统计推断在数模转换系统中的应用

组号：08　姓名：李晨阳（组长） 学号：5140309003　姓名：刘蓝晴 学号：5140309049

主要项目	说明
代码方面	《统计推断在模数转换系统中的应用》，刘学成，2014 年秋季学期，组号 05 参考了该组报告附录内的部分算法
算法描述	《统计推断在模数转换系统中的应用》，刘学成，2014 年秋季学期，组号 05 参考了该组遗传算法伪代码的表述
拟合方法描述，包括示意图	《统计推断在模数转换系统中的应用》，张世豪，2012 年秋季学期，组号 015 参考了该组对拟合方法的表述，引用了其示意图

摘要：作者在运用遗传算法的统计推断方法之后，结合多项式拟合和样条插值的统计数学的方法，对课程中提出的监测模块中传感器部件非线性特性进行了研究。

关键词：遗传算法；传感器部件非线性特性

1 引言

随着科技的发展，人们逐渐开始用电子设备监测模块对某些对某些特征量进行监测。但是批量生产的监测模块，由于制作工艺和偶然性等原因，不同的检测模块之间性能可能会有或大或小的差异。于是，在监测模块制作完成后，对监测模块的校准定标就成了一项很必要的工作。当然对每个模块进行标准的定标对用户来说当然是最好的，但是这样必然会造成生产成本过高，既耗费人力，又耗费物力，对厂家和用户都不利。于是我们采用计算机辅助分析，使用 MATLAB 作为工具程序语言，模拟定标过程。这样便可以在不影响用户使用器件的同时，大大的减小对该模块的定标所产生的成本。

2 课题目标

课题的目标找到几个特征点。对监测模块的输入 x 离散取值，通过估测的函数得到的估测值 $\hat{y}$ 与实际值 y 进行比较评价后，使定标成本尽可能减少。

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，$i = 1, 2, 3, ..., 50, 51$。

相应的符号对应的表示意义如下：

X　　传感部件的输出电压

Y 传感部件的输入，即被监测物理量

$\hat{Y}$ 监测模块的输出，即监测模块将传感部件输出 X 所转换成的读数（对 Y 的估测值）

$y_{i,j}$ 第 i 个样本之第 j 点对应的 Y 实测值（来自标准样本数据库）

$\hat{y}_{i,j}$ 第 i 个样本之第 j 点对应的 Y 估测值

$s_{i,j}$ 第 i 个样本之第 j 点对应的单点定标误差成本

q 单点测定成本

n_i 对样本 i 定标过程中的单点测定次数

S_i 对样本 i 的定标成本

M 标准样本数据库中的样本总数

C 基于标准样本数据库评价一个校准方案，算得的该方案总体成本[1]

3 基于特征点的插值拟合可用算法

3.1 暴力穷举法

若对所有数据逐一进行选取 k(k=5,6,7)个点，从中找出最优解。但由于数据过多，需要进行C_{51}^{k}次拟合或者求解方程，对于如此巨大的数目，计算过于复杂，无法完成如此多的拟合，或者求解多项式。因此，必须改变算法，利用统计知识来求解。

3.2 模拟退火算法

模拟退火算法来源于固体退火原理，是一种基于概率的算法，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[2]。

3.2.1 模拟退火算法流程图

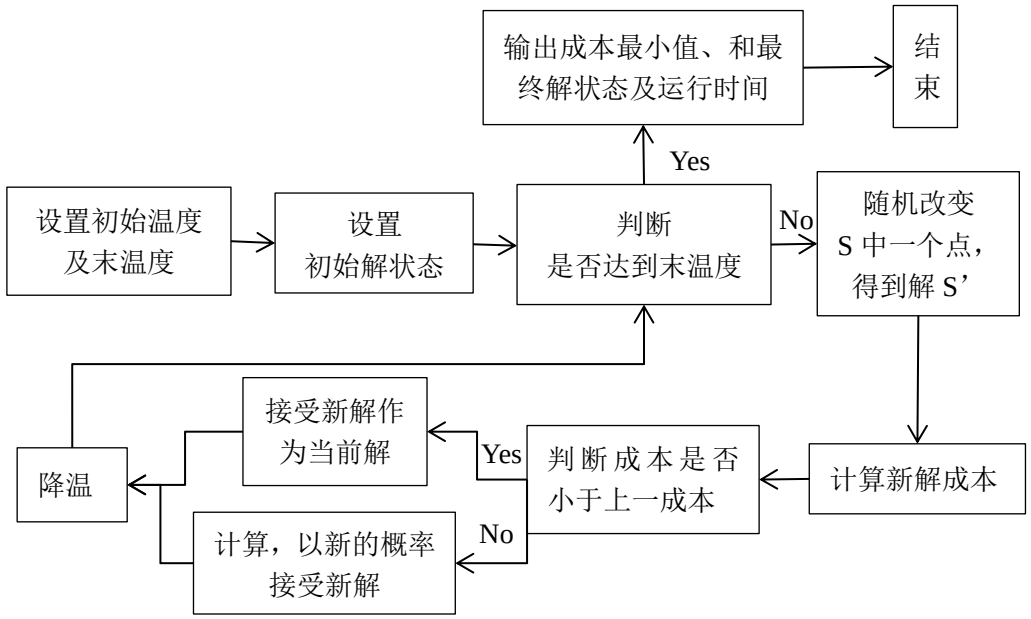

图 3-1 模拟退火算法流程图

3.2.2 模拟退火算法简述

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。

（1）初始化：初始温度 T0(设置为 10)，初始解状态 S，随机选择 51 个点中的 k 个点（k=5，6，7，8），终止条件设置为 T0＜0.01，每次迭代使新的 T0 为原来 T0 的 0.98 倍。

（2）对每次迭代做第(3)至第 6 步：

（3）随机取得 S 中的一个数，改变其值，再次排序后产生新解 S′，令 S = S′。

（4）计算新解成本，将其与当前成本最小值比较，若新解成本小于当前成本最小值，则新解被接受，更新最小成本值，且将其置为一个可能的最优解。

（5）倘若不然，增量Δt′为新解与上一解成本值的差值，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。

（7）T 逐渐减少，且 T 大于设定的末温度（0.01），然后转第 2 步。

3.3 遗传算法

遗传算法是模拟模拟自然界生物进化机制发展起来的随机全局搜索和优化方法，借鉴了达尔文的生物进化论和孟德尔的遗传学说。是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。在潜在的解决方案种群中逐次产生一个近似最优解的方案，在遗传算法的每一代，根据个体在问题域中的适应度值和从自然遗传学中借鉴来的再造方法进行个体选择，产生一个新的近似解。

3.3.1 遗传算法流程图

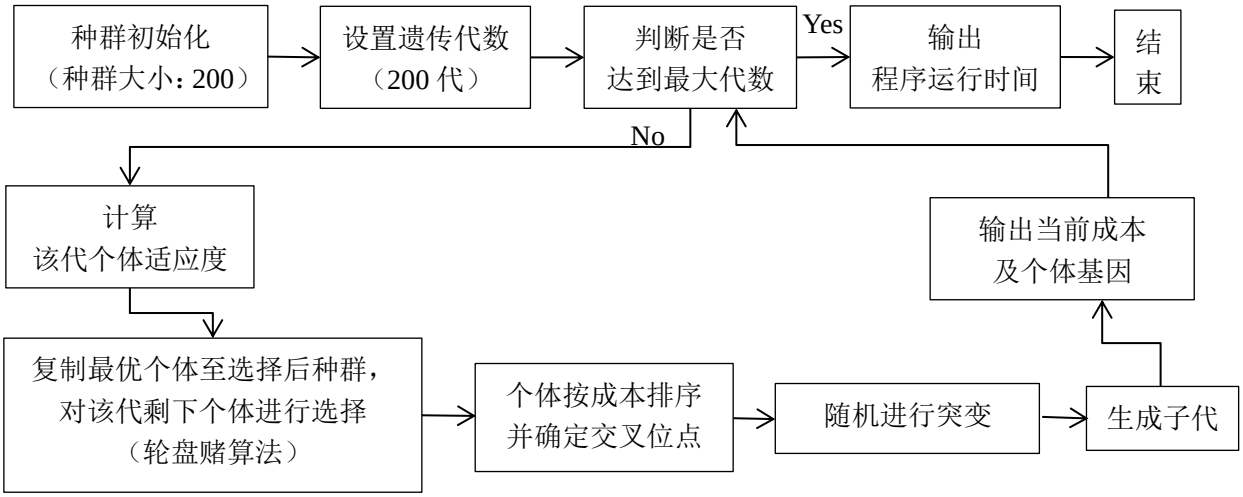

图 3-2 遗传算法流程图

3.3.2 遗传算法简述

染色体表示：初始染色体设置为 51bit 的二进制码 000....0，取点位置为 p 时，将特征值点所对应的位置上的值设为 1，我们设定种群初始大小为 200。

种群中个体中适应度的确定：对每一个特征值串进行三次样条插值及三次多项式拟合求得拟合函数后计算平均成本,将该代对每个个体选点方案算出的平均成本最大值的两倍减去对当前个体选点方案算出的平均成本的值作为适应度函数。

遗传算子：

（1）选择算子：使用轮盘选择算子。假设对该代所有 k 个个体取点方案计算出的平均成本分别为 $cost_1$，$cost_2$，……，$cost_k$，个体适应度为 s_i，令，s_i 与个体适应度正相关，然后在[0,1]区间上选取 k+1 个分界点 x_i，i=0,1,..,k，且 $x_0=0$，$x_k=1$，$Xi = \frac{\sum_{j=1}^{i} si}{\sum_{j=1}^{k} si}$，最后生成[0,1]上的随机数 r,若存在 $i\in\{0,1,2,...,k-1\}$使得 $x_i \leqslant r \leqslant x_i+1$,则第 i+1 个个体遗传到下一代。同时考虑到算子的随机性和变异算子、交叉算子的存在，强制性的将前一代中最优个体复制 1 个遗传到下一代。

（2）交叉因子：每一代交叉前根据不同个体得出的平均成本将所有个体排序，即将得出平均成本最小的个体置于种群矩阵第一行，平均成本其次的个体置于第二行……再按序交叉，第个个体交叉（）。

（3）变异因子：以一定的概率 p1=0.005 及 p1=0.01 来进行变异（一般选取变异概率在 0.001-0.1 之间）。在保证个体染色体有效长度不变的情况下，我们先生成随机数 r1∈{a1,a2...,ak}将 r1 所对应位置上的值设为 0，再生成随机数 r2∉{a1,a2,....,ak}将 r2 所对应位置上的值设为 1.

（4）设置遗传代数 genetation=200,选择算子，变异算子，交叉算子一次作用于下一代，最终输出结果几乎可以认为是在该取点个数下的最优解。

4 拟合方法

4. 1 三次多项式拟合方法

（1）确定一个可能解，对 k 个特征点组合 S={s1,s2,…,sk}对其试验，使用三次多项式拟合

$$u = a_1 d^3 + a_2 d^2 + a_3 d + a_4 \qquad (4\text{-}1)$$

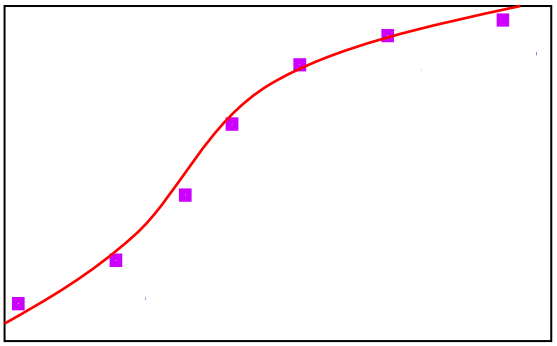

图 4-1 三次曲线拟合

（2）评估该曲线的拟合度。将特征点的坐标值代入上述三次多项式计算出相应的函数值，将计算值与实测值通过计算得出平均残差平方和，评价函数拟合度。

（3）用特定算法再次找另外可能解，继续进行（1）（2）步。不断优化当前解，直至达到终止条件。

4. 2 三次样条插值拟合法

（1）确定一个可能解，对 k 个特征点组合 S={s1,s2,…,sk}对其试验，使用三次样条差值拟合。具体方法为：首先对于非两端点，以四个连续点确定一条三次曲线，但仅在中间两点之间用该三次曲线表示，以此类推，所有非两端点之间均有三次曲线。两端点由端点处三个点用二次曲线拟合。

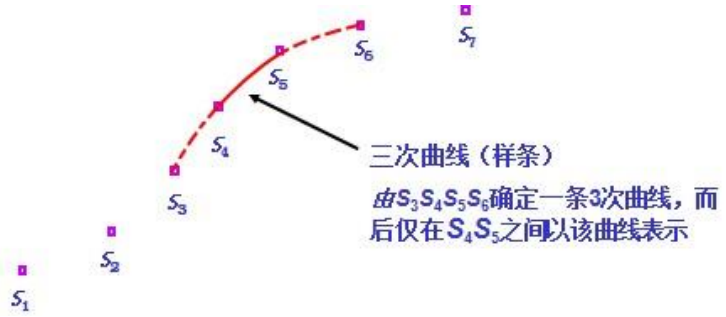

图 4-2 三次样条差值曲线拟合

（2）评估该曲线的拟合度。将特征点的坐标值代入得到三次样条差值后相应的函数值，将计算值与实测值通过计算得出残差平方和，评价函数拟合度。

（3）用特定算法再次找另外可能解，继续进行（1）（2）步。不断优化当前解，直至达到终止条件。

5 结果分析

5.1 模拟退火算法结果与取点个数、拟合方法的关系

表 5-1　模拟退火算法结果与取点个数、拟合方法的关系

拟合方法	数据点个数	特征值组合	成本	运行时间/秒
三次样条插值	5	{3、11、23、37、49}	108.30	78.101558
	6	{3、12、23、31、43、51}	96.70	78.434917
	7	{2、10、20、28、35、44、50}	95.50	77.707387
	8	{2、8、16、23、28、35、44、50}	102.89	78.612960
三次多项式拟合	5	{4、13、25、35、48}	114.95	367.929510
	6	{4、14、25、35、44、48}	125.10	367.438748
	7	{4、17、28、35、42、45、49}	137.71	366.378004
	8	{4、13、17、25、33、41、46、50}	148.11	370.202577

由表 5-1 可知，基于模拟退火算法，取点个数为 6 个或 7 个，拟合方法为三次样条插值方法最为合适，此时计算出的平均成本值最低。取点个数为 5 个和 8 个，拟合方法为三次多项式拟合时，平均成本都不太理想。根据实验结果可推测取 6 个点或 7 个点为最佳取点方案，样条插值法为较好的拟合方法。

5.2 遗传算法结果与取点个数、拟合方法以及变异概率的关系

5.2.1 遗传算法结果与取点个数、拟合方法的关系

表 5-2 遗传算法结果与取点个数、拟合方法的关系

拟合方法	数据点个数	变异概率	特征值组合	成本	运行时间（平均一代）
三次多项式	5		{3、17、30、40、49}	119.07	3 分 27 秒
	6		{4、14、25、34、43、49}	124.24	3 分 25 秒
	7		{4、13、21、28、36、44、49}	135.79	3 分
	8		{3、11、17、23、30、37、42、49}	147.16	4 分
三次样条插值	5	0.005	{3、14、28、40、49}	106.99	58.3
		0.01	{4、13、25、38、49}	107.14	59.5
	6	0.005	{3、12、22、31、42、50}	95.30	57.4
		0.01	{2、10、21、30、40、49}	95.49	63.8
	7	0.005	{3、9、19、26、34、43、50}	95.40	53.2
		0.01	{2、10、20、27、34、44、49}	95.80	64.7
	8	0.005	{2、10、19、25、31、37、45、51}	102.60	56.6
		0.01	{2、8、16、22、29、35、44、50}	102.95	54.6

由表 5-2 可得，基于遗传算法，在两种拟合方法下，数据点为 6 个和 7 个时，均成本最小且运行时间较短。而相比之下，三次样条插值方法比三次多项式方法成本小，运行时间短，故采取三次样条插值拟合方法。

5.2.2 遗传算法结果与变异概率的关系

表 5-3 变异概率对遗传算法结果的影响

数据点个数	变异概率	特征值组合	成本	运行时间（平均一代）
5	0.005	{3、14、28、40、49}	106.99	58.3
	0.01	{4、13、25、38、49}	107.14	59.5
6	0.005	{3、12、22、31、42、50}	95.30	57.4
	0.01	{2、10、21、30、40、49}	95.49	63.8
7	0.005	{3、9、19、26、34、43、50}	95.40	53.2
	0.01	{2、10、20、27、34、44、49}	95.80	64.7
8	0.005	{2、10、19、25、31、37、45、51}	102.60	56.6
	0.01	{2、8、16、22、29、35、44、50}	102.95	54.6

由表 5-3 可知，数据点个数一定的情况下，变异概率越大，最后计算出的成本越高，大体上运行时间变长。但由于三次多项式拟合不理想，且运行时间过长，故对变异概率讨论时基于三次样条插值方法。且遗传算法运行时间长，在此只取了变异概率为 0.005 和 0.01 的情况讨论，所取情况较少可能会导致结果出现一定偏差。

5. 3 模拟退火算法和遗传算法比较

表 5-4 模拟退火算法与遗传算法性能比较

数据点个数	搜索算法	特征值组合	成本	运行时间/秒
5	模拟退火算法	{3、11、23、37、49}	108.30	78.101558
	遗传算法	{3、14、28、40、49}	106.99	11656.392585
6	模拟退火算法	{3、12、23、31、43、51}	96.70	78.434917
	遗传算法	{3、12、22、31、42、50}	95.30	11486.381243
7	模拟退火算法	{2、10、20、28、35、44、50}	95.50	77.707387
	遗传算法	{3、9、19、26、34、43、50}	95.40	10642.375499
8	模拟退火算法	{2、8、16、23、28、35、44、50}	102.89	78.612960
	遗传算法	{2、10、19、25、31、37、45、51}	102.60	11316.896429

由表 5-4 可知，两种算法均在数据点为 6 个和 7 个时，成本最低，运行时间较短。模拟退火算法成本相比遗传算法较高，且结果带有较大的随机性，但程序运行所需总时间短；遗传算法得出的结果更优，计算出的成本更低，且每次实验得出的结果相近，结果的随机性不大。但程序运行时间过长，算法较为复杂。两种算法各有利弊，但均可得出较为优化的取点方案。

6 结论

根据多次实验所得结果，可以发现，遗传算法相对模拟退火算法可以得出更优解，但运行时间过长，模拟退火算法则可以大大减少求解时间，但结果没有遗传算法求解理想。在进行多次实验后，确定最佳取点方案为 3、12、22、31、42、50 的取点组合，确定最佳拟合方式为三次样条插值拟合。最终得出平均成本为 95.30。

7 拓展思考

7. 1 变异概率对遗传算法性能的影响

在进行遗传算法探索最优解时发现，改变变异概率后，解的收敛情况和最优解发生了显著改变。当变异概率为 0.005 时，迭代次数为 90~120 时，解出现收敛。当变异概率为 0.01 时，迭代次数为 120~150 时，解出现收敛。且在前者条件下得到的最优解普遍比后者条件下得到的最优解更优。虽然实验进行的较少，只选取了 0.005 和 0.01 两种变异概率，不具备普遍性。但结合理论分析，我们可以以一斑窥全豹。

理论上，变异算子是遗传算法中经常用到的遗传算子，它以很小的概率随机地改变染色体串上的某些位，虽然它属于遗传算法中的次要算子，但是它在恢复群体中失去多样性方面具有潜在的作用，可以增大种群的探索能力，防止迭代时陷入局部最优解。通常遗传算法实现变异的方法是赋予每一个基因一个相对比较小的变异概率，通过随机模拟而决定一个基因是否变异。变异概率过小使解有一定的局限性，遍历性差；变异概率较大使得进化的随机性增大，也不容易得到稳定的解。

当前使用的遗传算法在每一代对每一个体的变异概率保持恒定。其存在的问题在于：

（1）样本多样性丢失依然严重；

（2）变异概率确定只能依赖于经验选择；

（3）在进行变异的时机把握上存在欠缺，在有些情况下体现不出变异算子的作用，或者效果不太明显，甚至可能将出现的最优解淘汰。

这些问题导致了遗传算法的早熟收敛现象的出现，同时也有可能导致进化速度减慢，增加优化迭代次数。为了尽量避免这种情况的发生，可改进变异控制策略。

7.1.1 改进变异控制策略算法说明

进化过程中可能出现几种变异的情况，如果当前样本不存在早熟收敛的可能性，则取消本次变异操作；如果存在早熟收敛的迹象，则视样本多样性丢失的状况分别进行处理。如果样本的多样性保持较好，忽略本次变异操作；如果样本的多样性较差，进行变异操作；如果样本的多样性丢失严重，则既要进行变异操作，同时也要引入移民，加强变异操作。

7.1.2 变异时机的确定

（1）样本多样性丧失的判定

可构造如下判定条件。

对于采用二进制串表示的基因，个体的阶就是指个体中 1 或 0 的数目，表示为 Q(t,1)或 Q(t,0)。对于二进制串表示的基因，个体 t 的定义长度指个体中 1 或 0 的第一个确定位置与最后一个确定位置之间的距离，记为 W(t,1)或 W(t,0)。

定义假定第 t 代种群规模为 N，种群中不同个体的数目为 n，如果 $Q(t_1,1)= Q(t_2,1)=Q_c$ 且 $W(t_1,1)= W(t_2,1)=W_c$，则称这两个样本是相似的，$S(Q_c,W_c)$称为一个样式,种群的样式总数 s 称为群体的多样度，一般$0 \le s \le n \le N$。若$s/N \ge p_s$，称种群多样性较好；若$s/N < p_s$且$n/N \ge p_n$，称种群多样性较差；若$n/N \le p_n$，称种群多样性丢失严重。其中$p_s > p_n$，二者由经验值取得。

（2）早熟收敛的可能性判定

在实现时借助邻域的概念进行判定，

$$f(x_i) - \sum_1^m f(x_i)|/m \le \delta \tag{7-1}$$

则样本进入以$\sum_1^m f(x_i)/m$为中心，δ为半径的邻域。式中$f(x_i)$为适应函数；m为群体规模。在具体确定δ时，首先随机选择样本 x_1、x_2，则$\delta = |f(x_1) - f(x_2)|/2$；若所有的当前样本满足式(7-1)，则认为存在早熟收敛的可能性。

7.1.3 适当移民

在改进变异控制过程中，为了避免出现超级个体，在样本多样性丢失严重的情况下，适当的移民可以防止发生封闭竞争。在具体操作时，采用随机产生的“高品质”移民替代重复样本中的某个个体，使得进化过程更易跳出早熟收敛的区域[4]。

8 致谢

衷心感谢老师和助教的悉心指点和纠正！

9 参考文献

[1] 袁焱. 统计推断在模数转换系统中的应用课程设计课题和要求

[2] 百度百科 模拟退火算法

[3] 朱衡君主编 肖燕彩、邱成、齐红元编著 MATLAB 语言及实践教程（第 2 版）

[4] 张 烨、崔杜武、黑新宏、王若峻 一种改进变异控制策略的遗传算法（西安理工大学学报）

附页：

1. 答案检验程序

```
%test_ur_answer.m
%%%%%%% 答案检验程序 2015-11-04 %%%%%%%%

my_answer=[    3    12    22    31    42    50    ];%把你的选点组合填写在此
my_answer_n=size(my_answer,2);

%  标准样本原始数据读入
minput=dlmread('20150915dataform.csv');
[M,N]=size(minput);
nsample=M/2; npoint=N;
x=zeros(nsample,npoint);
y0=zeros(nsample,npoint);
y1=zeros(nsample,npoint);
for i=1:nsample
      x(i,:)=minput(2*i-1,:);
      y0(i,:)=minput(2*i,:);
end
my_answer_gene=zeros(1,npoint);
my_answer_gene(my_answer)=1;


%  定标计算
index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for j=1:nsample
      %  请把你的定标计算方法写入函数 mycurvefitting
      y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

%  成本计算
Q=12;
errabs=abs(y0-y1);

le0_4=(errabs<=0.4);
le0_6=(errabs<=0.6);
le0_8=(errabs<=0.8);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
```

```
sij=0.1*(le0_6-le0_4)+0.7*(le0_8-le0_6)+0.9*(le1_0-le0_8)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+
12*(le5_0-le3_0)+25*g5_0;
si=sum(sij,2)+Q*ones(nsample,1)*my_answer_n;
cost=sum(si)/nsample;

% 显示结果
fprintf('\n 经计算，你的答案对应的总体成本为%5.2f\n',cost);

%mycurvefitting.m
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

% 将你的定标计算方法写成指令代码，以下样式仅供参考
y1=interp1(x_premea,y0_premea,x,'spline');
end
```

2. 小组代码

（1）遗传算法

%GA.m

```
tic;
size_of_pop = 200;  %种群初始大小
size_of_sample = 7; %基因中编码为 1 的位数
pop = zeros( size_of_pop , size_of_sample );

for i = 1 : size_of_pop
    pop( i ,  :  ) = randperm( 51,size_of_sample );
end
pop=sort( pop,2,'ascend' );
cost_pop = zeros( 1 , size_of_pop );  %每个种群的成本

for i = 1 : size_of_pop
    cost_pop( i ) = cost_calc( pop( i , : ) ); %第 i 个种群的成本
end

for generation = 1 : 200
    selection;
    crossover;
    mutation;
    for i = 1 : size_of_pop
        cost_pop( i )=cost_calc( pop( i , : ) );
end
```

```
    min_pop = min( cost_pop );
    fprintf( '%2d %5.2f \n' , generation , min_pop );

    for i = 1 : size_of_pop
        if ( cost_pop( i ) == min_pop )
            for j = 1 : size_of_sample
                fprintf( '%2d     ' , pop( i , j ) ); %输出成本最低时所选择点的位置
                terminal_point( 1 , j ) = pop( i ,j );
            end
            fprintf( '\n' );
            break;
        end
    end
end
toc;
```

%selection.m　选择（采用轮盘赌算法）

```
fitness =   -cost_pop + 2*max( cost_pop ); %确定适应度函数
ppop = fitness / sum( fitness );    %各样本被选中概率
qpop = zeros( 1 , size_of_pop );

for i = 1 : size_of_pop
    qpop( i ) = sum( ppop( 1 : i ) ); %i 号样本及之前所有样本选中概率之和
end

size_of_tmp_pop = size_of_pop;
tmp_pop = zeros( size_of_tmp_pop , size_of_sample );
pmin = min( cost_pop );

for i = 1 : size_of_pop
    if ( cost_pop( i ) == pmin )
        break;
    end
end
%将当代最优个体复制一个至选择后种群
tmp_pop( 1 , : ) = pop( i , : );   %第一行为当前情况成本最低的样本

for i = 2 :   size_of_tmp_pop
    r = rand();
    for j = 1 : size_of_pop
        if ( qpop( j ) >= r )
            break;
        end
    end
```

```
        tmp_pop( i , : ) = pop( j , : );    %选择当代大于或等于 r 的第一个个体至选择后种群
end

pop = tmp_pop;
size_of_pop = size_of_tmp_pop;


%crossover.m    杂交
tmp_pop = [ pop';cost_pop ]';
tmp_pop = sortrows( tmp_pop,size_of_sample + 1 );
pop = tmp_pop( : , 1:size_of_sample );
for i = 1 : size_of_pop/2
    while 1
        r=randi([ 2 size_of_sample],1);
        tmp = pop( 2*i-1,r:size_of_sample );
        pop( 2*i-1,r:size_of_sample ) = pop( 2*i,r:size_of_sample );
        pop( 2*i,r:size_of_sample ) = tmp;
        if (( length(unique(pop( 2*i-1,:)))==size_of_sample) &&(length(unique(pop( 2*i,: )))
== size_of_sample))
            break;
        else
            pop( 2*i,r:size_of_sample ) = pop( 2*i-1,r:size_of_sample );
            pop( 2*i-1,r:size_of_sample ) = tmp;
        end
    end
end
pop=sort( pop,2,'ascend' );

%mutation.m    变异
pm = 0.005;    %设置突变概率为 0.005

for i = 1 : size_of_pop
      r = rand();
      if ( r < pm )
          %fprintf('%2d ',imutation(pop(i,:)));
            pop( i , : ) = inner_mutation( pop( i , : ) );
      end
end

%inner_mutation.m    变异实际实现函数
function new_result = inner_mutation( variable )
N = 51;
new_result = zeros( 1 , length( variable ) );
answer_gene = zeros( 1 , N );
```

```
answer_gene( variable ) = 1;
index = randi( N , 1 );
%编码为 1 的基因变为 0，编码为 0 的基因变为 1
if ( ismember( index , variable ) )
    answer_gene( index ) = 0;
    index = randi( N , 1 );
    while ( ismember( index , variable ) )
        index = randi( N , 1 );
    end
    answer_gene( index ) = 1;
%编码为 0 的基因变为 1，编码为 1 的基因变为 0
else
    answer_gene( index ) = 1;
    while ( ~ismember( index , variable ) )
        index = randi( N , 1 );
    end
    answer_gene( index ) = 0;
end

j = 1;
for i = 1 : N
    if ( answer_gene( i ) == 1 )
        new_result( j ) = i;
        j = j + 1;
    end
end
end
```

（2）模拟退火算法

```
%模拟退火算法
    size_of_sample = 8;
    A = randperm( 51 ); %随机打乱 51 个点
    B = sort(A( 1:size_of_sample )); %取打乱顺序后 A 中的前 5/6/7 个点进行排序
    min_score = 0; %总分数
    last_score = 0; %上一次成本
    cost = 0; %当次成本
    num = 0; %计数模拟次数
    B_min = B; %误差最小

    T = 0.01; %末温度
    T0 = 10; %初始温度

    tic;
    while ( T0 > T )
```

```
        num = num + 1;
        remain = setdiff( A,B ); %A,B 差集
        E = remain( randperm(51-size_of_sample) ); %打乱顺序
        F = randperm( size_of_sample );
        S = B;
        S( 1,F(1) ) = E( 1,F(1)+1 );
        S = sort( S );

        cost = cost_calc( S );
        if ( num == 1 ) %第一次计算成本
            min_score = cost;
            last_score = min_score;
            B_min = S;
            B = S;
        elseif cost < min_score %花费代价少于当前代价最小值
            fprintf('\n 经计算，第%2d 次成本为%5.2f\n',num,cost);
            min_score = cost;
            last_score = cost;
            B_min = S;
            B = S;
        elseif rand < exp( ( last_score - cost ) / T0 )
            last_score = cost;
          B = S;
        end
        T0 = T0 * 0.98;
    end;
    toc;
    fprintf('\n 经计算，最终总体成本为%5.2f\n',min_score);
    fprintf('%2d ',B_min);
    B_min;
```

(3) 成本计算函数及曲线拟合函数

%cost_calm.m 成本计算

```
function cost = cost_calc( gene )
size_of_gene = length( gene );  %基因编码中 1 的数目
data = csvread( '20150915dataform.csv' );
[M,N] = size( data );
x = zeros( M/2,N );  %原样本数据的横坐标
y0 = zeros( M/2,N ); %原样本数据的纵坐标
y1 = zeros( M/2,N );
for i = 1 : M/2
    x( i , : ) = data( 2*i-1 , : );
    y0( i , : ) = data( 2*i , : );
end
```

```
gene_seq = zeros( 1 , N ); %51 个基因
gene_seq( gene ) = 1;
index_tmp = logical ( gene_seq ); %选择值为 1 的列
x_optimal = x( : , index_tmp );
y0_optimal = y0( : , index_tmp );
for i = 1 : M/2
      y1( i , : )=curvefitting 1( x_optimal( i , : ) , y0_optimal( i , : ) );
end
Q = 12; sij = 0;
D_value = abs( y0-y1 );
cost1 = ( D_value <= 0.4 );
cost2 = ( D_value <= 0.6 );
cost3 = ( D_value <= 0.8 );
cost4 = ( D_value <= 1 );
cost5 = ( D_value <= 2 );
cost6 = ( D_value <= 3 );
cost7 = ( D_value <= 5 );
cost8 = ( D_value > 5 );
sij = 0.1 *( cost2 - cost1 ) + 0.7 *( cost3 - cost2 ) + 0.9 * ( cost4 - cost3 ) + 1.5 *( cost5 -
cost4 )+ 6 * ( cost6 - cost5 ) + 12 *( cost7 - cost6 )+ 25 * cost8;
si = sum( sij , 2 ) + Q * ones ( M/2 , 1 ) * size_of_gene;
cost = sum( si ) / ( M/2 );    %成本
end
```

%curvefitting1.m　得出三次样条插值拟合值

```
function y = curvefitting1( x_value , y0_value )
x = ( 5.0 : 0.1 : 10.0 );
y = interp1( x_value ,y0_value , x , 'spline' );
end
```

%curvefitting2.m　得出三次多项式拟合值

```
function y = curvefitting2( x_value , y0_value )
x = ( 5.0 : 0.1 : 10.0 );
y1 = polyfit( x_value ,y0_value    , 3 );
y = polyval( y1 , x );
end
```