统计推断在数模转换系统中的应用

组号：41　小组成员：周元超 5140309045 徐梓嘉 5140309038

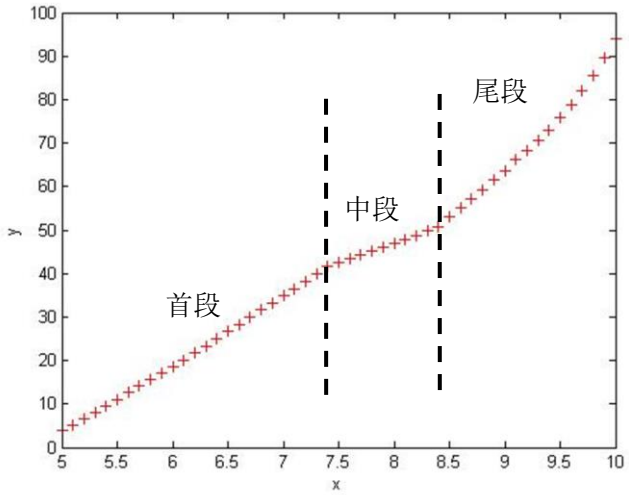
1 引言

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。

2 数学模型的建立

2.1 已有数据

已得的数据共有 400 组，每组 51 个测试点，大致的特性曲线如下：



- 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。

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”。

该表格以 CSV 格式制作为电子文件。表格中奇数行存放的取值，偶数行存放对应的取值。第 2i – 1 行存放第 i 个样本的 X 数值，第 2i 行相应列存放对应的实测 Y 数值。

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。

- 单点定标误差成本

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

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。

- 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。

- 某一样本个体的定标成本

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

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。

- 校准方案总成本

按式（3）计算评估校准方案的总成本，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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \quad (3)$$

总成本较低的校准方案，认定为较优方案。

4 插值

4.1 插值法

利用拟合得到的函数曲线在某区间内的值，作为原本数值未知的点的近似取值。利用多项式插值，对给定的数据集进行插值的 n 阶多项式就被给定的数据点所唯一的定义出来。但是，对于同样的数据进行插值的 n 阶样条并不是唯一的，为了构建一个唯一的样条差值式它还必须满足另外 n-1 个自由度

4.2 三次样条插值

早期工程师制图时，把富有弹性的细长木条（所谓样条）用压铁固定在样点上，在其他地方让它自由弯曲，然后沿木条画下曲线，称为样条曲线。

三次样条插值（简称 Spline 插值）是通过一系列形值点的一条光滑曲线，数学上通过求解三弯矩方程组得出曲线函数组的过程。

三次样条函数:

定义:函数 $S(x)\in C[a,b]$ ，且在每个小区间$[x_j, x_{j+1}]$上是三次多项式，其中

$a=x_0<...<x_n=b$ 是给定节点，则称 S(x)是节点 x0,x1,...xn 上的三次样条函数。

若在节点 x_j 上给定函数值 $Y_j=f(X_j).(j=0,1,,n)$ ，并成立

$S(x_j)=y_j .(j=0,1,,n)$ ，则称 S(x)为三次样条插值函数。

实际计算时还需要引入边界条件才能完成计算。边界通常有自然边界（边界点的二阶导为 0），夹持边界（边界点导数给定），非扭结边界（使两端点的三阶导与这两端点的邻近点的三阶导相等）。一般的计算方法书上都没有说明非扭结边界的定义，但数值计算软件如 Matlab 都把非扭结边界条件作为默认的边界条件。[1]

5 选点

虽然用所有数据点（51 个）进行拟合最后误差更小，但是大量的测量带来的成本远超过误差，所以我们需要从其中选出尽量少的一些点，并让最后拟合的误差成本处于一个可以接受的范围内

我们每一组数据统一选取七个数据点进行拟合，根据经验，七个点中有首尾两个点，剩下的五个点则在中间选取。我们用三组数据分别进行选点测试，如果结果相差不大即默认可以代替整个样本进行取点。

5.1 穷举法

即从（2,3,4,5,6）开始计算最后的成本，一直到（45,46,47,48,49,50），取其中成本最低的一组取点方法，但我们可以看出 $C_{49}^{5}=1906884$，内存和时间复杂度都决定它难以算出结果。

5.2 遗传算法[2]

5.2.1 遗传算法简介

遗传算法是计算数学中用于解决最佳化的搜索算法，是进化算法的一种。进化算法最初是借鉴了进化生物学中的一些现象而发展起来的，这些现象包括遗传、突变、自然选择以及杂交等。遗传算法通常实现方式为一种计算机模拟。对于一个最优化问题，一定数量的候选解（称为个体）的抽象表示（称为染色体）的种群向更好的解进化。传统上，解用二进制表示（即 0 和 1 的串），但也可以用其他表示方法。进化从完全随机个体的种群开始，之后一代一代发生。在每一代中，整个种群的适应度被评价，从当前种群中随机地选择多个个体（基于它们的适应度），通过自然选择和突变产生新的生命种群，该种群在算法的下一次迭代中成为当前种群。

5.2.2 遗传算法的基本运算过程

a)初始化:设置进化代数计数器 t=0，设置最大进化代数 T=100，随机生成 M=400 个个体作为初始群体 P(0)。

b)个体评价:计算群体 P(t)中各个个体的适应度。其中适应度函数取成本的倒数。

c)选择运算:将选择算子作用于群体。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。选择操作是建立在群体中个体的适应度评估基础上的。我们的选择使用的是轮盘赌算法，即根据每个个体的适应度占总适应度的比例来确

定其存活概率。

d)交叉运算:将交叉算子作用于群体，遗传算法中起核心作用的就是交叉算子。本组取交叉概率为 0.5，交叉位点用随机数确定。

e)变异运算:将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。群体 P(t)经过选择、交叉、变异运算之后得到下一代群体 P(t+1)。

f)终止条件判断:若 t=100,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

5.2.3 遗传算法的缺陷

1. 编码不规范及编码存在表示的不准确性。

2. 单一的遗传算法编码不能全面地将优化问题的约束表示出来。考虑约束的一个方法就是对不可行解采用阈值，这样，计算的时间必然增加。

3.遗传算法通常的效率比其他传统的优化方法低。

4.遗传算法容易过早收敛。

5.遗传算法对算法的精度、可行度、计算复杂性等方面，还没有有效的定量分析方法。

5.2.4 结果

Generation1;

141.1245

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

Generation5;

103.6165

1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

Generation100;

97.9595

1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

最终结果求得的成本为 97.9595，取点为[1,10,19,28,33,43,51]

5.3 模拟退火算法

5.3.1 模拟退火算法的步骤

第一步是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。

第二步是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。

第三步是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。

第四步是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。

5.3.2 计算中的问题

（1）每次降温的程度：一开始我们选择的是 0.95，但是结果不理想，成本较高，随后我们将其改为 0.999，这次运算成本显著变小，但是运行时间用了十多分钟，最后经过多次试验我们选择了

（2）由于每次循环只改变一个点，退火算法对初始值的依赖性非常强，而且比较容易陷入局部最优解。

5.3.3 计算结果

95.2148　所选取的点为【2　9　20　27　34　44　50】

6. 总结

从结果来看，遗传算法与退火算法相差不大，退火算法的结果略微占优；从时间上看，退火算法比遗传算法明显快捷，这也可能与我们使用的遗传算法中使用了较多的循环语句有关。从代码来看，退火算法相比之下更为简洁，但是和遗传算法的变异的随机性和任意性相比，模拟退火算法在每次循环只能改变一个基因点，对初始数据的依赖性更大，很容易就陷入局部最优解。综合来看，两种算法都有其优劣，但作为模拟算法，并不能真正求出问题的最优解，只能通过增加循环次数来使答案更靠近最优解。

参考文献：

【1】百度百科——三次样条插值。

【2】搜狗百科——遗传算法词条

该报告还引用了统计推断课程提供的相关资料

参考他人报告或代码的申明

统计推断课程，2015 年秋季学期第 41 组，成员周元超 学号 5140309045，徐梓嘉 学号 5140309038，在报告编写过程中，以下方面参考了往届报告，现列表说明：

主要参考项目	说明
代码方面	《统计推断在数模模数转换中的应用》，刘昊，2013 年秋季学期，组号 03 在该组报告附录提供的程序代码基础上，进行了少量修改。
算法描述方面，包含流程图	《统计推断在数模转换系统中的应用》，吴炜，2009 年秋季学期，组号 208 参考了该组报告的算法描述文字，引用了其流程图。
拓展问题研究	
结果统计和分析	

除了以上注明的参考内容，和报告中列出的引用文献注解，本报告其他部分都不包含任何其他个人或集体已经发表或撰写过的作品成果

附录：

程序代码：

①穷举法：

```
function [best] = EA(  )
s=0;
best=100000;
f=fittype('poly3');
data=xlsread('20150915dataform.csv');
data=data';
for i=1:400;
    a=data(:,2*i-1);
    b=data(:,2*i);
    for num1=2:50
        for num2=num1+1:50
            for num3=num2+1:50
                for num4=num3+1:50
                    for num5=num4+1:50
                        x=(a(1) a(num1) a(num2) a(num3) a(num4) a(num5)
a(51))'
                        y=(b(1) b(num1) b(num2) b(num3) b(num4) b(num5)
a(51))'
                        myline=fit(x,y,f);
                        for cost=1:51
                            if(abs(b(i)-myline(4.9+0.1*i))<=0.4)
                                s=s;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=0.6)
                                s=s+0.1;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=0.8)
                                s=s+0.7;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=1)
                                s=s+0.9;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=2)
                                s=s+1.5;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=3)
                                s=s+6;
                            elseif (abs(b(i)-myline(4.9+0.1*i))<=5)
                                s=s+12;
                            else s=s+25;
                            end
```

```
                                s=s+84;
                                s=s/400;
                                if(s<best)best=s;
                                end
                            end
                        end
                    end
                end
            end
        end
```

②遗传算法：

```
function s = average_cost(matrix,array) %求平均每组的成本
    head=1;tail=51;
    sum=0; %记录总成本
    for i=1:400
        x=matrix(2*i-1,head:tail);
        y=matrix(2*i,head:tail);
        sum=sum+cost(x,y,array);
    end
    s=sum/400;
end
function f= cost(x,y,array)  %计算每组成本
    j=0;groupcost=0;num=51;
    xline=zeros(1,sum(array)); %group 中第 i 行不为 0 的元素个数
    yline=zeros(1,sum(array));
    for i=1:num
        if(array(i)==1)
        j=j+1;
        xline(j)=x(i);
        yline(j)=y(i);
        end
    end
    x1=5.0:0.1:10.0;  %从 5.1 到 10.0 步长为 0.1 进行插值和拟合
    y1=interp1(xline,yline,x1,'spline');
    dx=zeros(1,num);
    for i=1:num
        d(i)=abs((y(i)-y1(i))); %每个点的误差
    end
    for i=1:num %计算误差成本
        if(dx(i)<=0.4)
            groupcost=groupcost+0;
        elseif(dx(i)<=0.6)
```

```
            groupcost=groupcost+0.1;
        elseif(dx(i)<=0.8)
            groupcost=groupcost+0.7;
        elseif(dx(i)<=1)
            groupcost=groupcost+0.9;
        elseif(dx(i)<=2)
            groupcost=groupcost+1.5;
        elseif(dx(i)<=3)
            groupcost=groupcost+6;
        elseif(dx(i)<=5)
            groupcost=groupcost+12;
        else
            groupcost=groupcost+25;
        end
    end
    f=groupcost+12*sum(array); %加上测量成本

function f=geneticalgorithm() %遗传算法主程序
    groupsize=51; %群体大小
    N=400;%种群数
    crossPer=0.5; %交叉概率
    generation=100;%循环代数
    varPer=0.01; %变异概率
    A=xlsread('20150915dataform.csv');
    group=zeros(N,groupsize);%种群矩阵(行表示种群，列表示群体）
    fitness=zeros(1,N);%种群适应度矩阵
    good_result=0;
    best=zeros(1,groupsize);
    for i=1:N
        group(i,1)=1;group(i,groupsize)=1;
        g=round(rand(1,5)*48)+2; %避免取到首位两个点
        for j=1:4
            cost(i)=average_cost(A,group(i,:));
        end
    end
    cost=zeros(1,N);
    for i=1:N
        cost(i)=average_cost(A,group(i,:));
    end
    [value,flag]=min(cost);
    good_result=value;
    best=group(flag,:);
    for g=1:generation%循环代数
        for k=1:N%计算适应度
```

```
            fitness(k)=1/average_cost(A,group(k,:));
        end
        sumfitness=sum(fitness(:));
        fitnessPer=zeros(1,N);
        for t=1:N
            fitnessPer(t)=fitness(t)/sumfitness;
        end
        latergroup=zeros(N,groupsize);
        fitnessPer=cumsum(fitnessPer);
    for i=1:N
        a=rand(1,1);
        if(a<=fitnessPer(1))
            latergroup(i,:)=group(1,:);
        end
        for j=1:N-1
            if (fitnessPer(j)<a&&a<=fitnessPer(j+1))
                latergroup(i,:)=group(j,:);
            end
        end
    end
    latergroup(1,:)=best;
%按照 crossPer 决定参与交叉的染色体数，从经过选择后的数组中选择染色体配对交叉
    for i=1:crossPer*N
        location=randi(groupsize-1);%交换的位置随机选择
        indiv1=randi(N);%随机选取两个个体进行交叉
        indiv2=randi(N);
        tmp=latergroup(indiv1,location+1:groupsize);%
latergroup(indiv1,location+1:groupsize)=latergroup(indiv2,location+1:groupsize);
        latergroup(indiv2,location+1:groupsize)=tmp;
    end
%选择出交叉后的最优个体，并与先前的最优个体比较
    cost=zeros(1,N);
    for i=1:N
        cost(i)=average_cost(A,latergroup(i,:));
    end
    [value,flag]=min(cost);
    if value>good_result
        latergroup(1,:)=best;
    end
    if value<=good_result
        good_result=value;
```

```
            best=latergroup(flag,:);
        end
        crossedgroup=latergroup;
    %每一个种群的每一个个体以 varPer 的概率变异
        for i=1:N%变异
            for j=2:groupsize-1
                tmp=unifrnd(0,1);%0~1 随机数
                if(tmp<=varPer&&crossedgroup(i,j)==1)
                    crossedgroup(i,j)=0;
                end
                if(tmp<=varPer&&crossedgroup(i,j)==0)
                    crossedgroup(i,j)=1;
                end
            end
        end
    %选择出变异后的最优个体，并与先前的最优个体比较
        cost=zeros(1,N);
        for i=1:N
            cost(i)=average_cost(A,crossedgroup(i,:));
        end
        [value,flag]=min(cost);
        if value>good_result
            crossedgroup(1,:)=best;
        end
         if value<=good_result
            good_result=value;
            best=crossedgroup(flag,:);
         end
        group=crossedgroup;
    %展示
        disp(['Generation',num2str(g),';']);
        disp([num2str(good_result),' ',num2str(best)]);
        end
    end
end
```

③退火算法

```
function f = anneal_new(  )
T0=100;
T_end=0.01;
data=xlsread('20150915dataform.csv');
x=data(1:2:800,1:51);
```

```
y=data(2:2:800,1:51);
array=randperm(51);
gene=sort(array(1:7));gene_min=gene;
cost_min=998;
while(T0>T_end)
    array=randperm(51);
    rest=setdiff(array,gene);
    rest_new=rest(randperm(44));
    point=randi([1,7]);
    gene_now=gene;
    gene_now(1,point)=rest_new(1,point);
    gene_now=sort(gene_now);
    M=zeros(400,51);
for i=1:400
    M(i,:)=interp1(x(i,gene_now),y(i,gene_now),x(i,:),'spline');

end
   Q=12;
errabs=abs(M-y);

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(400,1)*7;
cost=sum(si)/400;
if cost<cost_min
       cost_min=cost;
       gene_min=gene_now;
       gene=gene_min;
elseif rand<exp((cost_min-cost)/T0)
       gene=gene_now;
end
    T0=T0*0.999;
end
disp([num2str(gene_min),' ',num2str(cost_min)]);
```

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

x=[5.0:0.1:10.0];

y1=interp1(x_premea,y0_premea,x,'spline');
end
function y2=test_ur_answer()
my_answer=gene_new;
my_answer_n=size(my_answer,2);

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;


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_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;

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