d / 6;
61  }
62
63  d_vect eiler_algo(double l, double r, double h, double y0, double dy0) {
64      double x = l, dy = dy0, y = y0;
65      int n = static_cast<int>((r - l) / h);
66      d_vect res(n);
67      res[0] = y0;
68      for(int i = 0; i < n-1; i++) {
69          double dy1 = dy + h * d_f(x, y, dy);
70          y = y + h * dy;
71          res[i+1] = y;
72          dy = dy1;
73          x += h;
74      }
75      return res;
76  }
77
78  vector<d_vect> runge_algo(double l, double r, double h, double y0, double dy0) {
79      double x = l, dy = dy0, y = y0;
80      int n = static_cast<int>((r - l) / h);
81      d_vect res(n), res_dy(n);
82      res[0] = y0;
83      res_dy[0] = dy0;
84      for(int i = 0; i < n-1; i++) {
85          double dy1 = dy + diff_dy(x, y, dy, h);
86          y = y + diffY(x, y,dy, h);
87          res[i+1] = y;
88          res_dy[i+1] = dy1;
89          dy = dy1;
90          x += h;
91      }
92      return {res, res_dy};
93  }
94
95  d_vect adams(double l, double r, double h, double y0, double dy0) {
```

```
96      int n = static_cast<int>((r - l) / h);
97      d_vect res(n), res_dy(n), x(n);
98      auto runge = runge_algo(l, r, h, y0, dy0);
99      for(int i = 0; i < 4; i++) {
100         x[i] = l + h * i;
101         res[i] = runge[0][i];
102         res_dy[i] = runge[1][i];
103     }
104     for (int i = 4; i < n; i++){
105         res[i] = res[i - 1] + h / 24 * (55 * res_dy[i - 1] - 59 * res_dy[i - 2] + 37 *
                res_dy[i - 3] - 9 * res_dy[i - 4]);
106         res_dy[i] = res_dy[i - 1] + h / 24 * (55 * d_f(x[i - 1], res[i - 1], res_dy[i
                -1]) - 59 * d_f(x[i - 2], res[i - 2], res_dy[i-2]) + 37 * d_f(x[i - 3], res
                [i - 3], res_dy[i-3]) - 9 * d_f(x[i - 4], res[i - 4], res_dy[i-4]));
107         x[i] = x[i - 1] + h;
108     }
109     return res;
110 }
111
112 d_vect RR(d_vect Y1, d_vect Y2, int n, int p){
113     d_vect R(n);
114     for (int i = 0; i < n; i++)
115         R[i] = (Y1[i * 2] - Y2[i]) / (pow(2, p) - 1);
116     return R;
117 }
118
119 d_vect deviation(d_vect y_t, d_vect Y, int n){
120     d_vect eps(n);
121     for (int i = 0; i < n; i++)
122         eps[i] = abs(y_t[i] - Y[i]);
123     return eps;
124 }
125
126 int main() {
127     double h = 0.1, l = 1, r = 2, y0 = 2*exp(1), dy0 = 2*exp(1);
128     int n = static_cast<int>((r - l) / h);
129
130     d_vect res_eiler_ = eiler_algo(l, r, h, y0, dy0), res_eiler_2 = eiler_algo(l, r, 2*
            h, y0, dy0);
131     d_vect resrunge_algo = runge_algo(l, r, h, y0, dy0)[0];
132     d_vect resrunge_algo2 = runge_algo(l, r, 2*h, y0, dy0)[0];
133     d_vect resadams = adams(l, r, h, y0, dy0);
134     d_vect resadams2 = adams(l, r, 2*h, y0, dy0);
135
136     double num = l;
137     int iter = 0;
138     d_vect realY(n);
139     while (num < r-h) {
140         realY[iter] = correct_f(num);
```

```
141         cout << "eiler_algo: " << res_eiler_[iter] << " real func: " << correct_f(num)
                << " runge_algo: "<<resrunge_algo[iter] << " adams: "<<resadams[iter] <<
                endl;
142         num+=h;
143         iter+=1;
144     }
145
146     d_vect r_eiler_ = RR(res_eiler_, res_eiler_2, n / 2, 2), rrunge_algo = RR(
            resrunge_algo, resrunge_algo2, n / 2, 5);
147     d_vect radams = RR(resadams, resadams2, n / 2, 5), l_eiler_ = deviation(realY,
            res_eiler_, n);
148     d_vect lrunge_algo = deviation(realY, resrunge_algo, n), ladams = deviation(realY,
            resadams, n);
149
150     for(int i = 0; i < r_eiler_.size(); i++)
151         cout << "eiler_algo: " << r_eiler_[i] << " runge_algo: "<<rrunge_algo[i] << "
                adams: "<<radams[i] << endl;
152     cout << endl << endl;
153     for(int i = 0; i < l_eiler_.size(); i++)
154         cout << "eiler_algo: " << l_eiler_[i] << " runge_algo: "<<lrunge_algo[i] << "
                adams: "<<ladams[i] << endl;
155 }
```

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: 9

$xy''-(2x+1)y' +(x+1)y=0,$ $y'(0)=1,$ $y' (1)-2y(1)=0$	$y(x) = e^x (x^2 +1)$

Рис. 3: Входные данные

5 Результаты работы

```
correct function 5.43656 shooting_algo 0.333337 difference_method 5.43656
correct function 6.63921 shooting_algo 0.445754 difference_method 6.95436
correct function 8.10109 shooting_algo 0.586273 difference_method 8.02331
correct function 9.87041 shooting_algo 0.760418 difference_method 9.37935
correct function 12.0034 shooting_algo 0.974653 difference_method 11.1024
correct function 14.5655 shooting_algo 1.23653 difference_method 13.2927
correct function 17.6328 shooting_algo 1.55486 difference_method 16.0751
correct function 21.2937 shooting_algo 1.93989 difference_method 19.6045
correct function 25.6505 shooting_algo 2.40355 difference_method 24.0732
correct function 30.822 shooting_algo 2.95968 difference_method 29.7181
correct function 36.9453 shooting_algo 3.62432 difference_method 36.9453

deviation shooting_algo 5.10323 rung shooting_algo -1.50177e-05 deviation difference_method 0 rung difference_method 0
deviation shooting_algo 6.19345 rung shooting_algo -5.04423e-07 deviation difference_method 0.315158 rung difference_method -0.194303
deviation shooting_algo 7.51481 rung shooting_algo 2.56317e-05 deviation difference_method 0.0777717 rung difference_method -0.176951
deviation shooting_algo 9.10999 rung shooting_algo 6.91164e-05 deviation difference_method 0.491061 rung difference_method -0.126676
deviation shooting_algo 11.0287 rung shooting_algo 0.000137943 deviation difference_method 0.900949 rung difference_method -0.0241854
```

Рис. 4: Вывод программы в консоли

6 Исходный код

```cpp
#include <bits/stdc++.h>

using namespace std;
using d_vect = vector<double >;

const double precision = 0.0001;

double func(double x) {
    return exp(x) * (x * x + 1);
}

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

double g(double y, double dy) {
    return dy - 2*y;
}

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

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

double f(double x) {
    return 0;
}

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

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

double K(double x, double y, double dy, double h, int i){
    if (i == 0)
        return h * dfunc(x, y, dy);
    else if (i == 3)
        return h * dfunc(x + h, y + L(x, y, dy, h, i - 1), dy + K(x, y, dy, h, i-1));
```

```
48      else
49          return h * dfunc(x + h / 2, y + L(x, y, dy, h, i - 1) / 2, dy + K(x, y, dy, h,
                i-1) / 2);
50  }
51
52  double diff_DY(double x, double y, double dy, double h){
53      double d = 0;
54      for (int i = 0; i < 4; ++i)
55          if (i == 0 || i == 3)
56              d += K(x, y, dy, h, i);
57          else
58              d += 2 * K(x, y, dy, h, i);
59      return d / 6;
60  }
61
62  double diff_Y(double x, double y, double dy, double h){
63      double d = 0;
64      for (int i = 0; i < 4; ++i)
65          if (i == 0 || i == 3)
66              d += L(x, y, dy, h, i);
67          else
68              d += 2 *L(x, y, dy, h, i);
69      return d / 6;
70  }
71
72
73  vector<d_vect> rung(double l, double r, double h, double y0, double dy0) {
74      double x = l;
75      int n = (int)((r - l) / h);
76      double dy = dy0, y = y0;
77      d_vect res(n+1), resDY(n+1);
78      res[0] = y0;
79      resDY[0] = dy0;
80      for(int i = 1; i <= n; ++i) {
81          double dy1 = dy + diff_DY(x, y, dy, h);
82          y = y + diff_Y(x, y,dy, h);
83          res[i] = y;
84          resDY[i] = dy1;
85          dy = dy1;
86          x += h;
87      }
88      return {res, resDY};
89  }
90
91  d_vect shooting_algo(double l, double r, double h, double dy0) {
92      double n1 = 1, n2 = 0.8, n3, g1, g2, g3;
93      vector<d_vect> res1 = rung(l, r, h, n1, dy0);
94      double res1y = res1[0][res1[0].size()-1], res1dy = res1[1][res1[1].size()-1];
95      vector<d_vect> res2 = rung(l, r, h, n2, dy0);
```

```
96      double res2y = res2[0][res2[0].size()-1], res2dy = res2[1][res2[1].size()-1];
97      g1 = g(res1y, res1dy);
98      g2 = g(res2y, res2dy);
99      vector<d_vect> res;
100     while (abs(g2) > precision) {
101         n3 = n2 - (n2 - n1) / (g2 - g1) * g2;
102         res = rung(l, r, h, n3, dy0);
103         double resy = res[0][res[0].size()-1], resdy = res[1][res[1].size()-1];
104         g3 = g(resy, resdy);
105         n1 = n2;
106         n2 = n3;
107         g1 = g2;
108         g2 = g3;
109     }
110     return res[0];
111 }
112
113 d_vect difference_method(double l, double r, double h, double y0, double y1) {
114     int n = (int) ((r - l) / h);
115     vector<d_vect> A(n, d_vect(n));
116     d_vect d(n);
117     double x = l+h;
118     for (int i = 0; i < n; ++i) {
119         A[i][i] = -2 + h * h * q(x);
120         if (i > 0) A[i][i - 1] = 1 - p(x) * h / 2;
121         if (i < n - 1) A[i][i + 1] = (1 + p(x) * h / 2);
122         x+=h;
123     }
124     d[0] = h * h * f(l+h) - (1 - p(l+h) * h / 2) * y0;
125     d[n - 1] = h * h * f(r-h) - (1 + p(r-h) * h / 2) * y1;
126     x = l+2*h;
127     for (int i = 1; i < n - 1; ++i) {
128         d[i] = h * h * f(x);
129         x+=h;
130     }
131     d_vect P(n), Q(n);
132     for(int i = 0; i < n; ++i) {
133         if(i == 0) {
134             P[i] = -A[i][i+1] / A[i][i];
135             Q[i] = d[i] / A[i][i];
136         } else if(i == n - 1) {
137             P[i] = 0;
138             Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1])
                    ;
139         } else {
140             P[i] = -A[i][i+1] / (A[i][i] + A[i][i - 1] * P[i - 1]);
141             Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1])
                    ;
142         }
```

```
143     }
144     d_vect y(n+1);
145     for(int i = n - 1; i >= 0; --i) {
146         if(i == n - 1) y[i] = Q[i];
147         else y[i] = P[i] * y[i+1] + Q[i];
148     }
149     y[0] = y0;
150     y[n] = y1;
151     return y;
152 }
153 
154 d_vect RR(d_vect Y1, d_vect Y2, int n, int p){
155     d_vect R(n);
156     for (int i = 0; i < n; ++i)
157         R[i] = (Y1[i * 2] - Y2[i]) / (pow(2, p) - 1);
158     return R;
159 }
160 
161 d_vect deviation(d_vect Yt, d_vect Y, int n){
162     d_vect eps(n);
163     for (int i = 0; i < n; ++i)
164         eps[i] = abs(Yt[i] - Y[i]);
165     return eps;
166 }
167 
168 int main() {
169     double l = 1, r = 2, dy0 = 1, h = 0.1;
170     double y0 = func(l), y1 = func(r);
171 
172     d_vect shooting = shooting_algo(l, r, h, dy0), difference = difference_method(l, r,
            h, y0, y1);
173     d_vect real(shooting.size());
174 
175     double x = l;
176     for(int i = 0; i < real.size(); i++) {
177         real[i] = func(x);
178         x+=h;
179         cout << "correct function " << real[i] << " shooting_algo " << shooting[i] << "
                difference_method " << difference[i] << endl;
180     }
181     cout << endl;
182     d_vect shooting2 = shooting_algo(l, r, h*2, dy0), difference2 = difference_method(l
            , r, h*2, y0, y1);
183     d_vect runge_shoot = RR(shooting, shooting2, real.size() / 2, 2), runge_diff = RR(
            difference, difference2, real.size() / 2, 2);
184     d_vect shoot_deviation = deviation(shooting, real, real.size() / 2), diff_deviation
            = deviation(difference, real, real.size() / 2);
185     for(int i =0; i < real.size() / 2; i++)
```

```
186         cout << "deviation shooting_algo " << shoot_deviation[i] << " rung
                shooting_algo " << runge_shoot[i] << " deviation difference_method " <<
                diff_deviation[i] << " rung difference_method " << runge_diff[i] << endl;
187 }
```