计算方法数值实验 2-实验报告

221900180 田永铭

2024 年 5 月 18 日

一、实验题目

利用 matlab 实现 4 阶 Adams 预测-校正方法的 PECE 模式算法求常微分方程初值问题:

$$\begin{cases} y' = \dfrac{4x^2 - 2xy}{1 + x^2}, \quad 0 \le x \le 5 \\ y(0) = 2 \end{cases}$$

在 $x_i = ih, i = 1, 2, \cdots, 50 (h = 0.1)$ 点上的 $y(x_i)$ 的近似值 y_i, 保留 6 位小数. 用经典的 4 阶 Runge-Kutta 方法提供初始出发值 y_1, y_2, y_3.
输出：列表输出 $i, x_i, y_i, y(x_i), y(x_i) - y_i$, 输出结果要清晰.

二、实验原理

- **实验原理——4 阶 Runge-Kutta：** 4 阶 Runge-Kutta 方法是利用泰勒级数推导出的高精度的单步法，其中经典格式用的最多，可以用它求解常微分方程的初值问题。它的表达形式如下：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_1) \\ K_3 = f(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_2) \\ K_4 = f(x_n + h, y_n + hK_3) \end{cases}$$

- **实验原理——Adams—PECE:** 基于数值积分，可以构造出 Adams 线性多步法，通过泰勒公式来分析误差，Adams 显式格式和隐式格式能匹配四阶成 Adams 预测-校正

(PECE) 系统，它可以利用上述 4 阶 Runge-Kutta 方法来提供初值，然后根据以下公式来进行求解常微分方程的初值问题:

$$\begin{cases} P : \bar{y}_{n+1} = y_n + \dfrac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \\ E : \bar{f}_{n+1} = f(x_{n+1}, \bar{y}_{n+1}) \\ C : y_{n+1} = y_n + \dfrac{h}{24}(9\bar{f}_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}) \\ E : f_{n+1} = f(x_{n+1}, y_{n+1}) \end{cases}$$

三、核心代码以及说明

3.1定义常微分方程函数和真实数值解函数

```
f = @(x,y) ( (4*x.^2-2*x.*y)/(1+x.^2) );
exact_solution = @(x) (2+ (4/3)*x.^3)/(1+x.^2);
y = double(zeros(1,100));y(1) = 2;x = double(zeros(1,100));
h = 0.1;
for n = 1:51
        x(n) = 0.1*(n-1);
end
```

这部分定义了常微分方程函数和真实数值解函数，并在区间以步长为 1 取从 0 到 50 的点，注意 matlab 下表是从 1 开始的。

3.2实现经典四阶 Runge_Kutta 方法

```
% 利用经典四阶Runge_Kutta方法预测y2,y3,y4的函数
function [y2,y3,y4] = classical_Runge_Kutta(f,x,y,h)
        for n = 1:3
                K1 = f(x(n),y(n));
                K2 = f(x(n)+h/2,y(n)+h/2*K1);
                K3 = f(x(n)+h/2,y(n)+h/2*K2);
                K4 = f(x(n)+h,y(n)+h*K3);
                y(n+1) = y(n) + h/6*(K1 + 2*K2 + 2*K3 + K4);
        end
        y2 = y(2);y3 = y(3);y4 = y(4);
```

```
end
```

由此我们可以求出初值 y_2, y_3, y_4, 而 y_1 是 2 已经有了。由这四个值我们就可以开始使用 PECE 系统了。

3.3实现 PECE 系统

```
% 省去利用经典四阶Runge_Kutta方法预测y2,y3,y4的代码, 上面已经展示
y_bar = double(zeros(1,100));
f_bar = double(zeros(1,100));
for n = 4:50
    %预测
    y_bar(n+1) = y(n) + h/24*(55*frac(n)-59*frac(n-1) ...
         +37*frac(n-2)-9*frac(n-3));
    f_bar(n+1) = f(x(n+1),y_bar(n+1));
    %校正
    y(n+1) = y(n) + h/24*(9*f_bar(n+1)+19*frac(n) ...
         -5*frac(n-1)+frac(n-2));
    frac(n+1) = f(x(n+1),y(n+1));
end
```

简单代入公式即可实现。

3.4优化 PECE 系统的实现

注意到，上面 PECE 系统的实现开了很多的变量空间，这样十分浪费空间。而老师上课所讲的是可以只用 4 个存储空间来实现类似的代码，具体做法是：只记录 y_1, y_2, y_3, y_4 四个值，在一步更新后用后一个值写入前一个，将结果写入最后的 y_4。同时，y_bar 和 f_bar 变量其实也只需要一个空间，我之前完全浪费了空间。优化后的实现如下：

```
% 好的实现, frac这个存储预测值的地方只需要4个空间
frac = double(zeros(1,4));
% 好的PECE实现, 省空间
for n = 4:50
        % 预测
        y_bar = y(n) + h/24*(55*frac(4)-59*frac(3) ...
                +37*frac(2)-9*frac(1));
        f_bar = f(x(n+1),y_bar);
```

```
        %校正
        y(n+1) = y(n) + h/24*(9*f_bar+19*frac(4) ...
                -5*frac(3)+frac(2));
        frac(1) = frac(2);frac(2) = frac(3);frac(3) = frac(4);
        frac(4) = f(x(n+1),y(n+1));
end
```

3.5处理结果输出

我既格式化输出了题目要求的各种结果，又画了一张图来展示预测值和真实值的相似度。具体代码如下:

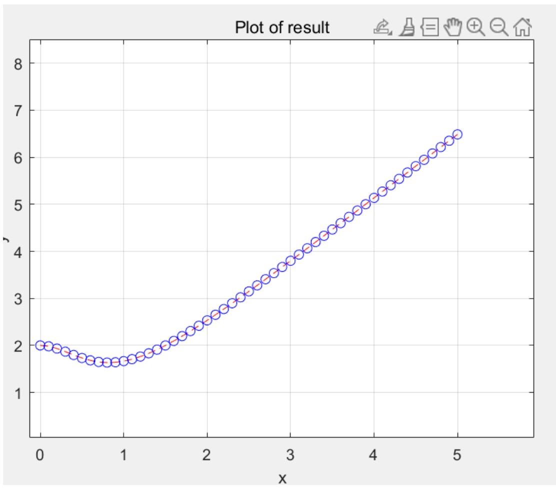
```
% 格式化输出结果,结果通过命令行窗口查看，空格符号显示了出来
fprintf('␣i␣␣␣xi␣␣␣␣yi␣␣␣␣␣␣y(xi)␣␣␣␣␣␣y(xi)␣-␣yi\n');
fprintf('-------------------------------------------\n');
for i = 1:n+1
        yi = y(i);
        xi = x(i);
        exact_yi = exact_solution(xi);
        fprintf('%2d␣␣%2.1f␣␣%7.6f␣␣%7.6f␣␣%.10e\n'
                , i-1, xi, yi, exact_yi, exact_yi - yi);
end
% 绘制图象，准确值为红色虚线，预测值为蓝色圆点
xx = linspace(0, 5, 100);  % 在 0 到 5 范围内生成 100 个点
yy = double(zeros(1,100));
for i = 1 : 100
        yy(i) = exact_solution(xx(i));
end
plot(xx, yy,'r--');
title('Plot␣of␣result');
xlabel('x');
ylabel('y');
grid on;
hold on;
plot(x, y, 'bo');
hold off;
```

四、实验结果

首先是数据结果的展示，图片如下：

i	xi	yi	y(xi)	y(xi) - yi
0	0.0	2.000000	2.000000	0.0000000000e+00
1	0.1	1.981518	1.981518	7.7992388769e-08
2	0.2	1.933333	1.933333	2.3490647427e-07
3	0.3	1.867890	1.867890	3.5165366130e-07
4	0.4	1.797630	1.797701	7.1591273496e-05
5	0.5	1.733229	1.733333	1.0452445611e-04
6	0.6	1.682253	1.682353	9.9989504458e-05
7	0.7	1.649145	1.649217	7.2094973141e-05
8	0.8	1.635735	1.635772	3.7003873369e-05
9	0.9	1.641983	1.641989	5.8314011002e-06
10	1.0	1.666683	1.666667	-1.6697407185e-05
11	1.1	1.708024	1.707994	-3.0319783097e-05
12	1.2	1.763971	1.763934	-3.6848369039e-05
13	1.3	1.832504	1.832466	-3.8544040680e-05
14	1.4	1.911749	1.911712	-3.7343140891e-05
15	1.5	2.000035	2.000000	-3.4639881345e-05
16	1.6	2.095911	2.095880	-3.1330282128e-05
17	1.7	2.198143	2.198115	-2.7937799627e-05
18	1.8	2.305685	2.305660	-2.4737658471e-05
19	1.9	2.417665	2.417643	-2.1852733342e-05
20	2.0	2.533353	2.533333	-1.9318931374e-05
21	2.1	2.652143	2.652126	-1.7126413338e-05
22	2.2	2.773531	2.773516	-1.5244060910e-05
23	2.3	2.897099	2.897085	-1.3633201882e-05
24	2.4	3.022497	3.022485	-1.2254816367e-05
25	2.5	3.149436	3.149425	-1.1072962808e-05
26	2.6	3.277673	3.277663	-1.0056109143e-05
27	2.7	3.407006	3.406996	-9.1773656594e-06
28	2.8	3.537263	3.537255	-8.4141883279e-06
29	2.9	3.668304	3.668296	-7.7478652236e-06
30	3.0	3.800007	3.800000	-7.1629499092e-06
31	3.1	3.932272	3.932265	-6.6467215767e-06
32	3.2	4.065012	4.065006	-6.1887058296e-06
33	3.3	4.198155	4.198150	-5.7802658695e-06
34	3.4	4.331640	4.331635	-5.4142620165e-06
35	3.5	4.465414	4.465409	-5.0847723463e-06
36	3.6	4.599432	4.599427	-4.7868655786e-06
37	3.7	4.733656	4.733651	-4.5164174551e-06
38	3.8	4.868053	4.868048	-4.2699626759e-06
39	3.9	5.002595	5.002591	-4.0445756433e-06
40	4.0	5.137259	5.137255	-3.8377743499e-06
41	4.1	5.272023	5.272019	-3.6474428065e-06
42	4.2	5.406870	5.406867	-3.4717683111e-06
43	4.3	5.541785	5.541782	-3.3091905598e-06
44	4.4	5.676755	5.676752	-3.1583602267e-06
45	4.5	5.811768	5.811765	-3.0181051596e-06
46	4.6	5.946814	5.946811	-2.8874026432e-06
47	4.7	6.081885	6.081882	-2.7653565970e-06
48	4.8	6.216974	6.216972	-2.6511787032e-06
49	4.9	6.352075	6.352073	-2.5441727818e-06
50	5.0	6.487182	6.487179	-2.4437217307e-06

然后是图象的展示，我的代码生成的图表中红色虚线为真实值，蓝色圆点为预测值。图象如下：



分析数据结果，可以知道，在保留 6 位小数的前提下，准确值和估计值的误差大概在 1e-6 左右，这已经是非常好的结果，这进一步说明了 PECE 方法的有效性。同时，我还发现，在计算该实验的常微分方程的表现上，PECE 表现很好，随着点从 0 向 5 靠近，最后的精度甚至在提升，而一般的情况应该是精度越往后越差。当然这不是必然，因为我还验证了其他的函数，PECE 也会出现越往后精度越差的情况。不过这个实验足以说明 PECE 方法的有效性。

分析图表结果，可以观察到蓝色圆点几乎就正好落在红色虚线上，这说明 PECE 方法的预测值和真实值非常逼近。这也进一步验证了 Adam-PECE 方法的有效性。

五、总结

1. 此次实验我全部采用 matlab 实现，全程独立实现无参考。我主动选择了使用 matlab 这个之前没怎么使用过的工具，了解了它的基础使用方法，并利用它完成了数值实验。正如老师所说，这是一款数学上的很好的工具，我为掌握它的基础用法而开心。
2. 通过这次实验，我对 Adams-PECE 方法的理解更深了一步，理解了算法的原理、预测精确度、注意点和操作细节等等。
3. 通过这次实验，我理论联系实际，将书本知识应用到程序上，激发了对计算方法这门课程的极大的兴趣。同时也明白了，数值实验是有误差的，是“差不多”的，但是做实验必须是仔细和精确到每一个细节的，例如我在优化 Adams-PECE 的时候，就是受到老师课堂的启发，主动降低了实验所需存储空间的开销，使得算法更加完美。
4. 我还将实验结果进一步延展，利用我实现的方法来做书上一直在做的一个问题：

$$\begin{cases} y' = y - \dfrac{2x}{y}, \quad 0 < x < 1 \\ y(0) = 1 \end{cases}$$

惊喜的是，得出了与书上 PECE 计算表格几乎完全一致的结果，这个结果也显然优于简单的 Euler 方法等方法。这进一步验证了这次实验的成功。