统计推断在数模转换系统中的应用

组号 47　　牛文浩 5130309090，邵东生 5130309081

摘要：本文为上海交通大学电子信息与电气工程学院课程设计《统计推断在数模转换系统中的应用》的课题论文。本小组运用 Matlab 数学软件并结合统计推断的方法，通过 469 样本数据进行研究，使用模拟退火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，三次样条插值拟合，线性插值拟合

1 引言

在工程时间和科学实验中，当我们需要寻找两个变量间的关系是，在理论上我们可以通过计算得出两个变量之间的确定的函数关系，但实际却很难找到非常精确描述两个变量的函数。通常采取的方法是通过多种不同曲线拟合，算出多种不同的方案，并选择某一统计量作为评定优劣标准，从而选出最优的方案。

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

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。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，获得信号作为 Y 的读数（监测模块对 Y 的估测值）。

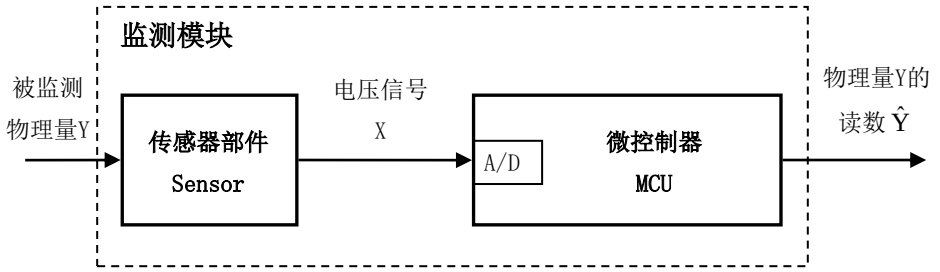

图 1.2 监测模块组成框图

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。

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，特制定以下成本计算规则。

1.3.1 单点定标误差成本

$$s_{i,j} = \begin{cases} 0 & \text{if } |\hat{y}_{i,j} - y_{i,j}| \le 1 \\ 1 & \text{if } 1 < |\hat{y}_{i,j} - y_{i,j}| \le 2 \\ 2 & \text{if } 2 < |\hat{y}_{i,j} - y_{i,j}| \le 3 \\ 4 & \text{if } 3 < |\hat{y}_{i,j} - y_{i,j}| \le 5 \\ 10 & \text{if } |\hat{y}_{i,j} - y_{i,j}| > 5 \end{cases} \quad (1.3.1)$$

单点定标误差的成本按式（1.3.1）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。

1.3.2 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=20。

1.3.3 某一样本个体的定标成本

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

对样本 i 总的定标成本按式（1.3.3）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。

1.3.4 校准方案总体成本

按式（1.3.3）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。总体成本较低的校准方案，认定为较优方案。

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

2 方案设计

2.1 基本思路

实验的目标是对给定的成本函数（目标函数），找到其最优解。实验中大致有两个优化方向，即对取点方案最优化解的搜索和拟合方式的选择。本实验采用的模拟退火法的最优化解的搜索方法，拟合方式选取过三次样条插值和线性插值的拟合方法。

2.2 模拟退火算法

2.2.1 模拟退火算法原理

模拟退火算法是通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法。

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。

2.2.2 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。

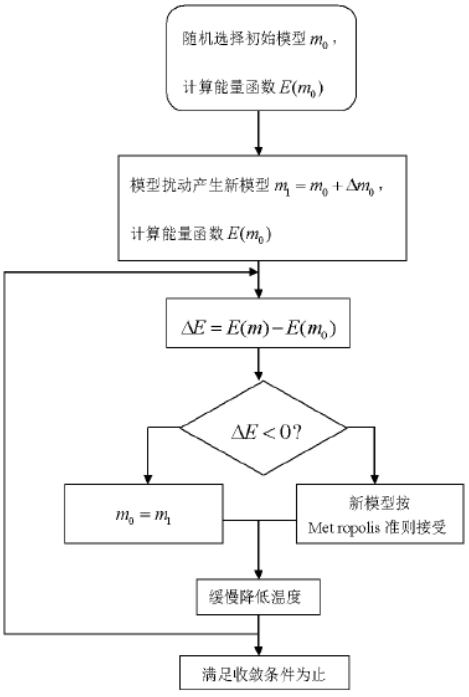

图 2.2.2 模拟退火算法流程图

2.2.3 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 $\Delta t'<0$ 则接受 S′作为新的当前解 S，否则以概率 $\exp(-\Delta t'/T)$ 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 1 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。

2.2.4 应用算法的具体问题及解决方案

（1）模拟退火法的自变量一般为实数，而这里自变量要求是整数，解决这个问题主要要解决以下两点关键问题(假设取 N 个点)。一是如何用一定范围（[1,51]）的不重复的整数初始化粒子的速度与位置向量？二是如何保证粒子的速度，位置向量经过更新函数更新后还是整数？针对以上的两个问题解决方法如下：利用 randomperm(x)函数产生 1-51 的随机序列(1-51 所有的数都出现且只出现一次)然后取从中选取 N 点，就相当于随机产生的 1-51 的 N 个随机数；仍然保留原有的速度更新公式，只不过利用 round(x)对最后的结果进行取整。

（2）模拟退火算法对自变量的范围没有限制，而本问题中自变量的范围是有限制的。本问题中自变量的范围是[1,51](把 x 数据的下标看作自变量)，在更新粒子的位置与速度向量时，可能会是使粒子的位置向量的元素大于 51(或者小于 1)，对于这种问题本实验采用如下方式对粒子更新后的位置向量做出限制本小组本想过直接令 x(i,j)=mod(x(i,j),51)，但是这样的搜索算法的搜索效率显然不高，因为退火算法要求新解在旧解的一个小邻域内会使得冷却效果比较好，但是直接取模会使得位置向量元素超出范围时经过处理后数值相对于旧值变化太大。故采取如上方法。但是这个方法存在一个问题，后文会进行讨论。

```
for j=1:length(x(i,:))
        if x(i,j)>51
            x(i,j)=51-mod(x(i,j),51);
        end
        if x(i,j)<1
            x(i,j)=1+mod(x(i,j),51);
        end
    end
```

2.2.5 算法参数

2.2.5.1 算法参数说明

（1）函数原型[xm,fv]=SimuPSO(fitness,N,c1,c2,lamda,M,D)；

（2）N：粒子数目，粒子数目的多少根据问题的复杂程度自行决定。一般的优化问题取 20-40 个粒子完全可以得到很好的结果了，对于比较简单的问题 10 个粒子已经足够可以取得好的结果。对于特别的复杂问题粒子数可以取到 100 以上；

（3）c1：学习因子 1；c1：学习因子 2；学习因子使粒子具有自我总结和向群体中优秀个体学习的能力，从而向群体内或邻域内最优点靠近，通常取 c1,c2 为 2，但也有其他的取值，一般 c1 等于 c2,且范围在 0-4 之间。

（4）lamda：退火常数；

（5）M：最大迭代次数；

（6）D：自变量个数；

2.2.5.2 算法参数对问题的影响

（1）N：该问题在忽略每次测试的偶然误差后，经过测试 N 取 20 或者 40 得出的成本差距不是很大。

（2）lamda：经测试取 1.00 效果最好。

（3）M：理论上迭代次数越大越接近最优解，但是考虑到时间成本笔者认为应该取 50-100 之间。

（4）D：这个参数与取点方案中的选点个数一致，本实验取 7。下文会有更进一步的讨论

2.3 拟合方式的选取

2.3.1 选取三次样条插值拟合的结果

三次样条插值的结果大多为 95 附近，例如：

[xm,vm]=SelPOS(@fitness,40,2.05,2.05,1,50,7)

xm=35 3 28 51 18 44 11

vm=96.7431

2.3.2 选取线性插值拟合的结果

线性插值的结果成本多在 90 以下，例如：

[xm,vm]=SelPOS(@fitness,40,2.05,2.05,1,50,7)

xm=47 46 40 48 44 38 39

vm=84.0384

2.3.3 结果分析

很明显线性插值拟合的结果明显优于三次样条插值拟合,这样的结果很明显是合乎事实的。因为数据点的分布大致分为 3 段每段近似为直线，只不过斜率不尽相同。若是用三次样条插值，在近似直线段上的误差明显要比三次样条插值要大不少。而线性插值只需要将取点在中间变化段集中就可以得到比较优秀的解。线性插值拟合得出的结果也证实了这一点。

3. 结果展示

最终选择线性插值拟合的方法，取点方案如下：

[xm,vm]=SelPOS(@fitness,40,2.05,2.05,1,50,7)

xm=47 46 40 48 44 38 39

vm=84.0384

4. 问题思考

4.1 如何解决模拟退火法不能确定取点个数的问题

模拟退火法的一大缺点是它必须在给定的选取点的个数，才能继续搜索最优解。但是如何得到最优的粒子个数呢？本实验采取两种办法：

（1）从一个较小的值开始枚举，直到遇见第一个峰值。

优点：设计比较简单。

缺点：虽然理论上最优解点的个数应该比较小，但是不能保证第一个峰值之后没有更优的解

（2）每次都把取点方案中的取点个数设置为 $D=D_0=51$ 个(即所有点)，由于本实验的算法没有对重复的取点做处理(即可能会出现取点方案 xm 中有多个元素相同)，若出现重复的点则说明改点对最优解的贡献比较大，记下所有有重复出现的点的个数 D_1，而后令 $D=D_1$。重复上述过程直到取点方案中没有出现重复的点。

优点：实现了对取点个数的启发式搜索。

缺点：由于第一次搜索需要取到所有点，会严重增加运行时间。由于笔者电脑配置和算法上的问题(下文会详细介绍)使得运行时间比较长，加上这个算法后运行时间严重超出可以接受的范围。故最后并没有采用。

4.2 有关运算时间的思考

(1)模拟退火算法本来是 O(N)的时间复杂度，但是如果要检查向量中元素是否超出范围就必须要在模拟退火算法中主循环中再加上一个循环，这样就使时间复杂度提升到了 $O(N^2)$。适当减少粒子数目 N 可以减少运行时间。但是不能把 N 减的太小，太小的 N 会使成本上升。

(2)与电脑配置有关，intel i7 的处理器明显比 intel i5 要快。

5. 参考文献

[1]龚纯，王正林. 《精通 MATLAB 最优化算法(第二版)》 电子工业出版社

[2]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 模拟退火算法函数(SelPSO.m)

```
%allData=xlsread('20141010dataform.csv');
function [xm,fv]=SelPSO(fitness,N,c1,c2,lamda,M,D)
    x=zeros(N,D);
    v=zeros(N,D);
    p=zeros(1,N);
    y=zeros(N,D);
    Tfit=zeros(1,N);
    ComFit=zeros(1,N);
    for i=1:N
        %for j=1:D
            %x(i,j)=randn;
```

```
            %v(i,j)=randn;
        A=randperm(51);
        B=randperm(51);
        B=B-1;
        for j=1:D
            x(i,j)=A(j);
            v(i,j)=B(j);
        end
end
for i=1:N;
        %disp(x(i,:));
        p(i)=fitness(x(i,:));
        y(i,:)=x(i,:);
end
pg=x(N,:);
for i=1:(N-1)
        if fitness(x(i,:))<fitness(pg)
            pg=x(i,:);
        end
end

T=fitness(pg)/log(5);
for t=1:M
        groupFit=fitness(pg);
        for i=1:N
            Tfit(i)=exp(-(p(i)-groupFit)/T);
        end
        SumTfit=sum(Tfit);
        Tfit=Tfit/SumTfit;
        pBet=rand();
        for i=1:N
            ComFit(i)=sum(Tfit(1:i));
            if pBet<=ComFit(i)
                pg_plus=x(i,:);
                break;
            end
        end
        C=c1+c2;
        ksi=2/abs(2-C-sqrt(C^2-4*C));
        for i=1:N
            v(i,:)=ksi*(v(i,:)+c1*rand*(y(i,:)-x(i,:))+c2*rand*(pg_plus-x(i,:)));
            %for j=1:D
            v(i,:)=round(v(i,:)); %½«v±ä³ÉÕûÊý
            %end
```

```
            x(i,:)=x(i,:)+v(i,:);
            for j=1:length(x(i,:))% ³¬·¶Î§Ô õÃ ´°ì
                if x(i,j)>51
                    x(i,j)=51-mod(x(i,j),51);
                end
                if x(i,j)<1
                    x(i,j)=1+mod(x(i,j),51);
                end
            end
            if fitness(x(i,:))<p(i)
                p(i)=fitness(x(i,:));
                y(i,:)=x(i,:);
            end
            if p(i)<fitness(pg)
                pg=y(i,:);
            end
        end

        T=T*lamda;
    end
    xm=pg';
    disp(xm);
    fv=fitness(pg);
    disp(fv);
end

%[xm,vm]=PSO(@fitness,40,2,2,0.5,1000,30)
%disp(xm);
%disp(vm);
```

6.1.2成本计算函数（fitness.m）

```
function C=fitness(x)
my_answer=x;
%disp(my_answer);
my_answer_n=size(my_answer,2);

minput=xlsread('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;

index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for j=1:nsample
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

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*g5_0;
si=sum(sij,2)+Q*ones(nsample,1)*my_answer_n;
cost=sum(si)/nsample;
C=cost;
end
```

6.1.3拟合函数(mycurvefitting.m)

```
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];
y1=interp1(x_premea,y0_premea,x);
end
```

6.1.4执行语句

```
>>[xm,fv]=SelPSO(fitness,N,c1,c2,lamda,M,D)
```