统计推断在数模转换系统中的应用

组号：68　　姓名：包明天 学号：5130309619　姓名：陈琢 学号：5140309457

摘要：统计推断学是一类重要的数理统计方法，在各种自然科学和社会科学研究、工程技术等领域都有广泛的应用。在本课程中，我们使用概率论与数理统计中的部分知识以及数学建模在MATLAB中的应用，寻求某产品内部监测模块校准工序的优化方案。所需样本由老师提供，共计400组，每组样本观测点51个。
关键词：MATLAB，统计推断，数模转换，数学建模

ABSTRACT: Statistical inference learning is a kind of important mathematical statistics method, is widely applied in the fields of all kinds of natural science and social science research and engineering technology. In this course, we use the application of probability theory and mathematical statistics knowledge and part of MATLAB genetic algorithm in the optimization scheme for a product, the internal monitoring module calibration procedure. The required samplesare provided by the teacher, a total of 469 groups, each group sample observation point 51.
Key words:MATLAB，Statistical inference, Digital to analog conversion, Three times spline interpolatio

1 引言

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）

2 数据样本分析

本次课程由老师提供400组数据，每组共计51个数据点，首先通过origin拟合给出数可

以得到下图。

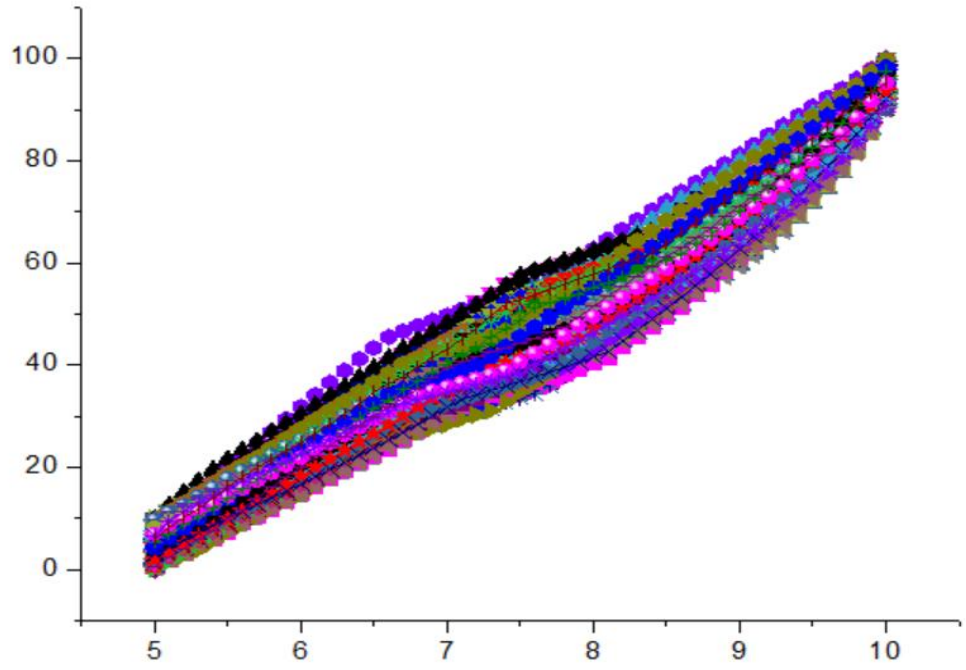

图1 获取数据拟合图像

通过拟合图像可以看出，图像范围较大，无法作为一个定标所用函数，所以我们需要通过数学模型拟合出一个尽量符合实际情况的曲线。

3.拟合方式的选择

3.1 成本计算

此函数所用的成本计算方式是老师给定的，所以可以直接使用。

$$s_{i,j}=\begin{cases} 0 & |y_0-y_1|\leqslant 0.4 \\ 0.1 & 0.4\leqslant |y_0-y_1|\leqslant 0.6 \\ 0.7 & 0.6\leqslant |y_0-y_1|\leqslant 0.8 \\ 0.9 & 0.8\leqslant |y_0-y_1|\leqslant 1.0 \\ 1.5 & 1.0\leqslant |y_0-y_1|\leqslant 2.0 \\ 6 & 2.0\leqslant |y_0-y_1|\leqslant 3.0 \\ 12 & 3.0\leqslant |y_0-y_1|\leqslant 5.0 \\ 25 & |y_0-y_1|>5 \end{cases}$$

其中y_0表示第i 个样本之第j 点Y 的实测值，y_1表示定标后的估计值，该点的相应误差成本

即为$s_{i,j}$。实施一次单点测定的成本以符号Q记。本课题指定Q=12。

某一个体的成本为 $S_i=\sum_{j=1}^{51} s_{i,j}+Qn_i$ ，n_i为单点测定次数。

总成本为$C=\frac{1}{M}\sum_{i=1}^{51} S_i$。

3.2 三次样条插值法

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，通过求解三弯矩方程组得出曲线函数组的过程。

3.3 多项式插值

在用pchip插值的过程中，matlab会基于所给的函数值来帮你估算各导数值。估算的原则是保证导数值能够正确的反映散点图的形状和变化趋势。比如在散点图是单调递增的区间内，相应点的导数值就会是正的；在散点图表现出存在局部极值点的区间，相应的导数也会产生正负的变化。

3.4 拟合方式的选择

在了解了两种插值方式的基本原理之后，我们在首先利用了三次样条插值法，但经过多次计算，发现几乎每次计算的最佳成本都在100以上，之后我们改为多项式插值，终于使成本降至100以下，所以最后我们选择的是多项式插值。

4.遗传算法

4.1遗传算法的介绍

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群(population)开始的,而一个种群则由经过基因(gene)编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加

适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

4.2 遗传算法的实现方法

4.2.1设计思路

把每个取点方案用51 位二进制数对应的一维数组表示。相应位数对应为1 则表示取出该位置的点，1 的个数即为取点的个数。适应度函数计算种群中每种基因型的适应度及生存概率,根据随机数模拟出存活的个体及其基因型（赌轮盘法）。为达到尽可能随机的目的，我们采取依据概率进行交叉互换的方法，我们将交叉概率设为0.8，根据生成概率数确定随机配对点从而得到新个体。之后再随机取若干个位置，令这这些位置的基因发生变异，得到最终的第二代个体可能的基因型。之后再重复相同的操作直至指定的循环代数，循环结束。最终获得拟合较好的取点方案。

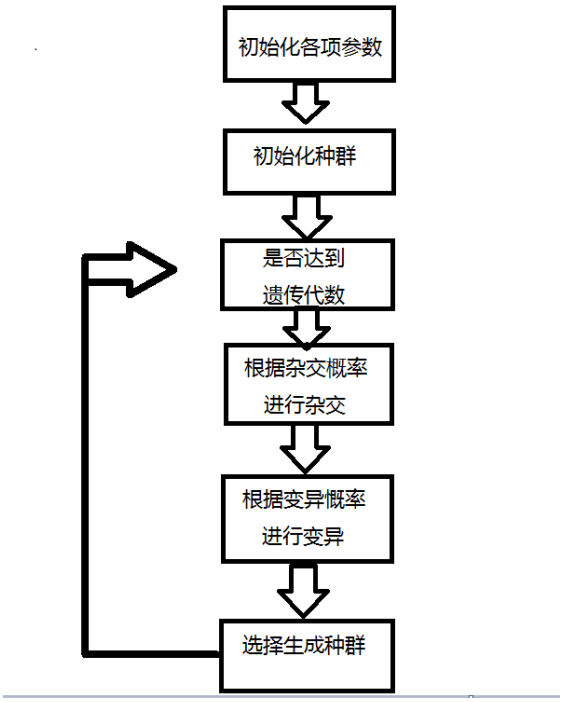

图2　遗传算法流程图

4.4.2种群初始化

我们通过matlab编程，实现了整个过程。过程包含了拟合计算成本过程fitnessfun，基因交叉过程crossfun，基因突变过程mutationfun，以及整体循环mainfun。

首先我们利用0和1来代表是否选取某一点，即0代表不选，1代表选，从而构成一个种群，之后我们将在此基础上进行运算。

4.4.3杂交

将传入的数据，利用之前设定的crossnum=0.8的交叉概率确定是否进行杂交，当随机数小于0.8时，则杂交，并由此确定杂交点，将每相隔一行的两组数据依据随机点进行杂交拼接。

在第一次面谈中，老师指出交叉概率其实是越高越好甚至直接取为1就可以，但我们为了尽量模拟一种随机的效果，同时为了让计算更贴近实际情况，我们最终将交叉概率定为0.8。

4.4.4变异

将杂交之后的数据，再根据设定的变异概率change=0.05确定是否变异，即若判断所用的随机数小于0.05，则将种群中该数由0变为1（选用该组数据）或由求1变为0（弃用该组数据）。但是1和51不能发生突变。

4.4.5选择

首先，我们计算每一个方案的适应度。即直接的取为拟合成本的倒数——因为本题需要寻找拟合成本最小的取点方案。则该方案成本越低，则适应度越高，该方案就在排序中就越靠前，同时为了让适应度高的方案更容易被选中，我们利用“轮盘赌”的方法进行随机选择，即适应度越高，该方案权重越高，被选中的几率就越大。通过选择过程，我们就可以生成一个新的种群。

4.4.5遗传终止

通过我们事先设定好的代数，来决定遗传算法的终止，此处我们设定的遗传代数为genen=100。

5.结果分析

在拟合方式上，我们首先选用了三次样条插值法进行拟合，但成本一直高于 100，之后改用插值法，使得最终成本低于 100，达到了 98.0872，而选出的最优个体为 1，10，22，31，44，51.

4.参考文献

[1] 袁焱.统计推断在数模转换系统中的应用课程讲义
[2]《数学建模算法与应用》.司守奎编著.国防工业出版社
[2]百度百科.遗传算法
[3]维基百科.遗传算法 -维基百科，自由的百科全书

附录文本：实验代码

1. test_ur_answer

```
my_answer=[ 1,10,22,31,44,51 ];.

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);
```

2. mycurvefitting

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

x=[5.0:0.1:10.0];

% ½«ÄãµÄ¶¨±ê¼ÆËã·½·¨Ð´ÉÖ¸Áî´úÂëÊ¹ÓÔÏÂÑùÊ½½ö¹©²Î¿¼
y1=interp1(x_premea,y0_premea,x,'pchip');


end
```

3. mainfun

```
function mainfun()
tic;
genen=100; %代数
n=30; %种群个体数
change=0.05; %变异概率
crossnum=0.8;%交叉概率
pop=zeros(n,51); %用以保存种群的矩阵
temp=zeros(n,51);
fitness=zeros(1,n); %适应度
for i=1:n %种群矩阵赋值
for j=2:50
if rand<0.1
pop(i,j)=1;
end
pop(i,1)=1;
pop(i,51)=1;
end
end
for q=1:genen
ave=0;    for i=1:n
```

```
if length(find(pop(i,:)))<=2
%当种群个数小于2时重新选择
fitness(i)=0;
else t=fitnessfun(pop(i,:));
fitness(i)=10/t;
ave=ave+t;
end
end
ave=ave/n;
[~,s]=max(fitness); %最优个体
fitness=fitness/sum(fitness); %轮盘赌
best=pop(s,:);%保存最优个体
ft=cumsum(fitness);
for i=1:n
t=find(rand<=ft);
temp(i,:)=pop(t(1),:);
end
temp(n,:)=best;%最后一行保存最优个体
pop=temp;
pop=crossfun(crossnum,pop);
pop=mutationfun(change,pop);
best_=zeros(1,1);
j=1;
for i=1:51
if best(i)==1
best_(j)=i;
j=j+1;
end
end
fprintf('代数:%d\n',q);
fprintf('最优个体¸öÌå: %d\n');
disp(best_);
fprintf('最佳成本:%d\n');
disp(fitnessfun(best));
fprintf('平均成本:%d\n');
disp(ave);
end
toc;
```

4. fitnessfun

```
function x=fitnessfun(a)
data=csvread('20150915dataform.csv');
```

```
x=zeros(1,51); %数据导入，x值
y=zeros(400,51); %数据导入，y值
Y=zeros(400,1);
fits=zeros(400,51); %拟合值
n=0;
b=zeros(1,1);
c=zeros(1,1);
for i=1:51
if a(i)==1
n=n+1;
b(1,n)=i;
end
end
for j=1:n
c(1,j)=data(1,b(1,j)); %取样点 x
end
x=data(1,:);
for i=1:400
y(i,:)=data(2*i,:);
for j=1:n
Y(i,j)=data(2*i,b(1,j));
end
end
for j=1:400
fits(j,:)=interp1(c(1,:),Y(j,:),x(1,:),'pchip'); %插值拟合
end
err=abs(fits-y); %计算成本
Q=12;
le0_4=(err<=0.4);
le0_6=(err<=0.6);
le0_8=(err<=0.8);
le1_0=(err<=1);
le2_0=(err<=2);
le3_0=(err<=3);
le5_0=(err<=5);
g5_0=(err>5);

ERR=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;
x=sum(sum(ERR))/400+12*n;
```

5. crossfun

```
function [ newpop ] = crossfun(crossnum,pop) %个体交叉
[px,py]=size(pop);
newpop=zeros(px,py);
a=randperm(px);
for i=1:2:(px-1)
x=a(i);y=a(i+1);
if(rand<crossnum) %小于crossnum时进行交叉
cpoint=ceil(rand*49); %生成随机交叉点
newpop(x,:)=[pop(x,1:cpoint),pop(y,cpoint+1:py)];
newpop(y,:)=[pop(y,1:cpoint),pop(x,cpoint+1:py)];
else
newpop(x,:)=pop(x,:);
newpop(y,:)=pop(y,:);
end
end
end
```

6.mutationfun

```
function [ newpop ]= mutationfun( pm,pop )
px=size(pop,1);
newpop=zeros(size(pop));
for i=1:px
newpop(i,:)=pop(i,:);
if(rand<pm)
mpoint=round(rand*50+1);
newpop(i,mpoint)=1-pop(i,mpoint);
end
end
end
```