常微分方程数值解实验题

吴佳龙 2018013418

摘要

结合理论分析和编程计算，运用不同方法计算了一常微分方程初值问题的数值解，并与精确解比较。运用的方法分别为：古典四级四阶 Runge-Kutta 方法和隐式二级四阶 Runge-Kutta 方法（Gauss 方法）。

1 问题

求解：

$$\begin{cases} \frac{du}{dt} = -2000u(t) + 999.75v(t) + 1000.25 \\ \frac{dv}{dt} = u(t) - v(t) \end{cases}$$

初始条件为 $u(0) = 0, v(0) = -2$。其精确解为

$$\begin{cases} u(t) = -1.499875e^{-0.5t} + 0.499875e^{-2000.5t} + 1 \\ v(t) = -2.99975e^{-0.5t} - 0.00025e^{-2000.5t} + 1 \end{cases}$$

分别用古典四级四阶 Runge-Kutta 方法和隐式二级四阶 Runge-Kutta 方法计算，计算区间取成 $[0, 20]$，并与精确解比较。

2 古典四级四阶 Runge-Kutta 方法

2.1 算法原理

2.1.1 用 Taylor 展开构造高阶数值方法

取 $y(x+h) \approx y(x) + hy'(x)$ 得到单步法，即 1 阶 Euler 方法

$$y_{n+1} = y_n + hf(x_n, y_n)$$

取 $y(x+h) \approx y(x) + hy'(x) + \frac{1}{2}h^2y''(x)$ 得到 2 阶单步法

$$\begin{aligned} y_{n+1} = &\, y_n + hf(x_n, y_n) \\ &+ \frac{h^2}{2}\left[\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right](x_n, y_n) \end{aligned} \qquad (1)$$

如此构造下去，可得到三阶方法以及更高阶的方法，但是该类方法需要计算很多偏导数，并不实用。

2.1.2 Runge-Kutta 方法

Runge-Kutta 方法采用了不同点上函数值的不同组合来提高精度同时避免函数 f 的偏导数的计算。其一般形式为

$$y_{n+1} = y_n + h\varphi(x_n, y_n; h)$$

$$\varphi(x_n, y_n; h) = \sum_{i=1}^{s} b_i k_i$$

$$k_i = f\left(x_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right)$$

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, 2, \cdots, s$$

例如，在二级方法中，将 k_2 进行 Taylor 展开，并将 k_1, k_2 代入 φ，要求 φ 的前三项与公式 1 中的增量函数相等，即可解得 a, b, c（详见课本 P361）

2.1.3 古典四级四阶 Runge-Kutta 方法

将 RK 方法用 RK 表描述

$$\begin{array}{c|c} c & A \\ \hline & b^\top \end{array}$$

古典四级四阶 Runge-Kutta 方法对应的 RK 表为

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

具体计算格式

$$\begin{cases} y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ k_1 = f\left(x_n, y_n\right) \\ k_2 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2}hk_2\right) \\ k_4 = f\left(x_n + h, y_n + hk_3\right) \end{cases}$$

2.2 算法实现

古典四级四阶 Runge-Kutta 方法的 MATLAB 实现如下：

```
function y = myRungeKutta44(f, y0, a, b, h)
% 古典四级四阶 Runge−Kutta 方法
% 求解微分方程 dy/dx = f(x, y), x in [a,b]; y
    (a) = y0
% h 为步长
n = floor((b−a)/h);
x = a + h*(0:n);
y = y0;
yn = y0;
for i = 1:n
    xn = x(i);
    k1 = f(xn, yn);
    k2 = f(xn + h/2, yn + h/2*k1);
    k3 = f(xn + h/2, yn + h/2*k2);
    k4 = f(xn + h, yn + h*k3);
    yn1 = yn + h/6*(k1+2*k2+2*k3+k4);
    y = [y yn1]; yn = yn1;
end
end
```

3 隐式二级四阶 Runge-Kutta 方法（Gauss 方法）

3.1 算法原理

3.1.1 隐式二级四阶 Runge-Kutta 方法

隐式二级四阶 Runge-Kutta 方法对应的 RK 表为

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

具体计算格式

$$\begin{aligned} y_{n+1} &= y_n + h\left(\tfrac{1}{2}k_1 + \tfrac{1}{2}k_2\right) \\ k_1 &= f\left(x_n + \left(\tfrac{1}{2} - \tfrac{\sqrt{3}}{6}\right)h, y_n + \tfrac{1}{4}hk_1 + \left(\tfrac{1}{4} - \tfrac{\sqrt{3}}{6}\right)hk_2\right) \\ k_2 &= f\left(x_n + \left(\tfrac{1}{2} + \tfrac{\sqrt{3}}{6}\right)h, y_n + \left(\tfrac{1}{4} + \tfrac{\sqrt{3}}{6}\right)hk_1 + \tfrac{1}{4}hk_2\right) \end{aligned}$$

3.1.2 隐式方法的迭代计算

可用迭代的方法求解隐式方法，具体地，先给出 k_1, k_2 的近似值 $k_1^{(0)}, k_2^{(0)}$，然后用显式迭代

$$\begin{aligned} k_1^{(s+1)} &= f\left(x_n + \left(\tfrac{1}{2} - \tfrac{\sqrt{3}}{6}\right)h, y_n + \tfrac{1}{4}hk_1^{(s)} + \left(\tfrac{1}{4} - \tfrac{\sqrt{3}}{6}\right)hk_2^{(s)}\right) \\ k_2^{(s+1)} &= f\left(x_n + \left(\tfrac{1}{2} + \tfrac{\sqrt{3}}{6}\right)h, y_n + \left(\tfrac{1}{4} + \tfrac{\sqrt{3}}{6}\right)hk_1^{(s)} + \tfrac{1}{4}hk_2^{(s)}\right) \end{aligned}$$

直至 $||k_1^{(s)} - k_1^{(s+1)}|| < \varepsilon, ||k_2^{(s)} - k_2^{(s+1)}|| < \varepsilon$

该方法可由 Gauss 求积公式导出，因此也称 Gauss 方法。

3.2 算法实现

隐式二级四阶 Runge-Kutta 方法的 MATLAB 实现如下：

```
function y = myRungeKutta24(f, y0, a, b, h,
    eps)
% 隐式二级四阶 Runge−Kutta 方法
% 求解微分方程 dy/dx = f(x, y), x in [a,b]; y
    (a) = y0
% h 为步长
c1 = 1/2−sqrt(3)/6; c2 = 1/2+sqrt(3)/6;
b1 = 1/2; b2 = 1/2;
a11 = 1/4; a12 = 1/4−sqrt(3)/6;
a21 = 1/4+sqrt(3)/6; a22 = 1/4;

n = floor((b−a)/h);
x = a + h*(0:n);
y = y0;
yn = y0;
for i = 1:n
    xn = x(i);
    k1 = f(xn, yn); k2 = k1;
    while true % 迭代
        new_k1 = f(xn+c1*h, yn + a11*h*k1
            + a12*h*k2);
        new_k2 = f(xn+c2*h, yn + a21*h*k1
            + a22*h*k2);
```

```
        if (max_error(k1, new_k1)< eps &&
                max_error(k2, new_k2)< eps)
            break
        end
        k1 = new_k1; k2 = new_k2;
    end
    yn1 = yn + h*(b1*k1+b2*k2);
    y = [y yn1]; yn = yn1;
end
end
```

4 计算结果与方法比较

4.1 误差

选取步长 $h = 0.001$ ，隐式二级四阶 RK 方法中的 $eps = 10^{-7}$，以上两种方法的计算结果和误差见表 1 和表 2。

可以看到两种方法都能得到较精确的数值解，但精度有差异，隐式二级四阶 RK 方法比古典四级四阶 RK 方法精度好。

4.2 方法的阶和步长的影响

以上实现的两种方法都是 4 阶方法，取隐式二级四阶 RK 方法中的 $eps = 10^{-12}$，修改不同的步长 h 得到计算结果如表 3。

其中平均误差定义为 $\mathrm{mean}\{|y_n - y(x_n)|\}, n = 0, 1, \cdots$， 最大误差定义为 $\max\{|y_n - y(x_n)|\}, n = 0, 1, \cdots$

可以看到, 随着 h 减少 1 个数量级，误差的大小减少约 4 个数量级，这符合方法是 4 阶的。

5 总结

本次实验对古典四级四阶 RK 方法和隐式二级四阶 RK 方法进行了理论分析和编程计算，得到了一常微分方程初值问题的数值解，并比较了他们的计算误差。

本次实验还探究了步长 h 对误差的影响，结果符合预期，与两种方法为四阶方法的事实相符。

Table 1: 古典四级四阶 Runge-Kutta 方法的计算结果和误差

x_n	$y(x_n)$	y_n (RK44)	$y(x_n) - y_n$ (误差)
5	(-0.4967, -1.9938)	(-0.4907, -1.9938)	$(6.0163 \times 10^{-3}, -3.0089 \times 10^{-6})$
10	(-0.4931, -1.9863)	(-0.4931, -1.9863)	$(2.5503 \times 10^{-5}, -1.2755 \times 10^{-8})$
15	(-0.4894, -1.9788)	(-0.4894, -1.9788)	$(1.0525 \times 10^{-7}, -5.2636 \times 10^{-11})$
20	(-0.4857, -1.9714)	(-0.4857, -1.9714)	$(4.3420 \times 10^{-10}, -2.1672 \times 10^{-13})$

Table 2: 隐式二级四阶 Runge-Kutta 方法的计算结果和误差

x_n	$y(x_n)$	y_n (RK24)	$y(x_n) - y_n$ (误差)
5	(-0.4967, -1.9938)	(-0.4967, -1.9938)	$(4.0484 \times 10^{-5}, -2.0247 \times 10^{-8})$
10	(-0.4931, -1.9863)	(-0.4931, -1.9863)	$(4.7797 \times 10^{-9}, -2.3901 \times 10^{-12})$
15	(-0.4894, -1.9788)	(-0.4894, -1.9788)	$(3.1969 \times 10^{-12}, 1.3989 \times 10^{-14})$
20	(-0.4857, -1.9714)	(-0.4857, -1.9714)	$(1.2546 \times 10^{-14}, 5.4401 \times 10^{-14})$

Table 3: 步长 h 对误差的影响

h	平均误差（RK44）	最大误差（RK44）	平均误差（RK24）	最大误差（RK24）
10^{-3}	4.300212×10^{-6}	9.909147×10^{-2}	1.367054×10^{-7}	3.763211×10^{-3}
10^{-4}	9.826336×10^{-11}	2.900773×10^{-6}	1.395697×10^{-11}	4.100364×10^{-7}
10^{-5}	1.551279×10^{-14}	2.495640×10^{-10}	8.662294×10^{-14}	4.090728×10^{-11}