计算方法LAB7

PB20000126 葛哲凯

一、算法描述

Romberg算法：

1. 计算积分区间两端点函数值f(a),f(b), 计算T1;
2. 将区间[a,b]分半，计算f((a+b)/2), T2,S1;
3. 再将区间分半，算出f(a+(b-a)/2)和f(a+3(b-a)/4), 由此计算T4，S2，C1;
4. 再将区间分半，计算T8，S4，C2，进而计算R1;
5. 再将区间分半，重复第四步工作，计算T16，S8，C4，R2，反复进行这一过程，可以计算R1，R2，R4，,,,, ,直到最终两个R之差不超过给定精度即可。

其计算过程是将区间逐次分半，加速得到积分近似值，因此称为**逐次分半加速法**

模拟质点运动：

1. 根据$a_x(t),a_y(t)$，使用Romberg算法求得$v_x(t),v_y(t)$
2. 根据$v_x(t),v_y(t)$，使用Romberg算法求得$x(t),y(t)$

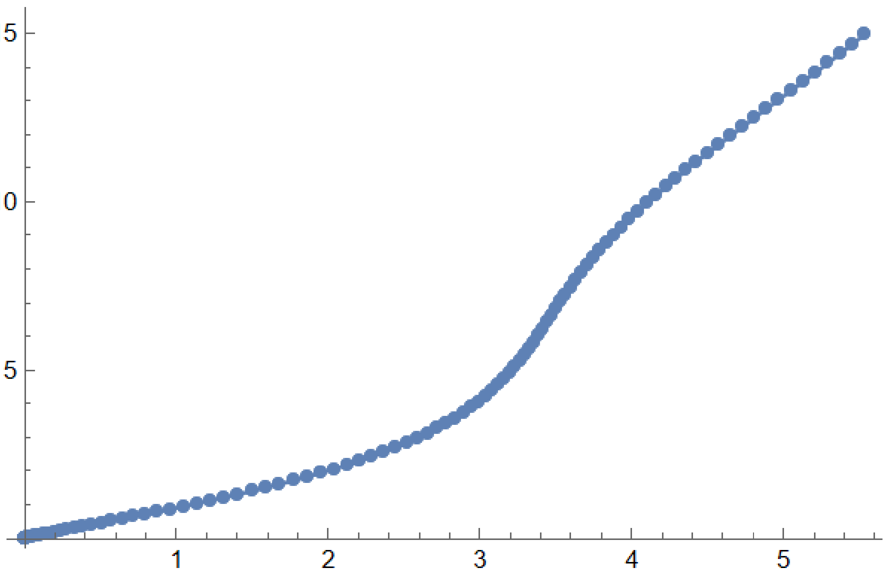
二、实验结果

当$M=8$时，运行结果如下：

```
{0,0 },{0.000137185,0.000155019 },{0.00102216,0.00115882 },{0.00327356,0.00366921 },{0.00742936,0.00818836 },{0.0139661,0.0151038 },
{0.0233072,0.0247169 },{0.0358262,0.0372634 },{0.0518488,0.0529278 },{0.0716543,0.0718552 },{0.0954757,0.0941587 },{0.123501,0.11992
6 },{0.155872,0.149225 },{0.192687,0.182107 },{0.234002,0.21861 },{0.279826,0.258759 },{0.330131,0.302573 },{0.384846,0.350061 },{0.
443862,0.401228 },{0.507034,0.456072 },{0.574182,0.514587 },{0.645094,0.576763 },{0.719528,0.642589 },{0.797217,0.71205 },{0.877866,
0.785128 },{0.961164,0.861806 },{1.04678,0.942063 },{1.13437,1.02588 },{1.22357,1.11323 },{1.31403,1.20409 },{1.40537,1.29845 },{1.4
9723,1.39627 },{1.58924,1.49753 },{1.68104,1.6022 },{1.77228,1.71027 },{1.86263,1.82171 },{1.95175,1.93649 },{2.03935,2.05459 },{2.1
2514,2.17598 },{2.20885,2.30064 },{2.29026,2.42854 },{2.36914,2.55966 },{2.44533,2.69397 },{2.51865,2.83146 },{2.589,2.97209 },{2.65
628,3.11584 },{2.72043,3.2627 },{2.78143,3.41263 },{2.83927,3.56561 },{2.89399,3.72163 },{2.94565,3.88065 },{2.99436,4.04266 },{3.04
022,4.20763 },{3.0834,4.37555 },{3.12405,4.54639 },{3.16239,4.72013 },{3.19861,4.89675 },{3.23296,5.07622 },{3.26569,5.25854 },{3.29
705,5.44368 },{3.32733,5.63161 },{3.35679,5.82233 },{3.38573,6.0158 },{3.41443,6.21202 },{3.44318,6.41096 },{3.47226,6.61261 },{3.50
194,6.81694 },{3.5325,7.02394 },{3.56418,7.23359 },{3.59723,7.44587 },{3.63188,7.66078 },{3.66832,7.87828 },{3.70676,8.09836 },{3.74
734,8.32101 },{3.79022,8.54621 },{3.83552,8.77394 },{3.88332,9.00419 },{3.93369,9.23694 },{3.98668,9.47218 },{4.04231,9.70989 },{4.1
0055,9.95006 },{4.16137,10.1927 },{4.22471,10.4377 },{4.29049,10.6851 },{4.35858,10.935 },{4.42887,11.1872 },{4.5012,11.4418 },{4.57
539,11.6988 },{4.65126,11.9581 },{4.7286,12.2197 },{4.8072,12.4836 },{4.88683,12.7499 },{4.96726,13.0184 },{5.04823,13.2892 },{5.129
51,13.5623 },{5.21086,13.8376 },{5.29202,14.1152 },{5.37275,14.395 },{5.45283,14.677 },{5.53202,14.9612 },
```

Output in VScode

Out[8]=



Graph in Mathematica

三、算法比较

M	4	8	12	16	20
达到要求精度的比例	0.0686937	0.648624	1	1	1

当$M \geq 12$后，均达到精度，达到误差要求的次数/调用总次数 = 1

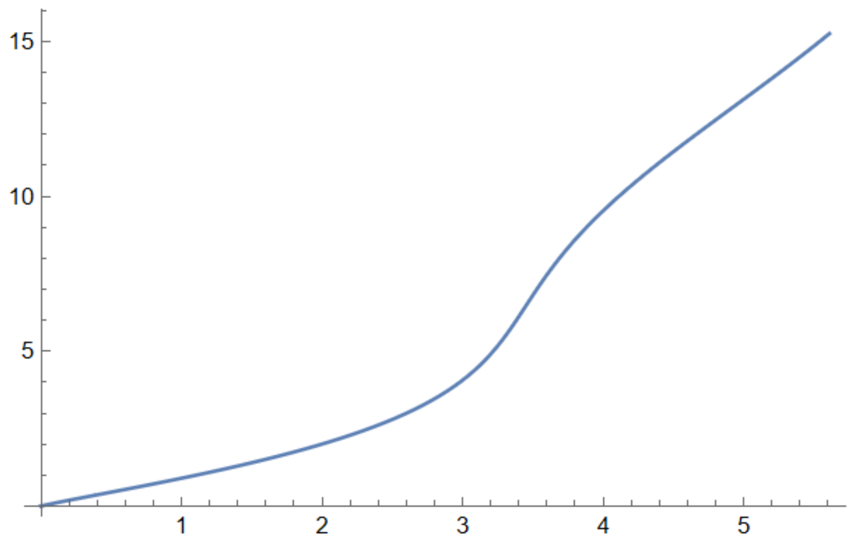
四、结果和分析

- **结果与MMA模拟结果一致，y方向速度恒增，x方向速度变化趋势类似正弦波的衰弱分布**
- **M在一定范围内时，越大，精度越高，计算次数越多,此时精度依赖于M和e，当M大于某个临界值后，M变化不影响精度和计算次数，此时精度只依赖于e。**

  可以将M取值很大，就能保证一定达到设置精度，或取精度次数/总次数>某临界值，减少运算次数，使其大多数达到精度。

```
In[82]:= Ax[t_] = Sin[t] / (Sqrt[t] + 1);
         Ay[t_] = Log[t + 1] / (t + 1);
         X[t_] = NIntegrate[Ax[r], {s, 0, t}, {r, 0, s}];
                 数值积分
         Y[t_] = NIntegrate[Ay[r], {s, 0, t}, {r, 0, s}];
                 数值积分
         data = Table[{X[t], Y[t]}, {t, 0.1, 10, 0.1}];
                表格
         ListLinePlot[data]
         绘制点集的线条
```

Out[87]=



Simulate in Mathematica