吴宇迪 计 181 10182403

求解初值问题

$$\begin{cases} y' = \dfrac{2x}{3y^2} \\ y(0) = 1 \end{cases}$$

精确解 $y = \sqrt[3]{1+x^2}$ ， x=0.1 的精确解 1.0033222835420891993
欧拉方法:

wuyudi_5_1.m

```matlab
f = @(x, y) 2 * x / (3 * y * y);
disp(1 + 0.1 * f(0, 1))
```

MATLAB Command Window

```
 Toolbox Path Cache read in 0.02 seconds.
MATLAB Path initialized in 0.42 seconds.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

     1
```

值=1
改进欧拉法:

wuyudi_5_1.m　MendEuler.m

MendEuler.m > ...

```matlab
function E = MendEuler(f, a, b, N, ya)
    %f是微分方程右端函数句柄
    %a,b是自变量的取值区间[a,b]的端点
    %N是区间等分的个数
    %ya表初值y(a)
    %E=[x',y']是自变量X和解Y所组成的矩阵
    h = (b - a) / N;
    y = zeros(1, N + 1);
    % x = zeros(1, N + 1);
    y(1) = ya;
    x = a:h:b;

    for i = 1:N
        y1 = y(i) + h * feval(f, x(i), y(i));
        y2 = y(i) + h * feval(f, x(i + 1), y1);
        y(i + 1) = (y1 + y2) / 2;
    end

    E = [x', y'];
```

求解结果

wuyudi_5_1.m × MendEuler.m

wuyudi_5_1.m

```matlab
disp(MendEuler(@(x, y) 2 * x / (3 * y^2), 0, 1, 10, 1))
```

MATLAB Command Window

```
 Toolbox Path Cache read in 0.03 seconds.
MATLAB Path initialized in 0.38 seconds.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

                   0   1.000000000000000
   0.100000000000000   1.003333333333333
   0.200000000000000   1.013180434398852
   0.300000000000000   1.029171244550309
   0.400000000000000   1.050751079998023
   0.500000000000000   1.077252310612832
   0.600000000000000   1.107965053358377
   0.700000000000000   1.142194135689444
   0.800000000000000   1.179297284217600
   0.900000000000000   1.218705575564524
   1.000000000000000   1.259930265862033
```

约为 1.0033333333333333

4 阶龙格库塔方法

wuyudi_5_1.m RungKutta4.m ×

RungKutta4.m > ...

```matlab
function R = RungKutta4(f, a, b, N, ya)
    %f 是微分方程右端函数句柄
    %a,b 是自变量的取值区间 [a,b]的端点
    %N 是区间等分的个数
    %ya表初值 y(a)
    %R=[x',y']是自变量 X 和解 Y所组成的矩阵
    h = (b - a) / N;
  %  x = zeros(1, N + 1);
    y = zeros(1, N + 1);
    x = a:h:b;
    y(1) = ya;

    for i = 1:N
        k1 = feval(f, x(i), y(i));
        k2 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k1);
        k3 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k2);
        k4 = feval(f, x(i) + h, y(i) + h * k3);
        y(i + 1) = y(i) + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
    end

    R = [x', y'];
```

计算结果

wuyudi_5_1.m × | RungKutta4.m

wuyudi_5_1.m

```matlab
disp(RungKutta4(@(x, y) 2 * x / (3 * y^2), 0, 1, 10, 1))
```

MATLAB Command Window

```
  Toolbox Path Cache read in 0.03 seconds.
MATLAB Path initialized in 0.36 seconds.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

                   0   1.000000000000000
   0.100000000000000   1.003322292719565
   0.200000000000000   1.013159438200695
   0.300000000000000   1.029142535439115
   0.400000000000000   1.050717679021904
   0.500000000000000   1.077217479999272
   0.600000000000000   1.107931808368870
   0.700000000000000   1.142164929384162
   0.800000000000000   1.179273883780467
   0.900000000000000   1.218689083410464
   1.000000000000000   1.259921221582087
```

结果 1.003322292719565

四阶亚当姆斯方法

wuyudi_5_1.m | CAdams4PC.m | Adams4PC.m ×

Adams4PC.m > ...

```matlab
function A = Adams4PC(f, a, b, N, ya)
    %f 是微分方程右端函数句柄
    %a,b 是自变量的取值区间 [a,b]的端点
    %N 是区间等分的个数
    %ya表初值 y(a)
    %A=[x',y']是自变量 X 和解 Y 所组成的矩阵
    if N < 4
        return;
    end

    h = (b - a) / N;
    %  x = zeros(1, N + 1);
    y = zeros(1, N + 1);
    x = a:h:b;
    y(1) = ya;
    %  F = zeros(1, 4);

    for i = 1:N

        if i < 4%用四阶 Runge-Kutta 方法求初始解
            k1 = feval(f, x(i), y(i));
            k2 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k1);
            k3 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k2);
            k4 = feval(f, x(i) + h, y(i) + h * k3);
            y(i + 1) = y(i) + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
        else
            F = feval(f, x(i - 3:i), y(i - 3:i));
            py = y(i) + (h / 24) * (F * [-9, 37, -59, 55]'); %预报
            p = feval(f, x(i + 1), py);
            F = [F(2) F(3) F(4) p];
            y(i + 1) = y(i) + (h / 24) * (F * [1, -5, 19, 9]'); %校正
        end

    end

    A = [x', y'];
```

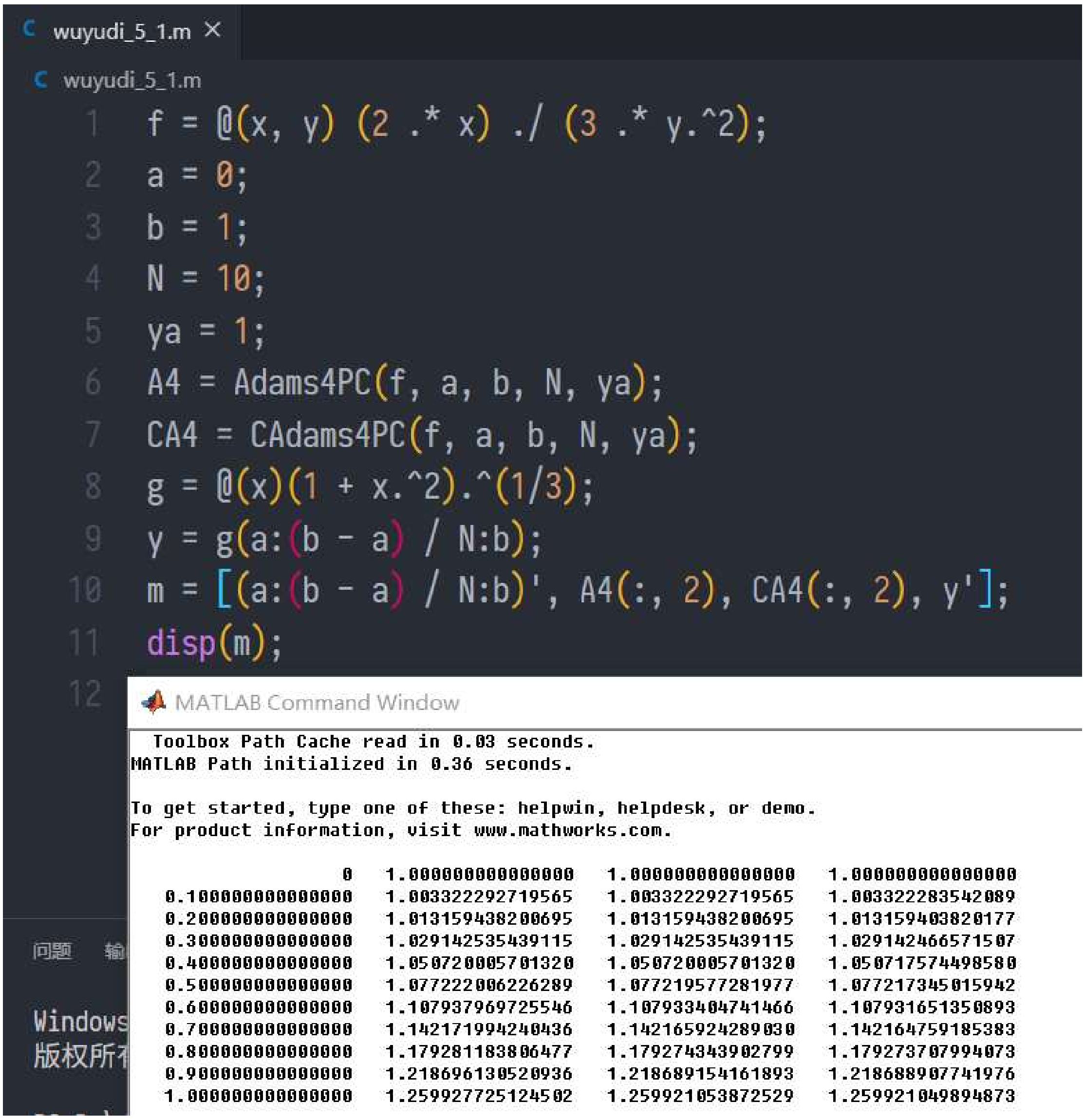
改进的四阶亚当姆斯预估校正系统如图。

wuyudi_5_1.m   CAdams4PC.m

CAdams4PC.m > ...

```matlab
function A = CAdams4PC(f, a, b, N, ya)
    %f 是微分方程右端函数句柄
    %a,b 是自变量的取值区间 [a,b]的端点
    %N 是区间等分的个数
    %ya表初值 y(a)
    %A=[x',y']是自变量 X 和解 Y 所组成的矩阵
    if N < 4
        return;
    end

    h = (b - a) / N;
    y = zeros(1, N + 1);
    x = a:h:b;
    y(1) = ya;

    for i = 1:N

        if i < 4%用四阶 Runge-Kutta 方法求初始解
            k1 = feval(f, x(i), y(i));
            k2 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k1);
            k3 = feval(f, x(i) + h / 2, y(i) + (h / 2) * k2);
            k4 = feval(f, x(i) + h, y(i) + h * k3);
            y(i + 1) = y(i) + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
        elseif i == 4
            F = feval(f, x(i - 3:i), y(i - 3:i));
            py = y(i) + (h / 24) * (F * [-9, 37, -59, 55]'); %预报
            p = feval(f, x(i + 1), py);
            F = [F(2) F(3) F(4) p];
            y(i + 1) = y(i) + (h / 24) * (F * [1, -5, 19, 9]'); %校正
            p = py; c = y(i + 1);
        else
            F = feval(f, x(i - 3:i), y(i - 3:i));
            py = y(i) + (h / 24) * (F * [-9, 37, -59, 55]'); %预报
            my = py - 251 * (p - c) / 270; %改进
            m = feval(f, x(i + 1), my);
            F = [F(2) F(3) F(4) m];
            cy = y(i) + (h / 24) * (F * [1, -5, 19, 9]'); %校正
            y(i + 1) = cy + 19 * (py - cy) / 270; %改进
            p = py; c = cy;
        end

    end

    A = [x', y'];
```

结果如图


wuyudi_5_1.m

```
f = @(x, y) (2 .* x) ./ (3 .* y.^2);
a = 0;
b = 1;
N = 10;
ya = 1;
A4 = Adams4PC(f, a, b, N, ya);
CA4 = CAdams4PC(f, a, b, N, ya);
g = @(x)(1 + x.^2).^(1/3);
y = g(a:(b - a) / N:b);
m = [(a:(b - a) / N:b)', A4(:, 2), CA4(:, 2), y'];
disp(m);
```

MATLAB Command Window

```
 Toolbox Path Cache read in 0.03 seconds.
MATLAB Path initialized in 0.36 seconds.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

                   0   1.000000000000000   1.000000000000000   1.000000000000000
   0.100000000000000   1.003322292719565   1.003322292719565   1.003322283542089
   0.200000000000000   1.013159438200695   1.013159438200695   1.013159403820177
   0.300000000000000   1.029142535439115   1.029142535439115   1.029142466571507
   0.400000000000000   1.050720005701320   1.050720005701320   1.050717574498580
   0.500000000000000   1.077222006226289   1.077219577281977   1.077217345015942
   0.600000000000000   1.107937969725546   1.107933404741466   1.107931651350893
   0.700000000000000   1.142171994240436   1.142165924289030   1.142164759185383
   0.800000000000000   1.179281183806477   1.179274343902799   1.179273707994073
   0.900000000000000   1.218696130520936   1.218689154161893   1.218688907741976
   1.000000000000000   1.259927725124502   1.259921053872529   1.259921049894873
```

第一列是 x 的值，第 2 列是四阶亚当姆斯方法、第 3 列是改进的四阶亚当姆斯预估校正系统。第 4 列是精确值。

四. 实验体会

通过常微分方程的差分方法实验，进一步对常微分方程求解的方法有了理解与感悟。可以更加熟练的针对不同的要求应用和设计出不同的算法来计算，并且对于应用 matlab 求解常微分方程有了认识，对于 matlab 的操作使用更加的准确纯熟。

对于常微分方程的各种算法的精度在此进行详细的分析。显式的 Euler 格式虽然很结构简单、

计算量小，但是它的精度很低；改进的 Euler 格式，相对于 Euler 格式，明显的改善了精度，并且计算量也是可取的。四阶 Runge-Kutta 格式具有更高的精度，但是计算量比较大。四阶 Adams 预报校正系统是在 Runge-Kutta 的基础上进行修改，改善了精度以及计算量。改进的四阶 Adams 预报矫正系统效果最好。