Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

# Лабораторная работа №4 по курсу «Численные методы»

Студент: А. А. Литвина
Преподаватель: И. Э. Иванов
Группа: М8О-306Б
Дата:
Оценка:
Подпись:

Москва, 2020

# Вариант №13

# 1 Численные методы решения задачи Коши

## 1 Постановка задачи

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.

Дано:
$y'' - 2\,\mathrm{tg}(x)y' - 3y = 0.$
$y(0) = 1$
$y'(0) = 3$
$x \in [0; 1]$
$h = 0.1$

## 2 Описание

Сведем ОДУ второго порядка к системе уравнений первого порядка:

$$\begin{cases} y' = z \\ z' = f(x, y, z) \end{cases}$$

Для нашей задачи:

$$\begin{cases} y' = z \\ z' = 2\,\mathrm{tg}(x)z + 3y \\ y(0) = 1 \\ z(0) = 3 \end{cases}$$

Применим к данной системе следующие методы.

### 2.1 Метод Эйлера

Для одиночного ОДУ формула имеет вид:

$$\Delta y_k = hf(x_k, y_k),$$

$$y_{k+1} = y_k + \Delta y_k.$$

Для системы уравнений:

$$\Delta y_k = hz_k,$$
$$\Delta z_k = hf(x_k, y_k, z_k),$$
$$y_{k+1} = y_k + \Delta y_k,$$
$$z_{k+1} = z_k + \Delta z_k.$$

Оценим погрешность путем сравнения с точным решением:
$\varepsilon_k = |y_k - y_t(x_k)|$, где $y_t(x_k)$ - точное решение.

Оценим погрешность методом Рунге-Ромберга.
Посчитаем $y_k^{h/2}$ - $y_k$ при шаге h/2. Вычислим

$$R^h = \frac{y_k^h - y_k^{h/2}}{(\frac{1}{2})^p - 1}.$$

В данном случае $p = 1$.

### 2.2 Метод Рунге-Кутты 4-го порядка

Для одиночного ОДУ:

$$y_{k+1} = y_k + \Delta y_k$$
$$\Delta y_k = \frac{1}{6}(K_1^k + 2K_2^k + 2K_3^k + K_4^k)$$
$$K_1^k = hf(x_k, y_k)$$
$$K_2^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k)$$
$$K_3^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2^k)$$
$$K_4^k = hf(x_k + h, y_k + K_3^k)$$

Для нашей системы:

$$K_{1y} = hz$$
$$K_{1z} = hf(x, y, z)$$
$$K_{2y} = h(z + \frac{1}{2}K_{1z})$$

$$K_{2z} = hf(x + \frac{1}{2}h, y + \frac{1}{2}K_{1y}, z + \frac{1}{2}K_{1z})$$

$$K_{3y} = h(z + \frac{1}{2}K_{2z})$$

$$K_{3z} = hf(x + \frac{1}{2}h, y + \frac{1}{2}K_{2y}, z + \frac{1}{2}K_{2z})$$

$$K_{4y} = h(z + K_{3z})$$

$$K_{4z} = hf(x + h, y + K_{3y}, z + K_{3z})$$

### 2.3 Метод Адамса 4-го порядка

Метод Адамса, как и все многошаговые методы, не является самостартующим, то есть для его использования необходимо иметь решения в первых четырех узлах. В узле $x_0$ решение известно из начальных условий, остальные три решения получим с помощью метода Рунге-Кутты того же порядка.

Формула для одиночного ОДУ:

$$y_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}),$$

где $f_k$ - значение подынтегральной функции в узле $x_k$.

Для нашей системы:

$$\Delta y = \frac{h}{24}(55z_i - 59z_{i-1} + 37z_{i-2} - 9z_{i-3})$$

$$\Delta z = \frac{h}{24}(55f(x_i, y_i, z_i) - 59f(x_{i-1}, y_{i-1}, z_{i-1}) + 37f(x_{i-2}, y_{i-2}, z_{i-2}) - 9f(x_{i-3}, y_{i-3}, z_{i-3}))$$

$$y_{i+1} = y_i + \Delta y$$

$$z_{i+1} = z_i + \Delta z$$

## 3 Исходный код

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

using namespace std;

double sol(double x) {
  return exp(-pow(2,0.5)*x)*((2+3*pow(2,0.5))*exp(2*x*pow(2,0.5)) +2-3*pow(2,0.5))/(4*
      cos(x));
}

double f_z(double x, double y, double z) {
  return 2*tan(x)*z+3*y;
}

void eiler(double x, double y, double z, double b, double h) {
  double h2=h/2;
  double x2=x;
  double y2=y;
  double z2=z;
  double R, del_y, del_z, del_y2, del_z2, eps;
  int n=(b-x)/h;
  cout << "EILER\n";
  cout << "x\ty\ty_t\teps\tR\n";
  cout << "---------------------------------\n";
  for (int i=0; i<n; i++) {
    del_y=h*z;
    del_z=h*f_z(x,y,z);
    y+=del_y;
    z+=del_z;
    x+=h;

    del_y2=h2*z2;
    del_z2=h2*f_z(x2,y2,z2);
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    del_y2=h2*z2;
    del_z2=h2*f_z(x2,y2,z2);
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    eps=abs(y-sol(x));
    R=(y-y2)/(0.5-1);
    cout << x << " | " << y << " | " << sol(x) << " | " << eps << " | " << R << "\n\n";
```

```
47    }
48  }
49
50  void runge_cutta(double x, double y, double z, double b, double h) {
51    double K1_y, K2_y, K3_y, K4_y, K1_z, K2_z, K3_z, K4_z, del_y, del_z;
52    double K1_y2, K2_y2, K3_y2, K4_y2, K1_z2, K2_z2, K3_z2, K4_z2, del_y2, del_z2;
53    double eps, R;
54    int n=(b-x)/h;
55    double h2=h/2;
56    double x2=x;
57    double y2=y;
58    double z2=z;
59    cout << "RUNGE_CUTTA\n";
60    cout << "x\ty\ty_t\teps\tR\n";
61    cout << "--------------------------------\n";
62    for (int i=0; i<n; i++) {
63      K1_y=h*z;
64      K1_z=h*f_z(x,y,z);
65      K2_y=h*(z+0.5*K1_z);
66      K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
67      K3_y=h*(z+0.5*K2_z);
68      K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
69      K4_y=h*(z+K3_z);
70      K4_z=h*f_z(x+h, y+K3_y, z+K3_z);
71
72      del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
73      del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
74      y+=del_y;
75      z+=del_z;
76      x+=h;
77
78      K1_y2=h2*z2;
79      K1_z2=h2*f_z(x2,y2,z2);
80      K2_y2=h2*(z2+0.5*K1_z2);
81      K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
82      K3_y2=h2*(z2+0.5*K2_z2);
83      K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
84      K4_y2=h2*(z2+K3_z2);
85      K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);
86
87      del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
88      del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
89      y2+=del_y2;
90      z2+=del_z2;
91      x2+=h2;
92
93      K1_y2=h2*z2;
94      K1_z2=h2*f_z(x2,y2,z2);
95      K2_y2=h2*(z2+0.5*K1_z2);
```

```
    K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
    K3_y2=h2*(z2+0.5*K2_z2);
    K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
    K4_y2=h2*(z2+K3_z2);
    K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);

    del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
    del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    eps=abs(y-sol(x));
    int p=4;
    R=(y-y2)/(pow(0.5,p)-1);
    cout << x << " | " << y << " | " << sol(x) << " | " << eps << " | " << R << "\n\n";
  }
}

void adams(double x, double y, double z, double b, double h) {
  double K1_y, K2_y, K3_y, K4_y, K1_z, K2_z, K3_z, K4_z, del_y, del_z;
  double K1_y2, K2_y2, K3_y2, K4_y2, K1_z2, K2_z2, K3_z2, K4_z2, del_y2, del_z2;
  int n=(b-x)/h;
  vector <double> X (n,0);
  vector <double> Y (n,0);
  vector <double> Z (n,0);
  vector <double> X2 (n,0);
  vector <double> Y2 (n,0);
  vector <double> Z2 (n,0);
  X[0]=x;
  Y[0]=y;
  Z[0]=z;
  double eps, R;
  double h2=h/2;
  double x2=x;
  double y2=y;
  double z2=z;
  X2[0]=x;
  Y2[0]=y;
  Z2[0]=z;
  cout << "ADAMS\n";
  cout << "x\ty\ty_t\teps\tR\n";
  cout << "---------------------------------\n";
  for (int i=1; i<4; i++) {
    K1_y=h*z;
    K1_z=h*f_z(x,y,z);
    K2_y=h*(z+0.5*K1_z);
    K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
    K3_y=h*(z+0.5*K2_z);
```

```
    K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
    K4_y=h*(z+K3_z);
    K4_z=h*f_z(x+h, y+K3_y, z+K3_z);

    del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
    del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
    y+=del_y;
    z+=del_z;
    x+=h;

    K1_y2=h2*z2;
    K1_z2=h2*f_z(x2,y2,z2);
    K2_y2=h2*(z2+0.5*K1_z2);
    K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
    K3_y2=h2*(z2+0.5*K2_z2);
    K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
    K4_y2=h2*(z2+K3_z2);
    K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);

    del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
    del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    K1_y2=h2*z2;
    K1_z2=h2*f_z(x2,y2,z2);
    K2_y2=h2*(z2+0.5*K1_z2);
    K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
    K3_y2=h2*(z2+0.5*K2_z2);
    K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
    K4_y2=h2*(z2+K3_z2);
    K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);

    del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
    del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    X[i]=x;
    Y[i]=y;
    Z[i]=z;

    X2[i]=x2;
    Y2[i]=y2;
    Z2[i]=z2;

    int p=4;
```

```
194     eps=abs(y-sol(x));
195     R=(y-y2)/(pow(0.5,p)-1);
196     cout << x << " | " << y << " | " << sol(x) << " | " << eps << " | " << R << "\n\n";
197   }
198
199   for (int i=3; i<n; i++) {
200     del_y=h*(55*Z[i]-59*Z[i-1]+37*Z[i-2]-9*Z[i-3])/24;
201     del_z=h*(55*f_z(X[i],Y[i],Z[i])-59*f_z(X[i-1],Y[i-1],Z[i-1])+37*f_z(X[i-2],Y[i-2],Z
          [i-2])-9*f_z(X[i-3],Y[i-3],Z[i-3]))/24;
202     X[i+1]=X[i]+h;
203     Y[i+1]=Y[i]+del_y;
204     Z[i+1]=Z[i]+del_z;
205
206     del_y2=h2*(55*Z2[i]-59*Z2[i-1]+37*Z2[i-2]-9*Z2[i-3])/24;
207     del_z2=h2*(55*f_z(X2[i],Y2[i],Z2[i])-59*f_z(X2[i-1],Y2[i-1],Z2[i-1])+37*f_z(X2[i
          -2],Y2[i-2],Z2[i-2])-9*f_z(X2[i-3],Y2[i-3],Z2[i-3]))/24;
208     X2[i]+=h2;
209     Y2[i]+=del_y2;
210     Z2[i]+=del_z2;
211
212     del_y2=h2*(55*Z2[i]-59*Z2[i-1]+37*Z2[i-2]-9*Z2[i-3])/24;
213     del_z2=h2*(55*f_z(X2[i],Y2[i],Z2[i])-59*f_z(X2[i-1],Y2[i-1],Z2[i-1])+37*f_z(X2[i
          -2],Y2[i-2],Z2[i-2])-9*f_z(X2[i-3],Y2[i-3],Z2[i-3]))/24;
214     X2[i+1]=X2[i]+h2;
215     Y2[i+1]=Y2[i]+del_y2;
216     Z2[i+1]=Z2[i]+del_z2;
217
218     int p=4;
219     eps=abs(Y[i+1]-sol(X[i+1]));
220     R=(Y[i+1]-Y2[i+1])/(pow(0.5,p)-1);
221     cout << X[i+1] << " | " << Y[i+1] << " | " << sol(X[i+1]) << " | " << eps << " | "
          << R << "\n\n";
222   }
223 }
224
225 int main() {
226   double x=0;
227   double y=1;
228   double z=3;
229   double b=1;
230   double h=0.1;
231   eiler(x,y,z,b,h);
232   runge_cutta(x,y,z,b,h);
233   adams(x,y,z,b,h);
234 }
```

# 4 Консоль

## 4.1 Метод Эйлера

```
EILER
x       y       y_t       eps       R
-----------------------------------------------
0.1 | 1.3 | 1.3176 | 0.0176002 | 0.015

0.2 | 1.63 | 1.68182 | 0.0518239 | 0.0456146

0.3 | 2.00562 | 2.11298 | 0.107353 | 0.0955461

0.4 | 2.44537 | 2.63755 | 0.192177 | 0.171323

0.5 | 2.9725 | 3.29172 | 0.31922 | 0.283497

0.6 | 3.61756 | 4.12685 | 0.509292 | 0.448827

0.7 | 4.42227 | 5.21879 | 0.796519 | 0.694196

0.8 | 5.44562 | 6.68435 | 1.23873 | 1.06387

0.9 | 6.77403 | 8.71255 | 1.93852 | 1.63352

1 | 8.53936 | 11.6288 | 3.08947 | 2.53932
```

## 4.2 Метод Рунге-Кутты

```
RUNGE_CUTTA
x       y        y_t         eps          R
-------------------------------------------------
0.1 | 1.31759 | 1.3176 | 5.43266e-06 | 5.42026e-06

0.2 | 1.68181 | 1.68182 | 1.26719e-05 | 1.26346e-05

0.3 | 2.11295 | 2.11298 | 2.33839e-05 | 2.33009e-05

0.4 | 2.63751 | 2.63755 | 4.03183e-05 | 4.01524e-05
```

0.5 | 3.29165 | 3.29172 | 6.83921e-05 | 6.80728e-05

0.6 | 4.12673 | 4.12685 | 0.000116975 | 0.000116362

0.7 | 5.21859 | 5.21879 | 0.000205108 | 0.000203905

0.8 | 6.68398 | 6.68435 | 0.000374505 | 0.000372037

0.9 | 8.71182 | 8.71255 | 0.000725483 | 0.000720038

1 | 11.6273 | 11.6288 | 0.00153016 | 0.00151677

### 4.3 Метод Адамса

ADAMS

x y y_t eps R

---

0.1 | 1.31759 | 1.3176 | 5.43266e-06 | 5.42026e-06

0.2 | 1.68181 | 1.68182 | 1.26719e-05 | 1.26346e-05

0.3 | 2.11295 | 2.11298 | 2.33839e-05 | 2.33009e-05

0.4 | 2.63657 | 2.63755 | 0.000977881 | 0.0667285

0.5 | 3.28842 | 3.29172 | 0.0032979 | 0.0432658

0.6 | 4.1194 | 4.12685 | 0.00745049 | 0.0640961

0.7 | 5.20327 | 5.21879 | 0.0155244 | 0.106836

0.8 | 6.65286 | 6.68435 | 0.0314858 | 0.160124

0.9 | 8.64814 | 8.71255 | 0.0644113 | 0.261512

1 | 11.4922 | 11.6288 | 0.136623 | 0.463777

# 2 Численные методы решения краевой задачи для ОДУ

## 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.

Дано:
$(e^x + 1)y'' - 2y' - e^x y = 0$

$y'(0) = \frac{3}{4}$

$y'(1) = \frac{e^2(e+2)}{(e+1)^2}$

## 2 Описание

### 2.1 Метод стрельбы

Сведем ОДУ второго порядка к системе уравнений первого порядка:

$$\begin{cases} y' = z \\ z' = f(x, y, z) \end{cases}$$

Для нашей задачи:

$$\begin{cases} y' = z \\ z' = \frac{2z+e^x y}{e^x+1} \\ z(0) = \frac{3}{4} \\ z(1) = \frac{e^2(e+2)}{(e+1)^2} \end{cases}$$

Обозначим неизвестное $y(0) = \eta$. Примем $\eta_0 = 1$ и $\eta_1 = 0.8$. Решим задачу Коши методом Рунге-Кутты в правом конце отрезка для $\eta_0$ и $\eta_1$: $y(b, \eta_0, z_0)$ и $y(b, \eta_1, z_0)$, где $b$ - правый конец отрезка, $z_0 = z(0)$. Найдем следующую $\eta$ по формуле:

$$\eta_{j+2} = \eta_{j+1} - \frac{\eta_{j+1} - \eta_j}{\Phi(\eta_{j+1}) - \Phi(\eta_j)}\Phi(\eta_{j+1}),$$

где $\Phi(\eta) = y(b, \eta, z_0) - z_1$, $z_1 = z(1)$.

Будем продолжать поиск $\eta$ до выполнения условия: $|\Phi(\eta_k)| \leq \varepsilon$, $\varepsilon = 0.0001$.

Еще раз решим задачу Коши методом Рунге-Кутты с найденым параметром $\eta_k$. Полученная табличная функция будет приближенным решением краевой задачи.

### 2.2 Конечно-разностный метод

Представим уравнение в виде: $y'' + p(x)y' + q(x)y = f(x)$.

Для нашей задачи: $p(x) = -\frac{2}{e^x+1}$, $q(x) = -\frac{e^x}{e^x+1}$, $f(x) = 0$, h возьмем равным 0.1.

Запишем СЛАУ с трехдиагональной матрицей:

$$A(x_k)y_{k-1} + B(x_k)y_k + C(x_k)y_{k+1} = h^2 f(x_k), \qquad k = 2, .., N-1,$$

где $A(x) = 1 - \frac{p(x)h}{2}$, $B(x) = -2 + h^2 q(x)$, $C(x) = 1 + \frac{p(x)h}{2}$.

На левой и правой границах аппроксимируем производную односторонней разностью 1-го порядка:

$$y'_0 = \frac{y_1 - y_0}{h} \Rightarrow -y_0 + y_1 = y'_0 h$$

$$y'_N = \frac{y_N - y_{N-1}}{h} \Rightarrow -y_{N-1} + y_N = y'_N h$$

Получим систему уравнений:

$$\begin{cases} -y_1 + y_2 = 0.075 \\ 1.04502y_1 - 2.0055y_2 + 0.957444y_3 = 0 \\ 1.04256y_2 - 2.00574y_3 + 0.957444y_4 = 0 \\ 1.04013y_3 - 2.00599y_4 + 0.959869y_5 = 0 \\ 1.03775y_4 - 2.00622y_5 + 0.962246y_6 = 0 \\ 1.03543y_5 - 2.00646y_6 + 0.964566y_7 = 0 \\ 1.03318y_6 - 2.00668y_7 + 0.966819y_8 = 0 \\ 1.031y_7 - 2.0069y_8 + 0.968997y_9 = 0 \\ -y_9 + y_{10} = 0.252167 \end{cases}$$

Далее применяем к полученной системе метод прогонки.

## 3 Исходный код

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

using namespace std;

const double eps=0.0001;

double f_z(double x, double y, double z) {
  return (2*z+exp(x)*y)/(exp(x)+1);
}

double sol(double x) {
  return exp(x)-1+1/(exp(x)+1);
}

double runge_cutta(double x, double y, double z, double b, double h) {
  double K1_y, K2_y, K3_y, K4_y, K1_z, K2_z, K3_z, K4_z, del_y, del_z;
  int n=(b-x)/h;
  for (int i=0; i<n; i++) {
    K1_y=h*z;
    K1_z=h*f_z(x,y,z);
    K2_y=h*(z+0.5*K1_z);
    K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
    K3_y=h*(z+0.5*K2_z);
    K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
    K4_y=h*(z+K3_z);
    K4_z=h*f_z(x+h, y+K3_y, z+K3_z);

    del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
    del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
    y+=del_y;
    z+=del_z;
    x+=h;
  }
  return z;
}

double runge_cutta2(double x, double y, double z, double b, double h) {
  double K1_y, K2_y, K3_y, K4_y, K1_z, K2_z, K3_z, K4_z, del_y, del_z;
  int n=(b-x)/h;
  for (int i=0; i<n; i++) {
    K1_y=h*z;
    K1_z=h*f_z(x,y,z);
    K2_y=h*(z+0.5*K1_z);
    K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
    K3_y=h*(z+0.5*K2_z);
```

```
48      K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
49      K4_y=h*(z+K3_z);
50      K4_z=h*f_z(x+h, y+K3_y, z+K3_z);
51
52      del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
53      del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
54      y+=del_y;
55      z+=del_z;
56      x+=h;
57
58      K1_y=h*z;
59      K1_z=h*f_z(x,y,z);
60      K2_y=h*(z+0.5*K1_z);
61      K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
62      K3_y=h*(z+0.5*K2_z);
63      K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
64      K4_y=h*(z+K3_z);
65      K4_z=h*f_z(x+h, y+K3_y, z+K3_z);
66
67      del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
68      del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
69      y+=del_y;
70      z+=del_z;
71      x+=h;
72    }
73    return z;
74  }
75
76  double Fi(double eta, double x,double z0, double z1,double b, double h) {
77    return runge_cutta(x,eta,z0,b,h)-z1;
78  }
79
80  double Fi2(double eta, double x,double z0, double z1,double b, double h) {
81    return runge_cutta2(x,eta,z0,b,h)-z1;
82  }
83
84  double new_eta(double eta0, double eta1, double x,double z0, double z1,double b,
        double h) {
85    return eta1-Fi(eta1,x,z0,z1,b,h)*(eta1-eta0)/(Fi(eta1,x,z0,z1,b,h)-Fi(eta0,x,z0,z1,b
        ,h));
86  }
87
88  double new_eta2(double eta0, double eta1, double x,double z0, double z1,double b,
        double h) {
89    return eta1-Fi2(eta1,x,z0,z1,b,h)*(eta1-eta0)/(Fi2(eta1,x,z0,z1,b,h)-Fi2(eta0,x,z0,
        z1,b,h));
90  }
91
92  void shooting_method(double x,double z0,double z1,double b, double h) {
```

```
93  double K1_y, K2_y, K3_y, K4_y, K1_z, K2_z, K3_z, K4_z, del_y, del_z;
94  double K1_y2, K2_y2, K3_y2, K4_y2, K1_z2, K2_z2, K3_z2, K4_z2, del_y2, del_z2;
95  double eta0=1;
96  double eta1=0.8;
97  double n_eta;
98  double n_Fi;
99  double e,R;
100
101 double h2=h/2;
102 double eta0_2=eta0;
103 double eta1_2=eta1;
104 double n_eta2;
105 double n_Fi2;
106 int n=(b-x)/h;
107
108 n_Fi=Fi(eta1,x,z0,z1,b,h);
109 while (abs(n_Fi)>eps) {
110   n_eta=new_eta(eta0, eta1, x, z0, z1, b, h);
111   eta0=eta1;
112   eta1=n_eta;
113   n_Fi=Fi(eta1,x,z0,z1,b,h);
114 }
115
116 n_Fi2=Fi2(eta1_2,x,z0,z1,b,h2);
117 while (abs(n_Fi2)>eps) {
118   n_eta2=new_eta2(eta0_2, eta1_2, x, z0, z1, b, h2);
119   eta0_2=eta1_2;
120   eta1_2=n_eta2;
121   n_Fi2=Fi2(eta1_2,x,z0,z1,b,h2);
122 }
123
124 double z=z0;
125 double y=eta1;
126
127 double x2=x;
128 double z2=z0;
129 double y2=eta1_2;
130
131 cout << "SHOOTING METHOD\n";
132 cout << "x\ty\teps\tR\n";
133 cout << "---------------------------------\n";
134
135 for (int i=0; i<n; i++) {
136   K1_y=h*z;
137   K1_z=h*f_z(x,y,z);
138   K2_y=h*(z+0.5*K1_z);
139   K2_z=h*f_z(x+0.5*h, y+0.5*K1_y, z+0.5*K1_z);
140   K3_y=h*(z+0.5*K2_z);
141   K3_z=h*f_z(x+0.5*h, y+0.5*K2_y, z+0.5*K2_z);
```

```
    K4_y=h*(z+K3_z);
    K4_z=h*f_z(x+h, y+K3_y, z+K3_z);

    del_y=(K1_y+2*K2_y+2*K3_y+K4_y)/6;
    del_z=(K1_z+2*K2_z+2*K3_z+K4_z)/6;
    y+=del_y;
    z+=del_z;
    x+=h;

    K1_y2=h2*z2;
    K1_z2=h2*f_z(x2,y2,z2);
    K2_y2=h2*(z2+0.5*K1_z2);
    K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
    K3_y2=h2*(z2+0.5*K2_z2);
    K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
    K4_y2=h2*(z2+K3_z2);
    K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);

    del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
    del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    K1_y2=h2*z2;
    K1_z2=h2*f_z(x2,y2,z2);
    K2_y2=h2*(z2+0.5*K1_z2);
    K2_z2=h2*f_z(x2+0.5*h2, y2+0.5*K1_y2, z2+0.5*K1_z2);
    K3_y2=h2*(z2+0.5*K2_z2);
    K3_z2=h2*f_z(x2+0.5*h2, y2+0.5*K2_y2, z2+0.5*K2_z2);
    K4_y2=h2*(z2+K3_z2);
    K4_z2=h2*f_z(x2+h2, y2+K3_y2, z2+K3_z2);

    del_y2=(K1_y2+2*K2_y2+2*K3_y2+K4_y2)/6;
    del_z2=(K1_z2+2*K2_z2+2*K3_z2+K4_z2)/6;
    y2+=del_y2;
    z2+=del_z2;
    x2+=h2;

    int p=4;
    e=abs(y-sol(x));
    R=(y-y2)/(pow(0.5,p)-1);
    cout << x << " | " << y << " | " << e << " | " << R << "\n\n";
  }
}

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

```
191
192 double q(double x) {
193   return -exp(x)/(exp(x)+1);
194 }
195
196 double f(double x) {
197   return 0;
198 }
199
200 double A(double x,double h) {
201   return 1-p(x)*h/2;
202 }
203
204 double B(double x,double h) {
205   return -2+pow(h,2)*q(x);
206 }
207
208 double C(double x,double h) {
209   return 1+p(x)*h/2;
210 }
211
212 void run_method(vector <vector<double>> A, vector <double> b, double x_i,
213   double h, vector <vector<double>> A2, vector <double> b2) {
214   int N=b.size();
215   int i;
216   double y,e,R;
217   vector <double> x (N);
218   vector <double> al (N);
219   vector <double> bet (N);
220
221   double h2=h/2;
222   double y2;
223   int N2=b2.size();
224   vector <double> x2 (N2);
225   vector <double> al2 (N2);
226   vector <double> bet2 (N2);
227
228   y=A[0][0];
229   al[0]=-A[0][1]/y;
230   bet[0]=b[0]/y;
231   for (i=1; i<N-1; i++) {
232     y=A[i][i]+A[i][i-1]*al[i-1];
233     al[i]=-A[i][i+1]/y;
234     bet[i]=(b[i]-A[i][i-1]*bet[i-1])/y;
235   }
236
237   y=A[i][i]+A[i][i-1]*al[i-1];
238   bet[i]=(b[i]-A[i][i-1]*bet[i-1])/y;
239
```

```
  x[N-1]=bet[N-1];
  for (i=N-2; i>=0; i--) {
    x[i]=al[i]*x[i+1]+bet[i];
  }

  y2=A2[0][0];
  al2[0]=-A2[0][1]/y2;
  bet2[0]=b2[0]/y2;
  for (i=1; i<N2-1; i++) {
    y2=A2[i][i]+A2[i][i-1]*al2[i-1];
    al2[i]=-A2[i][i+1]/y2;
    bet2[i]=(b2[i]-A2[i][i-1]*bet2[i-1])/y2;
  }

  y2=A2[i][i]+A2[i][i-1]*al2[i-1];
  bet2[i]=(b2[i]-A2[i][i-1]*bet2[i-1])/y2;

  x2[N2-1]=bet2[N2-1];
  for (i=N2-2; i>=0; i--) {
    x2[i]=al2[i]*x2[i+1]+bet2[i];
  }

  cout << "FINITE_DIF_METHOD\n";
  cout << "x\ty\teps\tR\n";
  cout << "---------------------------------\n";
  for (i=0; i<N; i++) {
    x_i+=h;
    int p=4;
    e=abs(sol(x_i)-x[i]);
    R=(x[i]-x2[2*i])/(pow(0.5,p)-1);
    cout << x_i << " | " << x[i] << " | " << e << " | " << R << endl << endl;
  }
}

void finite_dif_method(double x,double z0,double z1,double b,double h) {
  int N=(b-x)/h;
  vector <double> s (N,0);
  vector <vector <double>> matrix (N,vector <double> (N,0));
  double x_i=x+h;

  double h2=h/2;
  int N2=(b-x)/h2;
  double x_i2=x+h2;
  vector <double> s2 (N2,0);
  vector <vector <double>> matrix2 (N2,vector <double> (N2,0));

  matrix[0][0]=-1;
  matrix[0][1]=1;
  s[0]=h*z0;
```

```
289
290   for (int i=1; i<N-1; i++) {
291     x_i+=h;
292     matrix[i][i-1]=A(x_i,h);
293     matrix[i][i]=B(x_i,h);
294     matrix[i][i+1]=C(x_i,h);
295     s[i]=pow(h,2)*f(x_i);
296   }
297   x_i+=h;
298
299   matrix[N-1][N-2]=-1;
300   matrix[N-1][N-1]=1;
301   s[N-1]=h*z1;
302
303   matrix2[0][0]=-1;
304   matrix2[0][1]=1;
305   s2[0]=h2*z0;
306
307   for (int i=1; i<N2-1; i++) {
308     x_i2+=h2;
309     matrix2[i][i-1]=A(x_i2,h2);
310     matrix2[i][i]=B(x_i2,h2);
311     matrix2[i][i+1]=C(x_i2,h2);
312     s2[i]=pow(h2,2)*f(x_i2);
313   }
314   x_i2+=h2;
315
316   matrix2[N2-1][N2-2]=-1;
317   matrix2[N2-1][N2-1]=1;
318   s2[N2-1]=h2*z1;
319
320   run_method(matrix,s,x,h,matrix2,s2);
321 }
322
323 int main() {
324   double x=0;
325   double z0=0.75;
326   double z1=exp(2)*(exp(1)+2)/pow((exp(1)+1),2);
327   double b=1;
328   double h=0.1;
329   shooting_method(x,z0,z1,b,h);
330   finite_dif_method(x,z0,z1,b,h);
331 }
```

# 4 Консоль

## 4.1 Метод стрельбы

```
SHOOTING METHOD
x        y        eps         R
---------------------------------
0.1 | 0.580198 | 6.3068e-06 | -1.46167

0.2 | 0.671575 | 6.13546e-06 | -1.47393

0.3 | 0.775422 | 5.95803e-06 | -1.49581

0.4 | 0.893143 | 5.77337e-06 | -1.5286

0.5 | 1.02627 | 5.57991e-06 | -1.57365

0.6 | 1.17647 | 5.37558e-06 | -1.63241

0.7 | 1.34557 | 5.15777e-06 | -1.7064

0.8 | 1.53557 | 4.92325e-06 | -1.79726

0.9 | 1.74866 | 4.66807e-06 | -1.90671

1 | 1.98723 | 4.38753e-06 | -2.03663
```

## 4.2 Конечно-разностный метод

```
FINITE_DIF_METHOD
x        y        eps         R
---------------------------------
0.1 | 1.27056 | 0.690363 | -0.449043

0.2 | 1.34556 | 0.673986 | -0.445825

0.3 | 1.43537 | 0.659957 | -0.444775

0.4 | 1.54179 | 0.64865 | -0.446182
```

0.5 | 1.66672 | 0.640454 | -0.450347

0.6 | 1.81223 | 0.635767 | -0.457584

0.7 | 1.98056 | 0.635 | -0.468222

0.8 | 2.17414 | 0.638576 | -0.482607

0.9 | 2.39559 | 0.646935 | -0.501103

1 | 2.64776 | 0.660532 | -0.524099

# Выводы

В этой лабораторной работе я познакомилась с численными методами решения задачи Коши: методом Эйлера, Рунге-Кутты 4-го порядка и Адамса 4-го порядка. А также с численными методами решения краевой задачи для ОДУ: методом стрельбы и конечно-разностный методом. Для каждого метода я оценила погрешность решения методом Рунге-Ромберга и путем сравнения с точным решением.