用 SA 算法求解多极小函数 实验报告

自 45 柳荫 2014011858

选择的函数是

F(x,y) = 5 * x ^ 5 – 11 * x ^ 4 + 13 * x ^ 3 – 12 * x ^ 2 + y * x;

其中 y 由用户输入，

程序用 C++写，可直接编译运行，输入 y 回车则显示计算过程和最终结果。
计算过程中，左边是 x 值，右边是函数值；且每降一次温，空一行。

对 y = 2 做 20 次实验，有 12 次结果是-3.0797，3 次是-3.0796，3 次是-3.0795，1 次是-3.0794，1 次是-3.0006.

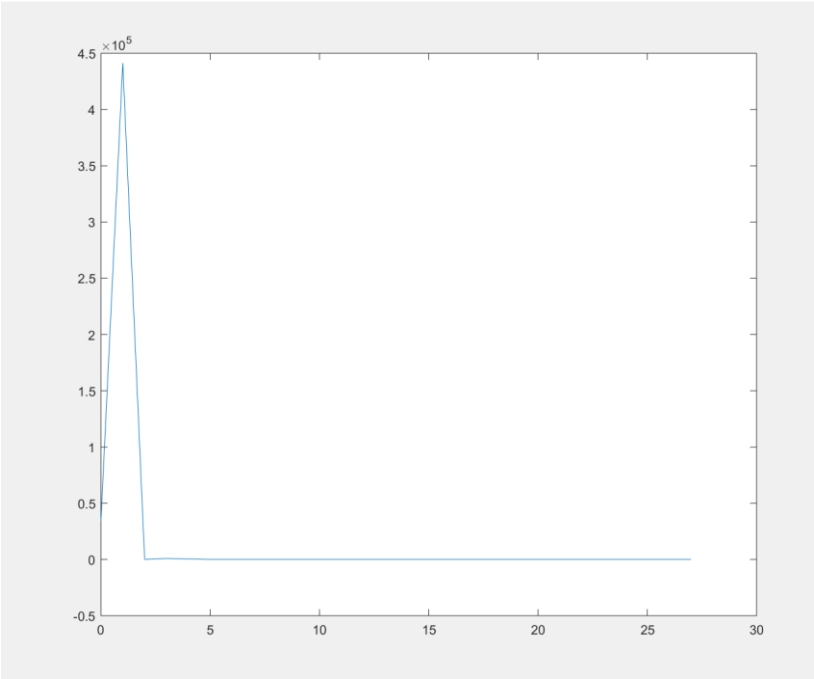
```
>> mean(a)

ans =

   -3.0757

>> var(a)

ans =

   3.1235e-04
```

其最优是-3.0797， 最差是-3.0006，平均是-3.0757，方差是 0.0003。

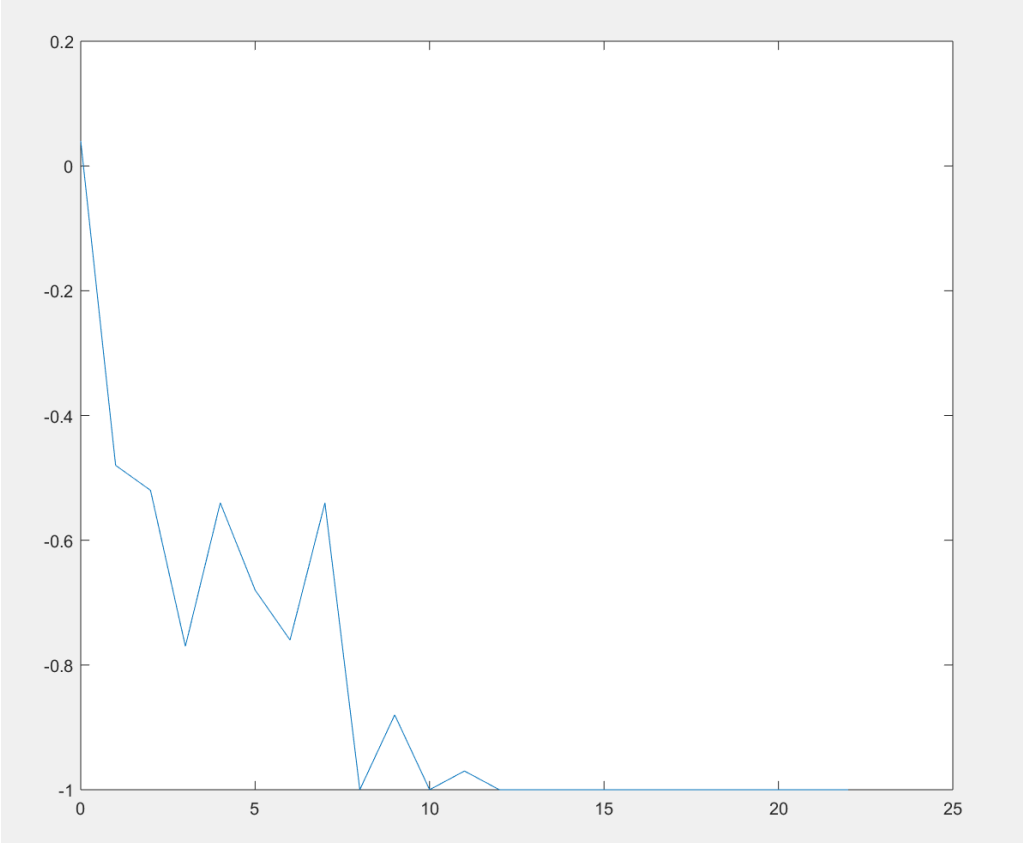
为画出某次仿真中目标函数的变化曲线，暂时把程序中的默认的最小值从 10^99 改为 10^8，于是用 y = 4 做仿真实验：

```
4
6.26179 33969.8
10.184  441935
0.0109867       0.0425153

3.25779 930.902
2.88107 457.432
0.0109867       0.0425153

1.18488 -0.486542
0.738231        -0.527462
1.12977 -0.769078

0.741111        -0.535349
0.79417 -0.676376
1.12977 -0.769078

0.741111        -0.535349
0.981613        -0.996398

1.097   -0.877885
0.981613        -0.996398

1.051   -0.968777
0.981613        -0.996398
1.00741 -0.999388

0.991742        -0.999261
0.998661        -0.99998
0.999924        -1

0.999118        -0.999991
0.998661        -0.99998
0.999924        -1

0.999118        -0.999991
0.998661        -0.99998
0.999924        -1

-1.0000

--------------------------------
Process exited after 4.798 seconds with return value 0
请按任意键继续. . .
```

目标函数变化曲线如下：



除了一开始一下子降低了许多，后来的变化情况相对小看不出来，若舍去开始很大的值，就得到：



符合模拟退火目标函数的变化规律，在起伏中下降。

模拟退火算法在函数优化中的特点：

退火算法中各个参数值的选择和设置对运行的结果和效果有较大影响,对参数进行合理的优化,才能得到最理想的结果。算法中关于初始温度、温度终止、降温速度是求解函数优化问题重要的环节。
比如此题中如果退温速度很快就能从初始温度降到终止温度的话,则很可能求不到最优解，退火相对过快，效果不好；而退火过慢的话，则又不必要，本来很快就能找到最优解的。
另外 ,模拟退火算法比局部搜索算法搜索的解空间更大些 ,因此 ,更有可能达到整体最优解 ,即使达不到 ,所得近似最优解的质量也比局部搜索算法好。我个人感觉模拟退火是对爬山法的一个巨大的飞跃,它不会局限于极值点走不下去。

实验体会：

多数时候都能找到最优解，偶尔会比最优解差一点，所以要找到最优解的话，除了对程序进行合理的优化，还要进行多次运行，找一个结果最小的。感觉实验还是很有意思的。