统计推断在数模模数转化中的应用

组号：06　　姓名：谢昊男(组长) 学号：5130309135　姓名：徐冬　学号：5130309141

摘要：作者通过模拟退火算法和统计方法对大数据量进行分析，为某型产品内部的一个监测模块，寻求校准工序的优化方案。用模拟退火算法得到样本中的最优解，从而提高所取点的拟合程度。模拟退火算法能一定程度上降低生产的成本，提高生产效率。
关键词：拟合；模拟退火算法

Application of Statistic Inference in A/D & D/A Conversion System

ABSTRACT: The author used the simulated annealing algorithm and statistical methods to analyze large amount of data, in which to find the optimized method for the calibration process a certain type of products within a monitoring module, Using simulated annealing algorithm to get the optimal solution sample, so as to improve the fitting degree of what we take.Simulated annealing algorithm to a certain extent, reduce the production cost, improve production efficiency.

Key words:fitting; Simulation Annealing Algorithm

1. 引言[1]

在实际生产中，由于每一个器件存在个体差异，所以在产品出厂时需要对每一个特定的器件进行校准和标定，这样使用者才能方便的使用这样的器件。在器件批量生产时，对每个器件都进行完全精确的标定固然是对用户来说时最好不过的，然而这样必定会使生产成本加大，对厂家和对用户都不好。有没有一种方法，能够在不影响用户使用器件的同时，能够大大减小对器件标定所产生的成本呢？我们可以设想存在一种函数对应关系，能够在测量少数点的的情况下，就能推知所有需要标定的值（在误差允许范围内）。

2. 研究目的和背景[4]

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。而在生产时每个元件个体存在差异，若是对所有 469 个样本逐个进行定标，则增加很多成本。于是我们便探索如何寻求更高效的定标方法？

每组数据共 51 个点，减少测量点数势必可以直接降低定标成本，但同时，也会导致所绘出的曲线与实际测量得到的曲线相差较大。所以，我们实验的方向是在减少测量点的基础上，同时保证所拟合出来的曲线与实际曲线很接近。

实验时，假定选取 7 个点来推定所有点分布情况时，定标成本最小。 在后续实验对比中，我们还将找出实际情况下，选取几个点成本最优，以及在什么条件下能更好地降低成本。研究方法框图如下：

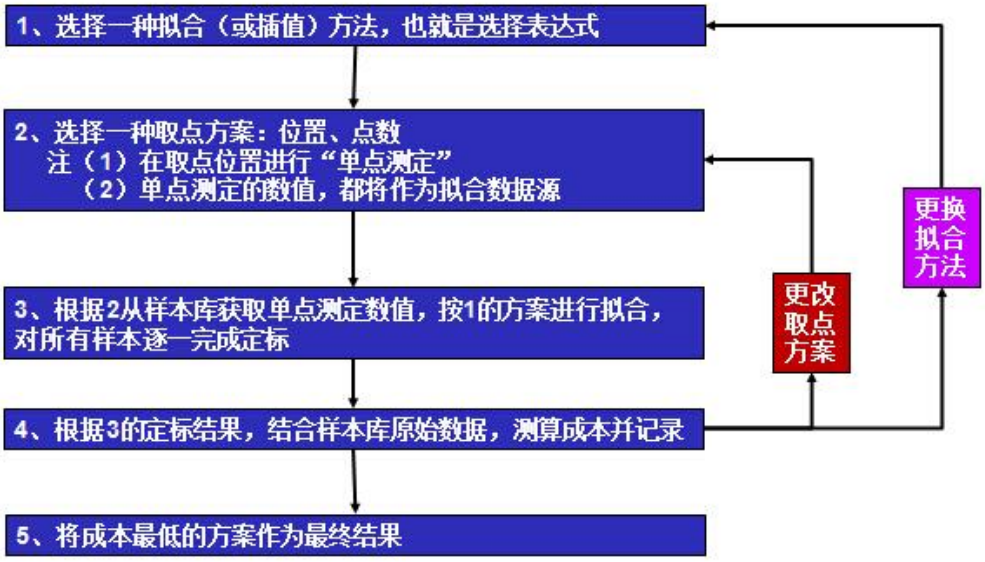

图 2.1 研究方法框图[2]

3. 算法的选取

算法是该实验程序运行的基础，通过算法的实现，找出一共 51 组数据中，每一组合适的数据，并评测其成本。

3.1穷举法[1]

数学中，很多问题都涉及到寻求最优解。有的问题，可行解数目较少，比如只有 2^3=8 中可行解。此时，只需要逐一列举，并进行比对就可以得出结论找出最优解。但是对于本实验，由于数据过多，如果使用穷举法，选取 7 个测试点时，计算机必须要完成 C_{51}^{7} 次拟合，才能找出最优解。如此巨大的数目，要求解多项式，即使是计算机处理起来也将费时费力，甚至无法完成。（NP-hard 问题）因此，必须要选取更加合理高效的算法，使用统计知识求解。

3.2 遗传算法（GA）[1]

遗传算法是借鉴生物界自然选择和群体进化机制形成的一种全局寻优算法。与传统的优化算法相比，遗传算法具有如下优点：

①.不是从单个点，而是从多个点构成的群体开始搜索；

②.在搜索最优解过程中，只需要由目标函数值转换得来的适应值信息，而不需要导数等其它辅助信息；

③.搜索过程不易陷入局部最优点。

算法实现过程：

（1）随机选择亲代样本，对其进行评估

（2）筛选掉适应度较低的样本，留下适应度高的样本

（3）对适应度高的样本进行交叉配对，产生第二代

（4）从整体中随机取得另一部分样本加入第二代

（5）重复以上步骤，不断进行优胜劣汰直至得到最优解。

遗传算法难以控制最优解的点数，不利于拓展试验的进行，此外遗传算法所需时间较长，而且代码难度较高，实现起来较为繁琐，故该算法不作为本实验的首选算法，若时间允许仍可作为算法对比的尝试。

3.3 模拟退火算法（SA）[1]

模拟退火算法是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。

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。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。

算法实现过程：

（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：产生新解 S′

（3）(计算增量Δt′=C(S′)-C(S)，其中 C(S)为评价函数

（4）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.

（5）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。

（6）T 逐渐减少，且 T->0，然后转第 2 步。

以概率接受某点，从某种程度上避免了局部最优解，此外，为更加有效的避免局部最优解，我们组在对比实验中添加了恒温过程的对比。

相比于遗传算法，模拟退火算法可以轻易控制测试点的个数，便于对比试验的进行。此外，该代码运行时间快，效率高，而且代码简单有效，故我们组将模拟退火算法作为首选算法。

4. 拟合方法选取与讨论

拟合，就是用最接近的函数的关系，去表征数据点所反映的输入输出关系。拟合方法选取是否得当直接影响了实验结果是否能够具有良好的适应度。不同的拟合方法有不同优劣。下面初步就三次多项式拟合法和三次样条插值法的实现过程作介绍。

4.1 多项式拟合[1]

多项式曲线拟合在实际中应用广泛，若用最小二乘法拟合，通常利用微机进行矩阵运算能快捷方便地予以处理,多项式拟合的表达式见图 4.1.1：

$$y = p_0 x^0 + p_1 x^1 + p_2 x^2 + \cdots + p_n x^n$$

图 4.1.1 多项式拟合示意图[1]

通过曲线的变化规律，很容易推断出，采用多项式进行拟合是合理的。但当次数增加时函数拟合速度会变慢，并且多项式拟合的阶次过大时在数值上可能拟合效果较好，但会引起拟合曲线的震荡，拟合出来的曲线与事实相差较大。在本课程中我们只讨论阶次较小、且能够拟合曲线变化规律的三次多项式拟合法。

选取 7 个合适的点，通过三次多项式拟合得到一条拟合曲线，对比曲线中的每一个点与实验测的数据，通过评测函数计算其成本，同时考虑其所需时间。

4.2 三次样条插值[1]

三次样条插值法对于中间部分的相邻 4 个点采用三次曲线拟合，并在中间两点处做出拟合曲线；对于边界部分的 2 个点，选取与之靠近的三个点采用二次曲线拟合，

并在靠近边界的两点处做出拟合曲线。如此操作，在断点处斜率和曲率都将连续。（见图 4.1.2）

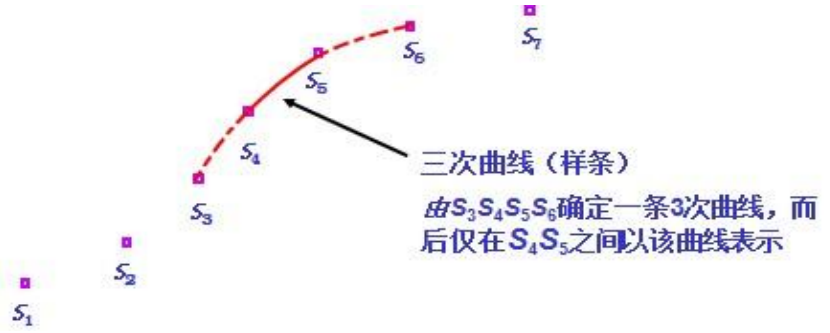

图 4.1.2 样条插值示意图[1]

选取 7 个相同的点，通过三次样条插值拟合得到一条拟合曲线，对比曲线中的每一个点与实验测的数据，通过评测函数计算其成本。对比两种拟合方式所得成本，及其所需时间，便可评测出拟合方式的优劣。

5 实验内容

5.1 方法的确定

首先，选择一种取点方式。穷举法代码实现较为简单，但代码运行起来耗费大量时间空间，效率很低，故不予考虑；遗传算法代码较为复杂，且代码运行所需时间较多，故经综合考虑，我们组选择了模拟退火算法作为取点方式。

其次，选择一种拟合方式。我们随意选择了一组等间距的 7 点取样，通过样条插值和四次多项式拟合两种拟合方式，分别测试了成本，发现样条插值法更加精确，同时所需时间更少，故我们选择样条插值法作为首选拟合方法。此外，不同拟合方法的精确度我们会以对比实验的方式展示出来。

最后，确定首选点数。由于该传感器部件的输入输出特性呈分段线性，一共分为三段，故猜测 5、6、7、8 点时，拟合结果精确且总成本较低。本次实验主体部分假设 7 个点时总成本最低，并因此开展实验，同时，会在对比实验部分找出最合适最有效的取点个数。

5.2 拟合方法及其实现

通过对 matlab 的学习，发现样条插值拟合法和多项式拟合法都有固定的内置函数，分别为：y1=interp1(x_premea,y0_premea,x,'spline')[3]；p=polyfit(X,Y,n)[3]。内置函数的直接实现为探索哪种拟合方法更有效提供了有效的支持。

5.3 取点算法及其实现

(1) 通过 xlsread[3]函数，读入老师提供的 469 组数据，并将表格中的 x 和 y 分别取出；

(2) 随机函数随机排序并选取前七个点，作为初始选点方案。再次排序，得到随机的几个点；

(3) 用模拟退火算法作为大循环，通过对退火初始温度和末温度的设定，以及每次降温幅度的设定，确定函数主体部分执行次数；

(4) 在循环内前部分加入随机变化一个点，生成新的七个点组合。并用样条插值拟合，得到每点处拟合曲线的值，同时算出拟合值与实验值的差值；

(5) 循环中间部分加入 51 个点的评价函数，计算并评测出当前取点方案的成本值。并计算 469 组数据的平均成本值，作为这次取点的总成本数；

(6) 循环的后半部分决定是否接受这个取点方案。若计算成本分值小于当前最小成本分值，则接受该方案，同时将该方案作为当前最优方案，下次取点时以这个方案为基础；若大于，则以 exp((score_save-cost)/Tk)的概率接受该方案。其中 score_save 表示上一次接受的成本，cost 表示当前成本，Tk 为温度；

(7) 输出最优解及其所花费时间。

5.4 算法框图：

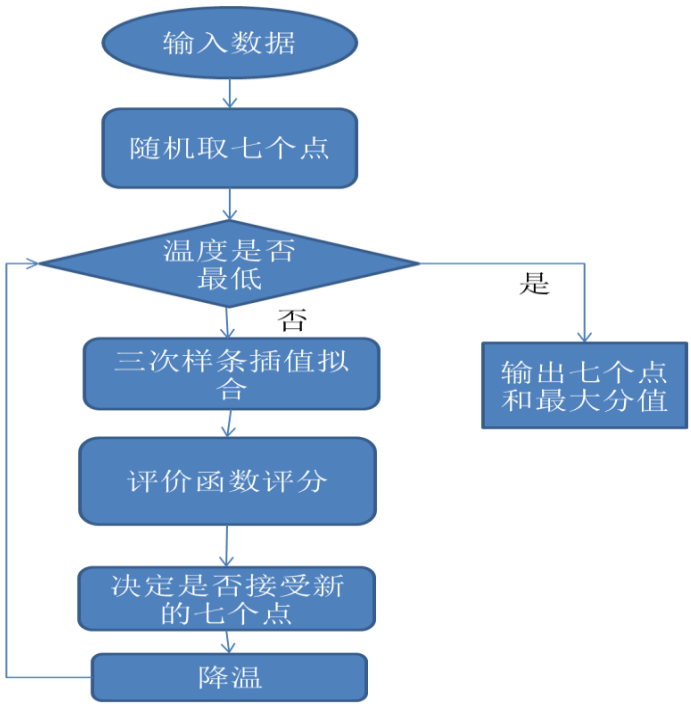

图 5.4.1　模拟退火算法框图[1]

5.5 算法分析

温度的设定。设置初始温度为 100℃，末温为 0.01℃，降温幅度为 0.97，仅通过降温，可进行约 300 组实验数据的测定，能充分找到最优解。在对比实验中改变初始温度，或者增加恒温过程，使实验数据测定量增加，来探究能否找出更有效的取点，使总成本降低。

局部最优解的避免。在每次产生新的 7 个点时，算法中实现的机制是随机替换掉最优取点中的某一个，这样做在取点次数累计的过程中，可以使每次取点的成本逐渐降低，趋近于最优解，同时更容易找到最优解。若在该点附近取点进行替换，则很容易造成局部最优解，而用一个随机的点进行替换，从某种程度上降低了局部最优解的概率。此外，算法中添加依概率接受某一点这一步，使即将陷入局部最优解的取点可以随时引入新的有效点，产生新的取点方案，从而更加不易产生局部最优解。此外，为了说明模拟退火算法不易产生局部最优解，对比实验中引入了恒温过程，或者设定一个初始值再进行实验。

运行效率。从代码角度分析，主代码为一个 while 循环，循环中除了拟合和评价函数，只有一些 if 语句判定，整个代码简单易懂而且十分有效，运行时间比较短，故运行效率较高。

评价函数。评价函数由老师提供，能计算不同取点个数时的成本，而且简单有效，这为实验及相关对比试验的进行打下了基础。

6 实验结果及分析

6.1 程序运行结果展示

编号	运行结果							成本	时间
1	3	10	19	27	31	43	49	95.55	219.570869
2	2	8	20	25	34	44	50	95.06	223.468040
3	3	10	20	27	35	44	51	94.58	238.080953
4	3	10	18	26	33	43	50	94.69	246.862352
5	2	10	19	26	34	45	50	94.86	213.213227
6	3	11	19	28	34	44	50	95.35	225.011382

表 6.1 样条插值拟合 7 点模拟退火

6.2 实验结果分析

由运行的结果可以看出，样条插值 7 点模拟退火的运行成本最低大概为 94.58，最高大概为 95.55;运行时间最少大概为 213.213227 秒,最多大概为 246.862352 秒。

由实验结果可以看出，7 个点大概可以较为精确的模拟出监测模块中传感器部件的输入输出特性，且用时较短，大概不到 4 分钟就能退火完成。可是 7 个点一定是最优的取点个数吗？多项式拟合算法是不是更为精确呢？增加初始温度是否可以更加降低成本呢？降低了初始温度是否成本就一定增加呢？带着这些疑问，且为了使实验结果更加优化，我们做了许多对比实验。

7 实验拓展研究

7.1 三次多项式拟合和样条插值拟合对比分析

7.1.1. 分析原因

我们组在基础探究部分发现，用高次多项式逼近的拟合算法同样可以精确的得出拟合曲线，于是我们想探究那种方法更为有效。

7.1.2. 程序核心部分

```
[p,v]=polyfit(X,Y,3) [3];
y1=interp1(x_premea,y0_premea,x,'spline') [3]
```

7.1.3. 程序运行结果

编号	运行结果							成本	时间
1	4	14	22	30	39	44	49	133.00	298.242581
2	4	13	22	30	39	41	48	133.93	282.639608
3	4	15	20	28	39	47	50	135.07	279.632727

表 7.1.3 三次多项式拟合 7 点模拟退火

7.1.4 结果分析

由表 6.1 与表 7.1.3 对比可知，三次多项式拟合算法拟合结果误差较大，所得成本较高，从某种程度上讲，三次多项式拟合不能成为该定标实验的实验拟合方法。从时间角度分析，三次多项式拟合耗费时间稍微多一些，整体上与样条插值拟合相差不多。故，三次样条插值拟合更加细腻精确，其性能远优于三次多项式拟合。

7.2 三次多项式拟合和四次多项式拟合对比分析

7.2.1. 分析原因

用三次多项式法不够精确，所以我们组尝试，用四次多项式法是否能更精确，总成本是否能更低，是否优于三次样条插值算法。

7.2.2. 程序核心部分

[p,v]=polyfit(X,Y,n)[3]; （n 取 3 或 4）

7.2.3. 程序运行结果

编号	运行结果							成本	时间
1	2	10	20	28	36	46	50	115.79	409.170717
2	2	7	20	29	37	43	49	116.02	422.094916
3	1	4	13	23	32	43	49	117.08	431.676546

表 7.2.3 四次多项式拟合 7 点模拟退火

7.2.4 结果分析

由表 7.2.3 与表 7.1.3 对比可知，四次多项式拟合算法比三次多项式拟合要精确许多，但与此同时，耗费的时间也要长许多。相比于三次样条插值法，四次多项式拟合成本依旧很高，而且耗时也更长，总的代价较大，故不必再考虑更高次项的多项式拟合了，三次样条插值为最精确，同时最省时的拟合算法。

7.3 不同初始温度下的探究

7.3.1. 分析原因

由于初始温度设为 100 度，末温度设为 0.01。我们小组思考当温度改变时会不会对实验结果产生一定的影响，为此，我们将温度改成 10 度，或者 500 度，分别进行了实验。理论上，温度降低将减少退火次数；反之温度升高会增加退火次数。

7.3.2. 程序运行结果

编号	运行结果							成本	时间
1	1	7	20	28	33	44	51	96.45	171.269564
2	2	6	17	25	34	44	51	96.79	174.240129
3	1	8	20	28	33	43	50	95.32	172.184732

表 7.3.2.1 初始温度为 10 度时 7 点模拟退火

编号	运行结果							成本	时间
1	2	8	20	26	32	44	49	95.18	265.006265
2	2	11	21	26	33	43	50	94.47	258.991079
3	3	8	20	26	33	43	50	94.33	265.133814

表 7.3.2.2 初始温度为 500 度时 7 点模拟退火

7.3.3 结果分析

由表 7.3.2.1、表 7.3.2.2 与表 6.1 对比可知：10 度时，随着退火次数的减少，找到的解确实没有 100 度时的解成本那样低，普遍成本要高出 1 左右，但所花费时间要更少，故对时间效率要求较高时可以考虑进行适当降温处理；500 度时，随着退火次数的增加，找到的解确实比 100 度时的解成本低一些，但低的不多，成本普遍比较接近，可是所花费时间要多一些，故对成本要求较高时可以考虑进行适当升温处理。

7.4 六个点和七个点拟合效果分析

7.4.1. 分析原因

由拓展实验2可以看出，三次样条插值拟合的效果好于三次拟合和四次拟合，那么针对三次样条插值，为何非要取到七个点呢？六个点是否能更加有效的降低成本呢？为此，我们对六个点的三次样条插值法进行了实验。

7.4.2. 程序运行结果

编号	运行结果						成本	时间
1	3	11	21	31	42	50	93.29	194.245806
2	4	11	23	31	43	50	94.73	197.925836
3	3	10	21	30	42	49	94.21	194.766841
4	2	10	21	30	41	49	93.46	196.107749
5	2	11	21	30	41	50	93.41	178.421250
6	2	10	22	31	44	50	94.32	194.657024

表 7.4.2 样条插值拟合 6 点模拟退火

7.4.3 结果分析

由表 7.4.2 与表 6.1 对比可知，用 6 个点运行模拟退火算法，总成本更低，普遍成本比 7 个点时低 1 左右，所耗费的时间也更少一些，所以 6 个点效果更优。故否定了之前的假设，7 个点不一定是效果最优的取点方式，6 个点效果更好。

结果分析：由于曲线呈分段线性特性，且一共分为三段，由两点确定一条直线的原理，一共确定 6 个点就能较为精确的将曲线趋势表现出来；此外，减少一个测试点能较大的降低总成本，故 6 个点从原理上成本可以优于 7 个点。

7.5 五个点和八个点拟合效果分析

7.5.1. 分析原因

由上一个对比试验可以继续思考，6 个点就一定是最好吗？5 个点或者稍微多一点，取 8 个点，成本是不是会更低呢？由于曲线呈分段线性，故将其中一个点取在转折点附近，理论上也能准确的表征曲线趋势，故 5 个点理论上也是可行的；此外，取 8 个点能更加精确的描述曲线，成本也可能降低。

7.5.2. 程序运行结果

编号	运行结果					成本	时间
1	4	14	26	39	49	104.30	196.437556
2	2	14	26	38	49	106.48	192.929100
3	4	14	27	39	49	104.15	173.058865

表 7.5.2.1 三次样条插值拟合 5 点模拟退火

编号	运行结果								成本	时间
1	2	9	20	26	33	40	45	51	102.08	232.544510
2	1	8	17	23	30	37	46	49	103.69	229.106652
3	2	9	19	25	30	35	44	49	102.56	227.536604

表 7.5.2.2 三次样条插值拟合 8 点模拟退火

7.5.3 结果分析

由表 7.5.2.1、表 7.5.2.2 与表 6.1 和表 7.4.2 对比可知：取 5 个点和取 8 个点都会使总成本增加，且增加幅度较多，就时间而言，5 个点与 6 个点时间接

近；8 个点与 7 个点时间接近。相比较这四种取点个数，6 个点成本最优，且耗时最少，故 6 个点为最合适的取点方法。少于 5 个点或者大于 8 个点，理论上成本只减不增，故不再考虑。

7.6 固定初始值效果分析

7.6.1. 分析原因

经试验发现，大部分的最优取点都有一定的规律性，于是我们组思考是不是陷入了局部最优解。为了分析这个问题，我们首先将随机取初始点改为，初始取定点 1，2,3,4,5,6,7 这 7 个点，这样若陷入局部最优解，则结果应与最初结果相差较大。

7.6.2. 程序运行结果

编号	运行结果							成本	时间
1	2	7	20	28	34	44	51	95.87	193.997014
2	2	10	20	28	35	45	51	94.99	228.113039
3	4	8	19	26	34	46	50	96.91	231.569703

表 7.6.2 固定初始值 7 点模拟退火

7.6.3 结果分析

由表 7.6.2 和表 6.1 对比，忽略成本及时间变化的随机性，成本值的分布范围基本相同，时间花费也相差无几。可以看出，固定初始值得到的成本数据和未固定初始值的数据非常相似，可以判定：最初点的取值对最终结果并无影响；且可初步认定并没有陷入局部最优解的情况。

7.7 模拟退火算法改进——加入恒温过程

7.7.1. 分析原因

上一个对比试验中，已经初步认定了此算法没有陷入局部最优解，为增加说服力。我们组又对算法进行了改进，增加了恒温过程。

恒温过程的作用：

在分析中发现，由于温度是递减规律的，一开始温度较高，则相应 rand< exp((score_save-cost)/Tk)事件发生概率较大，故有较大的可能性取到很多不同的组合差别较大的点；但是当温度降到一定程度时概率会降低，一旦达到一个较优解，就很有可能陷入局部换点但不改变整体的困境。为此，需要在达到温度很低之前就跳出局部最优解。于是我们在实验中引入了恒温过程。

加入恒温过程理论上会增加代码运行时间，但却可以验证是否存在局部最优解的情况。若存在，则找到的取点方案成本比没有恒温过程要低。

7.7.2. 程序核心部分

在原主体循环部分加一个循环：循环条件为 n<Tk/10,n 初始为 0，每次循环自动加 1。这样便增大了在温度较大时的循环次数，达到恒温退火的效果。

7.7.3. 程序运行结果

编号	运行结果							成本	时间
1	4	9	20	27	34	44	49	95.48	368.692931
2	2	10	20	26	32	44	48	96.03	358.899718
3	2	10	20	27	34	44	49	94.47	371.421075

表 7.7.3 样条插值拟合 7 点模拟恒温退火

7.7.4 结果分析

由表 7.7.3 与表 6.1 对比可知，加入了恒温过程计算出的成本与不加恒温过程的成本分布相似，可知原实验的取点并未陷入局部最优解。此外，加入恒温过程增加了取点个数，增加了程序运行时间，使效率反而降低了。故实验时不必添加恒温过程。

8 实验总结

通过实验本身以及上述所有对比试验，我们发现，在当前的评价函数基础上，6 个点是成本最低的取点个数，三次样条插值法是最合适的拟合方法，从达到最优解同时节省时间两方面考虑，不用对温度进行升温或者降温操作，且该算法不易陷入局部最优解中，无须固定初始值，或者加入恒温退火过程。

故采用模拟退火算法选择 6 个点取样，运用三次样条插值法进行数据拟合，是最节省成本，最高效的定标方式。

9 参考文献

[1] 网络资源. http://baike.baidu.com/

[2] 袁焱. 统计推断讲座 3：问题求解的途径

[3] 袁东，肖广兵. 详解 MATLAB 快速入门与应用

[4] 袁焱. “统计推断”课程设计的要求 V2.1 2014-11-22

10 附录

程序基础部分代码：

```
DATA=xlsread('20141010dataform.csv');%读入数据表中的数据
x=DATA(1:2:end,1:end);%将表格中的x分别取出
y=DATA(2:2:end,1:end);%将表格中的y分别取出

A=randperm(51);%随机打乱51个点
B=sort(A(1:7)); %随机取7个点，并进行排序
score_min=0;%总分数
score_save=0;%保存上一个数据成本
cost=0;%保存当次成本
num=0;%计数模拟的次数
B_min=B;%记录误差最小的情况

Tf=0.01;%末温度
Tk=10;%初始温度

tic;%开始运行记录时间
while Tk>Tf
    num=num+1;
```

```
remain=setdiff(A,B);%寻求A,B差集
E=remain(randperm(44));%再次打乱顺序
F=randperm(7);
S=B;
S(1,F(1))=E(1,F(1)+1);
S=sort(S);%以上代码用于随机替换7个数中的一个数字

%%%%%%%%以下代码用于计算当前取点的成本，取自老师给的测试函数%%%%%%%
my_answer=S;%当前选点组合
my_answer_n=size(my_answer,2);

% 标准样本原始数据读入
minput=dlmread('20141010dataform.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);%选点处的x
y0_optimal=y0(:,index_temp);%选点处的y
for j=1:nsample
    % 通过调用mycurvefitting函数，实现三次样条插值拟合
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

% 成本计算
Q=12;
errabs=abs(y0-y1);
le0_5=(errabs<=0.5);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
```

```
sij=0.5*(le1_0-le0_5)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+12*(le5_0-le3_0)+25*g
5_0;
    si=sum(sij,2)+Q*ones(nsample,1)*my_answer_n;
    cost=sum(si)/nsample;%cost为当前的总成本
    fprintf('\n经计算，当前总成本为%5.2f\n',cost);
    %%%%%%%%%以上代码用于计算当前取点的成本，取自老师给的测试函数%%%%%%%
    if num==1 %第一次计算成本
        score_min=cost;
        score_save=cost;
        B_min=S;
        B=S;
    elseif cost<score_min %花费代价小于上一次
        fprintf('\n经计算，当前最优总成本为%5.2f\n',cost);
        score_min=cost;
        score_save=cost;
        B_min=S;
        B=S;
    elseif rand<exp((score_save-cost)/Tk)%决定点是否变化
        score_save=cost;
        B=S;
    end
    Tk=Tk*0.97;%降温
end
toc;%记录结束的时间
fprintf('\n经计算，最终最优总成本为%5.2f\n',score_min);
B_min%输出最优取点方案
```