《数值分析》之

常微分方程数值方法

徐岩

中国科学技术大学数学系

yxu@ustc.edu.cn

http://staff.ustc.edu.cn/~yxu/

高精度格式-Taylor级数方法

方法的要点是$y(x)$的Taylor级数展开：

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + \frac{h^4}{4!}y^{(4)}(x) + \cdots$$

因此对于固定的x和h，为了计算出$y(x+h)$的值，我们只需要知道在x点$y(x)$的各阶导数值

$$\begin{aligned} y'(x) &= f(x,y) \\ y''(x) &= f_x(x,y) + f_y(x,y)y'(x) \\ y'''(x) &= \cdots \end{aligned}$$

所以，可以构造格式

$$\begin{aligned} y(x_{n+1}) = y(x_n) &+ hf(x_n, y(x_n)) \\ &+ \frac{h^2}{2}\left(f_x(x_n, y(x_n)) + f_y(x_n, y(x_n))f(x_n, y(x_n))\right) \end{aligned}$$

为了应用Taylor级数方法，我们需要假定f的各阶偏导数存在，例如

$$\begin{cases} y' = \cos x - \sin y + x^2 \\ y(-1) = 3 \end{cases}$$

- 这些导数值可以从给定的微分方程和初值条件中得到：

$$\begin{aligned} y' &= \cos x - \sin y + x^2 \qquad \text{(已知条件)} \\ y'' &= -\sin x - y' \cos y + 2x \\ y''' &= -\cos x - y'' \cos y + (y')^2 \sin y + 2 \\ y^{(4)} &= \sin x - y''' \cos y + 3y'y'' \sin y + (y')^3 \cos y \end{aligned}$$

  我们当然还可以继续下去。如果我们决定仅应用Taylor展开中到h^4之前的项，那么其它项共同构成方法的截断误差，所对应的方法称为四阶方法
- 注意求导中要应用$d \sin y(x)/dx$的链式法则
- 当然可以执行各种代换，使得右边不出现y的导数y', y'', y''', ...。但如果是按上面给出的次序应用这些公式的话，就不必进行这种代换
- 运行Mathematica程序ode_taylor.nb

In[19]:=
```
M = 200; h = 0.01; t = -1.0; x = 3.0; sol = Table[{t, x}, {n, 0, M}];
For[k = 1, k <= M, k++, x1 = Cos[t] - Sin[x] + t^2;
 x2 = -Sin[t] - x1 * Cos[x] + 2 * t; x3 = -Cos[t] - x2 * Cos[x] + (x1^2) * Sin[x] + 2;
 x4 = Sin[t] + ((x1^3) - x3) * Cos[x] + 3 * x1 * x2 * Sin[x];
 x = x + h * (x1 + h / 2 * (x2 + h / 3 * (x3 + h / 4 * x4))); t = t + h;
 Print[k, "\t", t, "\t", x]; sol[[k]] = {t, x}]
```

```
1	-0.99	3.014
2	-0.98	3.02803
3	-0.97	3.04209
4	-0.96	3.05617
5	-0.95	3.07028
6	-0.94	3.08443
7	-0.93	3.09861
8	-0.92	3.11282
9	-0.91	3.12708
10	-0.9	3.14137
11	-0.89	3.15571
12	-0.88	3.17008
13	-0.87	3.18451
14	-0.86	3.19898
15	-0.85	3.2135
16	-0.84	3.22807
17	-0.83	3.24269
18	-0.82	3.25736
19	-0.81	3.27209
20	-0.8	3.28687
21	-0.79	3.30172
22	-0.78	3.31662
23	-0.77	3.33159
24	-0.76	3.34662
25	-0.75	3.36171
26	-0.74	3.37687
27	-0.73	3.39209
28	-0.72	3.40739
29	-0.71	3.42275
30	-0.7	3.43819
31	-0.69	3.45369
32	-0.68	3.46928
33	-0.67	3.48493
```

34	−0.66	3.50066
35	−0.65	3.51647
36	−0.64	3.53236
37	−0.63	3.54832
38	−0.62	3.56437
39	−0.61	3.58049
40	−0.6	3.5967
41	−0.59	3.61299
42	−0.58	3.62937
43	−0.57	3.64582
44	−0.56	3.66237
45	−0.55	3.67899
46	−0.54	3.6957
47	−0.53	3.7125
48	−0.52	3.72939
49	−0.51	3.74636
50	−0.5	3.76341
51	−0.49	3.78056
52	−0.48	3.79779
53	−0.47	3.81511
54	−0.46	3.83251
55	−0.45	3.85001
56	−0.44	3.86759
57	−0.43	3.88525
58	−0.42	3.903
59	−0.41	3.92084
60	−0.4	3.93876
61	−0.39	3.95677
62	−0.38	3.97486
63	−0.37	3.99303
64	−0.36	4.01129
65	−0.35	4.02962
66	−0.34	4.04804
67	−0.33	4.06654
68	−0.32	4.08511
69	−0.31	4.10376
70	−0.3	4.12249

71	−0.29	4.14129
72	−0.28	4.16016
73	−0.27	4.1791
74	−0.26	4.19812
75	−0.25	4.2172
76	−0.24	4.23634
77	−0.23	4.25555
78	−0.22	4.27482
79	−0.21	4.29415
80	−0.2	4.31354
81	−0.19	4.33298
82	−0.18	4.35248
83	−0.17	4.37203
84	−0.16	4.39162
85	−0.15	4.41126
86	−0.14	4.43095
87	−0.13	4.45067
88	−0.12	4.47043
89	−0.11	4.49023
90	−0.1	4.51006
91	−0.09	4.52992
92	−0.08	4.54981
93	−0.07	4.56972
94	−0.06	4.58965
95	−0.05	4.60961
96	−0.04	4.62957
97	−0.03	4.64955
98	−0.02	4.66954
99	−0.01	4.68954
100	7.5287×10^{-16}	4.70954
101	0.01	4.72954
102	0.02	4.74953
103	0.03	4.76952
104	0.04	4.78951
105	0.05	4.80948
106	0.06	4.82944
107	0.07	4.84938

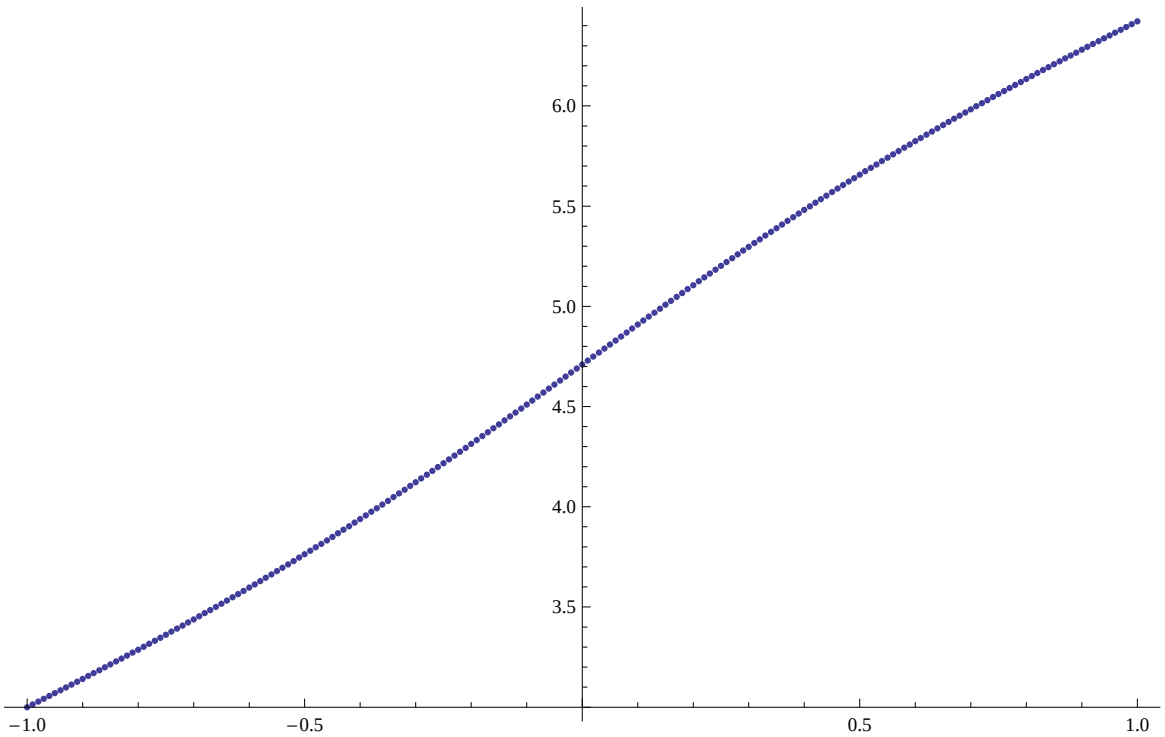
108	0.08	4.8693
109	0.09	4.8892
110	0.1	4.90907
111	0.11	4.92891
112	0.12	4.94872
113	0.13	4.9685
114	0.14	4.98824
115	0.15	5.00794
116	0.16	5.02759
117	0.17	5.04721
118	0.18	5.06677
119	0.19	5.08629
120	0.2	5.10575
121	0.21	5.12516
122	0.22	5.14451
123	0.23	5.16381
124	0.24	5.18304
125	0.25	5.20221
126	0.26	5.22132
127	0.27	5.24036
128	0.28	5.25933
129	0.29	5.27823
130	0.3	5.29706
131	0.31	5.31581
132	0.32	5.33449
133	0.33	5.3531
134	0.34	5.37162
135	0.35	5.39007
136	0.36	5.40844
137	0.37	5.42673
138	0.38	5.44494
139	0.39	5.46306
140	0.4	5.4811
141	0.41	5.49906
142	0.42	5.51693
143	0.43	5.53472
144	0.44	5.55242

145	0.45	5.57003
146	0.46	5.58756
147	0.47	5.605
148	0.48	5.62236
149	0.49	5.63962
150	0.5	5.65681
151	0.51	5.6739
152	0.52	5.69091
153	0.53	5.70783
154	0.54	5.72467
155	0.55	5.74142
156	0.56	5.75808
157	0.57	5.77466
158	0.58	5.79115
159	0.59	5.80756
160	0.6	5.82389
161	0.61	5.84014
162	0.62	5.8563
163	0.63	5.87238
164	0.64	5.88839
165	0.65	5.90431
166	0.66	5.92016
167	0.67	5.93593
168	0.68	5.95162
169	0.69	5.96724
170	0.7	5.98278
171	0.71	5.99825
172	0.72	6.01365
173	0.73	6.02898
174	0.74	6.04424
175	0.75	6.05944
176	0.76	6.07456
177	0.77	6.08963
178	0.78	6.10463
179	0.79	6.11957
180	0.8	6.13445
181	0.81	6.14927

182	0.82	6.16403
183	0.83	6.17874
184	0.84	6.19339
185	0.85	6.20799
186	0.86	6.22254
187	0.87	6.23704
188	0.88	6.2515
189	0.89	6.26591
190	0.9	6.28028
191	0.91	6.2946
192	0.92	6.30889
193	0.93	6.32313
194	0.94	6.33734
195	0.95	6.35152
196	0.96	6.36566
197	0.97	6.37977
198	0.98	6.39386
199	0.99	6.40791
200	1.	6.42194

In[21]:= **ListPlot[sol]**

Out[21]=



In[27]:=
```
M = 200; h = -0.01; t = 1.0; x = 6.42194; sol = Table[{t, x}, {n, 0, M}];
For[k = 1, k <= M, k++, x1 = Cos[t] - Sin[x] + t^2;
 x2 = -Sin[t] - x1 * Cos[x] + 2 * t; x3 = -Cos[t] - x2 * Cos[x] + (x1^2) * Sin[x] + 2;
 x4 = Sin[t] + ((x1^3) - x3) * Cos[x] + 3 * x1 * x2 * Sin[x];
 x = x + h * (x1 + h / 2 * (x2 + h / 3 * (x3 + h / 4 * x4)));  t = t + h;
 Print[k, "\t", t, "\t", x]; sol[[k]] = {t, x}]
```

```
1    0.99    6.40791
2    0.98    6.39385
3    0.97    6.37977
4    0.96    6.36566
5    0.95    6.35151
6    0.94    6.33734
7    0.93    6.32313
8    0.92    6.30888
9    0.91    6.2946
10    0.9    6.28027
11    0.89    6.2659
12    0.88    6.25149
13    0.87    6.23704
14    0.86    6.22254
15    0.85    6.20799
16    0.84    6.19338
17    0.83    6.17873
18    0.82    6.16402
19    0.81    6.14926
20    0.8    6.13444
21    0.79    6.11956
22    0.78    6.10462
23    0.77    6.08962
24    0.76    6.07456
25    0.75    6.05943
26    0.74    6.04424
27    0.73    6.02897
28    0.72    6.01364
29    0.71    5.99824
30    0.7    5.98277
31    0.69    5.96723
32    0.68    5.95161
33    0.67    5.93592
```

34	0.66	5.92015
35	0.65	5.9043
36	0.64	5.88838
37	0.63	5.87238
38	0.62	5.85629
39	0.61	5.84013
40	0.6	5.82388
41	0.59	5.80756
42	0.58	5.79115
43	0.57	5.77465
44	0.56	5.75807
45	0.55	5.74141
46	0.54	5.72466
47	0.53	5.70782
48	0.52	5.6909
49	0.51	5.67389
50	0.5	5.6568
51	0.49	5.63962
52	0.48	5.62235
53	0.47	5.60499
54	0.46	5.58755
55	0.45	5.57002
56	0.44	5.55241
57	0.43	5.53471
58	0.42	5.51692
59	0.41	5.49905
60	0.4	5.48109
61	0.39	5.46305
62	0.38	5.44493
63	0.37	5.42672
64	0.36	5.40843
65	0.35	5.39006
66	0.34	5.37161
67	0.33	5.35309
68	0.32	5.33448
69	0.31	5.3158
70	0.3	5.29705

71	0.29	5.27822
72	0.28	5.25932
73	0.27	5.24035
74	0.26	5.22131
75	0.25	5.2022
76	0.24	5.18303
77	0.23	5.1638
78	0.22	5.1445
79	0.21	5.12515
80	0.2	5.10574
81	0.19	5.08628
82	0.18	5.06676
83	0.17	5.0472
84	0.16	5.02758
85	0.15	5.00793
86	0.14	4.98823
87	0.13	4.96849
88	0.12	4.94871
89	0.11	4.9289
90	0.1	4.90906
91	0.09	4.88919
92	0.08	4.86929
93	0.07	4.84937
94	0.06	4.82943
95	0.05	4.80947
96	0.04	4.7895
97	0.03	4.76951
98	0.02	4.74952
99	0.01	4.72953
100	-7.5287×10^{-16}	4.70953
101	−0.01	4.68953
102	−0.02	4.66953
103	−0.03	4.64954
104	−0.04	4.62956
105	−0.05	4.6096
106	−0.06	4.58964
107	−0.07	4.56971

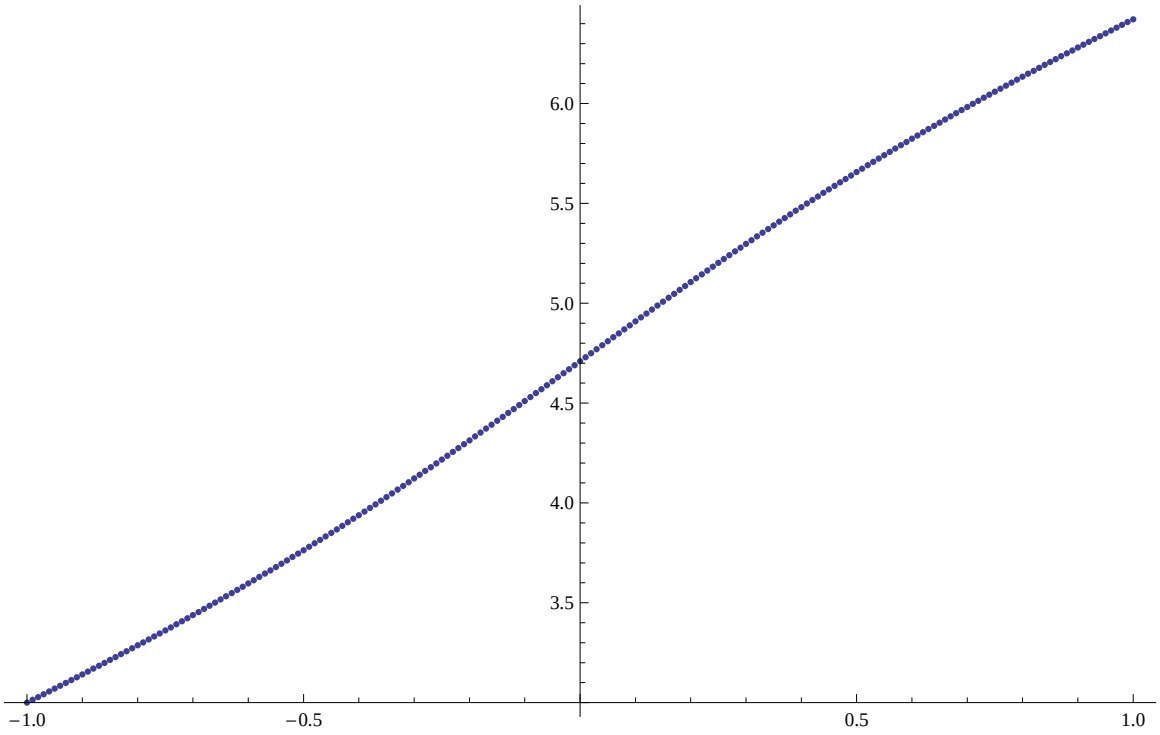
108	−0.08	4.5498
109	−0.09	4.52991
110	−0.1	4.51005
111	−0.11	4.49022
112	−0.12	4.47042
113	−0.13	4.45066
114	−0.14	4.43094
115	−0.15	4.41125
116	−0.16	4.39161
117	−0.17	4.37202
118	−0.18	4.35247
119	−0.19	4.33297
120	−0.2	4.31353
121	−0.21	4.29414
122	−0.22	4.27481
123	−0.23	4.25554
124	−0.24	4.23633
125	−0.25	4.21719
126	−0.26	4.19811
127	−0.27	4.1791
128	−0.28	4.16015
129	−0.29	4.14128
130	−0.3	4.12248
131	−0.31	4.10375
132	−0.32	4.0851
133	−0.33	4.06653
134	−0.34	4.04803
135	−0.35	4.02962
136	−0.36	4.01128
137	−0.37	3.99302
138	−0.38	3.97485
139	−0.39	3.95676
140	−0.4	3.93875
141	−0.41	3.92083
142	−0.42	3.90299
143	−0.43	3.88524
144	−0.44	3.86758

145	−0.45	3.85
146	−0.46	3.83251
147	−0.47	3.8151
148	−0.48	3.79778
149	−0.49	3.78055
150	−0.5	3.76341
151	−0.51	3.74635
152	−0.52	3.72938
153	−0.53	3.71249
154	−0.54	3.6957
155	−0.55	3.67898
156	−0.56	3.66236
157	−0.57	3.64582
158	−0.58	3.62936
159	−0.59	3.61299
160	−0.6	3.5967
161	−0.61	3.58049
162	−0.62	3.56436
163	−0.63	3.54832
164	−0.64	3.53235
165	−0.65	3.51646
166	−0.66	3.50066
167	−0.67	3.48492
168	−0.68	3.46927
169	−0.69	3.45369
170	−0.7	3.43818
171	−0.71	3.42274
172	−0.72	3.40738
173	−0.73	3.39209
174	−0.74	3.37686
175	−0.75	3.3617
176	−0.76	3.34661
177	−0.77	3.33158
178	−0.78	3.31662
179	−0.79	3.30171
180	−0.8	3.28687
181	−0.81	3.27208

182	−0.82	3.25735
183	−0.83	3.24268
184	−0.84	3.22806
185	−0.85	3.21349
186	−0.86	3.19897
187	−0.87	3.1845
188	−0.88	3.17008
189	−0.89	3.1557
190	−0.9	3.14137
191	−0.91	3.12707
192	−0.92	3.11282
193	−0.93	3.0986
194	−0.94	3.08442
195	−0.95	3.07028
196	−0.96	3.05616
197	−0.97	3.04208
198	−0.98	3.02803
199	−0.99	3.014
200	−1.	3.

In[24]:=
```
ListPlot[sol]
```

Out[24]=



局部截断误差的累加

- 在上面的算法的每一步中，因为不包含Taylor级数中涉及$h^5, h^6, \ldots$的项，所以局部截断误差是$\mathcal{O}(h^5)$
- 因此当$h \to 0$时，局部截断误差类似于Ch^5。但我们并不知道C是多大
- 不过此例中$h = 0.01$, 因此$h^5 = 10^{-10}$, 每一步中的误差粗略地具有10^{-10}的量级，因此几百步后这此小的误差累加起来，可能不太会损坏精度
- 另外，在每一步中，$y(x_k)$的估计值y_k中已包含误差，进一步地计算继续增加这些误差，因此在得到的数值解中，不要盲目地采用所有的数字

解的验证

- 因此我们需要给出一种方法，来确定最终解的有效数字到底是多少？
- 在此例中我们有$y_{200} = 6.42194$. 以这个值作为同样方程的初值，并且取$h = -0.01$, 重复前面的求解过程，得到$x = -1.0$时解为3.00000, 它与原来的初值几乎相同，因此我们可以认为原来的解具有六位精度

误差的数值估计

- 在n阶方法中，Taylor级数展开到h^n项，那么有如下的误差估计

$$E_n = \frac{1}{(n+1)!} h^{n+1} y^{(n+1)}(x+\theta h), \qquad 0 < \theta < 1$$

- 因此可以用简单的有限差分逼近估计这个误差。例如，对上前例，$n = 4$, $h = 0.01$, 那么

$$E_4 \approx \frac{1}{5!} h^5 \frac{y^{(4)}(x+h) - y^{(4)}(x)}{h} = \frac{h^4}{120}[y^{(4)}(x+h) - y^{(4)}(x)]$$

Taylor级数方法

优点

- 方法概念简单，并且具有高精度的潜力。如果能很容易地得到$y(x)$的20阶导数，则没有什么能阻止我们使用20阶的方法。应用这样高的阶，同样的精度情形下可以采用较大的步长，如$h = 0.2$. 穿过给定区间需要的步数变少，从而有可能减小计算量
- 可以应用符号计算系统执行非数值类型的计算，从而把相当复杂的表达式的微分和积分转换到这些系统中进行。这些系统还可以把计算表达式转化为所需要的代码

Taylor级数方法

缺点

- 依赖于给定的微分方程的反复求导，因此在解曲线经过的$x-y$平面的区域内函数$f(x,y)$必须具有所需要的偏导数。而这样的条件对于解的存在性是不必要的
- 需要对问题进行初步的分析工作。从而在这个步骤中造成的误差可能被忽略而且始终不被发现
- 对于各阶求导必须单独编程，增加了编程的复杂性以及编程错误出现的可能性，代码的可读性下降

延迟型微分方程

- 在一些实际问题中有一类特殊类型的微分方程，称为延迟型微分方程(delay differential equation)或具有延迟变量的微分方程(differential equation with retarded argument)
- 人口模型以及混合问题通常具有这种特征，即$y'(x)$的值与y在x的前面值上的函数值有关
- 例如：

$$y'(x) = f(y(x-1))$$

若知道y在$x-1$上值，微分方程就能够计算$y'(x)$的值。为了从$x=0$开始积分微分方程，我们需要在$x=-1$开始的$y(x)$的变化情况。因此必须提供$y(x)$在区间$[-1,0]$上的值作为初值：

$$\begin{cases} y'(x) = y(x-1) & x \geqslant 0 \\ y(x) = x^2 & -1 \leqslant x \leqslant 0 \end{cases}$$

- 上例中第二个等式给出所需要的$y(x)$的值。若x限定在区间$[0,1]$中，则$x-1$在$[-1,0]$中，因此

$$\begin{cases} y'(x) = y(x-1) = (x-1)^2 & 0 \leqslant x \leqslant 1 \\ y(0) = 0 \end{cases}$$

这是一个通常的ODE，通过积分可以得到解为

$$y(x) = \frac{1}{3}(x-1)^3 + \frac{1}{3}, \qquad 0 \leqslant x \leqslant 1$$

- 如果解被延拓到下一个区间$[1,2]$上，则可以类似处理。此时，对于$x \in [1,2]$, 我们有

$$\begin{cases} y'(x) = y(x-1) = \frac{1}{3}(x-2)^3 + \frac{1}{3} & 1 \leqslant x \leqslant 2 \\ y(1) = \frac{1}{3} \end{cases}$$

也可以得到显式解。类似计算可以一直持续下去

复杂情形

- 对于复杂的方程，如

$$y'(x) = \sin[y(x-1)^3] + \log[y(x) + x^5]$$

我们需要借助于数值方法在每一个区间上求解：Taylor级数方法

- 例如，考虑

$$\begin{cases} y'(x) = 2y(x-1) + y(x) & x > 0 \\ y(x) = x^3 & -1 \leqslant x \leqslant 0 \end{cases}$$

- 为了在区间$[0,1]$中求解，采用如下截断的Taylor展开：

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x)$$

以步长h向前进行求解

- 我们需要提供下述导数表达式：

$$\begin{aligned}
y'(x) &= 2y(x-1) + y(x) = 2(x-1)^2 + y(x) \\
y''(x) &= 2y'(x-1) + y'(x) = 4(x-2)^2 + 2(x-1)^2 + y'(x) \\
y'''(x) &= 2y''(x-1) + y''(x) = 8(x-3)^2 + 8(x-2)^2 \\
&\quad + 2(x-1)^2 + y''(x)
\end{aligned}$$

- 基于上述信息，可以得到$[0,1]$中的离散点上$y(x)$的值。同时为了在下一区间内使用，需要存放在这些离散点上的$y'(x)$, $y''(x)$和$y'''(x)$的值。若不改变h的值，那么可以在每个区间上应用适当的存储值类似处理