统计推断在数模转换系统中的应用

组号 37　　贾煜 5140309118，黄兆丰 5140309113

摘要：本文为上海交通大学电子信息与电气工程学院的《统计推断在数模转换中的应用》的课程论文。本小组针对传感器的非线性特性，为该传感器的批量生产设计一种成本合理的传感特性校准方案，并以论文的形式提供解决方案。本研究在 Matlab 编程环境下，结合统计推断的思想方法，采用模拟退火算法和遗传算法合理更换取点方案，最终计算并比较成本确定较优方案。

关键词：Matlab，模拟退火算法，遗传算法

Application of Statistic Model in Digital to Analog Conserve System

ABSTEACT: This report aims to study the nonlinear characteristic curve of a certain kind of sensor and provide methods to calibrate it for mass-production with all cost taken into consideration. Our team use Matlab environment combining with statistic model to generate a calibrating plan which compares simulated annealing with genetic algorithm and point out the best solution with the least cost.

Key words: Matlab, Simulated Annealing, Genetic Algorithm

1 引言

工程或实验中，我们经常遇到要寻找两个特定相关变量函数关系的问题。在具备一定先验知识的前提下，我们可以直接确定关系函数，再利用公式确定特定情况下的参数。但是通常二者的数学关系是复杂且无法推导的，此时我们就需要借助统计推断的思想方法，合理选择方案，全面考虑测定成本和误差成本，找到最优解。[1]

1.1 课题概述

假定有某型投入批量试生产的电子产品，其内部有一个模块，功能是监测某项与外部环境有关的物理量（可能是温度、压力、光强等）。该监测模块中传感器部件的输入输出特性呈明显的非线性。本课题要求为该模块的批量生产设计一种成本合理的传感特性校准（定标工序）方案。

1.2 模型建立

1.2.1 模型叙述

为了对本课题展开有效讨论，需建立一个数学模型，对问题的某些方面进行必要的描述和限定。

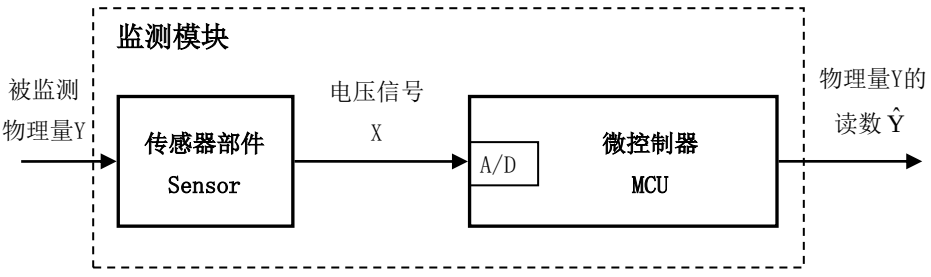

图 1.2 监测模块组成框图

监测模块的组成框图如图 1.2。其中，传感器部件（包含传感器元件及必要的放大电路、调理电路等）的特性是我们关注的重点。传感器部件监测的对象物理量以符号 Y 表示；传感部件的输出电压信号用符号 X 表示，该电压经模数转换器（ADC）成为数字编码，并能被微处理器程序所读取和处理，获得信号 $\hat{Y}$ 作为 Y 的读数（监测模块对 Y 的估测值）。[2]

所谓传感特性校准，就是针对某一特定传感部件个体，通过有限次测定，估计其 Y 值与 X 值间一一对应的特性关系的过程。数学上可认为是确定适用于该个体的估测函数 $\hat{y}=f(x)$的过程，其中 x 是 X 的取值，$\hat{y}$ 是对应 Y 的估测值。

1.2.2 实际模型

考虑实际工程中该监测模块的应用需求，同时为便于在本课题中开展讨论，我们将问题限于 X 为离散取值的情况，规定

$$X\in\{x_1,x_2,x_3,...,x_{50},x_{51}\}=\{5.0,5.1,5.2,...,9.9,10.0\}$$

相应的 Y 估测值记为 $\hat{y}_i=f(x_i)$， Y 实测值记为 y_i， $i=1,2,3,...,50,51$。

1.3 成本计算

为评估和比较不同的校准方案，本实验按要求使用如下的成本计算公式：

1.3.1 单点定标误差成本

$$s_{i,j}=\begin{cases}0 & \text{if } |\hat{y}_{i,j}-y_{i,j}|\le 0.4\\ 0.1 & \text{if } 0.4<|\hat{y}_{i,j}-y_{i,j}|\le 0.6\\ 0.7 & \text{if } 0.6<|\hat{y}_{i,j}-y_{i,j}|\le 0.8\\ 0.9 & \text{if } 0.8<|\hat{y}_{i,j}-y_{i,j}|\le 1\\ 1.5 & \text{if } 1<|\hat{y}_{i,j}-y_{i,j}|\le 2\\ 6 & \text{if } 2<|\hat{y}_{i,j}-y_{i,j}|\le 3\\ 12 & \text{if } 3<|\hat{y}_{i,j}-y_{i,j}|\le 5\\ 25 & \text{if } |\hat{y}_{i,j}-y_{i,j}|>5\end{cases} \quad (1.3.1)$$

其中 $y_{i,j}$ 表示第 i 个样本之第 j 点 Y 的实测值，$\hat{y}_{i,j}$ 表示定标后得到的估测值（读数），该点的相应误差成本以符号 $s_{i,j}$ 记。

1.3.2 单点测定成本

本实验根据实际情况制定单点测定成本 q=12。

1.3.3 某一样本个体的定标成本

$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i \quad (1.3.3)$$

式中 n_i 表示对该样本个体定标过程中的单点测定次数。

1.3.4 校准方案总成本

即使用该校准方案对标准样本库中每个样本个体逐一定标，取所有样本个体的定标成本的统计平均。

$$C = \frac{1}{M}\sum_{i=1}^{M} S_i \qquad (1.3.4)$$

总成本较低的校准方案，认定为较优方案。

2 方案设计

2.1 基本思路

本实验的实质是对给定的成本函数，找到其最优解。实验中有两个优化方向，即取点方案和拟合方法。前者关于所选取的测定点的位置和点数。后者为选择拟合插值方法，即数学表达式。拟合方式、选点个数和选点方式三者为主要研究对象。本实验采用的模拟退火法的最优化解的搜索方法，拟合方式选择三次 Hermite 插值拟合。

2.2 选择拟合方法

本实验中我们采用三次 Hermite 插值拟合的方法。Hermite 插值法的基本思想是，构造一个函数使得对于每一个已知的 x 值，该点的函数值以及函数的一阶导数值与实际测定值相等。Hermit 插值相对于仅满足函数值相等的 Lagrange 插值精确度更高，但相对而言花费的时间更长。

Hermite 插值法在 Matlab 中有内置的拟合函数 pchip()，调用方式是 pchip(x,y)，其中 x 和 y 分别是储存横坐标和纵坐标数据的矩阵。

2.3 选择取点的个数

每组数据共有 51 个等间隔的点，并且从数据中我们可以发现 Y 的值关于 X 单调递增。相对于选择一些必须取定的点，我们更偏向采用在给定取点个数的前提下完全随机选择的方式，使得到的结果更加接近最优解。经过多次尝试，我们认为取点个数在 7 次较为合适。个数较少会使得到的函数曲线与实际相差较大，误差成本上升，而取点过多将导致测定成本大幅提高。

2.4 模拟退火算法

2.4.1 模拟退火算法的引入

模拟退火算法是爬山法的改进算法。爬山法就是在当前解的邻域内寻找一个新解，如果该新解优于当前解，就放弃当前解，否则依然使用当前解。爬山法的缺点在于其很容易陷入局部最优解，而忽略了全局最优解。简单地说，就是当我们使用爬山法找到一个在其邻域中最优的解后，生成的新解永远不如当前解，使得系统认为当前解即为最优解，但是在该邻域之外完全有可能存在更优的解。因此爬山法对初始值十分敏感，而且往往找不到全局的最优解。

模拟退火算法是在爬山法的基础上，将一定不接受不如当前解的新解改为以一定概率接受。这样就让程序能够跳出局部最优解，直到连续一定次数都无法接受新解，才认为当前解是最优解，解决了爬山法的最显著问题。

2.4.2 模拟退火算法原理

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 e(-ΔE/(kT))，其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值，算法终止时

的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值 t 及其衰减因子 Δt、每个 t 值时的迭代次数 L 和停止条件 S。

2.4.3 模拟退火算法的模型

模拟退火算法可以分解为解空间、目标函数和初始解三部分。模拟退火算法的基本思想:

(1) 初始化：初始温度 T(充分大)，初始解状态 S(是算法迭代的起点)，每个 T 值的迭代次数 L；

(2) 对 k=1,……,L 做第(3)至第 6 步；

(3) 产生新解 S′；

(4) 计算增量 Δt′=C(S′)-C(S)，其中 C(S)为评价函数；

(5) 若 Δt′<0 则接受 S′作为新的当前解，否则以概率 exp(-Δt′/T)接受 S′作为新的当前解；

(6) 如果满足终止条件则输出当前解作为最优解，结束程序。终止条件通常取为连续若干个新解都没有被接受时终止算法。

(7) T 逐渐减少，且 T→0，然后转第 2 步。

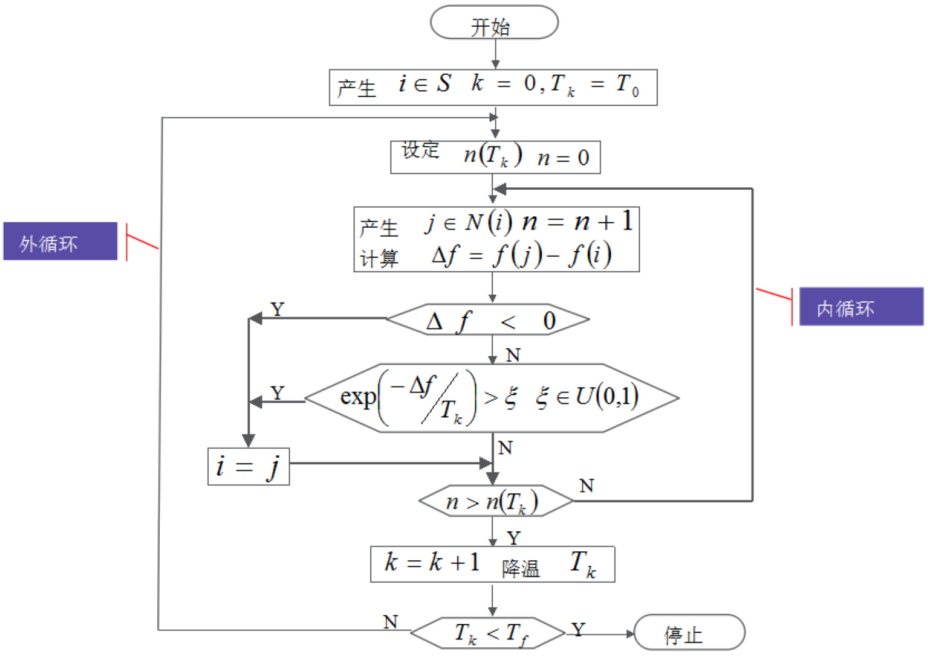

图 2.4.3 模拟退火算法流程图

2.4.4 模拟退火算法的步骤

模拟退火算法新解的产生和接受可分为如下四个步骤：

(1) 由一个产生函数从当前解产生一个位于解空间的新解；为便于后续的计算和接受，减少算法耗时，通常选择由当前新解经过简单地变换即可产生新解的方法，如对构成新解的全部或部分元素进行置换、互换等，注意到产生新解的变换方法决定了当前新解的邻域结构，因而对冷却进度表的选取有一定的影响。

(2) 计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。

(3) 判断新解是否被接受,判断的依据是一个接受准则，最常用的接受准则是 Metropolis 准则: 若 Δt′<0 则接受 S′作为新的当前解 S，否则以概率 exp(-Δt′/T)接受 S′作为新的当前解 S。

(4) 当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现了一次迭代。可在此基础

上开始下一轮试验。而当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。模拟退火算法与初始值无关，算法求得的解与初始解状态 S(是算法迭代的起点)无关；模拟退火算法具有渐近收敛性，已在理论上被证明是一种以概率 l 收敛于全局最优解的全局优化算法；模拟退火算法具有并行性。

2.4.5 对于模拟退火算法的思考

同时我们应该注意到模拟退火算法的一些缺点。模拟退火算法虽然方法简单，对单个样本的优化而言，收敛速度却比较慢，花费的时间较长，特别是解决实际问题时，在最优选点个数未知的情况下，却必须提前指定选点的个数，可能要尝试很多的值，不适合解决复杂度很高的问题。更进一步地，虽然较之于爬山法做了改进，模拟退火算法依然有可能陷入局部最优解。

2.5 遗传算法

2.5.1 遗传算法的引入

遗传算法参考了物种进化过程中基因的传递，引入自然选择的概念，在随机生成的方案中淘汰掉成本较高的个体，再以剩余的个体为基础生成下一批方案并计算成本，如此循环直到进行足够多次的选择，就可以认为剩余方案中成本最低的为最优解。

2.5.2 遗传算法的原理

遗传算法模拟自然选择对基因的作用，以成本选择每一个方案淘汰与否。自然界中，任何种群都有一个基因库，基因库中某个基因在所有等位基因中所占的比例称为该基因的基因频率。拥有能更好适应自然环境基因的个体更容易生存下来并进行繁衍，将优秀的基因传递下去，使得该基因的基因频率逐渐提升。而拥有不适应自然环境基因的个体很可能没有存活到足以繁衍后代的年龄就在竞争中死亡了，此种基因就逐渐被取代。与此同时，基因并非是一成不变的，产生后代的过程中会有时出现基因突变(mutation)和基因重组(recombination)。顾名思义，基因突变是指原本不存在的基因会以一定的概率突然出现，而基因重组是后代的基因链结合了父辈其中一个个体的一部分基因和另一个体另一部分基因，基因的组合发生了改变，但没有新的基因产生。

遗传算法是把每一个样本点的选择与否看作一个基因，在本次研究的问题中，即每个个体有 51 个基因。对于每一代，计算每个个体的适应度，让适应度越高（成本越低）的个体在后代中所占的比例越高，然后以一定概率对后代的基因进行变异和互换，得到最终生成的后代，如此根据前一代不断生成后代直到一定代数，就能认为该代基因中适应度最高的基因为最优解。

2.5.3 遗传算法的模型

遗传算法的基本思想如下：

(1)初始群体的生成：随机产生了几组初始种群开始进化。

(2)选择：从群体中选择适应度高的个体，淘汰适应度低的个体。选择的目的是把优化的个体遗传到下一代。以适应度作为选择依据，适应度越高，被选择的概率就越高。

(3)交叉：在自然界生物进化过程中起核心作用的是基因突变和基因重组。交叉是指把两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃性的提高。

(4)变异:变异算子的基本内容是对群体中的个体串的某些基因位置上的基因值作变动。变异能够使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算子已接近最优解邻域时，利用变异算子的这种局部随机搜索能力可以加速向最优解收敛。显然，此种情况下的变异概率应取较小值，否则接近最优解的积木块会因变异而遭到破坏。另外通过变异还可以维持群体多样性，以防止出现未成熟收敛现象。此时收敛概率应取较大值。

(5)转至第 2 步。

(6)当最迭代次数达到预设的代数时，算法终止。

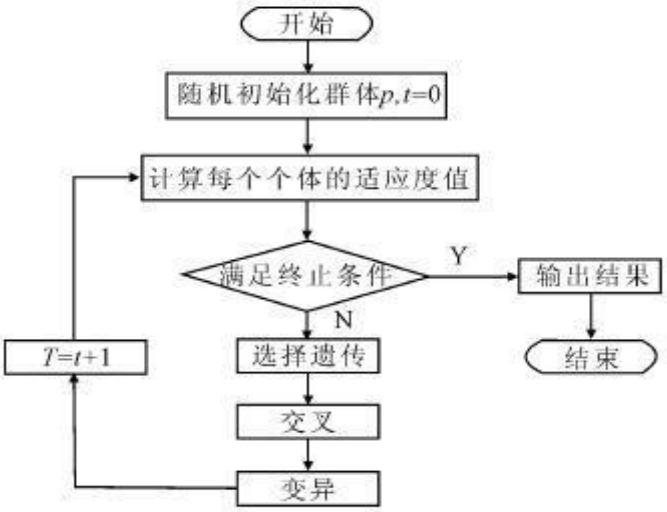

图 2.5.3 遗传算法流程图

2.5.4 对于遗传算法的思考

遗传算法较为真实地模拟了自然界中物竞天择适者生存的过程，由于每一代中存在较多不同的个体，基本能够保证找到全局最优解，对初始值不敏感，有效地防止了陷入局部最优解的错误，也无需事先指定选点个数。然而遗传算法的缺点也是显而易见的。假如每一代中个体过多，算法的复杂度就会非常高，导致运行效率极低。而如果每一代中个体数不足，较优基因就容易被偶然淘汰，这种情形在物种进化中被称为遗传漂变，即在很小种群中某一个体的死亡会对整个种群的基因频率产生很大影响，这样的种群很难得到有效进化。同时，正如自然界中进化是一个漫长的过程一样，若要让变异可控，交叉互换率和变异率就必定很低，进化的速度就会十分缓慢。最后，遗传算法只进行同一代中的横向比较，而不进行多代中最优解的纵向比较，因而无法像模拟退火算法一样自行决定终止，只能繁衍到指定的代数再输出该代的最优解，这样就有可能浪费很多代的计算。实际操作中，如果已知最优选点个数，遗传算法的运行时间往往是模拟退火算法的 3 至 4 倍。[3]

3. 结果展示

3.1 模拟退火算法的结果

目前实现的程序运行的结果为：

```
position: [ 4 16 26 35 48 ]
min_cost: 84.7482500
```

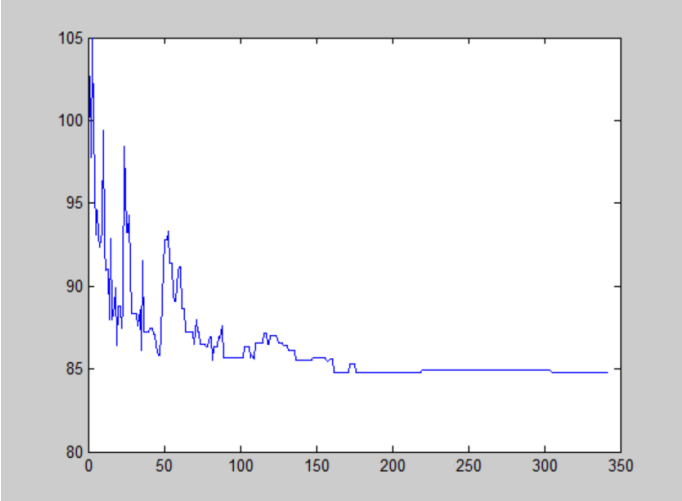

图3.1 模拟退火算法收敛曲线

表3-1-1 测试点为6个时模拟退火算法运行结果比较

temp \ k	0.001	0.005	0.01
1000	86.61875	86.6145	86.6145
500	86.6145	86.6145	86.61875
300	86.61875	86.61875	86.61875

表3-1-2 测试点为5个时模拟退火算法运行结果比较

temp \ k	0.001	0.005	0.01
1000	84.74825	84.74825	84.74825
500	84.74825	85.14625	84.74825
300	84.74825	84.74825	84.74825

而当测试点为7个时，测试成本已经达91，不是最佳方案。

3.2 遗传算法的结果

目前实现的程序运行的结果为：

```
position: [  4 16 26 35 48 ]
min_cost: 84.7482500
```

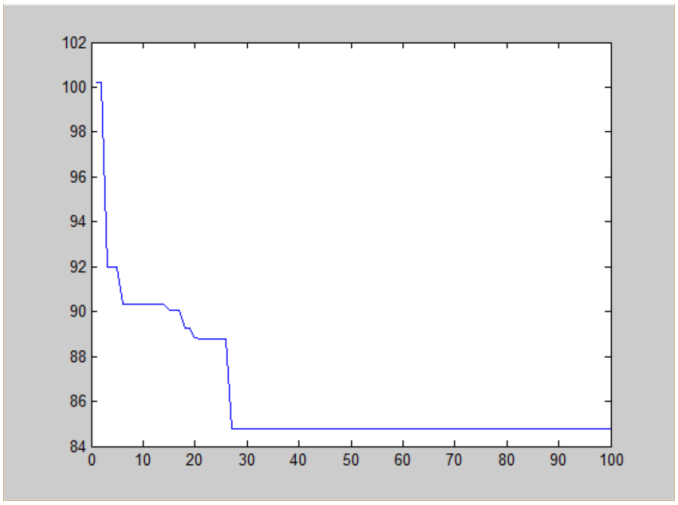

图3.2 遗传算法收敛曲线

表3-2 遗传算法程序运行结果

mutation	0.02	0.05	0.1
1	88.0780000	85.1462500	86.7205000
2	88.7880000	86.6145000	88.8230000
3	86.7765000	84.7482500	86.6670000

3.3 两种算法程序运行结果对比

表3-3 遗传算法与模拟退火结果对比

遗传算法		模拟退火算法（5个样本点）	
测试点组合	定标成本	测试点组合	定标成本
4 15 25 35 48	85.1462500	4 16 26 35 48	84.74825
4 15 24 31 39 49	86.6145000	4 16 26 35 48	84.74825
4 15 25 35 48	85.1462500	4 16 26 35 48	84.74825
3 13 21 28 36 48	87.8602500	4 16 26 35 48	84.74825
4 15 23 30 38 49	86.6187500	4 16 26 35 48	84.74825
4 16 26 35 48	84.7482500	4 16 26 35 48	84.74825

本次研究针对的具体问题规模不大，因而无论是模拟退火算法还是遗传算法都能在有限的时间内得到结果，只是相对来说模拟退火算法所需的时间更短。经过多次尝试，我们发现模拟退火算法取点个数在5个时得到的成本最小，这与遗传算法产生的结果相吻合。

4. 结论

综合所有的结果，我们认为在现实中传感器大规模生产的先行定标时，鉴于最优选点个数的未知性，建议采用遗传算法进行定标，同时不指定必须选定的样本点，并采用三次Hermite插值拟合。这样得到的定标结果不仅成本较低，而且需要的时间较短，适合作为普遍的定标准则。

然而我们也注意到了遗传算法结果不稳定的问题。尽管种群数量已经很大（本次实验中设定为200），有足够的变异空间，最终结果依然对初始值非常敏感。限于个人计算机的配置情况，我们无法继续提高种群数量，工业化生产中应该不存在这种问题。我们尝试不引入交叉互换，通过对比发现有无交叉互换过程对最终结果的影响不大，说明本次研究的问题对交叉互换不敏感，因而实际定标中可以根据具体情况酌情删去该步骤。

5. 参考文献

[1]周建兴, 岂兴明, 矫津毅, 常春藤 等.《MATLAB 从入门到精通》 人民邮电出版社
[2]上海交大电子工程系·统计推断在数模转换系统中的应用课程讲义
[3]张德丰.《MATLAB 程序设计与经典应用》 电子工业出版社

6. 附录

6.1 代码展示

6.1.1 模拟退火算法

⑴ 主程序**(main.m)**

```
data=csvread('20150915dataform.csv');
```

```
y=zeros(400,51);
y(1:400,:)=data(2:2:800,:); % 得到了所有的 y 值
num=7; % 测试点数量
temp=1000; % 初始温度
waveform=zeros(1,floor(log(1/temp)/log(0.98))+1);
% floor()向下取整 此处列数等于总循环次数
monte=2*num; % 内部蒙特卡洛循环次数
k=0.005; % 玻尔兹曼常数
n=1; % 记录正在进行的循环次
flag=1; % 是否接受新状态， 初始时接受了新状态

state=state_init(num);
while temp>1
    for i=1:monte
        if flag % 接受新状态时才重新计算成本
            cost=total_cost(state,y);
        end
        state0=state_update(state);
        cost0=total_cost(state0,y);
        flag=if_accept(cost,cost0,temp,k); % 是否接受新状态
        if flag
        state=state0;
        end
    end
    fprintf('%3d [',n);
    fprintf('%2d ',state);
    if flag
        waveform(n)=cost0;
        fprintf('] %10.7f\n',cost0);
    else
        waveform(n)=cost;
        fprintf('] %10.7f\n',cost);
    end
    temp=temp*0.98;
    n=n+1;
end
plot(1:n-1,waveform); % 画出收敛曲线
write(state,y); % 计算得到的解的成本并写入文件
```

⑵状态初始化函数**(state_init.m)**

```
function out = state_init(num)
%   随机产生初始状态

out=zeros(1,num);
```

```
for i=1:num
    t=round(rand()*51)+1;
    for j=1:i
        if any(out(i)==t)
        %while any(out(i)==t)            % any(x) x 非零时返回 true
            t=round(rand()*51)+1;
        end
    end
    out(i)=t;
end
out=sort(out);        % sort()排序
end
```

(3) 总成本计算函数**(total_cost.m)**

```
function out = total_cost(state,y)
%   计算状态成本
x=5:0.1:10;
c=length(state);   %   测试点数量

xx=5+(state-1)*0.1;   %   测试点 x 值
yy=y(:,state);   %   测试点值矩阵
f=pchip(xx,yy);   %  分段三次 hermit 插值
dy=ppval(f,x)-y;   %   误差值          ppval(f,x)求得 f 上 x 对应的 y^值

%   误差成本
t=abs(dy);
t0=sum(sum(t<=0.4));          %  判断语句执行后生成了 0,1 为元素的矩阵。
t1=sum(sum(t<=0.6))-t0;
%  第一个 sum 先得到一个列向量，第二个 sum 再得到满足条件的个数
t2=sum(sum(t<=0.8))-t0-t1;          % t0=(t<=0.4);
t3=sum(sum(t<=1))-t0-t1-t2;
t4=sum(sum(t<=2))-t0-t1-t2-t3;
t5=sum(sum(t<=3))-t0-t1-t2-t3-t4;
t6=sum(sum(t<=5))-t0-t1-t2-t3-t4-t5;
t7=sum(sum(t>5));
error_cost=0.1*t1+0.7*t2+0.9*t3+1.5*t4+6*t5+12*t6+25*t7;

out=12*c+error_cost/400;   %   成本
end
```

(4) 状态更新函数**(state_update.m)**

```
function out = state_update(state)
%     随机扰动更新状态
out=state;
```

```
pos1=floor(rand()*51)+1;
while ~any(out==pos1)
    pos1=floor(rand()*51)+1; %   随机选择一个已选定的测试点
end

pos2=floor(rand()*51)+1;
while any(out==pos2)
    pos2=floor(rand()*51)+1; %   随机选择一个未选定的点
end

pos=find(out==pos1);
out(pos(1))=pos2;   %   更新
out=sort(out);
end
```

(5)判断是否接受新状态函数**(if_accept.m)**

```
function out = if_accept(cost,cost0,temp,k)
%    是否接受新状态
out=0;
if cost0<cost   %   成本降低时一定接受
    out=1;
else
    t=rand();
    dE=cost-cost0;
    if t<exp(dE/(k*temp))   %   成本升高时按规则有一定几率接受
        out=1;
    end
end
end
```

(6)写文件函数**(write.m)**

```
function out =write(in,y)
%    计算成本并将最小成本写入文件
out=total_cost(in,y);

fname=fopen('min_cost.txt','a');
fprintf(fname,'position: [');
fprintf(fname,'%2d ',in);
fprintf(fname,']\nmin_cost: %10.7f\n\n',out);
fclose(fname);
end
```

6.1.2 遗传算法

(1)主程序**(main.m)**

```
data=csvread('20150915dataform.csv');
population=200; % 种群数量
crossing_over=0.9; % 交叉概率
mutation=0.05; % 变异概率
generation=100; % 繁衍代数
y=zeros(400,51);
y(1:400,:)=data(2:2:800,:);
waveform=zeros(1,generation);
gene=geneinit(population);
for g=1:generation
cost=adapt(gene,y,population);
gene=select(gene,cost,population);
gene=generate(gene,population,crossing_over);
gene=mutate(gene,population,mutation);
waveform(g)=min(cost);
fprintf('%3d [',g);
fprintf('%2d ',find(gene(1,:)==1));
fprintf('] %10.7f\n',min(cost));
end
plot(1:generation,waveform); % 画图
xx=find(gene(1,:)==1);
write(xx,y); % 写文件
```

(2) 初始化种群函数**(geneinit.m)**

```
function out = geneinit(population)
% 随机产生初始种群
out=round(rand(population,51)-0.3);
end
```

(3) 适应度函数**(adapt.m)**

```
function out = adapt(gene,y,population)
out=zeros(population,1);
x=5:0.1:10;
for i=1:population
c=sum(gene(i,:)==1);
pos=find(gene(i,:)==1);
thisx=5+(pos-1)*0.1;
thisy=y(:,pos);
f=pchip(thisx,thisy);
yfault=ppval(f,x)-y;
out(i)=12*c+errorcost(yfault)/400; % 单个个体平均成本
end
end
```

(4) 自然选择函数**(select.m)**

```
function out = select(gene,cost,population)
out=zeros(population,51);
cost0=(max(cost)-cost).^0.5; %  适应度
s0=sum(cost0);
s=zeros(population+1);
s(1)=0;
s(population+1)=1;
s(2:population)=sum(cost0(1:population-1))/s0;
for i=2:population
t=rand();
j=search(t,s,1,population+1);
out(i,:)=gene(j,:);
end
cost_sort=[(1:population)',cost];
cost_sort=sortrows(cost_sort,2);
out(1,:)=gene(cost_sort(1,1),:);
end
```

(5)交叉互换函数**(generate.m)**

```
function out = generate(gene,population,crossing_over) %交叉互换
for i=2:floor(population/2+1)
out=gene;
mid=floor(rand()*50)+1;
t=rand();
if t<=crossing_over
out(i,mid+1:51)=gene(population-i+2,mid+1:51);
out(population-i+2,mid+1:51)=gene(i,mid+1:51);
end
end
end
```

(6)基因突变函数**(mutate.m)**

```
function out = mutate(gene,population,mutation) %  基因突变
out=gene;
for i=2:population;
for j=1:50;
t=rand();
if t<=mutation
out(i,j)=~out(i,j);
end
end
end
end
```

(7) 二分查找函数**(search.m)**

```
function [out] = search(input,s,low,high)
mid=floor((low+high)/2);
if input<=s(mid)
if input>s(mid-1)
out=mid-1;
else
out=search(input,s,low,mid);
end
else
if input<=s(mid+1)
out=mid;
else
out=search(input,s,mid,high);
end
end
end
```

(8) 误差成本计算函数**(errorcost.m)**

```
function [out] = errorcost(dy)
t=abs(dy);
t0=sum(sum(t<=0.4));
t1=sum(sum(t<=0.6))-t0;
t2=sum(sum(t<=0.8))-t0-t1;
t3=sum(sum(t<=1))-t0-t1-t2;
t4=sum(sum(t<=2))-t0-t1-t2-t3;
t5=sum(sum(t<=3))-t0-t1-t2-t3-t4;
t6=sum(sum(t<=5))-t0-t1-t2-t3-t4-t5;
t7=sum(sum(t>5));
out=0.1*t1+0.7*t2+0.9*t3+1.5*t4+6*t5+12*t6+25*t7;
end
```

(9) 写文件函数**(write.m)**

```
function [out] =write(in,y)
out=length(in)*12;
x=5:0.1:10;
thisx=5+(in-1)*0.1;
thisy=y(:,in);
f=pchip(thisx,thisy);
yfault=ppval(f,x)-y;
out=out+errorcost(yfault)/400;
fid=fopen('answer.txt','a');
```

```
fprintf(fid,'position: [ ');
fprintf(fid,'%2d ',in);
fprintf(fid,'] min_cost: %10.7f\n\n',out);
fclose(fid);
end
```