统计推断在数模转换系统中的应用

组号 48　　肖弼 5130309590，贾浩 5130309546

摘要：本文通过启发式搜索并结合统计数学的方法，对传感器的非线性特性进行了研究，进而对传感器性能进行定标校准。本课题是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。通过启发式搜索，根据已有样本数据库寻找最优测试点的个数和位置以减少传感器定标成本

关键词：传感器，定标，遗传算法，模拟退火

Application of Statistic Model in A/D and D/A Converse System

ABSTRACT： By combining heuristic search and method of Statistical Mathematics, the nonlinear characteristic curve of a kind of sensor was studied in this report, and then calibrate performance of the sensor. This issue is to design a calibration program that has a reasonable cost to make mass production of the sensor. By heuristic search, based on the existing database to find the optimal number and location of test points in order to reduce the cost of the sensor calibration

Key words: Sensor, Calibrate, Genetic Algorithm, Simulated Annealing

1 引言

对传感器密集选点测试，连点成线可以得到其特性曲线。由于制造精度所限，即使同一批次的产品，其特性曲线不完全相同。为了校准传感器，即对传感器进行定标，必须对每个传感器进行测量，那么就要寻找合适的测量方法来简化定标方案。为了简化定标方案可以密集选点测试，测量出部分样品的特性曲线进行统计分析。

2 某传感器的定标

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。实验室中，我们可以逐点测定每个电压设点对应的定标成本，但是涉及到大批量的工业生产时，逐块测定势必会增加生产成本，降低可行性。那么是否可以通过几组事前观测值来完成对系统的定标？

2.1 定标方案概述

定标方案的成本包括测试成本及误差成本。为了降低定标的成本，可以通过减少测试点数量，通过已有测试点来估计其他点的值，这样可以减少测试成本。但如果测试点太少，估算值误差势必会增大，从而造成误差成本过高。在确定测试点数量的情况下，测试点的位置也应当谨慎选取，因为不同的测试位置得到的最终结果也不会完全相同。另外，估算其他测试点值的估算方法也对成本有影响，应当选取合适的估算方法。综上所述，一个完整的定标方案包括：测试点的数量和位置以及对特性曲线的估算方法。最佳定标方案即使定标成本最小的方案。

2.2 定标方案的评定

为评估和比较不同的校准方案，特制定以下成本计算规则。

(1)单点定标误差成本

$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.5\\ 0.5 & \text{if } 0.5<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases} \tag{1}$$

单点定标误差的成本按式（1）计算，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。

(2)单点测定成本

实施一次单点测定的成本以符号 q 记。本课题指定 q=12。

(3)某一样本个体的定标成本

$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i \tag{2}$$

对样本 i 总的定标成本按式（2）计算，式中 n_i 表示对该样本个体定标过程中的单点测定次数。

(4)校准方案总体成本

按式（3）计算评估校准方案的总体成本，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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \tag{3}$$

总体成本较低的校准方案，认定为较优方案

2.3 数据来源

前期已经通过试验性小批量生产，制造了一批传感部件样品，并通过实验测定了每个样品的特性数值。这可以作为本课题的统计学研究样本。数据被绘制成表格，称为本课题的“标准样本数据库”。

3 定标方案设计方法

定标方案包括测试点数量和位置的选取以及估算方法的选取。显然测试点的选取依赖于估算方法，因此首先选定一种估算方案，在此条件下选择测试点的数量及位置。最后通过尝试不同的估算方法来确定最佳定标方案。

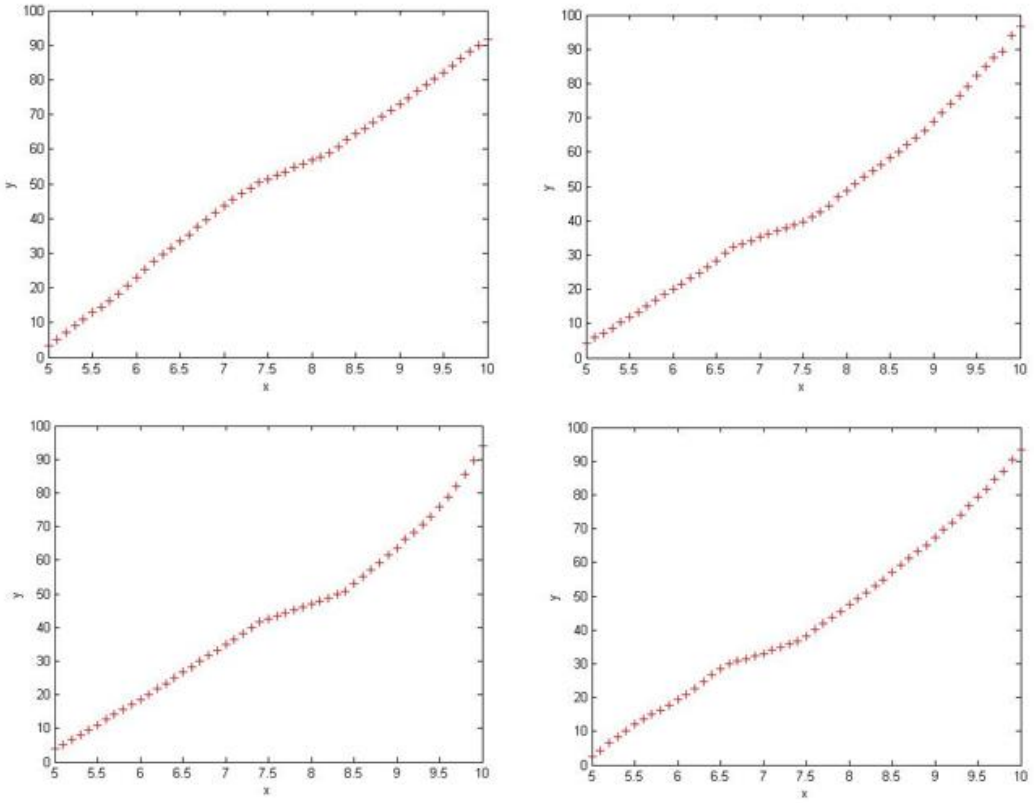

图 1 传感器特性曲线

通过观察发现，一个传感部件个体的输入输出特性有以下主要特征：Y 取值随 X 取值的增大而单调递增；X 取值在[5.0,10.0]区间内，Y 取值在[0,100]区间内；不同个体的特性曲线形态相似但两两相异；特性曲线按斜率变化大致可以区分为首段、中段、尾段三部分，中段的平均斜率小于首段和尾段；首段、中段、尾段单独都不是完全线性的，且不同个体的弯曲形态有随机性差异；不同个体的中段起点位置、终点位置有随机性差异。

本课题中，选择测定点的组合方案，相当于解一个组合优化问题。备选的组合方案数量巨大，是典型的 NP-hard 问题（即算法复杂度不能用问题阶数 n 的多项式来表示）。例如，不妨假设最优方案是选取 9 个测试点，则总共的选择方案有C_{51}^{6}≈30 亿种。对于如此巨大的计算量，目前的计算机尚难以用暴力穷举的方法进行求解。

然而，对于 NP-hard 问题，可以尝试使用启发式搜索算法来求解。典型的启发式搜索有遗传算法和模拟退火等。

4 定标方案设计

利用遗传算法以及模拟退火寻找最佳的测试点组合。

4.1 遗传算法

遗传算法是基于达尔文的生物进化论和孟德尔的遗传学的一种通用的求解优化问题的适应性搜索方法，其基本原理是仿效生物界中物竞天择、适者生存的演化法则。遗传算法把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及编译等运算来交换种群中染色体的信息，最终生成符合优化目标的染色体。遗传算法流程图如图 2 所示。

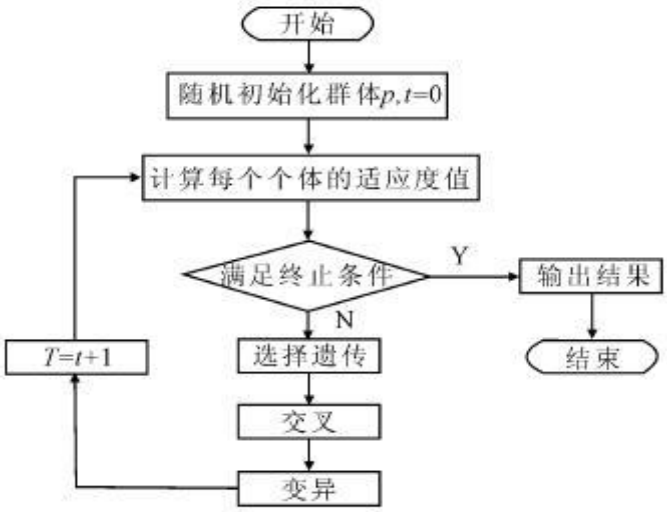

图 2 遗传算法流程图

(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直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。以评价函数得分作为选择依据，得分越高，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生物遗传基因的重组（和变异）。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。

当最优个体的评价值不再上升或者迭代次数达到预设的代数时，算法终止。

4.1.1 编码方式与初始种群

在本问题中，每个个体代表一种选点方案，由长度为 51 的数组表示，数组中有若干个 0 和 1，这个数组称为基因。若数组中第 k 个数为 1 则表示选取该点，为 0 则表示不选取，大多数遗传算法都采用这种二进制编码。这种编码方式在本问题中的优点是方便交叉和变异。

初始种群大小为 pop，每个个体的每个位置有 0.3 的概率为 1。

4.1.2 适应度、自然选择与交叉变异

自然选择的方式一般有轮盘赌选法和排序选择法。事实上，排序选择法可以视为一种固定了适应度函数的轮盘赌选法。所谓轮盘赌选法如图 3 所示。每个个体被选择的概率与适应度成正比例，所有个体的概率和为 1。

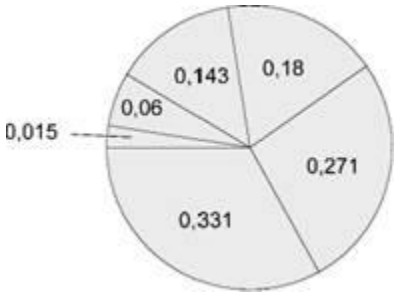

图 3 轮盘赌选法

对于个体 k（即测试点组合），其成本用 cost(k)表示，为对样本库中所有传感器定标成本的平均值。根据问题要求，cost(k)越小适应度越大，适用于这种情况的常用适应度函数有 1/(cost+c)和 max(cost)-cost(k)+c 等，其中 c 为常数，需保证每个个体的适应度不小于 0，max(cost)为所有个体中最大的定标成本，其他适应度函数大多由这两种拓展而来。为了防止早熟，适应度函数要求收敛不应太快，同时又可以有效筛选出优秀的个体。经过测试，最终适应度函数选择了$\sqrt{\max(\text{cost})-\text{cost}(k)}$。

交叉方式主要有单点交叉、两点交叉、多点交叉和均匀交叉。均匀交叉实际上是多点交叉更一般的形式。对于一个大搜索空间，均匀交叉被发现优于单点和两点交叉。另外，单点交叉可能优于两点交叉。在本问题中，搜索空间不是很大，同时出于程序运行速度方面的考虑，选择了单点交叉[1]。

变异方式是每个个体的每个位置都有一定概率变异，用 pm 表示。

4.1.3 重要参数的设置

遗传算法的主要参数有种群数量 pop、交叉概率 pc 和变异概率 pm。种群数量应尽量大，但考虑到计算机性能所限设定为 200。交叉概率，即配对的两个个体交叉互换染色体的概率，一般保持在 1.0 附近，使绝大多数父个体能够交叉，本题中设置为 0.9。变异概率 pm 一般在 0.1/L~1/L 之间，L 为编码序列长度[2]。经测试，pc 设置为 0.02 使得每个个体变异次数的数学期望略小于 1。收敛曲线如图 4 所示。

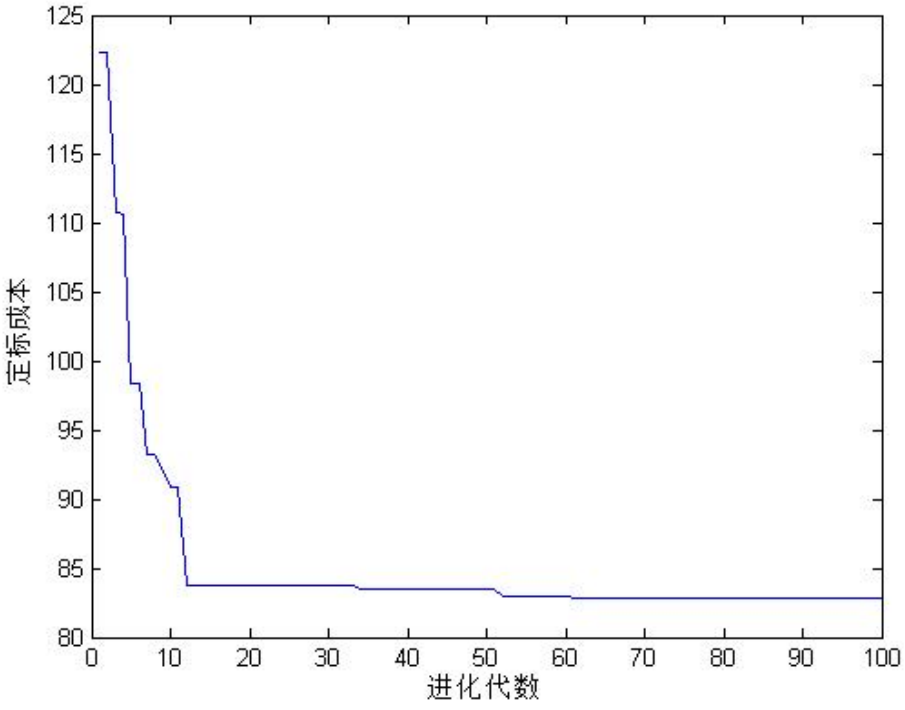

图 4 遗传算法收敛曲线

4.1.4 精英保留

为了不使每代种群中的最优个体，即精英，被淘汰，首先确定了一个精英个体直接复制到下一代。本题中使用了此方法。

4.2 模拟退火

“模拟退火”的原理和金属退火的原理近似：我们将热力学的理论套用到统计学上，将搜寻空间内每一点想像成空气内的分子；分子的能量，就是它本身的动能；而搜寻空间内的每一点，也像空气分子一样带有“能量”，以表示该点对命题的合适程度。算法先以搜寻空间内一个任意点作起始：每一步先选择一个“邻居”，然后再计算从现有位置到达“邻居”的概率。模拟退火流程图如图 5 所示。

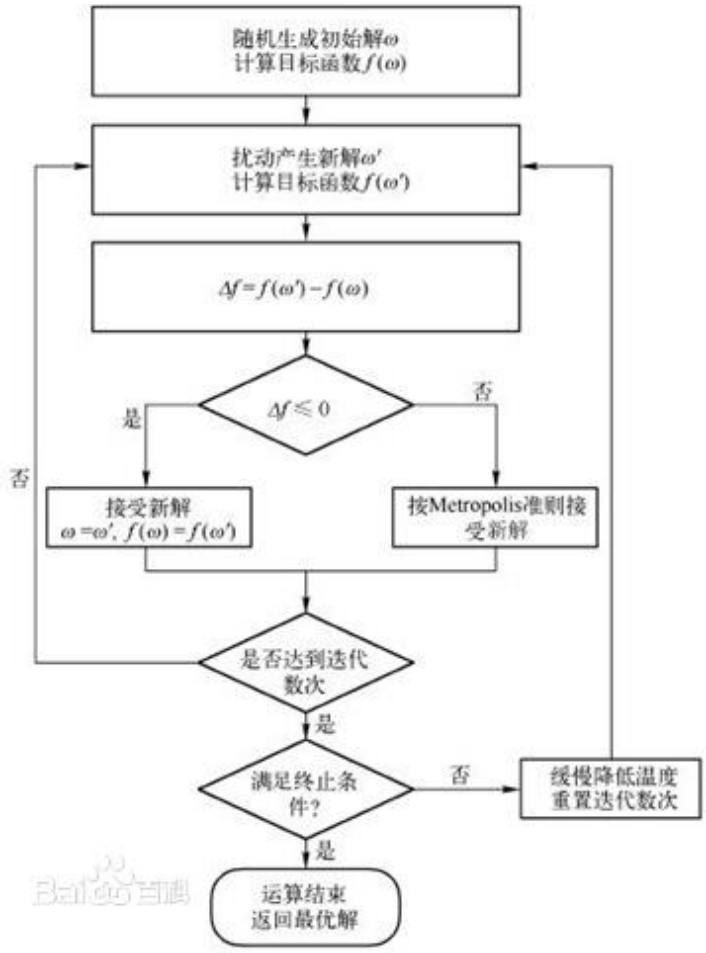

图 5 模拟退火流程图

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 Δf=C(S′)-C(S)，其中 C(S)为评价函数

(5) 若 Δf<0 则接受 S′作为新的当前解，否则以概率 exp(-Δt′/(KT))接受 S′作为新的当前解(k 为波尔兹曼常数）.

(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->0，然后转第 2 步。

终止条件通常取为连续若干个新解都没有被接受或温度降低到某预定值时终止算法[3]。

4.2.1 初始状态的设定

首先确定测试点的个数 num，然后随机产生 num 个测试点为初始状态。初始温度设为 temp。

4.2.2 状态能量的计算与状态更新

每个状态的能量即为所有样本定标成本的平均值。随着温度的降低，能量会越来越低。典型的模拟退火算法对于每个温度，内部循环 monte 次取较优解，称为蒙特卡洛循环。状态更新方法为对当前状态进行随机扰动，即随机改变某个测试点的位置，计算新状态的能量，然后选择是否接受新状态。

4.2.3 重要参数的设定

模拟退火的主要参数为玻尔兹曼常数 k、初始温度 temp、结束温度、内部循环次数 monte 和温度下降方式。玻尔兹曼常数影响收敛速度，显然从理论分析可以看出 k 越小收敛速度越慢，temp 越大循环次数越多越有可能得到最优解，温度下降速度与玻尔兹曼常数共同影响收敛速度。显然可以先确定初始温度和结束温度，通过改变 k 和温度下降速度调节收敛速度。在确定初始温度为 1000、结束温度为 1 的情况下，经测试最终确定 k 为 0.005，温度按照比例 0.98 下降。而 monte 结合程序运行效率和结果准确性考虑，定为 2*num。5 个测试点的收敛曲线如 6 所示。改变测试点数量后收敛曲线形态类似。

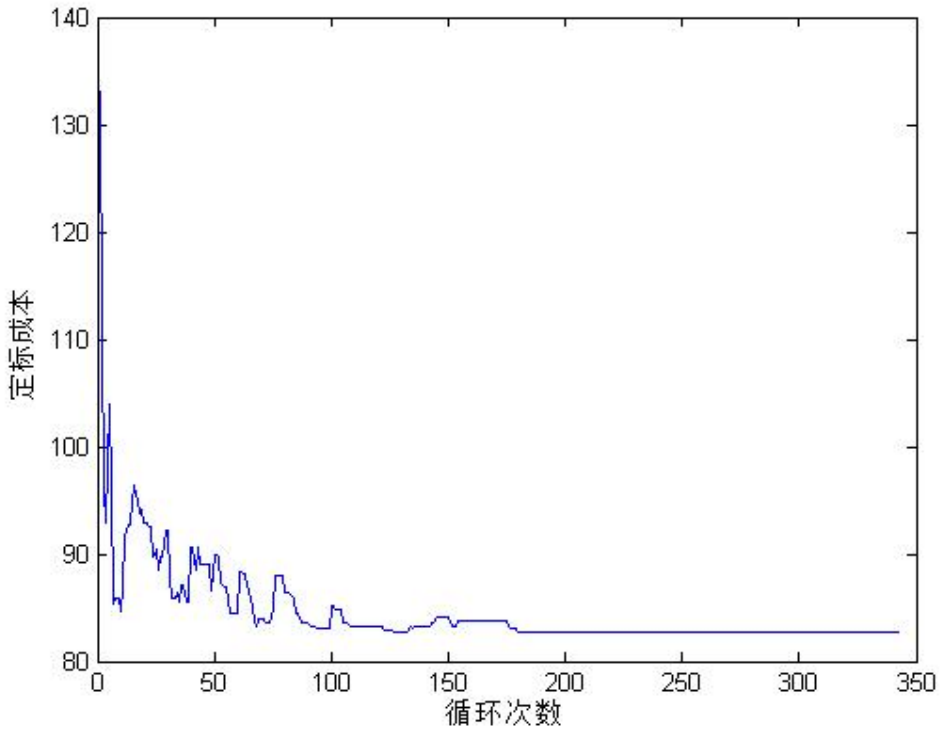

图 6 模拟退火收敛曲线

4.2.4 关于模拟退火的补充

由于确定了初始测试点的个数，因此需要对不同测试点数量分别计算并进行对比。

4.3 估算方法的选择

通过对特性曲线的观察，发现使用插值拟合是比较合适的方法。线性插值无法拟合出特性曲线的拐点，因此在三次样条插值 spline 和多项式插值 pchip 中择优选取。典型的三

次样条插值和多项式插值的特征如图 7 所示，可以看出因为拐点的存在，三次样条的误差较大，尤其是在拐点及拐点临近区域，而三次多项式插值则更好地拟合了曲线。综上，估算方法，即特性曲线拟合方法选定为多项式插值。

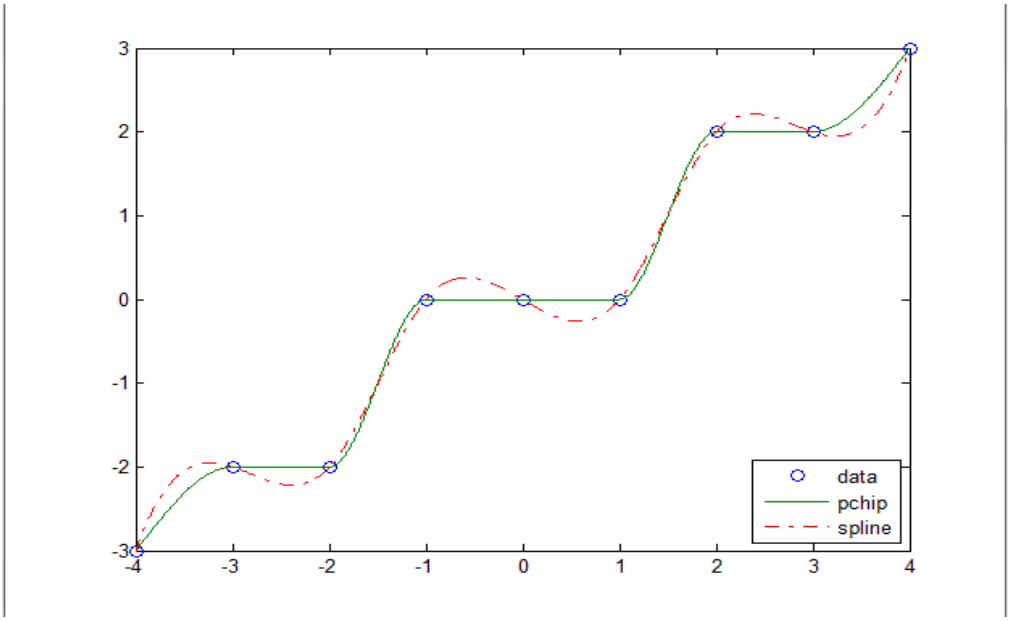

图 7 三次样条插值和多项式插值

5 程序运行结果与两种算法的对比

5.1 遗传算法

表 1 遗传算法程序运行结果

pc	0.005	0.01	0.015	0.02
1	90.4850746	83.5405117	86.9104478	83.4573561
2	84.2707889	85.3027719	85.3560768	82.7654584
3	85.4594883	84.5362473	87.0874200	85.2761194

5.2 模拟退火

表 2 测试点为 4 个时模拟退火程序运行结果

temp \ k	0.001	0.005	0.01
1000	96.4029851	96.4029851	96.4029851
500	96.4029851	96.4029851	96.4029851
100	96.4029851	96.4029851	96.4029851

表 3 测试点为 5 个时模拟退火程序运行结果

temp \ k	0.001	0.005	0.01
1000	82.7654584	82.7654584	83.3688699
500	82.7654584	82.7654584	82.7654584
100	82.7654584	82.7654584	82.7654584

表 4 测试点为 6 个时模拟退火程序运行结果

temp \ k	0.001	0.005	0.01
1000	85.0906183	85.0906183	85.0906183
500	85.0906183	85.0906183	85.1162047
100	85.0906183	85.0906183	85.1247335

当测试点多于 6 个时，测试成本已经达 84，显然不可能是最佳方案，因此最佳方案应

为 5 个。可以看出，temp=1000 时可以基本保证得到最优解，k=0.001 和 0.005 均可进一步保证得到最优解，实际上为了使收敛曲线特征更明显，即能量状态有不断波动的过程，k=0.005 更为合适。有兴趣者可以运行程序进行对比。

程序源代码及注释见于附录。

5.3 两种算法的对比

多次运行程序后发现，遗传算法每次得到的结果不完全相同。而模拟退火不仅收敛程度优于遗传算法，得到的解也一直是相对最优的。不仅如此，模拟退火的效率也比遗传算法高得多。随机选择其中几次运行结果列于表 5。

表 5 遗传算法与模拟退火结果对比

遗传算法		模拟退火（5 个测试点）	
测试点组合	定标成本	测试点组合	定标成本
4 16 26 35 48	82.7654584	4 16 26 35 48	82.7654584
4 17 27 36 48	83.4573561	4 16 26 35 48	82.7654584
4 16 26 35 48	82.7654584	4 16 26 35 48	82.7654584
4 16 25 33 43 51	86.4701493	4 16 26 35 48	82.7654584
4 17 27 36 48	83.4573561	4 16 26 35 48	82.7654584
3 14 23 30 38 49	85.2761194	4 16 26 35 48	82.7654584
4 17 27 36 49	83.3688699	4 16 26 35 48	82.7654584
4 16 26 33 43 51	86.5458422	4 16 26 35 48	82.7654584

6 结论

(1)根据程序运行结果，最佳定标方案应为：测试点组合为[4 16 26 35 48]，拟合方法为多项式插值 pchip，对于样本库的平均定标成本为 82.7654584。

(2)经过对比可以看出，遗传算法不一定收敛于最优解，且得出的解不如模拟退火的好。原因在于遗传算法依赖于初始种群而模拟退火不依赖。计算机性能限制了遗传算法初始种群的大小，导致初始种群不一定具有最优秀的基因片段，所以最终不一定收敛于全局最优解。并且从理论证明上讲，模拟退火已经被证明了以一定概率收敛于最优解，而遗传算法的正确性尚未得到完全的证明。

7 参考文献

[1] D.K.Pratihar. Soft Computing[M]. 北京:科学出版社.

[2] D.K.Pratihar. Soft Computing[M]. 北京:科学出版社.

[3] 百度百科. 模拟退火词条.

附录

程序源代码及注释

遗传算法

main.m

```
data=csvread('20141010dataform.csv');
pop=200;  %  种群数量
```

```
pc=0.9;  %  交叉概率
pm=0.02;  %  变异概率
n=100;  %  进化代数
y=zeros(469,51);
y(1:469,:)=data(2:2:938,:);
waveform=zeros(1,n);

gene=geneinit(pop);
for g=1:n
    cost=adapt(gene,y,pop);
    gene=select(gene,cost,pop);
    gene=generate(gene,pop,pc);
    gene=mutate(gene,pop,pm);
    waveform(g)=min(cost);
    fprintf('%3d      [',g);  %  输出当前最优个体
    fprintf('%2d ',find(gene(1,:)==1));
    fprintf(']      %10.7f\n',min(cost));
end

plot(1:n,waveform);  %  画出收敛曲线
xx=find(gene(1,:)==1);
assess(xx,y);  %  将得到的解写入文件
```

geneinit.m

```
function out = geneinit(pop)
%  随机产生升初始种群

out=round(rand(pop,51)-0.2);

end
```

adapt.m

```
function out = adapt(gene,y,pop)
%  计算每个个体平均成本

out=zeros(pop,1);
x=5:0.1:10;
for i=1:pop
    c=sum(gene(i,:)==1);  %  测试点数量
    pos=find(gene(i,:)==1);  %  测试点位置
    xx=5+(pos-1)*0.1;  %  测试点x值
    yy=y(:,pos);  %  测试点y值矩阵
    f=pchip(xx,yy);
    dy=ppval(f,x)-y;  %  误差
    out(i)=12*c+errorcost(dy)/469;  %  单个个体平均成本
end
```

```
end
```

select.m

```
function out = select(gene,cost,pop)
%  自然选择，out(1)为父代最优个体予以保留

out=zeros(pop,51);
cost0=(max(cost)-cost).^0.5;  %  适应度
s0=sum(cost0);
s=zeros(pop+1);
s(1)=0;
s(pop+1)=1;
s(2:pop)=sum(cost0(1:pop-1))/s0;
for i=2:pop
    t=rand();
    j=search(t,s,1,pop+1);
    out(i,:)=gene(j,:);
end

cost_sort=[(1:pop)',cost];
cost_sort=sortrows(cost_sort,2);
out(1,:)=gene(cost_sort(1,1),:);

end
```

generate.m

```
function out = generate(gene,pop,pc)
%  交叉，保留gene(1)
%  i与pop-i+2配对

for i=2:floor(pop/2+1)
    out=gene;
    mid=floor(rand()*50)+1;
    t=rand();
    if t<=pc
        out(i,1:mid)=gene(pop-i+2,1:mid);
        out(pop-i+2,1:mid)=gene(i,1:mid);
        out(i,mid+1:51)=gene(pop-i+2,mid+1:51);
        out(pop-i+2,mid+1:51)=gene(i,mid+1:51);
    end
end

end
```

mutate.m

```
function out = mutate(gene,pop,pm)
```

```
%  变异

out=gene;
for i=2:pop;
    for j=1:50;
        t=rand();
        if t<=pm
            out(i,j)=~out(i,j);
        end
    end
end

end
```

assess.m

```
function [out] =assess(in,y)
%   计算最优解成本并写入文件

out=length(in)*12;
x=5:0.1:10;
xx=5+(in-1)*0.1;
yy=y(:,in);  %  测试点y值矩阵
f=pchip(xx,yy);
dy=ppval(f,x)-y;  %  误差
out=out+errorcost(dy)/469;
fid=fopen('answer.txt','a');
fprintf(fid,'position: [ ');
fprintf(fid,'%2d ',in);
fprintf(fid,']    mean_cost: %10.7f\n\n',out);
fclose(fid);

end
```

search.m

```
function [out] = search(in,s,l,r)
%  二分法查找，用于轮盘赌选

mid=floor((l+r)/2);
if in<=s(mid)
    if in>s(mid-1)
        out=mid-1;
    else
        out=search(in,s,l,mid);
    end
else
    if in<=s(mid+1)
```

```
        out=mid;
    else
        out=search(in,s,mid,r);
    end
end

end
```

errorcost.m

```
function [out] = errorcost(dy)
%  计算单个个体平均误差成本

t=abs(dy);

t0=sum(sum(t<=0.5));
t1=sum(sum(t<=1))-t0;
t2=sum(sum(t<=2))-t0-t1;
t3=sum(sum(t<=3))-t0-t1-t2;
t4=sum(sum(t<=5))-t0-t1-t2-t3;
t5=sum(sum(t>5));

out=0.5*t1+1.5*t2+6*t3+12*t4+25*t5;

end
```

模拟退火

main.m

```
data=csvread('20141010dataform.csv');
y=zeros(469,51);
y(1:469,:)=data(2:2:938,:);
num=5;  %  测试点数量
temp=1000;  %  初始温度
waveform=zeros(1,floor(log(1/temp)/log(0.98))+1);
monte=2*num;  %  内部蒙特卡洛循环次数
k=0.005;  %  玻尔兹曼常数
g=1;  %  记录循环次数
flag=1;  %  是否接受新状态，初始时接受了新状态

state=stateinit(num);
while temp>1
    for i=1:monte
        if flag  %  接受新状态时才重新计算成本，减少重复计算
            cost=calcost(state,y);
        end
        state0=stateupdate(state);
        cost0=calcost(state0,y);
```

```
        flag=accept(cost,cost0,temp,k);  %  是否接受新状态
        if flag
            state=state0;
        end
    end
    fprintf('%3d      [',g);
    fprintf('%2d ',state);
    if flag
        waveform(g)=cost0;
        fprintf(']      %10.7f\n',cost0);
    else
        waveform(g)=cost;
        fprintf(']      %10.7f\n',cost);
    end
    temp=temp*0.98;
    g=g+1;
end

plot(1:g-1,waveform);  %  画出收敛曲线
assess(state,y);  %  计算得到的解的成本并写入文件
```

stateinit.m

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

out=zeros(1,num);
for i=1:num
    t=round(rand()*51)+1;
    while any(out==t)
        t=round(rand()*51)+1;
    end
    out(i)=t;
end
out=sort(out);

end
```

calcost.m

```
function out = calcost(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);
```

```
dy=ppval(f,x)-y;  %  误差
out=12*c+errorcost(dy)/469;  %  成本

end
```

stateupdate.m

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

out=state;
pos1=floor(rand()*51)+1;  %  随机选择一个已选定的测试点
while ~any(out==pos1)
    pos1=floor(rand()*51)+1;
end
pos2=floor(rand()*51)+1;  %  随机选择一个未选定的点
while any(out==pos2)
    pos2=floor(rand()*51)+1;
end
pos=find(out==pos1);
out(pos(1))=pos2;  %  更新
out=sort(out);

end
```

accept.m

```
function out = 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();
    delta=cost-cost0;
    if t<exp(delta/(k*temp))  %  成本升高时按规则有一定几率接受
        out=1;
    end
end

end
```

assess.m

```
function [out] =assess(in,y)
%   计算成本并写入文件

out=12*length(in);
x=5:0.1:10;
```

```
xx=5+(in-1)*0.1;  %  测试点x值
yy=y(:,in);  %  测试点y值矩阵
f=pchip(xx,yy);
dy=ppval(f,x)-y;  %  误差
out=out+errorcost(dy)/469;
fid=fopen('answer.txt','a');
fprintf(fid,'position: [ ');
fprintf(fid,'%2d ',in);
fprintf(fid,']    mean_cost: %10.7f\n\n',out);
fclose(fid);

end
```

errorcost.m

```
function [out] = errorcost(dy)
%  平均误差成本函数

t=abs(dy);

t0=sum(sum(t<=0.5));
t1=sum(sum(t<=1))-t0;
t2=sum(sum(t<=2))-t0-t1;
t3=sum(sum(t<=3))-t0-t1-t2;
t4=sum(sum(t<=5))-t0-t1-t2-t3;
t5=sum(sum(t>5));

out=0.5*t1+1.5*t2+6*t3+12*t4+25*t5;

end
```